Pannon Egyetem
University of Pannonia

Adatbányászat

a hatékonyság eszköze



Elmélet és gyakorlat Pythonban

2., teljesen átdolgozott kiadás

Abonyi János

Pannon Egyetem · Veszprém · 2026

nyílt hozzáférésű tankönyvsorozat

webkönyv · PDF · EPUB · Colab

complexsystemsintelligence.github.io/adatbanyaszat

Abonyi János
Adatbányászat – a hatékonyság eszköze
Elmélet és gyakorlat Pythonban
2., teljesen átdolgozott kiadás

Az első kiadás: Abonyi János (szerk.): *Adatbányászat – a hatékonyság eszköze.* ComputerBooks, Budapest, 2006.

© Abonyi János, 2026
Kiadó: Pannon Egyetem, Veszprém
Változat: 2.0.0
DOI: 10.5281/zenodo.24413676

A könyv szövege, ábrái és táblázatai a Creative Commons Nevezd meg! – Ne add el! 4.0 Nemzetközi (CC BY-NC 4.0) licenc alatt nem kereskedelmi célra szabadon felhasználhatók; kereskedelmi célú felhasználásukhoz a szerző előzetes engedélye szükséges. A programkód az MIT licenc alatt áll. A Pannon Egyetem neve, címere és logója a Pannon Egyetem arculati elemei, ezekre a fenti licencek nem terjednek ki.

Webkönyv, PDF és EPUB: https://complexsystemsintelligence.github.io/adatbanyaszat/
Forráskód, adatok és jegyzetfüzetek: https://github.com/ComplexSystemsIntelligence/adatbanyaszat

A második kiadás kutatási hátterét a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal OTKA 143482 számú projektje támogatta.

Szedés: Quarto és XeLaTeX; betűkészletek: IBM Plex és Raleway.



Pannon Egyetem
University of Pannonia

Tartalom

Előszó **1**
Kinek szól a könyv? . . . 1
A könyv felépítése . . . 1
A fejezetek szerkezete . . . 2
Interaktív anyagok . . . 3
Konvenciók . . . 3
A tankönyvsorozat . . . 3
Licenc és hivatkozás . . . 3
Köszönetnyilvánítás . . . 4

I. Alapok **5**

1. Adatból tudás: az adatbányászat folyamata **7**
1.1. Miért van szükség adatbányászatra? . . . 7
1.2. Alapfogalmak . . . 8
1.3. Adatbányászati feladattípusok . . . 10
1.4. Az adatbányászati projekt folyamata . . . 12
1.5. Felelős adatbányászat: adatvédelem, szabályozás, etika . . . 14
1.6. Eszközök: a Python adatelemzési ökoszisztémája . . . 16
1.7. A két futó esettanulmány . . . 17
1.8. Összefoglalás . . . 20
Ellenőrző kérdések . . . 21
Feladatok . . . 21
További olvasmányok . . . 22

2. Adattárházak és analitikus adatkezelés **23**
2.1. Operatív és analitikus adatkezelés . . . 23
2.2. Az adattárház . . . 24
2.3. A dimenzionális adatmodell . . . 26
2.4. Az adatkocka és az OLAP-műveletek . . . 28
2.5. Korszerű analitikus adatarchitektúrák . . . 30
2.6. Gyakorlat: csillagséma és OLAP Pythonban . . . 31
2.7. Összefoglalás . . . 38

Ellenőrző kérdések . . . 39
Feladatok . . . 39
További olvasmányok . . . 39

3. Adatok előkészítése 41
3.1. Adattípusok és mérési skálák . . . 41
3.2. Az adatminőség felmérése . . . 42
3.3. Hiányzó adatok . . . 44
3.4. Zaj és kiugró értékek . . . 47
3.5. Transzformációk . . . 50
3.6. Jellemzők kiválasztása . . . 54
3.7. Adatszivárgás és a szivárgásmentes folyamat . . . 56
3.8. Gyakorlat: a futó esetek megtisztítása . . . 58
3.9. Összefoglalás . . . 59
Ellenőrző kérdések . . . 59
Feladatok . . . 59
További olvasmányok . . . 60

4. Feltáró adatelemzés és dimenziócsökkentés 61
4.1. A feltáró adatelemzés szemlélete . . . 61
4.2. Egy változó eloszlásának jellemzése . . . 63
4.3. Több változó együttes vizsgálata . . . 68
4.4. Dimenziócsökkentés: főkomponens-elemzés . . . 70
4.5. Nemlineáris vetítések . . . 77
4.6. Összefoglalás . . . 81
Ellenőrző kérdések . . . 81
Feladatok . . . 82
További olvasmányok . . . 82

II. Mintázatok felügyelet nélkül 83

5. Csoportosítás 85
5.1. Mi a csoport? . . . 85
5.2. Hasonlóság és távolság . . . 86
5.3. Hierarchikus módszerek . . . 88
5.4. Partícionáló módszerek . . . 91
5.5. Fuzzy csoportosítás . . . 93
5.6. Modellalapú csoportosítás: Gauss-keverék modellek . . . 96
5.7. Sűrűségalapú csoportosítás: DBSCAN . . . 97
5.8. Önszerveződő térképek . . . 98
5.9. A csoportosítás értékelése . . . 99
5.10. Gyakorlat: vevőszegmentálás és üzemállapotok . . . 100
5.11. Összefoglalás . . . 107
Ellenőrző kérdések . . . 108

Feladatok . . . 109
További olvasmányok . . . 109

6. Gyakori mintázatok és asszociációs szabályok 111
6.1. Alapfogalmak . . . 111
6.2. Az Apriori-elv és az Apriori-algoritmus . . . 112
6.3. Mélységi és mintanövelő algoritmusok . . . 116
6.4. Tömör reprezentációk: zárt és maximális elemhalmazok . . . 117
6.5. Szabályok előállítása és értékelése . . . 117
6.6. Kiterjesztések . . . 119
6.7. Gyakorlat: kosárelemzés és riasztássorozatok . . . 120
6.8. Összefoglalás . . . 126
Ellenőrző kérdések . . . 126
Feladatok . . . 126
További olvasmányok . . . 127

III. Prediktív modellezés 129

7. A felügyelt tanulás alapjai és a modellek értékelése 131
7.1. Az általánosítás problémája . . . 131
7.2. Adatfelosztás és validáció . . . 133
7.3. Regressziós mérőszámok . . . 134
7.4. Osztályozási mérőszámok . . . 135
7.5. A modellek összehasonlítása . . . 137
7.6. Gyakorlat: lemorzsolódás és termékminőség . . . 138
7.7. Összefoglalás . . . 145
Ellenőrző kérdések . . . 145
Feladatok . . . 145
További olvasmányok . . . 146

8. Regresszió 147
8.1. Lineáris regresszió . . . 147
8.2. A becslés tulajdonságai és a modelldiagnosztika . . . 150
8.3. Regularizált regresszió . . . 153
8.4. Nemlineáris regresszió . . . 156
8.5. Gyakorlat: lágy szenzor és vevőérték-előrejelzés . . . 158
8.6. Összefoglalás . . . 162
Ellenőrző kérdések . . . 162
Feladatok . . . 163
További olvasmányok . . . 163

9. Osztályozás 165
9.1. Az osztályozási feladat . . . 165
9.2. Logisztikus regresszió . . . 166

9.3. Döntési fák . . . 166
9.4. Bayes-féle osztályozók . . . 170
9.5. A k-legközelebbi szomszéd módszer . . . 172
9.6. Támaszvektor-gépek . . . 172
9.7. Együttes modellek . . . 174
9.8. A modellek értelmezése . . . 175
9.9. Gyakorlat: lemorzsolódás, minőség és hibadiagnózis . . . 176
9.10. Összefoglalás . . . 185
Ellenőrző kérdések . . . 185
Feladatok . . . 186
További olvasmányok . . . 186

10. Neurális hálózatok és mélytanulás **187**
10.1. A mesterséges neuron . . . 187
10.2. Többrétegű hálózatok . . . 188
10.3. A hálózatok tanítása . . . 190
10.4. Mélytanulási architektúrák . . . 193
10.5. Gyakorlat: hálózatok NumPy-ban, `scikit-learn`-ben és PyTorch-ban 197
10.6. Összefoglalás . . . 208
Ellenőrző kérdések . . . 209
Feladatok . . . 209
További olvasmányok . . . 210

11. Anomáliák és eltérések észlelése **211**
11.1. Mi az anomália? . . . 211
11.2. Statisztikai módszerek . . . 212
11.3. Statisztikai folyamatszabályozás . . . 213
11.4. Távolság- és sűrűségalapú módszerek . . . 219
11.5. Izolációs erdő . . . 220
11.6. Egyosztályos SVM és rekonstrukciós módszerek . . . 220
11.7. Küszöbválasztás és értékelés . . . 221
11.8. Gyakorlat: csalásgyanús rendelések és üzemzavarok . . . 222
11.9. Összefoglalás . . . 231
Ellenőrző kérdések . . . 231
Feladatok . . . 232
További olvasmányok . . . 232

IV. Speciális adattípusok **233**

12. Idősorok elemzése **235**
12.1. Az idősorok szerkezete . . . 235
12.2. Előrejelzés . . . 237
12.3. Hasonlóság és reprezentáció . . . 239
12.4. Idősorok osztályozása és szegmentálása . . . 242

12.5.Gyakorlat: forgalom-előrejelzés, hűtési igény és jelalakok 243
12.6.Összefoglalás 254
Ellenőrző kérdések 255
Feladatok 255
További olvasmányok 256

13. Szövegbányászat **257**
13.1.A szöveg mint adat 257
13.2.A vektortér-modell 258
13.3.Szövegosztályozás 260
13.4.Szövegklaszterezés és témamodellek 261
13.5.Szóbeágyazások 263
13.6.Kulcsszavak és összefoglalás 263
13.7.Gyakorlat: vásárlói vélemények és műszaknapló 264
13.8.Összefoglalás 273
Ellenőrző kérdések 274
Feladatok 274
További olvasmányok 274

V. Alkalmazások **277**

14. Ügyfélkapcsolat-menedzsment és ügyfélanalitika **279**
14.1.A vevői életciklus és az analitika 279
14.2.RFM-elemzés és szegmentálás 280
14.3.Az ügyfélérték modellezése 281
14.4.Ajánlórendszerek 283
14.5.A kampányhatás mérése: A/B teszt és uplift-modellezés 284
14.6.Jogi és etikai keretek 285
14.7.Gyakorlat: a PannonShop vevőinek értéke és megtartása 286
14.8.Összefoglalás 298
Ellenőrző kérdések 298
Feladatok 298
További olvasmányok 299

15. Ipari folyamatanalitika **301**
15.1.Az ipari adatok sajátosságai 301
15.2.Lágy szenzorok 302
15.3.Több működési állapotú folyamatok felügyelete 304
15.4.Hibadiagnózis 304
15.5.Riasztáskezelés 304
15.6.Prediktív karbantartás 305
15.7.Gyakorlat: a Pannon üzem folyamatanalitikai rendszere 306
15.8.A modellek üzemeltetése 316
15.9.Összefoglalás 317

Ellenőrző kérdések . . . 317
Feladatok . . . 317
További olvasmányok . . . 318

Irodalomjegyzék **319**

Függelékek 335

A. Python-alapok **335**
A.1. A munkakörnyezet . . . 335
A.2. A nyelv alapelemei . . . 336
A.3. NumPy: tömbök és vektorizált számítás . . . 337
A.4. pandas: táblázatos adatok . . . 339
A.5. Matplotlib: ábrák . . . 340
A.6. scikit-learn: a gépi tanulás egységes felülete . . . 341
A.7. A könyv segédcsomagja . . . 342
A.8. Jó gyakorlatok . . . 342

B. Lineáris algebra az adatbányászathoz **345**
B.1. Vektorok . . . 345
B.2. Mátrixok és műveleteik . . . 346
B.3. Lineáris függetlenség, rang, inverz . . . 347
B.4. Legkisebb négyzetek és vetítés . . . 347
B.5. Sajátértékek és sajátvektorok . . . 348
B.6. Szinguláris érték szerinti felbontás . . . 349
B.7. Mátrixos deriválás . . . 350
B.8. Összefoglaló táblázat . . . 351

C. Valószínűségszámítási és statisztikai alapok **353**
C.1. Valószínűségi változók és eloszlások . . . 353
C.2. Feltételes valószínűség és a Bayes-tétel . . . 354
C.3. A többdimenziós normális eloszlás . . . 356
C.4. Paraméterbecslés . . . 356
C.5. Konfidenciaintervallum, hipotézisvizsgálat, bootstrap . . . 358

D. Az adatsorok leírása **361**
D.1. Miért szimulált adatok? . . . 361
D.2. PannonShop: online kiskereskedelem . . . 361
D.3. Pannon üzem: folyamatos vegyipari reaktor . . . 362
D.4. Kis adatsorok a kézi számolásokhoz . . . 363
D.5. Nyilvános valós adatsorok a további gyakorláshoz . . . 364

E. Jelölések **365**
Általános jelölések . . . 365

Valószínűség és statisztika 365
Modellek és tanulás 366
Idősorok 366
Mértékegységek és számírás 367

F. Magyar–angol szójegyzék **369**

Előszó

Az adatbányászat – az adatokban rejlő, korábban ismeretlen, hasznos és érthető összefüggések feltárása – az elmúlt két évtizedben a vállalatirányítás, a mérnöki gyakorlat és a tudományos kutatás mindennapi eszközévé vált. A könyv első kiadása 2006-ban jelent meg (Abonyi, 2006), amikor a módszerek még elsősorban speciális szoftvercsomagokban voltak elérhetők, és a „big data", a mélytanulás vagy az ipar 4.0 fogalma még nem létezett. Azóta a terület alapjaiban átalakult: a nyílt forrású Python-ökoszisztéma vált a közös nyelvvé, megjelentek az együttes modellek és a mély neurális hálózatok, és az adatvédelmi, etikai szempontok a módszertan részévé váltak. Ez a második kiadás ezért nem egyszerű frissítés: a könyvet teljesen átírtam, a régi fejezeteket korszerűsítettem és kibővítettem, és új fejezetekkel egészítettem ki – megőrizve az első kiadás gyakorlatias szemléletét és két alkalmazási szálát, az üzleti és a mérnöki adatbányászatot.

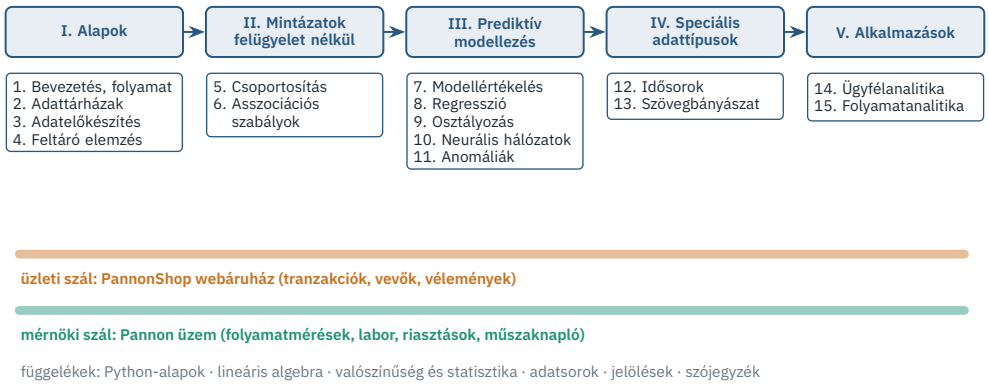
Kinek szól a könyv?

A könyv elsősorban mérnök-, gazdaságinformatikus- és adattudományi alapszakos (BSc) hallgatóknak készült, de haszonnal forgathatják a mesterszakos hallgatók és a gyakorló szakemberek is, akik az adatbányászat módszereit rendszerezett formában, a gyakorlati alkalmazáshoz szükséges mélységben szeretnék megismerni. Előismeretként középiskolai matematikát és az egyetemi analízis alapjait feltételezem; a szükséges lineáris algebrai, valószínűségszámítási és programozási ismereteket a függelékek részletesen összefoglalják.

A könyv felépítése

A tizenöt fejezet öt részre tagolódik (1. ábra). Az **I. rész** az alapokat teremti meg: az adatbányászati folyamatot, az adattárházakat, az adatok előkészítését és a feltáró elemzést. A **II. rész** a felügyelet nélküli mintázatkeresés két nagy területét, a csoportosítást és az asszociációs szabályokat tárgyalja. A **III. rész** a prediktív modellezésé: a modellek értékelésétől a regresszión, az osztályozáson és a neurális

hálózatokon át az anomáliák észleléséig. A **IV. rész** két speciális adattípussal, az idősorokkal és a szövegekkel foglalkozik, az **V. rész** pedig az ügyfélanalitika és az ipari folyamatanalitika egységes keretébe illeszti a korábbi módszereket.



1. ábra. A könyv felépítése és a két futó eset, amely a fejezeteken végighúzódik.

A fejezetek egymásra épülnek. A két **futó eset** – a PannonShop webáruház és a Pannon üzem – adatai minden fejezetben visszatérnek, és a korábbi fejezetekben elvégzett lépésekre (például az adattisztításra) építenek; így a módszerek nem elszigetelt példákon, hanem egy-egy összefüggő elemzési történet részeként jelennek meg. A futó esetek adatai szimuláltak, de valósághűek; előnyük, hogy a hátterigazság ismert, ezért minden módszer eredménye pontosan ellenőrizhető (D. függelék).

A fejezetek szerkezete

Minden fejezet egy rövid **útmutatóval** kezdődik, amely összefoglalja, mit és miért tárgyalunk, és mit tud majd az olvasó a fejezet végére. Az elméleti részeket **kézi számolási példák** kísérik, amelyekben egy kis adatsoron lépésről lépésre végigkövethetők az algoritmusok, és a fontosabb algoritmusok végén **összefoglaló keret** áll. A **gyakorlati részekben** a módszereket a futó esetekre alkalmazzuk; az üzleti és a mérnöki esettanulmányokat színes keretek jelölik. A fejezeteket összefoglalás, ellenőrző kérdések, önálló feladatok és további olvasmányok zárják.

A könyv minden ábrája, táblázata és számszerű eredménye a könyv fordításakor lefuttatott kódból származik, így a szöveg és az eredmények összhangja automatikusan biztosított. Ahol egy módszer egy adott feladaton nem teljesít jól, azt nem hallgatjuk el: ezek a „negatív” eredmények legalább annyira tanulságosak, mint a sikeresek.

Interaktív anyagok

A webes változatban a fejezetekhez **böngészőben futó alkalmazások** tartoznak, amelyekkel a módszerek viselkedése paraméterek változtatásával közvetlenül kipróbálható (például a k-közép algoritmus lépései, a PCA forgatása, egy döntési fa építése vagy egy ROC-görbe küszöbe). Minden fejezet kódja **Google Colab-jegyzetfüzetként** is elérhető, telepítés nélkül futtatható és módosítható. A könyv teljes forrása – a szöveg, a kód, az ábrák és az adatgenerátorok – nyilvános GitHub-tárolóban érhető el; a hibák bejelentését és a javítási javaslatokat ott várom.

Konvenciók

A szövegben a magyar helyesírás szerinti tizedesvesszőt használom, a programkódban és a kódkimenetekben a Python szabályai szerinti tizedespontot. A szakkifejezések angol megfelelőjét első előfordulásukkor zárójelben megadom, a teljes szójegyzék az F. függelékben található; a jelölések összefoglalása az E. függelékben. A kódrészletek a klasszikus, széles körben használt Python-csomagokra – NumPy, pandas, Matplotlib, SciPy, statsmodels, `scikit-learn` – épülnek; a mélytanulási példák a PyTorch keretrendszert használják.

A tankönyvsorozat

A könyv a Pannon Egyetem nyílt hozzáférésű tankönyvsorozatának része. A sorozat kötetei – köztük a *Rendszermérnökség* és a *Biztonságkritikus rendszerek* – egységes szerkezetet, arculatot és technikai megoldást követnek: a Quarto-alapú forrásból kereshető webkönyv, PDF és EPUB készül, a kiadásokat a Zenodo archiválja és DOI-azonosítóval látja el.

Licenc és hivatkozás

A könyv szövege és ábrái a Creative Commons Nevezd meg! – Ne add el! 4.0 Nemzetközi (CC BY-NC 4.0) licenc alatt nem kereskedelmi célra szabadon felhasználhatók; kereskedelmi célú felhasználásukhoz a szerző előzetes engedélye szükséges. A programkód az MIT licenc alatt áll. Hivatkozás:

> Abonyi János (2026): *Adatbányászat – a hatékonyság eszköze. Elmélet és gyakorlat Pythonban.* 2., teljesen átdolgozott kiadás. Pannon Egyetem, Veszprém. DOI: 10.5281/zenodo.24413676

A könyv kiadásait a Zenodo archiválja: a fenti DOI mindig a legfrissebb kiadásra mutat, a kiadásonkénti DOI-azonosítók a Zenodo-rekord oldalán és a tároló `CITATION.cff` fájljában találhatók.

Köszönetnyilvánítás

Az első kiadás a Veszprémi Egyetem Folyamatmérnöki Tanszékén, az IKTA 00142/2002 számú projekt és több OTKA-pályázat támogatásával készült. Köszönettel tartozom mindazoknak, akik az első kiadás megírásában és lektorálásában részt vettek, különösen Árva Péternek, Németh Sándornak, Vassányi Istvánnak, Feil Balázsnak, Balaskó Balázsnak, Pach Ferenc Péternek és Madár Jánosnak, valamint az esettanulmányokat és a lektori munkát vállaló gyakorló szakembereknek: Körmendi Györgynek, Cseh Zoltánnak, Halász Istvánnak, Robotka Zsoltnak, Rovnyai Jánosnak és Fekete Zoltánnak. Gondolataik és tapasztalataik ebben a kötetben is tovább élnek. A második kiadás kutatási hátterét a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal OTKA 143482 számú projektje támogatta.

Veszprém, 2026

Abonyi János

I. Alapok

1. Adatból tudás: az adatbányászat folyamata

Miről szól a fejezet?

Ez a fejezet kijelöli a könyv kereteit. Tisztázzuk, mit értünk adatbányászaton, milyen típusú kérdésekre ad választ, és milyen lépésekből áll egy adatbányászati projekt a célok megfogalmazásától a bevezetett modell karbantartásáig. Áttekintjük az adatok felhasználásának jogi és etikai kereteit, végül bemutatjuk a két futó esettanulmányt – a PannonShop webáruházat és a Pannon üzemet –, amelyekre a további fejezetek példái épülnek. **A fejezet végére az olvasó** képes lesz megkülönböztetni a felügyelt és a felügyelet nélküli tanulási feladatokat; egy gyakorlati problémát adatbányászati feladattípushoz rendelni; a CRISP-DM és a CRISP-ML(Q) fázisai mentén megtervezni egy elemzési projektet; és felismerni azokat a helyzeteket, amelyekben az adatkezelés adatvédelmi vagy MI-szabályozási kérdéseket vet fel.

1.1. Miért van szükség adatbányászatra?

A vállalatok, üzemek és intézmények működésük során folyamatosan adatokat rögzítenek: minden eladásról számla készül, minden szenzor percenként több mérést küld, minden ügyfélkapcsolat nyomot hagy egy informatikai rendszerben. Az adatok gyűjtése ma már olcsó és automatikus; a szűk keresztmetszet az, hogy a felhalmozott adatokból kinyerjük-e a döntésekhez szükséges *tudást*. Egy webáruház vezetője nem a számlatételek millióira kíváncsi, hanem arra, hogy mely vevői fognak valószínűleg elpártolni, és kiket érdemes megkeresni egy ajánlattal. Egy üzem technológusa nem a hőmérséklet-idősorokat akarja nézegetni, hanem időben szeretne értesülni arról, ha a folyamat a megszokottól eltérően kezd viselkedni.

Az adatbányászat azoknak a módszereknek és eljárásoknak a gyűjtőneve, amelyek ezt a lépést – az adatoktól a döntést segítő tudásig – rendszerezetten és ellenőrizhetően teszik meg. A terület a statisztika, a gépi tanulás, az adatbázis-kezelés

és a vizualizáció határán fejlődött ki, és mindegyiktől örökölt valamit: a statisztikától a bizonytalanság kezelését és a következtetés szigorát, a gépi tanulástól az automatikus modellépítést, az adatbázis-kezeléstől a nagy adatmennyiségek hatékony elérését, a vizualizációtól pedig azt a felismerést, hogy az eredményt végső soron embereknek kell megérteniük és elfogadniuk.

A könyv első kiadása (Abonyi, 2006) óta a terület eszköztára jelentősen kibővült. Általánossá váltak az együttes (ensemble) modellek és a mély neurális hálózatok, az elemzések túlnyomó része nyílt forráskódú Python-könyvtárakkal készül, és a modellek egyre gyakrabban kerülnek éles üzembe, ahol folyamatos felügyeletet igényelnek. Az alapkérdések azonban nem változtak: *milyen adatból, milyen módszerrel, milyen megbízhatósággal* juthatunk használható tudáshoz? Ez a könyv ezekre a kérdésekre keresi a választ.

1.2. Alapfogalmak

1.2.1. Adat, információ, tudás

A hétköznapi szóhasználatban az *adat*, az *információ* és a *tudás* gyakran összemosódik, az adatbányászatban azonban érdemes különbséget tenni közöttük.

- **Adat** a rögzített tény: „a 240679-es számlán 2 db kávébab szerepel 7990 Ft-os egységáron”. Önmagában kevés jelentést hordoz.
- **Információ** az adat feldolgozott, összefüggésbe helyezett formája: „decemberben a kávébab forgalma az éves átlag többszöröse”.
- **Tudás** olyan általánosítható összefüggés, amely cselekvést tesz lehetővé: „aki kávéfőző gépet vesz, nagy valószínűséggel kávébabot és vízkőoldót is vásárol – ezért érdemes a gép mellé ajánlani őket”.

Az adatbányászat ebben a láncban a középső és a felső szint közötti átmenetet támogatja: az adatokban rejlő, előre nem ismert, de érvényes és hasznos összefüggéseket tárja fel.

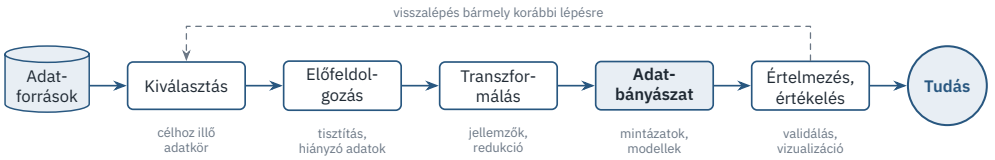
1.2.2. Mi az adatbányászat?

A tudásfeltárás (Knowledge Discovery in Databases, KDD) folyamatának máig legtöbbet idézett meghatározását Fayyad és mtsai. (1996) adta. Eszerint a tudásfeltárás az a nem triviális folyamat, amelynek során adatokban **érvényes**, **újszerű**, **potenciálisan hasznos** és **végső soron érthető** mintázatokat azonosítunk. A definíció mind a négy jelzője fontos:

- *érvényes*: a mintázat új adatokra is – bizonyos megbízhatósággal – teljesül, nem csupán a véletlen műve;

- *újszerű*: nem olyan összefüggés, amelyet a szakértők már ismernek (az, hogy a karácsonyi díszek decemberben fogynak, nem felfedezés);
- *hasznos*: valamilyen döntést vagy cselekvést támogat;
- *érthető*: a döntéshozó számára értelmezhető, ellenőrizhető formában jelenik meg.

A szakirodalom az *adatbányászat* kifejezést kétféle értelemben használja. Tágabb értelemben a teljes tudásfeltárási folyamat szinonimája, szűkebb értelemben e folyamat egyetlen lépése: a mintázatkereső algoritmusok alkalmazása (lásd a 1.1. ábrát). E könyv címében a tágabb, a fejezetekben – ahol a különbség számít – a szűkebb értelmet használjuk.



1.1. ábra. A tudásfeltárás (KDD) lépései Fayyad és mtsai. (1996) nyomán. Az adatbányászat szűkebb értelemben a folyamat egyetlen lépése, de a végeredmény minősége minden lépésen múlik, és a folyamat gyakran visszalép egy korábbi szakaszba.

1.2.3. Tanulás adatokból

Szinte minden adatbányászati módszer *induktív* tanuláson alapul: konkrét példákból általános összefüggésre következtet. Ennek formalizálásához bevezetjük a könyv alapjelöléseit (a teljes jelöléslistát az E. függelék tartalmazza).

Az elemzés tárgyai az **objektumok** (rekordok, megfigyelések, esetek), például vevők, számlák vagy 10 perces mérési időpontok. Minden objektumot n darab **jellemző** (attribútum, változó) ír le. Az N objektum adatai az **adatmátrixot** alkotják:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nn} \end{bmatrix} \in \mathbb{R}^{N \times n}, \qquad \mathbf{x}_k = [x_{k1}, x_{k2}, \ldots, x_{kn}]^\top. \quad (1.1)$$

Az adatmátrix k-adik **sora** a k-adik objektum, j-edik **oszlopa** a j-edik jellemző összes mért értéke. Ez a konvenció egyezik a `pandas` adattáblák és a `scikit-learn` könyvtár szokásával, így a képletek és a kódok közvetlenül megfeleltethetők egymásnak. (A szakirodalom egy része – köztük e könyv első kiadása is – helyenként

a transzponált, $n \times N$-es elrendezést használja; ilyenkor a képletekben a szorzások sorrendje megfordul.)

Felügyelt tanulásról beszélünk, ha minden objektumhoz ismert egy **célváltozó** (kimenet, címke) y_k értéke is, és olyan f függvényt keresünk, amely a jellemzőkből előrejelzi azt:

$$\hat{y} = f(\mathbf{x}; \boldsymbol{\theta}), \tag{1.2}$$

ahol $\boldsymbol{\theta}$ a modell **paramétereinek** vektora. Ha a célváltozó kategorikus (például „lemorzsolódik / marad”), **osztályozásról**, ha folytonos (például termékkoncentráció mol/L-ben), **regresszióról** beszélünk. A paramétereket úgy választjuk meg, hogy a modell a rendelkezésre álló tanítóadatokon a lehető legkisebb hibát kövesse el. Ha a hibát egy $L(y, \hat{y})$ **veszteségfüggvény** méri, a tanulás az *empirikus kockázat* minimalizálása:

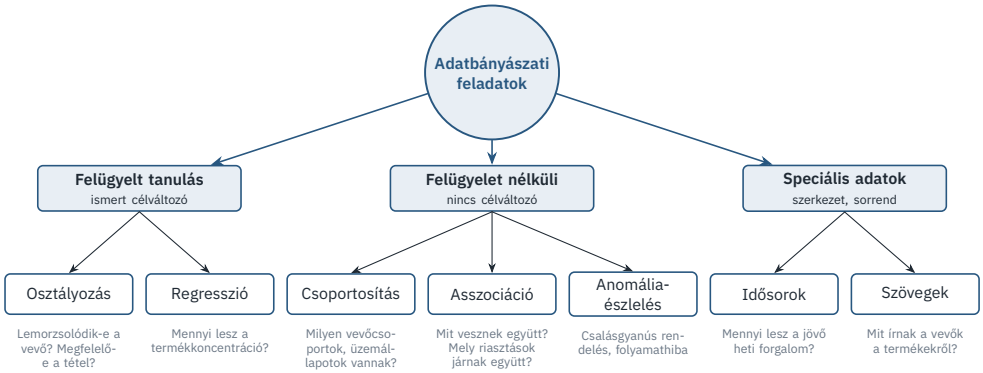
$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \; \frac{1}{N} \sum_{k=1}^{N} L\big(y_k,\; f(\mathbf{x}_k; \boldsymbol{\theta})\big). \tag{1.3}$$

A (1.3) képlet a könyv egyik legfontosabb egyenlete: a regressziós modellek, a logisztikus regresszió, a támaszvektor-gépek és a neurális hálózatok mind ennek speciális esetei, csak az f modellcsalád és az L veszteségfüggvény más. Látni fogjuk azt is, hogy a tanítóadatokon elért kis hiba még nem garancia a jó működésre: a valódi cél az **általánosítás**, vagyis hogy a modell *új*, a tanítás során nem látott adatokon is jól teljesítsen (7. fejezet).

Felügyelet nélküli tanulásban nincs célváltozó; az adatok belső szerkezetét keressük. Ilyen feladat a hasonló objektumok csoportokba rendezése (csoportosítás), a gyakran együtt előforduló elemek feltárása (asszociációs szabályok), az adatok tömörítése kevesebb dimenzióba (dimenziócsökkentés) vagy a többségtől eltérő objektumok felderítése (anomália-észlelés). A felügyelet nélküli eredmények értékelése nehezebb, hiszen nincs „helyes válasz”, amellyel összevethetnénk őket; ezért itt különösen fontos a szakterületi értelmezés.

1.3. Adatbányászati feladattípusok

A gyakorlati kérdések sokfélék, de néhány jól körülhatárolható feladattípusra vezethetők vissza (1.2. ábra). A feladattípus felismerése az elemzés első és talán legfontosabb lépése, mert ez határozza meg, milyen módszerek jöhetnek szóba, és hogyan kell az eredményt értékelni.



1.2. ábra. Az adatbányászati feladattípusok rendszere a könyv két futó esetének jellegzetes kérdéseivel. A felügyelt feladatokban ismert a célváltozó, a felügyelet nélküliekben az adatok belső szerkezetét keressük, a speciális adattípusok (idősorok, szövegek) saját reprezentációt igényelnek.

1.1. táblázat. Az adatbányászati feladattípusok áttekintése.

Feladattípus	Jellemző kérdés	Kimenet	Fejezet
Osztályozás	Melyik osztályba tartozik az objektum?	kategória, valószínűség	9.
Regresszió	Mekkora lesz egy mennyiség értéke?	szám	8.
Csoportosítás	Milyen természetes csoportok vannak?	csoportcímkék	5.
Asszociációs szabályok	Mi jár együtt mivel?	„ha A, akkor B” szabályok	6.
Anomália-észlelés	Mi tér el a megszokottól?	eltérési pontszám	11.
Dimenziócsökkentés	Hogyan tömöríthető az adat?	kevesebb új jellemző	4.
Idősor-előrejelzés	Hogyan alakul a jövőben?	jövőbeli értékek	12.
Szövegelemzés	Miről, milyen hangulatban szól a szöveg?	témák, címkék	13.

A 1.1. táblázat feladattípusai a gyakorlatban ritkán jelennek meg tisztán. Egy lemorzsolódás-előrejelző rendszer például csoportosítással kezdődhet (milyen vevőtípusok vannak?), osztályozással folytatódhat (ki fog elpártolni?), és asszociációs szabályokkal zárulhat (mit ajánljunk a megtartásukhoz?). A könyv utolsó két fejezete (14. és 15. fejezet) éppen ilyen összetett, több módszert összekapcsoló esettanulmányokat mutat be.

1.4. Az adatbányászati projekt folyamata

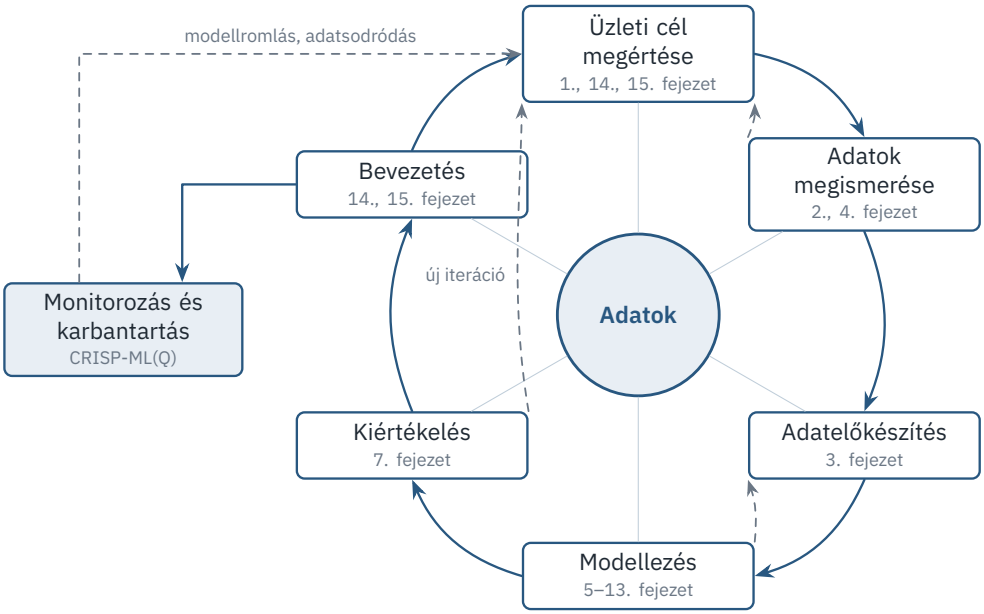
Az adatbányászat nem egyetlen algoritmus lefuttatása, hanem több lépésből álló, iteratív folyamat. A tapasztalat azt mutatja, hogy a projektek sikere sokkal inkább múlik a cél pontos megfogalmazásán, az adatok megismerésén és előkészítésén, valamint az eredmények üzleti értelmezésén, mint a modellezési algoritmus kiválasztásán.

1.4.1. A CRISP-DM folyamatmodell

Az ágazatfüggetlen CRISP-DM (Cross-Industry Standard Process for Data Mining) folyamatmodellt egy ipari konzorcium dolgozta ki (Chapman és mtsai., 2000; Wirth & Hipp, 2000). Az adatbányászati folyamatmodellek összehasonlító áttekintése szerint a CRISP-DM vált a terület *de facto* szabványává (Mariscal és mtsai., 2010). A modell hat fázist különböztet meg (1.3. ábra):

1. **Az üzleti cél megértése.** Mi a döntéshozó valódi problémája? Mi számít sikernek, és hogyan mérjük? Milyen erőforrások, korlátok és kockázatok vannak? A fázis végterméke az üzleti cél *adatbányászati* célként való megfogalmazása (például: „a következő negyedévben lemorzsolódó vevők azonosítása legalább 70%-os megbízhatósággal”).
2. **Az adatok megismerése.** Milyen adatok állnak rendelkezésre, honnan származnak, mennyire teljesek és megbízhatók? Az első feltáró elemzések (4. fejezet) gyakran már ebben a szakaszban módosítják a célokat.
3. **Adatelőkészítés.** Az elemzéshez szükséges adattábla előállítása: kiválasztás, tisztítás, összekapcsolás, transzformálás, új jellemzők képzése (3. fejezet). Ez a fázis rendszerint a projekt idejének jelentős részét felemészti.
4. **Modellezés.** A feladattípusnak megfelelő módszerek kiválasztása, a modellek felépítése és paramétereik hangolása (5–13. fejezet).
5. **Kiértékelés.** Megfelel-e a modell az *üzleti* céloknak? A statisztikai pontosság szükséges, de nem elégséges feltétel; azt is vizsgálni kell, hogy a modell használata megéri-e, és nincsenek-e nem kívánt mellékhatásai (7. fejezet).
6. **Bevezetés.** Az eredmények beépítése a mindennapi működésbe: egy jelentés, egy döntéstámogató felület vagy egy automatikusan futó, más rendszereket kiszolgáló modell formájában.

A fázisok sorrendje nem merev: a nyilak a leggyakoribb visszalépéseket jelzik. Az adatok megismerése során kiderülhet, hogy a cél az adott adatokból nem érhető el; a modellezés rámutathat az előkészítés hiányosságaira; a kiértékelés pedig gyakran új iterációt indít.



1.3. ábra. A CRISP-DM folyamatmodell hat fázisa (Chapman és mtsai., 2000) a könyv megfelelő fejezeteivel, kiegészítve a CRISP-ML(Q) monitorozási és karbantartási fázisával (Studer és mtsai., 2021). A szaggatott nyilak a gyakori visszalépéseket jelölik.

1.4.2. Kiterjesztés: a gépi tanulási modellek életciklusa

A CRISP-DM eredetileg egyszeri elemzési projektekre készült. Ma azonban a modellek gyakran hónapokig vagy évekig üzemelnek, miközben a környezetük változik: új termékek jelennek meg, a vevők szokásai módosulnak, egy üzemben kicserélnek egy szenzort. A Studer és mtsai. (2021) által javasolt **CRISP-ML(Q)** folyamatmodell ezért két lényeges ponton egészíti ki a CRISP-DM-et. Egyrészt minden fázishoz minőségbiztosítási lépéseket rendel (milyen kockázatok merülnek fel, és hogyan csökkenthetők), másrészt külön fázisként kezeli a bevezetett modell **monitorozását és karbantartását**.

Ennek oka, hogy egy éles üzemben futó modell teljesítménye idővel romolhat. A bemeneti adatok eloszlásának megváltozását **adatsodródásnak** (data drift), a bemenet és a célváltozó közötti kapcsolat megváltozását **koncepciósodródásnak** (concept drift) nevezzük (Gama és mtsai., 2014). Mindkettő ellen csak folyamatos méréssel és szükség esetén a modell újratanításával lehet védekezni. Az éles gépi tanulási rendszerekben a modell kódja általában csak kis része a teljes rendszernek: a körülötte lévő adatgyűjtő, ellenőrző és kiszolgáló infrastruktúra fenntartása legalább akkora feladat, mint maga a modellezés (Sculley és mtsai., 2015).

> Kulcsgondolat
>
> Egy adatbányászati projekt eredménye nem egy algoritmus, hanem egy *folyamat*, amely a döntéshozó kérdésétől a bevezetett és felügyelt megoldásig tart. A módszerek ismerete szükséges, de a siker leggyakrabban a cél pontos megfogalmazásán, az adatok minőségén és az eredmények értelmezésén múlik.

1.5. Felelős adatbányászat: adatvédelem, szabályozás, etika

Az adatbányászati eredmények emberek életére is hatással lehetnek: egy hitelbírálati modell döntést hoz arról, ki kap kölcsönt, egy toborzási modell előszűri a jelentkezőket. Ezért a módszertani kérdések mellett már a projekt elején tisztázni kell az adatkezelés jogi kereteit és az alkalmazás etikai kockázatait.

1.5.1. Személyes adatok és a GDPR

Az Európai Unióban a személyes adatok kezelését az általános adatvédelmi rendelet (GDPR) szabályozza (Európai Parlament és Tanács, 2016). Személyes adat

minden olyan információ, amely azonosított vagy azonosítható természetes személyre vonatkozik – egy vevőazonosító is az, ha más adatokkal összekapcsolva visszavezethető egy konkrét személyre. A rendelet 5. cikke rögzíti az adatkezelés alapelveit, amelyek közül az adatbányászatban különösen fontos:

- a **célhoz kötöttség**: az adatokat csak meghatározott, egyértelmű és jogszerű célból lehet gyűjteni, és ezzel össze nem egyeztethető módon nem szabad tovább kezelni;
- az **adattakarékosság**: csak a cél eléréséhez szükséges adatok kezelhetők;
- a **pontosság** és a **korlátozott tárolhatóság**;
- az **elszámoltathatóság**: az adatkezelőnek igazolnia kell a megfelelést.

A rendelet 22. cikke külön szabályozza a kizárólag automatizált adatkezelésen – köztük a profilalkotáson – alapuló, az érintettre joghatással járó vagy őt hasonlóan jelentős mértékben érintő döntéseket.

Az adatbányászati gyakorlatban két technikai fogalmat kell élesen megkülönböztetni. Az **álnevesítés** (pszeudonimizálás) során a közvetlen azonosítókat (név, e-mail-cím) helyettesítő kódra cseréljük, de a kapcsolat egy külön tárolt kulccsal visszaállítható; az ilyen adat a GDPR szerint továbbra is személyes adat. Az **anonimizálás** után a személy már semmilyen észszerűen valószínűsíthető eszközzel nem azonosítható. Ez nehezebb, mint elsőre látszik: a nem közvetlen azonosítók (például irányítószám, születési dátum, nem) kombinációja is egyedivé tehet egy rekordot. A k-*anonimitás* követelménye szerint minden rekordnak legalább $k - 1$ másik rekorddal meg kell egyeznie ezekben a kvázi-azonosítókban (Sweeney, 2002).

1.5.2. A mesterséges intelligenciáról szóló uniós rendelet

Az Európai Unió 2024-ben elfogadta a mesterséges intelligenciáról szóló rendeletet (AI Act) (Európai Parlament és Tanács, 2024). A rendelet kockázatalapú megközelítést követ: tiltja az elfogadhatatlan kockázatú gyakorlatokat, szigorú követelményeket támaszt a *magas kockázatú* rendszerekkel szemben (ilyenek például bizonyos foglalkoztatási, oktatási, illetve a természetes személyek hitelképességét értékelő alkalmazások), egyes rendszerekre átláthatósági kötelezettségeket ír elő, a többi alkalmazást pedig lényegében szabadon hagyja. A rendelet 2024. augusztus 1-jén lépett hatályba, a rendelkezései fokozatosan válnak alkalmazandóvá; a konkrét határidőket és az esetleges módosításokat mindig a hatályos joganyagban kell ellenőrizni.

A könyv módszerei önmagukban nem „magas kockázatúak” – a besorolás az *alkalmazási* területtől függ. Ugyanaz az osztályozó algoritmus lehet ártalmatlan, ha termékajánlást ad, és lehet magas kockázatú, ha munkavállalók kiválasztásában használják. Ezért a CRISP-DM első fázisában érdemes rögzíteni a tervezett alkalmazás jogi besorolását is.

1.5.3. Méltányosság és átláthatóság

A jogi megfelelésen túl az adatbányász felelőssége, hogy a modellje ne hátrányosítson meg csoportokat indokolatlanul. A torzítás forrása gyakran maga az adat: ha a múltbeli döntések igazságtalanok voltak, a belőlük tanult modell ezt újratermeli. A méltányosság mérésének és biztosításának módszereivel külön szakirodalom foglalkozik (Barocas és mtsai., 2023); a könyvben a modellek értelmezhetőségének eszközeit (9. fejezet) és a kiértékelés csoportonkénti bontását (7. fejezet) mutatjuk be, amelyek az ilyen problémák felismerésének első lépései.

Megjegyzés a könyv adatairól

A könyv mindkét futó esete *szintetikus*: az adatokat egy szimulátor állítja elő, ezért nem tartalmaznak valódi személyes adatot. A szimulátorok úgy készültek, hogy a valós adatok jellegzetes nehézségeit (hiányzó értékek, hibás rögzítés, szezonalitás, zaj) visszaadják, ugyanakkor a háttérben működő „igazság” ismert, így a módszerek eredménye pontosan ellenőrizhető.

1.6. Eszközök: a Python adatelemzési ökoszisztémája

Az első kiadás példái Excelre, a WEKA programcsomagra és MATLAB-programokra épültek. Ma az adatbányászati gyakorlatban – a kutatásban és az iparban egyaránt – a Python nyelv és a köré épült nyílt forráskódú könyvtárak dominálnak. A könyv következetesen a „klasszikus”, széles körben használt, jól dokumentált csomagokra támaszkodik:

1.2. táblázat. A könyvben használt fő Python-csomagok.

Csomag	Feladat a könyvben	Hivatkozás
NumPy	tömb- és mátrixműveletek, az algoritmusok saját implementációja	C. R. Harris és mtsai. (2020)
SciPy	optimalizálás, statisztika, interpoláció, távolságok	Virtanen és mtsai. (2020)
pandas	adattáblák, csoportosítás, idősorok	McKinney (2010)
Matplotlib, seaborn	ábrák	Hunter (2007); Waskom (2021)
scikit-learn	gépi tanulási modellek, előfeldolgozás, kiértékelés	Pedregosa és mtsai. (2011)
statsmodels	statisztikai modellek, idősor-modellek	Seabold & Perktold (2010)
mlxtend	gyakori elemhalmazok, asszociációs szabályok	Raschka (2018)
PyWavelets	wavelet-transzformáció	G. R. Lee és mtsai. (2019)
PyTorch	mély neurális hálózatok	Paszke és mtsai. (2019)

A Python alapjait és a fenti csomagok használatát az A. függelék foglalja össze, a módszerek mögötti lineáris algebrai és valószínűségszámítási fogalmakat pedig

a B. és a C. függelék.

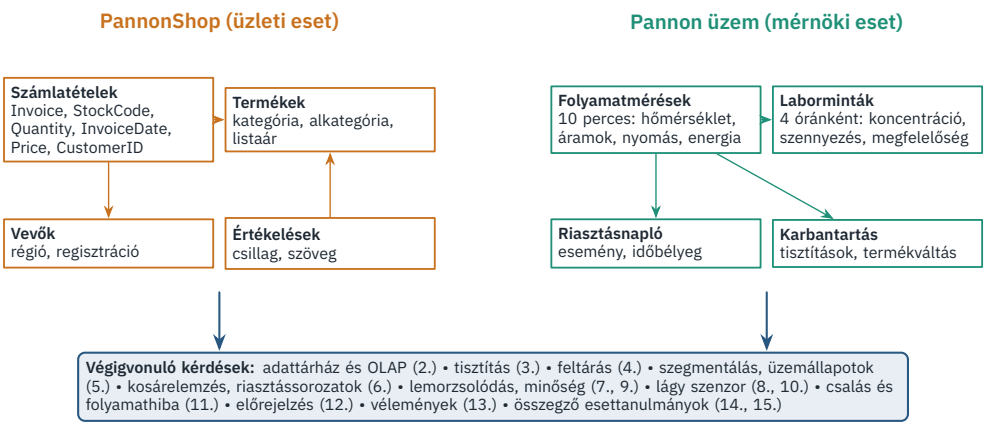
A könyvhöz egy kis segédcsomag is tartozik (`adatbanyaszat`), amely a futó esetek adatait állítja elő, és beállítja az ábrák egységes stílusát. Minden fejezet kódja így kezdődik:

```
import adatbanyaszat as ab
ab.set_style()           # a könyv ábrastílusa
shop = ab.load_shop()    # PannonShop – üzleti futó eset
plant = ab.load_plant()  # Pannon üzem – mérnöki futó eset
```

A kódok a könyv honlapjáról letölthető Colab-notebookokban telepítés nélkül, a böngészőben is futtathatók (lásd az Előszót).

1.7. A két futó esettanulmány

A módszereket a könyv két, végig ugyanazon adatokon dolgozó esettanulmányon mutatja be (1.4. ábra). Így az olvasó azt is látja, hogyan épülnek egymásra a fejezetek: amit a 3. fejezetben megtisztítunk, azt a 5. fejezetben csoportosítjuk, a 9. fejezetben pedig előrejelzésre használjuk.



1.4. ábra. A két futó esettanulmány adattáblái és a fejezeteken végigvonuló kérdések. Az üzleti eset tranzakciós adatai az Online Retail II adatsor sémáját követik, a mérnöki eset egy folyamatos üzemű reaktor méréseit szimulálja.

1.7.1. PannonShop: egy webáruház két éve

FUTÓ ESET · PANNONSHOP

A PannonShop kitalált magyar webáruház, amely kávét, teát, konyhai eszközöket, háztartási gépeket, dekorációs és ajándéktárgyakat, valamint írószereket árul. Az adatok 2024. január 1. és 2025. december 31. közötti számlatételeket tartalmaznak. A tábla szerkezete a nemzetközi gyakorlatban széles körben használt Online Retail II adatsorét követi (D. Chen és mtsai., 2012): minden sor egy számla egy tétele, a számlaszám (`Invoice`), a cikkszám (`StockCode`), a megnevezés (`Description`), a mennyiség (`Quantity`), az időpont (`InvoiceDate`), az egységár (`Price`), a vevőazonosító (`CustomerID`) és a régió (`Regio`) adataival.

```
shop = ab.load_shop()
tx = shop.transactions
print(tx.shape)
tx.head()
```

```
(68990, 8)
```

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	CustomerID	Regio
0	S240018	10032	Pléd	2	2024-01-01 07:21:00	9990.0	12003	Közép-Magyar
1	S240018	10040	Ajándékutalvány 10 000 Ft	1	2024-01-01 07:21:00	10000.0	12003	Közép-Magyar
2	S247721	10029	Teamécses 50 db	1	2024-01-01 08:27:00	1990.0	13489	Nyugat-Dunán
3	S247721	10031	Díszpárna	1	2024-01-01 08:27:00	5990.0	13489	Nyugat-Dunán
4	S242901	10003	Kávékapszula 30 db	1	2024-01-01 09:14:00	3290.0	12566	Közép-Dunánt

Az adatsor 68 990 számlatételt tartalmaz 15 919 számláról, 2 405 azonosított vevőtől, 48 termékre. Már az első sorok is mutatnak valamit abból, ami a valós adatokra jellemző: a postaköltség („POST”) nem termék, de számlatételként szerepel; a megnevezések között előfordulnak csupa nagybetűs, szóközzel lezárt változatok; a vendégvásárlásoknál hiányzik a vevőazonosító; a „C” betűvel kezdődő számlák pedig sztornók, negatív mennyiséggel. Ezekkel a 3. fejezetben foglalkozunk.

A szimulátor a háttérben öt vevőszegmenssel dolgozik (törzsvásárló, alkalmi vásárló, akcióvadász, viszonteladó, lemorzsolódó), amelyek eltérő gyakorisággal, eltérő termékkörből és eltérő mennyiségben vásárolnak, és eltérő ütemben pártolnak el. A szegmensek a modellek számára rejtettek – a 5. fejezetben azt vizsgáljuk, mennyire lehet őket a vásárlási szokásokból visszafejteni. A kosarakba a szimulátor néhány valószerű vásárlási szabályt is „beültetett” (például a kávéfőző géphez gyakran kávébab társul), amelyeket a 6. fejezet algoritmusainak kell megtalálniuk.

1.7.2. Pannon üzem: egy reaktor egy éve

FUTÓ ESET · PANNON ÜZEM

A Pannon üzem egy folyamatos üzemű, kevert tankreaktor, amelyben exoterm reakcióval három termékminőséget (A, B, C) gyártanak néhány napos kampányokban. A reaktor hőmérsékletét hűtőköpeny tartja az alapjelen; a hőcserélő felületén azonban lerakódás képződik, ezért a hűtési teljesítmény két tisztítás között fokozatosan romlik. A 10 perces mérések 2025 teljes évét lefedik; a termékkoncentrációt és a szennyezőtartalmat 4 óránként laboratóriumi minta méri.

```
plant = ab.load_plant()
m = plant.measurements
print(m.shape)
m.iloc[:, :8].head()
```

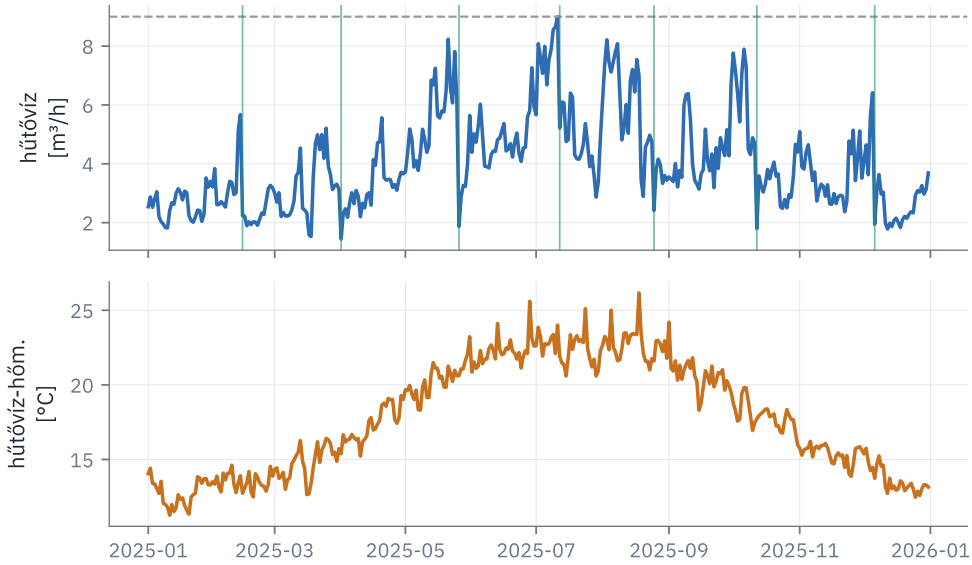
```
(52560, 14)
```

	ido	termek	T_kornyezet	F_betap	c_betap	T_betap	T_reaktor	T_alapjel
0	2025-01-01 00:00:00	B	-0.02	9.722	2.1051	73.98	86.07	86.0
1	2025-01-01 00:10:00	B	-0.11	9.670	2.1150	73.89	85.85	86.0
2	2025-01-01 00:20:00	B	-0.58	9.656	2.1131	74.14	86.09	86.0
3	2025-01-01 00:30:00	B	-0.44	9.711	2.1115	74.20	86.21	86.0
4	2025-01-01 00:40:00	B	-0.44	9.587	2.1180	74.01	86.05	86.0

A szimulátor egyszerűsített, de fizikailag konzisztens modellen alapul: a konverzió a hőmérséklettől és a tartózkodási időtől függ (Arrhenius-kinetika), a szükséges hűtővízáram a hőmérlegből adódik, nyáron pedig a melegebb hűtővíz miatt a lerakódott hőcserélő a teljesítőképessége határára kerülhet. Az év során ismert időpontokban különböző hibák lépnek fel: hőmérséklet-érzékelő elcsúszása (szenzor-drift), a betáplált nyersanyag koncentrációjának visszaesése, a hűtővízszelep akadozása és a hűtővíz átmeneti felmelegedése. Ezek felismerése a 11. és a 15. fejezet feladata.

```
napi = m.set_index("ido")[["F_hutoviz", "T_hutoviz"]].resample("D").mean()
fig, ax = plt.subplots(2, 1, sharex=True)
ax[0].plot(napi.index, napi.F_hutoviz, color=ab.PALETTE[0])
ax[0].axhline(9.0, color="0.6", lw=1, ls="--")
ax[0].set_ylabel("hűtővíz\n[m³/h]")
for d in plant.truth["cleanings"]:
    ax[0].axvline(d, color=ab.PALETTE[3], lw=0.8, alpha=0.6)
ax[1].plot(napi.index, napi.T_hutoviz, color=ab.PALETTE[1])
ax[1].set_ylabel("hűtővíz-hőm.\n[°C]")
```

```
fig.tight_layout()
```



1.5. ábra. A Pannon üzem egy évének áttekintése napi átlagokkal: a hűtővízáram a lerakódás miatt két tisztítás között fokozatosan nő, nyáron a melegebb hűtővíz miatt a szelep a határára kerül (szürke vonal: a szelep maximuma).

A 1.5. ábrán a függőleges zöld vonalak a tisztításokat jelölik. Már ez az egyszerű ábra is megmutatja az adatbányászat egyik alapvető tanulságát: a mért jelek mögött több, egymásra rakódó jelenség (évszakos változás, lerakódás, karbantartás, hibák) áll, amelyeket az elemzés során szét kell választanunk.

1.8. Összefoglalás

- Az adatbányászat az adatokból a döntést segítő tudásig vezető folyamat; a feltárt mintázatoknak érvényesnek, újszerűnek, hasznosnak és érthetőnek kell lenniük.
- A felügyelt tanulás ismert célváltozó előrejelzésére épít (osztályozás, regresszió), a felügyelet nélküli tanulás az adatok belső szerkezetét keresi (csoportosítás, asszociáció, anomáliák, dimenziócsökkentés).
- A felügyelt tanulás a veszteségfüggvény átlagának minimalizálásaként formalizálható (1.3), de a cél az új adatokra való általánosítás.

- A CRISP-DM hat fázisa (üzleti megértés, adatmegismerés, előkészítés, modellezés, kiértékelés, bevezetés) ma is az adatbányászati projektek vezérfonala; a CRISP-ML(Q) minőségbiztosítással és a bevezetett modellek monitorozásával egészíti ki.
- A személyes adatok kezelése a GDPR, az MI-alkalmazásoké az uniós MI-rendelet hatálya alá eshet; a besorolás az alkalmazási területtől függ.

Ellenőrző kérdések

1. Miben különbözik az adat, az információ és a tudás? Mondjon mindháromra egy-egy példát a Pannon üzem világából!
2. Melyik feladattípusba sorolná az alábbi kérdéseket? (a) Melyik vevőnk nyitja meg valószínűleg a hírlevelet? (b) Mennyi villamos energiát fogyaszt az üzem jövő héten?

   (c) Vannak-e olyan termékek, amelyeket rendszerint együtt vásárolnak? (d) Szokatlan-e a reaktor mostani viselkedése?
3. Miért nem elegendő, ha egy modell a tanítóadatokon kis hibát ér el?
4. Melyik CRISP-DM-fázisban dől el, hogy egy projekt sikeresnek tekinthető-e? Miért lehet szükség egy már bevezetett modell újratanítására?
5. Mi a különbség az álnevesítés és az anonimizálás között? Melyik esetben marad az adat személyes adat?

Feladatok

1. **Probléma megfogalmazása.** Válasszon egy Önhöz közel álló szervezetet (egyetem, munkahely, sportegyesület)! Fogalmazzon meg egy üzleti kérdést, majd alakítsa át adatbányászati céllá a CRISP-DM első fázisának megfelelően: határozza meg a célváltozót, a sikerkritériumot és a szükséges adatokat.
2. **Az adatok első megtekintése.** Töltse be a PannonShop adatait, és határozza meg:

   (a) hány különböző számla és vevő szerepel; (b) mekkora a sztornózott számlák aránya; (c) melyik hónapban volt a legtöbb rendelés!
3. **Az üzem egy hete.** Ábrázolja a Pannon üzem reaktorhőmérsékletét és hűtővízáramát egy tetszőleges hétre! Mit tapasztal a termékváltások idején?
4. **Kockázatelemzés.** A PannonShop vezetése automatikus hitelkeret-megállapítást tervez a viszonteladói számára a vásárlási előzmények alapján. Milyen adatvédelmi és méltányossági kérdéseket kell tisztázni a projekt indítása előtt?

További olvasmányok

A tudásfeltárás fogalmi kereteit Fayyad és mtsai. (1996) klasszikus cikke foglalja össze. A terület legátfogóbb tankönyvei J. Han és mtsai. (2022) és Tan és mtsai. (2019); a statisztikai tanulás elméleti alapjait Hastie és mtsai. (2009), bevezető szinten és Python-példákkal James és mtsai. (2023) tárgyalja. A CRISP-DM leírása Chapman és mtsai. (2000), a gépi tanulási modellek életciklusára kiterjesztett változata Studer és mtsai. (2021) munkájában olvasható.

2. Adattárházak és analitikus adatkezelés

Miről szól a fejezet?

Az adatbányászathoz megbízható, egységes és könnyen elérhető adatok kellenek. A vállalati adatok azonban sokféle, a napi működést kiszolgáló rendszerben szétszórva keletkeznek. Ez a fejezet azt mutatja be, hogyan szervezhetők ezek az adatok elemzésre alkalmas formába: mi az adattárház, hogyan épül fel a dimenzionális adatmodell (csillag-, hópehely- és galaxisséma), milyen műveletek végezhetők a többdimenziós adatkockán, és hogyan alakította át a területet az oszlopos tárolás, az adattó és a lakehouse-architektúra. A gyakorlati részben mindkét futó eset adataiból csillagsémát építünk, és OLAP-jellegű lekérdezéseket futtatunk.

A fejezet végére az olvasó képes lesz megkülönböztetni az operatív és az analitikus adatkezelést; egy üzleti folyamathoz tény- és dimenziótáblákat tervezni; az OLAP-műveleteket SQL-ben és pandasban végrehajtani; és indokolni egy analitikus adatarchitektúra megválasztását.

2.1. Operatív és analitikus adatkezelés

Egy webáruház adatbázisa másodpercenként sok kis tranzakciót szolgál ki: rendelést rögzít, készletet csökkent, szállítási címet módosít. Az ilyen **operatív**, más néven OLTP- (On-Line Transaction Processing) rendszerek tervezésének célja, hogy sok párhuzamos felhasználó gyorsan és ellentmondásmentesen tudjon kevés rekordot olvasni és írni. Ehhez a relációs adatmodell normalizált sémái illenek (Codd, 1970): minden tény egyetlen helyen szerepel, így a módosítás nem okoz ellentmondást.

Az elemző kérdései egészen mások. „Hogyan alakult az elmúlt két évben a kávékategória negyedéves árbevétele régiónként?” – egy ilyen lekérdezés sok százezer rekordot olvas, de egyetlen rekordot sem módosít, és sok tábla összekapcsolását igényli. Ha az elemző közvetlenül az operatív adatbázison dolgozna, lassítaná az ügyfélkiszolgálást, ráadásul az adatai hiányosak lennének: az operatív rendszerek gyakran csak az aktuális állapotot tárolják, a múltat felülírják. Az **analitikus**,

más néven OLAP- (On-Line Analytical Processing) rendszerek ezért külön adattárolóban, elemzésre optimalizált szerkezetben tárolják az adatokat (Codd és mtsai., 1993).

2.1. táblázat. Az operatív és az analitikus adatkezelés összehasonlítása.

Szempont	Operatív (OLTP)	Analitikus (OLAP)
Felhasználó	ügyintéző, alkalmazás	elemző, döntéshozó
Művelet	sok rövid olvasás és írás	kevés, sok rekordot olvasó lekérdezés
Adatok	aktuális, részletes	történeti, összesített is
Séma	normalizált	dimenzionális (csillag)
Optimalizálási cél	tranzakciós sebesség, konzisztencia	lekérdezési sebesség, áttekinthetőség
Tipikus kérdés	„Mi van a 240679-es rendelésben?”	„Hogyan alakul a forgalom régiónként?”

2.2. Az adattárház

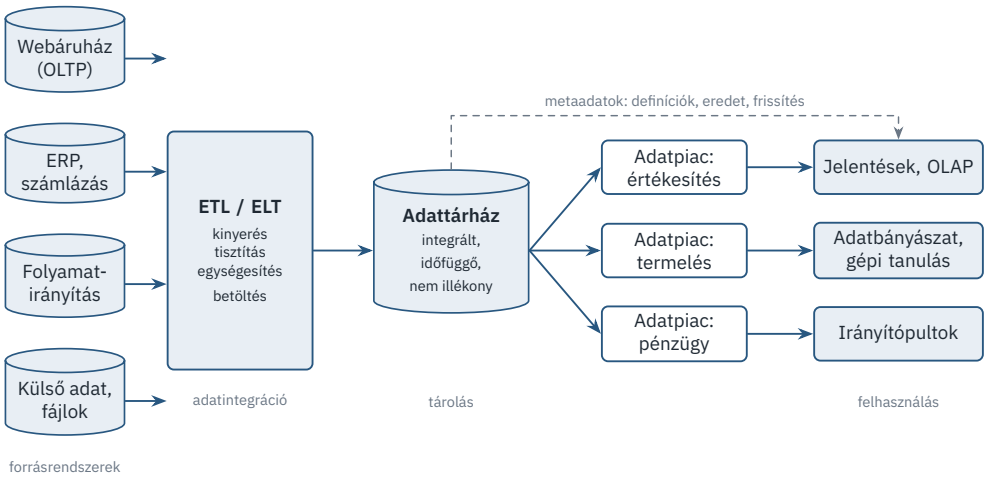
2.2.1. Definíció és architektúra

Az adattárház fogalmát Inmon (2005) négy tulajdonsággal írja le. Az adattárház

- **tárgyorientált**: a szervezet fő témái (vevő, termék, értékesítés, termelés) köré szerveződik, nem az alkalmazások köré;
- **integrált**: a különböző forrásrendszerek eltérő kódolásait, mértékegységeit és elnevezéseit egységesíti;
- **időfüggő**: minden adat egy időponthoz vagy időszakhoz kötődik, a múlt nem íródik felül;
- **nem illékony**: a betöltött adatokat nem módosítjuk, csak újabbakkal bővítjük.

Az adattárház tipikus felépítését a 2.1. ábra mutatja. A forrásrendszerekből az adatokat egy integrációs folyamat (ETL) gyűjti össze, tisztítja és tölti be az adattárházba. Az adattárházból témaspecifikus **adatpiacok** (data mart) szolgálják ki az egyes felhasználói köröket, amelyek jelentéseket, OLAP-elemzéseket, irányítópultokat és adatbányászati modelleket építenek rájuk. Az egészet **metaadatok** kísérik: mit jelent egy mező, honnan származik, mikor frissült.

Az adattárház-építésnek két klasszikus iskolája van. Inmon (2005) felülről lefelé haladó (*top-down*) megközelítésében először egy normalizált, vállalati szintű adattárház készül, és az adatpiacok ebből származnak. Kimball & Ross (2013) alulról felfelé építkező (*bottom-up*) megközelítésében az adatpiacok dimenzionális modellként egymás után készülnek el, és az egységességet a közös, úgynevezett **megfeleltetett dimenziók** (conformed dimensions) biztosítják. A gyakorlatban gyakran a kettő keveréke valósul meg; az elemző szempontjából a lényeg



2.1. ábra. Az adattárház tipikus architektúrája. Az integrációs réteg egységesíti a forrásrendszerek adatait, az adatpiacok egy-egy felhasználói kör igényeit szolgálják ki, a metaadatok pedig dokumentálják a teljes adatútvonalat.

mindkét esetben a dimenzionális modell, amelyet a következő alfejezetben tárgyalunk.

2.2.2. Adatintegráció: ETL és ELT

Az **ETL** (Extract–Transform–Load) a forrásadatok kinyerését, átalakítását és betöltését jelenti (Kimball & Caserta, 2004). Az átalakítás magában foglalja a tisztítást (hibás és hiányzó értékek kezelése, 3. fejezet), az egységesítést (azonos kódolás, mértékegység, dátumformátum), a duplikátumok kiszűrését, a helyettesítő kulcsok kiosztását és az összesítéseket. A modern, felhőalapú rendszerekben gyakori az **ELT** változat: a nyers adat először változatlanul bekerül a központi tárolóba, és az átalakítás ott, SQL-lekérdezésekkel történik. Így a nyers adat megmarad, és az átalakítási szabályok később is módosíthatók.

2.2.3. A változó dimenziók kezelése

A dimenziók tartalma idővel változik: egy vevő elköltözik, egy terméket átsorolnak másik kategóriába. A kérdés az, hogy a múltbeli tények a régi vagy az új dimenzióértékhez tartozzanak-e. Az ilyen **lassan változó dimenziók** (slowly changing dimensions, SCD) kezelésére Kimball & Ross (2013) több típust ír le, közülük a három leggyakoribb:

- **1. típus – felülírás**: a régi értéket felülírjuk, a múlt „átíródik”. Egyszerű, de a korábbi állapot elvész.
- **2. típus – új sor**: a változáskor új dimenziósor keletkezik új helyettesítő kulccsal, és az *érvényes-tól* / *érvényes-ig* mezők jelzik az érvényességi időszakot. A régi tények a régi sorra, az újak az újra mutatnak; a múlt hűen megmarad.
- **3. típus – új oszlop**: a korábbi értéket egy külön oszlopban őrizzük meg; csak egy korábbi állapotot tud követni.

Adatbányászati szempontból a 2. típus a legfontosabb: csak így biztosítható, hogy egy modell tanításakor minden tény azokkal a jellemzőkkel szerepeljen, amelyek *akkor* érvényesek voltak. Ennek elmulasztása a jövőbeli információ „beszivárgásához” vezethet (3. fejezet).

2.3. A dimenzionális adatmodell

2.3.1. Tények és dimenziók

A dimenzionális modell (Kimball & Ross, 2013) két táblatípusra épül. A **ténytábla** egy üzleti folyamat mérhető eseményeit tárolja (például egy számlatételt vagy egy órás termelési adatot), és két oszloptípust tartalmaz: a dimenziókra mutató **idegen kulcsokat** és a **mértékeket** (mennyiség, árbevétel, energiafogyasztás). A **dimenziótáblák** a tények leíró környezetét adják: *mikor* (dátum), *mit* (termék), *ki* (vevő), *hol* (régió) történt.

A tervezés első és legfontosabb döntése a **szemcsézettség** (granularitás) meghatározása: mit képvisel a ténytábla egy sora? A PannonShop esetében a legfinomabb szemcsézettség a számlatétel; a Pannon üzem esetében az órás átlag. A finomabb szemcsézettség rugalmasabb (bármilyen összesítés előállítható belőle), de nagyobb tárolási igénnyel jár. Az ökölszabály: a lehető legfinomabb, még értelmes szemcsézettséget tároljuk, mert összesíteni mindig lehet, részletezni utólag nem.

A mértékek összesíthetőségük szerint háromfélék:

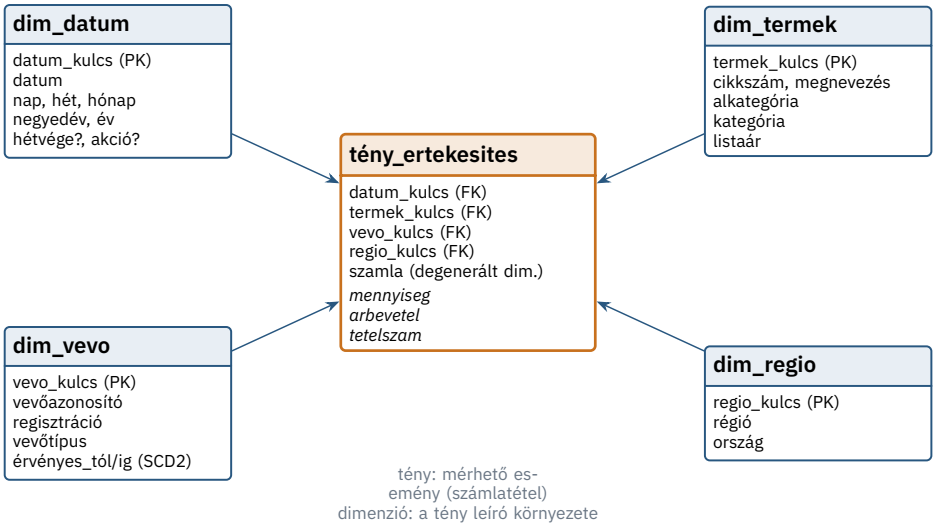
- **additív** mértékek minden dimenzió mentén összeadhatók (árbevétel, mennyiség);
- **félig additív** mértékek csak egyes dimenziók mentén adhatók össze (egy készletszint régiók mentén összegezhető, de időben nem – időben átlagolni kell);
- **nem additív** mértékek egyáltalán nem adhatók össze (arányok, átlagárak, hőmérsékletek); ezeket az additív összetevőikből kell újraszámolni.

Buktató: arányok összesítése

Az átlagár nem additív. Ha az egyik régióban 100 tétel átlagára 2000 Ft, a másikban 10 tételé 8000 Ft, a két régió együttes átlagára nem $(2000 + 8000)/2 = 5000$ Ft, hanem $(100 \cdot 2000 + 10 \cdot 8000)/110 \approx 2545$ Ft. Arányokat és átlagokat mindig a számlálójuk és nevezőjük külön összesítésével kell újraszámolni.

2.3.2. Csillagséma, hópehelyséma, galaxisséma

A **csillagsémában** a ténytábla a középen áll, és minden dimenziótáblához közvetlenül kapcsolódik (2.2. ábra). A dimenziótáblák szándékosan *nem normalizáltak*: a termékdimenzió például egyetlen táblában tartalmazza a terméket, az alkategóriát és a kategóriát, ezért egy lekérdezéshez kevés összekapcsolás kell. Az ismétlődő kategórianevek miatti redundancia a dimenziótáblák kis mérete miatt nem jelentős.



2.2. ábra. A PannonShop értékesítési csillagsémája. A ténytábla mértékeit (dőlt betű) a négy dimenzió írja le; a számlaszám ún. degenerált dimenzió, amelynek nincs külön táblája.

A **hópehelysémában** a dimenziókat normalizáljuk: a termékdimenzióból például külön kategóriatábla válik le. Ez csökkenti a redundanciát, de több összekapcsolást igényel, ezért az elemzési célú rendszerekben ritkábban választják. A **galaxisséma** (tényegyüttes, fact constellation) több ténytáblát tartalmaz, amelyek

közös dimenziókon osztoznak; a PannonShop esetében például az értékesítési és a vevőértékelési ténytábla közösen használhatja a dátum-, termék- és vevődimenziót.

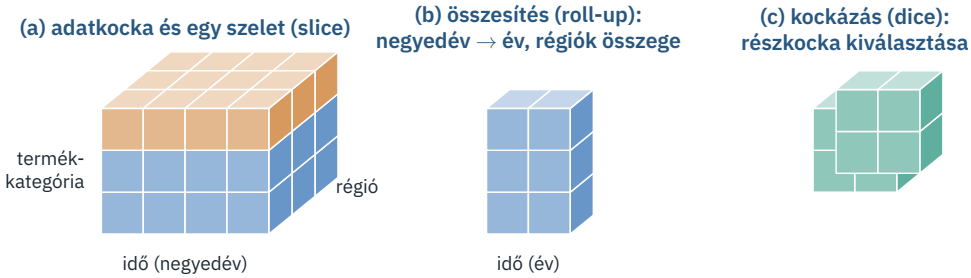
2.3.3. A dátumdimenzió

Szinte minden ténytábla kapcsolódik a dátumdimenzióhoz, amely az időbeli összesítések alapja. Ahelyett, hogy a lekérdezésekben minden alkalommal kiszámolnánk, melyik héthez vagy negyedévhez tartozik egy dátum, a dátumdimenzió soronként tárolja ezeket az attribútumokat – a munkanapokat, ünnepnapokat, akciós heteket is. Ez az egyszerű tábla az adatbányászatban is hasznos: a 12. fejezet idősor-modelljei éppen ilyen naptári jellemzőkből építkeznek.

2.4. Az adatkocka és az OLAP-műveletek

2.4.1. Az adatkocka

A dimenzionális modell természetes szemléltetése az **adatkocka**: d dimenzió esetén egy d dimenziós tömb, amelynek celláiban a mértékek összesített értékei állnak (2.3. ábra). A „kocka” elnevezés három dimenzióra utal, de az elv tetszőleges számú dimenzióra érvényes.



2.3. ábra. Az adatkocka és az OLAP-műveletek. (a) Az idő, termékkategória és régió által kifeszített kocka egy szelete (kiemelve) egy kategória összes cellája. (b) Az összesítés egy dimenzió hierarchiájában felfelé lép, vagy egy dimenziót teljesen összegez. (c) A kockázás több dimenzió mentén egyszerre szűkít.

Az alapvető OLAP-műveletek a következők:

- **Szeletelés** (slice): egy dimenzió értékét rögzítjük (például csak a „Kávé és tea” kategória), az eredmény eggyel kevesebb dimenziós.

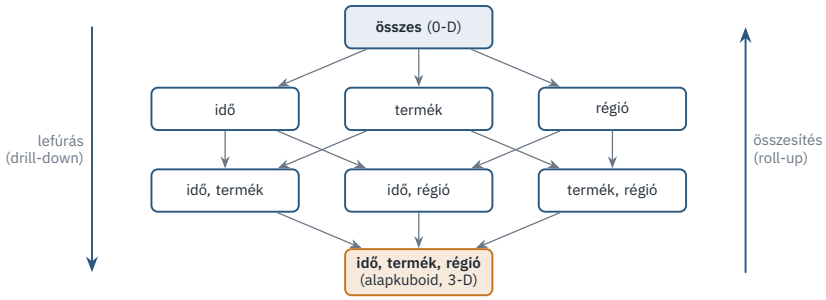
- **Kockázás** (dice): több dimenzió mentén egyszerre választunk részhalmazt (például két régió, két negyedév, két kategória).
- **Összesítés** (roll-up): egy dimenzió hierarchiájában feljebb lépünk (nap → hónap → negyedév → év), vagy egy dimenziót teljesen összegzünk.
- **Lefúrás** (drill-down): az összesítés fordítottja, a részletesebb szint felé.
- **Forgatás** (pivot): a dimenziók megjelenítési sorrendjének cseréje (sorok és oszlopok felcserélése a kimutatásban).

2.4.2. A kuboidok hálója

Egy d dimenziós kocka minden dimenzióhalmazára képezhető egy összesítés, amelyet **kuboidnak** nevezünk (Gray és mtsai., 1997). A kuboidok hálót alkotnak (2.4. ábra): a legalsó, d dimenziós **alapkuboid** a legrészletesebb, a legfelső, 0 dimenziós **csúcskuboid** egyetlen számot (a végösszeget) tartalmaz. Hierarchiák nélkül a kuboidok száma 2^d. Ha az i-edik dimenzióhoz L_i hierarchiaszint tartozik (a dátumnál például nap, hónap, negyedév, év: $L_i = 4$), akkor a kuboidok száma

$$T = \prod_{i=1}^{d} (L_i + 1), \tag{2.1}$$

ahol a „$+1$” a dimenzió teljes összegzésének („összes”) felel meg. Már néhány dimenzió és hierarchia esetén is nagyon sok kuboid adódik: négy dimenzió, egyenként négy szinttel $5^4 = 625$ kuboidot jelent. Ezért a rendszerek nem számolják ki előre az összeset, hanem csak a leggyakrabban használt vagy a legtöbb másikat kiszolgáló kuboidokat tárolják (**részleges materializáció**); ennek klasszikus mohó algoritmusát Harinarayan és mtsai. (1996) dolgozta ki.



2.4. ábra. A háromdimenziós (idő, termék, régió) adatkocka kuboidjainak hálója. Felfelé haladva összesítünk, lefelé részletezünk; egy kuboid minden fölötte lévőt kiszolgálhat.

Az SQL szabvány `GROUP BY ROLLUP` és `GROUP BY CUBE` kiegészítései (Gray és mtsai., 1997) egyetlen lekérdezéssel állítják elő egy hierarchia összes összesítési szintjét, illetve az összes kuboidot.

2.5. Korszerű analitikus adatarchitektúrák

2.5.1. Oszlopos tárolás

A hagyományos adatbázisok soronként tárolják az adatokat: egy rekord összes mezője egymás mellett van a lemezen. Ez az OLTP-műveleteknek kedvez, ahol egy teljes rekordot olvasunk vagy írunk. Az analitikus lekérdezések viszont tipikusan sok sor néhány oszlopát olvassák. Az **oszlopos tárolás** oszloponként helyezi el az adatokat, így egy lekérdezésnek csak a szükséges oszlopokat kell beolvasnia; ráadásul az azonos típusú, gyakran ismétlődő értékek nagyon jól tömöríthetők (Stonebraker és mtsai., 2005). Az oszlopos és soros tárolás közötti teljesítménykülönbség nem csupán a tárolási formátumból, hanem a végrehajtómotor ehhez illeszkedő kialakításából is fakad (Abadi és mtsai., 2008). Ma az analitikus adatbázisok és a nagy adatállományok fájlformátumai (például az Apache Parquet) túlnyomórészt oszlopos szerveződésűek.

2.5.2. Adattó és lakehouse

A nem strukturált és félig strukturált adatok (naplófájlok, szenzoradatok, képek, szövegek) megjelenésével elterjedt az **adattó** (data lake): olcsó objektumtárban, nyers formában, előzetes séma nélkül tárolt adatok gyűjteménye. Az adattó rugalmas, de könnyen „adatmocsárrá” válik, ha hiányzik a minőségbiztosítás és a metaadat-kezelés. A **lakehouse**-architektúra a két világ előnyeit próbálja egyesíteni: nyílt, oszlopos fájlformátumokban tárolt adatokra az adattárházakból ismert szolgáltatásokat (tranzakciókezelés, séma-érvényesítés, verziókövetés, gyors SQL-lekérdezés) épít, így ugyanazon az adaton futhatnak a jelentések és a gépi tanulási feladatok (Armbrust és mtsai., 2021).

A technológiák változnak, a dimenzionális gondolkodás azonban mindegyikben megmarad: a lakehouse „arany” rétegében is tény- és dimenziótáblák szolgálják ki az elemzőket. Ezért a következő gyakorlatban a klasszikus csillagsémát építjük fel.

2.6. Gyakorlat: csillagséma és OLAP Pythonban

2.6.1. A PannonShop csillagsémája SQLite-ban

A Python beépített `sqlite3` modulja egy teljes értékű, fájl- vagy memóriaalapú relációs adatbázist biztosít, így telepítés nélkül kipróbálható vele a csillagséma. Először a dimenziótáblákat állítjuk elő. A dátumdimenzió a vizsgált időszak minden napjára egy sort tartalmaz:

```
shop = ab.load_shop()
tx = shop.transactions.copy()

# --- dátumdimenzió ---
napok = pd.date_range("2024-01-01", "2025-12-31", freq="D")
dim_datum = pd.DataFrame({
    "datum_kulcs": napok.strftime("%Y%m%d").astype(int),
    "datum": napok.date.astype(str),
    "ev": napok.year,
    "negyedev": napok.year.astype(str) + "-N" + napok.quarter.astype(str),
    "honap": napok.strftime("%Y-%m"),
    "het_napja": napok.dayofweek + 1,          # 1 = hétfő
    "hetvege": (napok.dayofweek >= 5).astype(int),
})
dim_datum.head(3)
```

	datum_kulcs	datum	ev	negyedev	honap	het_napja	hetvege
0	20240101	2024-01-01	2024	2024-N1	2024-01	1	0
1	20240102	2024-01-02	2024	2024-N1	2024-01	2	0
2	20240103	2024-01-03	2024	2024-N1	2024-01	3	0

A termék- és régiódimenzió a törzsadatokból készül; a ténytábla a számlatételekből, amelyekben a természetes kulcsokat (cikkszám, régiónév) helyettesítő kulcsokra cseréljük. A postaköltség-sorokat kihagyjuk, mert nem termékértékesítést jelentenek; a sztornókat negatív mennyiségű tényként megtartjuk, így az összegzés automatikusan a nettó forgalmat adja.

```
dim_termek = shop.products.reset_index().rename(columns={"index":
↪  "termek_kulcs"})
dim_regio = (pd.DataFrame({"regio": sorted(tx.Regio.unique())})
             .reset_index().rename(columns={"index": "regio_kulcs"}))
dim_regio["orszag"] = np.where(dim_regio.regio.isin(["Ausztria",
↪  "Szlovákia", "Németország"]),
                               dim_regio.regio, "Magyarország")
```

```
f = tx[tx.StockCode != "POST"].copy()
f["Description"] = f.Description.str.strip()
f = (f.merge(dim_termek[["termek_kulcs", "StockCode"]], on="StockCode")
      .merge(dim_regio[["regio_kulcs", "regio"]], left_on="Regio",
 ↪  right_on="regio"))
teny = pd.DataFrame({
    "datum_kulcs": f.InvoiceDate.dt.strftime("%Y%m%d").astype(int),
    "termek_kulcs": f.termek_kulcs,
    "regio_kulcs": f.regio_kulcs,
    "vevo": f.CustomerID,                       # egyszerűség kedvéért
     ↪  természetes kulcs
    "szamla": f.Invoice,                        # degenerált dimenzió
    "mennyiseg": f.Quantity,
    "arbevetel": f.Quantity * f.Price,
})

con = sqlite3.connect(":memory:")
for nev, tabla in [("dim_datum", dim_datum), ("dim_termek", dim_termek),
                   ("dim_regio", dim_regio), ("teny_ertekesites", teny)]:
    tabla.to_sql(nev, con, index=False)
print(f"ténytábla: {len(teny):,} sor".replace(",", " "))
```

```
ténytábla: 66 653 sor
```

Most már egy tipikus elemzői kérdés – *negyedéves nettó árbevétel kategóriánként* – egyetlen, a csillagséma szerkezetét követő lekérdezés:

```
q = """
SELECT d.negyedev, t.Kategoria AS kategoria,
       SUM(f.arbevetel) / 1e6 AS arbevetel_mFt
FROM teny_ertekesites f
JOIN dim_datum  d ON f.datum_kulcs  = d.datum_kulcs
JOIN dim_termek t ON f.termek_kulcs = t.termek_kulcs
GROUP BY d.negyedev, t.Kategoria
ORDER BY d.negyedev, arbevetel_mFt DESC
"""
negyedeves = pd.read_sql(q, con)
negyedeves.head(6)
```

	negyedev	kategoria	arbevetel_mFt
0	2024-N1	Háztartási gépek	47.4
1	2024-N1	Dekoráció	30.7
2	2024-N1	Konyhai eszközök	30.4
3	2024-N1	Kávé és tea	29.5
4	2024-N1	Írószer	21.3
5	2024-N1	Ajándék	17.9

2.6.2. OLAP-műveletek a pandas kimutatástáblájával

A pandas `pivot_table` függvénye az adatkocka egy kétdimenziós nézetét állítja elő; a `margins=True` beállítás az összesítő sorokat és oszlopokat is hozzáadja, vagyis egyszerre két kuboidszintet mutat. A kocka három dimenziója itt az idő (negyedév), a termékkategória és a régió.

```
kocka = pd.read_sql("""
SELECT d.ev, d.negyedev, t.Kategoria AS kategoria, r.regio, r.orszag,
 ↪  f.arbevetel
FROM teny_ertekesites f
JOIN dim_datum d  ON f.datum_kulcs  = d.datum_kulcs
JOIN dim_termek t ON f.termek_kulcs = t.termek_kulcs
JOIN dim_regio r  ON f.regio_kulcs  = r.regio_kulcs
""", con)

# összesítés (roll-up): negyedév -> év, régió -> összes
kocka.pivot_table(index="kategoria", columns="ev", values="arbevetel",
                  aggfunc="sum", margins=True, margins_name="Összesen") / 1e6
```

ev kategoria	2024	2025	Összesen
Ajándék	82.3	131.4	213.7
Dekoráció	149.8	228.4	378.3
Háztartási gépek	264.1	369.2	633.3
Konyhai eszközök	160.6	198.3	358.9
Kávé és tea	153.9	223.9	377.8
Írószer	51.6	57.3	108.9
Összesen	862.3	1 208.6	2 070.9

A szeletelés egy dimenzió rögzítése, a kockázás több dimenzió mentén egyszerre történő szűrés:

```
# szelet: csak a "Kávé és tea" kategória, negyedév × ország
szelet = kocka[kocka.kategoria == "Kávé és tea"]
szelet.pivot_table(index="negyedev", columns="orszag", values="arbevetel",
                   aggfunc="sum").div(1e6).round(2).tail(4)
```

orszag negyedev	Ausztria	Magyarország	Németország	Szlovákia
2025-N1	3.7	67.8	2.2	1.0
2025-N2	2.3	40.4	0.8	0.0
2025-N3	1.3	31.5	0.9	0.2
2025-N4	1.9	66.3	1.9	1.5

```
# kocka-részlet (dice): két kategória, két régió, 2025 negyedévei
resz = kocka[kocka.kategoria.isin(["Háztartási gépek", "Ajándék"])
             & kocka.regio.isin(["Közép-Magyarország", "Nyugat-Dunántúl"])
             & (kocka.ev == 2025)]
resz.pivot_table(index=["kategoria", "regio"], columns="negyedev",
                 values="arbevetel", aggfunc="sum").div(1e6).round(2)
```

kategoria	negyedev regio	2025-N1	2025-N2	2025-N3	2025-N4
Ajándék	Közép-Magyarország	9.7	9.2	4.5	12.4
	Nyugat-Dunántúl	1.7	2.1	0.4	13.9
Háztartási gépek	Közép-Magyarország	27.8	23.8	14.0	50.2
	Nyugat-Dunántúl	8.4	3.9	3.4	9.5

Az SQLite nem ismeri a `GROUP BY ROLLUP` szerkezetet, de egy hierarchia összes összesítési szintje `UNION ALL`-lal könnyen előállítható. Az alábbi lekérdezés az év → negyedév hierarchia három szintjét adja vissza egyetlen táblában:

```
rollup = pd.read_sql("""
SELECT d.ev, d.negyedev, SUM(f.arbevetel)/1e6 AS mFt FROM teny_ertekesites
↪  f
  JOIN dim_datum d ON f.datum_kulcs = d.datum_kulcs GROUP BY d.ev,
↪  d.negyedev
UNION ALL
SELECT d.ev, 'összes', SUM(f.arbevetel)/1e6 FROM teny_ertekesites f
  JOIN dim_datum d ON f.datum_kulcs = d.datum_kulcs GROUP BY d.ev
UNION ALL
SELECT 'összes', 'összes', SUM(f.arbevetel)/1e6 FROM teny_ertekesites f
""", con)
rollup
```

	ev	negyedev	mFt
0	2024	2024-N1	177.3
1	2024	2024-N2	118.6
2	2024	2024-N3	144.7
3	2024	2024-N4	421.7
4	2025	2025-N1	318.9
5	2025	2025-N2	256.2
6	2025	2025-N3	200.3
7	2025	2025-N4	433.1
8	2024	összes	862.3
9	2025	összes	1 208.6
10	összes	összes	2 070.9

FUTÓ ESET · PANNONSHOP

A kimutatásokból már most látszik a PannonShop forgalmának két jelleg-

zetessége: a negyedik negyedév kiugró forgalma (karácsonyi szezon) és a háztartási gépek magas részesedése az árbevételből, amely a magas egységárakból fakad. Az első a 12. fejezet szezonális modelljeinek, a második a 14. fejezet vevőérték-számításának lesz fontos bemenete.

Tisztítatlan adatokon számoltunk

A fenti kimutatások a nyers számlatételekből készültek. A 3. fejezetben látni fogjuk, hogy néhány hibásan rögzített egységár és néhány szokatlanul nagy rendelés érezhetően torzítja az egyes cellák értékét. Egy valódi adattárházban az ETL folyamat feladata, hogy az ilyen hibákat a betöltés előtt kiszűrje vagy megjelölje.

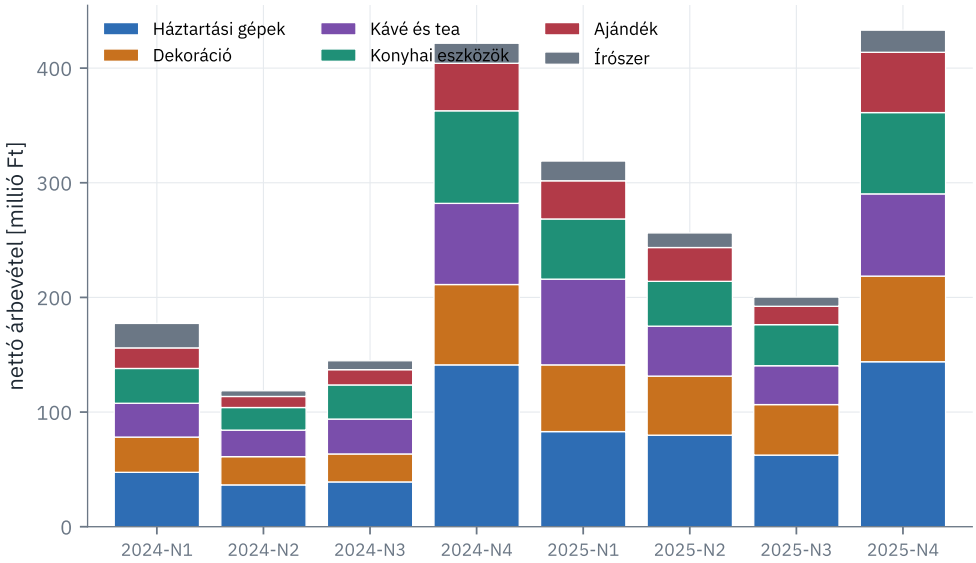
```
pv = negyedeves.pivot(index="negyedev", columns="kategoria",
    values="arbevetel_mFt")
pv = pv[pv.sum().sort_values(ascending=False).index]
ax = pv.plot(kind="bar", stacked=True, width=0.8, color=ab.PALETTE +
    ["#9AA5B1"], edgecolor="white", linewidth=0.6)
ax.set_xlabel("")
ax.set_ylabel("nettó árbevétel [millió Ft]")
ax.legend(ncol=3, fontsize=8, loc="upper left")
plt.xticks(rotation=0, fontsize=8)
plt.tight_layout()
```

2.6.3. A Pannon üzem termelési adatpiaca

A mérnöki esetben a ténytábla egy sora egy üzemóra. A mértékek közül a betáplált mennyiség és az energiafogyasztás additív, a hőmérsékletek és nyomások nem additívak. Egy fontos teljesítménymutató a **fajlagos energiafogyasztás**: az egységnyi betáplált nyersanyagra jutó villamos energia (kWh/m³). Mivel arányszám, az összesítéskor a számlálót és a nevezőt külön kell összegezni.

```
plant = ab.load_plant()
h = plant.hourly.dropna(subset=["F_betap", "E_fogyasztas"]).copy()
h = h[h.F_betap > 1.0]                              # a tisztítási
    leállásokat kihagyjuk
h["honap"] = h.ido.dt.to_period("M").astype(str)
h["energia_kWh"] = h.E_fogyasztas * 1.0             # órás
    átlagteljesítmény × 1 óra
h["betap_m3"] = h.F_betap * 1.0

def fajlagos(df):
```



2.5. ábra. A PannonShop negyedéves nettó árbevétele termékkategóriánként (millió Ft). A negyedik negyedévek kiemelkedése a karácsonyi szezont tükrözi.

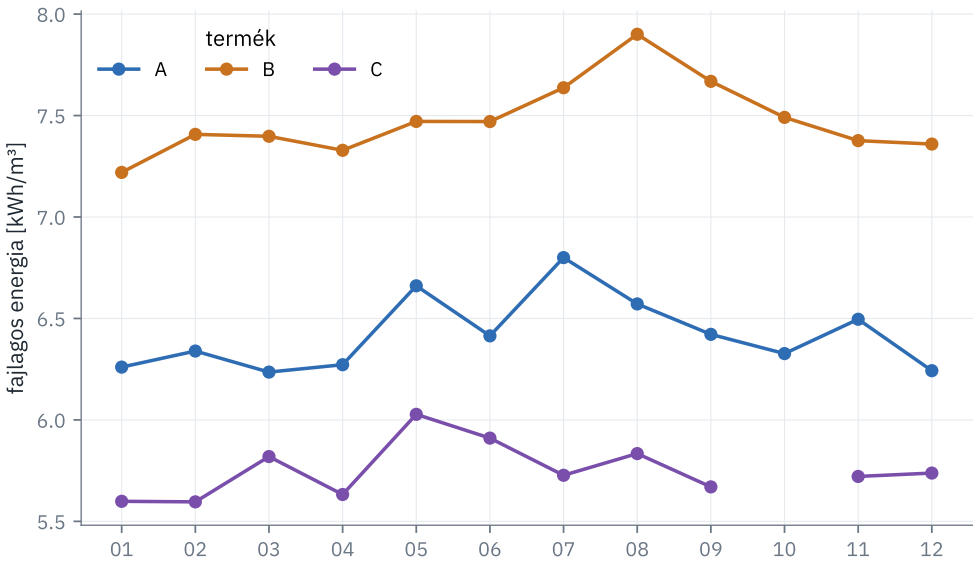
```
    """Fajlagos energia: összes energia / összes betáplálás (nem az órás
    ↪  arányok átlaga!)."""
    return df.energia_kWh.sum() / df.betap_m3.sum()

tabla = (h.groupby(["termek", "muszak"], observed=True)
          .apply(fajlagos, include_groups=False).unstack().round(3))
tabla
```

termek	muszak délelőtti	délutáni	éjszakai
A	6.5	6.5	6.4
B	7.5	7.5	7.4
C	5.8	5.8	5.7

```
havi = (h.groupby(["honap", "termek"]).apply(fajlagos,
 ↪  include_groups=False)
         .unstack())
ax = havi.plot(marker="o", color=ab.PALETTE[:3])
ax.set_xlabel("")
ax.set_ylabel("fajlagos energia [kWh/m³]")
ax.legend(title="termék", ncol=3)
```

```
plt.xticks(range(len(havi.index)), [m[5:] for m in havi.index])
plt.tight_layout()
```



2.6. ábra. A Pannon üzem havi fajlagos energiafogyasztása termékenként. A nyári hónapokban a nagyobb hűtővízáram (szivattyúzás) növeli az energiaigényt.

FUTÓ ESET · PANNON ÜZEM

A 2.6. ábra szerint a fajlagos energiafogyasztás nyáron megnő. Ennek oka a hűtés: melegebb hűtővíz és lerakódott hőcserélő mellett ugyanakkora hőmennyiség elvonásához több hűtővizet kell szivattyúzni. A fajlagos mutató a termékek között is különbözik, ezért egy havi összehasonlításnál a termékösszetétel változását is figyelembe kell venni – ez tipikus példája annak, amikor egy összesített mutató változása mögött az összetétel változása áll.

2.6.4. Kitérő: soros és oszlopos fájlformátum

A következő kis kísérlet a tárolási formátum hatását szemlélteti. A számlatételeket CSV-ként (soros, szöveges) és Parquet-formátumban (oszlopos, tömörített) is kiírjuk, majd összevetjük a méretüket és egyetlen oszlop beolvasásának idejét.

```
import os, time, tempfile
d = tempfile.mkdtemp()
csv_f, pq_f = os.path.join(d, "tx.csv"), os.path.join(d, "tx.parquet")
tx.to_csv(csv_f, index=False)
tx.to_parquet(pq_f, index=False)

t0 = time.perf_counter(); pd.read_csv(csv_f, usecols=["Price"]); t_csv =
↪  time.perf_counter() - t0
t0 = time.perf_counter(); pd.read_parquet(pq_f, columns=["Price"]); t_pq =
↪  time.perf_counter() - t0
print(f"CSV:     {os.path.getsize(csv_f)/1e6:5.1f} MB, egy oszlop
↪  beolvasása {t_csv*1000:6.1f} ms")
print(f"Parquet: {os.path.getsize(pq_f)/1e6:5.1f} MB, egy oszlop beolvasása
↪  {t_pq*1000:6.1f} ms")
```

```
CSV:       5.9 MB, egy oszlop beolvasása   51.9 ms
Parquet:   0.7 MB, egy oszlop beolvasása   11.0 ms
```

A pontos értékek a géptől függnek, de a nagyságrendi különbség jellemző: az oszlopos formátum kisebb, és csak a kért oszlopot kell beolvasni belőle.

Interaktív alkalmazás: OLAP-kocka: szeletelés, kockázás, összesítés

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/olap-kocka.html

Colab-notebook: 2. fejezet – csillagséma és OLAP

2.7. Összefoglalás

- Az operatív (OLTP) rendszerek a tranzakciók gyors és konzisztens kezelésére, az analitikus (OLAP) rendszerek a sok rekordot érintő lekérdezésekre optimalizáltak.
- Az adattárház tárgyorientált, integrált, időfüggő és nem illékony adatgyűjtemény; az adatintegrációt az ETL/ELT folyamat végzi.
- A dimenzionális modell ténytáblákból (mértékek) és dimenziótáblákból (leíró környezet) áll; a legfontosabb tervezési döntés a szemcsézettség.
- A mértékek additívak, félig additívak vagy nem additívak; arányokat a számlálójuk és nevezőjük külön összesítésével kell képezni.
- Az OLAP-műveletek (szeletelés, kockázás, összesítés, lefúrás, forgatás) az adatkocka kuboidjai között navigálnak; a kuboidok száma (2.1) szerint gyorsan nő.
- Az oszlopos tárolás, az adattó és a lakehouse a technológiát változtatta meg, a dimenzionális modellezés elveit nem.

Ellenőrző kérdések

1. Miért nem célszerű az elemzéseket közvetlenül az operatív adatbázison futtatni?
2. Mit jelent az adattárház négy Inmon-féle tulajdonsága? Mondjon mindegyikre példát a PannonShop világából!
3. Mi a szemcsézettség, és miért ajánlott a lehető legfinomabbat választani?
4. Mi a különbség az 1. és a 2. típusú lassan változó dimenzió között? Miért fontos ez egy lemorzsolódás-előrejelző modell tanításakor?
5. Hány kuboidja van egy négydimenziós kockának, ha a dimenziók hierarchiaszintjeinek száma 4, 3, 2 és 1?
6. Melyik mérték additív, félig additív, illetve nem additív: árbevétel, raktárkészlet, reaktorhőmérséklet, visszaküldési arány?

Feladatok

1. **Tervezés.** Tervezzen csillagsémát a PannonShop vásárlói értékeléseihez! Mi a szemcsézettség, melyek a mértékek és a dimenziók? Mely dimenziók közösek az értékesítési ténytáblával (galaxisséma)?
2. **Lekérdezés.** Az SQLite-adatbázisban kérdezze le, hogy hétvégén vagy hétköznap nagyobb-e az átlagos kosárérték! (Vigyázat: a kosárérték számlaszintű mérték, előbb számlánként kell összegezni.)
3. **Lefúrás.** A 2.5. ábrán a 2024. negyedik negyedév kiemelkedik. Fúrjon le havi, majd heti szintre, és állapítsa meg, mely hetekben koncentrálódik a forgalom!
4. **Összetételhatás.** Számítsa ki a Pannon üzem havi fajlagos energiafogyasztását termékbontás nélkül is! Mutassa meg egy példán, hogy az összesített mutató akkor is változhat, ha a termékenkénti értékek változatlanok!
5. **SCD.** Tegyük fel, hogy 2025. július 1-jén a „Teáskanna” termék a „Konyhai eszközök” kategóriából átkerül a „Kávé és tea” kategóriába. Valósítsa meg a változást
    2. típusú SCD-ként a termékdimenzióban, és mutassa meg, hogyan változik a kategóriánkénti 2025-ös árbevétel!

További olvasmányok

A dimenzionális modellezés alapműve Kimball & Ross (2013), az ETL-folyamatoké Kimball & Caserta (2004). Az Inmon-féle vállalati adattárház koncepcióját Inmon (2005) írja le. Az adatkocka és a kuboidháló elméletét Gray és mtsai. (1997), a részleges materializációt Harinarayan és mtsai. (1996)

tárgyalja. Az oszlopos adatbázisokról Abadi és mtsai. (2008), a lakehouse-architektúráról Armbrust és mtsai. (2021) ad áttekintést.

3. Adatok előkészítése

Miről szól a fejezet?

A valós adatok hiányosak, zajosak és ellentmondásosak. Mielőtt bármilyen modellt építenénk, meg kell ismernünk az adatok hibáit, és el kell döntenünk, hogyan kezeljük őket. Ez a fejezet sorra veszi az adatelőkészítés fő feladatait: az adattípusok és mérési skálák azonosítását, az adatminőség felmérését, a hiányzó értékek, a zaj és a kiugró értékek kezelését, a transzformációkat és kódolásokat, valamint a jellemzők kiválasztását. Külön alfejezet foglalkozik az adatszivárgással, a gyakorlati adatbányászat egyik leggyakoribb és legalattomosabb hibájával. A fejezet végére mindkét futó eset adatai megtisztított, elemzésre kész formába kerülnek.

A fejezet végére az olvasó képes lesz egy adattábla minőségi problémáit szisztematikusan feltárni; a hiányzás mechanizmusának megfelelő kezelési módot választani; robusztus módszerekkel kiugró értékeket keresni; a jellemzőket a modellhez illő skálára transzformálni és kódolni; és az előfeldolgozást szivárgásmentes `scikit-learn` folyamatba szervezni.

3.1. Adattípusok és mérési skálák

Egy változóval végezhető műveletek attól függnek, milyen **mérési skálán** mértük. Stevens (1946) klasszikus felosztása négy skálát különböztet meg:

3.1. táblázat. Mérési skálák és a rajtuk értelmezett műveletek.

Skála	Megengedett összevetés	Példa	Értelmes középérték
névleges (nominális)	egyenlő / nem egyenlő	régió, termékkategória	módusz
sorrendi (ordinális)	kisebb / nagyobb	csillagos értékelés, minőségi osztály	medián
intervallum	különbség	hőmérséklet °C-ban, dátum	átlag
arány	különbség és hányados	mennyiség, ár, áramlási sebesség	átlag, mértani közép

A névleges és a sorrendi skálájú változókat együtt **kategorikus**, az intervallum- és arányskálájúakat **numerikus** (folytonos vagy diszkrét) változóknak nevezzük.

A különbség nem formális szőrszálhasogatás: egy irányítószám számjegyekből áll, de névleges változó, átlagolása értelmetlen; a 20 °C nem „kétszer olyan meleg”, mint a 10 °C, mert a Celsius-skála nullpontja önkényes. Az adatbányászati algoritmusok többsége numerikus bemenetet vár, ezért a kategorikus változókat kódolni kell (3.5.3. alfejezet) – de úgy, hogy ne vigyünk be a kódolással nem létező sorrendet vagy távolságot.

A `pandas` adattípusai ezt a felosztást csak részben követik: a számok `int64` vagy `float64`, a szövegek `object` vagy `string`, a dátumok `datetime64`, és létezik egy külön `category` típus is a kategorikus változókra.

```
shop = ab.load_shop()
tx = shop.transactions
tx.dtypes
```

```
Invoice                   str
StockCode                 str
Description               str
Quantity                int64
InvoiceDate    datetime64[us]
Price                 float64
CustomerID                str
Regio                     str
dtype: object
```

3.2. Az adatminőség felmérése

Az adatminőség nem egyetlen tulajdonság. A felhasználói szemléletű adatminőségkutatás klasszikus eredménye szerint az adatok használói a pontosság mellett a teljességet, a konzisztenciát, az időszerűséget, az értelmezhetőséget és a hozzáférhetőséget is minőségi dimenziónak tekintik (R. Y. Wang & Strong, 1996). Az elemzés szempontjából hasznos továbbá a **rendezett adat** (tidy data) elve: minden változó egy oszlop, minden megfigyelés egy sor, minden megfigyelési egység egy tábla (Wickham, 2014). A számlatétel-tábla ennek megfelel – a vevőszintű elemzéshez azonban majd egy új, vevőnként egy sort tartalmazó táblát kell belőle képeznünk.

Az adatminőség felmérése a gyakorlatban egy ellenőrzőlista végigvitelét jelenti. Mennyi a hiányzó érték oszloponként? Vannak-e duplikált sorok? Értelmes tartományban vannak-e az értékek? Egységes-e a kódolás? Összeegyeztethetők-e a mezők egymással?

FUTÓ ESET · PANNONSHOP

Végezzük el ezt a felmérést a PannonShop számlatételein! Az egyes problématípusokat egyszerű logikai feltételekkel jelöljük meg, majd összesítjük.

```
p = shop.products.set_index("StockCode")
lista_ar = tx.StockCode.map(p.ListaAr)
problemak = pd.Series({
    "hiányzó vevőazonosító":          tx.CustomerID.isna().sum(),
    "sztornó (C-számla) tétel":       tx.Invoice.str.startswith("C").sum(),
    "postaköltség (nem termék)":       (tx.StockCode == "POST").sum(),
    "teljesen duplikált sor":          tx.duplicated().sum(),
    "nulla egységár":                  (tx.Price == 0).sum(),
    "listaár 20-szorosa feletti ár":   (tx.Price > 20 * lista_ar).sum(),
    "nem egységes megnevezés":         (tx.Description !=
     ↪  tx.Description.str.strip().str.capitalize()).sum(),
}, name="sorok száma")
problemak.to_frame().assign(arany=lambda d: (d["sorok száma"] / len(tx) *
 ↪  100).round(2))
```

	sorok száma	arany
hiányzó vevőazonosító	4716	6.84
sztornó (C-számla) tétel	1291	1.87
postaköltség (nem termék)	2337	3.39
teljesen duplikált sor	255	0.37
nulla egységár	139	0.20
listaár 20-szorosa feletti ár	32	0.05
nem egységes megnevezés	2893	4.19

A lista jól mutatja, hogy az adatminőségi problémák különböző természetűek, ezért különböző kezelést igényelnek:

- A **hiányzó vevőazonosító** a vendégvásárlásokból fakad. Nem pótolható (nem „tudhatjuk”, ki vásárolt), de nem is hiba: a forgalmi elemzésekben a sor megtartható, a vevőszintű elemzésekben viszont nem használható.
- A **sztornók** valós üzleti események. Egy forgalmi kimutatásban nettósítani kell őket, egy kosárelemzésben viszont a sztornózott számlát is el kell hagyni, mert az a kosár végül nem valósult meg.
- A **postaköltség** nem termék; a termékszintű elemzésekből ki kell zárni.
- A **duplikált sorok** rögzítési hibák (egy tétel kétszer került be), törölni kell őket. Óvatosan: két azonos sor lehet két valódi, azonos vásárlás is – itt az azonos számlaszám és időbélyeg teszi egyértelművé a hibát.
- A **nulla árú** tételek ajándékok; az árbevételhez nem járulnak hozzá, a kosárelemzésben viszont szerepelhetnek.
- A **listaár sokszorosán rögzített ár** tipikus elírás (két nullával több). Mivel a helyes érték kikövetkeztethető, javítható.

- A **nem egységes írásmód** (csupa nagybetű, szóköz a végén) egyszerű szövegnormalizálással orvosolható.

Az elírt egységárak csak a sorok töredékét érintik, mégis a teljes bruttó forgalom 3,5 százalékát adják – vagyis a 2. fejezet kimutatásait érezhetően torzították. Ez az adatminőség egyik alaptörvénye: kevés, de nagy értékű hiba is jelentősen eltorzíthatja az összesítéseket.

3.3. Hiányzó adatok

3.3.1. A hiányzás mechanizmusai

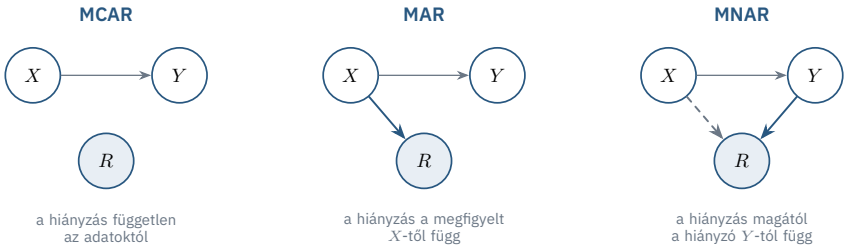
A hiányzó adatok kezelésének első kérdése nem az, hogy *mivel* pótoljuk őket, hanem az, hogy *miért* hiányoznak. Rubin (1976) nyomán három mechanizmust különböztetünk meg (3.1. ábra). Jelölje Y a hiányos változót, X a mindig megfigyelt változókat, R pedig a hiányzás jelzőjét ($R = 1$, ha Y értéke hiányzik).

- **Teljesen véletlenszerű hiányzás** (missing completely at random, MCAR): a hiányzás valószínűsége sem a megfigyelt, sem a hiányzó értékektől nem függ, $P(R = 1 \mid X, Y) = P(R = 1)$. Például egy szenzor véletlenszerű, rövid adatátviteli kiesései.
- **Véletlenszerű hiányzás** (missing at random, MAR): a hiányzás csak a *megfigyelt* változóktól függ, $P(R = 1 \mid X, Y) = P(R = 1 \mid X)$. Például ha a laboráns hétvégén ritkábban vesz mintát – a hiányzás a (megfigyelt) naptól függ, nem a koncentrációtól.
- **Nem véletlenszerű hiányzás** (missing not at random, MNAR): a hiányzás magától a hiányzó értéktől függ. Például egy érzékelő a méréshatára felett nem ad értéket, vagy a nagy jövedelműek gyakrabban tagadják meg a jövedelmük közlését.

A mechanizmus azért kulcsfontosságú, mert meghatározza, hogy egy egyszerű kezelés torzít-e. MCAR esetén a hiányos sorok elhagyása csak a mintanagyságot csökkenti, az eredményt nem torzítja. MAR esetén az elhagyás torzíthat, de a megfigyelt változók felhasználásával a torzítás korrigálható. MNAR esetén az adatokból magukból ez nem lehetséges; szakterületi ismeret vagy a hiányzás külön modellezése kell hozzá (Little & Rubin, 2019).

A következő kísérlet ezt szemlélteti. A Pannon üzem órás hűtővíz-hőmérsékletéből mesterségesen törlünk nagyjából 30–40%-nyi adatot a három mechanizmus szerint, és megnézzük, mennyire torzul a megmaradt adatok átlaga.

```
plant = ab.load_plant()
```



3.1. ábra. A hiányzási mechanizmusok Rubin (1976) nyomán. Az MCAR esetben a hiányzás független az adatoktól, MAR esetben a megfigyelt változóktól, MNAR esetben magától a hiányzó értéktől függ.

```
h = plant.hourly.dropna(subset=["T_hutoviz",
↪  "T_kornyezet"]).reset_index(drop=True)
y, x = h.T_hutoviz.to_numpy(), h.T_kornyezet.to_numpy()
rng = np.random.default_rng(1)

def logit_p(z, arany=0.3):
    """A z pontszámhoz tartozó hiányzási valószínűség, átlagosan 'arany'
    ↪  hiányzással."""
    z = (z - z.mean()) / z.std()
    b = np.log(arany / (1 - arany))
    return 1 / (1 + np.exp(-(b + 2.0 * z)))

hianyzik = {
    "MCAR": rng.random(len(y)) < 0.3,
    "MAR (környezeti hőm. függvénye)": rng.random(len(y)) < logit_p(x),
    "MNAR (saját értékének függvénye)": rng.random(len(y)) < logit_p(y),
}
pd.DataFrame({k: {"hiányzó arány": m.mean(), "megmaradt átlag":
↪  y[~m].mean(),
                  "torzítás [°C]": y[~m].mean() - y.mean()} for k, m in
                   ↪  hianyzik.items()}).T.round(3)
```

	hiányzó arány	megmaradt átlag	torzítás [°C]
MCAR	0.297	17.824	-0.014
MAR (környezeti hőm. függvénye)	0.392	15.766	-2.072
MNAR (saját értékének függvénye)	0.380	15.802	-2.036

A valódi átlag 17,84 °C. MCAR esetben a megmaradt adatok átlaga lényegében torzítatlan, a másik két esetben jelentősen eltér. MAR esetben azonban a torzítás javítható, mert a hiányzást okozó környezeti hőmérséklet ismert, és a hűtővíz

hőmérséklete jól becsülhető belőle (lásd az alábbi feladatot). MNAR esetben ez a lehetőség nincs meg.

3.3.2. Kezelési stratégiák

A hiányzó adatok kezelésének fő lehetőségei:

1. **Esetek elhagyása** (listwise deletion): minden hiányos sort törlünk. Egyszerű, MCAR esetén torzítatlan, de sok változó esetén a sorok nagy része elveszhet.
2. **Egyszerű pótlás**: az átlaggal, mediánnal (numerikus) vagy módusszal (kategorikus) helyettesítünk. Gyors, de a változó szórását alábecsüli, és a változók közötti kapcsolatokat gyengíti.
3. **Hiányjelző változó**: a pótlás mellé egy 0/1 jelzőt is felveszünk, így a modell „megtanulhatja”, ha a hiányzásnak magának is információtartalma van.
4. **Modellalapú pótlás**: a hiányzó értéket a többi változóból becsüljük. A k-legközelebbi szomszéd módszer a hasonló sorok értékeinek átlagát használja (Troyanskaya és mtsai., 2001); az iteratív (láncolt egyenletes) pótlás minden hiányos változóra regressziós modellt illeszt a többi változóból, és ezt felváltva ismétli (Buuren & Groothuis-Oudshoorn, 2011).
5. **Idősorokban** a szomszédos időpontok értékéből: lineáris interpolációval vagy az utolsó ismert érték továbbvitelével – de csak rövid hiányok esetén.

Hasonlítsuk össze ezeket a módszereket a Pannon üzem 10 perces adatain. Ehhez az eredetileg teljes adatokból véletlenszerűen törlünk értékeket, így a valódi érték ismert, és a pótlás hibája mérhető.

```
from sklearn.experimental import enable_iterative_imputer  # noqa: F401
from sklearn.impute import SimpleImputer, KNNImputer, IterativeImputer

oszlopok = ["T_kornyezet", "F_betap", "c_betap", "T_reaktor", "F_hutoviz",
 ↪  "T_hutoviz", "E_fogyasztas"]
teljes = ab.plant_clean().set_index("ido")[oszlopok].dropna().iloc[::6]   #
 ↪  óránkénti ritkítás
mask = rng.random(teljes.shape) < 0.10
mask[:, 0] = False                                                        # a
 ↪  környezeti hőm. mindig ismert
hianyos = teljes.mask(mask)

def rmse(potolt):
    return np.sqrt(((potolt - teljes)[mask].stack() ** 2).mean())

eredmeny = {
    "átlag": SimpleImputer(strategy="mean").fit_transform(hianyos),
    "medián": SimpleImputer(strategy="median").fit_transform(hianyos),
```

```
    "lineáris interpoláció (idő)":
     ↪  hianyos.interpolate(limit_direction="both").to_numpy(),
    "k-NN (k=5)": KNNImputer(n_neighbors=5).fit_transform((hianyos -
     ↪  hianyos.mean()) / hianyos.std())
                  * hianyos.std().to_numpy() + hianyos.mean().to_numpy(),
    "iteratív (láncolt regresszió)": IterativeImputer(random_state=0,
     ↪  max_iter=10).fit_transform(hianyos),
}
pd.Series({k: rmse(pd.DataFrame(v, index=teljes.index, columns=oszlopok))
           for k, v in eredmeny.items()}, name="RMSE").round(3).to_frame()
```

	RMSE
átlag	1.297
medián	1.305
lineáris interpoláció (idő)	0.363
k-NN (k=5)	0.581
iteratív (láncolt regresszió)	0.687

FUTÓ ESET · PANNON ÜZEM

Az eredmény a folyamatadatok természetét tükrözi. Az átlaggal vagy mediánnal történő pótlás a legrosszabb, mert figyelmen kívül hagyja, hogy a változók erősen összefüggnek egymással és önmaguk korábbi értékeivel. A változók közötti kapcsolatokat kihasználó modellalapú pótlás és az időbeli szomszédokat felhasználó interpoláció sokkal pontosabb. A gyakorlatban a kettő kombinálható: rövid kiesésekre interpoláció, hosszabb hiányokra modellalapú becslés – ez utóbbi már a 8. fejezet lágy szenzoraihoz vezet.

Buktató: a pótolt érték nem mért érték

A pótolt értékekkel a további elemzésben úgy dolgozunk, mintha mérések lennének, pedig becslések. Ezért (a) mindig jelöljük meg őket (hiányjelző változó), (b) a modell kiértékelésénél vegyük figyelembe, hogy a pótlás bizonytalansága nem jelenik meg az eredményben, és (c) a pótló modell paramétereit csak a tanítóadatokból becsüljük (3.7. alfejezet).

3.4. Zaj és kiugró értékek

3.4.1. Zaj és simítás

A **zaj** a mért érték véletlen ingadozása a valódi érték körül. A zaj nem hiba, hanem a mérés természetes velejárója; csökkentése simítással lehetséges. Az m hosszú,

középre igazított **mozgóátlag** az

$$\tilde{x}_t = \frac{1}{m} \sum_{i=-(m-1)/2}^{(m-1)/2} x_{t+i} \tag{3.1}$$

képlettel számolható (páratlan m esetén). A mozgóátlag a zajt csökkenti, de a hirtelen változásokat elkeni, és a kiugró értékekre érzékeny: egyetlen hibás érték m szomszédos simított értéket ront el. A **mozgómedián** ugyanebben az ablakban a mediánt veszi, így egy-egy kiugró értéket teljesen figyelmen kívül hagy, a lépcsőszerű változásokat pedig élesen megőrzi.

3.4.2. Kiugró értékek felismerése

A **kiugró érték** (outlier) olyan megfigyelés, amely a többitől annyira eltér, hogy felmerül a gyanú: más mechanizmus hozta létre – mérési hiba, rögzítési hiba, vagy éppen egy érdekes, ritka esemény. Az utóbbi esetben a kiugró érték nem eltávolítandó, hanem az elemzés célpontja (11. fejezet).

A legegyszerűbb módszer a **standardizált érték** (z-pontszám):

$$z_k = \frac{x_k - \bar{x}}{s}, \tag{3.2}$$

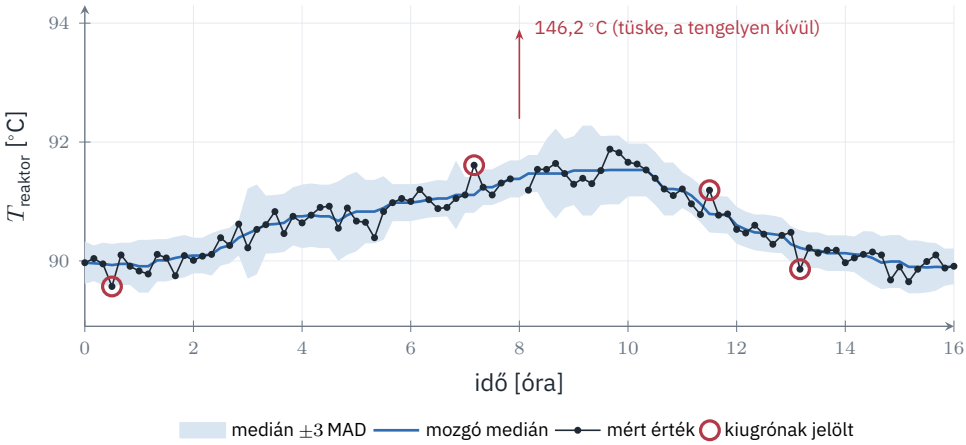
ahol $\bar{x}$ a mintaátlag, s a tapasztalati szórás; a $|z_k| > 3$ értékeket tekintjük kiugrónak. A módszer gyengéje, hogy az átlag és a szórás maga is érzékeny a kiugró értékekre: egy nagy hiba annyira megnövelheti a szórást, hogy önmaga már nem is tűnik kiugrónak (ez az *elfedés* jelensége). Ezért ajánlott a **robusztus z-pontszám**, amely az átlag helyett a mediánt, a szórás helyett a mediántól vett abszolút eltérések medián ját (MAD) használja (Leys és mtsai., 2013):

$$z_k^{\text{rob}} = \frac{x_k - \operatorname{med}(x)}{1{,}4826 \cdot \operatorname{MAD}}, \qquad \operatorname{MAD} = \operatorname{med}_k \big| x_k - \operatorname{med}(x) \big|. \tag{3.3}$$

Az $1{,}4826$-os szorzó azt biztosítja, hogy normális eloszlás esetén a nevező a szórás torzítatlan becslése legyen. Egy harmadik, a dobozdiagramokból ismert szabály (Tukey, 1977) az alsó és felső kvartilis (Q_1, Q_3) és az interkvartilis terjedelem ($\text{IQR} = Q_3 - Q_1$) alapján a $[Q_1 - 1{,}5\,\text{IQR},\ Q_3 + 1{,}5\,\text{IQR}]$ intervallumon kívüli pontokat jelöli.

Idősorokban a kiugró értékeket a *helyi* környezetükhöz kell viszonyítani, hiszen a folyamat szintje változik. A **Hampel-szűrő** a (3.3) képletet egy mozgó ablakban

alkalmazza: a mozgó mediántól k-szoros MAD-nál jobban eltérő pontokat jelöli (3.2. ábra).



3.2. ábra. A Hampel-szűrő működése a Pannon üzem reaktorhőmérsékletén (9 pontos ablak, $k = 3$). A 8. óránál egy adatátviteli hiba okozta, a tengely tartományán kívül eső tüske látható (146,2 °C); emellett a szűrő négy olyan pontot is megjelöl, amely csak alig lépi át a sávot – a küszöb megválasztása mindig kompromisszum a téves riasztás és a kimaradó hiba között.

```
m = plant.measurements.set_index("ido")
x = m.P_reaktor[m.F_betap > 1].dropna()
valodi = (x < 0.5) | (x > 3.4)          # a szimulátor ismert tüskéi: 0 és
↪  másfélszeres értékek
z = (x - x.mean()) / x.std()
zr = (x - x.median()) / (1.4826 * (x - x.median()).abs().median())
q1, q3 = x.quantile([0.25, 0.75])
jelolesek = {
    "z-pontszám (|z|>3)": z.abs() > 3,
    "robusztus z (|z|>3,5)": zr.abs() > 3.5,
    "Tukey-szabály (1,5 IQR)": (x < q1 - 1.5 * (q3 - q1)) | (x > q3 + 1.5 *
    ↪  (q3 - q1)),
    "Hampel-szűrő, k=4": ab.prep.hampel(x, window=7, k=4),
    "Hampel-szűrő, k=5": ab.prep.hampel(x, window=7, k=5),
}
pd.DataFrame({k: {"jelölt pont": v.sum(), "ebből valódi tüske": (v &
↪  valodi).sum()}
              for k, v in jelolesek.items()}).T.assign(**{"valódi tüskék
              ↪  száma": valodi.sum()})
```

	jelölt pont	ebből valódi tüske	valódi tüskék száma
z-pontszám (|z|>3)	50	50	50
robusztus z (|z|>3,5)	50	50	50
Tukey-szabály (1,5 IQR)	50	50	50
Hampel-szűrő, k=4	190	50	50
Hampel-szűrő, k=5	75	50	50

A reaktornyomás mérései között a szimulátor ismert helyeken nulla és másfélszeres értékű tüskéket helyezett el, így a módszerek pontossága ellenőrizhető. Ezek a *durva* hibák olyan messze esnek a többi értéktől, hogy mindhárom globális szabály hibátlanul megtalálja őket. A Hampel-szűrő is mindet megjelöli, de mellettük – kisebb k esetén – sok olyan pontot is, ahol a nyomás egy termékváltás vagy egy lassú ingadozás miatt gyorsan változik. A tanulság kettős. Egyrészt a módszert a keresett hibatípushoz kell választani: a durva hibákra a globális robusztus szabályok elegendők, a lokális módszer ott hasznos, ahol a kiugrás a folyamat *helyi* szintjéhez képest jelentős, de a teljes adat szórásához képest kicsi. Másrészt a küszöböt az ismert hibákkal kell kalibrálni – ezért használja a könyv segédcsomagja a tisztításhoz a konzervatívabb $k = 5$ értéket.

3.5. Transzformációk

3.5.1. Skálázás

Sok módszer – a távolságalapú csoportosítás, a k-legközelebbi szomszéd, a regularizált regresszió, a neurális hálózatok – érzékeny a változók mértékegységére. Ha egy vevő vásárlási értékét forintban, a vásárlások számát darabban mérjük, egy euklideszi távolságban a forint dominál, pusztán a nagyságrendje miatt. A leggyakoribb skálázások:

$$x'_{kj} = \frac{x_{kj} - \min_k x_{kj}}{\max_k x_{kj} - \min_k x_{kj}} \quad \text{(min–max)}, \qquad x'_{kj} = \frac{x_{kj} - \bar{x}_j}{s_j} \quad \text{(standardizálás)}, \tag{3.4}$$

valamint a robusztus skálázás, amely a medián és az interkvartilis terjedelem segítségével az (3.4) második képletének kiugró értékekre érzéketlen változata. A min–max skálázás a $[0, 1]$ intervallumba képez, de egyetlen kiugró érték az összes többit egy szűk sávba szoríthatja.

3.5.2. Eloszlás-transzformációk

A pénzügyi és mennyiségi változók (vásárlási érték, rendelésszám) eloszlása gyakran erősen **jobbra ferde**: a többség kis értékű, néhány vevő nagyon nagy értékű. Az ilyen változókat a **logaritmus** – vagy nullát is tartalmazó adatokra a $\log(1+x)$ – transzformáció közelíti a szimmetrikushoz. Általánosabb a pozitív adatokra értelmezett **Box–Cox-transzformáció** (Box & Cox, 1964),

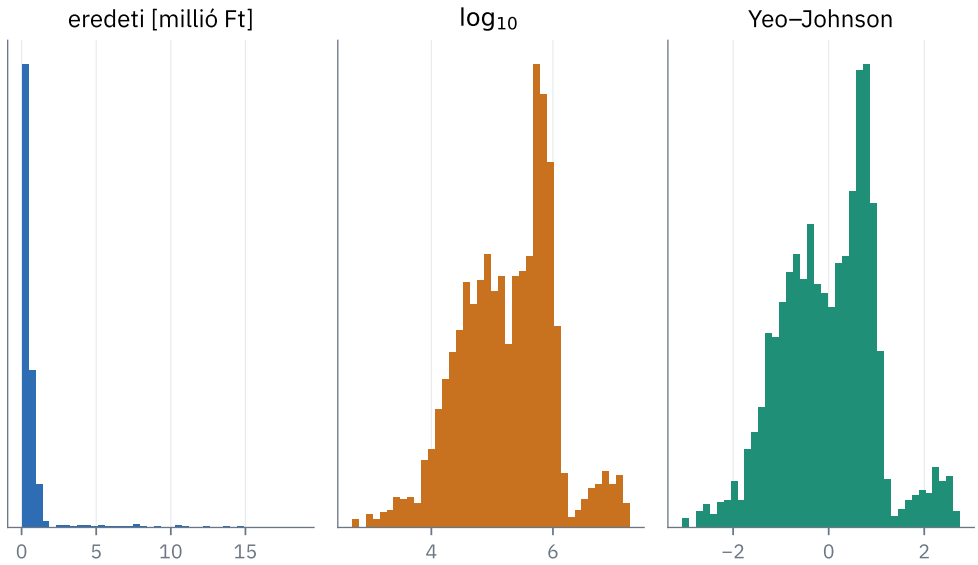
$$x^{(\lambda)} = \begin{cases} \dfrac{x^{\lambda}-1}{\lambda}, & \lambda \neq 0, \\ \ln x, & \lambda = 0, \end{cases} \tag{3.5}$$

amelynek λ paraméterét maximum-likelihood módszerrel úgy választjuk meg, hogy a transzformált adat a normális eloszláshoz a lehető legközelebb kerüljön. A **Yeo–Johnson-transzformáció** (Yeo & Johnson, 2000) ugyanezt az elvet nulla és negatív értékekre is kiterjeszti.

```
from sklearn.preprocessing import PowerTransformer
vevok = ab.customer_table()
M = vevok.M_ertek.to_numpy()
yj = PowerTransformer(method="yeo-johnson").fit_transform(M.reshape(-1,
↪  1)).ravel()
fig, ax = plt.subplots(1, 3)
ax[0].hist(M / 1e6, bins=40, color=ab.PALETTE[0]); ax[0].set_title("eredeti
↪  [millió Ft]")
ax[1].hist(np.log10(M), bins=40, color=ab.PALETTE[1]);
↪  ax[1].set_title("$\\log_{10}$")
ax[2].hist(yj, bins=40, color=ab.PALETTE[3]);
↪  ax[2].set_title("Yeo–Johnson")
for a in ax: a.set_yticks([])
fig.tight_layout()
```

3.5.3. Kategorikus változók kódolása

A névleges változókat a legtöbb modell számára **egy-forró** (one-hot) kódolással alakítjuk numerikussá: egy C értéket felvevő változó helyett C darab 0/1 értékű jelzőváltozót képzünk. Lineáris modelleknél az egyik jelzőt el kell hagyni, különben a jelzők összege konstans, és a modell paraméterei nem azonosíthatók egyértelműen (*dummy-csapda*). A sorrendi változók kódolhatók egész számokkal, ha a sorrend értelmes, a távolságok egyenlőségét azonban ezzel feltételezzük.



3.3. ábra. A PannonShop vevőinek teljes vásárlási értéke (M) eredeti skálán, logaritmus után és Yeo–Johnson-transzformáció után. Az eredeti eloszlás erősen jobbra ferde; a transzformációk után közel szimmetrikus.

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder(sparse_output=False, drop="first")
regio = shop.customers[["Regio"]].head(5)
pd.DataFrame(enc.fit_transform(shop.customers[["Regio"]])[:5],
             columns=enc.get_feature_names_out(),
 ↪  index=regio.Regio).iloc[:, :5]
```

Regio	Regio_Dél-Alföld	Regio_Dél-Dunántúl	Regio_Közép-Dunántúl	Regio_Közép-Magyarország	Regio_Ny
Közép-Magyarország	0.0	0.0	0.0	1.0	0.0
Nyugat-Dunántúl	0.0	0.0	0.0	0.0	1.0
Közép-Magyarország	0.0	0.0	0.0	1.0	0.0
Közép-Magyarország	0.0	0.0	0.0	1.0	0.0
Dél-Dunántúl	0.0	1.0	0.0	0.0	0.0

Sok értéket felvevő változóknál (például cikkszám, irányítószám) az egy-forró kódolás túl sok oszlopot eredményez. Ilyenkor szóba jön a **célváltozó szerinti kódolás** (target encoding), amely minden kategóriát a célváltozó adott kategórián belüli átlagával helyettesít – ez azonban a szivárgás egyik tipikus forrása, ha nem kizárólag a tanítóadatokból számoljuk (3.7. alfejezet).

3.5.4. Diszkretizálás

A **diszkretizálás** egy folytonos változót intervallumokra bont. Az *egyenlő szélességű* felosztás a tartományt azonos hosszú szakaszokra osztja, az *egyenlő gyakoriságú* (kvantilis alapú) felosztás úgy, hogy minden szakaszba nagyjából azonos számú megfigyelés essen. Ferde eloszlásoknál a két módszer nagyon eltérő eredményt ad: egyenlő szélesség esetén szinte minden megfigyelés az első intervallumba kerül. A diszkretizálás információvesztéssel jár, de értelmezhetőbbé tehet egy modellt, és egyes módszerek (például a 6. fejezet asszociációs szabályai) kategorikus bemenetet igényelnek.

```
from sklearn.preprocessing import KBinsDiscretizer
r = vevok[["R_nap"]]
eq_w = KBinsDiscretizer(n_bins=4, encode="ordinal",
 ↪  strategy="uniform").fit(r)
eq_f = KBinsDiscretizer(n_bins=4, encode="ordinal",
 ↪  strategy="quantile").fit(r)
pd.DataFrame({
    "egyenlő szélesség – határok": [np.round(eq_w.bin_edges_[0], 0)],
    "egyenlő gyakoriság – határok": [np.round(eq_f.bin_edges_[0], 0)],
}).T.rename(columns={0: "intervallumhatárok (nap)"})
```

	intervallumhatárok (nap)
egyenlő szélesség – határok	[0.0, 182.0, 365.0, 548.0, 730.0]
egyenlő gyakoriság – határok	[0.0, 23.0, 138.0, 373.0, 730.0]

3.5.5. Új jellemzők képzése

Az adatelőkészítés talán legtöbb szakértelmet igénylő része az új, a feladat szempontjából informatív jellemzők képzése. A számlatételekből a vevő viselkedését leíró vevőszintű táblát állítunk elő; ennek klasszikus magja az **RFM-modell** (Hughes, 1994): a legutóbbi vásárlás óta eltelt idő (*recency*), a vásárlások száma (*frequency*) és a vásárlások összértéke (*monetary*). Ezt kiegészítjük az átlagos kosárértékkel, a kosárméretttel, a vásárlások közötti átlagos idővel, az akciós időszakban történt vásárlások arányával és a kategóriánkénti költési arányokkal.

```
vevok = ab.customer_table(ref_date="2026-01-01")
vevok.head().round(2)
```

CustomerID	R_nap	F_szamla	M_ertek	atl_kosarertek	atl_kosarmeret	atl_mennyiseg	...	arany_ajandek	arany_dekor	arany_ge
12001	23	9	605752.0	67305.78	5.11	1.89	...	0.01	0.02	0.25
12002	44	3	223468.0	74489.33	3.67	2.93	...	0.07	0.12	0.52

CustomerID	R_nap	F_szamla	M_ertek	atl_kosarertek	atl_kosarmeret	atl_mennyiseg	...	arany_ajandek	arany_dekc
12003	39	6	132622.0	22103.67	2.00	1.53	...	0.12	0.67
12005	388	2	53940.0	26970.00	2.00	2.00	...	0.91	0.09
12006	32	4	1212704.0	303176.00	5.50	2.49	...	0.00	0.07

A folyamatadatokból hasonlóan képezhetők új jellemzők: időbeli eltolt értékek (*lag*), mozgóátlagok és mozgó szórások, naptári jellemzők (óra, a hét napja), vagy a fizikai ismeretekből adódó származtatott mennyiségek (például a hűtési teljesítmény a hűtővízáram és a hőmérséklet-különbség szorzataként). Az ilyen, szakterületi tudásra épülő jellemzők gyakran többet javítanak egy modellen, mint a modellcsalád cseréje.

3.6. Jellemzők kiválasztása

A sok jellemző nem feltétlenül jobb: a lényegtelen jellemzők zajt visznek a modellbe, növelik a számítási igényt és a túltanulás veszélyét, a modell értelmezését pedig megnehezítik. A jellemzőkiválasztás módszerei három csoportba sorolhatók (Guyon & Elisseeff, 2003; Kohavi & John, 1997):

- **Szűrő** (filter) módszerek a modelltől függetlenül, egy statisztikai mérőszám alapján rangsorolják a jellemzőket: varianciaküszöb, a célváltozóval vett korreláció vagy kölcsönös információ.
- **Burkoló** (wrapper) módszerek a modell teljesítménye alapján keresnek jó jellemzőhalmazt: például előre- vagy hátralépéses kiválasztással, vagy rekurzív jellemzőelhagyással. Pontosabbak, de számításigényesek, és könnyen túltanulnak.
- **Beágyazott** (embedded) módszerekben a kiválasztás a tanítás része: a 8. fejezet lasszó-regressziója nulla együtthatót ad a lényegtelen változóknak, a 9. fejezet döntési fái pedig csak a hasznos változókat használják vágásra.

A **Pearson-féle korrelációs együttható** két numerikus változó lineáris kapcsolatának erősségét méri:

$$r_{xy} = \frac{\sum_{k=1}^{N}(x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{N}(x_k - \bar{x})^2}\sqrt{\sum_{k=1}^{N}(y_k - \bar{y})^2}} \in [-1, 1]. \tag{3.6}$$

A korreláció csak lineáris kapcsolatot mér; a nemlineáris függések kimutatására a **kölcsönös információ** alkalmasabb, amely akkor és csak akkor nulla, ha a két változó független.

FUTÓ ESET · PANNON ÜZEM

Mely mért folyamatváltozók hordoznak információt a laborban mért termékkoncentrációról? A szűrő módszerekkel ez gyorsan felmérhető. (A célváltozó csak 4 óránként ismert, ezért csak a laborminták időpontjaiban dolgozunk.)

```
from sklearn.feature_selection import mutual_info_regression
d = ab.plant_clean().dropna(subset=["c_termek_labor"])
d = d[~d.leallas]
X = d[["T_kornyezet", "F_betap", "c_betap", "T_betap", "T_reaktor",
 ↪  "F_hutoviz",
       "T_hutoviz", "P_reaktor", "P_kevero", "E_fogyasztas"]].dropna()
y = d.loc[X.index, "c_termek_labor"]
szuro = pd.DataFrame({
    "|Pearson r|": X.corrwith(y).abs(),
    "kölcsönös információ": mutual_info_regression(X, y, random_state=0),
}).sort_values("kölcsönös információ", ascending=False)
szuro.round(3)
```

	|Pearson r|	kölcsönös információ
c_betap	0.887	1.340
T_reaktor	0.063	0.640
P_kevero	0.837	0.618
F_betap	0.755	0.558
T_betap	0.008	0.472
P_reaktor	0.209	0.332
E_fogyasztas	0.569	0.246
F_hutoviz	0.270	0.088
T_hutoviz	0.157	0.032
T_kornyezet	0.166	0.018

A kölcsönös információ szerint a betáplált koncentráció és a reaktorhőmérséklet áll az élen – a szimulátor konverziós modellje alapján ez várható is, hiszen a termékkoncentráció a betáplált koncentráció és a hőmérséklettől függő konverzió szorzata. Figyelemre méltó, hogy a reaktorhőmérséklet Pearson-korrelációja közel nulla: a kapcsolat nem egyszerűen lineáris, mert a hőmérséklet-alapjel és a betáplált koncentráció termékenként együtt változik. A keverőteljesítmény és a betáplálási áram erős korrelációja pedig abból fakad, hogy mindkettő a gyártott terméktől függ – közvetítő változók, nem okok. Az eredmény egyúttal figyelmeztet a szűrő módszerek korlátjára: egyenként rangsorolják a változókat, így nem veszik észre sem a redundanciát (több változó ugyanazt az információt hordozza), sem a kölcsönhatásokat (két változó csak együtt informatív).

3.7. Adatszivárgás és a szivárgásmentes folyamat

Adatszivárgásról (data leakage) akkor beszélünk, ha a modell tanítása során olyan információ jut a modellbe, amely az alkalmazás idején nem állna rendelkezésre (S. Kaufman és mtsai., 2012). A szivárgás következménye mindig ugyanaz: a modell a kiértékeléskor kiválóan teljesít, élesben pedig csalódást okoz. Leggyakoribb formái:

- **Jövőbeli információ** a jellemzők között: például egy lemorzsolódás-előrejelző modellben a „lemondta a hírlevelet” változó, amelyet a vevő éppen az elpártoláskor állít be.
- **Előfeldolgozás a teljes adaton**: ha a skálázás, a pótlás vagy a jellemzőkiválasztás paramétereit a teszthalmazzal együtt becsüljük, a teszthalmaz információja beszivárog a tanításba.
- **Duplikátumok és csoportok átfedése**: ha ugyanannak a vevőnek a rekordjai a tanító- és a teszthalmazba is kerülnek, a modell „felismeri” a vevőt.
- **Időbeli keveredés**: idősorokon véletlen felosztással a modell a jövőből tanul a múlt előrejelzéséhez (12. fejezet).

A második típus különösen alattomos, mert jóhiszemű, „szokásos” lépésekből adódik. A következő kísérlet ezt mutatja be. Tisztán véletlen adatokat generálunk – a jellemzők semmilyen kapcsolatban nincsenek a célváltozóval, így egyetlen modell sem lehet jobb a véletlen találgatásnál (50%) –, majd a 20 „legjobb” jellemzőt *a teljes adatbázison* választjuk ki, és csak ezután értékeljük a modellt keresztvalidációval.

```
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline

rng = np.random.default_rng(0)
Xz = rng.normal(size=(100, 5000))           # 100 megfigyelés, 5000
↪  zajváltozó
yz = rng.integers(0, 2, size=100)            # véletlen címkék

# HIBÁS: a kiválasztás a teljes adaton történik, csak utána validálunk
X_sel = SelectKBest(f_classif, k=20).fit_transform(Xz, yz)
hibas = cross_val_score(LogisticRegression(max_iter=1000), X_sel, yz,
↪  cv=5).mean()

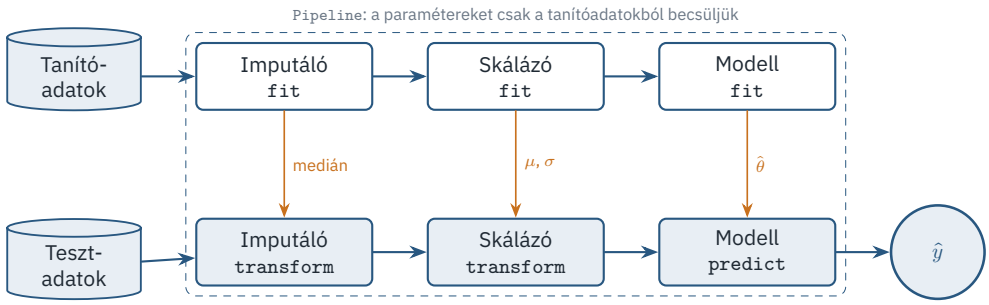
# HELYES: a kiválasztás a folyamat része, minden validációs körben csak a
↪  tanítórészen fut
pipe = make_pipeline(SelectKBest(f_classif, k=20),
↪  LogisticRegression(max_iter=1000))
helyes = cross_val_score(pipe, Xz, yz, cv=5).mean()
```

```
print(f"szivárgással:    {hibas:.2f}\nszivárgás nélkül: {helyes:.2f}")
```

```
szivárgással:    0.85
szivárgás nélkül: 0.48
```

A szivárgásos változat lényegesen jobb eredményt mutat a véletlennél – teljesen zajos adaton! A magyarázat: a teljes adaton kiválasztott 20 jellemző éppen azért került kiválasztásra, mert *ezen a mintán* véletlenül együtt mozog a címkékkel, és ez a véletlen együttmozgás a validációs részhalmazokban is megvan. Ezt a hibát az orvosbiológiai irodalom is dokumentálta (Ambroise & McLachlan, 2002).

A megoldás a `scikit-learn` **Pipeline** (folyamat) objektuma (3.4. ábra): az előfeldolgozási lépéseket és a modellt egyetlen egységbe fűzi, amelynek `fit` hívása minden lépés paramétereit csak a kapott (tanító)adatokból becsli, a `predict` hívás pedig ugyanezekkel a paraméterekkel alakítja át az új adatokat. A különböző típusú oszlopok eltérő kezelését a `ColumnTransformer` teszi lehetővé.



3.4. ábra. A szivárgásmentes adatfeldolgozási folyamat. Az imputáló, a skálázó és a modell paramétereit csak a tanítóadatokból becsüljük; a teszt- és az éles adatokat ugyanezekkel a – rögzített – paraméterekkel alakítjuk át.

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer

numerikus = ["R_nap", "F_szamla", "M_ertek", "atl_kosarertek",
↪  "atl_koztes_nap"]
kategorikus = ["Regio"]
elokeszito = ColumnTransformer([
    ("num", make_pipeline(SimpleImputer(strategy="median",
↪  add_indicator=True),
                          PowerTransformer()), numerikus),
    ("kat", OneHotEncoder(handle_unknown="ignore"), kategorikus),
])
```

```
tabla = vevok.join(shop.customers.set_index("CustomerID")[["Regio"]])
Xp = elokeszito.fit_transform(tabla)
print(Xp.shape)
print(elokeszito.get_feature_names_out()[:8])
```

```
(2394, 17)
['num__R_nap' 'num__F_szamla' 'num__M_ertek' 'num__atl_kosarertek'
 'num__atl_koztes_nap' 'num__missingindicator_atl_koztes_nap'
 'kat__Regio_Ausztria' 'kat__Regio_Dél-Alföld']
```

3.8. Gyakorlat: a futó esetek megtisztítása

A fejezetben kidolgozott lépéseket a könyv segédcsomagja egy-egy függvényben foglalja össze, hogy a további fejezetek ugyanabból a megtisztított adatból indulhassanak:

- `ab.prep.clean_transactions(tx, products)`: szövegnormalizálás, duplikátumok törlése, postaköltség és nulla árú sorok elhagyása, elírt egységárak javítása, sztornózott számlák eltávolítása; a `ab.shop_clean()` rövidítés ezt az alapértelmezett adatokra hívja;
- `ab.prep.customer_features(...)`: a vevőszintű RFM- és vásárlásiszerkezet-tábla (`ab.customer_table()`);
- `ab.prep.clean_plant(m)`: a fizikailag lehetetlen értékek és a Hampel-szűrő által jelölt tüskék eltávolítása, rövid hiányok interpolációja, a leállások jelölése (`ab.plant_clean()`).

```
tiszta = ab.shop_clean()
print(f"nyers: {len(tx):,} sor  ->  tisztított: {len(tiszta):,}
↪  sor".replace(",", " "))
print(f"nettó árbevétel (tisztított): {tiszta.Ertek.sum()/1e6:,.1f} millió
↪  Ft".replace(",", " "))
```

```
nyers: 68 990 sor  ->  tisztított: 63 687 sor
nettó árbevétel (tisztított): 1 986.6 millió Ft
```

Interaktív alkalmazás: Adattisztítás: hiányzó értékek pótlása és kiugró értékek szűrése

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/adattisztitas.html

Colab-notebook: 3. fejezet – adatelőkészítés

3.9. Összefoglalás

- A változó mérési skálája (névleges, sorrendi, intervallum, arány) meghatározza a rajta értelmes műveleteket és a megfelelő kódolást.
- Az adatminőség-felmérés szisztematikus ellenőrzőlista; a feltárt problémák különböző természetűek, és különböző kezelést igényelnek.
- A hiányzó adatok kezelésének első kérdése a mechanizmus (MCAR, MAR, MNAR); az egyszerű pótlás a szórást alábecsüli, a modellalapú pótlás a változók közötti kapcsolatokat is felhasználja.
- A kiugró értékek keresésére robusztus módszereket (medián, MAD, Hampel-szűrő) érdemes használni; a kiugró érték lehet hiba és lehet értékes ritka esemény is.
- A skálázás, az eloszlás-transzformációk, a kódolás és a diszkretizálás a modell igényeihez igazítják az adatokat; a jó új jellemzők szakterületi tudást hordoznak.
- Az adatszivárgás elkerülésének legfontosabb eszköze, hogy minden adatfüggő előfeldolgozási lépést csak a tanítóadatokon illesztünk (Pipeline).

Ellenőrző kérdések

1. Miért értelmetlen az irányítószámok átlaga? Milyen skálán mérjük a csillagos értékelést?
2. Mi a különbség az MCAR, a MAR és az MNAR mechanizmus között? Melyikbe sorolná a PannonShop hiányzó vevőazonosítóit?
3. Miért alábecsüli az átlaggal való pótlás a változó szórását?
4. Mi az elfedés jelensége, és miért védi ki a robusztus z-pontszám?
5. Miért kell a skálázás paramétereit csak a tanítóadatokból becsülni?
6. Mi a dummy-csapda, és hogyan kerülhető el?

Feladatok

1. **MAR korrekciója.** A hiányzó adatos kísérletben a MAR esetben pótolja a hiányzó hűtővíz-hőmérsékleteket a környezeti hőmérsékletre illesztett lineáris regresszióval! Mennyire csökken az átlag torzítása?
2. **Kiugró árak.** Termékenként számítsa ki az egységárak robusztus z-pontszámát! Mely tételek ára gyanús, és melyek magyarázhatók akciós árazással?
3. **Diszkretizálás.** Bontsa a vevők összes vásárlási értékét (M) öt egyenlő szélességű és öt egyenlő gyakoriságú intervallumra! Hány vevő esik az egyes intervallumokba? Melyik felosztás alkalmasabb egy szegmentációhoz, és miért?

4. **Szivárgás.** Ismételje meg a zajváltozós kísérletet 50, 500 és 5000 zajváltozóval! Hogyan függ a szivárgásos becslés optimizmusa a változók számától?
5. **Új jellemzők.** Képezzen a Pannon üzem adataiból egy „hűtési teljesítmény” jellemzőt (hűtővízáram × [reaktorhőmérséklet − hűtővíz-hőmérséklet])! Erősebb-e a kapcsolata a termékkoncentrációval, mint az eredeti változóké?

További olvasmányok

A hiányzó adatok elméletének alapműve Little & Rubin (2019); a többszörös pótlás gyakorlatát Buuren & Groothuis-Oudshoorn (2011) írja le. A jellemzőkiválasztásról Guyon & Elisseeff (2003) ad áttekintést. Az adatszivárgás formalizálását és felismerésének módszereit S. Kaufman és mtsai. (2012) tárgyalja. A rendezett adat elvéről Wickham (2014), az adatminőség dimenzióiról R. Y. Wang & Strong (1996) olvasható.

4. Feltáró adatelemzés és dimenziócsökkentés

Miről szól a fejezet?

Mielőtt modellt építenénk, meg kell ismernünk az adatainkat: hogyan oszlanak el az egyes változók, hogyan függnek össze egymással, vannak-e csoportok, trendek, szokatlan pontok. Ez a fejezet a feltáró adatelemzés (exploratory data analysis) eszköztárát mutatja be: a leíró statisztikákat, az egyváltozós eloszlások ábrázolását (hisztogram, magfüggvényes sűrűségbecslés, dobozdiagram, kvantilis ábrák), a többváltozós kapcsolatok vizsgálatát (korreláció, pontdiagram-mátrix, párhuzamos koordináták), végül a sokdimenziós adatok áttekintését lehetővé tevő dimenziócsökkentési módszereket – a főkomponens-elemzést, a többdimenziós skálázást és a t-SNE-t.

A fejezet végére az olvasó képes lesz egy adathalmazt leíró statisztikákkal és ábrákkal jellemezni; a megfelelő ábratípust kiválasztani; a főkomponens-elemzést levezetni, végrehajtani és értelmezni (megmagyarázott variancia, főkomponens-értékek, súlyok); és megítélni a nemlineáris vetítések erősségeit és csapdáit.

4.1. A feltáró adatelemzés szemlélete

A feltáró adatelemzés fogalmát Tukey (1977) vezette be, szembeállítva a hipotézisvizsgáló (megerősítő) statisztikával. A megerősítő elemzés egy előre megfogalmazott kérdésre keres választ; a feltáró elemzés célja éppen a kérdések *megtalálása*: mit mutatnak az adatok, mi meglepő bennük, milyen feltevéseket érdemes később ellenőrizni. Ennek fő eszköze a vizualizáció – nem azért, mert a számok rosszak, hanem mert egyetlen szám sok, nagyon különböző adatsort írhat le.

Ezt a legvilágosabban Anscombe (1973) négy adatsora mutatja (4.1. táblázat és 4.1. ábra). A négy adatsorban az x és y változók átlaga, szórása, korrelációja és a rájuk illesztett regressziós egyenes két tizedesjegy pontossággal megegyezik – az ábrák viszont négy teljesen különböző helyzetet mutatnak: egy közel lineáris kapcsolatot, egy görbe vonalút, egy egyetlen kiugró érték által eltorzított lineárisat, és egy olyat, ahol a korrelációt egyetlen pont hozza létre.

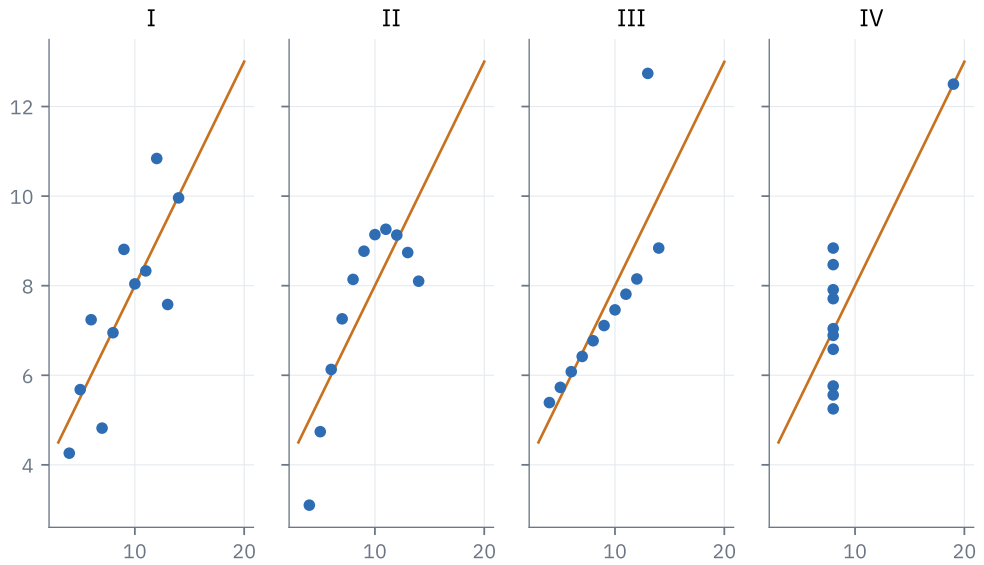
```
anscombe = {
    "I":   ([10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5],
            [8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82,
 ↪  5.68]),
    "II":  ([10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5],
            [9.14, 8.14, 8.74, 8.77, 9.26, 8.10, 6.13, 3.10, 9.13, 7.26,
 ↪  4.74]),
    "III": ([10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5],
            [7.46, 6.77, 12.74, 7.11, 7.81, 8.84, 6.08, 5.39, 8.15, 6.42,
 ↪  5.73]),
    "IV":  ([8, 8, 8, 8, 8, 8, 8, 19, 8, 8, 8],
            [6.58, 5.76, 7.71, 8.84, 8.47, 7.04, 5.25, 12.50, 5.56, 7.91,
 ↪  6.89]),
}
def jellemzok(x, y):
    x, y = np.array(x, float), np.array(y, float)
    b1, b0 = np.polyfit(x, y, 1)
    return {"átlag x": x.mean(), "átlag y": y.mean(), "szórás x":
     ↪  x.std(ddof=1),
            "szórás y": y.std(ddof=1), "korreláció": np.corrcoef(x, y)[0,
             ↪  1],
            "meredekség": b1, "tengelymetszet": b0}
pd.DataFrame({k: jellemzok(*v) for k, v in anscombe.items()}).round(2)
```

4.1. táblázat. Anscombe négy adatsorának leíró statisztikái.

	I	II	III	IV
átlag x	9.00	9.00	9.00	9.00
átlag y	7.50	7.50	7.50	7.50
szórás x	3.32	3.32	3.32	3.32
szórás y	2.03	2.03	2.03	2.03
korreláció	0.82	0.82	0.82	0.82
meredekség	0.50	0.50	0.50	0.50
tengelymetszet	3.00	3.00	3.00	3.00

```
fig, ax = plt.subplots(1, 4, sharex=True, sharey=True)
xx = np.array([3, 20])
for a, (k, (x, y)) in zip(ax, anscombe.items()):
    b1, b0 = np.polyfit(x, y, 1)
    a.scatter(x, y, s=18, color=ab.PALETTE[0], zorder=3)
    a.plot(xx, b0 + b1 * xx, color=ab.PALETTE[1], lw=1.2)
    a.set_title(k)
fig.tight_layout()
```

A tanulság egyszerű, de alapvető: **mielőtt egy összefüggést számszerűsítenénk, nézzük meg.** A vizualizáció tervezésének elveiről a további olvasmányok között ajánlott irodalom szól; itt csak egy, kísérletekkel alátámasztott szempontot eme-



4.1. ábra. Anscombe négy adatsora: azonos leíró statisztikák, teljesen eltérő szerkezet (Anscombe, 1973).

lünk ki: az emberi szem a közös skálán elhelyezkedő pozíciókat sokkal pontosabban hasonlítja össze, mint a szögeket, területeket vagy színárnyalatokat (W. S. Cleveland & McGill, 1984). Ezért mennyiségek összehasonlítására az oszlop- és pontdiagram jobb választás, mint a kördiagram.

4.2. Egy változó eloszlásának jellemzése

4.2.1. Leíró statisztikák

Egy numerikus változó $x_1, \ldots, x_N$ mintáját három szempontból jellemezzük: hol helyezkedik el (közép), mennyire szóródik, és milyen az alakja.

Középértékek. A számtani átlag $\bar{x} = \frac{1}{N}\sum_k x_k$ minden megfigyelést figyelembe vesz, de érzékeny a kiugró értékekre. A medián (a rendezett minta középső eleme) robusztus. A módusz a leggyakoribb érték; kategorikus változóknál ez az egyetlen értelmes középérték.

Szóródás. A tapasztalati szórásnégyzet és a szórás

$$s^2 = \frac{1}{N-1}\sum_{k=1}^{N}(x_k - \bar{x})^2, \qquad s = \sqrt{s^2}. \tag{4.1}$$

Az $N-1$-es nevező (Bessel-korrekció) teszi a becslést torzítatlanná; a részleteket a C. függelék tárgyalja. Robusztus szóródási mérőszám az interkvartilis terjedelem (IQR $= Q_3 - Q_1$) és a MAD (3.4. alfejezet).

Alak. A ferdeség és a csúcsosság a harmadik, illetve negyedik centrált momentumon alapul:

$$g_1 = \frac{\frac{1}{N}\sum_k (x_k-\bar{x})^3}{\left(\frac{1}{N}\sum_k (x_k-\bar{x})^2\right)^{3/2}}, \qquad g_2 = \frac{\frac{1}{N}\sum_k (x_k-\bar{x})^4}{\left(\frac{1}{N}\sum_k (x_k-\bar{x})^2\right)^{2}} - 3. \tag{4.2}$$

A normális eloszlás ferdesége és (a -3-mal korrigált) csúcsossága nulla. Pozitív ferdeség jobbra elnyúló, hosszú jobb oldali „farkú” eloszlást jelez – ilyen a vásárlási érték és a legtöbb mennyiségi változó.

```
vevok = ab.customer_table()
oszl = ["R_nap", "F_szamla", "M_ertek", "atl_kosarertek", "atl_koztes_nap"]
leiro = vevok[oszl].describe().T
leiro["ferdeség"] = vevok[oszl].skew()
leiro["csúcsosság"] = vevok[oszl].kurt()
leiro.round(1)
```

	count	mean	std	min	25%	50%	75%	max	ferdeség	csúcsosság
R_nap	2394.0	219.1	220.3	0.0	23.0	137.5	373.0	730.0	0.7	-0.7
F_szamla	2394.0	5.9	5.4	1.0	2.0	4.0	9.0	29.0	1.3	1.0
M_ertek	2394.0	781383.1	2076022.5	490.0	49230.0	208955.0	624605.0	18663770.0	4.9	25.9
atl_kosarertek	2394.0	124430.8	316794.4	490.0	23424.6	41174.4	78813.3	3876120.0	6.2	50.7
atl_koztes_nap	1880.0	80.7	73.5	0.0	38.5	55.9	97.1	700.0	2.7	10.1

FUTÓ ESET · PANNONSHOP

A vevőszintű táblában az átlag és a medián (50%) között nagy a különbség a vásárlási érték és az átlagos kosárérték esetén: az átlag többszöröse a mediánnak. Ez erős jobbra ferdeséget jelez – kevés, nagyon sokat vásárló vevő (a viszonteladók) húzza fel az átlagot. Az ilyen változókra a medián jobb „tipikus érték”, és a modellezés előtt logaritmikus vagy más eloszlástranszformáció indokolt (3.5. alfejezet).

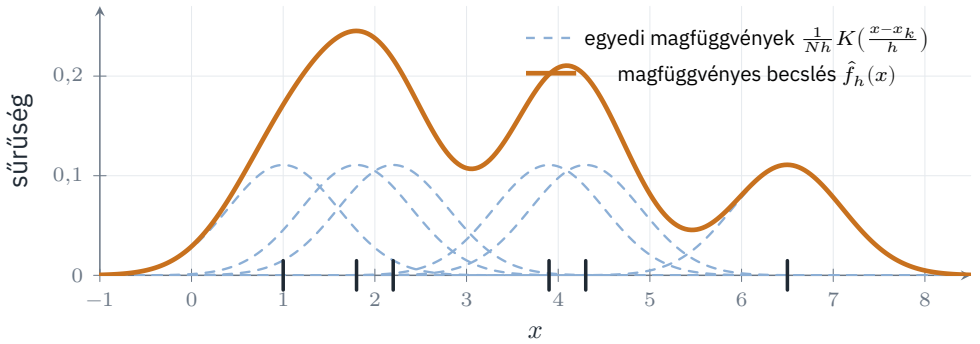
4.2.2. Hisztogram és magfüggvényes sűrűségbecslés

A **hisztogram** a változó tartományát egyenlő szélességű intervallumokra bontja, és mindegyik fölé az oda eső megfigyelések számával (vagy relatív gyakoriságával) arányos magasságú oszlopot rajzol. A kép erősen függ az intervallumszélességtől: túl keskeny intervallumoknál zajos, túl szélesnél elmossa a szerkezetet. A Freedman–Diaconis-szabály az intervallumszélességet a $h = 2\,\mathrm{IQR}\,N^{-1/3}$ képlettel választja (Freedman & Diaconis, 1981); ez a kiugró értékekre kevéssé érzékeny.

A **magfüggvényes sűrűségbecslés** (kernel density estimation, KDE) a hisztogram sima változata (4.2. ábra). Minden megfigyelés köré egy kis, egységnyi területű „dombot” (magfüggvényt, K) helyezünk, és ezeket összegezzük:

$$\hat{f}_h(x) = \frac{1}{Nh}\sum_{k=1}^{N} K\!\left(\frac{x - x_k}{h}\right), \qquad K(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}. \tag{4.3}$$

A h sávszélesség a hisztogram intervallumszélességének szerepét tölti be. Egy gyakran használt választás Silverman ökölszabálya, $h = 0{,}9\,\min(s, \mathrm{IQR}/1{,}34)\,N^{-1/5}$ (Silverman, 1986), amely közel normális eloszlásra optimális, többcsúcsú eloszlásokra viszont túlsimíthat.

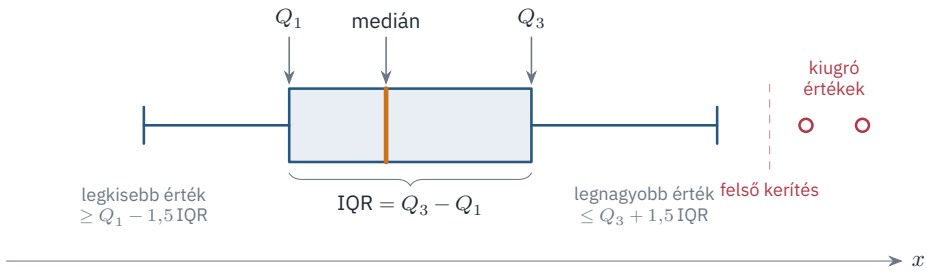


4.2. ábra. A magfüggvényes sűrűségbecslés elve: minden megfigyelés (függőleges vonalkák) köré egy Gauss-magfüggvényt helyezünk, a becslés ezek összege.

4.2.3. Dobozdiagram, kvantilis ábrák

A **dobozdiagram** (Tukey, 1977) öt számmal foglalja össze az eloszlást (4.3. ábra): a doboz a két kvartilis között húzódik, benne a medián, a „bajuszok” a kerítése-

ken belüli legszélső értékekig nyúlnak, a kerítéseken kívüli pontokat egyenként jelöljük. Előnye, hogy sok csoport eloszlása tömören egymás mellé rajzolható.

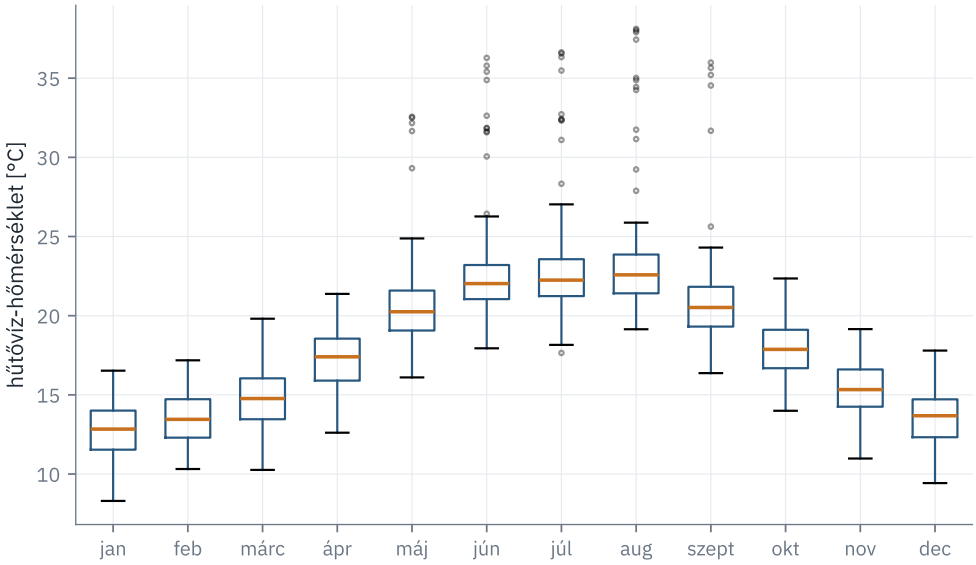


4.3. ábra. A dobozdiagram felépítése. A kerítések a kvartilisektől 1,5 interkvartilis terjedelemnyire vannak; a rajtuk kívül eső pontok a lehetséges kiugró értékek.

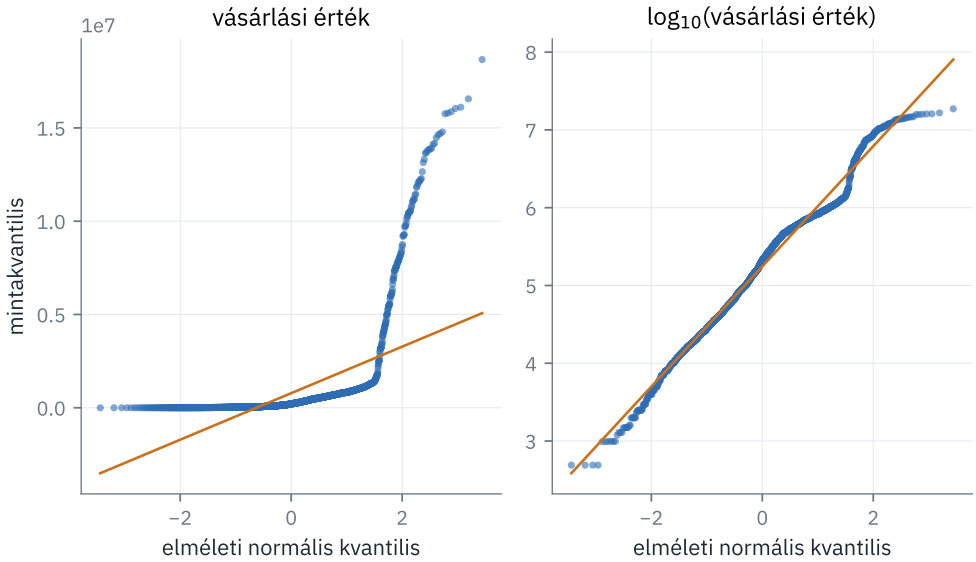
```
plant = ab.load_plant()
h = plant.hourly.dropna(subset=["T_hutoviz"])
csoport = [g.T_hutoviz.values for _, g in h.groupby(h.ido.dt.month)]
fig, ax = plt.subplots()
ax.boxplot(csoport, widths=0.6, flierprops=dict(marker="o", markersize=2,
 ↪  alpha=0.4),
           medianprops=dict(color=ab.PALETTE[1], lw=1.6),
           boxprops=dict(color=ab.PE_BLUE),
 ↪  whiskerprops=dict(color=ab.PE_BLUE))
ax.set_xticks(range(1, 13), ["jan", "feb", "márc", "ápr", "máj", "jún",
 ↪  "júl", "aug", "szept", "okt", "nov", "dec"])
ax.set_ylabel("hűtővíz-hőmérséklet [°C]")
fig.tight_layout()
```

A **kvantilis–kvantilis (Q–Q) ábra** két eloszlás alakját hasonlítja össze: a minta rendezett értékeit egy elméleti eloszlás (leggyakrabban a normális) megfelelő kvantiliseinek függvényében ábrázolja. Ha a minta az elméleti eloszlásból származik, a pontok egy egyenesre esnek; a görbülés ferdeséget, az „S” alak a normálisnál nehezebb vagy könnyebb farkakat jelez.

```
fig, ax = plt.subplots(1, 2)
for a, adat, cim in [(ax[0], vevok.M_ertek, "vásárlási érték"), (ax[1],
 ↪  np.log10(vevok.M_ertek), "log$_{10}$(vásárlási érték)")]:
    (osm, osr), (b, a0, r) = stats.probplot(adat, dist="norm")
    a.scatter(osm, osr, s=5, color=ab.PALETTE[0], alpha=0.6)
    a.plot(osm, a0 + b * osm, color=ab.PALETTE[1], lw=1.2)
    a.set_title(cim); a.set_xlabel("elméleti normális kvantilis")
ax[0].set_ylabel("mintakvantilis")
fig.tight_layout()
```



4.4. ábra. A hűtővíz-hőmérséklet havi eloszlása dobozdiagramokon. Jól látszik az évszakos változás és az, hogy nyáron a szóródás is nagyobb; a kiugró pontok egy része a hűtővíz-kiesésekhez tartozik.



4.5. ábra. Normális Q–Q ábra a vevők teljes vásárlási értékére (balra) és annak logaritmusára (jobbra). Az eredeti változó erősen jobbra ferde; a logaritmus jóval közelebb van a normálishoz, bár a két szélen még eltér tőle.

4.3. Több változó együttes vizsgálata

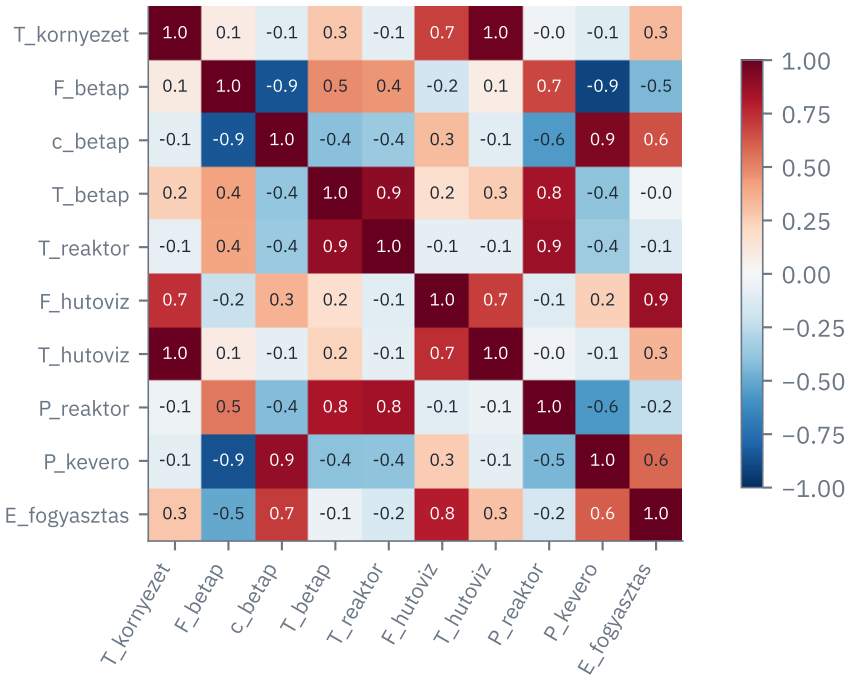
4.3.1. Korreláció

Két numerikus változó lineáris kapcsolatát a Pearson-féle korrelációs együttható méri (3.6). A **Spearman-féle rangkorreláció** (Spearman, 1904) ugyanezt a képletet a megfigyelések rangszámaira alkalmazza; így bármilyen monoton (nem csak lineáris) kapcsolatot kimutat, és kevéssé érzékeny a kiugró értékekre. Sok változó esetén a páronkénti korrelációkat **korrelációs mátrixba** rendezzük, és hőtérképen ábrázoljuk.

```
valt = ["T_kornyezet", "F_betap", "c_betap", "T_betap", "T_reaktor",
↪  "F_hutoviz",
        "T_hutoviz", "P_reaktor", "P_kevero", "E_fogyasztas"]
hh = plant.hourly[plant.hourly.F_betap > 1][valt].dropna()
P, S = hh.corr(method="pearson").to_numpy(),
↪  hh.corr(method="spearman").to_numpy()
M = np.where(np.triu(np.ones_like(P, bool), 1), P, S)
np.fill_diagonal(M, 1)
fig, ax = plt.subplots()
im = ax.imshow(M, cmap="RdBu_r", vmin=-1, vmax=1)
ax.set_xticks(range(len(valt)), valt, rotation=60, ha="right", fontsize=8)
ax.set_yticks(range(len(valt)), valt, fontsize=8)
for i in range(len(valt)):
    for j in range(len(valt)):
        ax.text(j, i, f"{M[i, j]:.1f}", ha="center", va="center",
↪  fontsize=6.5,
                color="white" if abs(M[i, j]) > 0.6 else "#1F2933")
ax.grid(False)
fig.colorbar(im, ax=ax, shrink=0.8)
fig.tight_layout()
```

A korreláció nem okozati kapcsolat

A 4.6. ábrán a betáplálási áram és a keverőteljesítmény erősen korrelál, holott egyik sem okozza a másikat: mindkettő a gyártott termékminőségtől függ. Az ilyen közös okot **zavaró változónak** (confounder) nevezzük. A korreláció mindig csak együttmozgást jelez; okozati következtetéshez kísérlet vagy a zavaró változók kontrollálása szükséges.



4.6. ábra. A Pannon üzem órás folyamatváltozóinak Pearson-féle (bal felső háromszög) és Spearman-féle (jobb alsó háromszög) korrelációs mátrixa.

4.3.2. Pontdiagram-mátrix és párhuzamos koordináták

A **pontdiagram-mátrix** minden változópárra egy pontdiagramot rajzol, az átlóban az egyes változók eloszlásával. Sok változónál áttekinthetetlenné válik, de 4–6 változóra kiváló első áttekintést ad, különösen ha a pontokat egy kategorikus változó szerint színezzük.

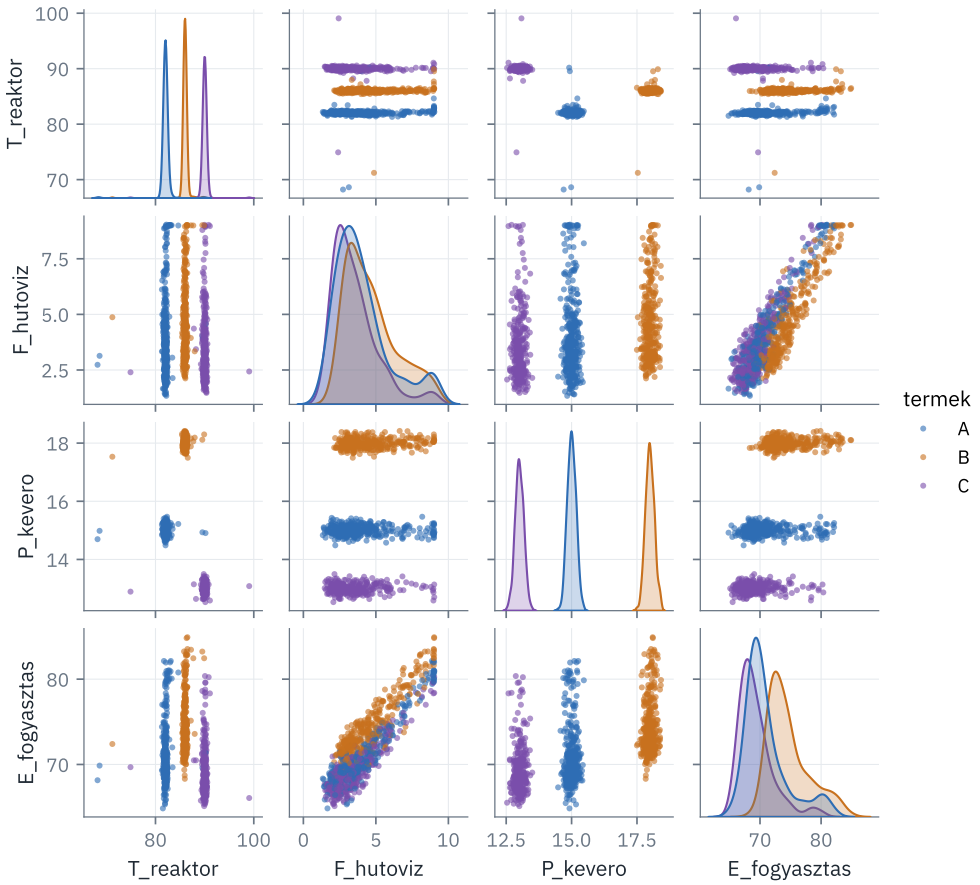
```
import seaborn as sns
minta = (plant.hourly[plant.hourly.F_betap > 1]
         .dropna(subset=["T_reaktor"]).sample(1000, random_state=0))
g = sns.pairplot(minta[["T_reaktor", "F_hutoviz", "P_kevero",
 ↪  "E_fogyasztas", "termek"]],
                 hue="termek", hue_order=["A", "B", "C"],
 ↪  palette=ab.PALETTE[:3],
                 plot_kws=dict(s=8, alpha=0.6, linewidth=0),
 ↪  diag_kind="kde", height=1.5)
```

A **párhuzamos koordináták** (Inselberg, 1985) a változókat egymással párhuzamos függőleges tengelyekre helyezik, és minden megfigyelést a tengelyeken felvett értékeit összekötő töröttvonalként ábrázolnak. Így tetszőleges számú változó egy ábrán jeleníthető meg; a csoportok egymástól elkülönülő vonalkötegekként, a negatív korreláció a tengelyek között kereszteződő vonalakként ismerhető fel. Az értelmezhetőséghez a változókat azonos skálára (például $[0, 1]$) kell hozni.

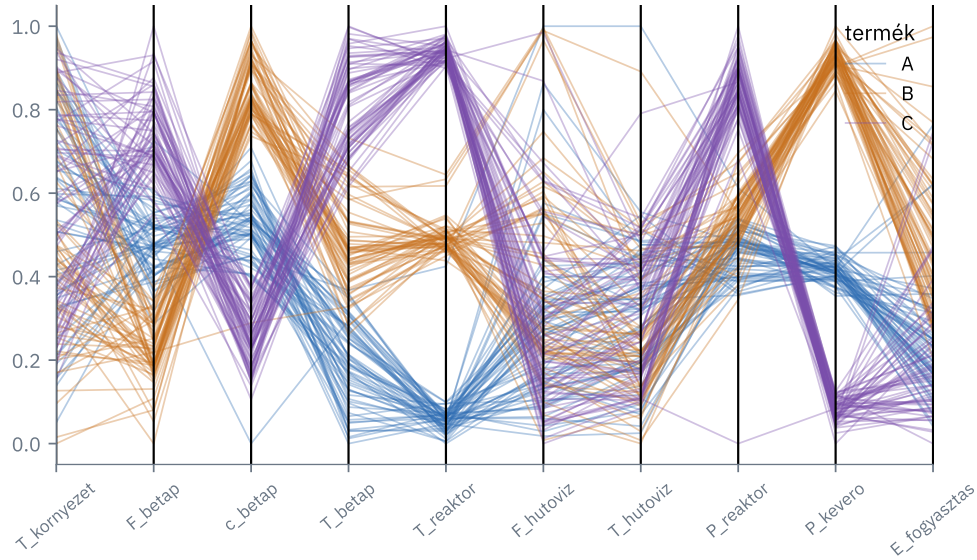
```
from pandas.plotting import parallel_coordinates
pk = plant.hourly[plant.hourly.F_betap > 1].dropna(subset=valt)
pk = pk.groupby("termek", group_keys=False).sample(60, random_state=1)
norm = (pk[valt] - pk[valt].min()) / (pk[valt].max() - pk[valt].min())
norm["termek"] = pk.termek.values
fig, ax = plt.subplots()
parallel_coordinates(norm, "termek", color=ab.PALETTE[:3], alpha=0.35,
 ↪  lw=0.8, ax=ax)
ax.tick_params(axis="x", rotation=40, labelsize=8)
ax.legend(title="termék", loc="upper right")
fig.tight_layout()
```

4.4. Dimenziócsökkentés: főkomponens-elemzés

Ha egy objektumot sok változó ír le, a változók közötti összefüggések miatt az adatok gyakran egy alacsonyabb dimenziós altér közelében helyezkednek el. A Pannon üzem tíz folyamatváltozója például nem tíz független információt hordoz: a hőmérsékletek, áramok és teljesítmények együtt mozognak, mert ugyanazok a



4.7. ábra. Pontdiagram-mátrix a Pannon üzem négy változójára, termékenként színezve (1000 véletlen órás minta). A termékek jól elkülönülő felhőket alkotnak a reaktorhőmérséklet és a keverőteljesítmény síkjában.



4.8. ábra. Párhuzamos koordináták a Pannon üzem órás adataira (min–max skálázás után, termékenként 60 véletlen óra).

fizikai törvények kötik össze őket. A **dimenziócsökkentés** célja, hogy az adatokat kevesebb új változóval írjuk le úgy, hogy a lényeges információ megmaradjon. Ez megkönnyíti a megjelenítést (2–3 dimenzió ábrázolható), csökkenti a zajt és a számítási igényt, és a korrelált változók problémáját is kezeli.

4.4.1. Levezetés: a legnagyobb variancia iránya

A **főkomponens-elemzés** (principal component analysis, PCA) Pearson (1901) és Hotelling (1933) munkáira megy vissza; korszerű áttekintését Jolliffe & Cadima (2016) adja. Az ötlet: keressük azt az irányt, amely mentén az adatok szórása a legnagyobb, majd erre merőlegesen a következő legnagyobb szórású irányt, és így tovább (4.9. ábra). A szükséges lineáris algebrai fogalmakat (skaláris szorzat, vetítés, sajátérték, sajátvektor, szinguláris felbontás) a B. függelék részletesen tárgyalja.

Legyen $\mathbf{X}_c$ az oszloponként centrált adatmátrix (minden oszlopból kivontuk az oszlop átlagát), és

$$\mathbf{C} = \frac{1}{N-1}\,\mathbf{X}_c^{\top}\mathbf{X}_c \in \mathbb{R}^{n\times n} \tag{4.4}$$

a tapasztalati kovarianciamátrix. Egy $\mathbf{v}$ egységvektor irányába vetítve az adatokat, a k-adik objektum vetülete $t_k = \mathbf{x}_{c,k}^\top \mathbf{v}$, a vetületek szórásnégyzete pedig

$$\frac{1}{N-1}\sum_{k=1}^{N} t_k^2 = \frac{1}{N-1}\,\mathbf{v}^\top \mathbf{X}_c^\top \mathbf{X}_c \mathbf{v} = \mathbf{v}^\top \mathbf{C}\,\mathbf{v}.$$

A legnagyobb szórású irány tehát a

$$\max_{\mathbf{v}} \ \mathbf{v}^\top \mathbf{C}\,\mathbf{v} \quad \text{feltéve, hogy} \quad \mathbf{v}^\top \mathbf{v} = 1 \tag{4.5}$$

feltételes szélsőérték-feladat megoldása. A Lagrange-függvény $\mathcal{L}(\mathbf{v}, \lambda) = \mathbf{v}^\top \mathbf{C}\mathbf{v} - \lambda(\mathbf{v}^\top \mathbf{v} - 1)$; a $\mathbf{v}$ szerinti gradienst nullává téve

$$\nabla_{\mathbf{v}} \mathcal{L} = 2\mathbf{C}\mathbf{v} - 2\lambda \mathbf{v} = \mathbf{0} \quad \Longrightarrow \quad \mathbf{C}\mathbf{v} = \lambda \mathbf{v}. \tag{4.6}$$

Az optimális irány tehát $\mathbf{C}$ egy sajátvektora, és mivel ilyenkor $\mathbf{v}^\top \mathbf{C}\mathbf{v} = \lambda \mathbf{v}^\top \mathbf{v} = \lambda$, a vetületek szórásnégyzete éppen a megfelelő sajátérték. A legnagyobb szórást a **legnagyobb sajátértékhez** tartozó sajátvektor adja: ez az **első főkomponens** iránya. Mivel $\mathbf{C}$ szimmetrikus és pozitív szemidefinit, sajátértékei nemnegatív valós számok, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$, a sajátvektorok pedig ortonormált bázist alkotnak. A második főkomponens az elsőre merőleges irányok közül a legnagyobb szórású – ez a második sajátvektor –, és így tovább.

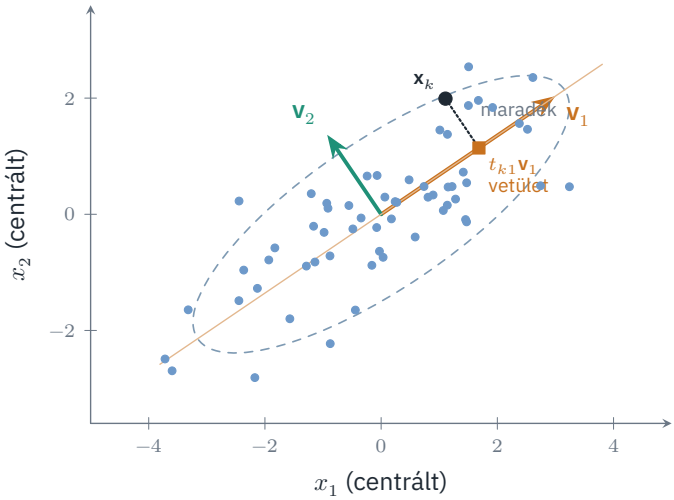
4.4.2. Főkomponens-értékek, súlyok és a megmaradt variancia

Gyűjtsük az első q sajátvektort a $\mathbf{V}_q = [\mathbf{v}_1, \ldots, \mathbf{v}_q] \in \mathbb{R}^{n \times q}$ mátrixba. Az objektumok új, q dimenziós koordinátái – a **főkomponens-értékek** (scores) – és az eredeti adatok ezekből visszaállított közelítése:

$$\mathbf{T} = \mathbf{X}_c \mathbf{V}_q \in \mathbb{R}^{N \times q}, \qquad \hat{\mathbf{X}}_c = \mathbf{T}\mathbf{V}_q^\top. \tag{4.7}$$

A $\mathbf{V}_q$ oszlopainak elemei a **súlyok** (loadings): megmutatják, hogy az egyes eredeti változók milyen mértékben járulnak hozzá egy főkomponenshez, és ezen keresztül a főkomponensek értelmezését teszik lehetővé.

Az összes változó együttes szórásnégyzete (a kovarianciamátrix nyoma) a sajátértékek összege: $\operatorname{tr}\mathbf{C} = \sum_{j=1}^{n} \lambda_j$. Az első q főkomponens által **megmagyarázott varianciahányad** és a visszaállítás átlagos négyzetes hibája:



4.9. ábra. A főkomponens-elemzés geometriája. Az első főkomponens ($\mathbf{v}_1$) a legnagyobb szórás iránya, a második ($\mathbf{v}_2$) erre merőleges. Egy pont t_{k1} főkomponens-értéke a $\mathbf{v}_1$ irányú vetületének előjeles hossza; a vetítéskor elhagyott rész a maradék.

$$\eta_q = \frac{\sum_{j=1}^{q} \lambda_j}{\sum_{j=1}^{n} \lambda_j}, \qquad \frac{1}{N-1}\|\mathbf{X}_c - \hat{\mathbf{X}}_c\|_F^2 = \sum_{j=q+1}^{n} \lambda_j. \tag{4.8}$$

A közelítés hibája tehát az **elhagyott** sajátértékek összege. (Az első kiadás e képletet tévesen a megtartott sajátértékek összegeként közölte; a megtartott sajátértékek összege a megmagyarázott variancia.) Ez az összefüggés egyben azt is mutatja, hogy a PCA a q dimenziós lineáris alterek közül a legkisebb négyzetes visszaállítási hibát adja – ez az Eckart–Young-tétel következménye (Eckart & Young, 1936).

A gyakorlatban a sajátérték-felbontás helyett numerikusan stabilabb a centrált adatmátrix **szinguláris felbontása** (SVD):

$$\mathbf{X}_c = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\top}, \qquad \lambda_j = \frac{\sigma_j^2}{N-1}, \qquad \mathbf{T} = \mathbf{U}_q\boldsymbol{\Sigma}_q. \tag{4.9}$$

A $\mathbf{V}$ jobb oldali szinguláris vektorai a főkomponens-irányok, a σ_j szinguláris értékek négyzetei a sajátértékekkel arányosak. A `scikit-learn` `PCA` osztálya ezt az utat követi.

Az algoritmus összefoglalása: főkomponens-elemzés

1. Centráljuk az adatmátrixot (oszloponként kivonjuk az átlagot); eltérő mértékegységű változók esetén standardizáljuk is (ez a korrelációs mátrixon végzett PCA).
2. Számítsuk ki a centrált mátrix szinguláris felbontását, vagy a kovarianciamátrix sajátérték-felbontását (4.4–4.6).
3. Rendezzük a sajátértékeket csökkenő sorrendbe; válasszuk meg a megtartott főkomponensek q számát (például $\eta_q \geq 0{,}9$, vagy a sajátérték-diagram „könyöke” alapján).
4. Számítsuk ki a főkomponens-értékeket ($\mathbf{T} = \mathbf{X}_c\mathbf{V}_q$), és értelmezzük a főkomponenseket a súlyok alapján.

Skálázás nélkül a nagy számok nyernek

A PCA a varianciát maximalizálja, ezért a nagy mértékegységű (nagy szórású) változók uralják. Ha a változók eltérő mértékegységűek – °C, m³/h, kW –, a PCA előtt standardizálni kell őket. Ilyenkor a kovarianciamátrix helyett lényegében a korrelációs mátrix sajátértékeit és sajátvektorait számoljuk.

4.4.3. Kézi megvalósítás és ellenőrzés

Valósítsuk meg a PCA-t NumPy-jal a Pannon üzem tíz folyamatváltozóján, és ellenőrizzük a (4.8) összefüggést!

```
Xh = plant.hourly[plant.hourly.F_betap > 1].dropna(subset=valt)
X = Xh[valt].to_numpy()
Z = (X - X.mean(0)) / X.std(0, ddof=1)          # standardizálás
N = Z.shape[0]

C = Z.T @ Z / (N - 1)                            # = korrelációs mátrix
lam, V = np.linalg.eigh(C)                       # szimmetrikus mátrix
 ↪ sajátfelbontása
order = np.argsort(lam)[::-1]
lam, V = lam[order], V[:, order]

q = 3
T = Z @ V[:, :q]                                 # főkomponens-értékek
Z_hat = T @ V[:, :q].T                           # visszaállítás
hiba = np.sum((Z - Z_hat) ** 2) / (N - 1)
print("sajátértékek:", np.round(lam, 3))
```

```
print(f"megmagyarázott varianciahányad (q={q}): {lam[:q].sum() /
↪  lam.sum():.3f}")
print(f"visszaállítási hiba: {hiba:.4f}   elhagyott sajátértékek összege:
↪  {lam[q:].sum():.4f}")
```

```
sajátértékek: [4.51  2.994 1.617 0.52  0.131 0.086 0.059 0.047 0.023 0.014]
megmagyarázott varianciahányad (q=3): 0.912
visszaállítási hiba: 0.8793   elhagyott sajátértékek összege: 0.8793
```

A két utolsó szám egyezik, ahogy a (4.8) képlet állítja. A `scikit-learn` SVD-alapú megvalósítása ugyanezt adja (a sajátvektorok előjele tetszőleges, ezért csak abszolút értékben hasonlítunk):

```
from sklearn.decomposition import PCA
pca = PCA().fit(Z)
print(np.allclose(pca.explained_variance_, lam),
      np.allclose(np.abs(pca.components_[:q]), np.abs(V[:, :q].T)))
```

```
True True
```

4.4.4. Értelmezés: a Pannon üzem főkomponensei

```
fig, ax = plt.subplots()
ax.bar(range(1, len(lam) + 1), lam, color=ab.PALETTE[0], width=0.6)
ax2 = ax.twinx()
ax2.plot(range(1, len(lam) + 1), np.cumsum(lam) / lam.sum(),
↪  color=ab.PALETTE[1], marker="o")
ax2.set_ylim(0, 1.05); ax2.grid(False); ax2.set_ylabel("kumulált hányad")
ax.set_xlabel("főkomponens"); ax.set_ylabel("sajátérték")
ax.set_xticks(range(1, len(lam) + 1))
fig.tight_layout()
```

```
suly = pd.DataFrame(V[:, :3], index=valt, columns=["PC1", "PC2", "PC3"])
suly.round(2)
```

	PC1	PC2	PC3
T_kornyezet	-0.02	-0.52	0.28
F_betap	0.42	-0.06	0.20
c_betap	-0.41	0.01	-0.31
T_betap	0.32	-0.28	-0.38
T_reaktor	0.33	-0.10	-0.51
F_hutoviz	-0.17	-0.50	-0.07
T_hutoviz	-0.02	-0.52	0.28
P_reaktor	0.40	-0.10	-0.33

	PC1	PC2	PC3
P_kevero	-0.41	0.04	-0.32
E_fogyasztas	-0.29	-0.34	-0.29

```
fig, ax = plt.subplots(1, 2, sharex=True, sharey=True)
for g, c in zip(["A", "B", "C"], ab.PALETTE):
    sel = (Xh.termek == g).to_numpy()
    ax[0].scatter(T[sel, 0], T[sel, 1], s=3, alpha=0.4, color=c, label=g)
ax[0].legend(title="termék", markerscale=4)
sc = ax[1].scatter(T[:, 0], T[:, 1], s=3, alpha=0.5, c=Xh.ido.dt.month,
 ↪  cmap="twilight")
fig.colorbar(sc, ax=ax[1], label="hónap")
for a in ax: a.set_xlabel("PC1")
ax[0].set_ylabel("PC2")
fig.tight_layout()
```

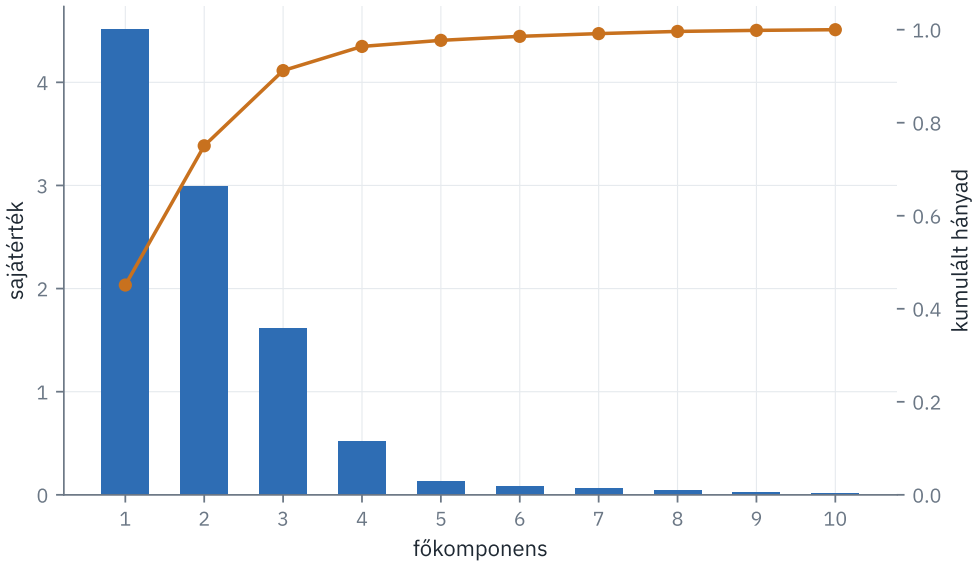
FUTÓ ESET · PANNON ÜZEM

A súlyok alapján a főkomponensek fizikailag értelmezhetők. Az egyik fő irányt a termékfüggő változók (betáplálás, betáplált koncentráció, keverőteljesítmény, reaktorhőmérséklet és -nyomás) határozzák meg – ez a *termékminőség-tengely*; egy másikat a környezeti és a hűtővíz-hőmérséklet, valamint a hűtővízáram – ez az *évszak- és hűtéstengely*. Tíz mért változó helyett így két-három, jól értelmezhető „rejtett” változóval írható le az üzem állapota. A 11. fejezetben erre építve fogjuk a hibákat felismerni: egy hibás állapot vagy szokatlan helyre kerül a főkomponensek terében, vagy a főkomponensekkel nem magyarázható maradékot hoz létre.

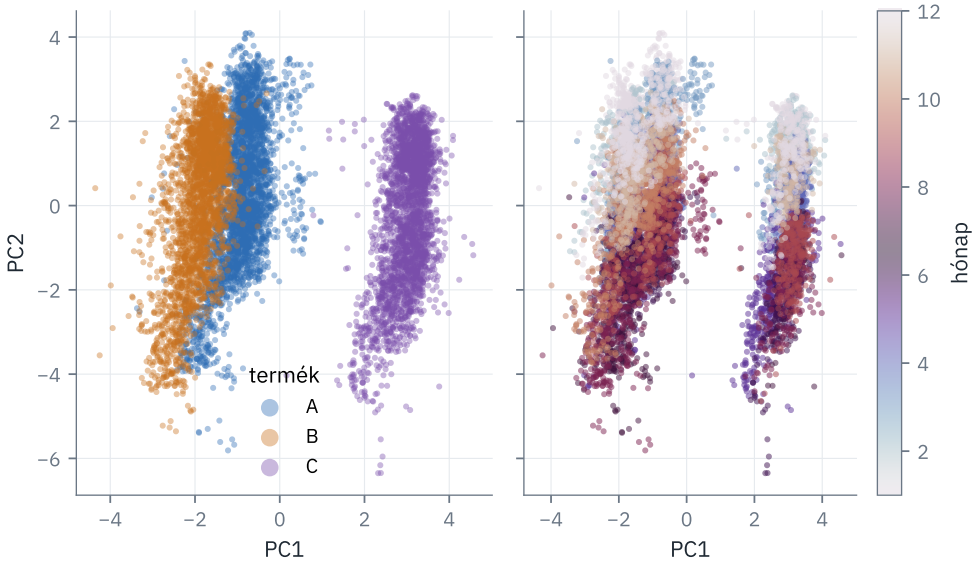
4.5. Nemlineáris vetítések

A PCA lineáris: az adatokat egy síkra (altérre) vetíti. Ha az adatok egy görbült felület mentén helyezkednek el, vagy a csoportok nem választhatók el lineárisan, nemlineáris módszerek adhatnak hűbb képet.

A **többdimenziós skálázás** (multidimensional scaling, MDS) az objektumok közötti távolságokból indul ki, és olyan alacsony dimenziós elhelyezést keres, amelyben a távolságok a lehető legjobban egyeznek az eredetiekkel. A klasszikus (metrikus) MDS (Torgerson, 1952) euklideszi távolságok esetén a PCA-val egyenértékű eredményt ad; a nemmetrikus változat csak a távolságok sorrendjét igyekszik megőrizni (Kruskal, 1964). A **Sammon-leképezés** (Sammon, 1969) a kis távolságok megőrzésére helyez nagyobb hangsúlyt; az eredeti (d^*_{ij}) és a leképezett (d_{ij}) távolságok eltérését az



4.10. ábra. A Pannon üzem folyamatváltozóinak sajátérték-diagramja (oszlopok) és a kumulált megmagyarázott varianciahányad (vonal).



4.11. ábra. A Pannon üzem órás adatainak első két főkomponense. Balra termékenként színezve: a három termék elkülönülő felhőket alkot. Jobbra a hónap szerint színezve: a termékeken belül az évszakos változás is kirajzolódik.

$$E = \frac{1}{\sum_{i<j} d^*_{ij}} \sum_{i<j} \frac{\left(d^*_{ij} - d_{ij}\right)^2}{d^*_{ij}} \tag{4.10}$$

hibafüggvénnyel méri, amelyben a $1/d^*_{ij}$ súly a közeli pontpárokat hangsúlyozza.

A **t-SNE** (t-distributed stochastic neighbor embedding) (Maaten & Hinton, 2008) a ma legelterjedtebb vizualizációs célú nemlineáris vetítés. Az eredeti térben minden pontpárhoz egy valószínűséget rendel, amely annál nagyobb, minél közelebbi szomszédok; az alacsony dimenziós térben hasonló, de nehéz farkú (Student-t) eloszlással számol, és a kettő közötti Kullback–Leibler-eltérést minimalizálja. A helyi szomszédsági szerkezetet kiválóan megőrzi, ezért a csoportok jól elkülönülnek. Használatakor azonban óvatosnak kell lenni (Wattenberg és mtsai., 2016):

- az eredmény erősen függ a *perplexitás* paramétertől (az effektív szomszédszámtól);
- a csoportok mérete és a csoportok közötti távolságok a t-SNE-ábrán **nem értelmezhetők**;
- az algoritmus véletlen kezdőpontból indul, így különböző futtatások eltérő képet adhatnak;
- tisztán véletlen adatokban is „csoportokat” mutathat.

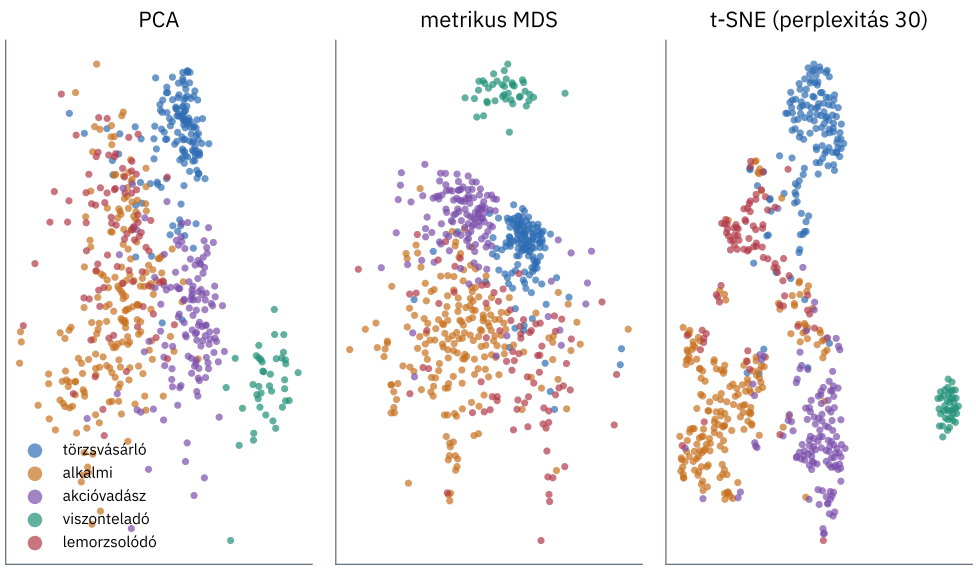
A t-SNE tehát feltáró vizualizációra való, nem a távolságokon alapuló további elemzésre. Hasonló célú, újabb módszer az UMAP (McInnes és mtsai., 2018).

FUTÓ ESET · PANNONSHOP

Vetítsük két dimenzióba a PannonShop vevőinek standardizált vásárlási jellemzőit PCA-val, metrikus MDS-sel és t-SNE-vel! Az ábrán a pontokat a szimulátor rejtett szegmensei szerint színezzük – ezt a valós elemzésben nem ismernénk, itt viszont megmutatja, melyik vetítés őrzi meg legjobban a csoportszerkezetet.

```
from sklearn.manifold import MDS, TSNE
from sklearn.preprocessing import StandardScaler
shop = ab.load_shop()
jell = ["R_nap", "F_szamla", "M_ertek", "atl_kosarertek", "atl_kosarmeret",
 ↪  "atl_mennyiseg",
        "akcios_arany", "arany_kave", "arany_gep", "arany_konyha",
         ↪  "arany_dekor", "arany_ajandek", "arany_iroszer"]
vt = vevok[jell].copy()
for c in ["F_szamla", "M_ertek", "atl_kosarertek", "atl_mennyiseg"]:
    vt[c] = np.log1p(vt[c])
vt = vt.sample(600, random_state=0)
```

```
Zv = StandardScaler().fit_transform(vt)
szegmens = shop.truth["segment"].reindex(vt.index).to_numpy()
vetuletek = {
    "PCA": PCA(n_components=2).fit_transform(Zv),
    "metrikus MDS": MDS(n_components=2, random_state=0, n_init=1,
     ↪  max_iter=300).fit_transform(Zv),
    "t-SNE (perplexitás 30)": TSNE(n_components=2, perplexity=30,
     ↪  random_state=0).fit_transform(Zv),
}
fig, ax = plt.subplots(1, 3)
szegm_nevek = ["törzsvásárló", "alkalmi", "akcióvadász", "viszonteladó",
 ↪  "lemorzsolódó"]
for a, (nev, Y) in zip(ax, vetuletek.items()):
    for s, c in zip(szegm_nevek, ab.PALETTE):
        sel = szegmens == s
        a.scatter(Y[sel, 0], Y[sel, 1], s=5, color=c, alpha=0.7, label=s)
    a.set_title(nev, fontsize=10); a.set_xticks([]); a.set_yticks([])
ax[0].legend(fontsize=7, markerscale=2.5, loc="lower left")
fig.tight_layout()
```



4.12. ábra. A PannonShop vevőinek kétdimenziós vetületei (600 véletlenszerűen kiválasztott vevő), a szimulátor rejtett szegmensei szerint színezve.

A vetületek megmutatják, hogy a szegmensek egy része (például a viszonteladók és az alkalmi vásárlók) jól elkülönül, mások – a törzsvásárlók és a később lemorzsolódók – erősen átfednek. Ez nem a módszerek hibája: a lemorzsolódó vevők

az elpártolásukig hasonlóan viselkednek, mint a törzsvásárlók, a különbség a *jövőbeli* viselkedésükben van. Ezt a kérdést a felügyelt tanulás eszközeivel, a 9. fejezetben vizsgáljuk.

Interaktív alkalmazás: Főkomponens-elemzés: irány, vetület és megmagyarázott variancia

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/pca-forgatas.html

Colab-notebook: 4. fejezet – feltáró elemzés és PCA

4.6. Összefoglalás

- A feltáró elemzés célja a kérdések megtalálása; fő eszköze a vizualizáció, mert a leíró statisztikák nagyon eltérő adatsorokat is azonosnak mutathatnak.
- Egy változó eloszlását középérték, szóródás és alak jellemzi; ferde eloszlásokra a medián és az IQR jobb mérőszám, mint az átlag és a szórás.
- A hisztogram, a magfüggvényes sűrűségbecslés, a dobozdiagram és a Q–Q ábra az eloszlás különböző tulajdonságait emeli ki.
- A korreláció lineáris (Pearson) vagy monoton (Spearman) együttmozgást mér, okozati kapcsolatot nem.
- A PCA a kovarianciamátrix sajátvektoraira vetít; a k-adik sajátérték a k-adik főkomponens varianciája, a visszaállítás hibája az elhagyott sajátértékek összege.
- A nemlineáris vetítések (MDS, Sammon, t-SNE) a helyi szerkezetet jobban megőrizhetik, de a t-SNE ábráin a távolságok és csoportméretek nem értelmezhetők.

Ellenőrző kérdések

1. Miért tanulságos Anscombe négy adatsora? Mit mutatna meg közülük egy hisztogram?
2. Mikor jobb középérték a medián, mint az átlag?
3. Mit jelent, ha egy Q–Q ábra pontjai jobbra felfelé elhajlanak az egyenestől?
4. Miért kell a PCA előtt standardizálni a Pannon üzem változóit?
5. Mekkora a visszaállítás hibája, ha $n = 5$ változó sajátértékei 2,8; 1,2; 0,6; 0,3; 0,1, és két főkomponenst tartunk meg? Mekkora a megmagyarázott varianciahányad?
6. Miért nem szabad egy t-SNE-ábrán a csoportok közötti távolságot értelmezni?

Feladatok

1. **Hisztogram és sávszélesség.** Ábrázolja a vevők átlagos kosárértékének logaritmusát hisztogramon 10, 30 és 100 intervallummal, valamint magfüggvényes becsléssel három különböző sávszélességgel! Melyik kép mutat meg két csúcsot, és mi lehet ennek az oka?
2. **Spearman vagy Pearson?** Számítsa ki a vevők vásárlásszáma (F) és vásárlási értéke (M) közötti Pearson- és Spearman-korrelációt! Miért tér el a kettő?
3. **PCA lépésről lépésre.** Végezze el a PCA-t a Pannon üzem adatain standardizálás nélkül! Melyik változó uralja az első főkomponenst, és miért?
4. **Hibák a főkomponensek terében.** Színezze a 4.11. ábra pontjait a szimulátor hibacímkéi szerint (`plant.truth["fault"]`, órás szintre összesítve)! Mely hibatípusok látszanak az első két főkomponens síkjában, és melyek nem?
5. **t-SNE érzékenysége.** Futtassa a t-SNE-t a vevőadatokon 5, 30 és 100 perplexitással, valamint három különböző véletlen kezdőértékkel! Mi marad állandó, és mi változik?

További olvasmányok

A feltáró adatelemzés klasszikusa Tukey (1977). A grafikus megjelenítés elveiről Tufte (2001) és Wilkinson (2005), az észlelési pontosságról W. S. Cleveland & McGill (1984) ír. A PCA átfogó, modern áttekintése Jolliffe & Cadima (2016); a mögötte álló lineáris algebrát Strang (2016) és Deisenroth és mtsai. (2020) tárgyalja. A t-SNE helyes használatához Wattenberg és mtsai. (2016) interaktív cikke ajánlott.

II. Mintázatok felügyelet nélkül

5. Csoportosítás

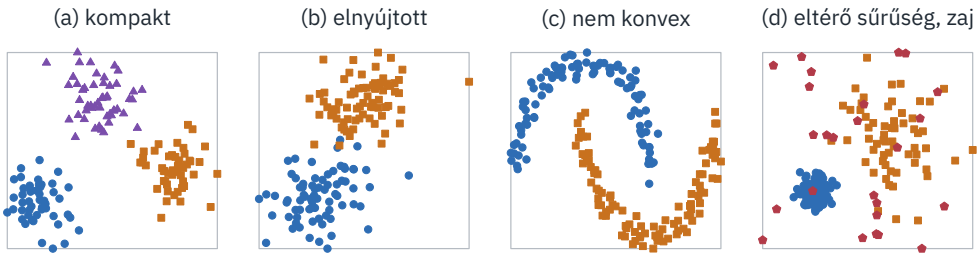
Miről szól a fejezet?

A csoportosítás (klaszterezés) az egymáshoz hasonló objektumok csoportokba rendezése előzetesen ismert csoportcímkék nélkül. A vevők szegmentálásától az üzemállapotok azonosításáig számos gyakorlati feladat mögött ez a felügyelet nélküli tanulási feladat áll. A fejezet a hasonlóság méréséből indul ki, majd bemutatja a módszerek fő családjait: a hierarchikus, a partícionáló (k-átlag, k-medoid), a fuzzy (c-átlag), a modellalapú (Gauss-keverék, EM-algoritmus), a sűrűségalapú (DBSCAN) és a neurális hálózatos (önszervező térkép) módszereket. Végül azzal foglalkozunk, hogyan ítélhető meg egy csoportosítás jósága, és hogyan választható meg a csoportok száma.
A fejezet végére az olvasó képes lesz a feladathoz és az adattípushoz illő távolságmértéket választani; a fő csoportosítási algoritmusok célfüggvényét és lépéseit felírni és megvalósítani; a csoportok számát belső validitási mutatókkal megválasztani; és egy szegmentálás eredményét üzletileg értelmezni.

5.1. Mi a csoport?

Intuitív értelemben egy csoport (klaszter) olyan objektumok halmaza, amelyek egymáshoz hasonlóbbak, mint a más csoportokba tartozókhoz. Ez a meghatározás szándékosan homályos: a „hasonlóság” és a „csoport” jelentése a feladattól függ. A 5.1. ábra négy jellegzetes helyzetet mutat. A kompakt, gömbszerű csoportokat szinte minden módszer megtalálja; az elnyújtott csoportokhoz a csoportok alakját is figyelembe vevő (például Mahalanobis-távolságot vagy kovarianciamátrixot használó) módszerek kellenek; a nem konvex, egymásba fonódó alakzatokat a sűrűségalapú vagy az egyszerű kapcsolódású hierarchikus módszerek ismerik fel; az eltérő sűrűségű csoportok és a zaj pedig külön kihívást jelentenek.

A csoportosítás eredményének formája is többféle lehet. A **kemény** (éles) csoportosításban minden objektum pontosan egy csoportba tartozik; a **fuzzy** csoportosításban minden objektum minden csoporthoz egy 0 és 1 közötti tagsági értékkel tartozik; a **valószínűségi** csoportosítás csoporttagsági valószínűségeket ad; a **hierarchikus** csoportosítás egymásba ágyazott csoportok fáját állítja elő.



5.1. ábra. Jellegzetes csoportalakok: (a) kompakt, (b) elnyújtott, (c) nem konvex, (d) eltérő sűrűségű csoportok zajjal (piros pontok). Nincs olyan módszer, amely mind a négy helyzetben jól működne.

5.2. Hasonlóság és távolság

5.2.1. Folytonos változók

Az objektumok közötti különbözőséget leggyakrabban egy $d(\mathbf{x}_i, \mathbf{x}_j)$ **távolságfüggvénnyel** mérjük, amely nemnegatív, szimmetrikus, csak azonos pontokra nulla, és teljesíti a háromszög-egyenlőtlenséget. A legismertebb a **Minkowski-távolság** családja:

$$d_p(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{l=1}^{n} |x_{il} - x_{jl}|^p \right)^{1/p}, \tag{5.1}$$

amelynek $p = 2$ esetén az euklideszi, $p = 1$ esetén a városi (Manhattan) távolság, $p \to \infty$ esetén pedig a Csebisev-távolság, $\max_l |x_{il} - x_{jl}|$ a speciális esete. A Minkowski-távolságok érzékenyek a változók skálájára, ezért használatuk előtt általában standardizálni kell (3.5. alfejezet).

A **Mahalanobis-távolság** (Mahalanobis, 1936) a változók szórását és egymással való korrelációját is figyelembe veszi:

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{C}^{-1} (\mathbf{x}_i - \mathbf{x}_j)}, \tag{5.2}$$

ahol $\mathbf{C}$ a kovarianciamátrix. Ha $\mathbf{C}$ az egységmátrix, az euklideszi távolságot kapjuk; ha diagonális, a standardizált euklideszit. A Mahalanobis-távolság az elnyújtott, korrelált csoportokat „gömbbé” alakítja, és a 11. fejezet anomália-észlelő módszereinek is alapja.

Szövegek, vásárlási kosarak és más sok, többnyire nulla értékű változóval leírt objektumok esetén a **koszinusz-hasonlóság** a szokásos választás, amely a két vektor által bezárt szög koszinusza, $\cos\theta = \mathbf{x}_i^\top \mathbf{x}_j / (\|\mathbf{x}_i\|\|\mathbf{x}_j\|)$; független a vektorok hosszától, csak az „irányukat” hasonlítja össze.

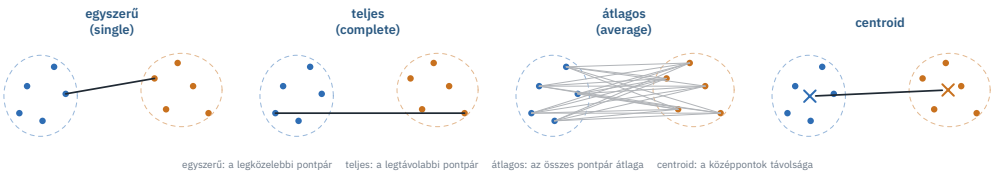
5.2.2. Kategorikus és vegyes változók

Bináris változók (például: megvásárolta-e a vevő az adott terméket) esetén két objektum összevetése egy 2×2-es gyakorisági táblázattal írható le: a az olyan változók száma, amelyek mindkét objektumban 1 értékűek, d a mindkettőben 0 értékűek, b és c pedig az eltérők. Az **egyszerű egyezési együttható** $(a+d)/(a+b+c+d)$ a közös nullákat is egyezésnek tekinti. Vásárlási kosaraknál ez félrevezető – két vevő nem lesz hasonló attól, hogy egyikük sem vett adventi koszorút –, ezért itt a **Jaccard-együttható**, $a/(a+b+c)$ a megfelelő, amely a közös hiányt figyelmen kívül hagyja.

Vegyes típusú változók esetén a **Gower-féle hasonlóság** (Gower, 1971) változónként számol egy $[0,1]$ közötti hasonlóságot – numerikus változóra $1 - |x_{il} - x_{jl}|/R_l$, ahol R_l a változó terjedelme, kategorikusra egyezés esetén 1, eltérés esetén 0 –, és ezeket (súlyozottan) átlagolja.

5.2.3. Csoportok közötti távolság

A hierarchikus módszerekhez két *csoport* távolságát is értelmezni kell (5.2. ábra). Az **egyszerű kapcsolódás** (single linkage) a legközelebbi pontpár, a **teljes kapcsolódás** (complete linkage) a legtávolabbi pontpár, az **átlagos kapcsolódás** (average linkage) az összes pontpár távolságának átlaga, a **centroid** módszer a csoportközéppontok távolsága. A **Ward-módszer** (Ward, 1963) azt a két csoportot vonja össze, amelyek összevonása a legkisebb mértékben növeli a csoporton belüli négyzetes eltérések összegét.



5.2. ábra. Csoportok közötti távolságok. Az egyszerű kapcsolódás hajlamos hosszú, láncszerű csoportok képzésére, a teljes kapcsolódás kompakt csoportokat részesít előnyben, az átlagos kapcsolódás a kettő közötti kompromisszum.

5.3. Hierarchikus módszerek

5.3.1. Az egyesítő eljárás

Az **egyesítő** (agglomeratív) hierarchikus csoportosítás minden objektumot külön csoportként indít, majd lépésenként mindig a két legközelebbi csoportot vonja össze, amíg egyetlen csoport nem marad. Az összevonások sorrendjét és az összevonáskori távolságokat a **dendrogram** (fadiagram) ábrázolja; a dendrogram egy adott magasságban történő „elvágása” adja a kívánt számú csoportot.

Az összevonás után az új csoport távolságát a többitől nem kell az eredeti pontokból újraszámolni: a fenti kapcsolódási szabályok mindegyike felírható a **Lance–Williams-formulával** (Lance & Williams, 1967). Ha az i és j csoportot összevonjuk, az új csoport távolsága egy harmadik, k csoporttól:

$$d_{(ij)k} = \alpha_i\, d_{ik} + \alpha_j\, d_{jk} + \beta\, d_{ij} + \gamma\, |d_{ik} - d_{jk}|. \tag{5.3}$$

5.1. táblázat. A Lance–Williams-formula együtthatói néhány kapcsolódási szabályra (n_i az i csoport elemszáma; α_j az α_i képletéből i és j cseréjével adódik).

Módszer	α_i	β	γ
egyszerű kapcsolódás	$1/2$	0	$-1/2$
teljes kapcsolódás	$1/2$	0	$+1/2$
átlagos kapcsolódás	$n_i/(n_i+n_j)$	0	0
Ward (négyzetes euklideszi távolságokra)	$(n_i+n_k)/(n_i+n_j+n_k)$	$-n_k/(n_i+n_j+n_k)$	0

Például egyszerű kapcsolódásra $d_{(ij)k} = \frac{1}{2}d_{ik} + \frac{1}{2}d_{jk} - \frac{1}{2}|d_{ik} - d_{jk}| = \min(d_{ik}, d_{jk})$, ahogy az egyszerű kapcsolódás definíciójából várható.

A **felosztó** (divizív) hierarchikus módszerek fordítva haladnak: egyetlen csoportból indulnak, és lépésenként a leginkább heterogén csoportot bontják ketté (L. Kaufman & Rousseeuw, 1990). Számításigényesebbek, ezért ritkábban használják őket.

Az algoritmus összefoglalása: egyesítő hierarchikus csoportosítás

1. Számítsuk ki az objektumok páronkénti távolságmátrixát; kezdetben minden objektum külön csoport.
2. Keressük meg a két legközelebbi csoportot, és vonjuk össze őket; jegyezzük fel az összevonási távolságot.
3. Frissítsük az új csoport távolságait a többitől a Lance–Williams-formulával (5.3).

4. Ismételjük a 2–3. lépést, amíg egy csoport marad; az eredményt dendrogramon ábrázoljuk, és a kívánt magasságban elvágjuk.

5.3.2. Kézi példa

Vegyük a segédcsomag hét pontját a síkon, és kövessük végig az egyszerű kapcsolódású eljárást! Először a távolságmátrixot számítjuk ki:

```
from scipy.spatial.distance import pdist, squareform
from scipy.cluster.hierarchy import linkage, dendrogram
P = ab.pontok_mini()
Dm = pd.DataFrame(squareform(pdist(P)), index=P.index, columns=P.index)
Dm.round(2)
```

	A	B	C	D	E	F	G
A	0.00	1.12	3.61	7.21	4.72	5.32	4.30
B	1.12	0.00	2.50	6.10	3.61	4.24	3.20
C	3.61	2.50	0.00	3.61	1.12	1.80	0.71
D	7.21	6.10	3.61	0.00	2.50	2.06	2.92
E	4.72	3.61	1.12	2.50	0.00	1.00	0.50
F	5.32	4.24	1.80	2.06	1.00	0.00	1.12
G	4.30	3.20	0.71	2.92	0.50	1.12	0.00

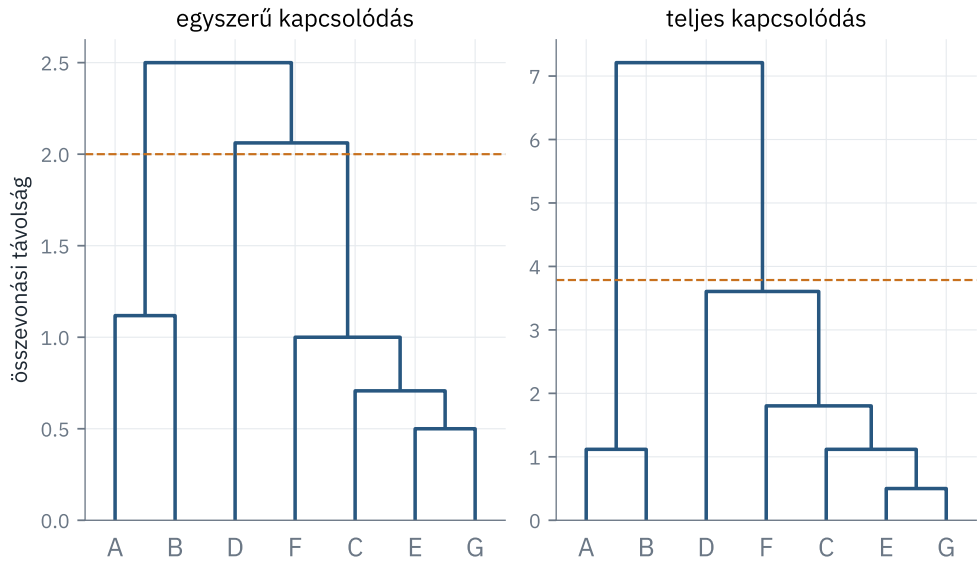
A legkisebb távolság az E és a G pont között van (0,50), ezért először ezek egyesülnek. Egyszerű kapcsolódásnál az $\{E, G\}$ csoport távolsága egy harmadik ponttól a két távolság minimuma; így a következő lépésben a C pont csatlakozik (0,71 távolságra G-től), majd az F pont (1,00 távolságra E-től). Ezután az A és B pont egyesül (1,12), a D pont az F-en keresztül kapcsolódik a négyelemű csoporthoz (2,06), végül a két megmaradt csoport a B–C pontpár távolságánál (2,50) olvad össze. Az összevonások teljes sorrendjét a `linkage` függvény adja meg: minden sor egy összevonás, az első két oszlop az összevont csoportok sorszáma (a 7-től kezdődő sorszámok a korábban létrejött csoportokat jelölik), a harmadik az összevonási távolság, a negyedik az új csoport elemszáma.

```
Z_single = linkage(P, method="single")
pd.DataFrame(Z_single, columns=["csoport_1", "csoport_2", "távolság",
↪  "elemszám"]).round(2)
```

	csoport_1	csoport_2	távolság	elemszám
0	4.0	6.0	0.50	2.0
1	2.0	7.0	0.71	3.0
2	5.0	8.0	1.00	4.0
3	0.0	1.0	1.12	2.0

	csoport_1	csoport_2	távolság	elemszám
4	3.0	9.0	2.06	5.0
5	10.0	11.0	2.50	7.0

```
fig, ax = plt.subplots(1, 2)
for a, m in zip(ax, ["single", "complete"]):
    Z = linkage(P, method=m)
    dendrogram(Z, labels=list(P.index), ax=a, color_threshold=0,
    ↪ above_threshold_color=ab.PE_BLUE)
    a.axhline(Z[-1, 2] * 0.8 if m == "single" else Z[-2, 2] * 1.05,
    ↪ color=ab.PALETTE[1], ls="--", lw=1)
    a.set_title("egyszerű kapcsolódás" if m == "single" else "teljes
    ↪ kapcsolódás")
ax[0].set_ylabel("összevonási távolság")
fig.tight_layout()
```



5.3. ábra. A hét pont dendrogramja egyszerű (balra) és teljes kapcsolódással (jobbra). A szaggatott vonal két csoportot eredményező elvágást jelöl.

A két dendrogram ugyanazokat a természetes csoportokat ($\{A, B\}$ és a többi pont) mutatja, de az összevonási távolságok és a D pont helyzete eltér: a teljes kapcsolódás a távoli D pontot csak nagy távolságnál olvasztja be, ami a kiugró pontok felismerésében segíthet.

5.4. Partícionáló módszerek

5.4.1. A k-átlag algoritmus

A **k-átlag** (k-means) módszer az adatokat előre megadott számú, c csoportra osztja úgy, hogy a csoportokon belüli négyzetes eltérések összege minimális legyen:

$$J = \sum_{i=1}^{c} \sum_{\mathbf{x}_k \in G_i} \|\mathbf{x}_k - \mathbf{v}_i\|^2, \tag{5.4}$$

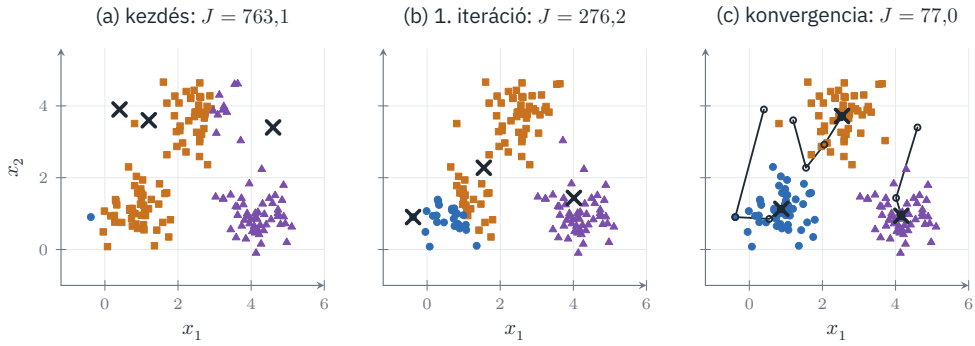
ahol G_i az i-edik csoport, $\mathbf{v}_i$ a középpontja (centroidja). A feladat globális optimumának megtalálása nehéz (NP-nehéz), ezért egy egyszerű iteratív eljárást, a Lloyd-algoritmust használjuk (Lloyd, 1982; MacQueen, 1967), amely két lépést ismételget (5.4. ábra):

1. **Hozzárendelés**: minden objektumot a legközelebbi középponthoz rendelünk.
2. **Frissítés**: minden középpontot a hozzá rendelt objektumok átlagára állítunk, $\mathbf{v}_i = \frac{1}{|G_i|} \sum_{\mathbf{x}_k \in G_i} \mathbf{x}_k$.

Mindkét lépés csökkenti (pontosabban nem növeli) J-t: a hozzárendelés rögzített középpontok mellett, a frissítés rögzített hozzárendelés mellett optimális (a négyzetes eltérések összegét az átlag minimalizálja). Mivel a lehetséges hozzárendelések száma véges, az algoritmus véges sok lépésben megáll – de csak egy **lokális** minimumban, amely a kezdő középpontoktól függ. Ezért az algoritmust több kezdőpontból érdemes futtatni, a kezdőpontokat pedig a **k-means++** eljárással célszerű választani, amely az első középpont után a további középpontokat a már kiválasztottaktól vett távolság négyzetével arányos valószínűséggel sorsolja (Arthur & Vassilvitskii, 2007).

A k-átlag gyors (lépésenként $O(N c\, n)$), egyszerű, és nagy adathalmazokra is jól skálázódik. Korlátai a célfüggvényéből fakadnak: gömbszerű, hasonló méretű csoportokat feltételez, érzékeny a kiugró értékekre (az átlag miatt), és a csoportok számát előre meg kell adni.

```
def kmeans_sajat(X, c, max_iter=100, seed=0):
    """A Lloyd-algoritmus NumPy-megvalósítása (k-means++ kezdéssel)."""
    rng = np.random.default_rng(seed)
    V = [X[rng.integers(len(X))]]
    for _ in range(1, c):                              # k-means++ kezdőpontok
        d2 = np.min(((X[:, None, :] - np.array(V)[None]) ** 2).sum(-1),
↪   axis=1)
```



5.4. ábra. A k-átlag algoritmus működése szándékosan rossz kezdő középpontokból (×). (a) Az első hozzárendelés, (b) az első frissítés utáni állapot, (c) a konvergencia; a vonalak a középpontok útját mutatják. A J célfüggvény minden lépésben csökken.

```
        V.append(X[rng.choice(len(X), p=d2 / d2.sum())])
    V = np.array(V)
    for it in range(max_iter):
        D2 = ((X[:, None, :] - V[None]) ** 2).sum(-1)  # N x c négyzetes
↪   távolságok
        lab = D2.argmin(1)                              # hozzárendelés
        V_uj = np.array([X[lab == i].mean(0) for i in range(c)])  # frissítés
        if np.allclose(V_uj, V):
            break
        V = V_uj
    J = D2[np.arange(len(X)), lab].sum()
    return lab, V, J, it + 1

lab, V, J, it = kmeans_sajat(P.to_numpy(), c=2)
print("címkék:", dict(zip(P.index, lab.tolist())), f"  J = {J:.2f},
↪  iterációk: {it}")
```

```
címkék: {'A': 1, 'B': 1, 'C': 0, 'D': 0, 'E': 0, 'F': 0, 'G': 0}   J = 8.52,
↪  iterációk: 2
```

5.4.2. A k-medoid algoritmus

A k**-medoid** módszer (L. Kaufman & Rousseeuw, 1990) a középpontokat a csoport egy tényleges objektumára, a **medoidra** korlátozza – arra az objektumra, amelynek a csoport többi tagjától vett távolságösszege minimális. Ennek két előnye van: a módszer tetszőleges távolság- vagy különbözőségmátrixszal működik (nem kell

átlagot számolni, így kategorikus vagy vegyes adatokra is alkalmazható), és kevésbé érzékeny a kiugró értékekre. A klasszikus PAM (Partitioning Around Medoids) algoritmus a medoidokat cserékkel javítja, amíg a teljes távolságösszeg csökken; ára a nagyobb számításigény.

5.5. Fuzzy csoportosítás

A valóságban a csoportok közötti határ gyakran nem éles: egy vevő viselkedése félúton lehet az alkalmi vásárló és a törzsvásárló között, egy üzemállapot átmeneti lehet két működési mód között. A **fuzzy csoportosítás** ezt tagsági értékekkel fejezi ki (5.5. ábra).

5.5.1. A fuzzy partíció és a c-átlag célfüggvény

Legyen $\mathbf{U} = [\mu_{ik}] \in [0,1]^{c \times N}$ a **fuzzy partíciós mátrix**, ahol μ_{ik} a k-adik objektum tagsági értéke az i-edik csoportban. A fuzzy partíció feltételei: minden tagsági érték a $[0,1]$ intervallumba esik, minden objektum tagsági értékeinek összege 1 ($\sum_{i=1}^{c} \mu_{ik} = 1$), és egyik csoport sem üres, de nem is tartalmaz minden objektumot teljes tagsággal ($0 < \sum_{k=1}^{N} \mu_{ik} < N$). A **fuzzy c-átlag** (FCM) algoritmus Dunn (1973) ötletén alapul, általános alakját Bezdek (1981) dolgozta ki. Célfüggvénye a k-átlag (5.4) célfüggvényének tagsági értékekkel súlyozott változata:

$$J_m(\mathbf{U}, \mathbf{V}) = \sum_{i=1}^{c} \sum_{k=1}^{N} \mu_{ik}^{m} \, D_{ik}^{2}, \qquad D_{ik}^{2} = (\mathbf{x}_k - \mathbf{v}_i)^{\top} \mathbf{A} (\mathbf{x}_k - \mathbf{v}_i), \tag{5.5}$$

ahol $m > 1$ a **fuzzy kitevő**, $\mathbf{A}$ pedig a távolságot meghatározó szimmetrikus, pozitív definit mátrix ($\mathbf{A} = \mathbf{I}$ esetén euklideszi távolság). Minél nagyobb m, annál „elmosódottabbak” a csoportok; $m \to 1$ esetén a k-átlag kemény partícióját kapjuk vissza. A gyakorlatban leggyakrabban $m = 2$-t használnak.

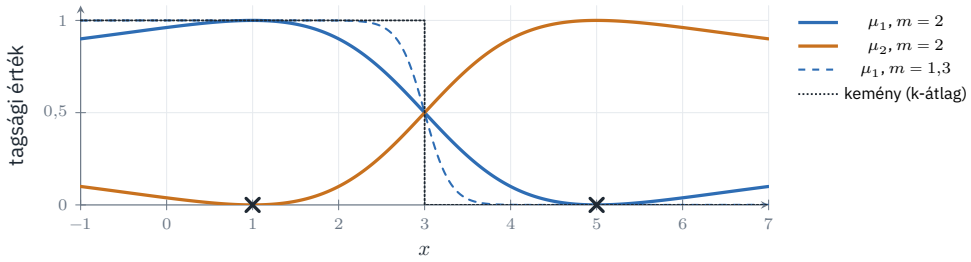
5.5.2. Az optimum feltételei

A célfüggvényt a $\sum_i \mu_{ik} = 1$ feltételek mellett minimalizáljuk. Rögzített $\mathbf{V}$ mellett minden k objektumra külön Lagrange-függvényt írunk fel:

$$\mathcal{L}_k = \sum_{i=1}^{c} \mu_{ik}^m D_{ik}^2 - \lambda_k \Big(\sum_{i=1}^{c} \mu_{ik} - 1 \Big), \qquad \frac{\partial \mathcal{L}_k}{\partial \mu_{ik}} = m\, \mu_{ik}^{m-1} D_{ik}^2 - \lambda_k = 0.$$

Ebből $\mu_{ik} = \left(\lambda_k / (m D_{ik}^2)\right)^{1/(m-1)}$; a λ_k-t az összegfeltételből kifejezve kapjuk a tagsági értékek frissítési szabályát. A középpontokra a $\partial J_m / \partial \mathbf{v}_i = \mathbf{0}$ feltétel adja a súlyozott átlagot:

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{c} \left(D_{ik} / D_{jk} \right)^{2/(m-1)}}, \qquad \mathbf{v}_i = \frac{\sum_{k=1}^{N} \mu_{ik}^m \mathbf{x}_k}{\sum_{k=1}^{N} \mu_{ik}^m}. \tag{5.6}$$



5.5. ábra. Fuzzy tagsági függvények egy dimenzióban két csoportközéppont (×, $v_1 = 1$, $v_2 = 5$) esetén. $m = 2$ mellett a tagság fokozatosan változik, kisebb fuzzy kitevőnél ($m = 1{,}3$) élesebbé válik, $m \to 1$ határesetben a k-átlag kemény hozzárendelését kapjuk.

Az algoritmus összefoglalása: fuzzy c-átlag

1. Válasszuk meg a csoportok számát (c), a fuzzy kitevőt (m, tipikusan 2), a távolságnormát (**A**) és a leállási küszöböt (ε); inicializáljuk **U**-t véletlenszerűen a partíciós feltételeknek megfelelően.
2. Számítsuk ki a középpontokat (5.6 jobb oldala).
3. Számítsuk ki a távolságokat (D_{ik}) és az új tagsági értékeket (5.6 bal oldala).
4. Ismételjük a 2–3. lépést, amíg $\max_{ik} |\mu_{ik}^{(t)} - \mu_{ik}^{(t-1)}| < \varepsilon$.

Az FCM euklideszi normával a k-átlaghoz hasonlóan gömbszerű csoportokat keres. A **Gustafson–Kessel-algoritmus** (Gustafson & Kessel, 1979) minden csoporthoz saját, adaptív normát (egy fuzzy kovarianciamátrixból származtatott $\mathbf{A}_i$ mátrixot) rendel, így elnyújtott, hipersík jellegű csoportokat is felismer; ez különösen

hasznos folyamatadatok lokálisan lineáris modelljeinek azonosításában. A fuzzy csoportosítás további változatait és rendszeridentifikációs alkalmazásait Abonyi & Feil (2007) tárgyalja részletesen.

```
def fcm(X, c, m=2.0, max_iter=200, eps=1e-6, seed=0):
    """Fuzzy c-átlag (euklideszi norma). Visszaad: U (c x N), V (c x n),
    ↪  J."""
    rng = np.random.default_rng(seed)
    U = rng.random((c, len(X)))
    U /= U.sum(0)
    for _ in range(max_iter):
        W = U ** m
        V = W @ X / W.sum(1, keepdims=True)                    # középpontok
        D = np.sqrt(((X[None, :, :] - V[:, None, :]) ** 2).sum(-1)) + 1e-12
        U_uj = 1.0 / ((D[:, None, :] / D[None, :, :]) ** (2 / (m -
↪  1))).sum(1)
        if np.max(np.abs(U_uj - U)) < eps:
            U = U_uj
            break
        U = U_uj
    J = ((U ** m) * D ** 2).sum()
    return U, V, J

U, V, J = fcm(P.to_numpy(), c=2)
pd.DataFrame(U.T, index=P.index, columns=["$\mu_1$", "$\mu_2$"]).round(3)
```

	μ_1	μ_2
A	0.984	0.016
B	0.986	0.014
C	0.195	0.805
D	0.098	0.902
E	0.012	0.988
F	0.015	0.985
G	0.041	0.959

A tagsági értékek a kézi példában is informatívak: az A és B pont egyértelműen az egyik, E, F és G egyértelműen a másik csoportba tartozik, a két csoport közé eső C pont tagsága viszont a legkevésbé határozott, és a távoli D pont tagsága is elmarad a csoportmag pontjaiétól. A fuzzy csoportosítás így a bizonytalan hozzárendeléseket is jelzi.

Fuzzy partíciók jóságának mérésére a **partíciós együttható**, $\mathrm{PC} = \frac{1}{N}\sum_i \sum_k \mu_{ik}^2$ (értéke $1/c$ és 1 közé esik, a nagyobb élesebb partíciót jelez), valamint a csoportok kompaktságát és szeparációját együtt mérő **Xie–Beni-index** (Xie & Beni, 1991) használható.

5.6. Modellalapú csoportosítás: Gauss-keverék modellek

A **modellalapú** csoportosítás azt feltételezi, hogy az adatok egy valószínűségi eloszlások keverékéből származnak, és minden csoport egy keverékkomponensnek felel meg. A leggyakoribb a **Gauss-keverék modell** (Gaussian mixture model, GMM):

$$p(\mathbf{x}) = \sum_{i=1}^{c} \pi_i \, \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \qquad \pi_i \geq 0, \;\; \sum_{i=1}^{c} \pi_i = 1, \tag{5.7}$$

ahol π_i a komponensek súlya, $\boldsymbol{\mu}_i$ a várható értéke, $\boldsymbol{\Sigma}_i$ a kovarianciamátrixa (a többdimenziós normális eloszlást a C. függelék ismerteti). A paramétereket maximum-likelihood módszerrel becsüljük; mivel a csoporttagság nem ismert, ezt az **EM-algoritmus** (Expectation–Maximization) (A. P. Dempster és mtsai., 1977) végzi, amely két lépést ismételget:

- **E-lépés**: a jelenlegi paraméterekkel kiszámítjuk, mekkora valószínűséggel származik az egyes objektumok a komponensekből (ezek a *felelősségek*):

$$\gamma_{ik} = \frac{\pi_i \, \mathcal{N}(\mathbf{x}_k \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{j=1}^{c} \pi_j \, \mathcal{N}(\mathbf{x}_k \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{5.8}$$

- **M-lépés**: a felelősségekkel súlyozva újrabecsüljük a paramétereket:

$$N_i = \sum_k \gamma_{ik}, \quad \pi_i = \frac{N_i}{N}, \quad \boldsymbol{\mu}_i = \frac{1}{N_i} \sum_k \gamma_{ik} \mathbf{x}_k, \quad \boldsymbol{\Sigma}_i = \frac{1}{N_i} \sum_k \gamma_{ik} (\mathbf{x}_k - \boldsymbol{\mu}_i)(\mathbf{x}_k \tag{5.9}$$

Az EM-algoritmus minden iterációban növeli (nem csökkenti) a log-likelihoodot, és egy lokális maximumhoz konvergál. Közeli rokonságban áll a k-átlaggal és az FCM-mel: a felelősségek „puha” tagsági értékek, a középpontok súlyozott átlagok. Ha minden kovarianciamátrixot $\sigma^2\mathbf{I}$-re rögzítünk és $\sigma \to 0$, a k-átlagot kapjuk vissza.

> **Javítás az első kiadáshoz képest**
>
> Az első kiadás az EM-algoritmust a WEKA programcsomag megvalósításának tulajdonságai alapján jellemezte (attribútumok függetlenségének feltevése, a csoportszám automatikus meghatározása). Ezek nem az EM-algoritmus, hanem egy konkrét implementáció beállításai: az általános GMM tetszőleges kovarianciamátrixot megenged, a csoportok számát pedig

> modellválasztási kritériummal kell megválasztani.

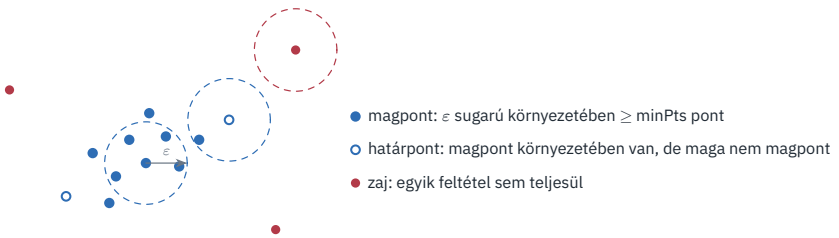
A modellalapú megközelítés előnye, hogy a csoportok száma statisztikailag megalapozottan választható. A **bayesi információs kritérium**, BIC $= -2 \ln \hat{L} + p \ln N$ (ahol $\hat{L}$ a maximális likelihood, p a becsült paraméterek száma) a jobb illeszkedést a paraméterek számával bünteti; a legkisebb BIC-értékű modellt választjuk.

5.7. Sűrűségalapú csoportosítás: DBSCAN

A **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise) (Ester és mtsai., 1996) a csoportokat a sűrű tartományok összefüggő részeiként értelmezi, amelyeket ritka tartományok választanak el egymástól. Két paramétere a környezet sugara (ε) és a „sűrűnek” tekintett környezethez szükséges legkisebb pontszám (minPts). A pontokat három típusba sorolja (5.6. ábra):

- **magpont**: az ε sugarú környezetében legalább minPts pont van (önmagát is beleértve);
- **határpont**: nem magpont, de egy magpont környezetébe esik;
- **zajpont**: egyik sem.

Két magpont ugyanabba a csoportba kerül, ha egymás környezetében lánccal elérhetők; a határpontok a szomszédos magpont csoportjához csatlakoznak, a zajpontok egyik csoportba sem. A DBSCAN tetszőleges alakú csoportokat felismer, a csoportok számát nem kell előre megadni, és a zajt külön kezeli. Gyengéje, hogy eltérő sűrűségű csoportok esetén nincs mindegyiknek megfelelő ε; ezt a problémát a hierarchikus változat, a **HDBSCAN** (Campello és mtsai., 2013) oldja meg, amely különböző sűrűségszinteken vizsgálja a csoportszerkezetet.



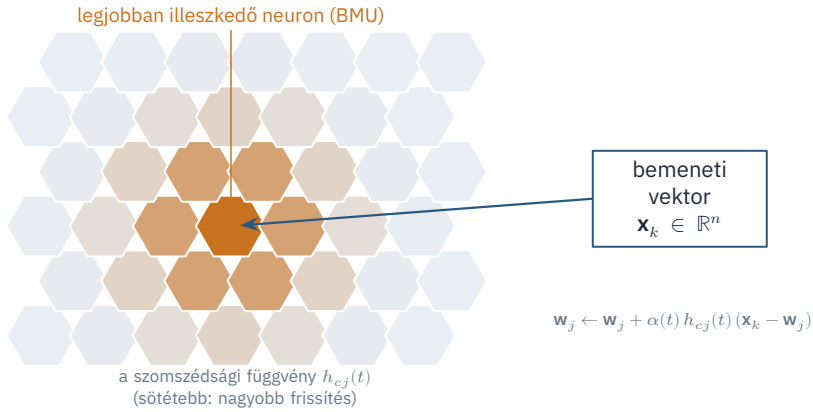
5.6. ábra. A DBSCAN ponttípusai minPts = 4 esetén: a magpontok (teli kék) környezetében elég sok pont van, a határpontok (üres kék) egy magpont környezetébe esnek, a zajpontok (piros) egyik csoporthoz sem tartoznak.

5.8. Önszerveződő térképek

Az **önszerveződő térkép** (self-organizing map, SOM) (Kohonen, 1982, 2001) egy neurális hálózat, amely a sokdimenziós adatokat egy kétdimenziós (négyzetes vagy hatszöges) neuronrácsra képezi le úgy, hogy a hasonló objektumok szomszédos neuronokra kerüljenek (5.7. ábra). Minden j neuronhoz egy $\mathbf{w}_j \in \mathbb{R}^n$ **prototípusvektor** tartozik. A tanítás során véletlen sorrendben vesszük az objektumokat; mindegyikhez megkeressük a **legjobban illeszkedő neuront** (best matching unit, BMU), $c = \arg\min_j \|\mathbf{x}_k - \mathbf{w}_j\|$, majd a BMU-t és rácsbeli szomszédait az objektum felé mozdítjuk:

$$\mathbf{w}_j \leftarrow \mathbf{w}_j + \alpha(t)\, h_{cj}(t)\, (\mathbf{x}_k - \mathbf{w}_j), \qquad h_{cj}(t) = \exp\left(-\frac{\|\mathbf{r}_c - \mathbf{r}_j\|^2}{2\sigma^2(t)}\right), \tag{5.10}$$

ahol $\mathbf{r}_j$ a j neuron rácsbeli helye, $\alpha(t)$ a tanulási ráta, $\sigma(t)$ a szomszédsági sugár; mindkettő a tanítás során csökken. A tanítás végén a rács a bemeneti tér „lenyomata”: a szomszédos neuronok prototípusai hasonlók. A csoportok határait az **U-mátrix** teszi láthatóvá, amely minden neuronnál a szomszédos prototípusoktól vett átlagos távolságot ábrázolja: a nagy értékek (gerincek) a csoportok közötti határokat jelzik.



5.7. ábra. Az önszerveződő térkép tanítása. Az aktuális bemeneti vektorhoz legközelebbi neuron (BMU) és rácsbeli szomszédai a bemenet felé mozdulnak; a frissítés mértéke a BMU-tól távolodva csökken.

5.9. A csoportosítás értékelése

Mivel nincs „helyes válasz”, a csoportosítás értékelése különleges kihívás. Három megközelítést különböztetünk meg.

Belső mutatók csak az adatokat és a csoportosítást használják. A **sziluett-együttható** (Rousseeuw, 1987) minden objektumra összeveti a saját csoportján belüli átlagos távolságot (a_k) a legközelebbi másik csoport pontjaitól vett átlagos távolsággal (b_k):

$$s_k = \frac{b_k - a_k}{\max(a_k, b_k)} \in [-1, 1], \qquad \bar{s} = \frac{1}{N}\sum_{k=1}^{N} s_k. \tag{5.11}$$

Az 1-hez közeli érték jól elhelyezett, a 0 körüli két csoport határán lévő, a negatív érték valószínűleg rossz csoportba sorolt objektumot jelez. A **Davies–Bouldin-index** (Davies & Bouldin, 1979) a csoportpárok átlagos „hasonlóságát” méri (a kisebb a jobb); a **Caliński–Harabasz-index** (Caliński & Harabasz, 1974) a csoportok közötti és a csoportokon belüli szóródás arányát (a nagyobb a jobb).

Külső mutatók a csoportosítást egy ismert besorolással vetik össze – ez a valóságban ritkán áll rendelkezésre, de a módszerek összehasonlításában és a mi szimulált adatainkon hasznos. A **korrigált Rand-index** (adjusted Rand index, ARI) (Hubert & Arabie, 1985) azt méri, hogy az objektumpárok mekkora hányadát kezeli a két felosztás egyformán (mindkettő ugyanabba vagy mindkettő különböző csoportba sorolja), a véletlenszerű egyezés várható értékével korrigálva: értéke véletlen felosztásnál 0 körüli, tökéletes egyezésnél 1.

A **stabilitás** azt vizsgálja, mennyire változik a csoportosítás, ha az adatokat kissé megváltoztatjuk (újramintavételezés, zaj hozzáadása) – egy valódi szerkezet stabil, a véletlen műterméke nem.

> **A csoportok száma nem „adat”**
>
> A belső mutatók segítenek a csoportok számának megválasztásában, de gyakran nem adnak egyértelmű választ, és különböző mutatók különböző számot javasolhatnak. A végső döntés a felhasználás céljától is függ: egy marketingkampányhoz 4–6 jól megkülönböztethető szegmens lehet ideális akkor is, ha a mutatók 3-at vagy 8-at sugallnak.

5.10. Gyakorlat: vevőszegmentálás és üzemállapotok

5.10.1. A PannonShop vevőinek szegmentálása

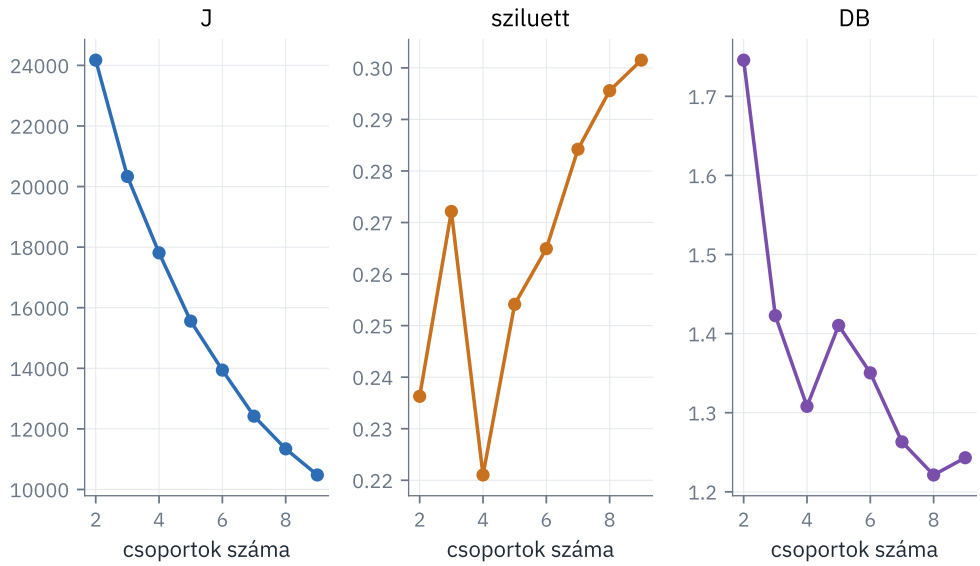
A 3. fejezetben előállított vevőszintű táblából indulunk. A ferde eloszlású változókat logaritmussal transzformáljuk, majd standardizálunk – a távolságalapú módszerek ezt igénylik.

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score, davies_bouldin_score,
↪  calinski_harabasz_score, adjusted_rand_score

shop = ab.load_shop()
vevok = ab.customer_table()
jell = ["R_nap", "F_szamla", "M_ertek", "atl_kosarertek", "atl_kosarmeret",
↪  "atl_mennyiseg",
        "akcios_arany", "arany_kave", "arany_gep", "arany_konyha",
        ↪  "arany_dekor", "arany_ajandek", "arany_iroszer"]
Xv = vevok[jell].copy()
for c in ["F_szamla", "M_ertek", "atl_kosarertek", "atl_mennyiseg"]:
    Xv[c] = np.log1p(Xv[c])
Zv = StandardScaler().fit_transform(Xv)
valodi = (shop.truth["segment"].reindex(vevok.index)            # csak
↪  értékeléshez!
          .fillna("egyéb").to_numpy())                          # (néhány új,
↪  egyszeri vevő)
```

Először a csoportok számát választjuk meg a belső mutatók alapján:

```
ks = range(2, 10)
eredm = []
for k in ks:
    km = KMeans(n_clusters=k, n_init=10, random_state=0).fit(Zv)
    eredm.append({"k": k, "J": km.inertia_, "sziluett":
↪  silhouette_score(Zv, km.labels_),
                  "DB": davies_bouldin_score(Zv, km.labels_), "ARI":
                  ↪  adjusted_rand_score(valodi, km.labels_)})
eredm = pd.DataFrame(eredm).set_index("k")
fig, ax = plt.subplots(1, 3)
for a, col, c in zip(ax, ["J", "sziluett", "DB"], ab.PALETTE):
    a.plot(eredm.index, eredm[col], marker="o", color=c); a.set_title(col);
↪  a.set_xlabel("csoportok száma")
fig.tight_layout()
```



5.8. ábra. A csoportok számának megválasztása a PannonShop vevőinél: a k-átlag célfüggvénye (könyökdiagram), a sziluett-együttható és a Davies–Bouldin-index a csoportok számának függvényében.

```
eredm.round(3)
```

k	J	sziluett	DB	ARI
2	24172.716	0.236	1.746	0.306
3	20333.543	0.272	1.423	0.362
4	17810.954	0.221	1.308	0.564
5	15557.892	0.254	1.410	0.551
6	13939.788	0.265	1.350	0.577
7	12419.768	0.284	1.263	0.498
8	11340.694	0.296	1.221	0.481
9	10478.617	0.301	1.243	0.506

A görbék a gyakorlatban jellemző képet mutatják: a könyökdiagramon nincs éles törés, a sziluett-együttható a vizsgált tartományban lassan nő (a legnagyobb érték 9 csoportnál adódik, de a különbségek kicsik), a Davies–Bouldin-index pedig lassan csökken. A belső mutatók tehát nem jelölnek ki egyértelmű csoportszámot. Az ARI oszlop (amelyet a valóságban nem ismernénk) azt mutatja, hogy a rejtett szegmensekkel 4–6 csoport esetén a legnagyobb az egyezés. Az értelmezhetőség és a marketingfelhasználás szempontjából is kezelhető 5 csoporttal dolgozunk tovább, és összehasonlítjuk a módszereket:

```
c = 5
modellek = {
    "k-átlag": KMeans(n_clusters=c, n_init=20,
    ↪ random_state=0).fit_predict(Zv),
    "hierarchikus (Ward)": AgglomerativeClustering(n_clusters=c,
    ↪ linkage="ward").fit_predict(Zv),
    "hierarchikus (átlagos)": AgglomerativeClustering(n_clusters=c,
    ↪ linkage="average").fit_predict(Zv),
    "Gauss-keverék": GaussianMixture(n_components=c,
    ↪ covariance_type="full", n_init=3,
    ↪ random_state=0).fit(Zv).predict(Zv),
    "fuzzy c-átlag (m=1,3)": fcm(Zv, c=c, m=1.3, seed=1)[0].argmax(0),
}
pd.DataFrame({nev: {"sziluett": silhouette_score(Zv, l), "Davies–Bouldin":
↪ davies_bouldin_score(Zv, l),
                    "ARI (rejtett szegmensek)": adjusted_rand_score(valodi,
                    ↪ l),
                    "legkisebb csoport": np.bincount(l).min()} for nev, l in
                    ↪ modellek.items()}).T.round(3)
```

	sziluett	Davies–Bouldin	ARI (rejtett szegmensek)	legkisebb csoport
k-átlag	0.254	1.410	0.551	153.0
hierarchikus (Ward)	0.234	1.463	0.466	163.0
hierarchikus (átlagos)	0.276	1.000	0.110	8.0
Gauss-keverék	0.171	2.039	0.517	163.0
fuzzy c-átlag (m=1,3)	0.214	1.592	0.480	190.0

FUTÓ ESET · PANNONSHOP

A k-átlag, a Ward-módszer és a Gauss-keverék hasonló minőségű felosztást ad, az átlagos kapcsolódású hierarchikus módszer viszont egy-két nagyon kicsi csoportot választ le (a kiugró vevőket), és a nagy tömeget egyben hagyja – ez a kapcsolódási szabály ismert tulajdonsága. A szegmentálás értéke azonban nem a mutatókon, hanem az értelmezhetőségen múlik. Nézzük meg a k-átlag csoportjainak profilját!

```
lab = modellek["k-átlag"]
profil = vevok.assign(csoport=lab).groupby("csoport").agg(
    vevok=("R_nap", "size"), R_median=("R_nap", "median"),
    F_median=("F_szamla", "median"),
    M_median_eFt=("M_ertek", lambda s: s.median() / 1e3),
    kosar_median_eFt=("atl_kosarertek", lambda s: s.median() / 1e3),
    mennyiseg=("atl_mennyiseg", "median"), akcios=("akcios_arany", "mean"),
    gep=("arany_gep", "mean"))
profil.round(2)
```

csoport	vevok	R_median	F_median	M_median_eFt	kosar_median_eFt	mennyiseg	akcios	gep
0	153	28.0	7.0	7093.86	925.72	20.21	0.11	0.28
1	560	329.5	2.0	32.61	18.46	1.50	0.12	0.05
2	767	23.0	11.0	446.94	41.08	1.65	0.11	0.18
3	646	157.5	4.0	440.27	95.33	2.52	0.46	0.58
4	268	426.5	2.0	37.07	18.84	1.58	0.10	0.07

```
pd.crosstab(pd.Series(lab, name="k-átlag csoport"), pd.Series(valodi,
↪  name="rejtett szegmens"))
```

k-átlag csoport \ rejtett szegmens	akcióvadász	alkalmi	egyéb	lemorzsolódó	törzsvásárló	viszonteladó
0	0	0	15	0	0	138
1	13	506	0	38	3	0
2	47	101	2	29	588	0
3	485	91	8	43	19	0
4	5	52	0	189	22	0

A profilok jól értelmezhetők: elkülönül a kevés, de nagyon nagy mennyiségben vásárló viszonteladói csoport; egy gyakran vásárló, közepes kosárértékű, rendszeres csoport; egy akciós időszakokra koncentráló csoport; egy régen vásárolt, ritkán visszatérő csoport; és egy alkalmi, kis értékű csoport. A kereszttábla szerint a rejtett szegmensek többsége egy-egy csoportban koncentrálódik, a törzsvásárlók és a később lemorzsolódók azonban részben keveredtek – ahogy a 4. fejezet vetületei is mutatták. A szegmensek elnevezése és a hozzájuk rendelt marketingakciók a 14. fejezet témája.

A fuzzy csoportosítás itt különösen hasznos: a vevők egy részének tagsági értékei között nincs domináns, ezek a „határeseti” vevők, akiknél egy kemény besorolás félrevezető lenne.

```
U, V, J = fcm(Zv, c=5, m=1.3, seed=1)
max_tagsag = U.max(0)
print(f"partíciós együttható: {(U ** 2).sum() / U.shape[1]:.3f}")
print(f"vevők, akiknek legnagyobb tagsági értéke 0,5 alatt van:
↪  {(max_tagsag < 0.5).mean():.1%}")
```

```
partíciós együttható: 0.738
vevők, akiknek legnagyobb tagsági értéke 0,5 alatt van: 11.5%
```

A fuzzy kitevő sokdimenziós adatokon

A szokásos $m = 2$ fuzzy kitevővel az FCM ezen a 13 dimenziós adaton szinte egyenletes tagsági értékeket ad, és a középpontok egy része összeolvad (az

egyik csoport a legnagyobb tagság szerint üres marad). Ennek oka, hogy sok dimenzióban a pontok távolságai a középpontoktól egymáshoz nagyon hasonlóvá válnak, így a (5.6) szabály szerinti tagsági értékek kiegyenlítődnek (Winkler és mtsai., 2011). A gyakorlati megoldás a kisebb fuzzy kitevő (itt $m = 1{,}3$), a dimenziószám csökkentése (például főkomponensekre), vagy a partíciós együttható ellenőrzése: ha az értéke $1/c$ közelében van, a partíció használhatatlanul elmosódott.

5.10.2. A Pannon üzem üzemállapotai

A mérnöki esetben arra vagyunk kíváncsiak, milyen jellegzetes működési állapotai vannak az üzemnek. A 4. fejezet főkomponens-értékeiből indulunk, és Gauss-keverék modellt illesztünk, a komponensek számát a BIC alapján választva.

```
from sklearn.decomposition import PCA
plant = ab.load_plant()
valt = ["T_kornyezet", "F_betap", "c_betap", "T_betap", "T_reaktor",
 ↪  "F_hutoviz",
        "T_hutoviz", "P_reaktor", "P_kevero", "E_fogyasztas"]
Hh = plant.hourly[plant.hourly.F_betap >
 ↪  1].dropna(subset=valt).reset_index(drop=True)
Zh = StandardScaler().fit_transform(Hh[valt])
Th = PCA(n_components=3).fit_transform(Zh)
bic = {k: GaussianMixture(n_components=k, covariance_type="full",
 ↪  random_state=0).fit(Th).bic(Th) for k in range(1, 9)}
fig, ax = plt.subplots()
ax.plot(list(bic), list(bic.values()), marker="o", color=ab.PALETTE[0])
ax.set_xlabel("komponensek száma"); ax.set_ylabel("BIC")
fig.tight_layout()
```

```
k_bic = min(bic, key=bic.get)
gmm = GaussianMixture(n_components=k_bic, covariance_type="full",
 ↪  random_state=0).fit(Th)
allapot = gmm.predict(Th)
print(f"BIC szerinti komponensszám: {k_bic};  ARI a termékcímkékkel:
 ↪  {adjusted_rand_score(Hh.termek, allapot):.2f}")
pd.crosstab(Hh.termek, pd.Series(allapot, name="üzemállapot"))
```

```
BIC szerinti komponensszám: 8;  ARI a termékcímkékkel: 0.55
```

üzemállapot termek	0	1	2	3	4	5	6	7
A	1383	0	0	0	1700	69	0	0

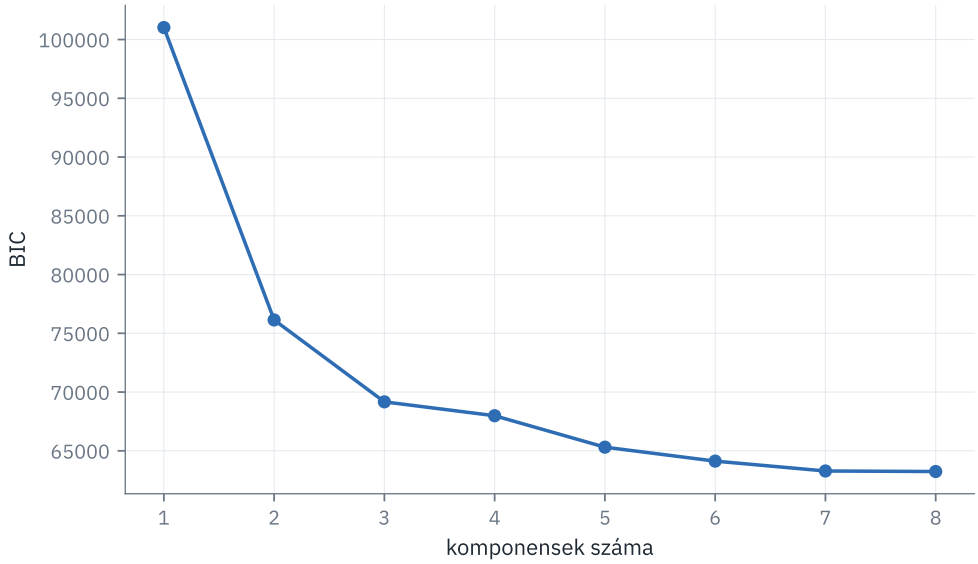
üzemállapot termek	0	1	2	3	4	5	6	7
B	0	0	1404	0	0	52	1352	0
C	0	90	0	1410	0	10	0	1234

```
hiba_oras = plant.truth["fault"].resample("1h").agg(lambda x:
↪  x.mode().iat[0])
(Hh.assign(allapot=allapot, hiba=Hh.ido.map(hiba_oras))
   .groupby("allapot").agg(orak=("ido", "size"), termek=("termek", lambda
↪  x: x.mode().iat[0]),
                           T_kornyezet=("T_kornyezet", "mean"),
↪  F_hutoviz=("F_hutoviz", "mean"),
                           E_fogyasztas=("E_fogyasztas", "mean"),
↪  honap_median=("ido", lambda x: x.dt.month.median()),
                           hibas_orak=("hiba", lambda x: (x !=
↪  "normál").mean()))
   .round(2))
```

allapot	orak	termek	T_kornyezet	F_hutoviz	E_fogyasztas	honap_median	hibas_orak
0	1383	A	19.59	5.23	72.80	7.0	0.14
1	90	C	18.58	6.91	75.67	5.0	0.11
2	1404	B	2.90	3.29	72.58	3.0	0.05
3	1410	C	18.06	4.33	69.77	6.0	0.03
4	1700	A	8.38	3.03	69.32	6.0	0.06
5	131	A	15.93	5.79	75.01	7.0	0.31
6	1352	B	15.74	5.81	76.58	8.0	0.11
7	1234	C	3.29	2.39	67.93	3.0	0.01

FUTÓ ESET · PANNON ÜZEM

A BIC a termékek számánál több komponenst javasol, és az állapotok profilja megmutatja az okát: minden termék működési tartománya egy hideg és egy meleg évszaki állapotra bomlik, amelyek a hűtővízáramban és az energiafogyasztásban különböznek. Egy külön, kis állapot a legnagyobb hűtési igényű órákat írja le (itt a legnagyobb a hűtővízáram), egy másik kis elemszámú, több termékre kiterjedő állapotban pedig az órák jelentős része hibás üzemállapothoz tartozik – a keverékmodell tehát a szokatlan működést is külön komponensbe sorolja. Az üzemállapotok azonosítása a folyamatfelügyelet fontos előkészítő lépése: az állapotfüggő felügyelet (15. fejezet) minden állapotra külön „normális” tartományt tanulhat – hogy ez mikor éri meg, azt ott vizsgáljuk meg.



5.9. ábra. A Gauss-keverék modell BIC-értéke a komponensek számának függvényében a Pannon üzem órás adatainak első három főkomponensén.

5.10.3. Önszerveződő térkép a vevőkre

Végül egy rövid, NumPy-ban írt SOM-megvalósítás a (5.10) frissítési szabály közvetlen átültetése. A 10×10-es rácsot a vevők standardizált jellemzőin tanítjuk.

```
def som_tanit(X, sorok=10, oszlopok=10, epochok=20, alpha0=0.5, sigma0=3.0,
↪   seed=0):
    rng = np.random.default_rng(seed)
    racs = np.array([(i, j) for i in range(sorok) for j in
↪   range(oszlopok)], float)
    W = X[rng.choice(len(X), sorok * oszlopok)]              # kezdés
↪   véletlen mintákból
    T = epochok * len(X)
    t = 0
    for ep in range(epochok):
        for k in rng.permutation(len(X)):
            frac = t / T
            alpha, sigma = alpha0 * (1 - frac), sigma0 * (1 - frac) + 0.5
            c = np.argmin(((W - X[k]) ** 2).sum(1))           # legjobban
↪   illeszkedő neuron
            h = np.exp(-((racs - racs[c]) ** 2).sum(1) / (2 * sigma ** 2))
            W += alpha * h[:, None] * (X[k] - W)
```

```
            t += 1
    return W, racs

W, racs = som_tanit(Zv, epochok=10)
umat = np.zeros(len(W))
for j in range(len(W)):
    szomsz = np.where(np.abs(racs - racs[j]).sum(1) == 1)[0]
    umat[j] = np.mean(np.linalg.norm(W[szomsz] - W[j], axis=1))
bmu = np.array([np.argmin(((W - x) ** 2).sum(1)) for x in Zv])
```

```
from matplotlib.colors import ListedColormap
szegm_nevek = ["törzsvásárló", "alkalmi", "akcióvadász", "viszonteladó",
↪  "lemorzsolódó"]
kod = {s: i for i, s in enumerate(szegm_nevek)}
domin = np.full(len(W), np.nan)
for j in range(len(W)):
    s = valodi[bmu == j]
    if len(s):
        domin[j] = kod.get(pd.Series(s).mode().iat[0], np.nan)
fig, ax = plt.subplots(1, 2)
im = ax[0].imshow(umat.reshape(10, 10), cmap="Greys_r");
↪  ax[0].set_title("U-mátrix")
ax[1].imshow(domin.reshape(10, 10), cmap=ListedColormap(ab.PALETTE[:5]),
↪  vmin=-0.5, vmax=4.5)
ax[1].set_title("leggyakoribb rejtett szegmens")
for a in ax: a.set_xticks([]); a.set_yticks([]); a.grid(False)
from matplotlib.patches import Patch
ax[1].legend(handles=[Patch(color=ab.PALETTE[i], label=s) for i, s in
↪  enumerate(szegm_nevek)],
           fontsize=7, loc="center left", bbox_to_anchor=(1.02, 0.5))
fig.tight_layout()
```

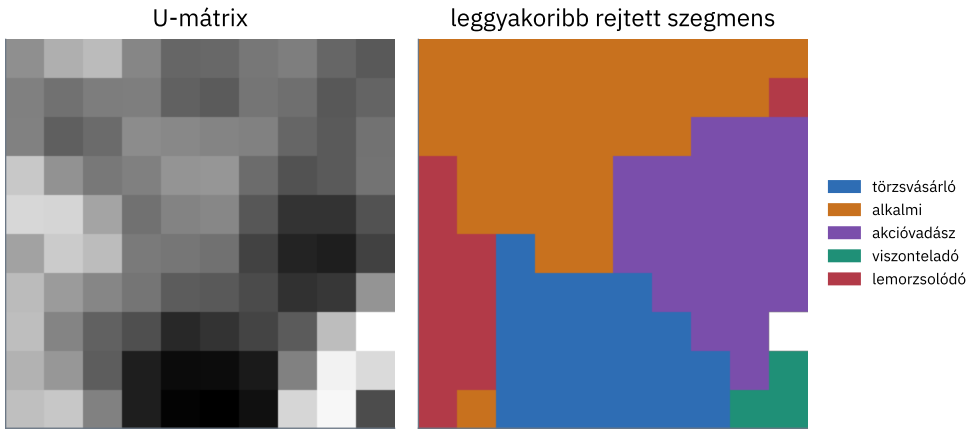
Interaktív alkalmazás: A k-átlag és a fuzzy c-átlag algoritmus lépésről lépésre

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/kmeans.html

Colab-notebook: 5. fejezet – csoportosítás

5.11. Összefoglalás

- A csoportosítás eredménye a távolságmértéktől és a módszer által feltételezett csoportalaktól függ; nincs minden helyzetben jól működő módszer.



5.10. ábra. A vevők önszerveződő térképe. Balra az U-mátrix (világos: nagy távolság a szomszédos prototípusok között – csoporthatár), jobbra a neuronokra leképezett vevők leggyakoribb rejtett szegmense.

- A hierarchikus módszerek egymásba ágyazott csoportokat (dendrogramot) adnak; a kapcsolódási szabályok a Lance–Williams-formulával egységesen kezelhetők.
- A k-átlag a csoporton belüli négyzetes eltérésösszeget minimalizálja; gyors, de lokális optimumba konvergál, és gömbszerű csoportokat feltételez.
- A fuzzy c-átlag tagsági értékeket ad, a frissítési szabályai a célfüggvény feltételes szélsőértékéből vezethetők le; a Gauss-keverék modell valószínűségi tagságot ad, és a csoportszám a BIC-kel választható.
- A DBSCAN tetszőleges alakú csoportokat talál, és a zajt külön kezeli; az önszerveződő térkép a hasonló objektumokat szomszédos rácspontokra képezi le.
- A csoportosítás értékelésére belső (sziluett, Davies–Bouldin, Caliński–Harabasz), külső (ARI) és stabilitási mutatók szolgálnak, de a végső mérce az értelmezhetőség.

Ellenőrző kérdések

1. Miért kell a csoportosítás előtt standardizálni a változókat? Mikor nem?
2. Miért jobb a Jaccard-együttható a vásárlási kosarak összehasonlítására, mint az egyszerű egyezési együttható?
3. Igazolja a Lance–Williams-formulából, hogy teljes kapcsolódásnál $d_{(ij)k} = \max(d_{ik}, d_{jk})$!

4. Miért nem növekedhet a k-átlag célfüggvénye egyik lépésben sem? Miért nem garantált mégis a globális optimum?
5. Mi történik az FCM tagsági értékeivel, ha $m \to 1$, illetve ha $m \to \infty$?
6. Milyen adatokon működik jól a DBSCAN, és miért nem működik jól eltérő sűrűségű csoportoknál?
7. Mit mér a sziluett-együttható, és mit jelent egy objektum negatív sziluettértéke?

Feladatok

1. **Kézi számolás.** Végezze el kézzel a hét pont teljes kapcsolódású hierarchikus csoportosításának első három lépését, és vesse össze a `scipy` eredményével!
2. **FCM levezetés.** Vezesse le a (5.6) középpont-frissítési képletét a $\partial J_m / \partial \mathbf{v}_i = \mathbf{0}$ feltételből euklideszi norma esetén!
3. **Kezdőpontok hatása.** Futtassa a saját k-átlag függvényt a vevőadatokon 20 különböző véletlen kezdőértékkel ($c = 5$)! Hányféle különböző J értéket kap? Mi a tanulság?
4. **DBSCAN paraméterezése.** A k-adik legközelebbi szomszéd távolságának rendezett ábrája (k = minPts) segít az ε megválasztásában. Készítse el az ábrát a 5.1. (c) ábra adataira, és futtassa a DBSCAN-t a kiválasztott paraméterekkel!
5. **Stabilitás.** Vegyen 20 bootstrap-mintát a vevőkből, futtassa mindegyiken a k-átlagot $c = 3, 5, 8$ csoporttal, és mérje az ARI-vel a felosztások páronkénti egyezését! Melyik csoportszám adja a legstabilabb szegmentálást?
6. **Gustafson–Kessel.** Egészítse ki az `fcm` függvényt a csoportonkénti adaptív normával (Gustafson & Kessel, 1979), és próbálja ki a 5.1. (b) ábra elnyújtott csoportjain!

További olvasmányok

A csoportosítás klasszikus monográfiája L. Kaufman & Rousseeuw (1990). A fuzzy csoportosítás elméletét Bezdek (1981), adatbányászati és rendszeridentifikációs alkalmazásait Abonyi & Feil (2007) tárgyalja. A Gauss-keverék modellek és az EM-algoritmus részletes leírása Bishop (2006) könyvében található. A sűrűségalapú módszerekről Ester és mtsai. (1996) és Campello és mtsai. (2013), az önszerveződő térképekről Kohonen (2001) ír. A belső validitási mutatók összehasonlításához J. Han és mtsai. (2022) csoportosítási fejezete ad jó kiindulópontot.

6. Gyakori mintázatok és asszociációs szabályok

Miről szól a fejezet?

Mely termékeket vásárolják együtt? Mely riasztások követik rendszeresen egymást egy üzemben? Az ilyen kérdésekre a gyakori elemhalmazok és az asszociációs szabályok feltárása ad választ. A fejezet bevezeti az alapfogalmakat (támogatottság, megbízhatóság), bemutatja a keresés hatékonyságát biztosító Apriori-elvet és a rá épülő algoritmusokat (Apriori, Eclat, FP-growth), a gyakori elemhalmazok tömör reprezentációit, a szabályok érdekességének mérőszámait, végül a többszintű, a mennyiségi és a sorozatmintákra vonatkozó kiterjesztéseket.

A fejezet végére az olvasó képes lesz egy kis tranzakciós adatbázison kézzel végigvinni az Apriori-algoritmust; nagy adatbázison FP-growth-szal gyakori elemhalmazokat és szabályokat keresni; a szabályokat a lift, a leverage és a conviction alapján kritikusan értékelni; és eseménynaplókban gyakori sorozatokat feltárni.

6.1. Alapfogalmak

A feladatot Agrawal, Imieliński, és mtsai. (1993) fogalmazta meg a vásárlói kosarak elemzésére. Legyen $\mathcal{I} = \{i_1, \dots, i_d\}$ az **elemek** (termékek) halmaza, és legyen az adatbázis N **tranzakció** (kosár) $\mathcal{T} = \{T_1, \dots, T_N\}$ gyűjteménye, ahol minden $T_k \subseteq \mathcal{I}$. Elemek egy $X \subseteq \mathcal{I}$ halmazát **elemhalmaznak**, k elemű elemhalmazt k-elemhalmaznak nevezünk.

Egy elemhalmaz **támogatottsága** (support) azoknak a tranzakcióknak az aránya, amelyek tartalmazzák:

$$s(X) = \frac{\sigma(X)}{N}, \qquad \sigma(X) = \big|\{T_k \in \mathcal{T} : X \subseteq T_k\}\big|. \tag{6.1}$$

Az X elemhalmaz **gyakori**, ha támogatottsága eléri a felhasználó által megadott $s_{\min}$ küszöböt.

Egy **asszociációs szabály** $X \Rightarrow Y$ alakú, ahol X és Y diszjunkt elemhalmazok (X az előzmény, Y a következmény). A szabály támogatottsága $s(X \Rightarrow Y) = s(X \cup Y)$, **megbízhatósága** (confidence) pedig annak becsült feltételes valószínűsége, hogy egy X-et tartalmazó kosár Y-t is tartalmaz:

$$c(X \Rightarrow Y) = \frac{s(X \cup Y)}{s(X)} \approx P(Y \mid X). \tag{6.2}$$

Az asszociációs szabályok feltárásának feladata az összes olyan szabály megtalálása, amelynek támogatottsága legalább $s_{\min}$, megbízhatósága legalább $c_{\min}$. A feladatot két lépésre bontjuk: (1) megkeressük a gyakori elemhalmazokat, (2) mindegyikből előállítjuk a megbízható szabályokat. A számításigényes rész az első lépés.

> **A keresési tér mérete**
>
> d elemből $2^d - 1$ nem üres elemhalmaz és $3^d - 2^{d+1} + 1$ lehetséges szabály képezhető (minden elem az előzménybe, a következménybe vagy egyikbe sem kerülhet, és egyik oldal sem lehet üres). A PannonShop mindössze 60 termékére ez $2^{60} \approx 10^{18}$ elemhalmaz – a kimerítő keresés reménytelen.

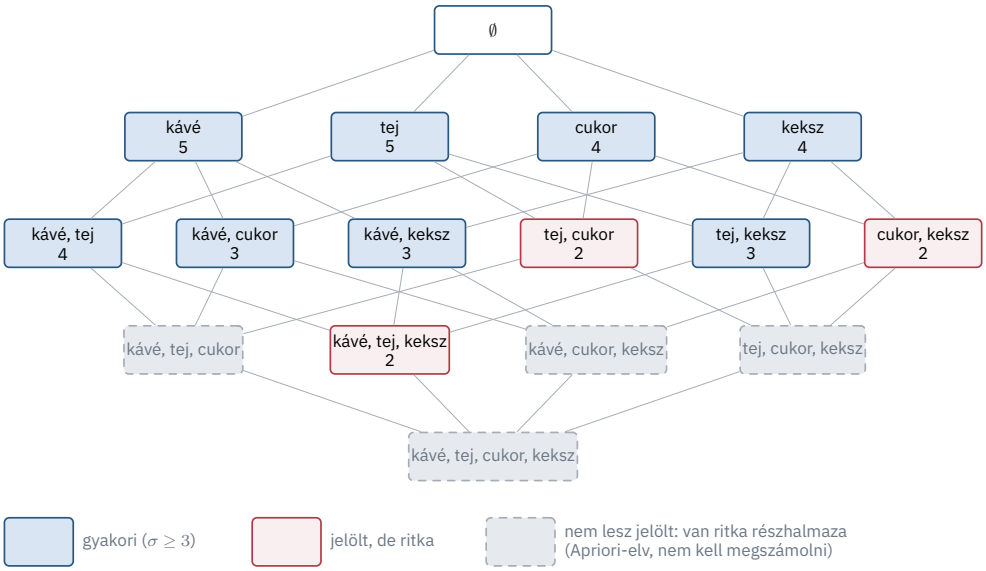
6.2. Az Apriori-elv és az Apriori-algoritmus

6.2.1. Az Apriori-elv

A keresési tér hatékony bejárásának kulcsa egy egyszerű megfigyelés: ha egy elemhalmaz gyakori, akkor minden részhalmaza is gyakori, hiszen minden olyan kosár, amely tartalmazza az elemhalmazt, a részhalmazait is tartalmazza. Formálisan a támogatottság **anti-monoton**:

$$X \subseteq Y \quad \Longrightarrow \quad s(X) \geq s(Y). \tag{6.3}$$

Ennek megfordítása a gyakorlatban hasznos alak: **ha egy elemhalmaz ritka, minden bővítése is ritka**, így a bővítéseit meg sem kell számolni (6.1. ábra).



6.1. ábra. Az elemhalmazok hálója a mini-kosárpélda négy termékére (a számok a támogatottságot jelzik, $\sigma_{\min} = 3$). A {tej, cukor} és a {cukor, keksz} ritka, ezért egyetlen bővítésük sem lehet gyakori; a {kávé, tej, keksz} jelölt ugyan, de a számolás után ritkának bizonyul.

6.2.2. Az algoritmus

Az **Apriori-algoritmus** (Agrawal & Srikant, 1994) szintenként halad: először a gyakori 1-elemhalmazokat keresi meg, majd a k-adik szinten a $(k-1)$-elemű gyakori halmazokból állítja elő a k-elemű **jelölteket**, és ezek támogatottságát egyetlen adatbázis-bejárással számolja meg. A jelöltképzés két lépésből áll:

1. **Összekapcsolás**: két gyakori $(k-1)$-elemhalmazt, amelyek (rendezett alakban) az első $k-2$ elemükben megegyeznek, egy k-elemű jelöltté egyesítünk.
2. **Metszés**: elhagyjuk azokat a jelölteket, amelyeknek van ritka $(k-1)$-elemű részhalmaza (6.3).

Az algoritmus összefoglalása: Apriori

1. Számoljuk meg az egyes elemek támogatottságát; F_1 a gyakori elemek halmaza.
2. $k = 2, 3, \ldots$ esetén, amíg F_{k-1} nem üres:
   a. állítsuk elő a C_k jelölthalmazt F_{k-1} elemeinek összekapcsolásával és metszéssel;
   b. egy adatbázis-bejárással számoljuk meg a jelöltek támogatottságát;
   c. F_k a C_k gyakori elemei.
3. A gyakori elemhalmazok: $F_1 \cup F_2 \cup \ldots$

6.2.3. Kézi példa

A segédcsomag nyolc kosarán kövessük végig az algoritmust $\sigma_{\min} = 3$ (azaz $s_{\min} = 37{,}5\%$) mellett!

```
kosarak = ab.kosar_mini()
for i, k in enumerate(kosarak, 1):
    print(f"T{i}: {', '.join(k)}")
```

```
T1: kávé, tej, cukor
T2: kávé, tej
T3: tea, citrom, cukor
T4: kávé, tej, keksz
T5: tea, citrom
T6: kávé, cukor, keksz
T7: kávé, tej, cukor, keksz
T8: tea, tej, keksz
```

Az elemek támogatottsága alapján $F_1 = \{$kávé (5), tej (5), cukor (4), keksz (4), tea (3)$\}$; a citrom (2) ritka. A C_2 jelöltek F_1 összes párja (10 db); közülük gyakori a {kávé, tej} (4), a {kávé, cukor} (3), a {kávé, keksz} (3) és a {tej, keksz} (3). A C_3 jelöltek összekapcsolással: {kávé, tej, cukor}, {kávé, tej, keksz} és {kávé, cukor, keksz}. A metszés az elsőt elhagyja (a {tej, cukor} ritka) és a harmadikat is (a {cukor, keksz} ritka), így csak a {kávé, tej, keksz} marad; ennek támogatottsága 2, tehát ritka. F_3 üres, az algoritmus megáll.

Írjuk meg az algoritmust Pythonban – a kód szinte szó szerint követi az összefoglalást:

```
from itertools import combinations

def apriori_sajat(tranzakciok, min_sigma):
    T = [frozenset(t) for t in tranzakciok]
    def szamol(jeloltek):
        return {c: sum(c <= t for t in T) for c in jeloltek}
    elemek = {frozenset([i]) for t in T for i in t}
    F = {c: s for c, s in szamol(elemek).items() if s >= min_sigma}
    osszes, k = dict(F), 2
    while F:
        elozo = sorted(tuple(sorted(c)) for c in F)
        jeloltek = set()
        for a, b in combinations(elozo, 2):                # összekapcsolás
            if a[:k - 2] == b[:k - 2]:
                c = frozenset(a) | frozenset(b)
                if all(frozenset(r) in F for r in combinations(sorted(c), k
                ↪  - 1)):   # metszés
                    jeloltek.add(c)
        F = {c: s for c, s in szamol(jeloltek).items() if s >= min_sigma}
        osszes.update(F)
        k += 1
    return osszes

gyakori = apriori_sajat(kosarak, min_sigma=3)
for c, s in sorted(gyakori.items(), key=lambda x: (len(x[0]), -x[1])):
    print(f"{set(c)}: {s}")
```

```
{'kávé'}: 5
{'tej'}: 5
{'keksz'}: 4
{'cukor'}: 4
{'tea'}: 3
{'kávé', 'tej'}: 4
{'cukor', 'kávé'}: 3
{'keksz', 'tej'}: 3
{'keksz', 'kávé'}: 3
```

6.2.4. Az Apriori továbbfejlesztései

Az Apriori fő költsége a sok jelölt és az ismételt adatbázis-bejárás. A klasszikus továbbfejlesztések ezeket csökkentik: a **DHP** (Park és mtsai., 1995) hasítótáblával szűri a C_2 jelölteket, és bejárásonként szűkíti az adatbázist; a **DIC** (Brin és mtsai., 1997) a bejárás közben, menet közben indít új jelölteket, így kevesebb bejárás kell. A ma használt algoritmusok azonban más elven működnek.

6.3. Mélységi és mintanövelő algoritmusok

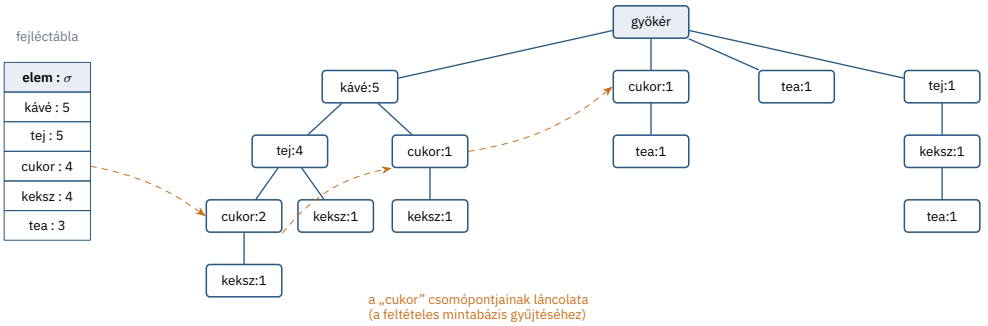
6.3.1. Vertikális adatábrázolás: Eclat

Az **Eclat** (Zaki, 2000) a tranzakciós adatbázist *függőlegesen* tárolja: minden elemhez a tartalmazó tranzakciók azonosítóinak listáját (tid-listáját) rendeli. Egy elemhalmaz támogatottsága így a tid-listák metszetének hossza – nem kell az adatbázist újra bejárni. A mini-példában a kávé tid-listája $\{1, 2, 4, 6, 7\}$, a tejé $\{1, 2, 4, 7, 8\}$, a metszet $\{1, 2, 4, 7\}$, tehát $\sigma(\{\text{kávé, tej}\}) = 4$. Az Eclat a hálót mélységben járja be, és a metszeteket fokozatosan szűkíti.

6.3.2. FP-growth

Az **FP-growth** (J. Han és mtsai., 2000) egyáltalán nem állít elő jelölteket. Két bejárással egy tömör prefixfát, az **FP-fát** építi fel (6.2. ábra): az első bejárás megszámolja az elemek gyakoriságát, a második minden tranzakció gyakori elemeit gyakoriság szerint csökkenő sorrendbe rendezi, és a fába illeszti. A közös előtagú tranzakciók közös utat használnak, így a fa gyakran sokkal kisebb, mint az adatbázis. A gyakori elemhalmazokat ezután rekurzívan bányászuk: minden elemre összegyűjtjük a fában hozzá vezető utakat (a **feltételes mintabázist**), ebből feltételes FP-fát építünk, és azon folytatjuk a keresést.

A „cukor” feltételes mintabázisa a fából kiolvasható: a {kávé, tej} út kétszer, a {kávé} út egyszer vezet cukor csomóponthoz (a gyökértől közvetlenül induló cukor csomópont előtt nincs elem). A cukorral együtt gyakori elem tehát csak a kávé (3), így a {kávé, cukor} gyakori, a {tej, cukor} (2) nem – egyezésben a hálóból kapott eredménnyel.



6.2. ábra. A mini-kosárpélda FP-fája ($\sigma_{\min} = 3$, az elemek gyakoriság szerint rendezve). A csomópontok az elem nevét és az adott úton átmenő tranzakciók számát mutatják; a fejléctábla szaggatott láncai az azonos elemű csomópontokat kötik össze.

6.4. Tömör reprezentációk: zárt és maximális elemhalmazok

Alacsony támogatottsági küszöbnél a gyakori elemhalmazok száma óriási lehet, hiszen egy k elemű gyakori halmaz minden részhalmaza is gyakori. Két tömör reprezentáció csökkenti ezt a redundanciát:

- Egy gyakori elemhalmaz **maximális**, ha egyik valódi bővítése sem gyakori (Bayardo, 1998). A maximális gyakori elemhalmazokból az összes gyakori elemhalmaz előállítható (mint részhalmazaik), de a támogatottságuk nem.
- Egy elemhalmaz **zárt**, ha egyik valódi bővítésének sem ugyanakkora a támogatottsága (Pasquier és mtsai., 1999). A zárt gyakori elemhalmazokból a gyakori elemhalmazok *és* támogatottságaik is visszaállíthatók: egy elemhalmaz támogatottsága a legkisebb őt tartalmazó zárt halmaz támogatottsága.

A kettő viszonya: minden maximális gyakori elemhalmaz zárt is, így maximális $\subseteq$ zárt $\subseteq$ gyakori.

6.5. Szabályok előállítása és értékelése

6.5.1. Szabályok a gyakori elemhalmazokból

Egy Z gyakori elemhalmazból minden $X \Rightarrow Z \setminus X$ szabály ($\emptyset \neq X \subset Z$) támogatottsága ugyanaz, $s(Z)$; csak a megbízhatóságot kell kiszámolni. A megbízhatóság általában nem anti-monoton, egy rögzített Z-ből képzett szabályokra

viszont igen: ha a következményt bővítjük (az előzményt szűkítjük), a megbízhatóság nem nőhet, mert $c = s(Z)/s(X)$ és a kisebb X támogatottsága nagyobb. Ezért ha egy szabály nem megbízható, a több elemű következményű „rokonai” sem azok.

6.5.2. Az érdekesség mérése

A támogatottság és a megbízhatóság önmagában félrevezető lehet. Ha egy termék nagyon népszerű, minden szabály, amelynek következménye, nagy megbízhatóságú lesz – akkor is, ha az előzménynek semmi köze hozzá. A **lift** a megbízhatóságot a következmény alapgyakoriságához viszonyítja:

$$\text{lift}(X \Rightarrow Y) = \frac{c(X \Rightarrow Y)}{s(Y)} = \frac{s(X \cup Y)}{s(X)\, s(Y)}. \tag{6.4}$$

A lift értéke 1, ha X és Y előfordulása (a mintán) független; 1 fölött pozitív, alatta negatív kapcsolatot jelez. További gyakori mérőszámok a **leverage** (más néven Piatetsky-Shapiro-mérték) (Piatetsky-Shapiro, 1991), amely a különbséget méri a független esethez képest, és a **conviction** (Brin és mtsai., 1997), amely azt méri, hányszor gyakrabban fordulna elő a szabály „megsértése” (X előfordul, de Y nem), ha X és Y független lenne:

$$\text{leverage}(X \Rightarrow Y) = s(X \cup Y) - s(X)\, s(Y), \qquad \text{conviction}(X \Rightarrow Y) = \frac{1 - s(Y)}{1 - c(X \Rightarrow} \tag{6.5}$$

A mérőszámok eltérő tulajdonságúak (szimmetrikusak-e, hogyan viselkednek ritka elemekre, hogyan függnek a kosarak számától); választásukhoz Tan és mtsai. (2004) ad szempontokat.

> **Buktató: a megbízható, de érdektelen szabály**
>
> Ha a vásárlók 80%-a vesz kávét, és a teát vásárlók 75%-a is vesz kávét, a „tea ⇒ kávé” szabály megbízhatósága 75% – mégis *negatív* kapcsolatot jelez, hiszen a teát vásárlók kevésbé valószínűen vesznek kávét, mint egy átlagos vásárló (lift = 0,75/0,80 ≈ 0,94). A megbízhatóságot mindig a következmény alapgyakoriságához kell viszonyítani.

Az asszociációs szabályok együttes előfordulást írnak le, **nem okozati kapcsolatot**. Egy erős szabály oka lehet egy közös harmadik tényező (például egy akció,

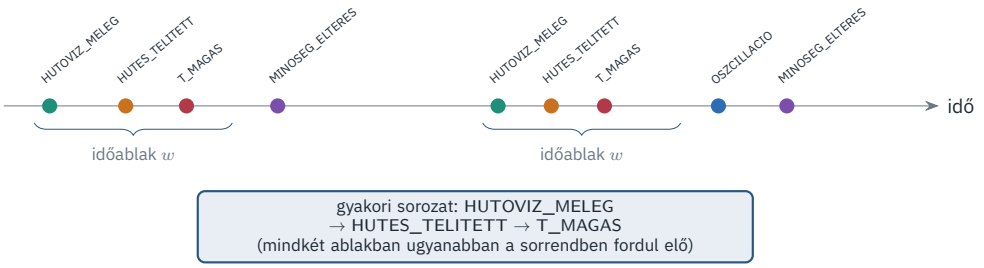
amelyben a két termék együtt szerepelt), és a sok tesztelt szabály között a véletlen is produkál „erős” szabályokat; a nagyon kis támogatottságú szabályokat ezért óvatosan kell kezelni.

6.6. Kiterjesztések

Többszintű szabályok. A termékek gyakran hierarchiába szerveződnek (termék → alkategória → kategória). Termékszinten egy kapcsolat túl ritka lehet ahhoz, hogy kimutassuk, kategóriaszinten viszont már erős. Az általánosított asszociációs szabályok (Srikant & Agrawal, 1995) egyszerre több szinten keresnek, és kiszűrik azokat az alacsonyabb szintű szabályokat, amelyek csak a magasabb szintű szabályt ismétlik.

Mennyiségi szabályok. Numerikus attribútumok (életkor, vásárlási érték, hőmérséklet) esetén az értékeket intervallumokra bontjuk, és az „attribútum ∈ intervallum” állításokat tekintjük elemeknek (Srikant & Agrawal, 1996a). Az eredmény erősen függ a diszkretizálástól: a túl keskeny intervallumok ritkák, a túl szélesek elmossák a kapcsolatot.

Sorozatminták. Ha a tranzakciók időrendje is számít (egy vevő egymást követő vásárlásai, egy üzem riasztásainak sorrendje), **gyakori sorozatokat** keresünk: olyan eseménysorozatokat, amelyek sok sorozatban ugyanabban a sorrendben fordulnak elő. A klasszikus algoritmusok a GSP (Srikant & Agrawal, 1996b), amely az Apriori-elvet viszi át sorozatokra, és a PrefixSpan (Pei és mtsai., 2001), amely az FP-growth-hoz hasonlóan vetített adatbázisokon dolgozik. Ipari riasztásnaplókban a gyakori riasztássorozatok feltárása a riasztási „árvizek” csökkentésének egyik eszköze (Dorgo & Abonyi, 2018; J. Wang és mtsai., 2016) (6.3. ábra).



6.3. ábra. Gyakori sorozat egy riasztásnaplóban. Ha ugyanaz az eseménysorrend egy rövid időablakon belül ismételten előfordul, a riasztások valószínűleg egy közös kiváltó okra vezethetők vissza.

6.7. Gyakorlat: kosárelemzés és riasztássorozatok

6.7.1. A PannonShop kosarai

A `mlxtend` csomag (Raschka, 2018) a gyakori elemhalmazok keresését `pandas` adattáblákon végzi, amelyekben minden sor egy kosár, minden oszlop egy termék logikai jelzője. A sztornózott számlák nélküli, megtisztított számlatételekből indulunk (3. fejezet).

```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import fpgrowth, apriori, association_rules

tiszta = ab.shop_clean()
kosar = tiszta.groupby("Invoice").Description.apply(lambda s:
 ↪  sorted(set(s)))
print(f"kosarak: {len(kosar):,}, átlagos kosárméret:
 ↪  {kosar.str.len().mean():.2f}".replace(",", " "))
te = TransactionEncoder()
B = pd.DataFrame(te.fit_transform(kosar), columns=te.columns_)
B.iloc[:3, :6]
```

```
kosarak: 15 295  átlagos kosárméret: 4.16
```

	Adventi koszorú	Ajándékcsomagolás	Ajándékkísérő kártya	Ajándékutalvány 10 000 Ft	Asztali futó	Díszdoboz
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False

```
import time
for alg, f in [("Apriori", apriori), ("FP-growth", fpgrowth)]:
    t0 = time.perf_counter()
    gy = f(B, min_support=0.005, use_colnames=True)
    print(f"{alg:10s}: {len(gy):5d} gyakori elemhalmaz,
     ↪  {time.perf_counter() - t0:6.2f} s")
```

```
Apriori   :   919 gyakori elemhalmaz,   0.35 s
FP-growth :   919 gyakori elemhalmaz,   0.27 s
```

Mindkét algoritmus ugyanazokat a gyakori elemhalmazokat adja; az FP-growth jellemzően gyorsabb, különösen alacsony küszöbnél. A szabályokat a lift szerint rendezzük:

```
gy = fpgrowth(B, min_support=0.005, use_colnames=True)
sz = association_rules(gy, metric="confidence", min_threshold=0.3)
```

```
sz["szabaly"] = sz.antecedents.apply(lambda s: ", ".join(sorted(s))) + "
↪  ⇒  " + sz.consequents.apply(lambda s: ", ".join(sorted(s)))
sz[["szabaly", "support", "confidence", "lift", "leverage", "conviction"]]
↪  \
  .sort_values("lift", ascending=False).head(12).round(3)
```

	szabaly	support	confidence	lift	leverage	conviction
93	Kávébab 1 kg, Teáskanna ⇒ Fekete tea 50 filter, Zöld t...	0.006	0.312	8.309	0.005	1.399
92	Fekete tea 50 filter, Kávébab 1 kg, Zöld tea 50 filter ...	0.006	0.649	7.913	0.005	2.617
106	Kávéfőző gép ⇒ Kávébab 1 kg, Vízkőoldó 500 ml	0.023	0.378	7.908	0.020	1.532
105	Kávébab 1 kg, Vízkőoldó 500 ml ⇒ Kávéfőző gép	0.023	0.485	7.908	0.020	1.823
76	Fekete tea 50 filter, Zöld tea 50 filter ⇒ Teáskanna	0.024	0.632	7.707	0.021	2.497
125	Kávékapszula 30 db, Tollkészlet ⇒ Pontozott füzet A5	0.005	0.440	6.304	0.005	1.661
124	Kávékapszula 30 db, Pontozott füzet A5 ⇒ Tollkészlet	0.005	0.609	5.964	0.005	2.295
127	Gyümölcstea 40 g, Tollkészlet ⇒ Pontozott füzet A5	0.006	0.413	5.915	0.005	1.584
126	Gyümölcstea 40 g, Pontozott füzet A5 ⇒ Tollkészlet	0.006	0.578	5.666	0.005	2.129
23	Illatgyertya, Kávébab 1 kg ⇒ Gyertyatartó	0.008	0.467	5.661	0.006	1.723
22	Gyertyatartó, Kávébab 1 kg ⇒ Illatgyertya	0.008	0.380	5.603	0.006	1.503
122	Pontozott füzet A5, Vízkőoldó 500 ml ⇒ Tollkészlet	0.006	0.565	5.535	0.005	2.064

FUTÓ ESET · PANNONSHOP

A legnagyobb liftű szabályok között szerepelnek a szimulátor által „beültetett” kapcsolatok (kávéfőző gép ⇒ kávébab, vízkőoldó; teáskanna ⇒ zöld és fekete tea; pontozott füzet ⇒ tollkészlet; illatgyertya ⇒ gyertyatartó). Ellenőrizzük szisztematikusan, hogy az algoritmus mindegyiket megtalálta-e, és mekkora lifttel!

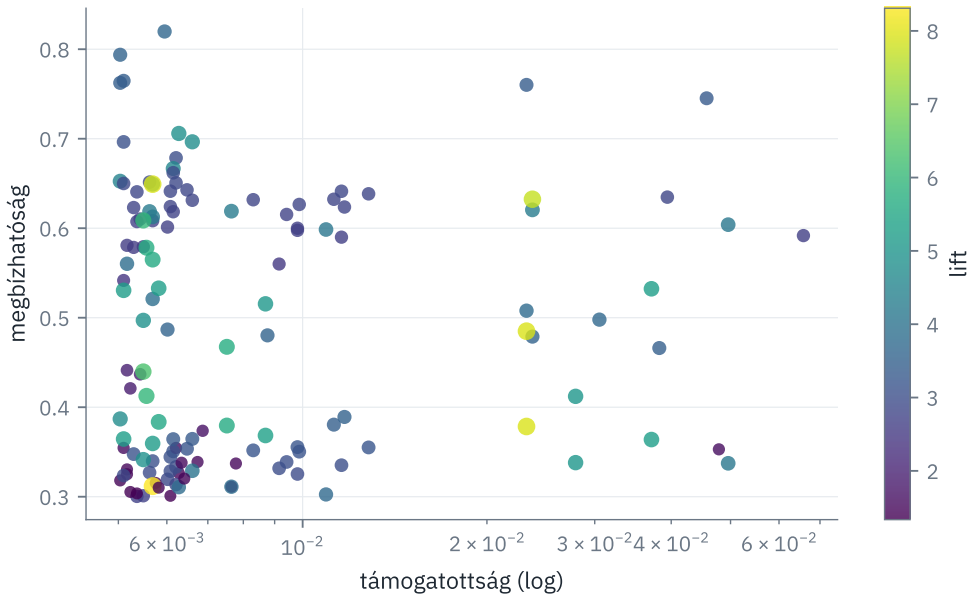
```
beultetett = ab.load_shop().truth["planted_rules"]
sorok = []
for ante, cons, p in beultetett:
    m = sz[(sz.antecedents == frozenset(ante)) & (sz.consequents ==
↪  frozenset([cons]))]
    sorok.append({"szabály": f"{ante[0]} ⇒ {cons}", "beültetett
↪  valószínűség": p,
                  "megtalálva": len(m) > 0,
                "megbízhatóság": m.confidence.iat[0] if len(m) else np.nan,
                  "lift": m.lift.iat[0] if len(m) else np.nan})
pd.DataFrame(sorok).round(3)
```

	szabály	beültetett valószínűség	megtalálva	megbízhatóság	lift
0	Kávéfőző gép ⇒ Kávébab 1 kg	0.70	True	0.745	3.306
1	Kávéfőző gép ⇒ Vízkőoldó 500 ml	0.45	True	0.498	3.672
2	Kávédaráló ⇒ Kávébab 1 kg	0.60	True	0.635	2.816
3	Teáskanna ⇒ Zöld tea 50 filter	0.55	True	0.604	4.111
4	Teáskanna ⇒ Fekete tea 50 filter	0.40	True	0.466	3.433
5	Pontozott füzet A5 ⇒ Tollkészlet	0.50	True	0.532	5.216
6	Tejhabosító kancsó ⇒ Kávébab 1 kg	0.50	True	0.592	2.625

	szabály	beültetett valószínűség	megtalálva	megbízhatóság	lift
7	Illatgyertya ⇒ Gyertyatartó	0.35	True	0.412	4.991

A megtalált szabályok megbízhatósága nem pontosan egyezik a beültetett valószínűséggel. Ennek két oka van: a következmény néha a szabálytól függetlenül, a vevő saját preferenciái miatt is a kosárba kerül (ez növeli a megbízhatóságot), és ha a kosárban több előzmény is van, a szabályok egymást erősítik.

```
fig, ax = plt.subplots()
sc = ax.scatter(sz.support, sz.confidence, c=sz.lift, cmap="viridis", s=12
↪  + 4 * sz.lift, alpha=0.8)
fig.colorbar(sc, ax=ax, label="lift")
ax.set_xscale("log"); ax.set_xlabel("támogatottság (log)");
↪  ax.set_ylabel("megbízhatóság")
fig.tight_layout()
```



6.4. ábra. A PannonShop asszociációs szabályai a támogatottság–megbízhatóság síkon, a lift szerint színezve. A kis támogatottságú, de nagy liftű szabályok (jobb felső színek a bal oldalon) a legérdekesebbek.

6.7.2. Kategóriaszintű szabályok

A termékeket kategóriájukkal helyettesítve kategóriaszintű kosarakat kapunk; a kategóriák között gyakoribb, de gyengébb kapcsolatok mutathatók ki:

```
termek_kat = ab.load_shop().products.set_index("Description").Kategoria
kat_kosar = tiszta.assign(kat=tiszta.Description.map
↪  (termek_kat)).groupby("Invoice").kat.apply(lambda s: sorted(set(s)))
te2 = TransactionEncoder()
BK = pd.DataFrame(te2.fit_transform(kat_kosar), columns=te2.columns_)
szk = association_rules(fpgrowth(BK, min_support=0.03, use_colnames=True),
↪  metric="lift", min_threshold=1.05)
szk["szabaly"] = szk.antecedents.apply(lambda s: ", ".join(sorted(s))) + "
↪  ⇒  " + szk.consequents.apply(lambda s: ", ".join(sorted(s)))
szk[["szabaly", "support", "confidence", "lift"]].sort_values("lift",
↪  ascending=False).head(8).round(3)
```

	szabaly	support	confidence	lift
36	Ajándék, Háztartási gépek ⇒ Dekoráció, Kávé és tea	0.035	0.376	1.666
33	Dekoráció, Kávé és tea ⇒ Ajándék, Háztartási gépek	0.035	0.156	1.666
31	Ajándék, Háztartási gépek, Kávé és tea ⇒ Dekoráció	0.035	0.528	1.360
38	Dekoráció ⇒ Ajándék, Háztartási gépek, Kávé és tea	0.035	0.091	1.360
53	Dekoráció, Írószer ⇒ Ajándék, Kávé és tea	0.034	0.318	1.354
54	Ajándék, Kávé és tea ⇒ Dekoráció, Írószer	0.034	0.144	1.354
26	Ajándék, Háztartási gépek ⇒ Dekoráció	0.048	0.517	1.332
27	Dekoráció ⇒ Ajándék, Háztartási gépek	0.048	0.125	1.332

6.7.3. Zárt és maximális elemhalmazok

```
def zart_maximalis(gy):
    halmazok = dict(zip(gy.itemsets, gy.support))
    zart = maxi = 0
    for X, s in halmazok.items():
        bovitesek = [Y for Y in halmazok if len(Y) == len(X) + 1 and X < Y]
        zart += all(abs(halmazok[Y] - s) > 1e-12 for Y in bovitesek)
        maxi += len(bovitesek) == 0
    return zart, maxi

for s_min in [0.02, 0.01, 0.005, 0.002]:
    g = fpgrowth(B, min_support=s_min, use_colnames=True)
    z, m = zart_maximalis(g)
    print(f"s_min = {s_min:<6}: gyakori {len(g):5d}   zárt {z:5d}
    ↪  maximális {m:5d}")
```

```
s_min = 0.02  : gyakori    81   zárt    81   maximális    56
```

```
s_min = 0.01  : gyakori   282   zárt   282   maximális   217
s_min = 0.005 : gyakori   919   zárt   919   maximális   736
s_min = 0.002 : gyakori  2781   zárt  2781   maximális  1863
```

(Elég az eggyel nagyobb gyakori bővítéseket vizsgálni: ha egy elemhalmaznak van azonos támogatottságú bővítése, akkor eggyel nagyobb is van, és egy nem gyakori bővítés nem lehet azonos támogatottságú.) Ezen az adaton minden gyakori elemhalmaz zárt is – a viszonylag kis, változatos kosaraknál ritka, hogy egy bővítés pontosan ugyanannyi kosárban forduljon elő –, a maximális elemhalmazok viszont lényegesen kevesebben vannak. Erősen korrelált elemeket tartalmazó adatokon (például összetartozó termékcsomagoknál) a zárt halmazok is jelentős tömörítést adnak.

6.7.4. Riasztássorozatok a Pannon üzemben

> **FUTÓ ESET · PANNON ÜZEM**
>
> A Pannon üzem riasztásnaplója időbélyeggel ellátott eseményekből áll. Arra vagyunk kíváncsiak, mely riasztások követik rendszeresen egymást rövid időn belül, mert az ilyen láncok egy közös kiváltó okra utalhatnak, és a kezelő a lánc első elemére reagálva megelőzheti a többit.

Egyszerű, de hatékony módszer a **rendezett párok** és hármasok számlálása egy időablakon belül. Minden A riasztásnál megnézzük, milyen riasztások követték a következő $w = 3$ órán belül; így megkapjuk az „A után w-n belül B” sorozatszabály megbízhatóságát, és ezt összevetjük B alapgyakoriságával (liftjével).

```
plant = ab.load_plant()
R = plant.alarms.copy()
w = pd.Timedelta(hours=3)
ido, nev = R.ido.to_numpy(), R.riasztas.to_numpy()
parok = []
for i in range(len(R)):
    kovetok = set(nev[(ido > ido[i]) & (ido <= ido[i] + w)]) - {nev[i]}
    parok += [(nev[i], b) for b in kovetok]
parok = pd.DataFrame(parok, columns=["A", "B"])
nA = R.riasztas.value_counts()
# B alapgyakorisága: a 3 órás ablakok mekkora hányadában fordul elő B
ablakok = pd.date_range(R.ido.min().floor("3h"), R.ido.max(), freq="3h")
alap = {b: np.isin(ablakok, R.loc[R.riasztas == b,
 ↪  "ido"].dt.floor("3h").unique()).mean() for b in nA.index}
sorozat = (parok.value_counts().rename("db").reset_index()
           .assign(megbizhatosag=lambda d: d.db / d.A.map(nA),
                   lift=lambda d: d.megbizhatosag / d.B.map(alap)))
```

```
sorozat[sorozat.db >= 8].sort_values("db",
↪  ascending=False).head(10).round(2)
```

	A	B	db	megbizhatosag	lift
0	HUTOVIZ_MELEG	HUTES_TELITETT	17	0.24	8.65
1	T_MAGAS	MINOSEG_ELTERES	16	0.22	3.31
2	HUTES_TELITETT	HUTOVIZ_OSZCILLACIO	13	0.14	4.00
3	HUTES_TELITETT	HUTOVIZ_MELEG	13	0.14	6.96
4	T_MAGAS	HUTOVIZ_OSZCILLACIO	12	0.16	4.65
5	HUTOVIZ_OSZCILLACIO	HUTES_TELITETT	9	0.09	3.20
6	HUTOVIZ_MELEG	T_MAGAS	9	0.12	5.50
7	HUTES_TELITETT	MINOSEG_ELTERES	9	0.10	1.48
8	HUTOVIZ_OSZCILLACIO	MINOSEG_ELTERES	8	0.08	1.17
9	MINOSEG_ELTERES	T_MAGAS	8	0.04	1.86

```
harmasok = []
for i in range(len(R)):
    utana = R[(R.ido > R.ido.iat[i]) & (R.ido <= R.ido.iat[i] +
↪  w)].riasztas.tolist()
    for j, b in enumerate(utana):
        for c in utana[j + 1:]:
            if len({nev[i], b, c}) == 3:
                harmasok.append((nev[i], b, c))
pd.Series([" → ".join(h) for h in
↪  harmasok]).value_counts().head(6).to_frame("előfordulás")
```

	előfordulás
MINOSEG_ELTERES → HUTOVIZ_MELEG → T_MAGAS	3
HUTOVIZ_OSZCILLACIO → HUTES_TELITETT → T_MAGAS	3
HUTOVIZ_MELEG → HUTES_TELITETT → T_MAGAS	3
HUTOVIZ_MELEG → T_MAGAS → HUTES_TELITETT	3
HUTES_TELITETT → HUTOVIZ_OSZCILLACIO → BETAP_KONC_ALACSONY	2
HUTOVIZ_MELEG → T_MAGAS → MINOSEG_ELTERES	2

A hármas sorozatok ritkák – egy év alatt csak néhány teljes lánc fordul elő –, a párok azonban egyértelmű képet adnak, és fizikailag értelmezhetők: a meleg hűtővizet a hűtés telítődése és a reaktorhőmérséklet emelkedése követi, a magas hőmérsékletet pedig minőségi eltérés. Más párok mögött mérési vagy szabályozási jelenség áll: a hűtővízszelep akadozásakor az oszcilláló szelepállás időnként eléri a határt, ezért a „hűtés telített” és a „hűtővíz-oszcilláció” riasztás együtt jár, holott a hűtés valójában nem telített. A sorozatminták így nemcsak a riasztási „árvizek” csökkentésére (a következményként fellépő riasztások elnyomására), hanem a kiváltó ok gyors azonosítására és a félrevezető riasztások felderítésére is használhatók. A 15. fejezetben ezt a gondolatot a hibadiagnosztikával kapcsoljuk össze.

Interaktív alkalmazás: Apriori lépésről lépésre: támogatottság, megbízhatóság, lift

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/apriori.html

Colab-notebook: 6. fejezet – asszociációs szabályok

6.8. Összefoglalás

- Az asszociációs szabályok keresése két lépés: a gyakori elemhalmazok feltárása és a megbízható szabályok előállítása; a számításigényes rész az első.
- A támogatottság anti-monoton: egy ritka elemhalmaz minden bővítése ritka. Ez teszi lehetővé az Apriori-algoritmus metszését.
- Az Eclat vertikális tid-listákkal, az FP-growth egy tömör prefixfával, jelöltképzés nélkül dolgozik; nagy adatbázisokon ezek a hatékonyabbak.
- A zárt gyakori elemhalmazok a támogatottságokat is megőrző, a maximálisak még tömörebb, de veszteséges reprezentációt adnak.
- A megbízhatóság a következmény alapgyakoriságát nem veszi figyelembe; a lift, a leverage és a conviction ezt korrigálja. A szabályok együttes előfordulást, nem okozatot írnak le.
- A módszer kiterjeszthető fogalmi hierarchiákra, numerikus attribútumokra és időrendezett eseménysorozatokra.

Ellenőrző kérdések

1. Mit jelent, hogy a támogatottság anti-monoton? Miért nem az a megbízhatóság általában?
2. Hány jelöltet állít elő az Apriori a második szinten, ha $|F_1| = 40$?
3. Mi a különbség a zárt és a maximális gyakori elemhalmaz között? Melyikből állíthatók vissza a támogatottságok?
4. Mikor lehet egy 90%-os megbízhatóságú szabály érdektelen?
5. Miért nem következik egy erős asszociációs szabályból, hogy az egyik termék ajánlása növelni fogja a másik eladását?

Feladatok

1. **Kézi Apriori.** Végezze el a mini-példát $\sigma_{\min} = 2$ küszöbbel! Állítsa elő a {kávé, tej, cukor} gyakori elemhalmazból képezhető összes szabályt, és számítsa ki a megbízhatóságukat és a liftjüket!

2. **FP-fa.** Rajzolja meg a mini-példa FP-fáját $\sigma_{\min} = 2$ esetén! Mi a „keksz” feltételes mintabázisa?
3. **Küszöbök hatása.** Ábrázolja a PannonShop gyakori elemhalmazainak és szabályainak számát a támogatottsági küszöb függvényében! Hol jelennek meg a beültetett szabályok?
4. **Szezonális szabályok.** Keressen szabályokat külön a decemberi és a többi hónap kosaraiban! Mely szabályok jellemzőek csak a karácsonyi időszakra?
5. **Időablak.** Vizsgálja meg, hogyan változnak a riasztássorozatok megbízhatóságai és liftjei 30 perces, 2 órás és 6 órás időablak esetén! Melyik ablak adja a fizikailag legjobban értelmezhető láncokat?

További olvasmányok

Az asszociációs szabályok és az Apriori eredeti leírása Agrawal, Imieliński, és mtsai. (1993) és Agrawal & Srikant (1994). A mintanövelő módszerekről J. Han és mtsai. (2000), a vertikális algoritmusokról Zaki (2000) ír. Az érdekességi mértékek összehasonlítását Tan és mtsai. (2004) végezte el. A terület átfogó tárgyalása J. Han és mtsai. (2022) két mintabányászati fejezetében olvasható. A riasztásnaplók sorozatbányászatáról Dorgo & Abonyi (2018), az ipari riasztáskezelés problémáiról J. Wang és mtsai. (2016) ad áttekintést.

III. Prediktív modellezés

7. A felügyelt tanulás alapjai és a modellek értékelése

Miről szól a fejezet?
Mielőtt a regressziós és osztályozó modellek családjait sorra vennénk, tisztázzuk, mit jelent, hogy egy modell „jó”, és hogyan mérhető ez megbízhatóan. A fejezet a felügyelt tanulás általános keretéből indul: az általánosítási hiba, a túltanulás és a torzítás–variancia felbontás fogalmaiból. Ezután a modellek kiértékelésének gyakorlati eszközeit tárgyalja: a tanító-, validációs és teszthalmazokra bontást, a keresztvalidációt (az idősoros és a csoportos változatokkal együtt), a regressziós és az osztályozási mérőszámokat, a ROC- és PR-görbéket, a döntési küszöb költségalapú megválasztását, a valószínűségek kalibrációját és a kiegyensúlyozatlan osztályok kezelését.
A fejezet végére az olvasó képes lesz egy modell várható teljesítményét torzítatlanul becsülni; az adatok szerkezetének megfelelő validációs sémát választani; a feladathoz illő mérőszámot kiválasztani és értelmezni; és a döntési küszöböt az üzleti költségek alapján megállapítani.

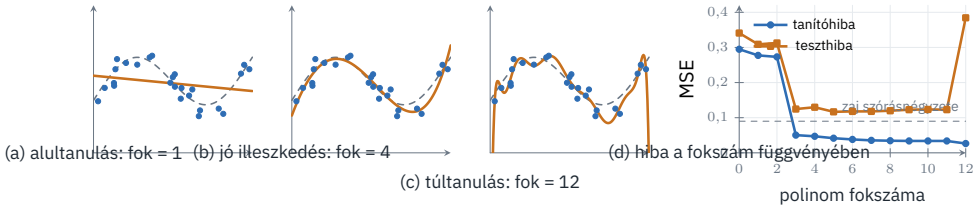
7.1. Az általánosítás problémája

A 1. fejezetben a felügyelt tanulást a tanítóadatokon számolt átlagos veszteség minimalizálásaként írtuk fel (1.3). A valódi cél azonban nem a tanítóadatok minél pontosabb visszaadása, hanem az, hogy a modell az ugyanabból a folyamatból származó *új* adatokon is jól teljesítsen. Ha az adatok egy $p(\mathbf{x}, y)$ együttes eloszlásból származnak, a modell **általánosítási hibája** (várható kockázata)

$$R(f) = \mathbb{E}_{(\mathbf{x},y)\sim p}\big[L(y, f(\mathbf{x}))\big], \tag{7.1}$$

amelyet nem ismerünk, csak becsülhetünk. A tanítóhiba ennek *optimista* becslése, hiszen a modell paramétereit éppen ezekre az adatokra hangoltuk. Minél rugalmasabb a modellcsalád, annál nagyobb lehet a különbség.

A 7.1. ábra ezt egy egyszerű kísérlettel szemlélteti. 25 zajos mérési pontra egyre magasabb fokszámú polinomot illesztünk. A tanítóhiba a fokszámmal monoton csökken – egy elég magas fokú polinom minden ponton áthaladhat –, a független teszthalmazon mért hiba viszont egy darabig csökken, majd növekedni kezd. Az alacsony fokszámú modell **alultanul** (nem képes leírni a valódi összefüggést), a magas fokszámú **túltanul** (a zajt is megtanulja).



7.1. ábra. A modellkomplexitás hatása. (a)–(c) Elsőfokú, negyedfokú és tizenkettedfokú polinom illesztése ugyanarra a 25 zajos pontra (szaggatott vonal: a valódi függvény). (d) A tanító- és a teszthiba a fokszám függvényében; a szaggatott vízszintes vonal a zaj szórásnégyzete, amely alá a teszthiba nem csökkenhet.

7.1.1. A torzítás–variancia felbontás

A jelenség négyzetes veszteség esetén pontosan elemezhető (Geman és mtsai., 1992; Hastie és mtsai., 2009). Tegyük fel, hogy $y = g(\mathbf{x}) + \varepsilon$, ahol ε nulla várható értékű, σ^2 szórásnégyzetű zaj, és a $\hat{f}$ modellt egy véletlen tanítóhalmazon becsüljük. Egy rögzített $\mathbf{x}_0$ pontban a várható négyzetes hiba – a zaj és a tanítóhalmaz véletlenségére vett várható érték – három részre bomlik:

$$\mathbb{E}\big[(y_0 - \hat{f}(\mathbf{x}_0))^2\big] = \underbrace{\sigma^2}_{\text{csökkenthetetlen zaj}} + \underbrace{\big(\mathbb{E}[\hat{f}(\mathbf{x}_0)] - g(\mathbf{x}_0)\big)^2}_{\text{torzítás}^2} + \underbrace{\mathbb{E}\big[(\hat{f}(\mathbf{x}_0) - \mathbb{E}[\hat{f}(\mathbf{x}_0)])^2\big]}_{\text{variancia}}. \tag{7.2}$$

A **torzítás** azt méri, mennyire tér el a modell *átlagos* előrejelzése (sok lehetséges tanítóhalmazon) a valódi függvénytől; a **variancia** azt, hogy mennyire ingadozik az előrejelzés tanítóhalmazról tanítóhalmazra. Az egyszerű modellek torzítása nagy, varianciája kicsi; a rugalmas modelleké fordítva. A jó modell a kettő közötti egyensúlyt találja meg. A (7.2) felbontás levezetéséhez az $y_0 - \hat{f} = (y_0 - g) + (g - \mathbb{E}\hat{f}) + (\mathbb{E}\hat{f} - \hat{f})$ összeget kell négyzetre emelni; a vegyes szorzatok várható értéke nulla, mert a zaj független a tanítóhalmaztól, és $\mathbb{E}[\mathbb{E}\hat{f} - \hat{f}] = 0$.

A felbontás a gyakorlatban két irányt jelöl ki. A torzítást rugalmasabb modellcsaláddal, jobb jellemzőkkel csökkenthetjük; a varianciát több adattal, egyszerűbb modellel, **regularizációval** (8. fejezet) vagy sok modell átlagolásával (9. fejezet, együttes modellek).

7.2. Adatfelosztás és validáció

7.2.1. Tanító-, validációs és teszthalmaz

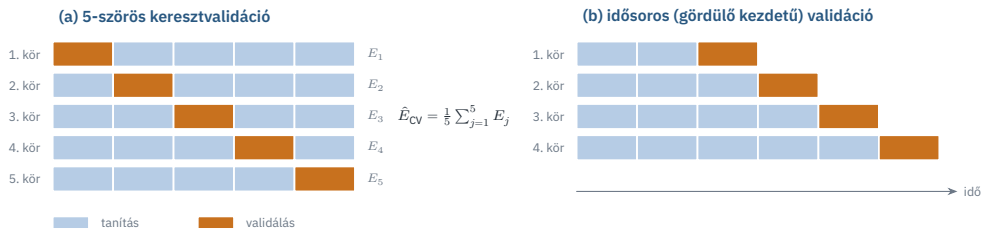
Az általánosítási hiba becsléséhez olyan adatok kellenek, amelyeket a modell a tanítás során nem látott. A legegyszerűbb a **visszatartásos** (hold-out) módszer: az adatokat véletlenszerűen tanító- és teszthalmazra bontjuk (például 80–20%-os arányban). Ha a modellnek hangolandó **hiperparaméterei** is vannak (például a polinom fokszáma, egy fa mélysége), a teszthalmaz nem használható a hangolásra, különben az is optimistává válik. Ilyenkor háromfelé bontunk: a **tanítóhalmazon** illesztjük a paramétereket, a **validációs halmazon** választjuk ki a hiperparamétereket, és a **teszthalmazt** csak egyszer, a végső modell értékelésére használjuk.

7.2.2. Keresztvalidáció

Kis adathalmazon a visszatartás pazarló, és a becslés erősen függ a véletlen felosztástól. A K**-szoros keresztvalidáció** (Stone, 1974) az adatokat K közel egyenlő részre (foldra) osztja, és K-szor tanít: mindig egy rész a validációs, a többi a tanítóhalmaz (7.2. ábra). Az általánosítási hiba becslése a K validációs hiba átlaga:

$$\hat{E}_{\text{CV}} = \frac{1}{K}\sum_{j=1}^{K}\frac{1}{|\mathcal{V}_j|}\sum_{k\in\mathcal{V}_j} L\big(y_k, \hat{f}^{(-j)}(\mathbf{x}_k)\big), \tag{7.3}$$

ahol $\mathcal{V}_j$ a j-edik validációs rész, $\hat{f}^{(-j)}$ pedig a nélküle tanított modell. A gyakorlatban a $K = 5$ vagy $K = 10$ jó kompromisszum a becslés torzítása és varianciája között (Kohavi, 1995). Osztályozásnál a **rétegzett** keresztvalidáció minden részben megőrzi az osztályarányokat; ez kiegyensúlyozatlan osztályoknál különösen fontos.



7.2. ábra. (a) Ötszörös keresztvalidáció: minden kör más részt tart vissza validálásra, a becslés a körök átlaga. (b) Idősoros validáció: mindig a múltból tanítunk, és a közvetlenül következő időszakon validálunk.

A véletlen felosztás feltételezi a függetlenséget

A keresztvalidáció azt feltételezi, hogy a megfigyelések függetlenek. Ha nem azok, a véletlen felosztás optimista becslést ad:

- **Idősorokban** a szomszédos időpontok erősen hasonlók; véletlen felosztásnál a modell „interpolál” a tanítóhalmaz közeli pontjai között, élesben viszont *előre* kell jeleznie. Ilyenkor a gördülő kezdetű (expanding window) validációt kell használni, ahol a modell mindig csak a múltból tanul (7.2. (b) ábra) (Bergmeir & Benítez, 2012).
- **Csoportos adatokban** (egy vevő több rekordja, egy gép több mérése) ugyanazon csoport rekordjai nem kerülhetnek a tanító- és a validációs halmazba is; ezt a csoportos keresztvalidáció biztosítja.

7.2.3. Hiperparaméter-hangolás és beágyazott keresztvalidáció

A hiperparamétereket rendszerint rácskereséssel (az összes kombináció kipróbálásával) vagy véletlen kereséssel választjuk; utóbbi sok hiperparaméter esetén gyakran hatékonyabb, mert a fontos hiperparaméterek több különböző értékét próbálja ki (Bergstra & Bengio, 2012). A kiválasztott modell keresztvalidációs hibája azonban már optimista, hiszen a legjobbat választottuk ki sok közül. Torzítatlan becsléshez **beágyazott** keresztvalidáció kell: egy külső ciklus becsli a teljesítményt, és minden külső körben egy belső keresztvalidáció hangolja a hiperparamétereket, kizárólag a külső kör tanítórészén.

7.3. Regressziós mérőszámok

Ha y_k a valódi, $\hat{y}_k$ az előrejelzett érték, a leggyakoribb mérőszámok:

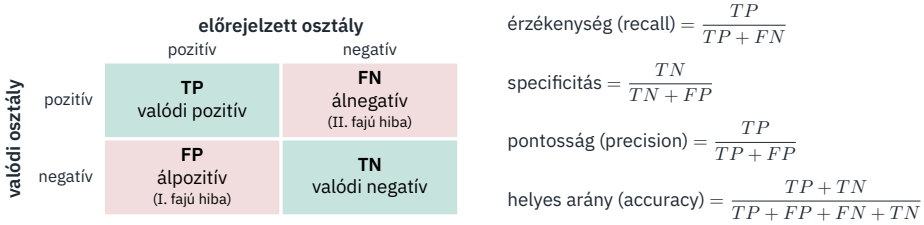
$$\text{MAE} = \frac{1}{N}\sum_k |y_k - \hat{y}_k|, \qquad \text{RMSE} = \sqrt{\frac{1}{N}\sum_k (y_k - \hat{y}_k)^2}, \qquad R^2 = 1 - \frac{\sum_k (y_k - \hat{y}_k)^2}{\sum_k (y_k - \bar{y})^2}. \tag{7.4}$$

Az MAE és az RMSE a célváltozó mértékegységében fejezi ki a hibát; az RMSE a nagy hibákat jobban bünteti. Az R^2 (determinációs együttható) azt méri, hogy a modell a célváltozó szórásnégyzetének mekkora hányadát magyarázza meg; az átlaggal való előrejelzés $R^2 = 0$-t ad, és új adatokon az R^2 negatív is lehet. A százalékos hibák (MAPE) könnyen értelmezhetők, de nulla közeli valódi értékeknél használhatatlanok.

7.4. Osztályozási mérőszámok

7.4.1. A tévesztési mátrix és a belőle származó mérőszámok

Kétosztályos feladatban az egyik osztályt – rendszerint a ritkábbat, a „keresettet” (lemorzsolódó vevő, hibás termék) – **pozitívnak**, a másikat negatívnak nevezzük. A **tévesztési mátrix** (keveredési mátrix, confusion matrix) a valódi és az előrejelzett osztályok szerint négy cellába sorolja az eseteket (7.3. ábra): valódi pozitív (TP), álnegatív (FN), álpozitív (FP), valódi negatív (TN).



7.3. ábra. A tévesztési mátrix és a belőle számolt fő mérőszámok. Az álpozitív a „téves riasztás”, az álnegatív a „kimaradt eset”.

A **helyes arány** (accuracy) a helyesen osztályozott esetek hányada. Kiegyensúlyozatlan osztályoknál félrevezető: ha a termékek 9%-a hibás, a „minden termék jó” triviális osztályozó is 91%-os helyes arányt ér el, miközben egyetlen hibát sem talál meg. Ezért az osztályonkénti mérőszámokat kell néznünk:

- **érzékenység** (recall, sensitivity, valódi pozitív arány): a pozitívak hányadát találja meg a modell;
- **specificitás** (valódi negatív arány): a negatívak hányadát sorolja helyesen;

- **pontosság** (precision): a pozitívnak jelzett esetek hányada valóban pozitív;
- az **F1-mérték** a pontosság és az érzékenység harmonikus közepe, $F_1 = 2PR/(P+R)$;
- a **kiegyensúlyozott helyes arány** az érzékenység és a specificitás átlaga.

A magyar szakirodalomban a *precision* fordítása nem egységes (pontosság, megbízhatóság, precizitás); e könyvben a *pontosság* a precision, a *helyes arány* az accuracy megfelelője. Többosztályos feladatban a mérőszámokat osztályonként számoljuk, és átlagoljuk: a *makroátlag* minden osztályt egyenlő súllyal, a *súlyozott átlag* az osztályok gyakoriságával vesz figyelembe.

7.4.2. A ROC-görbe és az AUC

A legtöbb osztályozó nem csak osztálycímkét, hanem egy **pontszámot** (például a pozitív osztály becsült valószínűségét) ad, és a címkét egy küszöbérték határozza meg. A küszöb csökkentésével több esetet jelzünk pozitívnak: nő az érzékenység, de nő az álpozitívak aránya is. A **ROC-görbe** (receiver operating characteristic) az összes lehetséges küszöbre ábrázolja a valódi pozitív arányt az álpozitív arány (1 − specificitás) függvényében (Fawcett, 2006). A véletlen találgatás az átlót adja, a tökéletes osztályozó a bal felső sarkon halad át.

A görbe alatti terület (**AUC**) egyetlen számban foglalja össze az osztályozó rangsorolási képességét. Fontos értelmezése: az AUC annak a valószínűsége, hogy egy véletlenszerűen választott pozitív eset pontszáma nagyobb, mint egy véletlenszerűen választott negatívé:

$$\mathrm{AUC} = P\big(\hat{s}(\mathbf{x}^{+}) > \hat{s}(\mathbf{x}^{-})\big). \tag{7.5}$$

Az AUC független a küszöbtől és az osztályaránytól. Erősen kiegyensúlyozatlan feladatokban azonban túl kedvező képet adhat, mert a sok negatív eset miatt az álpozitív arány akkor is kicsi marad, ha az álpozitívak száma nagy. Ilyenkor informatívabb a **pontosság–érzékenység (PR) görbe** és az alatta lévő terület (átlagos pontosság) (T. Saito & Rehmsmeier, 2015). A PR-görbén a véletlen osztályozó szintje a pozitív osztály aránya.

7.4.3. A döntési küszöb megválasztása

Az alapértelmezett 0,5-ös küszöb ritkán optimális. A küszöbválasztás üzleti döntés: a két hibatípus költsége általában nagyon különböző. Ha egy lemorzsolódó vevő elvesztése C_{FN}, egy fölösleges megtartási ajánlat C_{FP} költséggel jár, és a modell valószínűségei jól kalibráltak, akkor az a döntés minimalizálja a várható költséget, amely pozitívnak jelöl minden esetet, ahol

$$\hat{p}(\mathbf{x})\, C_{FN} > (1 - \hat{p}(\mathbf{x}))\, C_{FP} \quad \Longleftrightarrow \quad \hat{p}(\mathbf{x}) > \frac{C_{FP}}{C_{FP} + C_{FN}}. \tag{7.6}$$

Ha a kimaradt eset tízszer drágább a téves riasztásnál, az optimális küszöb $1/11 \approx 0{,}09$ – messze a 0,5 alatt. A (7.6) képlet csak jól kalibrált valószínűségekre érvényes; egyébként a küszöböt egy validációs halmazon, a várható költség közvetlen minimalizálásával kell meghatározni.

7.4.4. Kalibráció

Egy osztályozó **kalibrált**, ha a becsült valószínűségei a valódi relatív gyakoriságokat tükrözik: azok közül az esetek közül, amelyekre 0,3-et becsül, nagyjából 30% legyen pozitív. A kalibrációt **megbízhatósági diagrammal** ellenőrizzük (a becsült valószínűségek intervallumaiban ábrázoljuk a tényleges pozitív arányt), a valószínűségek összesített minőségét pedig a **Brier-pontszámmal**, $\frac{1}{N}\sum_k (\hat{p}_k - y_k)^2$ (Brier, 1950). Sok népszerű modell (például a döntési fák, a támaszvektorgépek, a naiv Bayes) rosszul kalibrált valószínűségeket ad; ezek utólag kalibrálhatók (Platt-skálázás, izotonikus regresszió) (Niculescu-Mizil & Caruana, 2005).

7.4.5. Kiegyensúlyozatlan osztályok

Ha a pozitív osztály ritka, a tanulóalgoritmus a többségi osztályt „preferálja”. A fő kezelési lehetőségek (H. He & Garcia, 2009): (1) a költségek beépítése a tanításba (osztálysúlyozás, a `scikit-learn` `class_weight` paramétere); (2) újramintavételezés: a ritka osztály felülmintavételezése vagy a gyakori alulmintavételezése, illetve szintetikus pozitív esetek előállítása (SMOTE) (Chawla és mtsai., 2002); (3) a döntési küszöb eltolása. Az újramintavételezést csak a tanítóhalmazon szabad végezni, a validációs és a teszthalmaz eredeti eloszlását meg kell őrizni. A gyakorlatban a küszöb megfelelő megválasztása és a súlyozás gyakran ugyanannyit ér, mint az újramintavételezés.

7.5. A modellek összehasonlítása

Két modell keresztvalidációs eredményeinek különbsége lehet véletlen is. Ugyanazokon a foldokon kiértékelt modellek eredményei páronként összehasonlíthatók, de a foldok eredményei nem függetlenek (a tanítóhalmazok átfednek), ezért a szokásos t-próba túl gyakran mutat ki „szignifikáns” különbséget. Több adathalmazon végzett összehasonlításra nemparaméteres próbák ajánlottak (Demšar,

2006). A legfontosabb gyakorlati szabály azonban egyszerűbb: **mindig hasonlítsunk egy egyszerű viszonyítási modellhez** (többségi osztály, átlag, előző érték), mert csak így derül ki, mennyit ér a bonyolult modell.

7.6. Gyakorlat: lemorzsolódás és termékminőség

7.6.1. A PannonShop lemorzsolódási feladata

> **FUTÓ ESET · PANNONSHOP**
>
> A lemorzsolódás-előrejelzés első és legfontosabb lépése a célváltozó pontos definiálása. A 2025. július 1-jei határnapot választjuk: a jellemzőket kizárólag az ezt megelőző vásárlásokból számoljuk, és azokat a vevőket tekintjük lemorzsolódónak, akik a megelőző évben vásároltak, de a határnapot követő fél évben nem. Így a tanítás során a modell pontosan azt az információt kapja, amely egy valós, július 1-jén meghozott döntéskor rendelkezésre állna.

```
churn = ab.churn_table(cutoff="2025-07-01", horizon_days=183)
jell = ["R_nap", "F_szamla", "M_ertek", "atl_kosarertek", "atl_kosarmeret",
 ↪  "atl_mennyiseg",
        "elso_vasarlas_nap", "akcios_arany", "arany_kave", "arany_gep",
         ↪  "arany_konyha",
        "arany_dekor", "arany_ajandek", "arany_iroszer"]
X, y = churn[jell], churn.lemorzsolodik
print(f"aktív vevők: {len(y)}, lemorzsolódási arány: {y.mean():.1%}")
```

```
aktív vevők: 1979, lemorzsolódási arány: 42.4%
```

Egy logisztikus regressziós modellt (9. fejezet) hasonlítunk össze a többségi osztályt előrejelző viszonyítási modellel, rétegzett 5-szörös keresztvalidációval. Az előfeldolgozás a folyamat része, így szivárgásmentes (3.7. alfejezet).

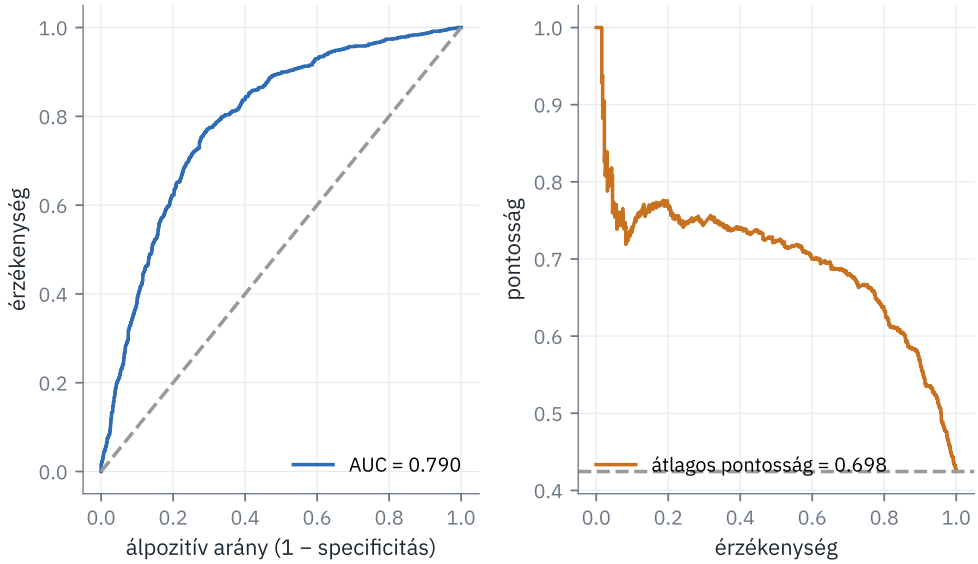
```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import FunctionTransformer, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.dummy import DummyClassifier
from sklearn.model_selection import StratifiedKFold, cross_validate,
 ↪  cross_val_predict
```

```
elo = make_pipeline(SimpleImputer(strategy="median"),
↪  FunctionTransformer(np.log1p), StandardScaler())
modellek = {
    "többségi osztály": DummyClassifier(strategy="most_frequent"),
    "logisztikus regresszió": make_pipeline(elo,
    ↪  LogisticRegression(max_iter=2000)),
}
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
metrikak = ["accuracy", "balanced_accuracy", "precision", "recall", "f1",
↪  "roc_auc", "average_precision"]
pd.DataFrame({nev: {m: cross_validate(mod, X, y, cv=cv,
↪  scoring=metrikak)[f"test_{m}"].mean()
                    for m in metrikak} for nev, mod in
                    ↪  modellek.items()}).round(3)
```

	többségi osztály	logisztikus regresszió
accuracy	0.576	0.728
balanced_accuracy	0.500	0.719
precision	0.000	0.687
recall	0.000	0.661
f1	0.000	0.674
roc_auc	0.500	0.790
average_precision	0.424	0.700

A többségi osztályt előrejelző modell helyes aránya is jelentős, mert a vevők többsége nem morzsolódik le; az érzékenysége és a pontossága viszont nulla. A logisztikus regresszió minden mérőszámon jobb, az AUC alapján rangsorolási képessége jó.

```
from sklearn.metrics import roc_curve, precision_recall_curve,
↪  roc_auc_score, average_precision_score
p_cv = cross_val_predict(modellek["logisztikus regresszió"], X, y, cv=cv,
↪  method="predict_proba")[:, 1]
fpr, tpr, _ = roc_curve(y, p_cv)
prec, rec, _ = precision_recall_curve(y, p_cv)
fig, ax = plt.subplots(1, 2)
ax[0].plot(fpr, tpr, color=ab.PALETTE[0], label=f"AUC = {roc_auc_score(y,
↪  p_cv):.3f}")
ax[0].plot([0, 1], [0, 1], ls="--", color="0.6")
ax[0].set_xlabel("álpozitív arány (1 – specificitás)");
↪  ax[0].set_ylabel("érzékenység"); ax[0].legend(loc="lower right")
ax[1].plot(rec, prec, color=ab.PALETTE[1], label=f"átlagos pontosság =
↪  {average_precision_score(y, p_cv):.3f}")
ax[1].axhline(y.mean(), ls="--", color="0.6")
ax[1].set_xlabel("érzékenység"); ax[1].set_ylabel("pontosság");
↪  ax[1].legend(loc="lower left")
fig.tight_layout()
```



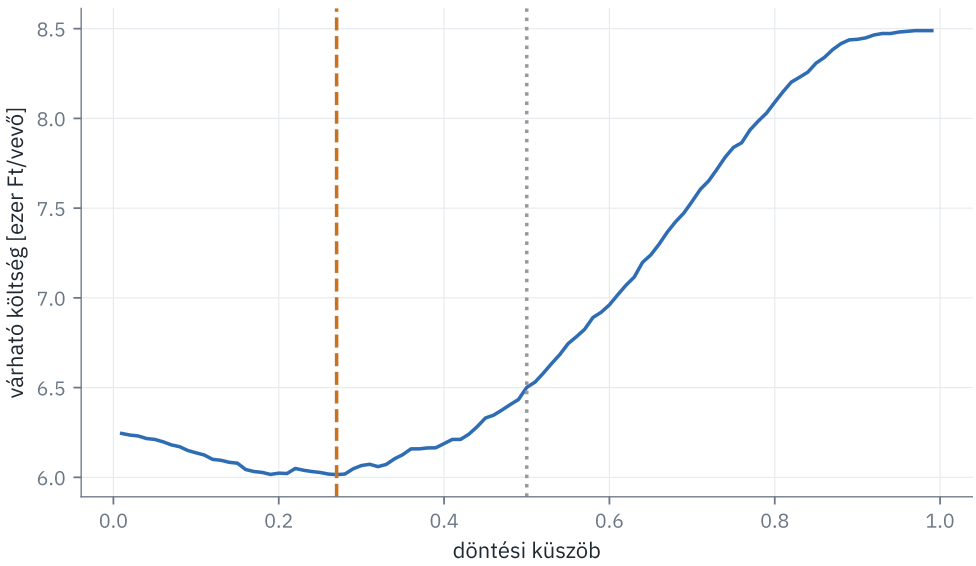
7.4. ábra. A lemorzsolódás-előrejelző logisztikus regresszió ROC-görbéje (balra) és pontosság–érzékenység görbéje (jobbra), keresztvalidált előrejelzésekből. A szaggatott vonal a véletlen osztályozó szintje.

Tegyük fel, hogy egy megtartási ajánlat (kupon) költsége 2 000 Ft, egy lemorzsolódó vevő elvesztése pedig átlagosan 20 000 Ft kiesett fedezetet jelent, és az ajánlat a megkeresett lemorzsolódók felét tartja meg. Ekkor egy álpozitív 2 000 Ft, egy álnegatív 20 000 Ft veszteség, egy valódi pozitív pedig $2\,000 + 0{,}5 \cdot 20\,000 = 12\,000$ Ft. A várható költséget a küszöb függvényében a keresztvalidált valószínűségekből számoljuk:

```
C_FP, C_FN, C_TP = 2_000, 20_000, 2_000 + 0.5 * 20_000
kuszobok = np.linspace(0.01, 0.99, 99)
koltseg = []
for t in kuszobok:
    jelolt = p_cv >= t
    tp, fp = np.sum(jelolt & (y == 1)), np.sum(jelolt & (y == 0))
    fn = np.sum(~jelolt & (y == 1))
    koltseg.append((tp * C_TP + fp * C_FP + fn * C_FN) / len(y))
koltseg = np.array(koltseg)
t_opt = kuszobok[koltseg.argmin()]
fig, ax = plt.subplots()
ax.plot(kuszobok, koltseg / 1000, color=ab.PALETTE[0])
ax.axvline(t_opt, color=ab.PALETTE[1], ls="--"); ax.axvline(0.5,
 ↪  color="0.6", ls=":")
```

```
ax.set_xlabel("döntési küszöb"); ax.set_ylabel("várható költség [ezer
↪  Ft/vevő]")
fig.tight_layout()
ft = lambda v: f"{v:,.0f}".replace(",", " ")
print(f"optimális küszöb: {t_opt:.2f};  költség: {ft(koltseg.min())}
↪  Ft/vevő "
      f"(a 0,5-ös küszöbnél {ft(koltseg[49])} Ft/vevő)")
```

```
optimális küszöb: 0.27;  költség: 6 015 Ft/vevő (a 0,5-ös küszöbnél 6 501 Ft/vevő)
```

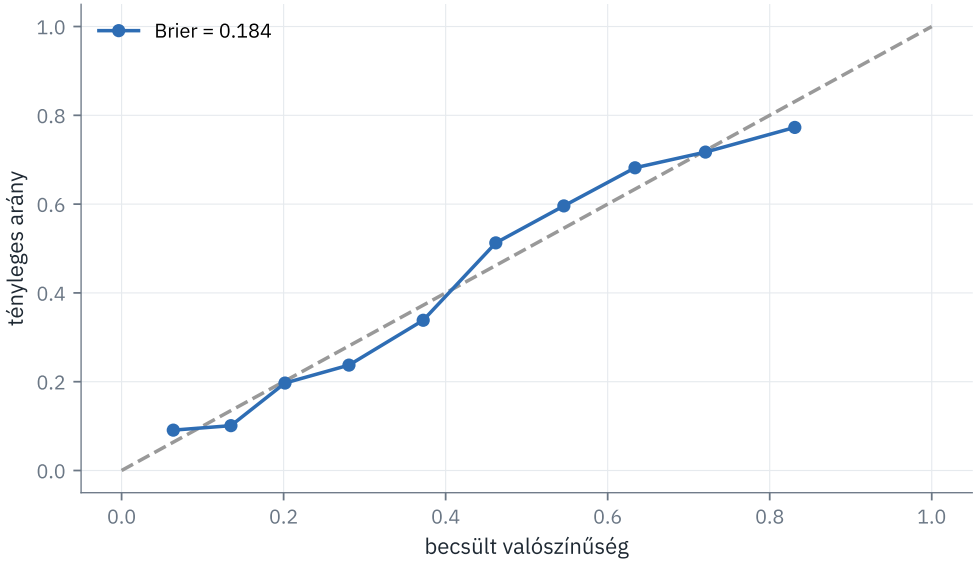


7.5. ábra. A lemorzsolódás-kezelés várható költsége vevőnként a döntési küszöb függvényében. A minimum messze a 0,5-ös alapértelmezés alatt van, mert a kimaradt lemorzsolódó vevő sokkal drágább a fölösleges ajánlatnál.

A költségoptimális küszöb jóval kisebb a 0,5-nél, és a megtakarítás vevőnként érezhető. Ugyanaz a modell tehát a küszöbtől függően nagyon különböző üzleti eredményt adhat – a modell kiértékelését mindig a felhasználási döntéssel együtt kell végezni.

```
from sklearn.calibration import calibration_curve
from sklearn.metrics import brier_score_loss
frac, mean_p = calibration_curve(y, p_cv, n_bins=10, strategy="quantile")
fig, ax = plt.subplots()
ax.plot([0, 1], [0, 1], ls="--", color="0.6")
```

```
ax.plot(mean_p, frac, marker="o", color=ab.PALETTE[0], label=f"Brier =
↪  {brier_score_loss(y, p_cv):.3f}")
ax.set_xlabel("becsült valószínűség"); ax.set_ylabel("tényleges arány");
↪  ax.legend(loc="upper left")
fig.tight_layout()
```



7.6. ábra. A lemorzsolódási valószínűségek megbízhatósági diagramja: a becsült valószínűségek tizedeiben a tényleges lemorzsolódási arány. A logisztikus regresszió jól kalibrált, a pontok közel vannak az átlóhoz.

7.6.2. A Pannon üzem termékminőségi feladata

FUTÓ ESET · PANNON ÜZEM

A laborminták mintegy 9%-a nem felel meg a specifikációnak. Szeretnénk a laboreredmény megérkezése előtt, a folyamatváltozókból előre jelezni a nem megfelelő tételeket. Ez egy kiegyensúlyozatlan osztályozási feladat, amelyben ráadásul az adatok időbeli sorrendje is számít.

```
q = ab.quality_table()
vlt = ["T_kornyezet", "F_betap", "c_betap", "T_reaktor", "F_hutoviz",
↪  "T_hutoviz",
```

```
        "P_reaktor", "P_kevero", "E_fogyasztas", "dT"]
Xq, yq = q[vlt], (~q.megfelelo).astype(int)              # pozitív osztály:
↪  nem megfelelő
print(f"minták: {len(yq)}, nem megfelelő arány: {yq.mean():.1%}")
osztalyozo = make_pipeline(StandardScaler(),
↪  LogisticRegression(max_iter=2000))
sulyozott = make_pipeline(StandardScaler(),
↪  LogisticRegression(max_iter=2000, class_weight="balanced"))
```

```
minták: 2176, nem megfelelő arány: 8.7%
```

Először azt vizsgáljuk meg, mennyire számít a validációs séma. A véletlen keresztvalidációt összevetjük az idősoros (gördülő kezdetű) validációval:

```
from sklearn.model_selection import KFold, TimeSeriesSplit
from sklearn.ensemble import RandomForestClassifier
erdo = RandomForestClassifier(n_estimators=300, min_samples_leaf=3,
↪  random_state=0, n_jobs=-1)
semak = {"véletlen 5-szörös": KFold(5, shuffle=True, random_state=0),
↪  "idősoros 5-szörös": TimeSeriesSplit(5)}
sor = {}
for nev_m, mod in {"logisztikus regresszió": osztalyozo, "véletlen erdő":
↪  erdo}.items():
    for nev_s, s in semak.items():
        r = cross_validate(mod, Xq, yq, cv=s, scoring=["roc_auc",
↪  "average_precision"])
        sor[(nev_m, nev_s)] = {"AUC": r["test_roc_auc"].mean(), "átlagos
↪  pontosság": r["test_average_precision"].mean()}
pd.DataFrame(sor).T.round(3)
```

		AUC	átlagos pontosság
logisztikus regresszió	véletlen 5-szörös	0.591	0.292
	idősoros 5-szörös	0.550	0.168
véletlen erdő	véletlen 5-szörös	0.958	0.763
	idősoros 5-szörös	0.863	0.532

Két tanulság is kiolvasható a táblázatból. Egyrészt a véletlen erdő (9. fejezet) a véletlen felosztásnál lényegesen jobbnak tűnik, mint idősoros validációnál. Ennek oka az időbeli függőség: a véletlen felosztásnál egy validációs minta szomszédos, szinte azonos körülmények között vett mintái a tanítóhalmazban vannak, és a rugalmas modell „megjegyzi” ezeket a körülményeket. Az idősoros validáció a valós alkalmazást tükrözi: a modellnek a jövőbeli, esetleg korábban nem látott helyzetekre kell előre jeleznie. Másrészt a logisztikus regresszió itt gyenge: a minőségi eltérés a célértéktől mindkét irányban bekövetkezhet (túl kicsi és túl nagy konverzió, illetve túlmelegedés), amit egyetlen lineáris döntési határ nem tud leírni – erre a 9. fejezet nemlineáris modelljei adnak megoldást.

Végül a kiegyensúlyozatlanság kezelését vizsgáljuk a véletlen erdővel. Az utolsó negyedév adatai alkotják a teszthalmazt, a modellt az azt megelőző időszakon tanítjuk:

```
from sklearn.metrics import confusion_matrix
vagas = q.ido >= "2025-10-01"
Xtr, Xte, ytr, yte = Xq[~vagas], Xq[vagas], yq[~vagas], yq[vagas]
erdo_s = RandomForestClassifier(n_estimators=300, min_samples_leaf=3,
 ↪  random_state=0, n_jobs=-1,
                                 class_weight="balanced_subsample")
p_erdo = erdo.fit(Xtr, ytr).predict_proba(Xte)[:, 1]
p_suly = erdo_s.fit(Xtr, ytr).predict_proba(Xte)[:, 1]
for nev, p, t in [("küszöb 0,5", p_erdo, 0.5), ("küszöb 0,3", p_erdo, 0.3),
 ↪  ("küszöb 0,15", p_erdo, 0.15),
                  ("osztálysúlyozás, küszöb 0,5", p_suly, 0.5)]:
    tn, fp, fn, tp = confusion_matrix(yte, p >= t).ravel()
    print(f"{nev:28s} TP={tp:3d} FN={fn:3d} FP={fp:3d} TN={tn:4d}  "
          f"érzékenység={tp / (tp + fn):.2f}  pontosság={tp / max(tp + fp,
           ↪  1):.2f}")
```

```
küszöb 0,5                   TP= 10 FN= 29 FP=  1 TN= 508  érzékenység=0.26
 ↪  pontosság=0.91
küszöb 0,3                   TP= 16 FN= 23 FP=  6 TN= 503  érzékenység=0.41
 ↪  pontosság=0.73
küszöb 0,15                  TP= 32 FN=  7 FP= 34 TN= 475  érzékenység=0.82
 ↪  pontosság=0.48
osztálysúlyozás, küszöb 0,5  TP= 12 FN= 27 FP=  5 TN= 504  érzékenység=0.31
 ↪  pontosság=0.71
```

Az alapértelmezett 0,5-ös küszöbbel a modell a nem megfelelő tételeknek csak kis részét találja meg, igaz, nagy pontossággal. A küszöb csökkentése – és kisebb mértékben az osztálysúlyozás is – az érzékenységet növeli a pontosság rovására. Hogy melyik működési pont a jobb, az ismét a hibák költségétől függ: egy kimaradt selejtes tétel kiszállítása többnyire sokkal drágább, mint egy fölösleges soron kívüli laborvizsgálat.

Interaktív alkalmazás: Döntési küszöb, tévesztési mátrix, ROC- és PR-görbe

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/roc-kuszob.html

Colab-notebook: 7. fejezet – modellértékelés

7.7. Összefoglalás

- A cél az általánosítás: a tanítóhiba az általánosítási hiba optimista becslése; a túl rugalmas modell túltanul, a túl merev alultanul.
- Négyzetes veszteségre a várható hiba a zaj, a torzítás négyzete és a variancia összege (7.2).
- A hiperparaméterek hangolására validációs halmazt vagy keresztvalidációt, a végső értékelésre érintetlen teszthalmazt vagy beágyazott keresztvalidációt kell használni.
- Idősorokban és csoportos adatokban a véletlen felosztás optimista; ilyenkor idősoros, illetve csoportos validáció kell.
- Kiegyensúlyozatlan osztályoknál a helyes arány félrevezető; az érzékenység, a pontosság, az F1, a PR-görbe és a költségalapú értékelés informatívabb.
- A döntési küszöb üzleti döntés; kalibrált valószínűségek esetén a hibák költségéből közvetlenül számolható (7.6).

Ellenőrző kérdések

1. Miért optimista becslés a tanítóhiba? Mikor nagyobb az optimizmus?
2. Mit jelent a torzítás és a variancia? Hogyan változnak a modell rugalmasságának növelésével?
3. Miért nem szabad a teszthalmazon hangolni a hiperparamétereket?
4. Miért ad optimista becslést a véletlen keresztvalidáció idősorokon?
5. Egy osztályozó érzékenysége 0,9, specificitása 0,9, a pozitív osztály aránya 1%. Mekkora a pontossága?
6. Mi az AUC valószínűségi jelentése? Mikor informatívabb a PR-görbe?
7. Ha egy kimaradt hiba ötször drágább egy téves riasztásnál, mekkora a (7.6) szerinti optimális küszöb?

Feladatok

1. **Torzítás és variancia kísérletileg.** Ismételje meg a polinomillesztést 200 különböző véletlen tanítóhalmazon, és egy rögzített x_0 pontban becsülje meg a torzítást és a varianciát a fokszám függvényében! Ellenőrizze a (7.2) felbontást!
2. **Keresztvalidációs becslés szórása.** Futtassa a lemorzsolódási modell 5-szörös keresztvalidációját 20 különböző véletlen felosztással! Mekkora az AUC-becslés szórása? Hogyan változik 10-szeres keresztvalidációval?

3. **Csoportos validáció.** Készítsen olyan lemorzsolódási táblát, amelyben ugyanaz a vevő több határnaphoz (2025. január 1., április 1., július 1.) is szerepel! Hasonlítsa össze a véletlen és a vevőnkénti csoportos (`GroupKFold`) keresztvalidáció eredményét!
4. **Kalibráció.** Vizsgálja meg a véletlen erdő valószínűségeinek kalibrációját a minőségi feladaton, és kalibrálja őket izotonikus regresszióval (`CalibratedClassifierCV`)!
5. **Költségérzékenység.** Hogyan változik az optimális küszöb és a várható költség, ha a kupon csak a megkeresett lemorzsolódók 20%-át tartja meg?

További olvasmányok

A torzítás–variancia felbontás és a modellválasztás elméletét Hastie és mtsai. (2009) és James és mtsai. (2023) tárgyalja. A keresztvalidációról Kohavi (1995), az idősoros validációról Bergmeir & Benítez (2012) ad áttekintést. A ROC-analízis bevezetése Fawcett (2006), a PR-görbék előnyeiről T. Saito & Rehmsmeier (2015) ír. A kalibrációról Niculescu-Mizil & Caruana (2005), a kiegyensúlyozatlan tanulásról H. He & Garcia (2009) olvasható. Az osztályozók statisztikai összehasonlításának módszertanát Demšar (2006) foglalja össze.

8. Regresszió

Miről szól a fejezet?

A regresszió egy folytonos célváltozó előrejelzése a magyarázó változókból. A fejezet a legkisebb négyzetes lineáris regresszió levezetésével és geometriai értelmezésével kezdődik, majd a becslés statisztikai tulajdonságait és a modelldiagnosztikát tárgyalja. Ezután a sok, egymással korrelált változó problémájára a regularizált regressziót (ridge, lasszó, elasztikus háló) mutatjuk be, végül a nemlineáris összefüggések leírására a bázisfüggvényes (polinomiális, spline-) regressziót és a paramétereiben nemlineáris, fizikai ismereteket is beépítő („szürkedoboz") modelleket. A gyakorlati részben a Pannon üzem termékkoncentrációjára lágy szenzort építünk, és a PannonShop vevőinek jövőbeli költését jelezzük előre.

A fejezet végére az olvasó képes lesz a normálegyenleteket levezetni és megoldani; a regressziós együtthatókat és a diagnosztikai ábrákat értelmezni; a regularizációs paramétert keresztvalidációval megválasztani; és nemlineáris, akár fizikai alapú regressziós modellt illeszteni.

8.1. Lineáris regresszió

8.1.1. A modell és a legkisebb négyzetek elve

A **lineáris regressziós modell** a célváltozót a magyarázó változók lineáris kombinációjaként írja le:

$$y_k = \theta_0 + \theta_1 x_{k1} + \cdots + \theta_n x_{kn} + \varepsilon_k, \qquad k = 1, \dots, N, \tag{8.1}$$

ahol ε_k a modell által nem magyarázott hiba. Mátrixos alakban – az adatmátrixot egy csupa egyes oszloppal kiegészítve, hogy a tengelymetszet is a paramétervektor része legyen – $\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\varepsilon}$, ahol $\mathbf{X} \in \mathbb{R}^{N\times(n+1)}$. A paramétereket a **legkisebb négyzetek** módszerével becsüljük: azt a $\boldsymbol{\theta}$-t keressük, amely a hibák négyzetösszegét minimalizálja,

$$J(\boldsymbol{\theta}) = \sum_{k=1}^{N} \left(y_k - \mathbf{x}_k^\top \boldsymbol{\theta}\right)^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}). \tag{8.2}$$

Ez a (1.3) empirikuskockázat-minimalizálás négyzetes veszteséggel. A J a $\boldsymbol{\theta}$ konvex másodfokú függvénye; gradiensét nullává téve (a mátrixderiválás szabályait a B. függelék foglalja össze)

$$\nabla_{\boldsymbol{\theta}} J = -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) = \mathbf{0} \quad \Longrightarrow \quad \mathbf{X}^\top \mathbf{X}\, \hat{\boldsymbol{\theta}} = \mathbf{X}^\top \mathbf{y}. \tag{8.3}$$

Ezek a **normálegyenletek**. Ha $\mathbf{X}^\top\mathbf{X}$ invertálható (az oszlopok lineárisan függetlenek), a megoldás egyértelmű:

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}. \tag{8.4}$$

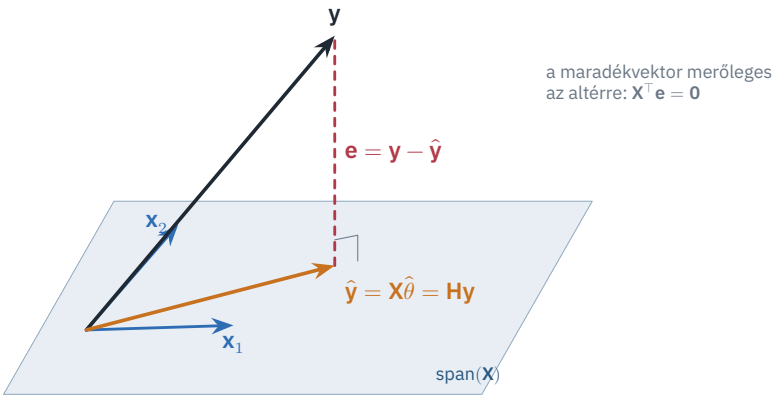
> **Numerikus megjegyzés**
>
> A (8.4) képlet a levezetés eredménye, nem számítási recept. Az $\mathbf{X}^\top\mathbf{X}$ mátrix kondíciószáma az $\mathbf{X}$ kondíciószámának négyzete, ezért közel kollineáris változóknál az invertálás numerikusan instabil. A szoftverek a feladatot az $\mathbf{X}$ QR- vagy szinguláris felbontásával oldják meg (Golub & Van Loan, 2013); NumPy-ban a `np.linalg.lstsq` függvény ezt teszi.

8.1.2. Geometriai értelmezés

A (8.3) egyenlet átrendezve $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}}) = \mathbf{0}$: a **maradékvektor** $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$ merőleges $\mathbf{X}$ minden oszlopára. Az illesztett értékek vektora, $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\theta}}$, tehát az $\mathbf{y}$ merőleges vetülete az $\mathbf{X}$ oszlopai által kifeszített altérre (8.1. ábra). A vetítést a

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top \tag{8.5}$$

kalapmátrix végzi ($\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$). A legkisebb négyzetes illesztés tehát az altér azon pontját választja, amely $\mathbf{y}$-hoz a legközelebb van.



8.1. ábra. A legkisebb négyzetes becslés geometriája: az illesztett értékek vektora a megfigyelések vektorának merőleges vetülete a magyarázó változók által kifeszített altérre; a maradék merőleges az altérre.

8.1.3. Kézi példa

Egyetlen magyarázó változó esetén a normálegyenletek zárt alakban megoldhatók: a meredekség $\hat{\theta}_1 = S_{xy}/S_{xx}$, ahol $S_{xy} = \sum_k (x_k - \bar{x})(y_k - \bar{y})$ és $S_{xx} = \sum_k (x_k - \bar{x})^2$, a tengelymetszet $\hat{\theta}_0 = \bar{y} - \hat{\theta}_1 \bar{x}$. A segédcsomag hat mérése a Pannon üzem hűtővízáramát és villamosenergia-fogyasztását tartalmazza:

```
d = ab.regresszio_mini()
x, y = d.F_hutoviz.to_numpy(), d.E_fogyasztas.to_numpy()
Sxx = np.sum((x - x.mean()) ** 2)
Sxy = np.sum((x - x.mean()) * (y - y.mean()))
t1 = Sxy / Sxx
t0 = y.mean() - t1 * x.mean()
print(f"x átlaga = {x.mean():.2f},  y átlaga = {y.mean():.3f}")
print(f"S_xx = {Sxx:.2f},  S_xy = {Sxy:.3f}")
print(f"meredekség = {t1:.4f} kW/(m³/h),  tengelymetszet = {t0:.3f} kW")
X1 = np.column_stack([np.ones_like(x), x])
print("lstsq:", np.round(np.linalg.lstsq(X1, y, rcond=None)[0], 4))
```

```
x átlaga = 4.50,  y átlaga = 69.600
S_xx = 17.50,  S_xy = 38.300
meredekség = 2.1886 kW/(m³/h),  tengelymetszet = 59.751 kW
lstsq: [59.7514  2.1886]
```

A modell szerint egységnyi (1 m³/h) hűtővízáram-növekedés mintegy 2,2 kW többletet villamos teljesítményt jelent – ez a hűtővízszivattyú fajlagos energiaigénye.

8.2. A becslés tulajdonságai és a modelldiagnosztika

8.2.1. Statisztikai tulajdonságok

Ha a hibák nulla várható értékűek, egymással korrelálatlanok és azonos σ^2 szórásnégyzetűek, a legkisebb négyzetes becslés torzítatlan, és a lineáris torzítatlan becslések között a legkisebb varianciájú (Gauss–Markov-tétel). A paraméterbecslés kovarianciamátrixa és a hibaszórásnégyzet torzítatlan becslése:

$$\operatorname{Cov}(\hat{\boldsymbol{\theta}}) = \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1}, \qquad \hat{\sigma}^2 = \frac{1}{N-n-1} \sum_{k=1}^{N} e_k^2. \tag{8.6}$$

Ha a hibák ráadásul normális eloszlásúak, a $\hat{\theta}_j / \mathrm{SE}(\hat{\theta}_j)$ hányados t-eloszlású, ami lehetővé teszi az együtthatókra vonatkozó konfidenciaintervallumok és hipotézisvizsgálatok (például $H_0 : \theta_j = 0$) végrehajtását (Montgomery és mtsai., 2021). A **korrigált determinációs együttható**, $\bar{R}^2 = 1 - (1 - R^2)\frac{N-1}{N-n-1}$, a változók számával bünteti az R^2-et, így változószámban eltérő modellek összehasonlítására alkalmasabb.

8.2.2. Diagnosztika

A becslés tulajdonságai a feltevéseken múlnak, ezért ezeket ellenőrizni kell. A legfontosabb eszközök a **maradékok ábrái**: a maradékok az illesztett értékek függvényében (görbült mintázat a nemlinearitás, tölcsér alak a nem állandó szórás jele), a maradékok normális Q–Q ábrája, idősorokban pedig a maradékok időbeli lefutása (autokorreláció). A **multikollinearitás** – a magyarázó változók közötti erős lineáris kapcsolat – az együtthatók becslését instabillá teszi (nagy standard hibák, előjelváltó együtthatók), az előrejelzést viszont nem feltétlenül rontja. Mérőszáma a **varianciainflációs tényező**, $\mathrm{VIF}_j = 1/(1 - R_j^2)$, ahol R_j^2 a j-edik változónak a többire illesztett regressziójából adódik; a 10 feletti érték erős kollinearitást jelez. A **nagy befolyású pontok** azonosítására a kalapmátrix átlójában lévő h_{kk} értékek (a pont „emelőereje”) és a Cook-féle távolság szolgál.

> **FUTÓ ESET · PANNON ÜZEM**
>
> Építsünk lineáris lágy szenzort a Pannon üzem termékkoncentrációjára! A célváltozó a 4 óránként mért laboreredmény, a magyarázó változók a mintavétel előtti óra átlagolt folyamatváltozói (7. fejezet). A `statsmodels` csomag a becslés teljes statisztikai összefoglalóját megadja.

```
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

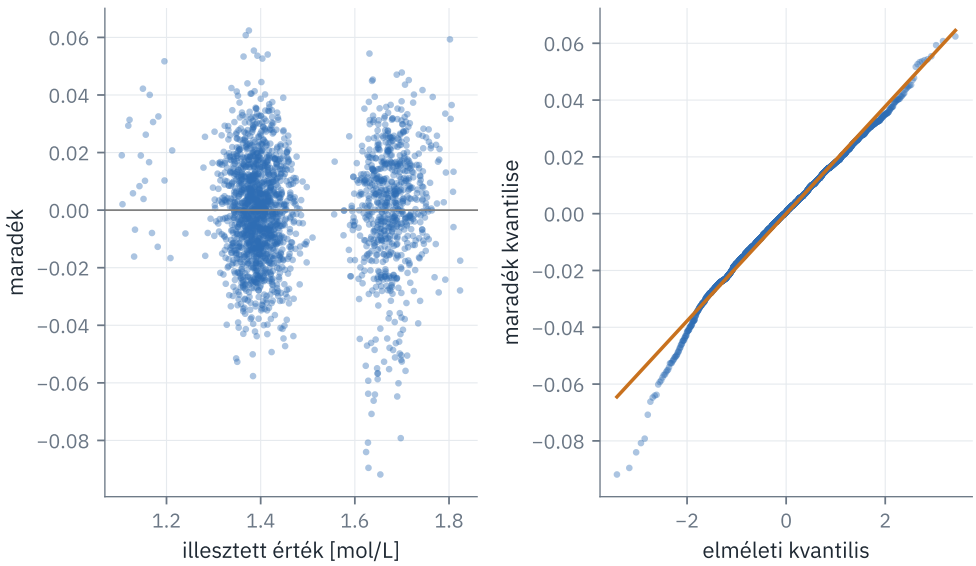
q = ab.quality_table()
valt = ["T_reaktor", "F_betap", "c_betap", "F_hutoviz", "T_hutoviz",
↪  "P_kevero", "E_fogyasztas"]
Xs = sm.add_constant(q[valt])
ols = sm.OLS(q.c_termek, Xs).fit()
tabla = pd.DataFrame({"együttható": ols.params, "std. hiba": ols.bse, "t":
↪  ols.tvalues,
                      "p": ols.pvalues, "VIF": [np.nan] +
                      ↪  [variance_inflation_factor(Xs.values, i)
                                                          for i in range(1,
                                                          ↪  Xs.shape[1
                                                          ↪  ])]})
print(f"R² = {ols.rsquared:.3f},  korrigált R² = {ols.rsquared_adj:.3f},
↪  σ̂ = {np.sqrt(ols.scale):.4f} mol/L")
tabla.round(4)
```

```
R² = 0.983,  korrigált R² = 0.982,  σ̂ = 0.0190 mol/L
```

	együttható	std. hiba	t	p	VIF
const	-1.8165	0.0299	-60.8038	0.0000	NaN
T_reaktor	0.0216	0.0001	148.3197	0.0000	1.3772
F_betap	-0.0363	0.0014	-25.8941	0.0000	8.6738
c_betap	0.7208	0.0075	95.6081	0.0000	9.8324
F_hutoviz	-0.0147	0.0011	-13.5842	0.0000	21.2274
T_hutoviz	0.0021	0.0002	9.7459	0.0000	4.3837
P_kevero	-0.0007	0.0008	-0.8532	0.3936	17.3277
E_fogyasztas	0.0061	0.0005	13.3635	0.0000	19.0516

```
from scipy import stats
fig, ax = plt.subplots(1, 2)
ax[0].scatter(ols.fittedvalues, ols.resid, s=4, alpha=0.4,
↪  color=ab.PALETTE[0])
ax[0].axhline(0, color="0.5", lw=0.8); ax[0].set_xlabel("illesztett érték
↪  [mol/L]"); ax[0].set_ylabel("maradék")
(osm, osr), (b, a0, _) = stats.probplot(ols.resid, dist="norm")
ax[1].scatter(osm, osr, s=4, alpha=0.4, color=ab.PALETTE[0]);
↪  ax[1].plot(osm, a0 + b * osm, color=ab.PALETTE[1])
ax[1].set_xlabel("elméleti kvantilis"); ax[1].set_ylabel("maradék
↪  kvantilise")
fig.tight_layout()
```

Az R^2 magas, de a diagnosztika több problémát is mutat. A VIF-értékek szerint a hűtővízáram és az energiafogyasztás (a szivattyú energiaigénye miatt), valamint a termékfüggő változók (betáplálás, betáplált koncentráció, keverőteljesítmény)



8.2. ábra. A lineáris lágy szenzor maradékai az illesztett értékek függvényében (balra) és normális Q–Q ábrán (jobbra). A maradékok szerkezete (sávok, görbület) és a nehéz farkak arra utalnak, hogy a lineáris modell nem írja le teljesen az összefüggést.

erősen kollineárisak, ezért az egyes együtthatók értelmezése bizonytalan – a keverőteljesítmény együtthatója például nem különbözik szignifikánsan nullától, holott a változó erősen korrelál a célváltozóval. A maradékok sávos szerkezete és nehéz farkai pedig arra utalnak, hogy az összefüggés nem lineáris, és a hibák egy része (a szenzor-drift időszakai) más mechanizmusból származik. A következő alfejezetek ezeket a problémákat kezelik.

8.3. Regularizált regresszió

Sok, egymással korrelált magyarázó változó esetén a legkisebb négyzetes becslés varianciája nagy (8.6): az $\mathbf{X}^\top\mathbf{X}$ közel szinguláris, kis sajátértékeinek inverze naggyá teszi az együtthatók szórását. A **regularizáció** a célfüggvényhez egy, a paraméterek nagyságát büntető tagot ad, és ezzel kis torzítás árán jelentősen csökkenti a varianciát (7.1. alfejezet).

8.3.1. Ridge-regresszió

A **ridge-regresszió** (Hoerl & Kennard, 1970) a paraméterek négyzetösszegét bünteti:

$$\hat{\boldsymbol{\theta}}_{\text{ridge}} = \arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2 = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}, \tag{8.7}$$

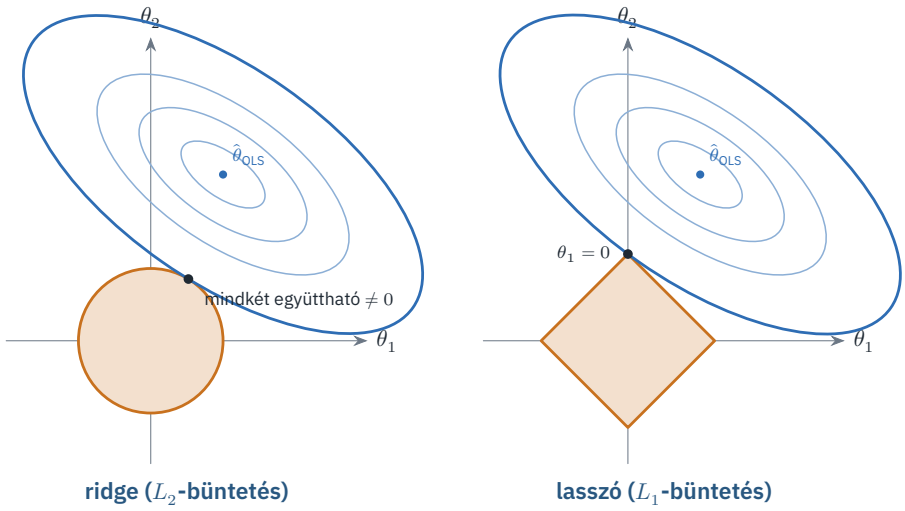
ahol $\lambda \geq 0$ a regularizációs paraméter (a tengelymetszetet nem büntetjük, és a változókat előtte standardizáljuk). A $\lambda\mathbf{I}$ hozzáadása $\mathbf{X}^\top\mathbf{X}$ minden sajátértékét λ-val növeli, így a mátrix mindig invertálható. A szinguláris felbontás nyelvén ($\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$) a ridge-becslés a j-edik főirány menti komponenst a $\sigma_j^2/(\sigma_j^2+\lambda)$ tényezővel zsugorítja: a kis szóródású, zajérzékeny irányokat erősen, a nagy szóródásúakat alig.

8.3.2. Lasszó és elasztikus háló

A **lasszó** (least absolute shrinkage and selection operator) (Tibshirani, 1996) az abszolút értékek összegét bünteti:

$$\hat{\boldsymbol{\theta}}_{\text{lasszó}} = \arg\min_{\boldsymbol{\theta}} \; \frac{1}{2N}\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_1. \tag{8.8}$$

Az L_1-büntetés nevezetes tulajdonsága, hogy elég nagy λ esetén egyes együtthatókat *pontosan nullává* tesz, vagyis változókiválasztást is végez. Ennek geometriai oka a 8.3. ábrán látható: a büntetett feladat egyenértékű a négyzetes hiba minimalizálásával egy korlátozó tartományon belül, és a hibafüggvény szintvonalai a gyémánt alakú L_1-tartományt nagy valószínűséggel egy csúcsában – ahol valamelyik koordináta nulla – érintik, míg a kör alakú L_2-tartományt általános helyzetű pontban. A lasszónak nincs zárt alakú megoldása; koordinátánkénti optimalizálással számolják. Erősen korrelált változók közül a lasszó önkényesen egyet választ; az **elasztikus háló** (Zou & Hastie, 2005) a két büntetés kombinációjával ($\lambda[\alpha\|\boldsymbol{\theta}\|_1 + (1-\alpha)\|\boldsymbol{\theta}\|_2^2/2]$) a korrelált változókat csoportosan tartja meg vagy hagyja el.

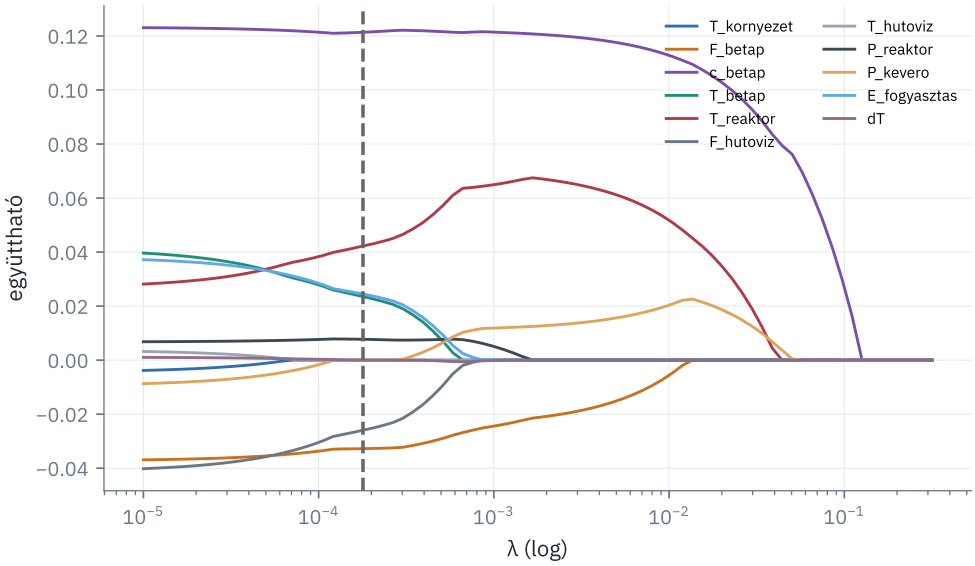


8.3. ábra. A ridge- és a lasszó-regresszió geometriája. A hibafüggvény szintvonalai (ellipszisek) a korlátozó tartományt (kör, illetve gyémánt) érintik; a lasszónál az érintési pont tipikusan egy csúcs, ahol valamelyik együttható pontosan nulla.

A λ értékét keresztvalidációval választjuk (7.2. alfejezet). Ehhez érdemes a **regularizációs utat** is megnézni: az együtthatókat λ függvényében ábrázolva látszik, melyik változó marad meg legtovább a modellben.

```
from sklearn.linear_model import LassoCV, lasso_path, RidgeCV,
↪  LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import TimeSeriesSplit
valt_b = ["T_kornyezet", "F_betap", "c_betap", "T_betap", "T_reaktor",
↪  "F_hutoviz",
          "T_hutoviz", "P_reaktor", "P_kevero", "E_fogyasztas", "dT"]
```

```
Zq = StandardScaler().fit_transform(q[valt_b])
yq = q.c_termek.to_numpy()
racs = np.logspace(-5, -0.5, 80)
alphas, coefs, _ = lasso_path(Zq, yq - yq.mean(), alphas=racs)
lcv = LassoCV(cv=TimeSeriesSplit(5), alphas=racs, max_iter=20000).fit(Zq,
↪  yq)
szinek = ab.PALETTE + ["#9AA5B1", "#3D4852", "#E0A458", "#5DA9E9",
↪  "#8E6C8A"]
fig, ax = plt.subplots()
for j, nev in enumerate(valt_b):
    ax.plot(alphas, coefs[j], lw=1.3, label=nev, color=szinek[j])
ax.set_xscale("log"); ax.axvline(lcv.alpha_, color="0.4", ls="--")
ax.set_xlabel("λ (log)"); ax.set_ylabel("együttható");
↪  ax.legend(fontsize=7, ncol=2, loc="upper right")
fig.tight_layout()
```



8.4. ábra. A lasszó regularizációs útja a lágy szenzor standardizált változóin: az együtthatók a büntetés erősségének függvényében. A függőleges vonal a keresztvalidációval (idősoros felosztással) választott λ.

8.4. Nemlineáris regresszió

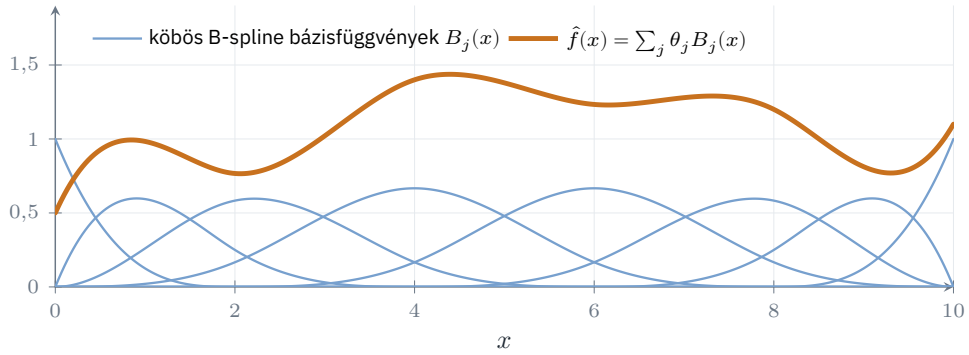
8.4.1. Paramétereiben lineáris modellek: bázisfüggvények

Sok nemlineáris összefüggés leírható olyan modellel, amely a *paramétereiben* lineáris:

$$\hat{y} = \sum_{j=1}^{M} \theta_j \, \phi_j(\mathbf{x}), \tag{8.9}$$

ahol a ϕ_j rögzített **bázisfüggvények**. A paraméterek ugyanúgy a legkisebb négyzetek módszerével becsülhetők, mint a lineáris esetben, csak az $\mathbf{X}$ helyére a $\Phi_{kj} = \phi_j(\mathbf{x}_k)$ mátrix kerül. Jellemző bázisok:

- **polinomok** $(1, x, x^2, \ldots)$: egyszerűek, de magas fokszámnál a tartomány szélein erősen oszcillálnak (7.1. ábra);
- **spline-ok**: szakaszonként polinomiális függvények, amelyek a szakaszhatárokon (a **csomópontokban**) folytonosan és simán (köbös spline-nál kétszer folytonosan deriválhatóan) csatlakoznak (Boor, 2001). A **B-spline-bázis** elemei kis tartójúak (csak néhány szomszédos szakaszon nem nullák), ezért az illesztés numerikusan stabil, és egy paraméter csak helyileg hat (8.5. ábra);
- **radiális bázisfüggvények**: $\phi_j(\mathbf{x}) = \exp(-\|\mathbf{x} - \mathbf{c}_j\|^2 / (2s^2))$ alakú „dombok” a $\mathbf{c}_j$ középpontok körül (Broomhead & Lowe, 1988); a középpontokat gyakran csoportosítással (k-átlag) választják.



8.5. ábra. Köbös B-spline bázisfüggvények egyenletes csomópontokkal (a vízszintes tengely jelölései) és belőlük súlyozott összegként előálló sima függvény.

Több változó esetén a bázisok száma a változószámmal gyorsan nő. Egy gyakori kompromisszum az **additív modell**, amely minden változóra külön spline-t illeszt, és ezeket összegzi; így a változók hatása egyenként, nemlineárisan is leírható és ábrázolható. Az ilyen sok paraméteres modelleket regularizációval kell kordában tartani.

8.4.2. Paramétereiben nemlineáris modellek

Ha a modellstruktúra fizikai, kémiai vagy közgazdasági ismeretekből adódik, a modell gyakran a paramétereiben is nemlineáris. A Pannon üzem esetében a reakciókinetika szerint egy elsőrendű reakció stacionárius konverziója kevert tankreaktorban

$$X = \frac{k(T)\,\tau}{1 + k(T)\,\tau}, \qquad k(T) = k_{\mathrm{ref}} \exp\!\left[-\frac{E_a}{R}\left(\frac{1}{T} - \frac{1}{T_{\mathrm{ref}}}\right)\right], \qquad \tau = \frac{V}{F}, \tag{8.10}$$

és a termékkoncentráció $c = c_{\mathrm{be}}\,X$. Itt T az abszolút hőmérséklet, F a betáplálási áram, V a reaktortérfogat, R az egyetemes gázállandó, a becsülendő paraméterek pedig a k_{ref} referencia-sebességi állandó és az E_a aktiválási energia. (Az Arrhenius-egyenletet egy T_{ref} referencia-hőmérsékletre vonatkoztatva írtuk fel; ez a két paraméter közötti erős korrelációt csökkenti, és numerikusan stabilabbá teszi a becslést.) Az ilyen **szürkedoboz-modellek** előnye, hogy kevés, fizikai jelentésű paraméterük van, és a mért tartományon kívül is megbízhatóbban extrapolálnak.

A paramétereket a nemlineáris legkisebb négyzetek módszerével becsüljük, iteratívan. A **Gauss–Newton-módszer** minden lépésben linearizálja a modellt az aktuális paraméterbecslés körül, és a linearizált feladat legkisebb négyzetes megoldásával lép: $\Delta\boldsymbol{\theta} = (\mathbf{J}^\top\mathbf{J})^{-1}\mathbf{J}^\top\mathbf{e}$, ahol $\mathbf{J}$ a modell paraméterek szerinti Jacobi-mátrixa, $\mathbf{e}$ a maradékvektor. A **Levenberg–Marquardt-módszer** (Levenberg, 1944; Marquardt, 1963) a $(\mathbf{J}^\top\mathbf{J} + \mu\mathbf{I})$ mátrixot használja – a ridge-regresszióhoz hasonlóan –, és a μ csillapítást adaptívan változtatja: kis μ esetén Gauss–Newton-, nagynál gradienslépést tesz. Ez a `scipy.optimize.curve_fit` alapértelmezett eljárása.

8.5. Gyakorlat: lágy szenzor és vevőérték-előrejelzés

8.5.1. Lágy szenzor a termékkoncentrációra

FUTÓ ESET · PANNON ÜZEM

A laboreredmény 4 óránként, órás késéssel érkezik; egy **lágy szenzor** (soft sensor) a folyamatosan mért változókból valós időben becsüli a termékkoncentrációt (Fortuna és mtsai., 2007; Kadlec és mtsai., 2009). Az alábbiakban öt modellt hasonlítunk össze: a lineáris, a ridge- és a lasszó-regressziót, egy additív spline-modellt, valamint a (8.10) szürkedoboz-modellt. A modelleket az első kilenc hónap adatain tanítjuk, és az utolsó negyedéven értékeljük, ahogy egy valódi bevezetésnél történne.

```
from scipy.optimize import curve_fit
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import SplineTransformer
from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import mean_absolute_error, mean_squared_error,
 ↪  r2_score

vagas = q.ido >= "2025-10-01"
tr, te = q[~vagas], q[vagas]
Xtr, Xte = tr[valt_b], te[valt_b]

modellek = {
    "lineáris (OLS)": make_pipeline(StandardScaler(), LinearRegression()),
    "ridge (λ: CV)": make_pipeline(StandardScaler(),
     ↪  RidgeCV(alphas=np.logspace(-3, 3, 30))),
    "lasszó (λ: CV)": make_pipeline(StandardScaler(),
     ↪  LassoCV(cv=TimeSeriesSplit(5), alphas=racs, max_iter=20000)),
    "additív spline + ridge": make_pipeline(StandardScaler(),
     ↪  SplineTransformer(n_knots=6, degree=3),
                                          RidgeCV(alphas=np.logspace(-3, 3,
 ↪  30))),
}
eredmeny = {}
for nev, mod in modellek.items():
    mod.fit(Xtr, tr.c_termek)
    p = mod.predict(Xte)
    eredmeny[nev] = p

# szürkedoboz-modell: c = c_be * k tau / (1 + k tau), k = k_ref exp(-Ea/R
 ↪  (1/T - 1/Tref))
R_GAS, V, T_REF = 8.314, 10.0, 358.15
```

```
def cstr(Xv, k_ref, Ea):
    T, F, c_be = Xv
    k = k_ref * np.exp(-Ea / R_GAS * (1 / (T + 273.15) - 1 / T_REF))
    tau = V / F
    return c_be * k * tau / (1 + k * tau)
par, cov = curve_fit(cstr, (tr.T_reaktor, tr.F_betap, tr.c_betap),
↪  tr.c_termek, p0=[2.0, 40_000])
eredmeny["szürkedoboz (CSTR)"] = cstr((te.T_reaktor, te.F_betap,
↪  te.c_betap), *par)

pd.DataFrame({nev: {"MAE": mean_absolute_error(te.c_termek, p),
                    "RMSE": np.sqrt(mean_squared_error(te.c_termek, p)),
                    "R²": r2_score(te.c_termek, p)} for nev, p in
                    ↪  eredmeny.items()}).T.round(4)
```

	MAE	RMSE	R²
lineáris (OLS)	0.0149	0.0189	0.9830
ridge (λ: CV)	0.0149	0.0189	0.9829
lasszó (λ: CV)	0.0151	0.0194	0.9822
additív spline + ridge	0.0146	0.0186	0.9835
szürkedoboz (CSTR)	0.0156	0.0212	0.9786

```
se = np.sqrt(np.diag(cov))
k0_becsult = par[0] * np.exp(par[1] / (R_GAS * T_REF))
print(f"k_ref = {par[0]:.3f} ± {se[0]:.3f} 1/h,   Ea = {par[1]/1000:.1f} ±
↪  {se[1]/1000:.1f} kJ/mol")
print(f"a megfelelő preexponenciális tényező: k0 = {k0_becsult:.2e} 1/h")
```

```
k_ref = 3.003 ± 0.004 1/h,   Ea = 60.1 ± 0.4 kJ/mol
a megfelelő preexponenciális tényező: k0 = 1.77e+09 1/h
```

Az eredmény első ránézésre meglepő: a mért működési tartományon belül mind az öt modell hasonlóan pontos, mert ebben a szűk tartományban az összefüggés közel lineáris. A szürkedoboz-modell mindössze két paraméterrel szinte ugyanezt a pontosságot éri el, és – ez a fontosabb – a becsült aktiválási energia és preexponenciális tényező közel esik a szimulátorban használt értékekhez (60 kJ/mol, illetve $1{,}7 \cdot 10^9$ 1/h). Kis lemaradását a szenzor-drift időszakai okozzák, amikor a mért hőmérséklet a valódinál nagyobb, és a fizikai modell „rossz” bemenetet kap.

A modellek közötti valódi különbség akkor jön elő, amikor a tanítóadatokban nem szereplő tartományban kell előre jelezni. Tanítsuk a modelleket csak az A és B termék adatain, és értékeljük a C terméken, amelyet magasabb hőmérsékleten és nagyobb betáplálással gyártanak:

```
tr2, te2 = q[q.termek != "C"], q[q.termek == "C"]
extra = {}
for nev in ["lineáris (OLS)", "additív spline + ridge"]:
    mod = modellek[nev].fit(tr2[valt_b], tr2.c_termek)
    extra[nev] = mod.predict(te2[valt_b])
par2, _ = curve_fit(cstr, (tr2.T_reaktor, tr2.F_betap, tr2.c_betap),
 ↪  tr2.c_termek, p0=[2.0, 40_000])
extra["szürkedoboz (CSTR)"] = cstr((te2.T_reaktor, te2.F_betap,
 ↪  te2.c_betap), *par2)
pd.DataFrame({nev: {"RMSE": np.sqrt(mean_squared_error(te2.c_termek, p)),
 ↪  "R²": r2_score(te2.c_termek, p)}
              for nev, p in extra.items()}).T.round(3)
```

	RMSE	R²
lineáris (OLS)	0.019	0.749
additív spline + ridge	0.052	-0.783
szürkedoboz (CSTR)	0.018	0.790

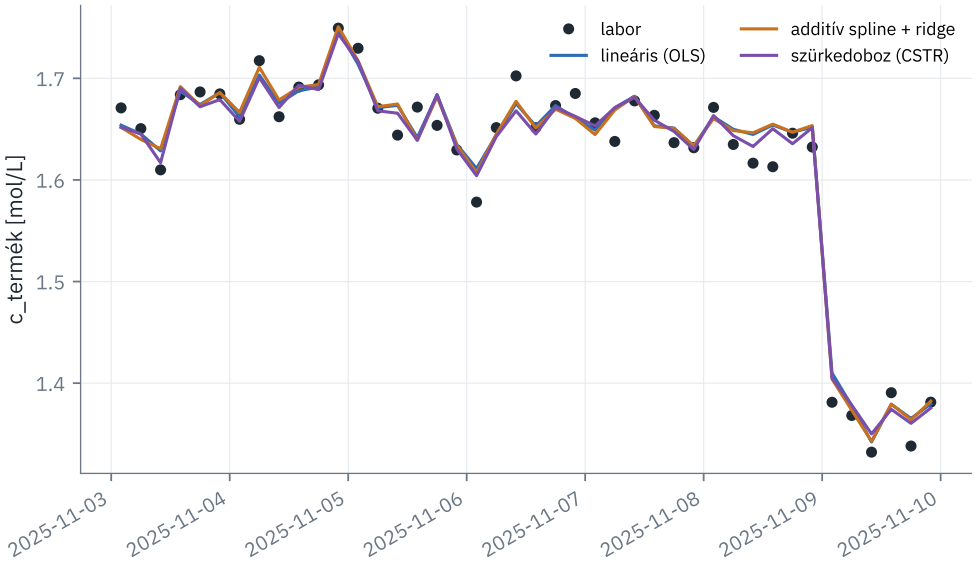
Az ismeretlen termékre a fizikai modell extrapolál a legjobban, a rugalmas spline-modell viszont teljesen használhatatlanná válik: a tanítási tartományon kívül a spline-ok viselkedését semmi sem köti meg. Ez a szürkedoboz-modellezés fő érve: a fizikai struktúra olyan tudást visz a modellbe, amely az adatokban nincs benne.

```
het = (te.ido >= "2025-11-03") & (te.ido < "2025-11-10")
fig, ax = plt.subplots()
ax.plot(te.ido[het], te.c_termek[het], "o", color="#1F2933", ms=4,
 ↪  label="labor")
for (nev, p), c in zip([(k, eredmeny[k]) for k in ["lineáris (OLS)",
 ↪  "additív spline + ridge", "szürkedoboz (CSTR)"]], ab.PALETTE):
    ax.plot(te.ido[het], p[het.to_numpy()], color=c, lw=1.4, label=nev)
ax.set_ylabel("c_termék [mol/L]"); ax.legend(fontsize=8, ncol=2)
fig.autofmt_xdate(); fig.tight_layout()
```

8.5.2. A vevők jövőbeli költése

FUTÓ ESET · PANNONSHOP

A PannonShop számára fontos kérdés, hogy egy vevő a következő fél évben várhatóan mennyit költ – ez a vevőérték (customer lifetime value) egyszerűsített, rövid távú változata (14. fejezet). A jellemzőket a 2025. július 1-je előtti vásárlásokból számoljuk, a célváltozó a következő 183 nap nettó vásárlási értéke.



8.6. ábra. A lágy szenzorok becslése a teszt-negyedév egy hetén a laboreredményekhez (pontok) képest.

```
from sklearn.model_selection import KFold, cross_val_predict
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import FunctionTransformer
churn = ab.churn_table(cutoff="2025-07-01", horizon_days=183)
tiszta = ab.shop_clean()
jovo = tiszta[(tiszta.InvoiceDate >= "2025-07-01") & (tiszta.InvoiceDate <
↪  "2026-01-01")]
koltes =
↪  jovo.groupby("CustomerID").Ertek.sum().reindex(churn.index).fillna(0)
jell = ["R_nap", "F_szamla", "M_ertek", "atl_kosarertek", "atl_kosarmeret",
↪  "atl_mennyiseg",
        "elso_vasarlas_nap", "akcios_arany", "arany_kave", "arany_gep"]
Xc, yc = churn[jell], np.log1p(koltes)                    # logaritmikus
↪  célváltozó
elo = make_pipeline(SimpleImputer(strategy="median"),
↪  FunctionTransformer(np.log1p), StandardScaler())
cv = KFold(5, shuffle=True, random_state=0)
for nev, mod in {"átlag (viszonyítás)": make_pipeline(elo,
↪  Ridge(alpha=1e9)),
                 "lineáris": make_pipeline(elo, LinearRegression()),
                 "ridge": make_pipeline(elo, RidgeCV(alphas=np.logspace(-2,
                  ↪  3, 30)))}.items():
```

```
    p = cross_val_predict(mod, Xc, yc, cv=cv)
    print(f"{nev:22s} R² = {r2_score(yc, p):6.3f}   RMSE(log) =
    ↪  {np.sqrt(mean_squared_error(yc, p)):.3f}")
```

```
átlag (viszonyítás)    R² = -0.001   RMSE(log) = 6.003
lineáris               R² =  0.280   RMSE(log) = 5.092
ridge                  R² =  0.279   RMSE(log) = 5.095
```

A vásárlási értéket logaritmikus skálán modellezzük, mert erősen jobbra ferde, és sok vevőnél nulla (nem vásárol). A lineáris modell a variancia egy részét magyarázza meg, de a nullák nagy aránya miatt egy kétlépéses modell – először osztályozás (vásárol-e egyáltalán), majd regresszió a vásárlók körében – jobban illik a feladathoz; ezt a 14. fejezetben valósítjuk meg.

Interaktív alkalmazás: Polinomiális regresszió és regularizáció: torzítás és variancia

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/regresszio.html

Colab-notebook: 8. fejezet – regresszió

8.6. Összefoglalás

- A legkisebb négyzetes becslés a normálegyenletek megoldása; geometriailag a célváltozó merőleges vetülete a magyarázó változók alterére.
- A becslés tulajdonságai a hibákra tett feltevéseken múlnak; ezeket maradékábrákkal, a multikollinearitást VIF-fel, a befolyásos pontokat a kalapmátrix átlójával ellenőrizzük.
- A ridge-regresszió a négyzetes, a lasszó az abszolút értékes büntetéssel csökkenti a varianciát; a lasszó változókat is kiválaszt. A büntetés erősségét keresztvalidációval választjuk.
- A bázisfüggvényes (polinomiális, spline-, RBF-) modellek paramétereikben lineárisak, így a legkisebb négyzetes eszköztár változatlanul alkalmazható rájuk.
- A fizikai ismeretekre épülő, paramétereiben nemlineáris szürkedoboz-modellek kevés, értelmezhető paraméterrel írják le a folyamatot; a Gauss–Newton- és a Levenberg–Marquardt-módszerrel illeszthetők.

Ellenőrző kérdések

1. Vezesse le a normálegyenleteket! Mikor nem egyértelmű a megoldás?

2. Miért merőleges a maradékvektor a magyarázó változók alterére?
3. Mit mér a VIF, és miért okoz gondot az együtthatók értelmezésében a multikollinearitás?
4. Miért ad a lasszó pontosan nulla együtthatókat, a ridge pedig nem?
5. Mit jelent, hogy egy modell „paramétereiben lineáris”? Paramétereiben lineáris-e az $y = \theta_1 e^{-x} + \theta_2 \sin x$ modell? És az $y = \theta_1 e^{-\theta_2 x}$?
6. Miért előnyös az Arrhenius-egyenletet referencia-hőmérsékletre vonatkoztatva felírni?

Feladatok

1. **Kézi számolás.** Számítsa ki a kézi példa R^2-ét és a meredekség standard hibáját (8.6)! Adja meg a meredekség 95%-os konfidenciaintervallumát!
2. **Ridge zárt alakban.** Valósítsa meg a (8.7) képletet NumPy-ban, és ellenőrizze a `scikit-learn` `Ridge` osztályával (vigyázzon a tengelymetszet kezelésére)!
3. **Termékenkénti modellek.** Illesszen külön lineáris lágy szenzort mindhárom termékre! Jobb-e az eredmény a közös modellnél? Mi a magyarázat?
4. **Robusztus illesztés.** Ismételje meg a szürkedoboz-modell illesztését a szenzor-drift időszakok nélkül (`plant.truth["fault"]`)! Mennyire változnak a becsült paraméterek? Próbálja ki a `scipy.optimize.least_squares` robusztus veszteségfüggvényét (`loss="soft_l1"`) is, amely a hibák ismerete nélkül csökkenti a kiugró pontok hatását!
5. **Kétlépéses vevőérték-modell.** Építsen logisztikus regressziót arra, hogy a vevő vásárol-e a következő fél évben, és lineáris regressziót a vásárlók költésére! A kettő szorzata a várható költés. Jobb-e így az előrejelzés?

További olvasmányok

A lineáris regresszió statisztikai elméletét és diagnosztikáját Montgomery és mtsai. (2021) tárgyalja részletesen. A regularizált és a bázisfüggvényes regresszió egységes tárgyalása Hastie és mtsai. (2009) és James és mtsai. (2023) könyvében található. A spline-okról Boor (2001) a klasszikus referencia. Az adatvezérelt lágy szenzorok ipari alkalmazásairól Kadlec és mtsai. (2009) és Fortuna és mtsai. (2007) ad áttekintést.

9. Osztályozás

Miről szól a fejezet?
Az osztályozás egy kategorikus célváltozó előrejelzése: lemorzsolódik-e a vevő, megfelel-e a termék, milyen hiba lépett fel a folyamatban. A fejezet a legfontosabb modellcsaládokat tárgyalja – a logisztikus regressziót, a döntési fákat, a Bayes-féle osztályozókat, a legközelebbi szomszéd módszert, a támaszvektor-gépeket és az együttes (ensemble) modelleket: a zsákolást, a véletlen erdőt és a gradiens-boostingot. Mindegyiknél a működési elvet, a tanítás célfüggvényét és a jellemző erősségeket és gyengeségeket mutatjuk be. Végül a modellek értelmezésének eszközeivel foglalkozunk, és a futó eseteken összehasonlítjuk a módszereket.
A fejezet végére az olvasó képes lesz egy döntési fát kézzel felépíteni az információnyereség alapján; a naiv Bayes-osztályozót kézzel alkalmazni; a logisztikus regresszió együtthatóit esélyhányadosként értelmezni; a fő modellcsaládokat egy feladaton összehasonlítani; és egy modell döntéseit permutációs fontossággal és parciális függéssel értelmezni.

9.1. Az osztályozási feladat

Az osztályozó egy $f : \mathbb{R}^n \to \{1, \dots, C\}$ leképezés, amely a jellemzőtér minden pontjához egy osztályt rendel. Az osztályokat elválasztó felületeket **döntési határoknak** nevezzük. A modellek abban különböznek, milyen alakú döntési határokat képesek leírni (lineáris, szakaszonként tengelyekkel párhuzamos, tetszőleges sima), és hogyan találják meg őket az adatokból.

A valószínűségi osztályozók a $P(y = c \mid \mathbf{x})$ feltételes valószínűségeket becsülik, és – egyenlő hibaköltségek esetén – a legnagyobb valószínűségű osztályt választják. Két megközelítést szokás megkülönböztetni. A **diszkriminatív** modellek (logisztikus regresszió, döntési fák, neurális hálózatok) közvetlenül a $P(y \mid \mathbf{x})$-et modellezik. A **generatív** modellek (naiv Bayes, Gauss-féle diszkriminanciaanalízis) az osztályonkénti $p(\mathbf{x} \mid y)$ eloszlást és az osztályok $P(y)$ a priori valószínűségét modellezik, és a Bayes-tétellel számolják a $P(y \mid \mathbf{x})$-et.

9.2. Logisztikus regresszió

A **logisztikus regresszió** (Cox, 1958) a pozitív osztály valószínűségét egy lineáris kifejezés szigmoid transzformáltjaként modellezi:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b), \qquad \sigma(z) = \frac{1}{1 + e^{-z}}. \tag{9.1}$$

A szigmoid függvény a valós számokat a $(0, 1)$ intervallumba képezi. Átrendezve a modell az **esély** (odds) logaritmusára lineáris:

$$\ln \frac{P(y = 1 \mid \mathbf{x})}{1 - P(y = 1 \mid \mathbf{x})} = \mathbf{w}^\top \mathbf{x} + b. \tag{9.2}$$

Ebből adódik az együtthatók értelmezése: ha a j-edik változó egy egységgel nő (a többi változatlansága mellett), az esély e^{w_j}-szeresére változik; ez az **esélyhányados**. A döntési határ, ahol $P = 0{,}5$, a $\mathbf{w}^\top \mathbf{x} + b = 0$ hipersík – a logisztikus regresszió tehát lineáris osztályozó.

A paramétereket maximum-likelihood módszerrel becsüljük. Egy megfigyelés valószínűsége $p_k^{y_k}(1 - p_k)^{1 - y_k}$, ahol $p_k = \sigma(\mathbf{w}^\top \mathbf{x}_k + b)$; a negatív log-likelihood a **kereszt-entrópia** (log-veszteség):

$$J(\mathbf{w}, b) = -\sum_{k=1}^{N} \big[y_k \ln p_k + (1 - y_k) \ln(1 - p_k) \big] + \lambda \lVert \mathbf{w} \rVert_2^2, \tag{9.3}$$

amelyhez regularizációs tagot is adunk. A J konvex, ezért egyetlen minimuma van, de zárt alakú megoldása nincs; gradiense meglepően egyszerű, $\nabla_{\mathbf{w}} J = \sum_k (p_k - y_k)\mathbf{x}_k + 2\lambda \mathbf{w}$, és Newton-típusú vagy kvázi-Newton-módszerekkel gyorsan minimalizálható. Több osztály esetén a szigmoid helyére a **softmax** függvény lép, $P(y = c \mid \mathbf{x}) = e^{z_c} / \sum_j e^{z_j}$, ahol $z_c = \mathbf{w}_c^\top \mathbf{x} + b_c$.

9.3. Döntési fák

9.3.1. Felépítés

A **döntési fa** a jellemzőteret tengelyekkel párhuzamos vágásokkal téglatestekre bontja, és minden téglatesthez egy osztályt rendel. Minden belső csomópont egy változóra vonatkozó tesztet tartalmaz (például „Hőmérséklet = magas?” vagy

„$x_j \leq 3{,}5$?”), a levelek pedig az osztálycímkéket. Egy új objektum besorolásához a gyökértől a levélig követjük a tesztek kimenetét. A fák legnagyobb előnye az **értelmezhetőség**: minden levélhez vezető út egy „ha … és … akkor …” szabály.

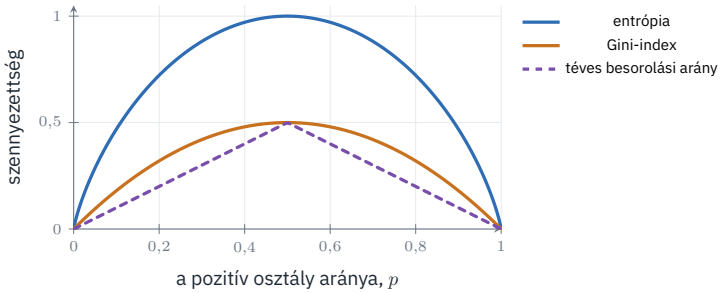
A fát felülről lefelé, mohó rekurzív felosztással építjük (Breiman és mtsai., 1984; Quinlan, 1986): minden csomópontban azt a tesztet választjuk, amely a gyermek-csomópontokat a lehető legtisztábbá teszi, majd a gyermekekben ugyanezt megismételjük, amíg egy leállási feltétel nem teljesül (tiszta csomópont, túl kevés objektum, maximális mélység).

9.3.2. Szennyezettségi mértékek

Egy csomópont **szennyezettségét** az osztályok $p_1, \ldots, p_C$ relatív gyakoriságai alapján mérjük (9.1. ábra). A három leggyakoribb mérték:

$$H = -\sum_{c=1}^{C} p_c \log_2 p_c \text{ (entrópia)}, \qquad G = 1 - \sum_{c=1}^{C} p_c^2 \text{ (Gini-index)}, \qquad E = 1 - \max_c p_c \text{ (téves b} \tag{9.4}$$

Mindhárom nulla, ha a csomópont tiszta, és maximális, ha az osztályok egyenlő arányban fordulnak elő. Az entrópia az információelméletből (Shannon, 1948) származik: azt méri, átlagosan hány bit kell egy véletlenszerűen választott objektum osztályának kódolásához.



9.1. ábra. Szennyezettségi mértékek kétosztályos esetben a pozitív osztály arányának függvényében. Az entrópia és a Gini-index szigorúan konkáv, ezért egy vágás akkor is csökkentheti őket, ha a téves besorolási arány nem változik – ezért ezeket használják a fák építésére.

Egy A teszt jóságát az **információnyereséggel** mérjük – a szennyezettség csökkenésével a gyermek-csomópontok méretével súlyozott átlagához képest:

$$IG(S, A) = H(S) - \sum_{v \in \text{értékek}(A)} \frac{|S_v|}{|S|} H(S_v), \tag{9.5}$$

ahol S_v az S azon részhalmaza, amelyben $A = v$. A sok értéket felvevő változókat az információnyereség előnyben részesíti (egy azonosító-szerű változó tökéletesen „tiszta”, de értelmetlen felosztást ad); ezt a C4.5 algoritmus (Quinlan, 1993) a **nyereségaránnyal** korrigálja, amely az információnyereséget a felosztás saját entrópiájával ($-\sum_v \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$) osztja. A CART algoritmus (Breiman és mtsai., 1984) csak bináris vágásokat végez, és a Gini-indexet használja; numerikus változóknál minden lehetséges küszöböt (a rendezett értékek közötti felezőpontokat) kipróbál.

9.3.3. Kézi példa

A segédcsomag 14 gyártási tételének minősítését négy kategorikus változó írja le (9.1. táblázat). Építsük fel a döntési fát az információnyereség alapján!

```
d = ab.minoseg_tabla()
d
```

9.1. táblázat. Tizennégy gyártási tétel minősítése.

	Homerseklet	Alapanyag	Muszak	Karbantartva	Minoseg
0	magas	A	nappali	nem	selejt
1	magas	A	nappali	igen	selejt
2	normál	A	nappali	nem	jó
3	alacsony	B	nappali	nem	jó
4	alacsony	B	éjszakai	nem	jó
5	alacsony	B	éjszakai	igen	selejt
6	normál	B	éjszakai	igen	jó
7	magas	B	nappali	nem	selejt
8	magas	B	éjszakai	nem	jó
9	alacsony	A	nappali	nem	jó
10	magas	A	éjszakai	igen	jó
11	normál	B	nappali	igen	jó
12	normál	A	éjszakai	nem	jó
13	alacsony	A	nappali	igen	selejt

A gyökérben 9 jó és 5 selejtes tétel van, az entrópia

$$H(S) = -\tfrac{9}{14}\log_2\tfrac{9}{14} - \tfrac{5}{14}\log_2\tfrac{5}{14} \approx 0{,}940.$$

A Hőmérséklet szerinti felosztás három részhalmazt ad: alacsony (3 jó, 2 selejt, $H \approx 0{,}971$), normál (4 jó, $H = 0$), magas (2 jó, 3 selejt, $H \approx 0{,}971$). Az információnyereség tehát

$$IG(S, \text{Hőmérséklet}) = 0{,}940 - \tfrac{5}{14} \cdot 0{,}971 - \tfrac{4}{14} \cdot 0 - \tfrac{5}{14} \cdot 0{,}971 \approx 0{,}247.$$

A többi változóra ugyanígy számolva:

```
def H(y):
    p = y.value_counts(normalize=True)
    return float(-(p * np.log2(p)).sum())

def info_nyereseg(df, attr, cel="Minoseg"):
    sulyozott = sum(len(s) / len(df) * H(s[cel]) for _, s in
↪   df.groupby(attr))
    return H(df[cel]) - sulyozott

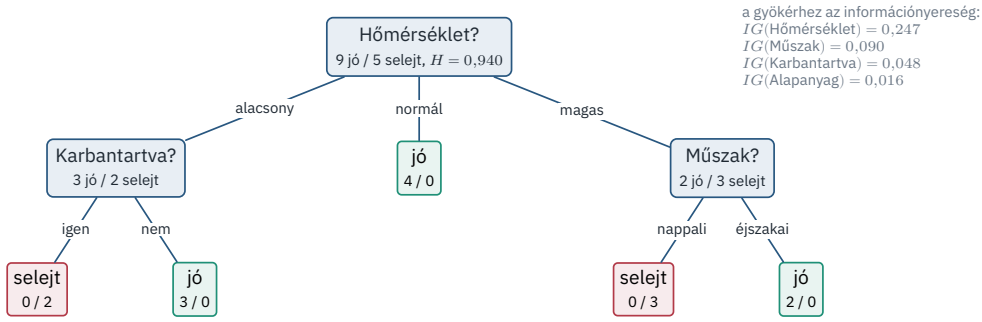
pd.Series({a: info_nyereseg(d, a) for a in ["Homerseklet", "Alapanyag",
↪   "Muszak", "Karbantartva"]},
          name="információnyereség").round(3).to_frame()
```

	információnyereség
Homerseklet	0.247
Alapanyag	0.016
Muszak	0.090
Karbantartva	0.048

A gyökérbe a Hőmérséklet kerül. A normál ág tiszta (levél: jó). Az alacsony ágban a Karbantartva, a magas ágban a Műszak adja a legnagyobb (tökéletes) nyereséget, így a fa két további szintben tiszta levelekhez jut (9.2. ábra). A fa szabályai könnyen olvashatók: magas hőmérsékleten a nappali műszakban gyártott tételek, alacsony hőmérsékleten pedig a karbantartás után gyártott tételek selejtesek. Tizennégy megfigyelésből természetesen nem vonhatunk le általános következtetést – a példa az algoritmus lépéseit szemlélteti; a fa tökéletes illeszkedése a tanítóadatokra éppen a következő pontban tárgyalt túltanulás veszélyére figyelmeztet.

9.3.4. A túltanulás elkerülése: metszés

A fát addig növesztve, amíg minden levél tiszta, a tanítóadatokon nulla hibát érünk el – és szinte biztosan túltanulunk. A fák mérete kétféleképpen korlátozható. Az **előmetszés** a növesztést állítja le idejekorán (maximális mélység,



9.2. ábra. A kézi példa döntési fája. A csomópontokban a jó és selejtes tételek száma, az éleken a teszt kimenete látható.

egy levélben legalább adott számú objektum, minimális nyereség). Az **utómetszés** előbb teljes fát növeszt, majd visszanyesi. A CART **költség–komplexitás metszése** (Breiman és mtsai., 1984) az

$$R_\alpha(T) = R(T) + \alpha\,|T| \tag{9.6}$$

kifejezést minimalizálja, ahol $R(T)$ a fa hibája (vagy szennyezettsége), $|T|$ a levelek száma, $\alpha \geq 0$ pedig a komplexitás ára; az α-t keresztvalidációval választjuk. A döntési fák hátránya a nagy variancia: az adatok kis változása egészen más fát eredményezhet. Ezt a hátrányt az együttes modellek használják ki és küszöbölik ki (9.7. alfejezet).

9.4. Bayes-féle osztályozók

9.4.1. A Bayes-tétel és a naiv Bayes-osztályozó

A Bayes-tétel szerint

$$P(y = c \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid y = c)\,P(y = c)}{p(\mathbf{x})}, \tag{9.7}$$

ahol a nevező nem függ az osztálytól, ezért a döntéshez elegendő a számlálót maximalizálni. A $p(\mathbf{x} \mid y)$ sokdimenziós eloszlás becsléséhez rengeteg adat kellene. A **naiv Bayes-osztályozó** azzal az erős egyszerűsítéssel él, hogy az osztályon belül a jellemzők feltételesen függetlenek:

$$\hat{y} = \arg\max_{c} \; P(y = c) \prod_{j=1}^{n} P(x_j \mid y = c). \tag{9.8}$$

Kategorikus jellemzőknél a $P(x_j = v \mid y = c)$ valószínűségeket relatív gyakoriságokkal becsüljük. Ha egy érték egy osztályban nem fordult elő, a becslés nulla, és ez az egész szorzatot nullává teszi; ezt a **Laplace-simítás** kerüli el: $P(x_j = v \mid y = c) = (N_{c,v} + 1)/(N_c + V_j)$, ahol V_j a változó lehetséges értékeinek száma. Folytonos jellemzőkre osztályonként normális eloszlást szokás feltételezni (Gauss-féle naiv Bayes). A függetlenségi feltevés a valóságban szinte sosem teljesül, a naiv Bayes osztályozási pontossága mégis gyakran meglepően jó; ennek okát Domingos & Pazzani (1997) elemezte: a helyes döntéshez nem kell pontos valószínűség, elég, ha a legnagyobb valószínűségű osztály helyes. A becsült valószínűségek viszont rendszerint rosszul kalibráltak.

9.4.2. Kézi példa

Soroljuk be a naiv Bayes-osztályozóval azt az új tételt, amelyet magas hőmérsékleten, B alapanyagból, éjszakai műszakban, karbantartás után gyártottak! A gyakoriságok a 9.1. táblázatból (Laplace-simítással) számolhatók:

```
uj = {"Homerseklet": "magas", "Alapanyag": "B", "Muszak": "éjszakai",
 ↪  "Karbantartva": "igen"}
pontszam = {}
for c, dc in d.groupby("Minoseg"):
    p = len(dc) / len(d)                                  # a priori
 ↪  valószínűség
    tenyezok = [f"P({c})={p:.3f}"]
    for attr, v in uj.items():
        V = d[attr].nunique()
        pv = ((dc[attr] == v).sum() + 1) / (len(dc) + V)   # Laplace-simítás
        p *= pv
        tenyezok.append(f"P({v}|{c})={pv:.3f}")
    pontszam[c] = p
    print(" · ".join(tenyezok), f" = {p:.5f}")
osszeg = sum(pontszam.values())
print({c: round(v / osszeg, 3) for c, v in pontszam.items()})
```

```
P(jó)=0.643 · P(magas|jó)=0.250 · P(B|jó)=0.545 · P(éjszakai|jó)=0.545 ·
 ↪  P(igen|jó)=0.364  = 0.01739
P(selejt)=0.357 · P(magas|selejt)=0.500 · P(B|selejt)=0.429 ·
 ↪  P(éjszakai|selejt)=0.286 · P(igen|selejt)=0.571  = 0.01249
{'jó': np.float64(0.582), 'selejt': np.float64(0.418)}
```

A tétel a naiv Bayes szerint inkább jó, mint selejtes – összhangban a döntési fával, amely a magas hőmérsékletű, éjszakai műszakos tételeket jónak sorolja.

9.4.3. Bayes-hálók

A **Bayes-hálók** (Pearl, 1988) a teljes függetlenség helyett a változók közötti feltételes függetlenségeket egy irányított körmentes gráffal írják le: minden változó csak a gráfbeli szüleitől függ, és az együttes eloszlás a $p(x_1, \ldots, x_n) = \prod_j p(x_j \mid \text{szülők}(x_j))$ szorzatra bomlik. A naiv Bayes olyan speciális Bayes-háló, amelyben az osztályváltozó minden jellemző egyetlen szülője. A Bayes-hálók szakértői tudással is felépíthetők, ezért a diagnosztikai és kockázatelemzési alkalmazásokban (orvosi diagnózis, hibafa-elemzés) népszerűek.

9.5. A k-legközelebbi szomszéd módszer

A **k-legközelebbi szomszéd** (k-NN) módszer (Cover & Hart, 1967) nem épít explicit modellt: egy új objektumot a tanítóhalmaz k hozzá legközelebbi elemének többségi osztályába sorol (regressziónál az értékeiket átlagolja). Az ilyen „lusta” módszerek tanítása csak az adatok eltárolása, az előrejelzés viszont minden új objektumnál a teljes tanítóhalmaz átvizsgálását igényli (amit megfelelő adatszerkezetek gyorsíthatnak). Kis k esetén a döntési határ nagyon szabálytalan (nagy variancia), nagy k esetén elsimul (nagy torzítás); a k értékét keresztvalidációval választjuk. Egy klasszikus eredmény szerint végtelen sok tanítóadat esetén az 1-NN hibája legfeljebb kétszerese a lehető legkisebb (Bayes-) hibának.

A módszer a távolságmértéken áll vagy bukik: a változókat standardizálni kell, és sok dimenzióban – különösen sok lényegtelen változó esetén – a távolságok egyre kevésbé különböztetik meg a közeli és a távoli pontokat (a **dimenzió átka**). Ugyanez a jelenség okozta az FCM elmosódását a 5. fejezetben.

9.6. Támaszvektor-gépek

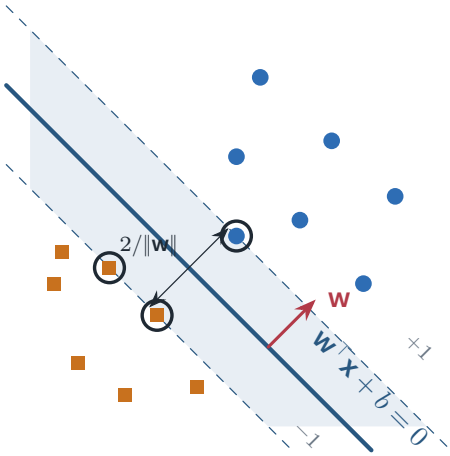
9.6.1. A maximális margó

Ha két osztály lineárisan elválasztható, végtelen sok elválasztó hipersík létezik. A **támaszvektor-gép** (support vector machine, SVM) (Cortes & Vapnik, 1995) azt választja, amelyik a két osztálytól a lehető legtávolabb halad, vagyis a **margó** – a hipersík és a legközelebbi pontok távolsága – maximális (9.3. ábra). A címkéket

$y_k \in \{-1, +1\}$ alakban kódolva, és a hipersíkot úgy normálva, hogy a legközelebbi pontokra $|\mathbf{w}^\top \mathbf{x}_k + b| = 1$ legyen, a margó szélessége $2/\|\mathbf{w}\|$. A maximális margójú hipersík tehát a

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{feltéve, hogy} \quad y_k(\mathbf{w}^\top \mathbf{x}_k + b) \geq 1, \quad k = 1, \ldots, N \tag{9.9}$$

kvadratikus programozási feladat megoldása. A megoldásban csak a margón fekvő pontok – a **támaszvektorok** – játszanak szerepet; a többi pont elhagyható anélkül, hogy a döntési határ változna.



bekarikázva: támaszvektorok (csak ezek határozzák meg a döntési határt)

9.3. ábra. A maximális margójú elválasztó hipersík. A szaggatott vonalak a margó határai, a bekarikázott pontok a támaszvektorok.

9.6.2. Puha margó és kernelek

A valós adatok ritkán választhatók el tökéletesen. A **puha margójú** SVM minden megfigyelésnek megenged egy $\xi_k \geq 0$ megsértést, és ezek összegét C súllyal bünteti: $\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_k \xi_k$, feltéve, hogy $y_k(\mathbf{w}^\top \mathbf{x}_k + b) \geq 1 - \xi_k$. Ez egyenértékű a

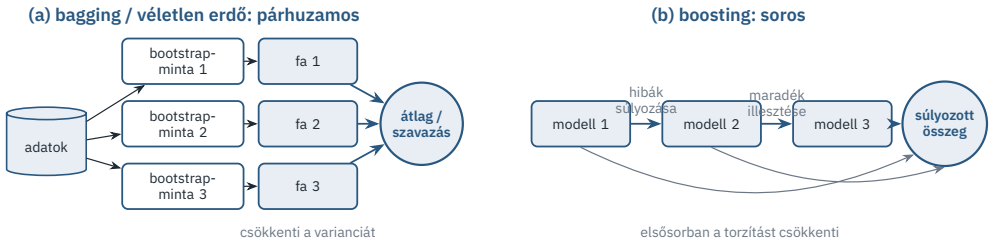
$$\min_{\mathbf{w},b} \sum_{k=1}^{N} \max(0,\, 1 - y_k(\mathbf{w}^\top \mathbf{x}_k + b)) + \frac{1}{2C}\|\mathbf{w}\|^2 \tag{9.10}$$

regularizált veszteségminimalizálással, amelyben a **csuklóveszteség** (hinge loss) szerepel – vagyis az SVM is a (1.3) keretbe illeszkedik. A nagy C kevés hibát enged (keskeny margó, nagy variancia), a kicsi széles margót és nagyobb torzítást ad.

A feladat duális alakjában az adatok csak skaláris szorzatok, $\mathbf{x}_i^\top \mathbf{x}_j$ formájában szerepelnek. Ha ezeket egy $K(\mathbf{x}_i, \mathbf{x}_j)$ **kernelfüggvénnyel** helyettesítjük, az SVM egy (akár végtelen dimenziós) jellemzőtérben keres lineáris elválasztást, ami az eredeti térben nemlineáris döntési határnak felel meg – anélkül, hogy a jellemzőteret ténylegesen ki kellene számolni (**kerneltrükk**). A leggyakoribb a radiális (Gauss-) kernel, $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, amelynek γ paramétere a döntési határ „hajlékonyságát” szabályozza. Az SVM közepes méretű, sok jellemzős feladatokon (például szövegosztályozásban, 13. fejezet) különösen erős; nagyon sok megfigyelésre a kernel-SVM tanítása lassú.

9.7. Együttes modellek

Az **együttes** (ensemble) modellek sok alapmodell előrejelzését kombinálják (9.4. ábra). Két fő stratégiájuk a variancia, illetve a torzítás csökkentését célozza.



9.4. ábra. Az együttes modellek két fő stratégiája. (a) A zsákolás és a véletlen erdő egymástól függetlenül, párhuzamosan tanított modellek előrejelzését átlagolja; (b) a boosting a modelleket egymás után tanítja, és mindegyik az előzőek hibáit igyekszik javítani.

9.7.1. Zsákolás és véletlen erdő

A **zsákolás** (bagging, bootstrap aggregating) (Breiman, 1996) a tanítóhalmazból visszatevéses mintavétellel (bootstrap) B mintát vesz, mindegyiken tanít egy modellt (tipikusan egy mély, nagy varianciájú fát), és az előrejelzéseket átlagolja, illetve szavazással kombinálja. Ha az egyes modellek varianciája σ^2, páronkénti korrelációjuk ρ, az átlag varianciája

$$\mathrm{Var}\Big(\frac{1}{B}\sum_{b=1}^{B}\hat{f}_b(\mathbf{x})\Big) = \rho\,\sigma^2 + \frac{1-\rho}{B}\,\sigma^2. \tag{9.11}$$

A modellek számának növelése a második tagot nullához viszi, az elsőt azonban nem: a korrelált modellek átlagolása csak korlátozottan csökkenti a varianciát. A **véletlen erdő** (Breiman, 2001) ezért a fák közötti korrelációt is csökkenti: minden vágásnál csak a változók egy véletlen részhalmazából (osztályozásnál tipikusan $\sqrt{n}$ változóból) választhat. A bootstrap-mintákból kimaradt („zsákon kívüli”, out-of-bag) megfigyelések ingyenes validációs halmazt adnak. A véletlen erdő kevés hangolással is jól működik, ezért a táblázatos adatok egyik legmegbízhatóbb „alapértelmezett” modellje.

9.7.2. Boosting

A **boosting** a modelleket egymás után tanítja, és minden új modell az addigiak gyengeségeire összpontosít. Az **AdaBoost** (Freund & Schapire, 1997) a hibásan osztályozott megfigyelések súlyát növeli, és a modelleket pontosságukkal súlyozva kombinálja. A **gradiens-boosting** (Friedman, 2001) általánosabb: a modell egy additív függvény, $F_M(\mathbf{x}) = \sum_{m=1}^{M} \nu\, h_m(\mathbf{x})$, ahol minden h_m egy kis fa, amelyet a veszteségfüggvény aktuális negatív gradiensére („pszeudomaradékaira”) illesztünk, ν pedig egy kis tanulási ráta. Négyzetes veszteségnél a pszeudomaradék éppen a közönséges maradék: minden új fa az addigi modell hibáit közelíti. A gradiens-boosting korszerű, hisztogramos megvalósításai (XGBoost, LightGBM, a `scikit-learn` `HistGradientBoostingClassifier` osztálya) nagy adathalmazokon is gyorsak (T. Chen & Guestrin, 2016; Ke és mtsai., 2017). Kiterjedt összehasonlító vizsgálatok szerint közepes méretű táblázatos adatokon a fa alapú együttes modellek ma is jellemzően legalább olyan jók, mint a mély neurális hálózatok (Grinsztajn és mtsai., 2022).

9.8. A modellek értelmezése

Az együttes modellek és a neurális hálózatok pontosak, de döntéseik közvetlenül nem követhetők. A modellfüggetlen értelmezési eszközök közül kettő különösen elterjedt (Molnar, 2022):

- A **permutációs fontosság** egy változó jelentőségét azzal méri, mennyit romlik a modell teljesítménye egy (lehetőleg független) teszthalmazon, ha a változó értékeit véletlenszerűen összekeverjük, és így a célváltozóval való kapcsolatát megszüntetjük (Breiman, 2001; Fisher és mtsai., 2019). Korrelált

változóknál óvatosan értelmezendő: ha két változó ugyanazt az információt hordozza, egyenként mindkettő fontossága kicsinek tűnhet.
- A **parciális függés** ábrája (Friedman, 2001) azt mutatja meg, hogyan változik a modell átlagos előrejelzése egy változó függvényében, a többi változó tapasztalati eloszlása felett átlagolva.

Ezek az eszközök a *modellt* magyarázzák, nem a valóságot: egy változó nagy fontossága azt jelenti, hogy a modell támaszkodik rá, nem azt, hogy okozati hatása van.

9.9. Gyakorlat: lemorzsolódás, minőség és hibadiagnózis

9.9.1. Modellek összehasonlítása a lemorzsolódási feladaton

FUTÓ ESET · PANNONSHOP

A 7. fejezet lemorzsolódási tábláján hét modellcsaládot hasonlítunk össze rétegzett 5-szörös keresztvalidációval. A távolság- és gradiensalapú módszereknél az előfeldolgozás (logaritmus, standardizálás) a folyamat része.

```
import time
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import FunctionTransformer, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier,
↪  HistGradientBoostingClassifier
from sklearn.model_selection import StratifiedKFold, cross_validate,
↪  GridSearchCV

churn = ab.churn_table(cutoff="2025-07-01", horizon_days=183)
jell = ["R_nap", "F_szamla", "M_ertek", "atl_kosarertek", "atl_kosarmeret",
↪  "atl_mennyiseg",
        "elso_vasarlas_nap", "akcios_arany", "arany_kave", "arany_gep",
        ↪  "arany_konyha",
        "arany_dekor", "arany_ajandek", "arany_iroszer"]
X, y = churn[jell].astype(float), churn.lemorzsolodik
```

```
elo = make_pipeline(SimpleImputer(strategy="median"),
 ↪  FunctionTransformer(np.log1p), StandardScaler())
modellek = {
    "logisztikus regresszió": make_pipeline(elo,
     ↪  LogisticRegression(max_iter=2000)),
    "döntési fa (mélység: CV)":
     ↪  GridSearchCV(make_pipeline(SimpleImputer(strategy="median"),
                                   DecisionTreeClassifier(random_state=0)),
                                  {"decisiontreeclassifier__max_depth": [2,
 ↪  3, 4, 5, 6, 8]}, cv=5, scoring="roc_auc"),
    "naiv Bayes (Gauss)": make_pipeline(elo, GaussianNB()),
    "k-NN (k=25)": make_pipeline(elo,
     ↪  KNeighborsClassifier(n_neighbors=25)),
    "SVM (RBF-kernel)": make_pipeline(elo, SVC(C=1.0, gamma="scale",
     ↪  probability=True, random_state=0)),
    "véletlen erdő": make_pipeline(SimpleImputer(strategy="median"),
                                   RandomForestClassifier(n_estimators=400,
 ↪  min_samples_leaf=5, random_state=0, n_jobs=-1)),
    "gradiens-boosting": HistGradientBoostingClassifier(max_iter=300,
     ↪  learning_rate=0.05, random_state=0),
}
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
sorok = {}
for nev, mod in modellek.items():
    t0 = time.perf_counter()
    r = cross_validate(mod, X, y, cv=cv, scoring=["roc_auc",
 ↪  "average_precision", "balanced_accuracy"])
    sorok[nev] = {"AUC": r["test_roc_auc"].mean(), "AUC szórás":
 ↪  r["test_roc_auc"].std(),
                  "átl. pontosság": r["test_average_precision"].mean(),
                 "kiegy. helyes arány": r["test_balanced_accuracy"].mean(),
                   ↪  "idő [s]": time.perf_counter() - t0}
pd.DataFrame(sorok).T.sort_values("AUC", ascending=False).round(3)
```

	AUC	AUC szórás	átl. pontosság	kiegy. helyes arány	idő [s]
véletlen erdő	0.822	0.010	0.753	0.737	5.090
SVM (RBF-kernel)	0.808	0.014	0.734	0.742	1.400
k-NN (k=25)	0.799	0.010	0.698	0.711	0.187
gradiens-boosting	0.798	0.015	0.722	0.711	3.054
logisztikus regresszió	0.790	0.015	0.700	0.719	0.119
naiv Bayes (Gauss)	0.770	0.010	0.669	0.713	0.095
döntési fa (mélység: CV)	0.768	0.017	0.658	0.708	2.131

A legjobb eredményt a véletlen erdő adja (AUC ≈ 0,82), de a különbségek kicsik: a logisztikus regresszió csak mintegy 0,03-dal marad el tőle, az SVM és a k-NN a kettő között helyezkedik el. A gradiens-boosting alapbeállításokkal nem jobb a véletlen erdőnél – a boosting érzékenyebb a hiperparaméterekre (tanulási ráta, fák mérete, iterációk száma), és ezen a mindössze kétezer vevőt tartalmazó

táblán hajlamos a túltanulásra. Az egyetlen döntési fa és a naiv Bayes a leggyengébb. A tanulság tipikus: egy jól előkészített, kevés jellemzős táblán (itt a 3. fejezet RFM-alapú változóival) már az egyszerű, értelmezhető modell is a lehetséges teljesítmény nagy részét eléri, a jó jellemzők többet érnek a bonyolultabb modellnél. (Mivel minden modellt ugyanazokon a részhalmazokon értékeltünk, a különbségek megbízhatóságát pontosabban a részhalmazonkénti, páros különbségek mutatják – lásd a feladatokat.)

A logisztikus regresszió együtthatói esélyhányadosként értelmezhetők. Mivel a változókat logaritmus után standardizáltuk, egy együttható azt mutatja meg, hányszorosára változik a lemorzsolódás esélye, ha a változó (logaritmusa) egy szórásnyival nő:

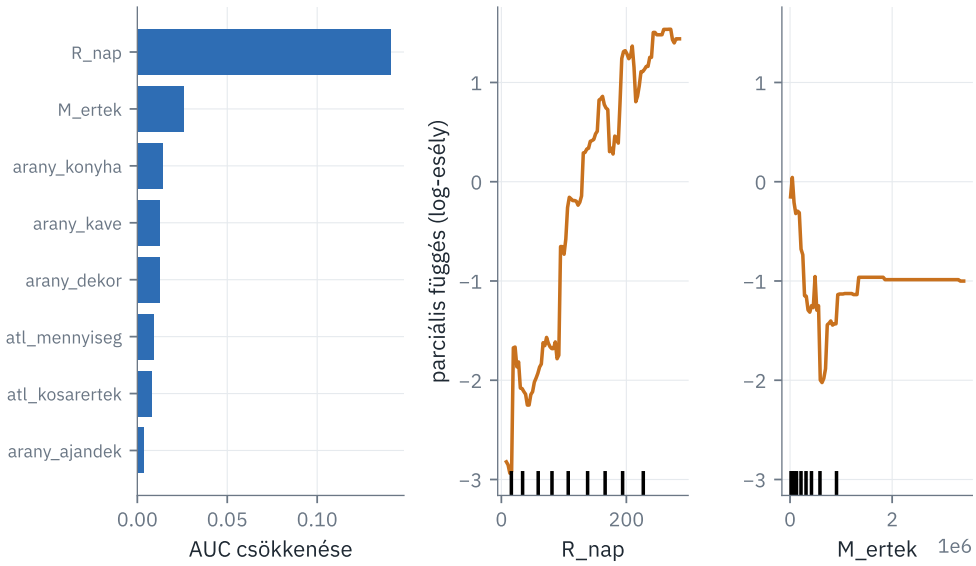
```
lr = modellek["logisztikus regresszió"].fit(X, y)
eh = pd.Series(np.exp(lr[-1].coef_[0]), index=jell, name="esélyhányados (1
↪  szórás)")
eh.sort_values().round(2).to_frame()
```

	esélyhányados (1 szórás)
F_szamla	0.41
atl_kosarmeret	0.77
atl_kosarertek	0.80
atl_mennyiseg	0.82
akcios_arany	0.91
elso_vasarlas_nap	1.19
arany_kave	1.30
M_ertek	1.31
arany_konyha	1.37
arany_ajandek	1.48
arany_dekor	1.59
arany_iroszer	1.63
R_nap	1.81
arany_gep	1.84

A legerősebb hatású változó a vásárlási gyakoriság: egy szórásnyival (logaritmikus skálán) gyakoribb vásárlás mellett a lemorzsolódás esélye kevesebb mint felére csökken. A legutóbbi vásárlás óta eltelt idő növeli a kockázatot – ez az RFM-modell „recency” komponensének klasszikus megfigyelése –, és hasonló erősségű a gépvásárlások aránya: aki főként egyszeri, nagy értékű eszközt vett, ritkábban tér vissza. Az együtthatókat azonban óvatosan kell értelmezni, mert *feltételes* hatásokat írnak le: a vásárlási érték (M) esélyhányadosa például azért nagyobb 1-nél, mert a gyakoriság (F) rögzítése mellett a nagyobb költés kevesebb, de nagyobb értékű vásárlást jelent. Erősen korrelált jellemzőknél (az R, F, M változók és az összegükben 1-et adó kategóriaarányok) az egyes együtthatók külön-külön nem értelmezhetők okozati hatásként.

A gradiens-boosting modellt permutációs fontossággal és parciális függéssel értelmezzük, egy visszatartott teszthalmazon:

```
from sklearn.model_selection import train_test_split
from sklearn.inspection import permutation_importance,
↪  PartialDependenceDisplay
Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.3, stratify=y,
↪  random_state=0)
gb = modellek["gradiens-boosting"].fit(Xtr, ytr)
pi = permutation_importance(gb, Xte, yte, scoring="roc_auc", n_repeats=20,
↪  random_state=0)
fontos = pd.Series(pi.importances_mean, index=jell).sort_values()
fig, ax = plt.subplots(1, 3, gridspec_kw={"width_ratios": [1.4, 1, 1]})
ax[0].barh(fontos.index[-8:], fontos.values[-8:], color=ab.PALETTE[0])
ax[0].set_xlabel("AUC csökkenése"); ax[0].tick_params(axis="y",
↪  labelsize=8)
PartialDependenceDisplay.from_estimator(gb, Xtr, [fontos.index[-1],
↪  fontos.index[-2]], ax=ax[1:],
                                        line_kw={"color": ab.PALETTE[1]})
ax[1].set_ylabel("parciális függés (log-esély)"); ax[2].set_ylabel("")
fig.tight_layout()
```



9.5. ábra. A gradiens-boosting lemorzsolódási modell értelmezése: permutációs fontosság a teszthalmazon (balra, az AUC csökkenése) és a két legfontosabb változó parciális függése (jobbra).

A gradiens-boosting modell döntéseiben messze a legutóbbi vásárlás óta eltelt idő a legfontosabb: a parciális függés szerint a lemorzsolódás log-esélye nagyjából három hónap inaktivitás után kezd meredeken nőni. Figyelemre méltó, hogy a lo-

gisztikus regresszióban legerősebb vásárlási gyakoriság itt nincs is az első nyolc változó között. Ez nem ellentmondás, hanem a permutációs fontosság fentebb említett korlátja: a gyakoriság erősen korrelál a vásárlási értékkel (Spearman-féle $\rho \approx 0{,}74$) és mérsékelten a recency változóval ($\rho \approx -0{,}47$), így a fák az információt jórészt ezeken keresztül „veszik fel”, és a gyakoriság összekeverése alig rontja a modellt. A két modell értelmezése tehát egymást kiegészítve, a változók közötti összefüggések ismeretében helyes.

9.9.2. A termékminőség előrejelzése

FUTÓ ESET · PANNON ÜZEM

A 7. fejezetben láttuk, hogy a lineáris logisztikus regresszió a minőségi feladaton gyenge, mert a selejt a célértéktől mindkét irányban való eltérésből adódhat. Hasonlítsuk össze a nemlineáris modelleket idősoros felosztással (az utolsó negyedév a teszthalmaz)!

```
from sklearn.metrics import roc_auc_score, average_precision_score
q = ab.quality_table()
vlt = ["T_kornyezet", "F_betap", "c_betap", "T_reaktor", "F_hutoviz",
 ↪  "T_hutoviz",
       "P_reaktor", "P_kevero", "E_fogyasztas", "dT"]
vagas = q.ido >= "2025-10-01"
Xq_tr, Xq_te = q.loc[~vagas, vlt], q.loc[vagas, vlt]
yq_tr, yq_te = (~q.megfelelo[~vagas]).astype(int),
 ↪  (~q.megfelelo[vagas]).astype(int)
mq = {
    "logisztikus regresszió": make_pipeline(StandardScaler(),
     ↪  LogisticRegression(max_iter=2000)),
    "döntési fa (mélység 5)": DecisionTreeClassifier(max_depth=5,
     ↪  random_state=0),
    "döntési fa (levélméret ≥ 10)":
     ↪  DecisionTreeClassifier(min_samples_leaf=10, random_state=0),
    "k-NN (k=15)": make_pipeline(StandardScaler(),
     ↪  KNeighborsClassifier(n_neighbors=15)),
    "SVM (RBF-kernel)": make_pipeline(StandardScaler(), SVC(C=3.0,
     ↪  probability=True, random_state=0)),
    "véletlen erdő": RandomForestClassifier(n_estimators=400,
     ↪  min_samples_leaf=3, random_state=0, n_jobs=-1),
    "gradiens-boosting": HistGradientBoostingClassifier(max_iter=300,
     ↪  learning_rate=0.05, random_state=0),
}
pd.DataFrame({nev: {"AUC": roc_auc_score(yq_te, (p := m.fit(Xq_tr,
 ↪  yq_tr).predict_proba(Xq_te)[:, 1])),
```

```
                  "átl. pontosság": average_precision_score(yq_te, p)} for
                    ↪  nev, m in mq.items()}).T \
    .sort_values("AUC", ascending=False).round(3)
```

	AUC	átl. pontosság
véletlen erdő	0.962	0.698
gradiens-boosting	0.958	0.732
döntési fa (levélméret ≥ 10)	0.930	0.545
SVM (RBF-kernel)	0.823	0.474
k-NN (k=15)	0.748	0.187
döntési fa (mélység 5)	0.563	0.235
logisztikus regresszió	0.528	0.138

Itt a nemlineáris modellek nagy fölényben vannak: a fa alapú együttes modellek a termékenként eltérő célérték körüli kétirányú eltéréseket és a hőmérséklet-túllépéseket is képesek leírni. Tanulságos a két döntési fa összevetése. Az 5 mélységű fa alig jobb a véletlennél, mert a selejt feltétele termékenként más-más koncentráció- és hőmérséklet-tartományt jelent, és ennek leírásához sok egymásba ágyazott vágás kell. A mélység korlátozása helyett a levelek minimális méretét előíró fa – amely mélyre nőhet, de apró, zajra illeszkedő leveleket nem hozhat létre – már közel jár az együttes modellekhez. A regularizáció *módja* tehát legalább annyira fontos, mint az erőssége.

9.9.3. Hibadiagnózis: többosztályos osztályozás

FUTÓ ESET · PANNON ÜZEM

Végül egy többosztályos feladat: a 10 perces folyamatadatokból azt szeretnénk felismerni, milyen hiba áll fenn éppen (ha van). A modellt az első félév adatain tanítjuk, és a második félévben értékeljük – a második félév hibái tehát a modell számára új eseményeket jelentenek. A nyers mért értékek helyett a hibák fizikai „ujjlenyomatát” leíró jellemzőket számolunk:

- a hőmérséklet eltérését az alapjeltől és a hűtővíz hőmérsékletét (hűtővíz-kiesés);
- a hűtővízáram egyórás szórását (a szelepakadás oszcillációt okoz);
- a hűtővízáram és az energiafogyasztás eltérését a heti mediántól;
- a betáplált koncentráció eltérését a termék névleges értékétől (betáplálási zavar);
- egy **analitikus redundanciát**: a nyomás a valódi hőmérséklettől és a terheléstől függ, ezért a $P - 0{,}025\,T_{\text{mért}} - 0{,}04\,F_{\text{betáp}}$ maradék eltolódása a hőmérséklet-érzékelő hibájára utal (szenzor-drift).

```
m = ab.plant_clean()
m = m[~m.leallas].copy()
m["hiba"] = ab.load_plant().truth["fault"].reindex(m.ido).to_numpy()
mi = m.set_index("ido")
felev_vege = "2025-06-30"

F = pd.DataFrame(index=mi.index)
F["dT"] = mi.T_reaktor - mi.T_alapjel
F["T_hutoviz"] = mi.T_hutoviz
F["T_hut_valt_1h"] = mi.T_hutoviz.diff(6)
F["F_hut_szoras_1h"] = mi.F_hutoviz.rolling("60min").std()
F["F_hut_rel"] = mi.F_hutoviz.rolling("6h").mean() /
 ↪  mi.F_hutoviz.rolling("7D").median() - 1
F["E_rel"] = mi.E_fogyasztas.rolling("6h").mean() /
 ↪  mi.E_fogyasztas.rolling("7D").median() - 1
# a betáplált koncentráció a termék névleges értékéhez képest (névleges
 ↪  érték: a tanítóadatok mediánja)
nevl = mi.loc[:felev_vege].groupby("termek").c_betap.median()
F["c_be_rel"] = (mi.c_betap / mi.termek.map(nevl).astype(float) -
 ↪  1).rolling("60min").mean()
# analitikus redundancia: nyomásmaradék a mért hőmérséklet és a terhelés
 ↪  alapján
P_rez = mi.P_reaktor - 0.025 * mi.T_reaktor - 0.04 * mi.F_betap
F["P_rezid"] = P_rez.rolling("6h").mean() - P_rez.rolling("7D").median()
F = F.dropna()
yh = mi.loc[F.index, "hiba"]
tanit = F.index <= felev_vege

def diagnozis(oszlopok):
    mod = HistGradientBoostingClassifier(max_iter=300, learning_rate=0.1,
                                         class_weight="balanced",
 ↪  random_state=0)
    return mod.fit(F.loc[tanit, oszlopok], yh[tanit]).predict(F.loc[~tanit,
     ↪  oszlopok])

from sklearn.metrics import recall_score, precision_score, f1_score
oszt = ["normál", "szenzor_drift", "betap_zavar", "szelep_akadas",
 ↪  "hutoviz_kieses"]
nyers = ["dT", "T_hutoviz", "T_hut_valt_1h", "F_hut_szoras_1h",
 ↪  "F_hut_rel", "E_rel"]
valtozatok = {"ujjlenyomat nélkül a betáplálásra és a driftre": nyers,
              "+ névleges koncentráció": nyers + ["c_be_rel"],
              "+ nyomásmaradék (teljes)": nyers + ["c_be_rel", "P_rezid"]}
eredm, pred = {}, None
for nev, cols in valtozatok.items():
    pred = diagnozis(cols)
    eredm[nev] = pd.Series(recall_score(yh[~tanit], pred, labels=oszt,
 ↪  average=None, zero_division=0), index=oszt)
```

```
    eredm[nev]["makro-F1"] = f1_score(yh[~tanit], pred, labels=oszt,
↪   average="macro", zero_division=0)
pd.DataFrame(eredm).round(2).rename_axis("felidézés (recall)")
```

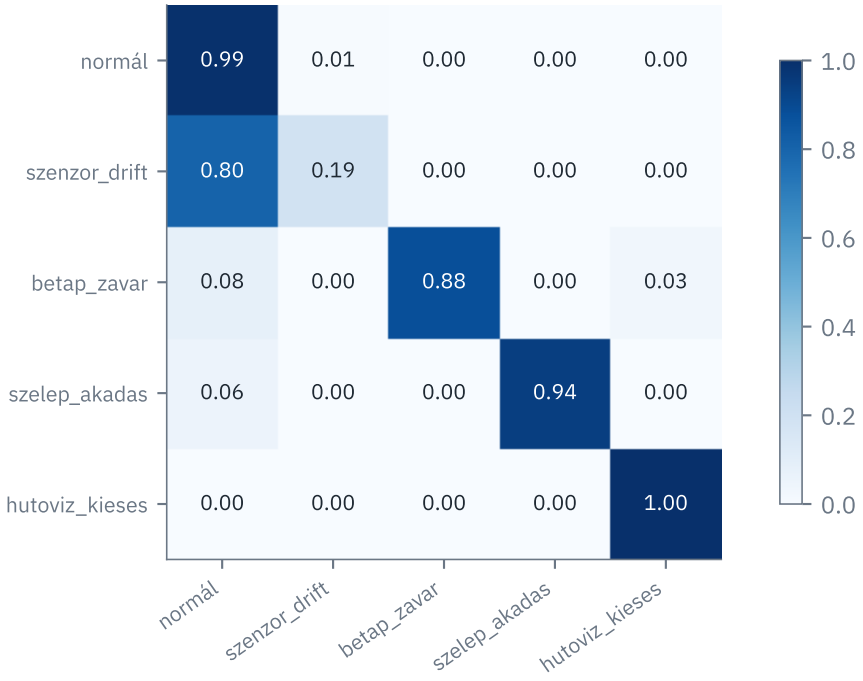
felidézés (recall)	ujjlenyomat nélkül a betáplálásra és a driftre	+ névleges koncentráció	+ nyomásmaradék (teljes)
normál	0.98	0.99	0.99
szenzor_drift	0.01	0.01	0.19
betap_zavar	0.01	0.84	0.88
szelep_akadas	0.97	0.91	0.94
hutoviz_kieses	1.00	1.00	1.00
makro-F1	0.59	0.77	0.83

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(yh[~tanit], pred, labels=oszt, normalize="true")
fig, ax = plt.subplots()
im = ax.imshow(cm, cmap="Blues", vmin=0, vmax=1)
ax.set_xticks(range(5), oszt, rotation=35, ha="right", fontsize=8);
↪ ax.set_yticks(range(5), oszt, fontsize=8)
for i in range(5):
    for j in range(5):
        ax.text(j, i, f"{cm[i, j]:.2f}", ha="center", va="center",
↪   fontsize=8,
                color="white" if cm[i, j] > 0.6 else "#1F2933")
ax.grid(False); fig.colorbar(im, ax=ax, shrink=0.8)
fig.tight_layout()
```

FUTÓ ESET · PANNON ÜZEM

A táblázat a szakterületi tudás értékét mutatja. A hűtővíz felmelegedése és a szelep akadozása már az általános jellemzőkből is jól felismerhető, a betáplálási zavar viszont csak akkor, ha a koncentrációt a *termék névleges értékéhez* viszonyítjuk: a 15%-os koncentrációcsökkenés ugyanis nagyságrendileg akkora, mint a termékek névleges értékei közötti különbség, így a nyers koncentrációból a modell nem tudja elválasztani a termékváltástól. A szenzor-drift a legnehezebb eset: a mérési hiba öt nap alatt nulláról fokozatosan nő, és a szabályozó a hibás mérést „követi”, így az epizód első felében a jelek még a zajszinten belül vannak; a nyomásmaradék csak a drift későbbi szakaszában jelez. A felügyelt hibadiagnózis másik korlátja, hogy csak a tanítóadatokban már előfordult hibatípusokat ismerheti fel – itt minden hibatípusból csak egy-három esemény áll rendelkezésre a tanításhoz –, egy teljesen új hiba pedig a „normál” vagy egy rossz osztályba kerül. Ezért a gyakorlatban a felügyelt diagnózist a 11. fejezet felügyelet nélküli anomália-észlelésével kombinálják.

Interaktív alkalmazás: Döntési fa építése: vágások, entrópia, Gini-index



9.6. ábra. A teljes jellemzőkészlettel tanított hibadiagnosztikai modell sorra normált tévesztési mátrixa a második félév adatain (sorok: valódi állapot, oszlopok: előrejelzett állapot).

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/dontesi-fa.html

Colab-notebook: 9. fejezet – osztályozás

9.10. Összefoglalás

- A logisztikus regresszió a log-esélyt modellezi lineárisan; a kereszt-entrópia konvex, az együtthatók esélyhányadosként értelmezhetők.
- A döntési fák mohó rekurzív felosztással, a szennyezettség (entrópia, Gini-index) csökkentésével épülnek; jól értelmezhetők, de nagy varianciájúak, ezért metszeni kell őket.
- A naiv Bayes a jellemzők feltételes függetlenségét feltételezi; egyszerű és gyors, a döntései gyakran jók, a valószínűségei kevésbé.
- A k-NN lusta, távolságalapú módszer; érzékeny a skálázásra és a dimenzió átkára.
- Az SVM a margót maximalizálja; a kerneltrükkel nemlineáris döntési határokat is tanul.
- Az együttes modellek közül a véletlen erdő a varianciát, a gradiens-boosting a torzítást csökkenti; táblázatos adatokon ezek a legerősebb általános célú modellek.
- A permutációs fontosság és a parciális függés a modellt magyarázza, nem az okozati viszonyokat.

Ellenőrző kérdések

1. Mit jelent, ha egy logisztikus regressziós együttható esélyhányadosa 2?
2. Miért részesíti előnyben az információnyereség a sok értékű változókat, és hogyan korrigálja ezt a nyereségarány?
3. Miért hasznos a Laplace-simítás a naiv Bayes-osztályozóban?
4. Hogyan hat a k értéke a k-NN torzítására és varianciájára?
5. Mik a támaszvektorok, és miért elegendő csak ezeket ismerni a döntéshez?
6. Miért csökkenti a véletlen erdő a változók véletlen részhalmazából történő vágással a fák közötti korrelációt, és miért javít ez a (9.11) szerint?
7. Miben különbözik a zsákolás és a boosting stratégiája?

Feladatok

1. **Kézi fa.** Számítsa ki a kézi példa gyökérvágásához a Gini-index szerinti nyereséget is! Ugyanaz a változó kerül a gyökérbe?
2. **Nyereségarány.** Egészítse ki a 9.1. táblázatot egy „Tételszám” oszloppal (1–14)! Mekkora az információnyeresége és a nyereségaránya? Mi a tanulság?
3. **Metszés.** Tanítson teljes mélységű döntési fát a lemorzsolódási adatokon, majd határozza meg a költség–komplexitás metszés α paraméterét keresztvalidációval (`cost_complexity_pruning_path`)! Hány levele van a metszett fának, és mennyire értelmezhetők a szabályai?
4. **SVM hangolása.** Hangolja a C és γ paramétert rácskereséssel a minőségi feladaton (idősoros keresztvalidációval)! Ábrázolja a validációs AUC-t hőtérképen!
5. **Új hibatípus.** Hagyja ki a tanításból az összes hűtővíz-kiesést, és vizsgálja meg, milyen osztályba sorolja őket a modell a teszthalmazon! Javasoljon eljárást az ismeretlen hibák felismerésére (segítség: a becsült osztályvalószínűségek)!

További olvasmányok

Az osztályozási módszerek átfogó tárgyalása Hastie és mtsai. (2009) és James és mtsai. (2023) könyvében, a valószínűségi megközelítésé Bishop (2006) könyvében található. A döntési fák alapművei Quinlan (1986), Quinlan (1993) és Breiman és mtsai. (1984); a támaszvektor-gépeké Cortes & Vapnik (1995). Az együttes modellekről Breiman (1996), Breiman (2001) és Friedman (2001) eredeti cikkei, a modellek értelmezéséről Molnar (2022) könyve ajánlott.

10. Neurális hálózatok és mélytanulás

Miről szól a fejezet?

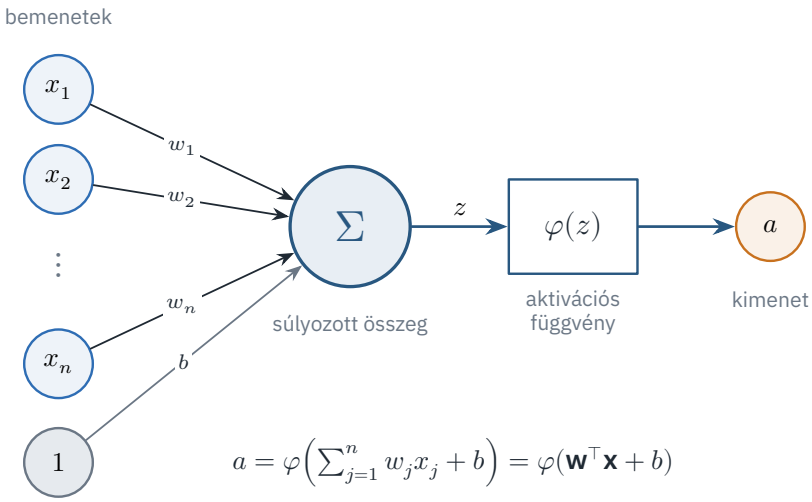
A mesterséges neurális hálózatok rugalmas, sok paraméteres függvényosztályok, amelyek paramétereit gradiensalapú optimalizálással illesztjük az adatokra. A fejezet a legegyszerűbb építőelemtől, a mesterséges neurontól indul, majd a többrétegű hálózatokat, a tanításukat lehetővé tevő hibavisszaterjesztést, az optimalizálás és a regularizáció gyakorlati eszközeit tárgyalja. Ezután a mélytanulás három, az adatbányászatban fontos architektúráját mutatjuk be: a konvolúciós, a rekurrens hálózatokat és az autoenkódereket. A gyakorlatban egy hálózatot NumPy-ban, „nulláról" programozunk le, majd a `scikit-learn` és a PyTorch eszközeivel egy hűtési igény modellt, egy autoenkóderes anomáliaészlelőt és egy konvolúciós hibadiagnosztikai modellt készítünk. A nagy nyelvi modellek (LLM-ek) és a transzformer-architektúra a könyv keretein kívül esnek.
A fejezet végére az olvasó képes lesz egy többrétegű hálózat előre- és hibavisszaterjesztését levezetni és vektorosan leprogramozni; a gradienst numerikusan ellenőrizni; megválasztani a veszteségfüggvényt, az optimalizálót és a regularizációt; felismerni, mikor érdemes – és mikor nem – mély hálózatot használni; valamint egy konvolúciós hálózatot és egy autoenkódert idősoros folyamatadatokra alkalmazni.

10.1. A mesterséges neuron

A mesterséges neuron (McCulloch & Pitts, 1943) a bemenetek súlyozott összegét egy nemlineáris **aktivációs függvényen** vezeti át (10.1. ábra):

$$a = \varphi\Big(\sum_{j=1}^{n} w_j x_j + b\Big) = \varphi(\mathbf{w}^\top \mathbf{x} + b). \tag{10.1}$$

Ha φ a szigmoid függvény, a neuron éppen a logisztikus regresszió (9.2. alfejezet); ha a lépcsőfüggvény, a **perceptron** (Rosenblatt, 1958). A perceptron tanítási szabálya minden hibásan osztályozott ($y_k \in \{-1, +1\}$) mintánál a súlyokat a



10.1. ábra. A mesterséges neuron: a bemenetek súlyozott összegét és az eltolást (b) egy nemlineáris aktivációs függvény képezi le a kimenetre.

minta irányába mozdítja: $\mathbf{w} \leftarrow \mathbf{w} + \eta\, y_k \mathbf{x}_k$. Lineárisan elválasztható adatokon ez véges számú lépésben elválasztó hipersíkot talál, de egyetlen neuron csak lineáris döntési határt képes leírni: a kizáró vagy (XOR) függvényt például nem tudja megtanulni. Ezt a korlátot Minsky & Papert (1969) elemezte részletesen, és csak a többrétegű hálózatok és a tanításukat lehetővé tevő hibavisszaterjesztés oldotta fel.

10.2. Többrétegű hálózatok

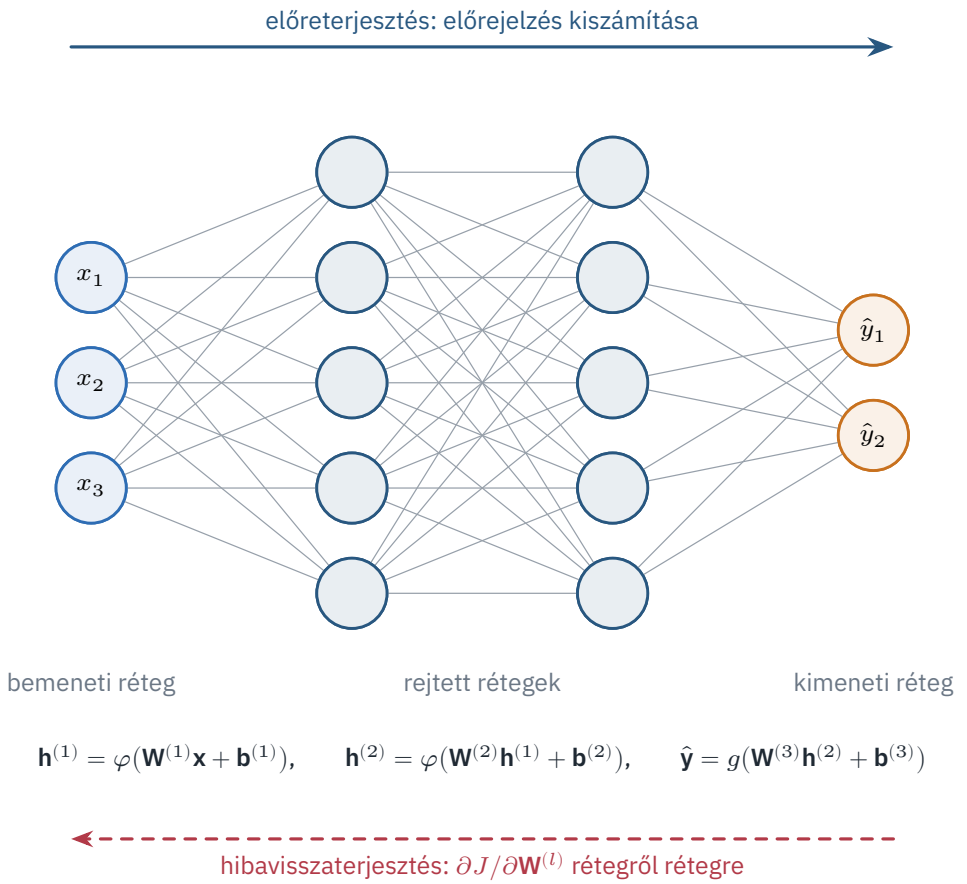
10.2.1. Felépítés és aktivációs függvények

A **többrétegű perceptron** (multilayer perceptron, MLP) neuronjai rétegekbe szerveződnek, és minden réteg az előző réteg kimenetét kapja bemenetként (10.2. ábra). Egy L rétegű hálózat egymásba ágyazott függvények kompozíciója:

$$\mathbf{h}^{(0)} = \mathbf{x}, \qquad \mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}, \qquad \mathbf{h}^{(l)} = \varphi\big(\mathbf{z}^{(l)}\big), \quad l = 1, \dots, L-1, \qquad \hat{\mathbf{y}} = g(\tag{10.2}$$

ahol $\mathbf{W}^{(l)}$ az l-edik réteg súlymátrixa, φ a rejtett rétegek elemenként alkalmazott aktivációs függvénye, g pedig a feladathoz illő kimeneti függvény: regressziónál az identitás, bináris osztályozásnál a szigmoid, többosztályos osztályozásnál a

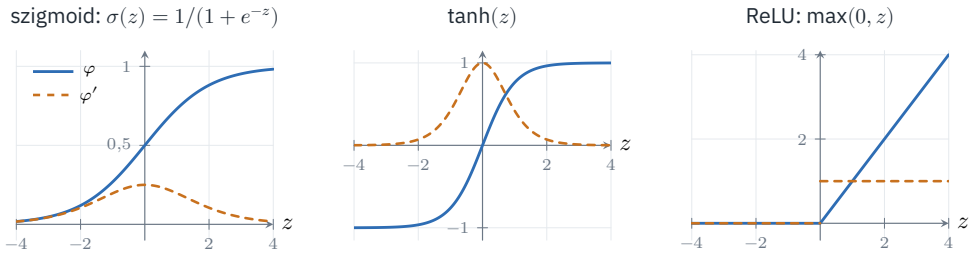
softmax. A nemlinearitás lényeges: ha φ lineáris volna, a rétegek egyetlen lineáris leképezéssé vonhatók össze, és a mély hálózat sem tudna többet a lineáris modellnél.



10.2. ábra. Kétrejtettrétegű, teljesen összekötött hálózat. Az előreterjesztés a bemenetből kiszámítja az előrejelzést, a hibavisszaterjesztés a kimenettől a bemenet felé haladva számítja ki a súlyok szerinti gradienseket.

Az aktivációs függvények közül a szigmoid és a tanh történetileg a legfontosabb, ma azonban a rejtett rétegekben jellemzően a **ReLU** (rectified linear unit, $\varphi(z) = \max(0, z)$) (Nair & Hinton, 2010) és változatai terjedtek el (10.3. ábra). Ennek oka a deriváltakban rejlik: a szigmoid deriváltja legfeljebb 0,25, és a függvény telítődő szakaszain közel nulla. A hibavisszaterjesztés során a gradiens rétegenként ilyen tényezőkkel szorzódik, ezért mély hálózatokban a bemenethez közeli rétegek gradiense exponenciálisan kicsivé válhat (**elenyésző gradiens**). A ReLU deriváltja a

pozitív tartományban 1, így ezt a problémát nagyrészt elkerüli, ráadásul kiszámítása is olcsó.



10.3. ábra. Gyakori aktivációs függvények (folytonos vonal) és deriváltjaik (szaggatott vonal). A szigmoid és a tanh deriváltja a telítődő tartományokban nullához tart, a ReLU deriváltja a pozitív tartományban állandó.

10.2.2. Univerzális approximáció

Az univerzális approximációs tétel (Cybenko, 1989; Hornik, 1991) szerint egyetlen, elegendően sok neuront tartalmazó rejtett réteggel bármely kompakt halmazon folytonos függvény tetszőleges pontossággal közelíthető. A tétel *létezést* mond ki: nem állítja, hogy a szükséges neuronszám kicsi, sem azt, hogy a gradiensalapú tanítás meg is találja a megfelelő súlyokat, és semmit sem mond az általánosításról. A gyakorlat azt mutatja, hogy a mélyebb (több rétegű) hálózatok ugyanannyi paraméterrel sok feladatban hatékonyabban reprezentálják a hierarchikus szerkezetű összefüggéseket, mint a széles, sekély hálózatok (Goodfellow és mtsai., 2016; Prince, 2023).

10.3. A hálózatok tanítása

10.3.1. Veszteségfüggvények

A hálózat paramétereit – az összes $\mathbf{W}^{(l)}$ és $\mathbf{b}^{(l)}$ együttesét $\boldsymbol{\theta}$-val jelölve – a (1.3) regularizált tapasztalati kockázat minimalizálásával határozzuk meg. A veszteségfüggvényt a feladat valószínűségi modellje adja: regressziónál normális eloszlású hibát feltételezve a négyzetes hiba, $\ell = \frac{1}{2}\|\hat{\mathbf{y}} - \mathbf{y}\|^2$; osztályozásnál a kategorikus eloszlás negatív log-likelihoodja, a **kereszt-entrópia**, $\ell = -\sum_c y_c \ln \hat{y}_c$, ahol $\mathbf{y}$ a valódi osztály egy-forró kódolása. A kimeneti függvény és a veszteség párosítása

nem véletlen: a szigmoid–kereszt-entrópia és a softmax–kereszt-entrópia párosnál a veszteség kimeneti réteg előtti $\mathbf{z}^{(L)}$ szerinti gradiense egyszerűen $\hat{\mathbf{y}} - \mathbf{y}$, és nem telítődik.

10.3.2. Hibavisszaterjesztés

A veszteség nem konvex a paraméterekben, ezért gradiensalapú iteratív módszerekkel minimalizáljuk. A gradiens hatékony kiszámításának módszere a **hibavisszaterjesztés** (backpropagation) (Rumelhart és mtsai., 1986), amely a láncszabály szisztematikus alkalmazása a hálózat számítási gráfján (10.4. ábra). Vezessük be az l-edik réteg **hibajelét**, $\boldsymbol{\delta}^{(l)} = \partial \ell / \partial \mathbf{z}^{(l)}$! A kimeneti rétegben (a fenti párosítás mellett) $\boldsymbol{\delta}^{(L)} = \hat{\mathbf{y}} - \mathbf{y}$, a rejtett rétegekben pedig a láncszabály szerint

$$\boldsymbol{\delta}^{(l)} = \left(\mathbf{W}^{(l+1)\top} \boldsymbol{\delta}^{(l+1)}\right) \odot \varphi'\left(\mathbf{z}^{(l)}\right), \tag{10.3}$$

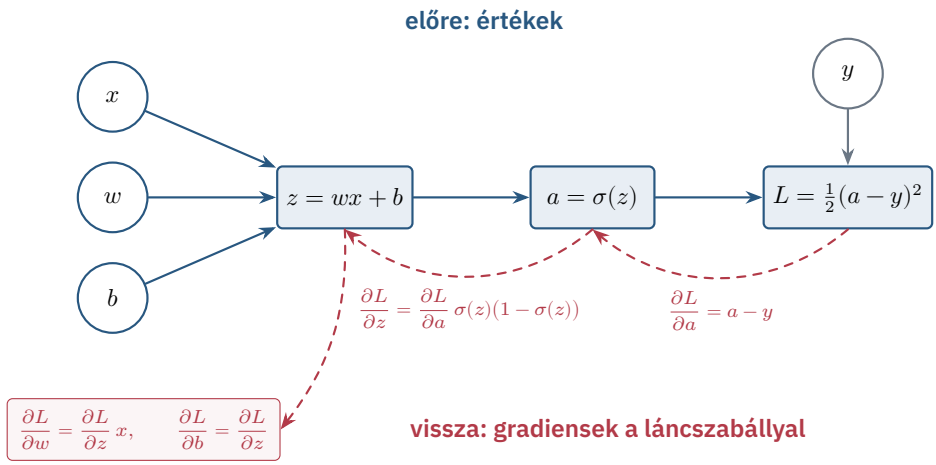
ahol $\odot$ az elemenkénti szorzás. Ebből a paraméterek szerinti gradiensek:

$$\frac{\partial \ell}{\partial \mathbf{W}^{(l)}} = \boldsymbol{\delta}^{(l)} \, \mathbf{h}^{(l-1)\top}, \qquad \frac{\partial \ell}{\partial \mathbf{b}^{(l)}} = \boldsymbol{\delta}^{(l)}. \tag{10.4}$$

Az algoritmus tehát egy előreterjesztéssel kiszámítja és eltárolja a rétegek $\mathbf{z}^{(l)}$ és $\mathbf{h}^{(l)}$ értékeit, majd egy visszaterjesztéssel a kimenettől a bemenet felé haladva rétegenként kiszámítja a hibajeleket és a gradienseket. A teljes gradiens kiszámításának költsége így nagyságrendileg megegyezik egyetlen előreterjesztés költségével – szemben a véges differenciákkal, amelyek paraméterenként egy-egy újabb előreterjesztést igényelnének. A modern keretrendszerek (PyTorch, JAX, TensorFlow) a számítási gráfból automatikusan állítják elő a gradienst (**automatikus differenciálás** (Baydin és mtsai., 2018)); a hibavisszaterjesztés ennek a fordított módú változata.

Az algoritmus összefoglalása: egy tanítási lépés

1. Válasszunk egy B elemű mini-batch-et a tanítóhalmazból.
2. **Előreterjesztés:** számítsuk ki és tároljuk a $\mathbf{z}^{(l)}$, $\mathbf{h}^{(l)}$ értékeket és a veszteséget (10.2).
3. **Visszaterjesztés:** $\boldsymbol{\delta}^{(L)} = \hat{\mathbf{y}} - \mathbf{y}$, majd $l = L-1, \dots, 1$-re (10.3); a gradiensek (10.4) szerint, a mini-batch elemeire átlagolva.
4. **Frissítés:** módosítsuk a paramétereket az optimalizáló szabálya szerint (például $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} J$).
5. Ismételjük, amíg a validációs veszteség javul (korai leállítás).



10.4. ábra. A hibavisszaterjesztés egyetlen neuronon. Az előreterjesztés során kiszámított értékekből a láncszabály a kimenettől visszafelé haladva lépésenként adja a veszteség gradiensét.

10.3.3. Optimalizálás

A teljes tanítóhalmazon számolt gradiens helyett a gyakorlatban kis, véletlenszerűen választott **mini-batch**-eken számolt becslést használunk (**sztochasztikus gradienscsökkentés**, SGD (Robbins & Monro, 1951)). A becslés zajos, de olcsó, és a zaj segít a nem konvex veszteségfelület rossz stacionárius pontjainak elkerülésében. Az alapváltozat gyorsítására két ötlet terjedt el. A **lendület** (momentum) (Polyak, 1964; Sutskever és mtsai., 2013) a gradiensek exponenciálisan súlyozott mozgóátlagával lép, ami a „völgyekben” csillapítja az oszcillációt és gyorsítja a haladást. Az **Adam** (Kingma & Ba, 2015) emellett paraméterenként adaptív lépésközt használ a gradiens második momentumának becslésével:

$$\begin{aligned} \mathbf{m}_t &= \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t, \qquad \mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2, \\ \hat{\mathbf{m}}_t &= \frac{\mathbf{m}_t}{1-\beta_1^t}, \quad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_2^t}, \quad \boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta\,\frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \varepsilon}, \end{aligned} \tag{10.5}$$

ahol $\mathbf{g}_t$ a mini-batch gradiense, a műveletek elemenként értendők, a szokásos értékek $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$, $\varepsilon = 10^{-8}$. A torzításkorrekció ($\hat{\mathbf{m}}_t$, $\hat{\mathbf{v}}_t$) a nulláról induló mozgóátlagok kezdeti alulbecslését ellensúlyozza.

A legfontosabb hiperparaméter a **tanulási ráta** (η): túl nagy értéknél a tanítás divergál, túl kicsinél lassú. A súlyok kezdeti értéke is döntő: a súlyokat véletlenszerűen, a rétegek méretéhez skálázott szórással inicializáljuk, hogy a jelek és

a gradiensek nagysága rétegről rétegre se ne nőjön, se ne csökkenjen el. Ilyen a tanh-hoz ajánlott Glorot-inicializálás (Glorot & Bengio, 2010), amelynél a súlyok egyenletes eloszlásúak a $\pm\sqrt{6/(n_{\text{be}}+n_{\text{ki}})}$ intervallumon, és a ReLU-hoz ajánlott He-féle inicializálás (K. He és mtsai., 2015), amelynél a szórás $\sqrt{2/n_{\text{be}}}$.

10.3.4. Regularizáció

A neurális hálózatok paraméterszáma gyakran a tanítóminták számával összemérhető vagy annál nagyobb, ezért a regularizáció elengedhetetlen (Goodfellow és mtsai., 2016):

- A **súlycsökkentés** (weight decay) a (8.3. alfejezet) ridge-regressziójához hasonlóan $\lambda\|\boldsymbol{\theta}\|_2^2$ büntetőtagot ad a veszteséghez; Adam esetén a büntetést a gradienstől elkülönítve érdemes alkalmazni (AdamW, (Loshchilov & Hutter, 2019)).
- A **korai leállítás** (Prechelt, 1998) a validációs veszteséget figyeli, és a tanítást akkor állítja le (a legjobb paramétereket visszaállítva), amikor az már nem javul.
- A **kiejtés** (dropout) (Srivastava és mtsai., 2014) tanítás közben minden lépésben véletlenszerűen kinullázza a neuronok egy részét, így a hálózat nem támaszkodhat egyetlen neuronra sem; hatása sok, egymással súlyokat megosztó részhálózat átlagolásához hasonló.
- A **batch-normalizálás** (Ioffe & Szegedy, 2015) a rétegek bemenetét minibatch-enként standardizálja; elsősorban a tanítást stabilizálja és gyorsítja, emellett enyhe regularizáló hatása is van.
- Az **adatbővítés** (data augmentation) a tanítóadatokat a feladat szempontjából lényegtelen transzformációkkal (képek tükrözése, idősorok kis eltolása, zaj hozzáadása) sokszorozza meg.

10.4. Mélytanulási architektúrák

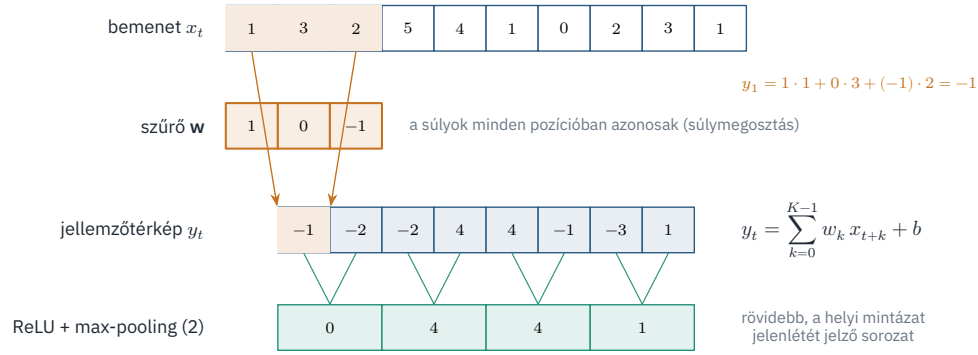
A „mély” jelző a sok egymásra épülő rétegre utal, amelyek a nyers adatokból lépésről lépésre egyre elvontabb jellemzőket tanulnak (LeCun és mtsai., 2015). A klasszikus adatbányászatban a jellemzőket mi magunk terveztük meg (lásd például a 9. fejezet hibadiagnosztikai jellemzőit); a mélytanulás ígérete, hogy elegendő adat esetén a jellemzők is a tanulás eredményei. Ennek ára a nagy adatigény és a számításigény, valamint a nehezebb értelmezhetőség. Az architektúrák abban különböznek, milyen szerkezeti feltevéseket (**induktív torzítást**) építenek be az adatokról.

10.4.1. Konvolúciós hálózatok

A **konvolúciós hálózatok** (convolutional neural network, CNN) (LeCun és mtsai., 1998) a rácsszerű adatok – képek, idő- és jelsorozatok – helyi mintázatait ismerik fel. Egydimenziós esetben egy K hosszú szűrő a bemeneti sorozat minden pozíciójában ugyanazokkal a súlyokkal számol súlyozott összeget (10.5. ábra):

$$y_t = \sum_{k=0}^{K-1} w_k \, x_{t+k} + b. \tag{10.6}$$

(Szigorúan véve ez keresztkorreláció; a mélytanulási szakirodalom mégis konvolúciónak nevezi.) Két szerkezeti feltevés teszi a CNN-eket hatékonnyá: a **helyi kapcsolódás** (egy kimenet csak a bemenet egy kis ablakától függ) és a **súlymegosztás** (ugyanaz a szűrő minden pozícióban), amelyből az eltolás-ekvivariancia következik – egy mintázatot ugyanaz a szűrő ismer fel, bárhol forduljon is elő. Többcsatornás bemenetnél (például több szenzor jelénél) a szűrő minden csatornára kiterjed, és egy réteg több szűrőt tartalmaz, amelyek mindegyike egy-egy jellemzőtérképet állít elő. Az **összevonó** (pooling) rétegek a jellemzőtérképeket ritkítják (például a szomszédos értékek maximumát véve), így a mélyebb rétegek egyre hosszabb időtávot „látnak”. Az egydimenziós CNN-ek az idősor-osztályozás és a gépállapot-figyelés erős, általános célú módszerei (Ismail Fawaz és mtsai., 2019; Kiranyaz és mtsai., 2021; Z. Wang és mtsai., 2017).



10.5. ábra. Egydimenziós konvolúció: a háromelemű szűrő (itt egy differenciaképző) végigcsúszik a bemeneten, és minden pozícióban ugyanazokkal a súlyokkal számol. A ReLU és a maximum-összevonás után egy rövidebb sorozat jelzi, hol fordult elő a keresett helyi mintázat.

10.4.2. Rekurrens hálózatok

A **rekurrens hálózatok** (recurrent neural network, RNN) sorozatokat dolgoznak fel elemről elemre, és egy $\mathbf{h}_t$ rejtett állapotban összegzik a múltat (10.6. ábra):

$$\mathbf{h}_t = \tanh\big(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h\big). \tag{10.7}$$

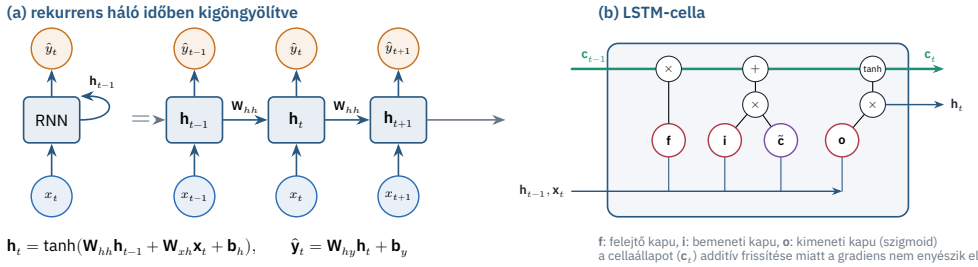
Ez egy nemlineáris állapottér-modell, amelynek paraméterei minden időlépésben azonosak. A tanítás a hálózat időben kigöngyölített változatán végzett hibavisszaterjesztéssel (backpropagation through time) történik. A hosszú sorozatokon a gradiens a $\mathbf{W}_{hh}$ sokszoros szorzata miatt elenyészik vagy felrobban, ezért az egyszerű RNN nehezen tanul hosszú távú függéseket (Bengio és mtsai., 1994). A **hosszú rövid távú memória** (long short-term memory, LSTM) (Hochreiter & Schmidhuber, 1997) ezt egy külön $\mathbf{c}_t$ cellaállapottal és három, szigmoid aktivációjú kapuval oldja meg:

$$\begin{aligned}
&\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_f), \qquad \mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_i), \quad \mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_o),\\
&\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_c), \quad \mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \qquad \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t).
\end{aligned} \tag{10.8}$$

A felejtő kapu ($\mathbf{f}_t$) dönti el, mennyit őrzünk meg a korábbi cellaállapotból, a bemeneti kapu ($\mathbf{i}_t$), hogy mennyi új információ kerül bele, a kimeneti kapu ($\mathbf{o}_t$), hogy mennyi jelenik meg belőle a rejtett állapotban. A cellaállapot additív frissítése miatt a gradiens a cellaállapot mentén sok időlépésen át is visszaterjedhet. Az egyszerűbb **GRU** (Cho és mtsai., 2014) két kapuval hasonló teljesítményt ér el. A rekurrens hálózatokat idősor-előrejelzésre, sorozatok osztályozására és anomáliaészlelésre is használják (Hundman és mtsai., 2018). A természetes nyelvi feldolgozásban az RNN-eket mára a figyelemmechanizmuson alapuló transzformerarchitektúra (Vaswani és mtsai., 2017) váltotta fel; ez a nagy nyelvi modellek alapja, tárgyalása azonban meghaladja e könyv kereteit.

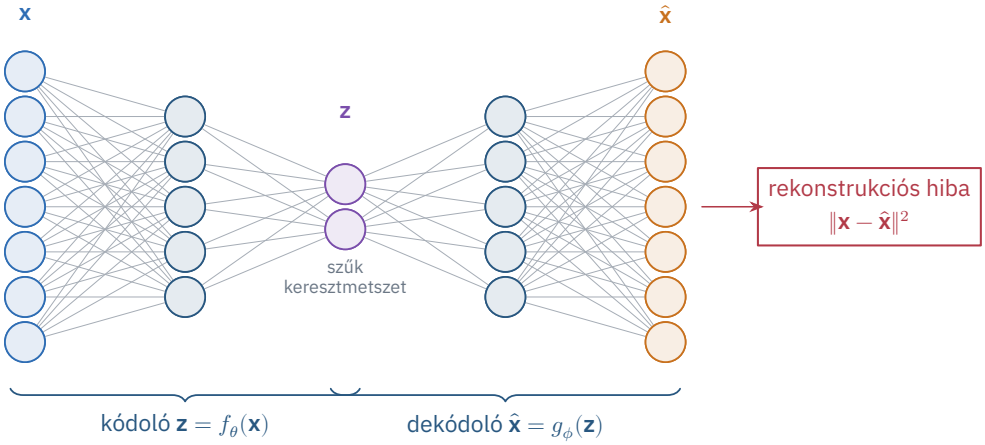
10.4.3. Autoenkóderek

Az **autoenkóder** felügyelet nélküli hálózat, amely a bemenetét igyekszik reprodukálni egy szűk keresztmetszeten át (10.7. ábra). A kódoló ($\mathbf{z} = f_\theta(\mathbf{x})$) a bemenetet egy alacsony dimenziós kódba képezi, a dekódoló ($\hat{\mathbf{x}} = g_\phi(\mathbf{z})$) ebből rekonstruálja; a tanítás a $\sum_k \|\mathbf{x}_k - g_\phi(f_\theta(\mathbf{x}_k))\|^2$ rekonstrukciós hibát minimalizálja. Lineáris aktivációk és négyzetes hiba esetén az optimális autoenkóder éppen a főkomponens-elemzés első q főkomponense által kifeszített alteret tanulja meg



10.6. ábra. (a) A rekurrens háló időben kigöngyölítve: minden időlépésben ugyanazok a súlyok dolgozzák fel a bemenetet és az előző rejtett állapotot. (b) Az LSTM-cella: a kapuk szabályozzák, mi kerül a cellaállapotba, mi marad meg belőle és mi jelenik meg a kimeneten.

(Baldi & Hornik, 1989); nemlineáris aktivációkkal az autoenkóder a PCA nemlineáris általánosítása (Hinton & Salakhutdinov, 2006; Kramer, 1991). Két fő alkalmazása a dimenziócsökkentés (a kód mint jellemző) és az **anomália-észlelés**: a csak normál működésen tanított autoenkóder a szokatlan mintákat rosszul rekonstruálja, így a rekonstrukciós hiba anomáliapontszámként használható (Pang és mtsai., 2021).



10.7. ábra. Autoenkóder: a kódoló a bemenetet egy szűk keresztmetszetű kódba sűríti, a dekódoló ebből rekonstruálja. A normál működésen tanított autoenkóder rekonstrukciós hibája a szokatlan mintákra nagy.

10.4.4. Mikor érdemes mély hálózatot használni?

A mélytanulás áttörést hozott a strukturálatlan adatok – képek, hang, szöveg, nyers jelsorozatok – feldolgozásában, ahol a jó jellemzők kézi megtervezése nehéz, és sok adat áll rendelkezésre. Táblázatos adatokon a helyzet más: közepes méretű adathalmazokon a gradiens-boosting és a véletlen erdő jellemzően legalább olyan pontos, gyorsabban tanítható és kevesebb hangolást igényel (Grinsztajn és mtsai., 2022). A neurális hálózatok mellett szól, ha a kapcsolat sima és a modellnek a tanítóadatok tartományán kívül is értelmesen kell viselkednie (a fák ott konstans értéket adnak), ha a bemenet nyers idősor vagy kép, ha több feladatot egyetlen modellel kell megoldani, vagy ha a modellt más, gradiensalapú rendszerekbe (például szabályozásba) kell beágyazni. Ellenük szól a kis adatmennyiség, az értelmezhetőség igénye és a hangolás költsége. A következő gyakorlatok mindkét esetre mutatnak példát.

10.5. Gyakorlat: hálózatok NumPy-ban, `scikit-learn`-ben és PyTorch-ban

10.5.1. Egy hálózat nulláról

Az alábbi kód egy kétrejtettrétegű ($2 \to 16 \to 16 \to 1$) hálózatot valósít meg NumPy-ban: a rejtett rétegekben tanh, a kimeneten szigmoid aktivációval, kereszt-entrópia veszteséggel, Glorot-inicializálással és Adam-optimalizálóval. A feladat a „két hold” nevű, nemlineárisan elválasztható kétosztályos mintaadathalmaz.

```
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split

X, y = make_moons(n_samples=600, noise=0.25, random_state=0)
Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.3, random_state=0)
ytr, yte = ytr[:, None].astype(float), yte[:, None].astype(float)

def init(meretek, rng):
    """Glorot-inicializálás: W ~ U(-a, a), a = sqrt(6/(n_be + n_ki))."""
    W, b = [], []
    for n_be, n_ki in zip(meretek[:-1], meretek[1:]):
        a = np.sqrt(6 / (n_be + n_ki))
        W.append(rng.uniform(-a, a, (n_be, n_ki)))
        b.append(np.zeros(n_ki))
    return W, b
```

```
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def elore(X, W, b):
    """Előreterjesztés: a rejtett rétegekben tanh, a kimeneten szigmoid."""
    A = [X]
    for l in range(len(W)):
        Z = A[-1] @ W[l] + b[l]
        A.append(sigmoid(Z) if l == len(W) - 1 else np.tanh(Z))
    return A

def veszteseg(p, y, eps=1e-12):
    return float(-np.mean(y * np.log(p + eps) + (1 - y) * np.log(1 - p +
    ↪ eps)))

def vissza(A, y, W):
    """Hibavisszaterjesztés a (-@eq-delta) és (-@eq-backprop-grad)
    ↪ egyenletek szerint."""
    N = len(y)
    dW, db = [None] * len(W), [None] * len(W)
    delta = (A[-1] - y) / N                          # szigmoid +
↪ kereszt-entrópia: dJ/dz = p - y
    for l in reversed(range(len(W))):
        dW[l] = A[l].T @ delta
        db[l] = delta.sum(axis=0)
        if l > 0:
            delta = (delta @ W[l].T) * (1 - A[l] ** 2)    # tanh'(z) = 1 -
↪ tanh(z)^2
    return dW, db
```

Mielőtt a hálózatot tanítanánk, ellenőrizzük a gradienst! Egy kiválasztott súlyt kis h értékkel megváltoztatva a veszteség központi differenciahányadosa, $(J(\theta + h) - J(\theta - h))/(2h)$, a parciális derivált $O(h^2)$ pontosságú közelítése. Ha a hibavisszaterjesztés helyes, a kettő sok tizedesjegyre megegyezik:

```
rng = np.random.default_rng(0)
W, b = init([2, 16, 16, 1], rng)
A = elore(Xtr[:20], W, b)
dW, _ = vissza(A, ytr[:20], W)
h = 1e-6
W[1][3, 5] += h;      Jp = veszteseg(elore(Xtr[:20], W, b)[-1], ytr[:20])
W[1][3, 5] -= 2 * h;  Jm = veszteseg(elore(Xtr[:20], W, b)[-1], ytr[:20])
W[1][3, 5] += h
print(f"analitikus: {dW[1][3, 5]:.10f}   numerikus: {(Jp - Jm) / (2 *
↪ h):.10f}")
```

```
analitikus: -0.0153392168   numerikus: -0.0153392167
```

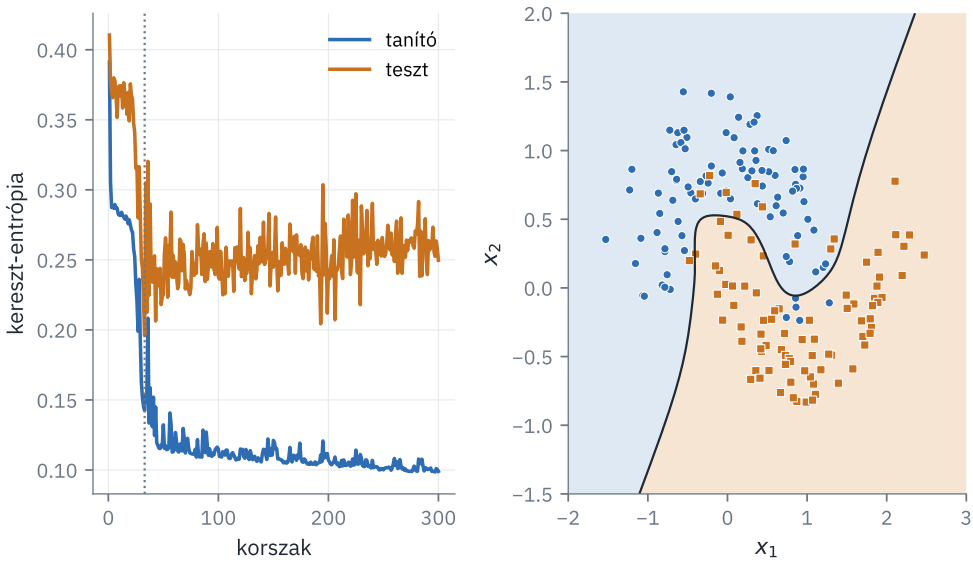
A tanítás 32 elemű mini-batch-ekkel, a (10.5) Adam-szabállyal történik; minden korszak (epoch) végén feljegyezzük a tanító- és a teszthalmazon mért veszteséget.

```
eta, beta1, beta2, eps = 0.01, 0.9, 0.999, 1e-8
params = W + b                                  # a listák ugyanazokra a
↪  tömbökre mutatnak
m_ = [np.zeros_like(p) for p in params]
v_ = [np.zeros_like(p) for p in params]
t, tortenet = 0, []
for epoch in range(300):
    perm = rng.permutation(len(Xtr))
    for k in range(0, len(Xtr), 32):
        idx = perm[k:k + 32]
        dW, db = vissza(elore(Xtr[idx], W, b), ytr[idx], W)
        t += 1
        for i, (p, g) in enumerate(zip(params, dW + db)):
            m_[i] = beta1 * m_[i] + (1 - beta1) * g
            v_[i] = beta2 * v_[i] + (1 - beta2) * g ** 2
            mhat, vhat = m_[i] / (1 - beta1 ** t), v_[i] / (1 - beta2 ** t)
           p -= eta * mhat / (np.sqrt(vhat) + eps)          # helyben frissít
    tortenet.append((veszteseg(elore(Xtr, W, b)[-1], ytr),
↪  veszteseg(elore(Xte, W, b)[-1], yte)))
tortenet = np.array(tortenet)
pontossag = lambda X_, y_: float(np.mean((elore(X_, W, b)[-1] > 0.5) == y_))
print(f"tanító pontosság: {pontossag(Xtr, ytr):.3f}, teszt pontosság:
↪  {pontossag(Xte, yte):.3f}")
print(f"a legkisebb teszt-veszteség: {tortenet[:, 1].min():.3f} a(z)
↪  {tortenet[:, 1].argmin() + 1}. korszakban; "
      f"a végén: {tortenet[-1, 1]:.3f}")
```

```
tanító pontosság: 0.960, teszt pontosság: 0.917
a legkisebb teszt-veszteség: 0.196 a(z) 33. korszakban; a végén: 0.250
```

```
fig, ax = plt.subplots(1, 2, gridspec_kw={"width_ratios": [1, 1.1]})
ax[0].plot(np.arange(1, 301), tortenet[:, 0], color=ab.PALETTE[0],
↪  label="tanító")
ax[0].plot(np.arange(1, 301), tortenet[:, 1], color=ab.PALETTE[1],
↪  label="teszt")
ax[0].axvline(tortenet[:, 1].argmin() + 1, color="#6B7785", ls=":", lw=1)
ax[0].set_xlabel("korszak"); ax[0].set_ylabel("kereszt-entrópia");
↪  ax[0].legend()
g0, g1 = np.meshgrid(np.linspace(-2, 3, 300), np.linspace(-1.5, 2, 240))
P = elore(np.c_[g0.ravel(), g1.ravel()], W, b)[-1].reshape(g0.shape)
ax[1].contourf(g0, g1, P, levels=[0, 0.5, 1], colors=["#DCE7F3",
↪  "#F6E3CF"], alpha=0.9)
ax[1].contour(g0, g1, P, levels=[0.5], colors="#1F2933", linewidths=1)
```

```
for c, (szin, jel) in enumerate([(ab.PALETTE[0], "o"), (ab.PALETTE[1],
↪  "s")]):
    sel = yte[:, 0] == c
    ax[1].scatter(Xte[sel, 0], Xte[sel, 1], s=14, color=szin, marker=jel,
↪  edgecolor="white", lw=0.4)
ax[1].set_xlabel("$x_1$"); ax[1].set_ylabel("$x_2$"); ax[1].grid(False)
fig.tight_layout()
```



10.8. ábra. A NumPy-ban programozott hálózat tanulása. Balra: a tanító- és a teszthalmazon mért veszteség a korszakok függvényében – a teszt-veszteség korán minimumot ér el, majd nő (túltanulás). Jobbra: a 300 korszak után kapott döntési határ a teszthalmaz pontjaival.

A tanítóhalmaz vesztesége folyamatosan csökken, a teszthalmazé viszont néhány tucat korszak után a minimumát elérve nőni kezd: a hálózat a tanítóminták zajos, egymásba nyúló pontjait is igyekszik helyesen osztályozni, és a döntési határ egyre szabálytalanabb lesz. A korai leállítás éppen ezt használja ki: egy validációs halmazon figyeli a veszteséget, és a legjobb állapotban állítja le a tanítást.

10.5.2. Hűtési igény modell: ahol a neurális hálózat előnyös

FUTÓ ESET · PANNON ÜZEM

A reaktor hűtővízigénye fizikailag nemlineáris: a hőmérleg szerint a szükséges hűtővízáram a reakcióhőtől, a reaktor és a hűtővíz hőmérséklet-különbségétől és a hőátadási tényezőtől függ, utóbbi pedig a hűtőköpeny lerakódása miatt két tisztítás között folyamatosan romlik. Egy adatalapú hűtési igény modell két dologra is használható: előre jelzi a várható hűtővízigényt (például a nyári telítődés kockázatát), és a modell maradékából követhető a lerakódás. A tisztítások időpontja az adatokból kiolvasható – ezek a leállások –, így bemenetként az utolsó leállás óta eltelt időt is felhasználhatjuk.

A modelleket a normál üzem órás átlagain, az első félév adatain tanítjuk, és a második félévben értékeljük. Ez szigorú próba: a nyári hónapok melegebb hűtővize és nagyobb terhelése kívül esik a tanítóadatok tartományán.

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import RidgeCV
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor,
↪  HistGradientBoostingRegressor
from sklearn.metrics import r2_score, mean_absolute_error

m = ab.plant_clean()
utolso_leallas = m.ido.where(m.leallas).ffill()
m["nap_leallas_ota"] = ((m.ido - utolso_leallas).dt.total_seconds() /
↪  86400) \
    .fillna((m.ido - m.ido.iloc[0]).dt.total_seconds() / 86400)
m = m[~m.leallas].copy()
m["hiba"] = ab.load_plant().truth["fault"].reindex(m.ido).to_numpy()
bemenet = ["T_reaktor", "T_alapjel", "T_hutoviz", "F_betap", "c_betap",
↪  "T_betap",
           "T_kornyezet", "nap_leallas_ota"]
h = (m[m.hiba == "normál"].set_index("ido")[bemenet + ["F_hutoviz"]]
     .resample("1h").mean().dropna())
tr = h.index < "2025-07-01"
racs = np.logspace(-3, 3, 13)
modellek = {
    "lineáris (ridge)": make_pipeline(StandardScaler(),
    ↪  RidgeCV(alphas=racs)),
    "másodfokú polinom (ridge)": make_pipeline(StandardScaler(),
    ↪  PolynomialFeatures(2),
```

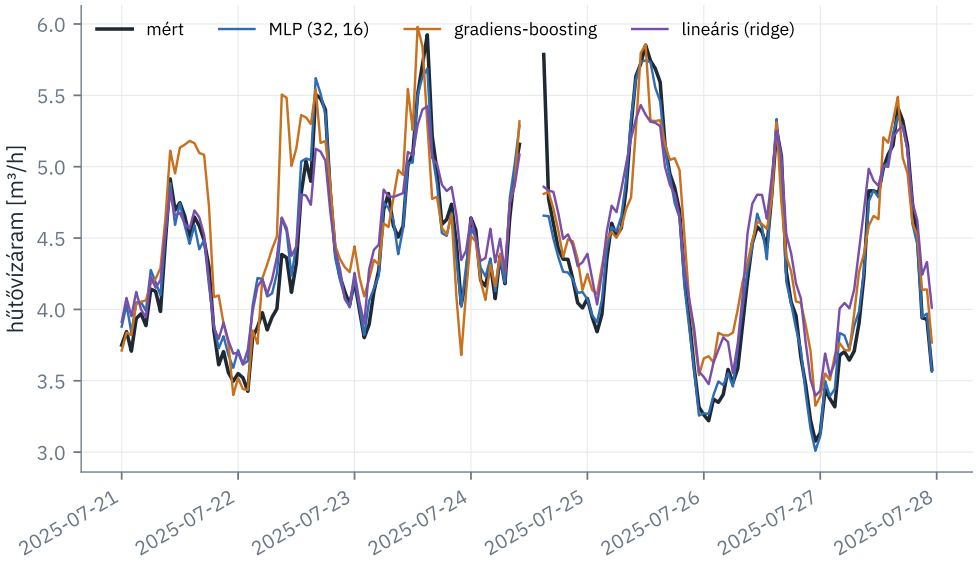
```
                                                StandardScaler(),
    RidgeCV(alphas=racs)),
    "véletlen erdő": RandomForestRegressor(n_estimators=300,
        min_samples_leaf=3, n_jobs=-1, random_state=0),
    "gradiens-boosting": HistGradientBoostingRegressor(random_state=0),
    "MLP (32, 16)": make_pipeline(StandardScaler(),
                                  MLPRegressor(hidden_layer_sizes=(32, 16),
    alpha=1e-3, early_stopping=True,
                                               max_iter=3000, random_state=0)),
}
eredm, becsles = {}, {}
for nev, mod in modellek.items():
    mod.fit(h.loc[tr, bemenet], h.F_hutoviz[tr])
    becsles[nev] = mod.predict(h.loc[~tr, bemenet])
    eredm[nev] = {"R² (2. félév)": r2_score(h.F_hutoviz[~tr],
    becsles[nev]),
                  "MAE [m³/h]": mean_absolute_error(h.F_hutoviz[~tr],
                      becsles[nev])}
# a 2. félév azon órái, amelyekben valamelyik bemenet a tanítóadatok
    tartományán kívül esik
kivul = ((h.loc[~tr, bemenet] > h.loc[tr, bemenet].max()) |
         (h.loc[~tr, bemenet] < h.loc[tr,
    bemenet].min())).any(axis=1).to_numpy()
for nev in modellek:
    hiba = np.abs(h.F_hutoviz[~tr].to_numpy() - becsles[nev])
    eredm[nev]["MAE, tartományon kívül"] = hiba[kivul].mean()
print(f"a tanítási tartományon kívül eső órák száma: {kivul.sum()} /
    {len(kivul)}")
pd.DataFrame(eredm).T.round(3)
```

```
a tanítási tartományon kívül eső órák száma: 58 / 4019
```

	R² (2. félév)	MAE [m³/h]	MAE, tartományon kívül
lineáris (ridge)	0.817	0.533	1.715
másodfokú polinom (ridge)	0.968	0.212	0.598
véletlen erdő	0.830	0.499	1.330
gradiens-boosting	0.894	0.379	0.967
MLP (32, 16)	0.990	0.114	0.276

```
teszt = pd.DataFrame(becsles,
    index=h.index[~tr]).assign(mert=h.F_hutoviz[~tr])
teszt = teszt.loc["2025-07-21":"2025-07-27"].asfreq("1h")       # a hiányzó
    órák NaN-ként szakítják meg a vonalat
fig, ax = plt.subplots()
ax.plot(teszt.index, teszt.mert, color="#1F2933", lw=1.6, label="mért")
for nev, szin in [("MLP (32, 16)", ab.PALETTE[0]), ("gradiens-boosting",
    ab.PALETTE[1]),
```

```
                  ("lineáris (ridge)", ab.PALETTE[2])]:
    ax.plot(teszt.index, teszt[nev], color=szin, lw=1.1, label=nev)
ax.set_ylabel("hűtővízáram [m³/h]"); ax.legend(ncol=4, fontsize=8,
↪  loc="upper left")
fig.autofmt_xdate()
fig.tight_layout()
```



10.9. ábra. A hűtési igény modellek előrejelzése a második félév egy nyári hetén (a hiányzó szakasz egy hibaesemény, amelyet a normál üzem modellezésekor kihagytunk). A lineáris modell a csúcsokat alulbecsli, a gradiens-boosting lépcsős és helyenként nagyot téved, a neurális hálózat szorosan követi a mért értéket.

FUTÓ ESET · PANNON ÜZEM

Ez az a helyzet, amelyben a neurális hálózat a legjobban teljesít: a kapcsolat sima és erősen nemlineáris (a hőátadási tényező és a hőmérséklet-különbség szorzatként hat), a bemenetek száma kicsi, és az értékelési időszak részben kívül esik a tanítóadatok tartományán. A lineáris modell nem tudja leírni a kölcsönhatásokat, a másodfokú polinomos bővítés már sokat javít. A fa alapú modellek egy sima, szorzatszerű kölcsönhatást szakaszonként konstans függvénnyel közelítenek, ami ennyi adatból pontatlan, és a tanítóadatok tartományán kívül konstans értéket adnak: a hibájuk ott többszöröse a tartományon belülinek. A tartományon kívül a neurális hálózat hibája is nő – a sima extrapoláció sem garantáltan helyes, csak a fizikai

kapcsolat simasága miatt itt jó közelítés. A 15. fejezetben az ilyen modellek karbantartási célú felhasználására visszatérünk.

10.5.3. Neurális hálózat a lemorzsolódási feladaton

FUTÓ ESET · PANNONSHOP

Táblázatos, közepes méretű adatokon a helyzet más. Hasonlítsuk össze az MLP-t a 9. fejezet lemorzsolódási modelljeivel ugyanazzal a keresztvalidációval!

```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import FunctionTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import StratifiedKFold, cross_val_score

churn = ab.churn_table(cutoff="2025-07-01", horizon_days=183)
jell = ["R_nap", "F_szamla", "M_ertek", "atl_kosarertek", "atl_kosarmeret",
 ↪  "atl_mennyiseg",
        "elso_vasarlas_nap", "akcios_arany", "arany_kave", "arany_gep",
         ↪  "arany_konyha",
        "arany_dekor", "arany_ajandek", "arany_iroszer"]
X, y = churn[jell].astype(float), churn.lemorzsolodik
elo = make_pipeline(SimpleImputer(strategy="median"),
 ↪  FunctionTransformer(np.log1p), StandardScaler())
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
jeloltek = {
    "logisztikus regresszió": make_pipeline(elo,
     ↪  LogisticRegression(max_iter=2000)),
    "MLP (32), α = 0,01": make_pipeline(elo,
     ↪  MLPClassifier(hidden_layer_sizes=(32,), alpha=0.01,
                                        early_stopping=True, max_iter=2000,
 ↪  random_state=0)),
    "MLP (64, 32), α = 0,1": make_pipeline(elo,
     ↪  MLPClassifier(hidden_layer_sizes=(64, 32), alpha=0.1,
                                             early_stopping=True,
 ↪  max_iter=2000, random_state=0)),
    "véletlen erdő": make_pipeline(SimpleImputer(strategy="median"),
                                   RandomForestClassifier(n_estimators=400,
 ↪  min_samples_leaf=5,
                                                          random_state=0,
 ↪  n_jobs=-1)),
```

```
}
pd.DataFrame({nev: {"AUC": (s := cross_val_score(mod, X, y, cv=cv,
 ↪  scoring="roc_auc")).mean(),
                    "AUC szórás": s.std()} for nev, mod in
                     ↪  jeloltek.items()}).T.round(3)
```

	AUC	AUC szórás
logisztikus regresszió	0.790	0.015
MLP (32), $\alpha = 0{,}01$	0.778	0.024
MLP (64, 32), $\alpha = 0{,}1$	0.796	0.016
véletlen erdő	0.822	0.010

A neurális hálózat itt nem hoz érdemi javulást a logisztikus regresszióhoz képest (a különbség a szóráson belül van), és elmarad a véletlen erdőtől; a kisebb, gyengébben regularizált hálózat eredménye ráadásul a részhalmazok között jobban szóródik. Kétezer vevő és tizennégy jól megtervezett jellemző esetén a hálózat rugalmassága nem kifizetődő – összhangban a fentebb idézett összehasonlító vizsgálatokkal.

10.5.4. Autoenkóder a folyamat felügyeletére

FUTÓ ESET · PANNON ÜZEM

Tanítsunk autoenkódert az első félév normál üzemének standardizált, 10 perces mérésein, és használjuk a rekonstrukciós hibát anomáliapontszámként a második félévben! Összevetésként a lineáris autoenkóderrel egyenértékű PCA-rekonstrukciót is kiszámítjuk, azonos (négydimenziós) kóddal.

```
from sklearn.decomposition import PCA
from sklearn.metrics import roc_auc_score

jelek = ["T_reaktor", "T_alapjel", "F_hutoviz", "T_hutoviz", "P_reaktor",
 ↪  "F_betap", "c_betap",
         "T_betap", "P_kevero", "E_fogyasztas", "T_kornyezet"]
Xp = m[jelek].reset_index(drop=True)
allapot = m.hiba.reset_index(drop=True)
elso = (m.ido < "2025-07-01").to_numpy()
normal_tanito = elso & (allapot == "normál").to_numpy()
sk = StandardScaler().fit(Xp[normal_tanito])
Z = sk.transform(Xp)

pca = PCA(n_components=4).fit(Z[normal_tanito])
hiba_pca = ((Z - pca.inverse_transform(pca.transform(Z))) ** 2).sum(axis=1)
```

```
ae = MLPRegressor(hidden_layer_sizes=(16, 4, 16), activation="tanh",
 ↪  alpha=1e-4,
                   early_stopping=True, max_iter=500,
 ↪  random_state=0).fit(Z[normal_tanito], Z[normal_tanito])
hiba_ae = ((Z - ae.predict(Z)) ** 2).sum(axis=1)

masodik = ~elso
sor = {}
for nev, e in [("PCA (4 komponens)", hiba_pca), ("autoenkóder (16-4-16)",
 ↪  hiba_ae)]:
    sor[nev] = {"összes hiba": roc_auc_score(allapot[masodik] != "normál",
 ↪  e[masodik])}
    for f in ["szenzor_drift", "betap_zavar", "szelep_akadas",
     ↪  "hutoviz_kieses"]:
        sel = masodik & allapot.isin(["normál", f]).to_numpy()
        sor[nev][f] = roc_auc_score(allapot[sel] == f, e[sel])
pd.DataFrame(sor).T.round(3).rename_axis("AUC (2. félév)")
```

AUC (2. félév)	összes hiba	szenzor_drift	betap_zavar	szelep_akadas	hutoviz_kieses
PCA (4 komponens)	0.805	0.816	1.0	0.572	1.0
autoenkóder (16-4-16)	0.782	0.776	1.0	0.579	1.0

FUTÓ ESET · PANNON ÜZEM

A rekonstrukciós hiba a hűtővíz-kiesést és a betáplálási zavart szinte tökéletesen jelzi, a szenzor-driftet részben, a szelepakadást viszont alig: ez utóbbi a változók közötti kapcsolatokat nem rontja el, csak a hűtővízáram ingadozását növeli, amit pillanatnyi értékekből álló bemenet nem „lát”. Tanulságos, hogy a nemlineáris autoenkóder nem jobb a PCA-nál. A normál üzemben a változók közötti kapcsolatok a működési tartományon belül közel lineárisak, így a lineáris altér jól leírja őket; a nemlineáris autoenkóder akkor hasznos, ha a normál működés valóban görbült sokaságon helyezkedik el – például spektrumoknál, rezgésjeleknél, képeknél. A 11. fejezetben az anomália-észlelés módszereit, köztük a PCA-alapú statisztikai folyamatfelügyeletet, részletesen tárgyaljuk.

10.5.5. Konvolúciós hálózat hibadiagnózisra (PyTorch)

A 9. fejezet hibadiagnosztikai modelljében a jellemzőket – a mozgóablakos szórásokat, a névleges értékhez viszonyított eltéréseket, a nyomásmaradékot – szakterületi tudás alapján mi terveztük meg. Egy egydimenziós konvolúciós hálózat a nyers, határás jelablakokból maga tanulhatja meg a hibák időbeli mintázatát. Az

alábbi kód a PyTorch (Paszke és mtsai., 2019) keretrendszert használja; a könyv fordításakor nem fut le, a kapcsolódó Colab-jegyzetfüzetben (amelyben a PyTorch előre telepítve van) futtatható.

```
import torch
import torch.nn as nn
from sklearn.metrics import classification_report

# --- ablakok: 6 óra (36 minta) × 7 jel; címke: az ablak végén fennálló állapot ---
teljes = ab.plant_clean()
teljes["hiba"] = ab.load_plant().truth["fault"].reindex(teljes.ido).to_numpy()
jelek = ["T_reaktor", "T_alapjel", "F_hutoviz", "T_hutoviz", "P_reaktor", "c_betap", "F_betap"]
tanito_idoszak = (teljes.ido < "2025-07-01").to_numpy()
normal = tanito_idoszak & (teljes.hiba == "normál").to_numpy() & ~teljes.leallas.to_numpy()
mu, sd = teljes.loc[normal, jelek].mean(), teljes.loc[normal, jelek].std()
Z = ((teljes[jelek] - mu) / sd).to_numpy(dtype="float32")
L = 36
ervenyes = ~teljes.leallas.to_numpy() & ~np.isnan(Z).any(axis=1)
vegek = [t for t in range(L - 1, len(Z)) if ervenyes[t - L + 1:t + 1].all()]
Xw = np.stack([Z[t - L + 1:t + 1].T for t in vegek])          # (minta, csatorna, idő)
osztalyok = ["normál", "szenzor_drift", "betap_zavar", "szelep_akadas", "hutoviz_kieses"]
yw = np.array([osztalyok.index(teljes.hiba.iloc[t]) for t in vegek])
tr = tanito_idoszak[vegek]

class CNN1D(nn.Module):
    def __init__(self, n_csatorna, n_osztaly):
        super().__init__()
        self.halo = nn.Sequential(
            nn.Conv1d(n_csatorna, 32, kernel_size=5, padding=2), nn.BatchNorm1d(32), nn.ReLU(),
            nn.Conv1d(32, 32, kernel_size=5, padding=2), nn.BatchNorm1d(32), nn.ReLU(),
            nn.MaxPool1d(2),
            nn.Conv1d(32, 64, kernel_size=3, padding=1), nn.BatchNorm1d(64), nn.ReLU(),
            nn.AdaptiveAvgPool1d(1), nn.Flatten(),
            nn.Dropout(0.3), nn.Linear(64, n_osztaly))

    def forward(self, x):
        return self.halo(x)

torch.manual_seed(0)
```

```
modell = CNN1D(len(jelek), len(osztalyok))
gyak = np.bincount(yw[tr], minlength=len(osztalyok))
kriterium = nn.CrossEntropyLoss(weight=torch.tensor(len(yw[tr]) /
 ↪  (len(osztalyok) * gyak), dtype=torch.float32))
opt = torch.optim.AdamW(modell.parameters(), lr=1e-3, weight_decay=1e-4)
Xt, yt = torch.from_numpy(Xw[tr]), torch.from_numpy(yw[tr])
for epoch in range(15):
    modell.train()
    perm = torch.randperm(len(Xt))
    for k in range(0, len(Xt), 256):
        idx = perm[k:k + 256]
        opt.zero_grad()
        loss = kriterium(modell(Xt[idx]), yt[idx])
        loss.backward()                                  # automatikus
 ↪  differenciálás
        opt.step()

modell.eval()
with torch.no_grad():
    pred = modell(torch.from_numpy(Xw[~tr])).argmax(dim=1).numpy()
print(classification_report(yw[~tr], pred, labels=range(len(osztalyok)),
 ↪  target_names=osztalyok, zero_division=0))
```

Egy próbafuttatásunkban (CPU-n fél perc alatt) a konvolúciós hálózat a betáplálási zavart és a szelepakadást a kézzel tervezett jellemzőkre épülő modellhez hasonló pontossággal ismerte fel – a betáplálási zavart úgy is, hogy a névleges koncentrációhoz viszonyítást nem adtuk meg neki, hiszen az ablakban szereplő alapjel a terméket is azonosítja. A hűtővíz-kiesésnél viszont több téves riasztást adott, a lassan kialakuló szenzor-driftet pedig ugyanúgy csak kis részben ismerte fel: hibatípusonként egy-három tanítóeseményből a hálózat sem tud csodát tenni. Az eredmény a véletlen inicializálástól és a tanítás hosszától is függ – ezt érdemes a jegyzetfüzetben több véletlenmaggal kipróbálni.

Interaktív alkalmazás: Neurális hálózat tanítása a böngészőben: rétegek, aktiváció, tanulási ráta

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/neuralis-halo.html

Colab-notebook: 10. fejezet – neurális hálózatok

10.6. Összefoglalás

- A mesterséges neuron a bemenetek súlyozott összegét nemlineáris aktivációs függvényen vezeti át; egyetlen neuron lineáris osztályozó.

- A többrétegű hálózatok egymásba ágyazott nemlineáris leképezések; a nemlinearitás nélkül a mélység semmit sem érne.
- A hibavisszaterjesztés a láncszabállyal, egyetlen előreterjesztés költségével számítja ki az összes paraméter szerinti gradienst; a helyességét numerikus gradiensellenőrzéssel érdemes igazolni.
- A tanítás mini-batch sztochasztikus gradienscsökkentéssel, jellemzően Adam-optimalizálóval történik; a tanulási ráta és az inicializálás döntő.
- A regularizáció eszközei a súlycsökkentés, a korai leállítás, a kiejtés, a batch-normalizálás és az adatbővítés.
- A konvolúciós hálók helyi, eltolás-invariáns mintázatokat, a rekurrens hálók (LSTM, GRU) időbeli függéseket tanulnak; az autoenkóderek felügyelet nélkül tömörítenek, és a rekonstrukciós hiba anomáliapontszámként használható.
- Táblázatos, közepes méretű adatokon a fa alapú együttes modellek rendszerint jobbak; a neurális hálózatok a nyers jelek, a sima nemlineáris kapcsolatok és az extrapoláció esetén előnyösek.

Ellenőrző kérdések

1. Miért nem tudja egyetlen perceptron megtanulni a XOR függvényt?
2. Miért nem érne semmit egy mély hálózat lineáris aktivációs függvényekkel?
3. Mit mond ki és mit nem az univerzális approximációs tétel?
4. Vezesse le a (10.3) rekurziót egy rejtett réteg esetén!
5. Miért hasznos a torzításkorrekció az Adam-optimalizálóban?
6. Mi az elenyésző gradiens problémája, és hogyan enyhíti a ReLU, illetve az LSTM?
7. Milyen kapcsolat van a lineáris autoenkóder és a főkomponens-elemzés között?
8. Mikor választana neurális hálózatot gradiens-boosting helyett, és mikor fordítva?

Feladatok

1. **XOR.** Tanítsa meg a NumPy-hálózatot a XOR függvényre egy, majd két rejtett neuronnal! Hány rejtett neuron szükséges, és mennyire függ az eredmény a véletlen inicializálástól?
2. **Korai leállítás.** Egészítse ki a NumPy-hálózat tanítását validációs halmazzal és korai leállítással (türelem: 20 korszak)! Hasonlítsa össze a teszthalmazon mért pontosságot a 300 korszakos tanításéval!

3. **Aktivációk és inicializálás.** Cserélje a tanh aktivációt ReLU-ra, és vizsgálja meg a tanulást Glorot-, He- és túl nagy szórású ($\sigma = 2$) inicializálással! Ábrázolja a rétegenkénti gradiensnormákat az első korszakokban!
4. **Hűtési igény.** Hagyja el a modell bemenetei közül az utolsó leállás óta eltelt időt, és ábrázolja a neurális modell maradékát a második félévben! Mit mutat a maradék a tisztítások között? (Segítség: a lerakódás.)
5. **LSTM-előrejelző.** A jegyzetfüzet CNN-modellje mintájára készítsen PyTorch-ban LSTM-modellt, amely az elmúlt 12 óra adataiból a következő óra hűtővízáramát jelzi előre! Hasonlítsa össze egy naiv (az utolsó értéket ismétlő) előrejelzéssel!

További olvasmányok

A neurális hálózatok és a mélytanulás alapjait Goodfellow és mtsai. (2016), Bishop & Bishop (2024) és Prince (2023) tankönyvei tárgyalják; Zhang és mtsai. (2023) interaktív, kódra épülő könyve PyTorch- példákkal kíséri az elméletet. Az idősoros mélytanulási modellekről Ismail Fawaz és mtsai. (2019), az egydimenziós konvolúciós hálók mérnöki alkalmazásairól Kiranyaz és mtsai. (2021), a mélytanulás alapú anomália-észlelésről Pang és mtsai. (2021) áttekintése ajánlott.

11. Anomáliák és eltérések észlelése

Miről szól a fejezet?

Az anomália-észlelés a szokatlan, a többségtől eltérő objektumok és események felismerése: csalásgyanús rendelések, meghibásodó berendezések, hibás mérések. A felügyelt osztályozással szemben itt rendszerint nincsenek – vagy alig vannak – címkézett példák, és az anomáliák ritkák, sokfélék, gyakran előre nem ismertek. A fejezet az anomáliák típusaival kezdődik, majd a statisztikai módszereket (robusztus z-érték, Mahalanobis-távolság), a statisztikai folyamatszabályozás egy- és többváltozós eszközeit (Shewhart-, EWMA- és CUSUM-kártya, Hotelling-féle T^2, PCA-alapú T^2 és Q statisztika), a távolság- és sűrűségalapú módszereket (LOF), az izolációs erdőt és az egyosztályos SVM-et tárgyalja. Külön foglalkozunk a küszöbválasztással és az értékelés buktatóival. A gyakorlatban a webáruház csalásgyanús rendeléseit és az üzem hibáit keressük.

A fejezet végére az olvasó képes lesz megkülönböztetni a pontszerű, a kontextuális és a kollektív anomáliákat; kiszámítani és értelmezni a Mahalanobis-távolságot, a T^2 és a Q statisztikát és ezek ellenőrzési határait; alkalmazni és összehasonlítani a LOF-ot, az izolációs erdőt és az egyosztályos SVM-et; és az anomália-észlelőket a feladathoz illő mértékekkel (átlagos pontosság, találati arány a lista elején, téves riasztási arány, észlelési késés) értékelni.

11.1. Mi az anomália?

Hawkins klasszikus meghatározása szerint az anomália (kiugró érték, outlier) olyan megfigyelés, amely annyira eltér a többitől, hogy felmerül a gyanú: más mechanizmus hozta létre (Hawkins, 1980). A definíció kulcsa a *más mechanizmus*: az anomália-észlelés célja nem egyszerűen a ritka értékek megtalálása, hanem azoké, amelyek mögött hiba, csalás vagy új jelenség áll. Ugyanaz a technika a 3. fejezetben az adattisztítás eszköze volt (ott a kiugró értékek zavaró zajt jelentettek); itt maguk az anomáliák az elemzés célpontjai.

Chandola és mtsai. (2009) három típust különböztet meg (11.1. ábra). A **pontszerű** anomália egyetlen, a többitől eltérő megfigyelés. A **kontextuális** anomália

értéke önmagában nem szokatlan, csak az adott környezetben (időpontban, helyen, működési állapotban): 25 °C-os hűtővíz nyáron normális, télen hibára utal. A **kollektív** anomália egymás utáni megfigyelések csoportja, amelyek egyenként normálisak, együtt azonban szokatlan mintázatot alkotnak – például egy szenzor „befagyott”, állandó jele.



11.1. ábra. Az anomáliák három típusa idősorokon. (a) Egyetlen kiugró érték; (b) önmagában normális érték szokatlan környezetben; (c) egyenként normális értékek szokatlan sorozata.

A rendelkezésre álló címkék szerint három feladattípust különböztetünk meg. **Felügyelt** esetben mindkét osztályra vannak címkék – ez a 9. fejezet erősen kiegyensúlyozatlan osztályozási feladata. **Félig felügyelt** esetben csak a normál működésre vannak (megbízható) példák, és az ettől való eltérést keressük; ez a tipikus ipari helyzet, ahol a hibamentes üzem adatai bőven rendelkezésre állnak. **Felügyelet nélküli** esetben semmilyen címke nincs, és azt feltételezzük, hogy az anomáliák ritkák és eltérnek a többségtől. Az alábbi módszerek többsége mindkét utóbbi helyzetben használható; a különbség az, hogy a modellt a tiszta normál adatokon vagy a teljes, szennyezett adathalmazon illesztjük (Aggarwal, 2017; Ruff és mtsai., 2021).

11.2. Statisztikai módszerek

11.2.1. Egyváltozós eset

A legegyszerűbb eljárás a standardizált érték: a $|z_i| = |x_i - \bar{x}|/s$ értéket egy küszöbbel (például 3-mal) vetjük össze. Az átlag és a szórás azonban maguk is érzékenyek a kiugró értékekre: néhány nagy kiugró érték megnöveli a szórást, és így elrejti saját magát (**maszkolás**). Ezért robusztus változatot érdemes használni (Leys és mtsai., 2013):

$$z_i^{\text{rob}} = \frac{x_i - \operatorname{med}(\mathbf{x})}{1{,}4826 \cdot \operatorname{MAD}(\mathbf{x})}, \qquad \operatorname{MAD}(\mathbf{x}) = \operatorname{med}_i \left| x_i - \operatorname{med}(\mathbf{x}) \right|, \tag{11.1}$$

ahol az 1,4826-os szorzó normális eloszlásnál a MAD-ot a szórás torzítatlan becslőjévé teszi. A 3. fejezet Hampel-szűrője ennek mozgóablakos változata.

11.2.2. A Mahalanobis-távolság

Több változó esetén az egyes változók külön-külön normális értékei együtt is lehetnek szokatlanok: a 11.2. ábra A pontja mindkét koordinátája szerint a szokásos tartományban van, de ellentmond a két változó közötti erős pozitív korrelációnak. A korrelációt a **Mahalanobis-távolság** (Mahalanobis, 1936) veszi figyelembe:

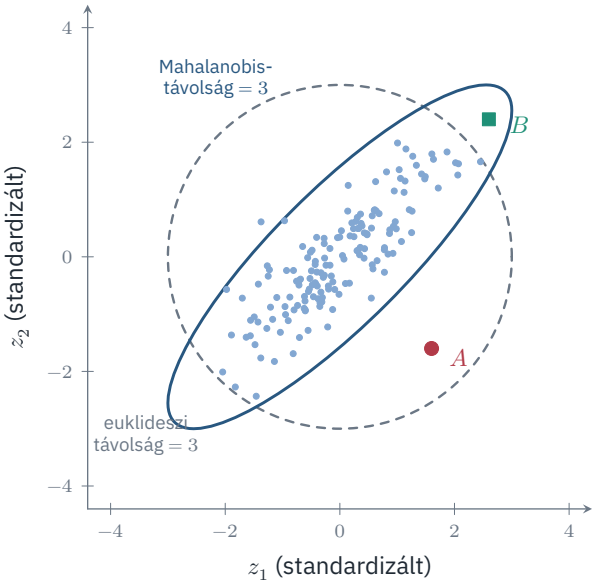
$$d_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}. \tag{11.2}$$

A $\boldsymbol{\Sigma} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$ sajátérték-felbontással (B. függelék) $d_M^2 = \sum_j (\mathbf{v}_j^\top (\mathbf{x}-\boldsymbol{\mu}))^2 / \lambda_j$: a távolságot a főkomponensek irányában mérjük, és mindegyik irányban az ottani szórással skálázzuk. Ezért a kis varianciájú irányokban – amelyeket a változók közötti összefüggések jelölnek ki – már kis eltérés is nagy távolságot ad. Ha az adatok n-dimenziós normális eloszlásúak, $d_M^2 \sim \chi^2_n$ eloszlású, így a küszöb a χ^2_n eloszlás egy felső kvantilise lehet.

A $\boldsymbol{\mu}$ és $\boldsymbol{\Sigma}$ becslése a kiugró értékek miatt itt is torzulhat. A **legkisebb kovariancia-determináns** (minimum covariance determinant, MCD) becslő az adatok azon h elemű részhalmazát keresi, amelynek kovarianciamátrixa a legkisebb determinánsú, és ebből becsüli a helyzetet és a szóródást; erre Rousseeuw & Van Driessen (1999) hatékony algoritmust adott (`sklearn.covariance.MinCovDet`).

11.3. Statisztikai folyamatszabályozás

A **statisztikai folyamatszabályozás** (statistical process control, SPC) a gyártási folyamatok ellenőrzésének klasszikus eszköztára (Montgomery, 2019). Alapgondolata, hogy a folyamat ingadozása két forrásból ered: a mindig jelen lévő, véletlenszerű „közös okokból” és a beavatkozást igénylő „különleges okokból”. Az ellenőrzőkártyák a folyamatjellemzőt a normál működésből becsült ellenőrzési határokkal vetik össze, és riasztanak, ha a folyamat „kikerül a szabályozott állapotból”.



11.2. ábra. Az euklideszi és a Mahalanobis-távolság összehasonlítása erősen korrelált adatokon. Az A pont az euklideszi kör belsejében van, de ellentmond a korrelációnak; a B pont távolabb van a középponttól, de illeszkedik az adatok szerkezetéhez.

11.3.1. Egyváltozós ellenőrzőkártyák

A **Shewhart-kártya** (Shewhart, 1931) minden új megfigyelést külön értékel, és jelez, ha az a $\mu \pm 3\sigma$ tartományon kívül esik. Nagy, hirtelen eltolódásokat gyorsan észlel, a kicsiket – egy-másfél szórásnyit – azonban csak lassan, mert nem „emlékszik” a korábbi megfigyelésekre. Az **EWMA-kártya** (Roberts, 1959) az exponenciálisan súlyozott mozgóátlagot figyeli:

$$z_t = \lambda x_t + (1-\lambda) z_{t-1}, \qquad \text{határ: } \mu \pm L\sigma\sqrt{\frac{\lambda}{2-\lambda}\left[1-(1-\lambda)^{2t}\right]}, \tag{11.3}$$

a **CUSUM-kártya** (Page, 1954) pedig a célértéktől való eltérések kumulált összegét:

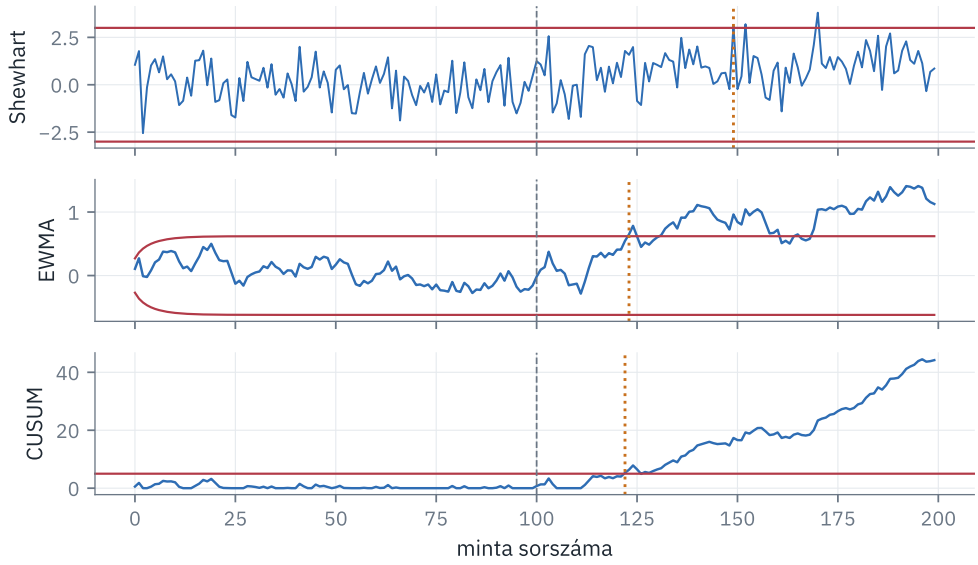
$$S_t^{+} = \max\!\left(0,\; S_{t-1}^{+} + x_t - \mu_0 - k\right), \qquad S_t^{-} = \max\!\left(0,\; S_{t-1}^{-} - x_t + \mu_0 - k\right), \tag{11.4}$$

és jelez, ha S_t^{+} vagy S_t^{-} meghaladja a h döntési határt. A k referenciaérték tipikusan az észlelni kívánt eltolódás fele (szórásegységben 0,5), h pedig 4–5. A két utóbbi kártya a kis, tartós eltolódásokat sokkal gyorsabban észleli, mint a Shewhart-kártya (11.3. ábra).

```
rng = np.random.default_rng(6)
x = rng.normal(0, 1, 200)
x[100:] += 0.75                                   # tartós eltolódás
lam, L = 0.1, 2.7
z = np.zeros_like(x); S = np.zeros_like(x)
for t in range(len(x)):
    z[t] = lam * x[t] + (1 - lam) * (z[t - 1] if t else 0.0)
    S[t] = max(0.0, (S[t - 1] if t else 0.0) + x[t] - 0.5)
hatar = L * np.sqrt(lam / (2 - lam) * (1 - (1 - lam) ** (2 * np.arange(1,
 ↪  201))))
def elso(riaszt, t0=100):                         # első riasztás a váltás
 ↪  után
    utana = np.flatnonzero(riaszt[t0:])
    return t0 + int(utana[0]) if len(utana) else None
jelzes = {"Shewhart": np.abs(x) > 3, "EWMA": np.abs(z) > hatar, "CUSUM": S
 ↪  > 5}
riasztasok = {nev: elso(r) for nev, r in jelzes.items()}
fig, ax = plt.subplots(3, 1, sharex=True)
ax[0].plot(x, color=ab.PALETTE[0], lw=0.9); ax[0].axhline(3,
 ↪  color=ab.PALETTE[4], lw=1); ax[0].axhline(-3, color=ab.PALETTE[4],
 ↪  lw=1)
```

```
ax[1].plot(z, color=ab.PALETTE[0], lw=1.1); ax[1].plot(hatar,
 ↪  color=ab.PALETTE[4], lw=1); ax[1].plot(-hatar, color=ab.PALETTE[4],
 ↪  lw=1)
ax[2].plot(S, color=ab.PALETTE[0], lw=1.1); ax[2].axhline(5,
 ↪  color=ab.PALETTE[4], lw=1)
for a, (nev, t) in zip(ax, riasztasok.items()):
    a.axvline(100, color="#6B7785", ls="--", lw=0.8)
    if t is not None:
        a.axvline(t, color=ab.PALETTE[1], ls=":", lw=1.3)
    a.set_ylabel(nev)
ax[2].set_xlabel("minta sorszáma")
fig.tight_layout()
print("észlelési késés [minta]:", {nev: (None if t is None else t - 100)
 ↪  for nev, t in riasztasok.items()},
      "| téves riasztás a váltás előtt:", {nev: int(r[:100].sum()) for nev,
       ↪  r in jelzes.items()})
```

```
észlelési késés [minta]: {'Shewhart': 49, 'EWMA': 23, 'CUSUM': 22} | téves
 ↪  riasztás a váltás előtt: {'Shewhart': 0, 'EWMA': 0, 'CUSUM': 0}
```



11.3. ábra. Egyváltozós ellenőrzőkártyák egy 0,75 szórásnyi tartós eltolódás (a 100. mintától, szaggatott vonal) észlelésére független, normális eloszlású adatokon. A függőleges pontozott vonalak az első riasztást jelölik.

Mindhárom kártya független megfigyeléseket feltételez. A 10 perces vagy órás folyamatadatok azonban erősen autokorreláltak, és ilyenkor az EWMA és különö-

sen a CUSUM az elméleti határokkal sok téves riasztást ad, mert a pozitívan autokorrelált zaj hosszú, egy irányba tartó kilengései eltolódásnak látszanak. Ilyenkor vagy egy idősormodell (például AR-modell, 12. fejezet) maradékára alkalmazzuk a kártyát, vagy a határokat a normál üzem adataiból, empirikusan határozzuk meg.

11.3.2. Többváltozós folyamatfelügyelet: T^2 és Q

Sok, egymással korreláló változó esetén a változónkénti kártyák egyrészt túl sok riasztást adnak (húsz független kártyánál a téves riasztás valószínűsége hússzoros), másrészt nem veszik észre a korrelációs szerkezet megsérülését (a 11.2. ábra A pontja). A **Hotelling-féle T^2 statisztika** (Hotelling, 1947) a Mahalanobis-távolság négyzete a normál üzemből becsült $\bar{\mathbf{x}}$ átlaggal és $\mathbf{S}$ kovarianciával; egy új megfigyelésre vonatkozó ellenőrzési határa p változó és N tanítómegfigyelés esetén

$$T^2_{\alpha} = \frac{p(N-1)(N+1)}{N(N-p)}\, F_{\alpha}(p,\, N-p). \tag{11.5}$$

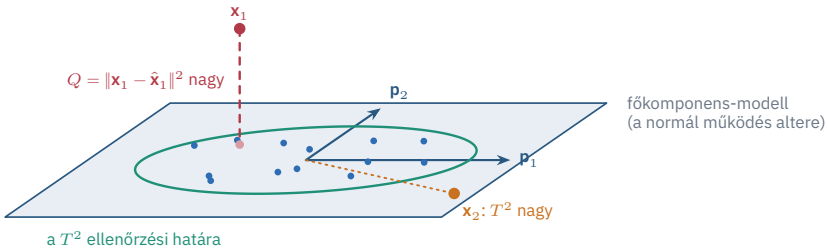
Ha a változók erősen korrelálnak, $\mathbf{S}$ rosszul kondicionált, és T^2 a kis sajátértékű irányok zajára is érzékeny. A **PCA-alapú folyamatfelügyelet** (Jackson & Mudholkar, 1979; MacGregor & Kourti, 1995; Qin, 2012) ezért a standardizált adatok terét két részre bontja (11.4. ábra): az első q főkomponens kifeszítette **modelltérre**, amely a normál működés szisztematikus változásait írja le, és a **maradéktérre**. A modelltérben a T^2, a maradéktérben a Q statisztika (más néven SPE, négyzetes előrejelzési hiba) méri az eltérést:

$$T^2 = \sum_{a=1}^{q} \frac{t_a^2}{\lambda_a}, \qquad Q = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 = \left\|(\mathbf{I} - \mathbf{P}\mathbf{P}^{\top})\mathbf{x}\right\|^2, \tag{11.6}$$

ahol $\mathbf{P}$ a főkomponens-irányok $n \times q$-s mátrixa, $t_a = \mathbf{p}_a^{\top}\mathbf{x}$ a főkomponens-értékek, λ_a a sajátértékek. A T^2 statisztika azt jelzi, ha a folyamat a szokásos összefüggések *mentén* szokatlanul messzire kerül (például szokatlanul nagy terhelés), a Q azt, ha a változók közötti *összefüggések* sérülnek (például egy szenzor hibája vagy egy új zavar). A Q ellenőrzési határát Jackson & Mudholkar (1979) adta meg:

$$Q_\alpha = \theta_1 \left[\frac{c_\alpha \sqrt{2\theta_2 h_0^2}}{\theta_1} + 1 + \frac{\theta_2 h_0 (h_0 - 1)}{\theta_1^2} \right]^{1/h_0}, \qquad \theta_i = \sum_{j=q+1}^{n} \lambda_j^i, \quad h_0 = 1 - \frac{2\theta_1 \theta_3}{3\theta_2^2} \tag{11.7}$$

ahol c_α a standard normális eloszlás $1-\alpha$ kvantilise. Riasztás után a **hozzájárulási ábra** segít a hiba helyének megtalálásában: a Q statisztika változónkénti tagjai, $r_j^2 = (x_j - \hat{x}_j)^2$, megmutatják, mely változók sértik leginkább a megszokott összefüggéseket. A hozzájárulás azonban nem diagnózis: a hiba gyakran olyan változókban „látszik”, amelyek csak következményként térnek el.



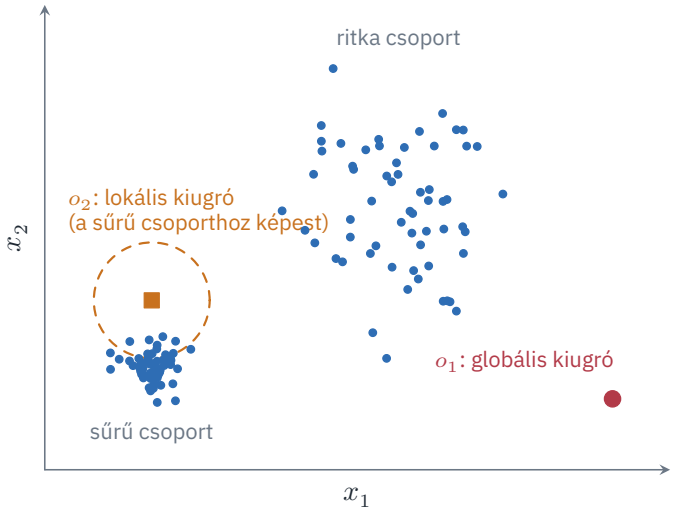
11.4. ábra. A PCA-alapú folyamatfelügyelet geometriája. A normál működés adatai a főkomponens-sík közelében helyezkednek el. Az $\mathbf{x}_1$ megfigyelés messze van a síktól (nagy Q: sérült összefüggések), az $\mathbf{x}_2$ a síkban, de a szokásos tartományon kívül (nagy T^2).

Az algoritmus összefoglalása: PCA-alapú folyamatfelügyelet

1. Gyűjtsünk adatokat a normál (hibamentes) üzemből; standardizáljuk a változókat.
2. Illesszünk PCA-modellt; válasszuk meg a megtartott komponensek q számát (például a magyarázott variancia vagy keresztvalidáció alapján).
3. Számítsuk ki a T^2 és Q ellenőrzési határait az (11.5)–(11.7) képletekkel vagy a normál adatok empirikus kvantiliseiből.
4. Minden új megfigyelésre számítsuk ki T^2-et és Q-t; riasszunk, ha valamelyik (adott számú egymást követő mintán át) meghaladja a határát.
5. Riasztás esetén a hozzájárulási ábrával keressük meg az érintett változókat.

11.4. Távolság- és sűrűségalapú módszerek

A távolságalapú módszerek egy objektumot annál inkább anomáliának tekintenek, minél távolabb vannak tőle a legközelebbi szomszédai; a legegyszerűbb pontszám a k-adik legközelebbi szomszéd távolsága. Ez azonban globális mérce: eltérő sűrűségű csoportok esetén a ritka csoport normál pontjai is nagy pontszámot kapnak, egy sűrű csoport közvetlen szomszédságában lévő, de ahhoz képest szokatlanul távoli pont viszont kicsit (11.5. ábra, o_2 pont).



11.5. ábra. Globális és lokális kiugró érték. Az o_1 minden mérce szerint kiugró; az o_2 a ritka csoport pontjaihoz mérve nincs messze a szomszédaitól, a közeli sűrű csoporthoz képest azonban szokatlan – ezt a lokális sűrűséget összehasonlító LOF ismeri fel.

A **lokális kiugrási tényező** (local outlier factor, LOF) (Breunig és mtsai., 2000) egy pont lokális sűrűségét a szomszédaiéhoz viszonyítja. Jelölje $N_k(p)$ a p pont k legközelebbi szomszédjának halmazát, $d_k(o)$ az o pont k-adik szomszédjának távolságát! Az **elérhetőségi távolság** $\mathrm{reach}_k(p,o) = \max\{d_k(o),\, d(p,o)\}$ – ez a nagyon közeli pontok távolságát kisimítja –, a **lokális elérhetőségi sűrűség** és a LOF pedig

$$\mathrm{lrd}_k(p) = \left(\frac{1}{|N_k(p)|} \sum_{o \in N_k(p)} \mathrm{reach}_k(p,o) \right)^{-1}, \qquad \mathrm{LOF}_k(p) = \frac{1}{|N_k(p)|} \sum_{o \in N_k(p)} \frac{\mathrm{lrd}_k(o)}{\mathrm{lrd}_k(p)}. \tag{11.8}$$

A LOF értéke 1 körüli, ha a pont sűrűsége a szomszédaiéhoz hasonló, és jóval 1 feletti, ha a pont ritkább környezetben van, mint a szomszédai. A módszer a k megválasztására érzékeny (tipikusan 10–50), és nagy adathalmazon a szomszédkeresés költséges.

11.5. Izolációs erdő

Az **izolációs erdő** (isolation forest) (Liu és mtsai., 2008) nem távolságot vagy sűrűséget becsül, hanem azt használja ki, hogy az anomáliák *könnyen elszigetelhetők*. Egy izolációs fa az adatokat véletlenszerűen választott változó mentén, véletlenszerűen választott küszöbbel vágja ketté, rekurzívan, amíg minden pont külön levélbe nem kerül. A ritka, a többitől távoli pontokat már néhány vágás elszigeteli, a sűrű tartományok belsejében lévő pontokhoz sok vágás kell (11.6. ábra). Az anomáliapontszám a sok fán átlagolt $E[h(\mathbf{x})]$ úthosszból számolható:

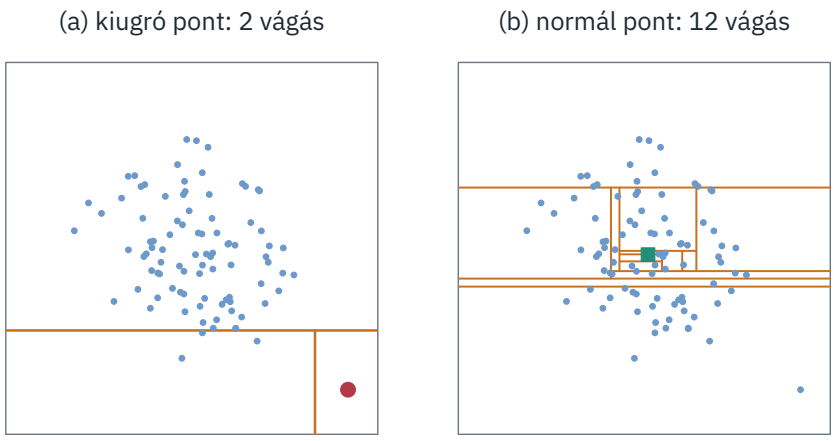
$$s(\mathbf{x},\psi) = 2^{-E[h(\mathbf{x})]/c(\psi)}, \qquad c(\psi) = 2H(\psi-1) - \frac{2(\psi-1)}{\psi}, \quad H(i) \approx \ln i + 0{,}5772, \tag{11.9}$$

ahol ψ a fák építéséhez használt részminta mérete, $c(\psi)$ pedig a sikertelen keresés átlagos úthossza egy ψ elemű bináris keresőfában (normalizáló tényező). Az 1-hez közeli pontszám anomáliát, a 0,5 alatti normál pontot jelez. Az izolációs erdő gyors (a fák kis, például 256 elemű részmintákon épülnek), jól skálázódik nagy dimenzióra és sok adatra, és kevés hangolást igényel, ezért a gyakorlatban az egyik legnépszerűbb felügyelet nélküli módszer; összehasonlító vizsgálatokban rendszerint az élmezőnyben van (Goldstein & Uchida, 2016; S. Han és mtsai., 2022). Gyengesége, hogy a tengelyekkel párhuzamos vágások miatt a változók közötti korrelációs szerkezet megsérülését (a 11.2. ábra A pontját) kevésbé érzékeli.

11.6. Egyosztályos SVM és rekonstrukciós módszerek

Az **egyosztályos SVM** (Schölkopf és mtsai., 2001) a normál adatokat egy kernel által meghatározott jellemzőtérben a lehető legnagyobb margóval választja el az origótól:

$$\min_{\mathbf{w},\boldsymbol{\xi},\rho} \ \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu N}\sum_{k=1}^{N}\xi_k - \rho \quad \text{feltéve, hogy} \quad \mathbf{w}^\top \phi(\mathbf{x}_k) \geq \rho - \xi_k, \quad \xi_k \geq 0. \tag{11.10}$$



11.6. ábra. Az izolációs erdő alapgondolata. Véletlen, tengelyekkel párhuzamos vágásokkal a kiugró pont már két vágás után elszigetelődik, a sűrű tartomány közepén lévő pont elszigeteléséhez sok vágás kell.

A $\nu \in (0, 1]$ paraméter felső korlát a tanítóadatok kiugrónak minősített hányadára és alsó korlát a támaszvektorok hányadára. Radiális kernellel a döntési határ a normál adatok köré simuló, tetszőleges alakú tartomány. A módszer érzékeny a γ kernelparaméterre és a változók skálázására, és nagy adathalmazon lassú.

A **rekonstrukciós módszerek** – a PCA Q statisztikája és a 10. fejezet autoenkódere – egy tömörítő modellt tanítanak a normál adatokon, és a rekonstrukciós hibát használják pontszámként. Ezek különösen akkor hasznosak, ha az anomália a változók közötti összefüggések megsérülésében nyilvánul meg.

11.7. Küszöbválasztás és értékelés

Az anomália-észlelők pontszámot adnak; a döntéshez küszöb kell. A küszöb meghatározható statisztikai eloszlásból (a χ^2, F vagy a (11.7) határok), a normál adatok empirikus kvantiliséből, vagy az anomáliák várható arányából (a `scikit-learn` `contamination` paramétere). A küszöb a téves riasztások és a kihagyott anomáliák közötti kompromisszumot rögzíti, ezért a végső választás – akárcsak a 7. fejezet költségalapú küszöbénél – üzleti vagy üzemeltetési döntés: mennyi téves riasztást képes a csalásfelderítő csoport vagy a műszakvezető naponta kivizsgálni?

Az értékelés több buktatót rejt (Campos és mtsai., 2016):

- **Rendkívül kiegyensúlyozatlan osztályok.** Ha a megfigyelések 0,1%-a anomália, az ROC-görbe alatti terület félrevezetően magas lehet: a negatívok óriási száma miatt kis téves pozitív *arány* is sok téves riasztást jelent. Az **átlagos pontosság** (AP, a PR-görbe alatti terület) és a **találati arány a lista elején** (precision@k: a k legnagyobb pontszámú objektum hányad része valódi anomália) a gyakorlati használathoz közelebb álló mérce, hiszen az elemzők a lista elejét vizsgálják ki.
- **Események és pontok.** Idősoroknál egy hibaesemény sok egymást követő megfigyelésből áll; az üzemeltető szempontjából az számít, hogy az eseményt észleltük-e, és mekkora **késéssel**, valamint hogy naponta hány téves riasztás érkezik.
- **A címkék hiánya és torzítottsága.** A „valódi” anomáliák listája ritkán teljes: a korábban fel nem ismert csalások a negatívok között vannak. Az értékelés ezért rendszerint alsó becslést ad.
- **Ritka nem azonos az érdekessel.** Egy felügyelet nélküli módszer a statisztikailag ritka objektumokat találja meg, amelyek nem feltétlenül a keresettek – a jellemzők megválasztása dönti el, milyen eltéréseket „lát” a módszer.

11.8. Gyakorlat: csalásgyanús rendelések és üzemzavarok

11.8.1. Csalásgyanús rendelések a webáruházban

FUTÓ ESET · PANNONSHOP

A PannonShop adatai között 25 olyan rendelés rejtőzik, amely mögött – a szimulációban – csalás áll; az elemző ezt nem tudja, csak azt, hogy a csalásfelderítő csoport naponta néhány rendelést tud kézzel ellenőrizni. Készítsünk rendelésszintű jellemzőket, és rangsoroljuk a rendeléseket felügyelet nélküli módszerekkel! Két jellemzőkészletet hasonlítunk össze: a csak a kosár *tartalmát* leíró jellemzőket (érték, tételszám, mennyiség, átlagár), illetve ezeket kiegészítve a rendelés *körülményeivel* (napszak, a vevő első rendelése-e, a szállítási régiók száma a rendelésen belül).

```
from sklearn.preprocessing import StandardScaler
from sklearn.covariance import EmpiricalCovariance
from sklearn.neighbors import LocalOutlierFactor
from sklearn.ensemble import IsolationForest
from sklearn.svm import OneClassSVM
from sklearn.metrics import average_precision_score, roc_auc_score
```

```
tx = ab.shop_clean()
tx["ora"] = tx.InvoiceDate.dt.hour
rend = (tx.groupby("Invoice")
          .agg(ertek=("Ertek", "sum"), tetel=("StockCode", "nunique"),
↪  menny=("Quantity", "sum"),
               atl_ar=("Price", "mean"), ora=("ora", "first"),
↪  vevo=("CustomerID", "first"),
               regiok=("Regio", "nunique"), datum=("InvoiceDate", "first"))
          .sort_values("datum"))
rend["vevo_sorszam"] = rend.groupby("vevo").cumcount() + 1
rend.loc[rend.vevo.isna(), "vevo_sorszam"] = 1                    #
↪  vendégvásárló
jellemzok = pd.DataFrame({
    "log_ertek": np.log1p(rend.ertek), "tetel": rend.tetel, "log_menny":
    ↪  np.log1p(rend.menny),
    "log_atl_ar": np.log(rend.atl_ar),
    "ora_sin": np.sin(2 * np.pi * rend.ora / 24), "ora_cos": np.cos(2 *
    ↪  np.pi * rend.ora / 24),
    "elso_rendeles": (rend.vevo_sorszam == 1).astype(int), "regiok":
    ↪  rend.regiok})
csalas =
↪  rend.index.isin(ab.load_shop().truth["fraud_invoices"]).astype(int)
↪  # csak az értékeléshez!
print(f"{len(rend)} rendelés, ebből {csalas.sum()} csalás
↪  ({csalas.mean():.2%})")

def ertekel(pontszam, k=25):
    elso_k = np.argsort(-pontszam)[:k]
    return {"AP": average_precision_score(csalas, pontszam), "AUC":
    ↪  roc_auc_score(csalas, pontszam),
            f"P@{k}": csalas[elso_k].mean()}

keszletek = {"tartalom": ["log_ertek", "tetel", "log_menny", "log_atl_ar"],
             "tartalom + körülmények": list(jellemzok.columns)}
eredm = {}
for knev, oszl in keszletek.items():
    Z = StandardScaler().fit_transform(jellemzok[oszl])
    pontok = {
        "max |z|": np.abs(Z).max(axis=1),
        "Mahalanobis": EmpiricalCovariance().fit(Z).mahalanobis(Z),
        "LOF (k = 35)": -LocalOutlierFactor(n_neighbors=35
        ↪  ).fit(Z).negative_outlier_factor_,
        "izolációs erdő": -IsolationForest(n_estimators=500,
        ↪  random_state=0).fit(Z).score_samples(Z),
        "egyosztályos SVM": -OneClassSVM(nu=0.01,
        ↪  gamma="scale").fit(Z).decision_function(Z),
    }
```

```
    for mnev, s in pontok.items():
        eredm[(knev, mnev)] = ertekel(s)
pd.DataFrame(eredm).T.round(3)
```

```
15295 rendelés, ebből 25 csalás (0.16%)
```

		AP	AUC	P@25
tartalom	max |z|	0.013	0.925	0.00
	Mahalanobis	0.124	0.991	0.12
	LOF (k = 35)	0.051	0.983	0.00
	izolációs erdő	0.082	0.988	0.00
	egyosztályos SVM	0.066	0.894	0.04
tartalom + körülmények	max |z|	0.811	0.994	0.80
	Mahalanobis	0.959	1.000	0.84
	LOF (k = 35)	0.970	1.000	0.88
	izolációs erdő	0.787	0.999	0.68
	egyosztályos SVM	0.331	0.991	0.24

Az eredmény két fontos tanulsággal szolgál. Egyrészt a ROC-görbe alatti terület szinte minden esetben 0,9 fölött van, mégis a csak a kosár tartalmát leíró jellemzőkkel az első 25 rendelés között alig van csalás: az AP és a precision@25 sokkal hívebben tükrözi, mennyire használható a rangsor. Másrészt a jellemzők megválasztása sokkal fontosabb, mint a módszeré. A kosár tartalma alapján a legszokatlanabb rendelések nem a csalások, hanem a viszonteladók nagy tételei – ezek valóban ritkák, de teljesen legitimek:

```
Z = StandardScaler().fit_transform(jellemzok[keszletek["tartalom"]])
s_tart = -IsolationForest(n_estimators=500,
 ↪  random_state=0).fit(Z).score_samples(Z)
szegmens = rend.vevo.map(ab.load_shop().truth["segment"]).fillna("vendég /
 ↪  új vevő")
szegmens[csalas == 1] = "CSALÁS"
pd.crosstab(szegmens.iloc[np.argsort(-s_tart)[:50]],
 ↪  "db").rename_axis(None).rename(columns={"db": "az első 50 között"})
```

col_0	az első 50 között
CSALÁS	7
alkalmi	2
törzsvásárló	1
vendég / új vevő	3
viszonteladó	37

FUTÓ ESET · PANNONSHOP

A körülményeket leíró jellemzőkkel – éjszakai időpont, új vevő, egy rendelésen belül eltérő szállítási régiók – a Mahalanobis-távolság és a LOF átlagos

pontossága 0,95 fölötti, és az első 25 rendelés közel 90%-a csalás; az egyosztályos SVM itt is a leggyengébb, mert érzékeny a kernelparaméterre. A valóságban persze a csalók alkalmazkodnak, és a mintázat változik; ezért a gyakorlatban a felügyelet nélküli rangsort az elemzők visszajelzései alapján fokozatosan felügyelt modellé fejlesztik, a szabályalapú szűrőkkel és a hálózati (közös címek, kártyák, eszközök) elemzéssel kombinálva (Bolton & Hand, 2002). Fontos az is, hogy a rangsor ne automatikusan utasítson el rendeléseket: a kiszűrt esetek emberi felülvizsgálata az adatvédelmi és a fogyasztóvédelmi elvárásoknak is megfelel (1. fejezet).

11.8.2. Az üzem többváltozós felügyelete

FUTÓ ESET · PANNON ÜZEM

Építsünk PCA-alapú felügyeleti rendszert a reaktorra! A modellt az első félév normál üzemének 10 perces adatain illesztjük; a pillanatnyi mért értékek mellé a 10. fejezet tanulsága alapján a hűtővízáram egyórás szórását is felvesszük, hogy a szelepakadás oszcillációja is látható legyen. A második félévben értékeljük a riasztásokat.

```
from scipy import stats
from sklearn.decomposition import PCA

P = ab.load_plant()
m = ab.plant_clean()
m = m[~m.leallas].copy()
m["hiba"] = P.truth["fault"].reindex(m.ido).to_numpy()
mi = m.set_index("ido")
jelek = ["T_reaktor", "T_alapjel", "F_hutoviz", "T_hutoviz", "P_reaktor",
 ↪  "F_betap", "c_betap",
         "T_betap", "P_kevero", "E_fogyasztas", "T_kornyezet"]
X = mi[jelek].copy()
X["F_hut_szoras_1h"] = mi.F_hutoviz.rolling("60min").std()
X = X.dropna()
allapot = mi.loc[X.index, "hiba"].to_numpy()
tanito = X.index < "2025-07-01"
normal_tanito = tanito & (allapot == "normál")

sk = StandardScaler().fit(X[normal_tanito])
Z = sk.transform(X)
pca = PCA().fit(Z[normal_tanito])
print("kumulált magyarázott variancia:",
 ↪  np.round(np.cumsum(pca.explained_variance_ratio_)[:7], 3))
```

```
kumulált magyarázott variancia: [0.453 0.724 0.87  0.934 0.974 0.984 0.99 ]
```

Öt főkomponenst tartunk meg, amelyek a normál üzem varianciájának mintegy 97%-át magyarázzák. Számítsuk ki a statisztikákat és az elméleti ellenőrzési határokat 99%-os szinten, és nézzük meg, hány riasztást adnak a *tanítóadatokon*!

```
q, alfa = 5, 0.99
pca = PCA(n_components=q).fit(Z[normal_tanito])
Tq = pca.transform(Z)
T2 = (Tq ** 2 / pca.explained_variance_).sum(axis=1)
R = Z - pca.inverse_transform(Tq)
Q = (R ** 2).sum(axis=1)

N, p = normal_tanito.sum(), Z.shape[1]
T2_hatar = q * (N - 1) * (N + 1) / (N * (N - q)) * stats.f.ppf(alfa, q, N -
↪  q)
lam = np.linalg.eigvalsh(np.cov(Z[normal_tanito].T))[::-1]
th1, th2, th3 = [(lam[q:] ** i).sum() for i in (1, 2, 3)]
h0 = 1 - 2 * th1 * th3 / (3 * th2 ** 2)
Q_hatar = th1 * (stats.norm.ppf(alfa) * np.sqrt(2 * th2 * h0 ** 2) / th1 + 1
                 + th2 * h0 * (h0 - 1) / th1 ** 2) ** (1 / h0)
pd.DataFrame({"elméleti határ": [T2_hatar, Q_hatar],
              "a normál tanítóadatok 99%-os kvantilise":
               ↪  [np.quantile(T2[normal_tanito], 0.99),
               ↪  np.quantile(Q[normal_tanito], 0.99)],
              "riasztási arány a tanítóadatokon": [(T2[normal_tanito] >
               ↪  T2_hatar).mean(), (Q[normal_tanito] > Q_hatar).mean()]},
             index=["T²", "Q"]).round(3)
```

	elméleti határ	a normál tanítóadatok 99%-os kvantilise	riasztási arány a tanítóadatokon
T²	15.093	25.169	0.03
Q	1.092	1.089	0.01

FUTÓ ESET · PANNON ÜZEM

A Q elméleti határa jól egyezik a tapasztalati kvantilissel, a T^2 határa viszont jóval alacsonyabb annál, és a normál üzemben is a vártnál többször riaszt. Ennek oka, hogy a T^2 eloszlása a három termék eltérő alapjelei és betáplálási koncentrációi miatt nem normális, hanem többmóduszú. A két lehetséges megoldás: állapotfüggő (termékenkénti) modellek – ezekre a 15. fejezetben visszatérünk – vagy empirikus határok. Itt az utóbbit választjuk (a normál tanítóadatok 99,5%-os kvantilise), és a téves riasztások további csökkentésére csak akkor riasztunk, ha három egymást követő minta (30 perc) a határ felett van.

```
T2_emp, Q_emp = np.quantile(T2[normal_tanito], 0.995),
↪  np.quantile(Q[normal_tanito], 0.995)
nyers = pd.Series((T2 > T2_emp) | (Q > Q_emp), index=X.index)
riaszt = nyers.rolling(3).sum().eq(3)
teszt = ~tanito
sor = {}
for nev, r in [("egyedi minták", nyers), ("3 egymást követő minta",
↪  riaszt)]:
    sor[nev] = {"téves riasztási arány (normál)": r[teszt & (allapot ==
↪  "normál")].mean()}
    for f in ["szenzor_drift", "betap_zavar", "szelep_akadas",
    ↪  "hutoviz_kieses"]:
        sor[nev][f] = r[teszt & (allapot == f)].mean()
pd.DataFrame(sor).T.round(3).rename_axis("2. félév")
```

2. félév	téves riasztási arány (normál)	szenzor_drift	betap_zavar	szelep_akadas	hutoviz_kieses
egyedi minták	0.016	0.220	1.000	0.995	1.000
3 egymást követő minta	0.005	0.093	0.993	0.991	0.933

```
esemenyek = P.truth["faults"]
esemenyek = esemenyek[esemenyek.kezdet >= "2025-07-01"].copy()
def keses(e):
    szakasz = riaszt[e.kezdet:e.veg]
    return (szakasz[szakasz].index.min() - e.kezdet).total_seconds() / 3600
    ↪  if szakasz.any() else np.nan
esemenyek["késés [óra]"] = esemenyek.apply(keses, axis=1).round(1)
esemenyek["időtartam [óra]"] = ((esemenyek.veg -
↪  esemenyek.kezdet).dt.total_seconds() / 3600 + 1 / 6).round(0)
esemenyek[["hiba", "kezdet", "időtartam [óra]", "késés
↪  [óra]"]].reset_index(drop=True)
```

11.1. táblázat. A második félév hibaeseményeinek észlelési késése a perzisztenciaszabállyal (az első riasztás a hiba kezdete után).

	hiba	kezdet	időtartam [óra]	késés [óra]
0	szenzor_drift	2025-07-10 06:50:00	120.0	10.0
1	hutoviz_kieses	2025-07-11 02:20:00	5.0	0.3
2	hutoviz_kieses	2025-07-24 10:20:00	5.0	0.3
3	hutoviz_kieses	2025-08-05 06:30:00	5.0	0.3
4	hutoviz_kieses	2025-08-18 15:30:00	5.0	0.3
5	betap_zavar	2025-08-19 06:30:00	48.0	0.3
6	hutoviz_kieses	2025-09-01 16:20:00	5.0	0.3
7	szelep_akadas	2025-09-23 17:00:00	72.0	0.7
8	szenzor_drift	2025-10-28 03:30:00	120.0	50.0

```
szakasz = (X.index >= "2025-07-01") & (X.index < "2025-09-30")
ido = X.index[szakasz]
fig, ax = plt.subplots(2, 1, sharex=True)
for a, stat, hatar, nev in [(ax[0], T2, T2_emp, "$T^2$"), (ax[1], Q, Q_emp,
↪  "$Q$")]:
    a.plot(ido, stat[szakasz], color=ab.PALETTE[0], lw=0.5)
    a.axhline(hatar, color=ab.PALETTE[4], lw=1.2)
    a.set_yscale("log"); a.set_ylabel(nev)
    for _, e in esemenyek[esemenyek.kezdet < "2025-09-30"].iterrows():
        szin = {"szenzor_drift": ab.PALETTE[2], "betap_zavar":
↪  ab.PALETTE[3],
                "szelep_akadas": ab.PALETTE[1], "hutoviz_kieses":
                ↪  ab.PALETTE[4]}[e.hiba]
        a.axvspan(e.kezdet, e.veg, color=szin, alpha=0.25, lw=0)
from matplotlib.patches import Patch
ax[0].legend(handles=[Patch(color=c, alpha=0.4, label=l) for l, c in
                      [("szenzor-drift", ab.PALETTE[2]), ("betáplálási
↪  zavar", ab.PALETTE[3]),
                       ("szelepakadás", ab.PALETTE[1]), ("hűtővíz-kiesés",
↪  ab.PALETTE[4])]],
            ncol=4, fontsize=7, loc="upper left")
fig.autofmt_xdate()
fig.tight_layout()
```

A hirtelen fellépő hibákat (hűtővíz-kiesés, betáplálási zavar, szelepakadás) a rendszer egy órán belül jelzi – a 20 perces késés a perzisztenciaszabály miatti minimum –, a lassan kialakuló szenzor-driftet viszont csak órák, illetve napok múlva, amikor a mérési hiba már számottevő. A hozzájárulási ábra megmutatja, mely változók sértik a megszokott összefüggéseket:

```
oszlopok = list(X.columns)
fig, ax = plt.subplots(1, 2, sharey=True)
for a, (cim, t0, t1) in zip(ax, [("hűtővíz-kiesés (aug. 5.)", "2025-08-05
↪  07:00", "2025-08-05 11:00"),
                                 ("szenzor-drift (nov. 1.)", "2025-10-31
↪  00:00", "2025-11-02 03:00")]):
    sel = (X.index >= t0) & (X.index <= t1)
    hj = pd.Series((R[sel] ** 2).mean(axis=0), index=oszlopok)
    a.barh(oszlopok, hj.values, color=ab.PALETTE[0])
    a.set_title(cim, fontsize=9); a.set_xlabel("átlagos hozzájárulás")
ax[0].tick_params(axis="y", labelsize=7)
fig.tight_layout()
```

FUTÓ ESET · PANNON ÜZEM

A hűtővíz-kiesésnél a hozzájárulás egyértelműen a hűtővíz hőmérsékleté-

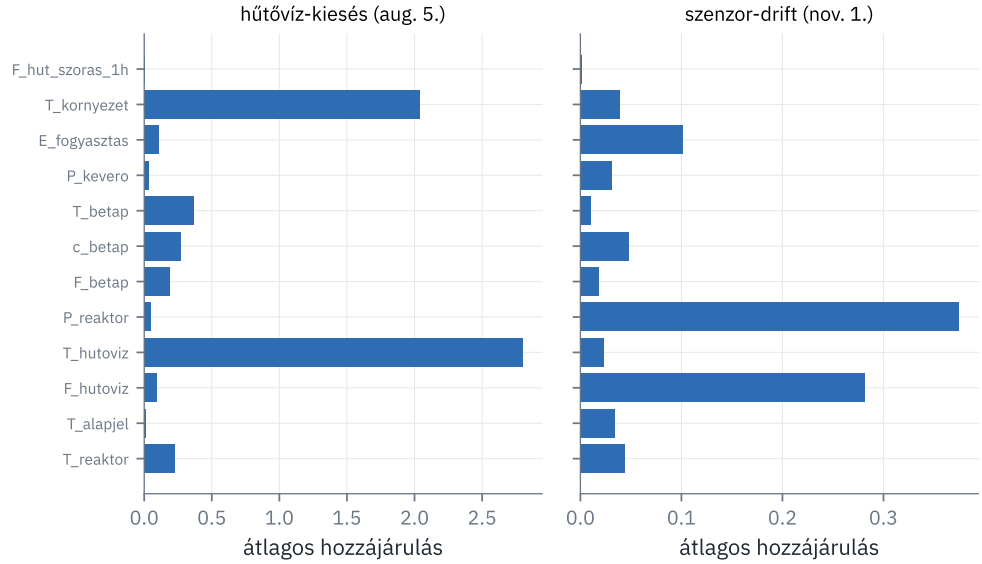


11.7. ábra. A T^2 és a Q statisztika (logaritmikus skálán) a második félév egy szakaszán az empirikus ellenőrzési határokkal. A színes sávok a valódi hibaeseményeket jelölik.

re mutat (a környezeti hőmérséklet azért jelenik meg, mert a normál üzemben a kettő együtt mozog, és ez az összefüggés sérül). A szenzor-driftnél a hőmérséklet-mérés maga alig tér el – hiszen a szabályozó a mért értéket tartja az alapjelen –, a hiba a nyomás és a hűtővízáram „ellentmondásában” látszik: a valódi hőmérséklet alacsonyabb, ezért a nyomás és a hűtési igény kisebb a mért hőmérséklethez tartozónál. Ez jól mutatja a hozzájárulási ábra korlátját: a hibás szenzor nem feltétlenül a legnagyobb hozzájárulású változó.

Végül vessük össze a statisztikák és az izolációs erdő pontszámainak eseménytípusonkénti elválasztóképességét (AUC) a második félévben:

```
iso = IsolationForest(n_estimators=300,
 ↪  random_state=0).fit(Z[normal_tanito])
pontszamok = {"T²": T2, "Q": Q, "T²/határ + Q/határ": T2 / T2_emp + Q /
 ↪  Q_emp,
              "izolációs erdő": -iso.score_samples(Z)}
tabla = {}
for nev, s in pontszamok.items():
    tabla[nev] = {}
```



11.8. ábra. A Q statisztika változónkénti hozzájárulása (átlagos r_j^2) két esemény idején: a hűtővíz-kiesésnél a hűtővíz hőmérséklete, a szenzor-drift késői szakaszában a nyomás és a hűtővízáram sérti a megszokott összefüggéseket.

```
    for f in ["szenzor_drift", "betap_zavar", "szelep_akadas",
     ↪  "hutoviz_kieses"]:
        sel = teszt & np.isin(allapot, ["normál", f])
        tabla[nev][f] = roc_auc_score(allapot[sel] == f, s[sel])
pd.DataFrame(tabla).T.round(3).rename_axis("AUC (2. félév)")
```

AUC (2. félév)	szenzor_drift	betap_zavar	szelep_akadas	hutoviz_kieses
T^2	0.588	0.849	1.000	0.968
Q	0.815	1.000	0.937	1.000
T^2/határ + Q/határ	0.780	0.997	1.000	1.000
izolációs erdő	0.655	0.837	0.887	0.951

A T^2 és a Q kiegészíti egymást: a szelepakadás (amely a hűtővízáram szórását, és így a modelltérben is szokatlan állapotot okoz) a T^2-ben, a betáplálási zavar és a szenzor-drift (sérült összefüggések) a Q-ban látszik jobban. Az izolációs erdő mindegyik hibatípusnál gyengébb a PCA-alapú statisztikáknál: a hibák itt elsősorban a változók közötti lineáris összefüggések megsérülésében nyilvánulnak meg, amire a tengelyekkel párhuzamos vágások kevésbé érzékenyek.

Interaktív alkalmazás: Anomália-észlelők összehasonlítása: Mahalanobis, LOF,

izolációs erdő

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/anomalia.html

Colab-notebook: 11. fejezet – anomáliák

11.9. Összefoglalás

- Az anomália más mechanizmus által létrehozott megfigyelés; lehet pontszerű, kontextuális vagy kollektív.
- A robusztus z-érték és a robusztus (MCD) kovarianciabecslés elkerüli, hogy a kiugró értékek elrejtsék magukat; a Mahalanobis-távolság a korrelációs szerkezetet is figyelembe veszi.
- Az EWMA- és a CUSUM-kártya a kis, tartós eltolódásokat gyorsabban észleli, mint a Shewhart-kártya, de autokorrelált adatokon a határokat korrigálni kell.
- A PCA-alapú folyamatfelügyelet T^2 statisztikája a modelltérben, Q statisztikája a maradéktérben méri az eltérést; a hozzájárulási ábra a hiba helyére utal, de nem diagnosztizál.
- A LOF a lokális sűrűséget, az izolációs erdő az elszigetelhetőséget, az egyosztályos SVM a normál tartomány határát modellezi.
- Az értékeléshez a kiegyensúlyozatlanság miatt az átlagos pontosság, a precision@k, a téves riasztási arány és az észlelési késés alkalmasabb, mint az AUC.
- A jellemzők megválasztása legalább annyira fontos, mint a módszeré: a módszer csak azt az eltérést láthatja, amelyet a jellemzők leírnak.

Ellenőrző kérdések

1. Miben különbözik a kontextuális és a kollektív anomália? Adjon példát mindkettőre a futó esetekből!
2. Mi a maszkolás, és hogyan védenek ellene a robusztus becslők?
3. Miért lehet egy pont kiugró a Mahalanobis-távolság szerint, ha minden koordinátája a szokásos tartományban van?
4. Mit jelez a nagy T^2, és mit a nagy Q érték a PCA-alapú folyamatfelügyeletben?
5. Miért kell kevesebb vágás egy kiugró pont elszigeteléséhez az izolációs fában?
6. Miért lehet félrevezető az AUC a csalásfelderítésben?
7. Miért ad az EWMA- és a CUSUM-kártya autokorrelált adatokon sok téves riasztást?

Feladatok

1. **Robusztus kovariancia.** Ismételje meg a csalásfelderítést a Mahalanobis-távolság MCD-alapú változatával (`MinCovDet`), csak a folytonos jellemzőkkel! Mi okoz gondot a diszkrét jellemzőknél?
2. **A LOF paramétere.** Vizsgálja meg a LOF átlagos pontosságát a $k = 5, 10, 20, 35, 50, 100$ értékekre! Mennyire érzékeny a módszer?
3. **Komponensszám.** Hogyan változik a PCA-alapú felügyelet téves riasztási aránya és a hibák észlelési aránya a megtartott komponensek számával ($q = 2, \dots, 8$)?
4. **Termékenkénti modellek.** Illesszen külön PCA-modellt mindhárom termékre, és vesse össze a téves riasztások számát az egyetlen modellel kapott értékkel!
5. **Autokorreláció.** Számítsa ki a 9. fejezet nyomásmaradékának órás átlagán az autokorrelációt, és alkalmazzon EWMA-kártyát az eredeti adatokra, illetve egy AR(1)-modell maradékára! Hasonlítsa össze a téves riasztások számát és a szenzor-drift észlelési késését!

További olvasmányok

Az anomália-észlelés átfogó áttekintése Chandola és mtsai. (2009) cikke és Aggarwal (2017) könyve; a klasszikus és a mélytanulási módszereket egységes keretben Ruff és mtsai. (2021) tárgyalja. A statisztikai folyamatszabályozásról Montgomery (2019), a többváltozós és PCA-alapú folyamatfelügyeletről MacGregor & Kourti (1995) és Qin (2012) ajánlott. A módszerek empirikus összehasonlítását Goldstein & Uchida (2016), Campos és mtsai. (2016) és S. Han és mtsai. (2022) végezte el; a csalásfelderítés statisztikai módszereiről Bolton & Hand (2002) ad áttekintést.

IV. Speciális adattípusok

12. Idősorok elemzése

Miről szól a fejezet?

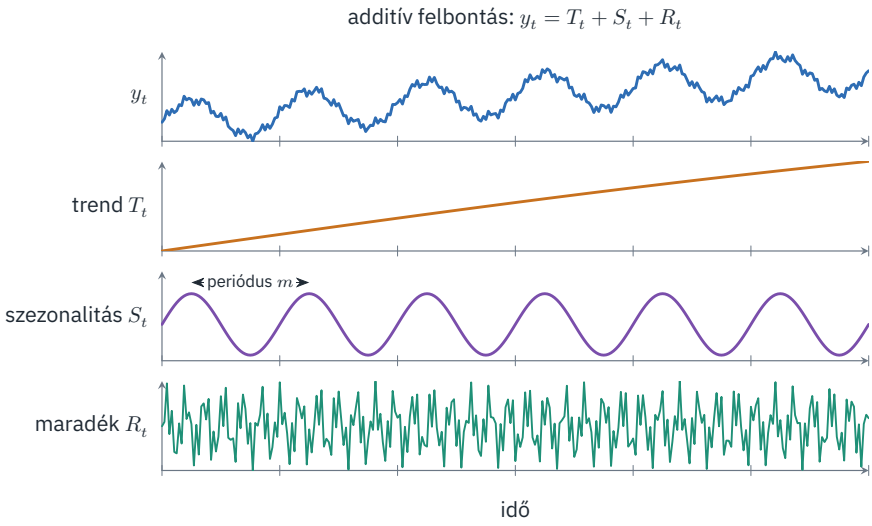
Az idősorok – időrendben rögzített megfigyelések – a futó esetek mindegyikében központi szerepet játszanak: a webáruház forgalma, az üzem mérései, a riasztási napló mind idősor. Az időbeli rendezettség új lehetőségeket és új veszélyeket hoz: a megfigyelések nem függetlenek, a jövő a múltból jelezhető előre, de a véletlen keresztvalidáció félrevezet. A fejezet az idősorok felbontásával és az autokorrelációval kezdődik, majd az előrejelzés klasszikus módszereit (exponenciális simítás, ARIMA-modellek, dinamikus regresszió) és értékelésük helyes módját tárgyalja. Ezután az idősorok hasonlóságát (dinamikus idővetemítés) és tömör reprezentációit (Fourier- és hullámtranszformáció, PAA, SAX) mutatjuk be, végül az idősorok osztályozását és szegmentálását.

A fejezet végére az olvasó képes lesz egy idősort trendre, szezonalitásra és maradékra bontani; az autokorrelációs függvényt értelmezni; exponenciális simító és (S)ARIMA-modellt illeszteni és gördülő kezdőpontú értékeléssel összehasonlítani egyszerű viszonyítási módszerekkel; a DTW-távolságot kézzel és programmal kiszámítani; és a legfontosabb idősor-reprezentációkat alkalmazni.

12.1. Az idősorok szerkezete

12.1.1. Felbontás

Egy $y_1, y_2, \ldots, y_T$ idősor változását rendszerint három komponensre bontjuk (12.1. ábra): a hosszú távú **trendre** (T_t), a rögzített periódussal ismétlődő **szezonalitásra** (S_t, periódusa m: órás adatoknál a napi ciklusra $m = 24$, heti adatoknál az éves ciklusra $m \approx 52$), és a **maradékra** (R_t). Additív felbontásban $y_t = T_t + S_t + R_t$; ha a szezonális ingadozás a szinttel arányos, multiplikatív felbontás ($y_t = T_t \cdot S_t \cdot R_t$) vagy a logaritmált sor additív felbontása a megfelelő. A klasszikus felbontás a trendet m hosszú centrált mozgóátlaggal becsüli, a szezonális komponenst pedig a trendtől megtisztított értékek periódusonkénti átlagaként. Az **STL** (seasonal-trend decomposition using LOESS) (R. B. Cleveland és mtsai., 1990) lokális regresszióval, iteratívan becsli a komponenseket; a szezonális mintázat lassan változhat, és robusztus változata a kiugró értékekre sem érzékeny (Hyndman & Athanasopoulos, 2021).



12.1. ábra. Az additív idősor-felbontás: a megfigyelt sor (felül) a trend, a periodikus szezonális komponens és a maradék összege.

12.1.2. Stacionaritás és autokorreláció

Egy idősor (gyengén) **stacionárius**, ha várható értéke és szórása időben állandó, és a két időpont közötti kovariancia csak a késleltetéstől függ. A stacionaritás az előrejelző modellek többségének alapfeltevése; a trend és a szezonalitás megsérti, ezért ezeket differenciálással ($\nabla y_t = y_t - y_{t-1}$, illetve szezonálisan $\nabla_m y_t = y_t - y_{t-m}$) vagy felbontással távolítjuk el. A stacionaritás formálisan is vizsgálható, például a kiterjesztett Dickey–Fuller-próbával (Dickey & Fuller, 1979).

Az idősor belső függőségi szerkezetét az **autokorrelációs függvény** (ACF) írja le:

$$r_k = \frac{\sum_{t=k+1}^{T} (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T} (y_t - \bar{y})^2}, \qquad k = 1, 2, \ldots \tag{12.1}$$

A **parciális autokorrelációs függvény** (PACF) a k késleltetésű korrelációt a közbülső késleltetések hatásának kiszűrése után méri. A két függvény mintázata az ARIMA-modellek rendjének megválasztását segíti, a modell maradékainak ACF-je pedig a modell megfelelőségét mutatja: jó modell maradéka fehér zaj, amit a Ljung–Box-próba (Ljung & Box, 1978) egyszerre több késleltetésre ellenőriz. Az autokorreláció a 11. fejezet ellenőrzőkártyáinál is gondot okozott, és ez az oka

annak is, hogy idősoroknál a véletlen keresztvalidáció túl optimista képet ad (7. fejezet).

12.2. Előrejelzés

12.2.1. Viszonyítási módszerek

Minden előrejelző modellt egyszerű viszonyítási (benchmark) módszerekkel kell összevetni (Hyndman & Athanasopoulos, 2021). A **naiv** előrejelzés az utolsó megfigyelést ismétli, $\hat{y}_{T+h} = y_T$; a **szezonális naiv** az egy periódussal korábbi értéket, $\hat{y}_{T+h} = y_{T+h-m}$; a **sodródásos** változat a naiv előrejelzést az eddigi átlagos változással tolja el. A nagy nemzetközi előrejelzési versenyek egyik visszatérő tanulsága, hogy ezeket a módszereket meglepően nehéz legyőzni, és hogy a bonyolult módszerek csak akkor jobbak, ha az adatok szerkezete ezt indokolja (Makridakis és mtsai., 2018).

12.2.2. Exponenciális simítás

Az **egyszerű exponenciális simítás** az előrejelzést a múltbeli értékek exponenciálisan csökkenő súlyú átlagaként képzi: $\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}$, és $\hat{y}_{T+h} = \ell_T$. Holt (Holt, 2004) kiterjesztése egy trendkomponenssel, Winters (Winters, 1960) pedig egy szezonális komponenssel egészítette ki; az additív **Holt–Winters-módszer** egyenletei

$$\begin{aligned} \ell_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}), \qquad & b_t &= \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1}, \\ s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}, \qquad & \hat{y}_{T+h} &= \ell_T + h\,b_T + s_{T+h-m(k+1)}, \end{aligned} \tag{12.2}$$

ahol ℓ_t a szint, b_t a trend, s_t a szezonális komponens, k pedig a $(h-1)/m$ egészrésze. A hosszú távú előrejelzések túlzott meredekségét a **csillapított trend** mérsékli, amely a trendet minden lépésben egy $\phi < 1$ tényezővel szorozza. A simítási paramétereket a maximum-likelihood vagy a legkisebb négyzetek elve szerint becsüljük; a módszercsalád állapottér-modellként (ETS: error–trend–seasonal) is felírható, ami előrejelzési intervallumokat és információs kritériumokon alapuló modellválasztást tesz lehetővé (Hyndman & Athanasopoulos, 2021).

12.2.3. ARIMA-modellek

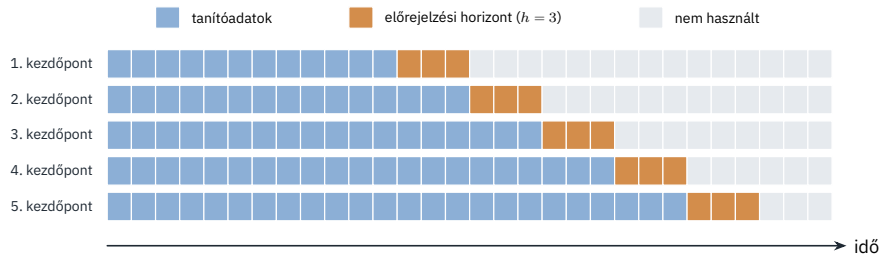
Az **autoregresszív** AR(p) modell az idősort saját késleltetett értékeinek lineáris függvényeként, a **mozgóátlag** MA(q) modell a korábbi véletlen sokkok lineáris kombinációjaként írja le. A kettő kombinációja a differenciált sorra az **ARIMA(p, d, q)** modell (Box és mtsai., 2015):

$$\left(1 - \sum_{i=1}^{p} \phi_i B^i\right)(1 - B)^d y_t = c + \left(1 + \sum_{j=1}^{q} \theta_j B^j\right)\varepsilon_t, \tag{12.3}$$

ahol B a késleltető operátor ($By_t = y_{t-1}$), ε_t fehér zaj. A szezonális **SARIMA(p, d, q)(P, D, Q)$_m$** modell ugyanezt a szerkezetet a szezonális késleltetésekre (B^m) is alkalmazza. A Box–Jenkins-módszertan lépései: a sor stacionáriussá tétele (differenciálás, transzformáció), a rend azonosítása az ACF és a PACF alapján (vagy információs kritériummal, például AIC), a paraméterek becslése, és a maradékok diagnosztikája. Ha a jövőbeli értékeket befolyásoló, ismert vagy előre jelezhető külső változók (termelési terv, időjárás-előrejelzés, akciók) vannak, a **dinamikus regresszió** – lineáris regresszió ARIMA-hibataggal (ARIMAX, SARIMAX) – ezeket is felhasználja.

12.2.4. Az előrejelzések értékelése

Az előrejelzést mindig olyan adatokon kell értékelni, amelyek a modell illesztésekor még nem voltak ismertek. Egyetlen tanító–teszt felosztás helyett a **gördülő kezdőpontú értékelés** (rolling origin, idősoros keresztvalidáció) több kezdőpontból készít előrejelzést, mindig csak a kezdőpont előtti adatokat felhasználva (12.2. ábra) (Bergmeir & Benítez, 2012; Hyndman & Athanasopoulos, 2021).



12.2. ábra. Gördülő kezdőpontú értékelés: minden kezdőpontban csak a korábbi adatokon illesztjük a modellt, és a következő h időpontra adott előrejelzést értékeljük.

A hibamértékek közül a MAE és az RMSE az eredeti mértékegységben értelmezhető, de sorok között nem hasonlítható össze. A MAPE (átlagos abszolút százalékos hiba) nullához közeli értékeknél instabil, és aszimmetrikusan bünteti a túl- és az alulbecslést. Hyndman & Koehler (2006) ezért a **skálázott abszolút hibát** (MASE) javasolta, amely a MAE-t a tanítóadatokon számolt naiv (vagy szezonális naiv) egylépéses előrejelzés MAE-jével osztja: az 1-nél kisebb érték azt jelenti, hogy a módszer jobb a naiv viszonyításnál.

12.3. Hasonlóság és reprezentáció

12.3.1. Dinamikus idővetemítés

Két idősor euklideszi távolsága az azonos időpontú értékeket veti össze, ezért már egy kis időbeli eltolódás vagy eltérő sebesség is nagy távolságot ad. A **dinamikus idővetemítés** (dynamic time warping, DTW) (Berndt & Clifford, 1994; Sakoe & Chiba, 1978) az időtengelyt rugalmasan „vetemíti”: azt a monoton illesztést keresi a két sor pontjai között, amely mentén az összesített eltérés a legkisebb (12.3. ábra). Az **a** és **b** sorok kumulált költségmátrixát dinamikus programozással töltjük ki:

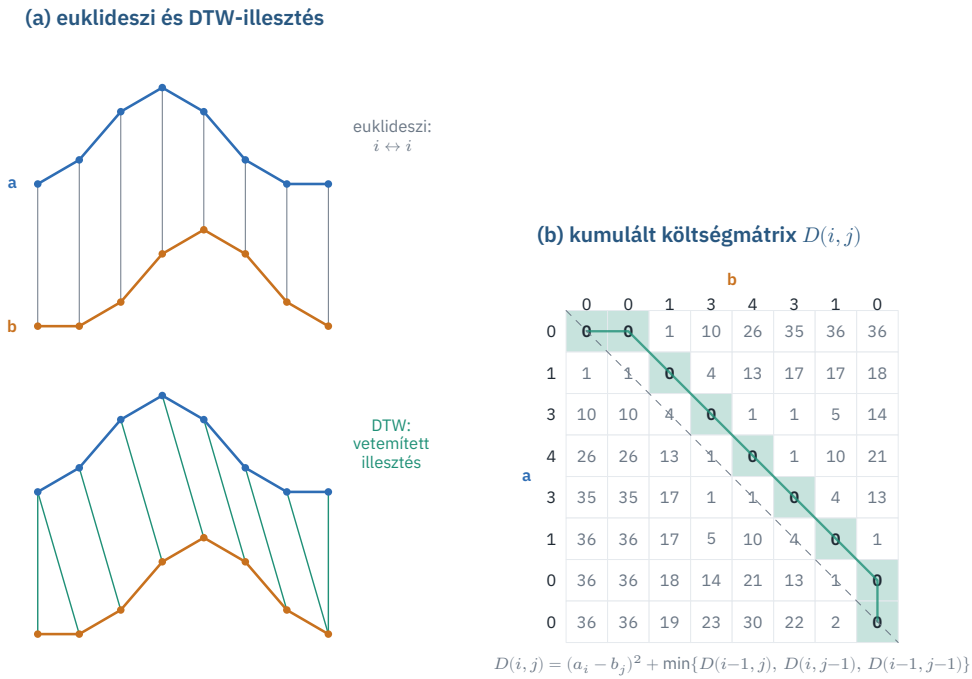
$$D(i,j) = (a_i - b_j)^2 + \min\bigl\{D(i-1,j),\; D(i,j-1),\; D(i-1,j-1)\bigr\}, \qquad D(0,0) = 0, \tag{12.4}$$

és $\mathrm{DTW}(\mathbf{a},\mathbf{b}) = \sqrt{D(n,m)}$. Az optimális illesztés (vetemítési út) a $D(n,m)$ cellából visszakövethető. A számítás költsége $O(nm)$; a **Sakoe–Chiba-sáv** az utat a főátló w szélességű környezetére korlátozza, ami gyorsít, és megakadályozza a fizikailag értelmetlen, túlzott vetemítést. Nagy adatbázisokban a DTW-t olcsó alsó korlátokkal (például az LB_Keogh-korláttal (Keogh & Ratanamahatana, 2005)) kombinálják, így a legtöbb jelöltet a teljes számítás nélkül ki lehet zárni (Rakthanmanon és mtsai., 2012).

12.3.2. Kézi példa

A segédcsomag két rövid jelalakja ugyanaz a „hegy”, egy lépéssel eltolva. Az euklideszi távolság a (12.4) rekurzió nélkül, a pontonkénti eltérésekből számolható:

```
a, b = ab.mini.idosor_mini()
print("a =", a, "\nb =", b)
print(f"euklideszi távolság: {np.sqrt(((a - b) ** 2).sum()):.3f}")
```



12.3. ábra. A dinamikus idővetemítés kézi példája. (a) Az euklideszi távolság az azonos indexű pontokat veti össze, a DTW a jelalakokat illeszti egymáshoz. (b) A kumulált költségmátrix a vetemítési úttal (kiemelve); a szaggatott vonal az euklideszi illesztésnek megfelelő főátló.

```
def dtw(a, b, w=None):
    """Kumulált költségmátrix a (-@eq-dtw) rekurzióval, opcionális
    ↪ Sakoe–Chiba-sávval."""
    n, m = len(a), len(b)
    D = np.full((n + 1, m + 1), np.inf)
    D[0, 0] = 0.0
    for i in range(1, n + 1):
        jlo, jhi = (1, m) if w is None else (max(1, i - w), min(m, i + w))
        for j in range(jlo, jhi + 1):
            D[i, j] = (a[i - 1] - b[j - 1]) ** 2 + min(D[i - 1, j], D[i, j -
↪ 1], D[i - 1, j - 1])
    return D

D = dtw(a, b)
print(f"DTW-távolság: {np.sqrt(D[-1, -1]):.3f}")
pd.DataFrame(D[1:, 1:].astype(int), index=[f"a{i+1}={v:g}" for i, v in
↪ enumerate(a)],
             columns=[f"b{j+1}={v:g}" for j, v in enumerate(b)])
```

```
a = [0. 1. 3. 4. 3. 1. 0. 0.]
b = [0. 0. 1. 3. 4. 3. 1. 0.]
euklideszi távolság: 3.464
DTW-távolság: 0.000
```

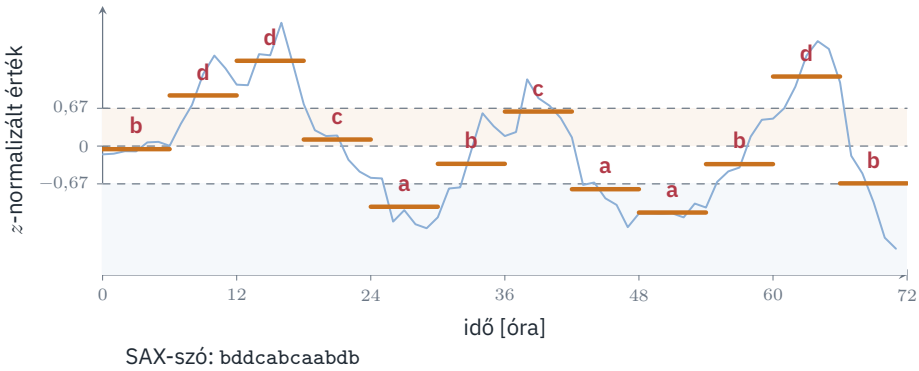
	b1=0	b2=0	b3=1	b4=3	b5=4	b6=3	b7=1	b8=0
a1=0	0	0	1	10	26	35	36	36
a2=1	1	1	0	4	13	17	17	18
a3=3	10	10	4	0	1	1	5	14
a4=4	26	26	13	1	0	1	10	21
a5=3	35	35	17	1	1	0	4	13
a6=1	36	36	17	5	10	4	0	1
a7=0	36	36	18	14	21	13	1	0
a8=0	36	36	19	23	30	22	2	0

A DTW-távolság nulla: a vetemítési út az a_1 értéket b_1-hez és b_2-höz is hozzárendeli, a hegyet egy lépéssel eltolva tökéletesen illeszti, végül a_8-at is b_8-hoz rendeli. Az euklideszi távolság ezzel szemben 3,46, pedig a két jelalak szemmel láthatóan ugyanaz.

12.3.3. Tömör reprezentációk

Hosszú idősorok hasonlóságkereséséhez, csoportosításához és indexeléséhez az idősort kevés együtthatóval leíró, tömör **reprezentációra** van szükség, amelyben a távolság jól közelíti (lehetőleg alulról becsüli) az eredeti távolságot (Faloutsos és mtsai., 1994). A legfontosabbak:

- A **diszkrét Fourier-transzformáció** (DFT) az idősort különböző frekvenciájú szinuszok összegeként írja le, $X_k = \sum_{t=0}^{T-1} y_t e^{-2\pi i k t/T}$; az első néhány együttható a lassú változásokat őrzi meg (Agrawal, Faloutsos, és mtsai., 1993). A **periodogram** ($|X_k|^2$ a frekvencia függvényében) a rejtett periodicitások felderítésének eszköze.
- A **diszkrét hullámtranszformáció** (DWT) (Daubechies, 1992) időben és frekvenciában egyaránt lokalizált bázisfüggvényeket használ; a legegyszerűbb Haar-hullámnál az együtthatók szomszédos szakaszok átlagai és különbségei (Chan & Fu, 1999). Hirtelen változásokat és tranzienseket jobban leír, mint a DFT.
- A **darabonként konstans közelítés** (piecewise aggregate approximation, PAA) (Keogh és mtsai., 2001) az idősort w egyenlő hosszú szakaszra bontja, és mindegyiket az átlagával helyettesíti.
- A **SAX** (symbolic aggregate approximation) (Lin és mtsai., 2007) a z-normalizált sor PAA-értékeit a standard normális eloszlás kvantilisei által meghatározott sávok szerint betűkké alakítja (12.4. ábra). Az így kapott szövegszerű reprezentációra a szövegbányászat és a sorozatminta-keresés eszközei (például a 6. fejezet módszerei) is alkalmazhatók.



12.4. ábra. A PAA és a SAX reprezentáció a hűtővízáram három napos, z-normalizált órás adatsorán. A 12 szakasz átlaga (vastag vonal) a normális eloszlás kvantilisei (szaggatott vonalak) szerint négybetűs ábécé elemeivé alakul.

12.4. Idősorok osztályozása és szegmentálása

Az idősor-osztályozás feladata egy teljes idősor (vagy idősorszakasz) osztályba sorolása: egy gépindítás jó vagy rossz lefutású volt-e, milyen hibatípus jellemző egy jelablakra. Kiterjedt összehasonlító vizsgálatok szerint az 1-legközelebbi

szomszéd osztályozó DTW-távolsággal évtizedekig nehezen legyőzhető viszonyítási alap volt (Bagnall és mtsai., 2017); az újabb módszerek – az idősorok jellemzőit véletlen konvolúciós szűrőkkel kinyerő ROCKET (A. Dempster és mtsai., 2020) és a 10. fejezet konvolúciós hálózatai (Ismail Fawaz és mtsai., 2019) – a pontosságot és a sebességet tekintve egyaránt előrelépést hoztak. A módszerek értékelésének közös alapja a nyilvános UCR-idősorarchívum (Dau és mtsai., 2019).

A **szegmentálás** az idősort belsőleg homogén szakaszokra bontja: működési állapotokra, termékváltásokra, hibaepizódokra. A klasszikus algoritmusok (csúszóablakos, felülről lefelé és alulról felfelé haladó módszerek) a szakaszokat egyszerű – például lineáris – modellekkel közelítik, és a közelítési hiba alapján vágnak (Keogh és mtsai., 2004); a **változáspont-észlelés** a szakaszok számát és határait egy büntetett költségfüggvény optimumaként határozza meg, amire a PELT-algoritmus lineáris időben ad egzakt megoldást (Killick és mtsai., 2012). Többváltozós folyamatadatoknál a szakaszok homogenitása a változók közötti összefüggések (például lokális PCA-modellek) alapján is mérhető; ilyen a fuzzy klaszterezésen alapuló szegmentálás, amely a szakaszhatárok bizonytalanságát is kezeli (Abonyi és mtsai., 2005).

12.5. Gyakorlat: forgalom-előrejelzés, hűtési igény és jelalakok

12.5.1. A heti rendelésszám előrejelzése

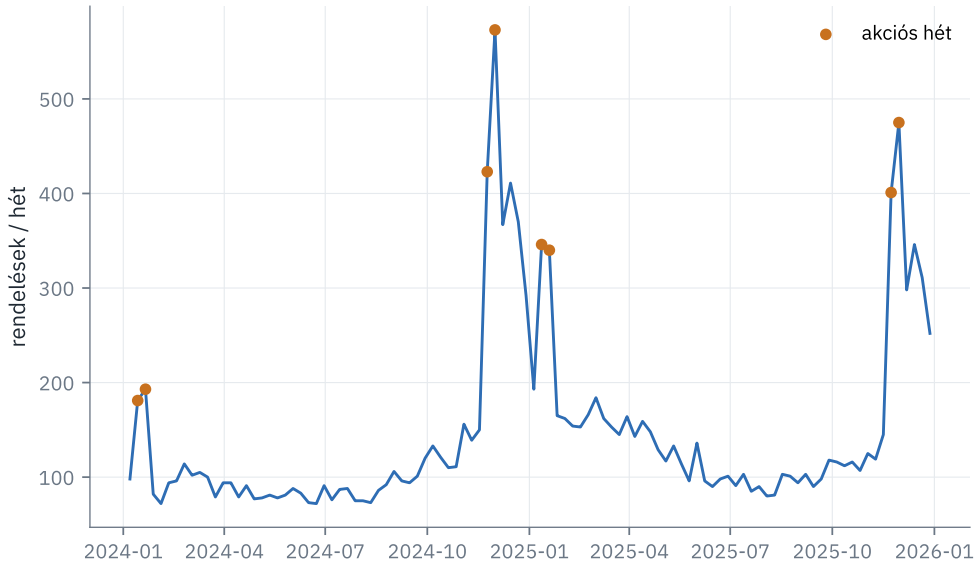
FUTÓ ESET · PANNONSHOP

A PannonShop logisztikája a következő négy hét rendelésszámát szeretné előre látni a raktári létszám tervezéséhez. A heti rendelésszám két év adatából áll; erős éves szezonalitás (az év végi csúcs), két akciós időszak és növekedés jellemzi.

```
tx = ab.shop_clean()
heti = tx.set_index("InvoiceDate").groupby(pd.Grouper(freq="W-SUN"
↪  )).Invoice.nunique()
heti = heti[(heti.index > "2024-01-01") & (heti.index <= "2025-12-28")]
het_sorszam = heti.index.isocalendar().week.to_numpy().astype(int)
akcio = np.isin(het_sorszam,
↪  ab.load_shop().truth["promo_weeks"][2025]).astype(float)
fig, ax = plt.subplots()
ax.plot(heti.index, heti.values, color=ab.PALETTE[0], lw=1.3)
```

```
ax.scatter(heti.index[akcio == 1], heti.values[akcio == 1],
 ↪  color=ab.PALETTE[1], s=18, zorder=3, label="akciós hét")
ax.set_ylabel("rendelések / hét"); ax.legend()
fig.tight_layout()
print(heti.groupby(heti.index.year).sum().rename("rendelések száma"))
```

```
InvoiceDate
2024    7001
2025    8207
Name: rendelések száma, dtype: int64
```



12.5. ábra. A PannonShop heti rendelésszáma. Az év végi csúcs és az akciós hetek (pontok) határozzák meg a sor szerkezetét; a második évben a forgalom nagyobb.

Négy módszert hasonlítunk össze gördülő kezdőpontú értékeléssel (a 65. héttől négyhetente, négyhetes horizonttal): a naiv és a szezonális naiv előrejelzést, a szezonális naiv előrejelzés növekedéssel korrigált változatát (az elmúlt nyolc hét és az egy évvel korábbi nyolc hét log-szintjének különbségével), valamint egy dinamikus harmonikus regressziót, amely a logaritmált rendelésszámot Fourier-tagokkal (az éves szezonalitás), trenddel, valamint az akciós és az ünnepi hetek indikátoraival magyarázza, AR(1) hibataggal. Az éves periódusú Holt–Winters- és SARIMA-modellekhez két év kevés: a szezonális komponens becsléséhez legalább két teljes periódus kellene már a legelső kezdőpontban is.

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

y = np.log(heti.to_numpy())
t = np.arange(len(y))
unnepi = np.isin(het_sorszam, [49, 50, 51, 52]).astype(float)
def fourier(t, K, P=52.18):
    return np.column_stack([f(2 * np.pi * k * t / P) for k in range(1, K +
    ↪ 1) for f in (np.sin, np.cos)])
Xdhr = np.column_stack([fourier(t, 3), t / 52, akcio, unnepi])
H = 4
kezdopontok = range(64, len(y) - H + 1, 4)

def naiv(o):            return np.repeat(y[o - 1], H)
def szez_naiv(o):       return y[o - 52:o - 52 + H]
def szez_naiv_nov(o):   return y[o - 52:o - 52 + H] + (y[o - 8:o].mean() -
↪ y[o - 60:o - 52].mean())
def dhr(o):
    illesztett = SARIMAX(y[:o], exog=Xdhr[:o], order=(1, 0, 0),
↪ trend="c").fit(disp=False, maxiter=200)
    return illesztett.forecast(H, exog=Xdhr[o:o + H])

skala = np.mean(np.abs(np.diff(heti.to_numpy()[:64])))     # naiv
↪ egylépéses MAE a tanítóadatokon
eredm = {}
for nev, f in [("naiv", naiv), ("szezonális naiv", szez_naiv),
               ("szezonális naiv + növekedés", szez_naiv_nov), ("dinamikus
↪ harmonikus regresszió", dhr)]:
    hibak = np.concatenate([np.abs(np.exp(f(o)) - heti.to_numpy()[o:o + H])
↪ for o in kezdopontok])
    eredm[nev] = {"MAE [rendelés/hét]": hibak.mean(), "MASE": hibak.mean()
↪ / skala}
pd.DataFrame(eredm).T.round(2)
```

	MAE [rendelés/hét]	MASE
naiv	45.45	1.48
szezonális naiv	28.12	0.91
szezonális naiv + növekedés	16.65	0.54
dinamikus harmonikus regresszió	26.74	0.87

FUTÓ ESET · PANNONSHOP

A legjobb eredményt a legegyszerűbb, a szakterületi tudást (az éves ismétlődést és a növekedést) közvetlenül felhasználó módszer adja: a növekedéssel korrigált szezonális naiv előrejelzés MASE-értéke jóval 1 alatt van. A dinamikus harmonikus regresszió ugyan jobb a szezonális naiv módszernél, de két év adatából a Fourier-tagok és a indikátorváltozók együtthatói bizonyta-

lanok, és a hirtelen év végi felfutást csak elmosódottan írják le. A tanulság általános: rövid, erősen szezonális soroknál a jól megválasztott viszonyítási módszer gyakran a legjobb választás, és egy bonyolultabb modell bevezetését mindig ehhez kell mérni.

12.5.2. Óránkénti hűtési igény előrejelzése

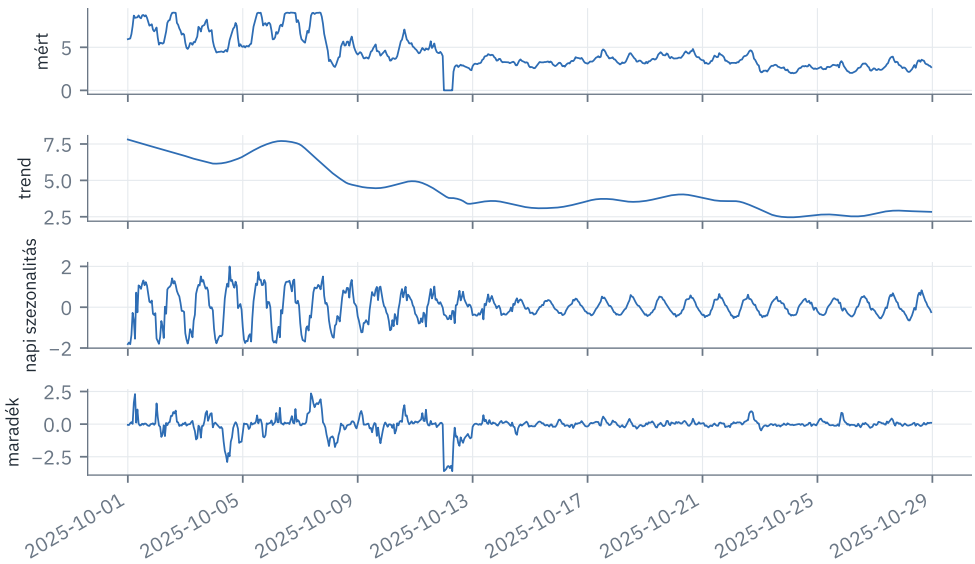
FUTÓ ESET · PANNON ÜZEM

A hűtővízrendszer üzemeltetője 24 órával előre szeretné látni a hűtővízigényt. Az órás adatok bőségesek, a sor napi periodicitású (a környezeti hőmérséklet napi járása miatt), és szintje a gyártott terméktől és a hűtőköpeny állapotától függ. Bontsuk fel a sort STL-lel, és vizsgáljuk meg az autokorrelációját!

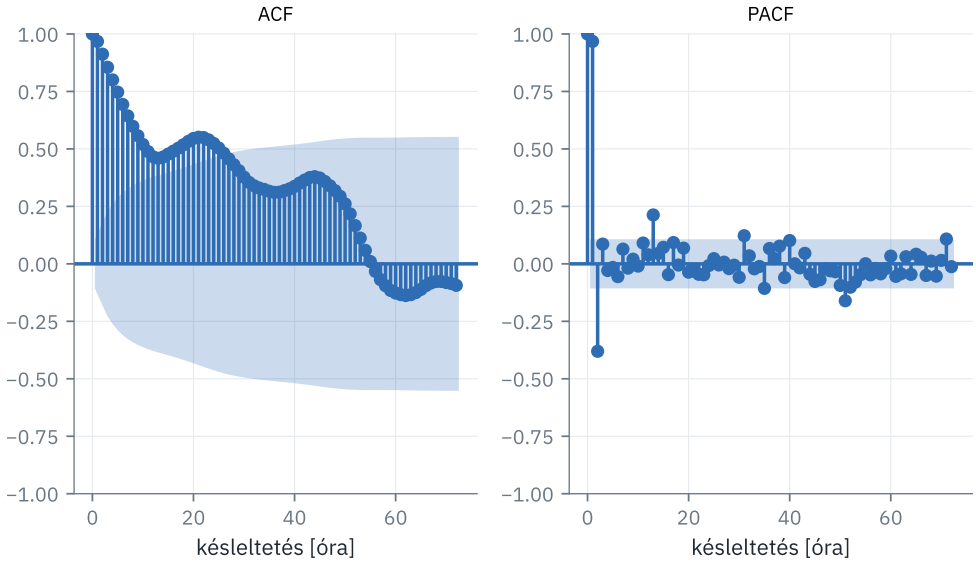
```
from statsmodels.tsa.seasonal import STL
ora = ab.load_plant().hourly.set_index("ido")
s = ora.F_hutoviz["2025-08-26":"2025-12-05"]
stl = STL(s["2025-10-01":"2025-10-28"], period=24, robust=True).fit()
fig, ax = plt.subplots(4, 1, sharex=True)
for a, (sor, nev) in zip(ax, [(stl.observed, "mért"), (stl.trend, "trend"),
 ↪  (stl.seasonal, "napi szezonalitás"), (stl.resid, "maradék")]):
    a.plot(sor.index, sor.values, color=ab.PALETTE[0], lw=0.8);
 ↪  a.set_ylabel(nev, fontsize=8)
fig.autofmt_xdate()
fig.tight_layout()
```

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
fig, ax = plt.subplots(1, 2)
plot_acf(s["2025-09-01":"2025-09-14"], lags=72, ax=ax[0],
 ↪  color=ab.PALETTE[0], vlines_kwargs={"colors": ab.PALETTE[0]})
plot_pacf(s["2025-09-01":"2025-09-14"], lags=72, ax=ax[1],
 ↪  color=ab.PALETTE[0], vlines_kwargs={"colors": ab.PALETTE[0]},
 ↪  method="ywm")
for a, cim in zip(ax, ["ACF", "PACF"]):
    a.set_title(cim, fontsize=9); a.set_xlabel("késleltetés [óra]")
fig.tight_layout()
```

Öt módszert hasonlítunk össze tizenöt kezdőponttal (október közepétől háromnaponta), 24 órás horizonttal, mindig az előző három hét adataira illesztve: a szezonális naiv előrejelzést (a tegnapi azonos óra), a csillapított trendű Holt–Winters-módszert, egy SARIMA$(1,0,1)(0,1,1)_{24}$ modellt, valamint két di-



12.6. ábra. A hűtővízáram órás idősorának STL-felbontása (24 órás periódus, robusztus változat) négy hétre. A trend az október 12-i leállás – a hűtőköpeny tisztítása – után jelentősen csökken, mert a tiszta hőátadó felület kevesebb hűtővizet igényel; a napi szezonalitás amplitúdója a környezeti hőmérséklet napi ingadozását követi.



12.7. ábra. A hűtővízáram órás sorának autokorrelációs (balra) és parciális autokorrelációs függvénye (jobbra) az első két hétre. A 24 óránként ismétlődő csúcsok a napi szezonalitást jelzik.

namikus regressziót. Ez utóbbiak a termelési tervből ismert alapjelet, betáplálási áramot és koncentrációt, valamint a környezeti hőmérsékletet is felhasználják; a jövőbeli környezeti hőmérsékletet itt a mért értékkel helyettesítjük, ami az időjárás-előrejelzés hibáját figyelmen kívül hagyja, tehát a valóságnál kissé optimista.

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.stats.diagnostic import acorr_ljungbox

kulso = ora.loc["2025-08-26":"2025-12-05", ["T_alapjel", "F_betap",
↪  "c_betap", "T_kornyezet"]]
H = 24
kezdo = pd.date_range("2025-10-15", "2025-11-28", freq="3D")
modszerek = {
    "szezonális naiv": lambda tr, Xtr, Xte: tr.to_numpy()[-24:],
    "Holt–Winters (csillapított)": lambda tr, Xtr, Xte:
    ↪  ExponentialSmoothing(
        tr.to_numpy(), trend="add", damped_trend=True, seasonal="add",
↪   seasonal_periods=24).fit().forecast(H),
    "SARIMA(1,0,1)(0,1,1)$_{24}$": lambda tr, Xtr, Xte: SARIMAX(
```

```
        tr.to_numpy(), order=(1, 0, 1), seasonal_order=(0, 1, 1,
↪  24)).fit(disp=False).forecast(H),
    "regresszió + AR(1) hiba": lambda tr, Xtr, Xte: SARIMAX(
        tr.to_numpy(), exog=Xtr, order=(1, 0, 0),
↪  trend="c").fit(disp=False).forecast(H, exog=Xte),
    "SARIMAX": lambda tr, Xtr, Xte: SARIMAX(
        tr.to_numpy(), exog=Xtr, order=(1, 0, 1), seasonal_order=(0, 1, 1,
↪  24)).fit(disp=False).forecast(H, exog=Xte),
}
eredm = {}
for nev, f in modszerek.items():
    mae, mase = [], []
    for o in kezdo:
        tr_idx = slice(o - pd.Timedelta(days=21), o - pd.Timedelta(hours=1))
        te_idx = slice(o, o + pd.Timedelta(hours=H - 1))
        elorejelzes = f(s[tr_idx], kulso[tr_idx].to_numpy(),
↪  kulso[te_idx].to_numpy())
        hiba = np.mean(np.abs(elorejelzes - s[te_idx].to_numpy()))
        skala = np.mean(np.abs(s[tr_idx].diff(24).dropna()))         #
↪  szezonális naiv MAE a tanítóablakon
        mae.append(hiba); mase.append(hiba / skala)
    eredm[nev] = {"MAE [m³/h]": np.mean(mae), "MASE (szezonális)":
↪  np.mean(mase),
                  "a kezdőpontok hányadában jobb a szez. naivnál": None}
    eredm[nev]["_mae"] = mae
alap = np.array(eredm["szezonális naiv"]["_mae"])
for nev in eredm:
    eredm[nev]["a kezdőpontok hányadában jobb a szez. naivnál"] =
↪  np.mean(np.array(eredm[nev].pop("_mae")) < alap)
pd.DataFrame(eredm).T.astype(float).round(3)
```

	MAE [m³/h]	MASE (szezonális)	a kezdőpontok hányadában jobb a szez. naivnál
szezonális naiv	0.528	0.846	0.000
Holt–Winters (csillapított)	0.509	0.783	0.533
SARIMA(1,0,1)(0,1,1)$_{24}$	0.547	0.823	0.600
regresszió + AR(1) hiba	0.456	0.655	0.733
SARIMAX	0.415	0.631	0.800

```
utolso = SARIMAX(s["2025-11-05":"2025-11-25"].to_numpy(), order=(1, 0, 1),
↪  seasonal_order=(0, 1, 1, 24)).fit(disp=False)
lb = acorr_ljungbox(utolso.resid[25:], lags=[24, 48])
print("Ljung–Box-próba a SARIMA maradékain (p-érték):",
↪  lb["lb_pvalue"].round(4).to_dict())
```

```
Ljung–Box-próba a SARIMA maradékain (p-érték): {24: 0.0269, 48: 0.1404}
```

FUTÓ ESET · PANNON ÜZEM

A tisztán idősoros modellek (Holt–Winters, SARIMA) legfeljebb kevéssel jobbak a szezonális naiv előrejelzésnél – a kezdőpontok megválasztásától függően akár rosszabbak is –, igazán nagy javulást a külső változók hoznak: a termelési terv és a környezeti hőmérséklet ismeretében a hűtési igény szintjének változásai – amelyeket a tisztán idősoros modellek csak utólag követnek – előre láthatók. A szezonális tagot is tartalmazó SARIMAX a legjobb. A SARIMA-modell maradékaiban a Ljung–Box-próba szerint rövid távon (24 óráig) szignifikáns autokorreláció marad: a termékváltások okozta szintugrásokat a tisztán idősoros modell nem tudja leírni – ez is a külső változók felhasználása mellett szól. A táblázat utolsó oszlopa arra is figyelmeztet, hogy az átlagos hiba mögött nagy kezdőpontonkénti ingadozás áll: kevés kezdőpont esetén a módszerek sorrendje véletlenszerűen is megfordulhat, ezért érdemes sok kezdőpontot használni, és a hibák eloszlását is megnézni. Ez ugyanaz a tanulság, mint a 10. fejezet hűtési igény modelljénél: a folyamat fizikájából ismert bemenetek felhasználása többet ér, mint a modell bonyolultsága.

12.5.3. Jelalakok osztályozása dinamikus idővetemítéssel

A DTW előnye ott mutatkozik meg, ahol az azonos osztályba tartozó jelalakok időben eltolódnak vagy eltérő sebességgel zajlanak le. Ennek klasszikus mintapéldája a „henger–harang–tölcsér” (Cylinder–Bell–Funnel, CBF) feladat (N. Saito, 1994), amely az UCR-archívum egyik referenciaadatsora is (Dau és mtsai., 2019): három jelalak (állandó szintű szakasz, felfutó és lefutó rámpa), véletlen kezdettel, hosszal és amplitúdóval, zajjal terhelve. A DTW-t sok sorpárra egyszerre, a párok mentén vektorizálva számítjuk.

```
def cbf(n_osztalyonkent, rng, L=128):
    """A Cylinder–Bell–Funnel mintaadatsor generátora (Saito, 1994)."""
    t = np.arange(L)
    X, y = [], []
    for c in range(3):
        for _ in range(n_osztalyonkent):
            a = rng.integers(16, 33); b = min(a + rng.integers(32, 97), L -
↪ 1)
            chi = ((t >= a) & (t <= b)).astype(float)
           alak = [chi, chi * (t - a) / (b - a), chi * (b - t) / (b - a)][c]
            X.append((6 + rng.normal()) * alak + rng.normal(0, 1, L));
↪ y.append(c)
    return np.array(X), np.array(y)
```

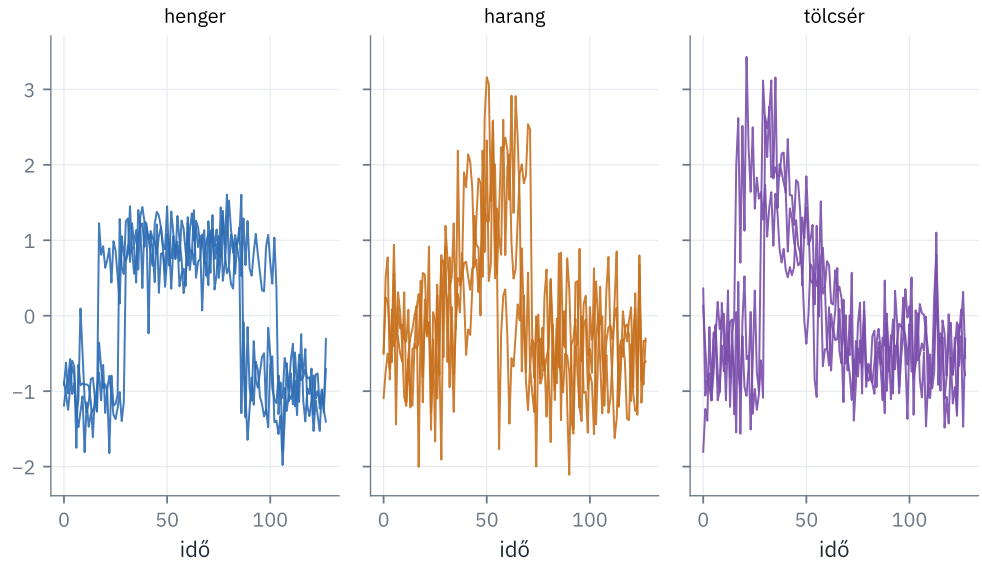
```
def dtw_parok(A, B, w=None):
    """DTW-távolság az A[k] és B[k] sorpárokra, a párok mentén
    ↪ vektorizálva."""
    n, L = A.shape
    elozo = np.full((n, L + 1), np.inf); elozo[:, 0] = 0.0
    for i in range(1, L + 1):
        akt = np.full((n, L + 1), np.inf)
        jlo, jhi = (1, L) if w is None else (max(1, i - w), min(L, i + w))
        for j in range(jlo, jhi + 1):
            c = (A[:, i - 1] - B[:, j - 1]) ** 2
            akt[:, j] = c + np.minimum(np.minimum(elozo[:, j], akt[:, j -
↪ 1]), elozo[:, j - 1])
        elozo = akt
    return np.sqrt(elozo[:, L])

rng = np.random.default_rng(0)
Xtr, ytr = cbf(10, rng); Xte, yte = cbf(100, rng)
znorm = lambda X: (X - X.mean(axis=1, keepdims=True)) / X.std(axis=1,
↪ keepdims=True)
Xtr, Xte = znorm(Xtr), znorm(Xte)
ii, jj = np.meshgrid(np.arange(len(Xte)), np.arange(len(Xtr)),
↪ indexing="ij")
tav = {"euklideszi": np.sqrt(((Xte[ii.ravel()] - Xtr[jj.ravel()]) **
↪ 2).sum(axis=1))}
for w in [4, 12, None]:
    tav[f"DTW, w = {w if w else 'nincs korlát'}"] =
↪ dtw_parok(Xte[ii.ravel()], Xtr[jj.ravel()], w=w)
pd.Series({nev: (ytr[d.reshape(len(Xte), len(Xtr)).argmin(axis=1)] !=
↪ yte).mean() for nev, d in tav.items()},
          name="1-NN tévesztési arány").round(3).to_frame()
```

	1-NN tévesztési arány
euklideszi	0.177
DTW, w = 4	0.067
DTW, w = 12	0.010
DTW, w = nincs korlát	0.003

```
fig, ax = plt.subplots(1, 3, sharey=True)
for c, nev in enumerate(["henger", "harang", "tölcsér"]):
    for k in range(3):
        ax[c].plot(Xtr[ytr == c][k], color=ab.PALETTE[c], lw=0.9, alpha=0.9)
    ax[c].set_title(nev, fontsize=9); ax[c].set_xlabel("idő")
fig.tight_layout()
```

Osztályonként mindössze tíz tanítómintával az euklideszi 1-NN osztályozó a tesztminták mintegy hatodát téveszti el, a DTW-vel a tévesztés néhány százalékra, korlátozás nélküli vetemítéssel szinte nullára csökken – összhangban



12.8. ábra. A Cylinder–Bell–Funnel feladat osztályonként három-három mintája. Az osztályon belüli eltérés főként időbeli (kezdet, hossz), amit az euklideszi távolság nagy különbségként érzékel, a DTW viszont kiegyenlít.

az UCR-archívumban közölt eredményekkel. A Sakoe–Chiba-sáv szélessége itt kompromisszum: a keskeny sáv gyorsabb, de túl kevés vetemítést enged. A futó üzemi esetben a DTW nem hozna ilyen javulást: ott a hibák nem eltolt jelalakokban, hanem szintekben és ingadozásokban különböznek – a távolságmértéket mindig a feladat jellegéhez kell választani.

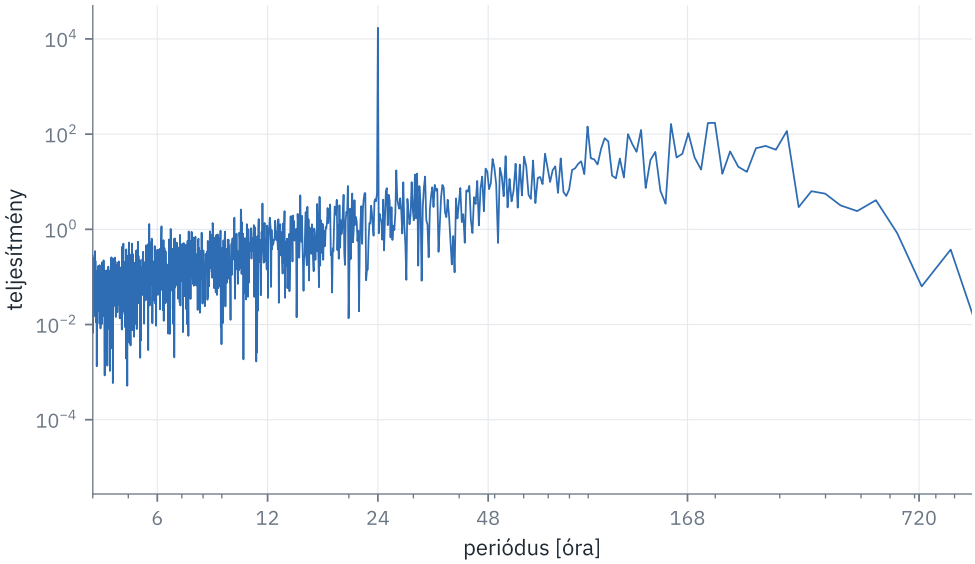
12.5.4. Periodicitás és tömörítés

Végül nézzük meg, mit árul el a periodogram a környezeti hőmérsékletről, és mennyire jól őrzik meg a különböző reprezentációk a hűtővízáram egy hetes sorát azonos számú (24) együtthatóval!

```
sor = ora.T_kornyezet["2025-04-01":"2025-09-30"]
x = (sor - sor.rolling(168, center=True, min_periods=1).mean()).to_numpy()
 ↪  # a heti mozgóátlag (lassú változás) levonása
X = np.fft.rfft(x)
freq = np.fft.rfftfreq(len(x), d=1.0)                  # ciklus / óra
P = np.abs(X[1:]) ** 2 / len(x)
fig, ax = plt.subplots()
```

```
ax.semilogy(1 / freq[1:], P, color=ab.PALETTE[0], lw=0.8)
ax.set_xscale("log"); ax.set_xlim(4, 1000)
ax.set_xticks([6, 12, 24, 48, 168, 720], ["6", "12", "24", "48", "168",
 ↪  "720"])
ax.set_xlabel("periódus [óra]"); ax.set_ylabel("teljesítmény")
fig.tight_layout()
print(f"a legerősebb periódus: {1 / freq[1:][np.argmax(P)]:.1f} óra")
```

```
a legerősebb periódus: 24.0 óra
```



12.9. ábra. A környezeti hőmérséklet órás sorának periodogramja (logaritmikus skálán) a periódusidő függvényében, a heti mozgóátlag levonása után. A legerősebb csúcs a 24 órás napi ciklus.

```
import pywt
het = ora.F_hutoviz["2025-10-20":"2025-10-26"].to_numpy()           # 168
 ↪  óra
k = 24
z = (het - het.mean()) / het.std()
paa = np.repeat(z.reshape(k, -1).mean(axis=1), len(z) // k)         # 24
 ↪  szakasz, egyenként 7 óra
# DFT: a 12 legkisebb frekvenciájú komplex együttható = 24 valós szám
F = np.fft.rfft(z); F[12:] = 0; dft = np.fft.irfft(F, n=len(z))
# Haar-DWT: a 24 legnagyobb abszolút értékű együttható
egyutt = pywt.wavedec(z, "haar")
```

```
lapos, szerk = pywt.coeffs_to_array(egyutt)
kuszob = np.sort(np.abs(lapos))[-k]
dwt = pywt.waverec(pywt.array_to_coeffs(np.where(np.abs(lapos) >= kuszob,
↪  lapos, 0), szerk, output_format="wavedec"), "haar")[:len(z)]
rmse = lambda r: float(np.sqrt(np.mean((z - r) ** 2)))
pd.Series({"PAA (24 szakasz)": rmse(paa), "DFT (12 komplex = 24 valós
↪  együttható)": rmse(dft),
           "Haar-DWT (24 legnagyobb együttható)": rmse(dwt)}, name="RMSE
           ↪  (z-normalizált)").round(3).to_frame()
```

	RMSE (z-normalizált)
PAA (24 szakasz)	0.342
DFT (12 komplex = 24 valós együttható)	0.221
Haar-DWT (24 legnagyobb együttható)	0.241

Ezen a heti soron a DFT a legpontosabb, mert a jelet a sima napi ciklus uralja, amelyet néhány alacsony frekvenciás szinusz jól leír. A Haar-hullámtranszformáció a legnagyobb együtthatók megtartásával a lokális változásokhoz alkalmazkodik, ezért hirtelen szintugrásokat és tranzienseket tartalmazó soroknál kerül előnybe; az egyenletes felbontású PAA a legkevésbé pontos. A PAA és a SAX előnye az egyszerűség és az, hogy a távolságuk alsó korlátja az eredeti sorok euklideszi távolságának, ami gyors indexelést tesz lehetővé (Keogh és mtsai., 2001; Lin és mtsai., 2007).

Interaktív alkalmazás: Dinamikus idővetemítés: illesztés, költségmátrix és Sakoe–Chiba-sáv

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/dtw.html

Colab-notebook: 12. fejezet – idősorok

12.6. Összefoglalás

- Az idősorok trendre, szezonalitásra és maradékra bonthatók; az STL robusztus, rugalmas felbontást ad.
- Az autokorreláció az idősorok alapvető jellemzője: az előrejelzés erre épül, a független mintavételt feltételező módszereket (véletlen keresztvalidáció, ellenőrzőkártyák) viszont félrevezeti.
- Az exponenciális simítás és az (S)ARIMA-modellek a klasszikus előrejelzési eszközök; a külső változókat a dinamikus regresszió (SARIMAX) használja fel.

- Az előrejelzéseket gördülő kezdőpontú értékeléssel, skálázott hibamértékkel (MASE) és egyszerű viszonyítási módszerekhez mérten kell értékelni.
- A dinamikus idővetemítés az időbeli eltolódásokat és sebességkülönbségeket kiegyenlítő távolságmérték; a Sakoe–Chiba-sáv és az alsó korlátok gyorsítják.
- A DFT, a DWT, a PAA és a SAX tömör reprezentációk; a SAX szimbolikus sorozattá alakítja az idősort.
- Az idősor-osztályozás erős viszonyítási alapja az 1-NN DTW; a szegmentálás homogén szakaszokra bontja a sort.

Ellenőrző kérdések

1. Mikor alkalmazunk multiplikatív felbontást, és hogyan vezethető vissza additívra?
2. Mit jelez, ha az ACF 24, 48, 72 óra késleltetésnél csúcsokat mutat?
3. Miért nem alkalmas a véletlen keresztvalidáció idősoros előrejelzők értékelésére?
4. Mit jelent, ha egy módszer MASE-értéke 1-nél nagyobb?
5. Mi a különbség a Holt–Winters-módszer additív és multiplikatív változata között?
6. Miért lehet a DTW-távolság nulla két különböző hosszú sor között?
7. Hogyan függ a SAX-reprezentáció az ábécé méretétől és a szakaszok számától?

Feladatok

1. **Kézi DTW.** Számítsa ki kézzel az $\mathbf{a} = (1, 2, 3)$ és a $\mathbf{b} = (1, 1, 2, 3)$ sorok DTW-távolságát és a vetemítési utat! Ellenőrizze a `dtw()` függvénnyel!
2. **Heti előrejelzés.** Bővítse a heti rendelésszám előrejelzését egy olyan változattal, amely a szezonális naiv előrejelzést az akciós hetekre külön korrigálja! Javul-e a MASE?
3. **SARIMA-rendválasztás.** Válassza meg a hűtési igény SARIMA-modelljének rendjét AIC alapján egy kis rácson ($p, q \in \{0, 1, 2\}$, $Q \in \{0, 1\}$), és vizsgálja meg a maradékok ACF-jét!
4. **Előrejelzési intervallum.** Ábrázolja a SARIMAX-modell 24 órás előrejelzését 95%-os előrejelzési intervallummal (`get_forecast().conf_int()`) egy kiválasztott napra! Hány mért érték esik az intervallumon kívül a hét kezdőpont során?

5. **Szegmentálás.** Bontsa szakaszokra a reaktorhőmérséklet alapjelének és a hűtővízáram órás sorának együttesét egy egyszerű alulról felfelé haladó algoritmussal (szakaszonként konstans közelítés)! Mennyire esnek egybe a szakaszhatárok a termékváltásokkal?

További olvasmányok

Az előrejelzés módszereinek kiváló, ingyenesen elérhető tankönyve Hyndman & Athanasopoulos (2021); az ARIMA-modellek klasszikus műve Box és mtsai. (2015). Az idősor-osztályozás módszereit Bagnall és mtsai. (2017), a mélytanulási megközelítéseket Ismail Fawaz és mtsai. (2019) tekinti át. A DTW gyors számításáról Keogh & Ratanamahatana (2005) és Rakthanmanon és mtsai. (2012), az idősor-reprezentációkról Lin és mtsai. (2007) cikke ajánlott; a szegmentálás módszereit Keogh és mtsai. (2004) foglalja össze.

13. Szövegbányászat

Miről szól a fejezet?

A szervezetek adatainak jelentős része szöveg: vásárlói vélemények, panaszlevelek, műszaknaplók, karbantartási jegyek, jelentések. A szövegbányászat ezekből nyer ki strukturált, elemezhető információt. A fejezet a szöveg előfeldolgozásával kezdődik – különös tekintettel a magyar nyelv sajátosságaira –, majd a vektortér-modellt és a TF-IDF súlyozást, a szövegosztályozást, a szövegklaszterezést és a témamodelleket (LSA, NMF, LDA), a disztribúciós szóbeágyazásokat, végül a kulcsszó- és mondatkinyerést tárgyalja. A nagy nyelvi modellek a könyv keretein kívül esnek; az itt bemutatott módszerek azonban ma is a gyors, átlátható és olcsó szövegelemzés alapjai, és a nyelvi modellek értékelésének viszonyítási pontjai.

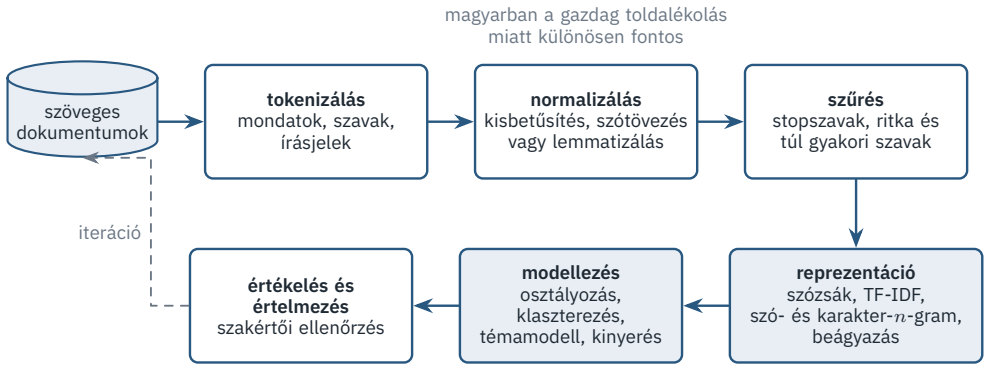
A fejezet végére az olvasó képes lesz egy magyar nyelvű szövegkorpuszt előfeldolgozni; a dokumentum–szó mátrixot és a TF-IDF súlyokat kézzel és programmal kiszámítani; szövegosztályozót építeni és értékelni; témamodellt illeszteni és értelmezni; és megítélni, mikor érdemes szó-, illetve karakterszintű reprezentációt használni.

13.1. A szöveg mint adat

A szövegbányászatban a vizsgált egység a **dokumentum** (egy vélemény, egy naplóbejegyzés, egy e-mail), a dokumentumok összessége a **korpusz**. A dokumentumokat elemi egységekre, **tokenekre** bontjuk – rendszerint szavakra, néha szórészletekre vagy karakterekre –, és a modellek ezek előfordulásaiból dolgoznak (13.1. ábra) (Manning és mtsai., 2008).

Az előfeldolgozás tipikus lépései:

- **Tokenizálás**: a szöveg mondatokra és szavakra bontása; az írásjelek, számok, rövidítések kezelése.
- **Normalizálás**: kisbetűsítés, a szóalakok közös alakra hozása. A **szótövezés** (stemming) szabályokkal levágja a toldalékokat (az angol nyelvre a Porter-algoritmus (Porter, 1980) a klasszikus); a **lemmatizálás** morfológiai elemzéssel a szótári alakot állítja elő.



13.1. ábra. A szövegbányászat folyamata. Az előfeldolgozás lépései (tokenizálás, normalizálás, szűrés) a nyelvtől és a feladattól függenek; a reprezentáció megválasztása legalább annyira meghatározza az eredményt, mint a modell.

- **Szűrés**: a tartalmilag üres, nagyon gyakori **stopszavak** (névelők, kötőszavak) és a nagyon ritka szavak elhagyása. A feladattól függ, mi számít stopszónak: hangulatelemzésben a „nem” tagadószó például kulcsfontosságú.

A **magyar nyelv** agglutináló: a szavakhoz sok toldalék kapcsolódhat (*kávéfőzőnkkel*, *szállításról*, *megérkezett*), és a szóösszetétel is gyakori. Ugyanannak a szónak ezért sok alakja fordul elő, ami a szóalapú reprezentációkat ritkábbá és zajosabbá teszi, mint angol nyelvű szövegeknél. A magyar nyelvre a HuSpaCy (Orosz és mtsai., 2022; Orosz és mtsai., 2023) ad korszerű, nyílt forrású tokenizáló, lemmatizáló és szófaji címkéző eszközláncot. Egyszerű, nyelvfüggetlen alternatíva a **karakter-n-gramok** használata: a szavakat átfedő, például 3–5 karakteres részletekre bontjuk (*szállítás* → *szá, zál, áll, llí, lít, ítá, tás, …*), így a közös tövű szóalakok sok közös jellemzőn osztoznak, és az elírások, ékezethiányok, rövidítések sem okoznak teljes egyezéshiányt (Cavnar & Trenkle, 1994).

A szavak gyakoriságeloszlása rendkívül ferde: a **Zipf-törvény** (Zipf, 1949) szerint a r-edik leggyakoribb szó gyakorisága közelítőleg $1/r$-rel arányos. Néhány szó nagyon gyakori, a szókincs túlnyomó része viszont csak egyszer-kétszer fordul elő – ez a szöveges adatok ritka (sparse) szerkezetének oka.

13.2. A vektortér-modell

13.2.1. Szózsák és TF-IDF

A legtöbb klasszikus módszer a **szózsák** (bag of words) modellre épül: a dokumentumot a szavai előfordulási számának vektorával írjuk le, a szavak sorrend-

jét figyelmen kívül hagyva. N dokumentum és n szavas szókincs esetén ez az $N \times n$-es **dokumentum–szó mátrix**, amelynek elemei a $\mathrm{tf}_{t,d}$ gyakoriságok. A mátrix rendkívül ritka: egy rövid vélemény a több ezer szavas szókincsből csak néhány szót tartalmaz, ezért ritka mátrixként tároljuk.

A nyers gyakoriság a mindenütt előforduló szavakat túlsúlyozza. A **TF-IDF** súlyozás (Salton & Buckley, 1988; Spärck Jones, 1972) a szó dokumentumon belüli gyakoriságát azzal szorozza, mennyire ritka a szó a korpuszban:

$$w_{t,d} = \mathrm{tf}_{t,d} \cdot \mathrm{idf}_t, \qquad \mathrm{idf}_t = \ln \frac{N}{\mathrm{df}_t}, \tag{13.1}$$

ahol df_t azon dokumentumok száma, amelyekben a t szó előfordul. A minden dokumentumban szereplő szó súlya nulla, a csak egy-két dokumentumban előfordulóé nagy. A gyakorlatban számos változat létezik: logaritmikus tf ($1 + \ln \mathrm{tf}$), simított idf (a `scikit-learn` alapértelmezése $\ln \frac{1+N}{1+\mathrm{df}_t} + 1$), és a dokumentumvektorok egységnyi hosszra normálása.

13.2.2. Koszinusz-hasonlóság

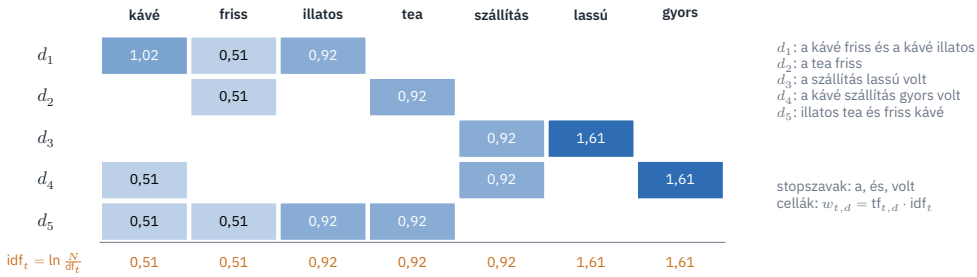
Két dokumentum hasonlóságát a vektoraik által bezárt szög koszinuszával mérjük:

$$\cos(\mathbf{d}_i, \mathbf{d}_j) = \frac{\mathbf{d}_i^{\top} \mathbf{d}_j}{\|\mathbf{d}_i\| \, \|\mathbf{d}_j\|}. \tag{13.2}$$

A koszinusz-hasonlóság független a dokumentumok hosszától (egy kétszer olyan hosszú, ugyanolyan arányú szavakat tartalmazó dokumentum vektora azonos irányú), és nemnegatív súlyoknál 0 és 1 közé esik. Egységnyi hosszra normált vektoroknál a koszinusz-hasonlóság a skaláris szorzat, az euklideszi távolság négyzete pedig $2(1 - \cos)$, így a k-közép és más euklideszi módszerek is alkalmazhatók.

13.2.3. Kézi példa

A segédcsomag öt rövid dokumentumának dokumentum–szó mátrixa a stopszavak (*a, és, volt*) elhagyása után a 13.2. ábrán látható. A *kávé* szó három dokumentumban fordul elő, így $\mathrm{idf} = \ln(5/3) \approx 0{,}51$; az első dokumentumban kétszer szerepel, súlya $2 \cdot 0{,}51 = 1{,}02$. A csak egyszer előforduló *lassú* és *gyors* idf-je $\ln 5 \approx 1{,}61$ – ezek a szavak jellemzik leginkább a saját dokumentumukat.



13.2. ábra. A kézi példa TF-IDF súlyozású dokumentum–szó mátrixa. A cellák színe a súllyal arányos; az alsó sor az idf-értékeket mutatja.

```
docs = ab.mini.szoveg_mini()
stop = {"a", "és", "volt"}
tokenek = [[w for w in d.split() if w not in stop] for d in docs]
szokincs = ["kávé", "friss", "illatos", "tea", "szállítás", "lassú",
 ↪  "gyors"]
tf = np.array([[t.count(w) for w in szokincs] for t in tokenek],
 ↪  dtype=float)
idf = np.log(len(docs) / (tf > 0).sum(axis=0))
W = tf * idf
normalt = W / np.linalg.norm(W, axis=1, keepdims=True)
cos = normalt @ normalt.T
pd.DataFrame(cos, index=[f"d{i+1}" for i in range(5)], columns=[f"d{i+1}"
 ↪  for i in range(5)]).round(3)
```

	d1	d2	d3	d4	d5
d1	1.000	0.170	0.000	0.186	0.747
d2	0.170	1.000	0.000	0.000	0.707
d3	0.000	0.000	1.000	0.236	0.000
d4	0.186	0.000	0.236	1.000	0.092
d5	0.747	0.707	0.000	0.092	1.000

A legnagyobb hasonlóság az első és az ötödik dokumentum között van (mindkettő friss, illatos kávéról szól), a harmadik és a negyedik dokumentumot a *szállítás* szó köti össze, a második és a harmadik dokumentumnak pedig nincs közös szava.

13.3. Szövegosztályozás

A szövegosztályozás – hangulatelemzés, levélszemét-szűrés, panaszok irányítása, naplóbejegyzések hibatípus szerinti besorolása – a szöveges adatok egyik leggya-

koribb felhasználása. A TF-IDF vektorok nagy dimenziósak és ritkák, a dokumentumok száma ehhez képest kicsi; ilyen helyzetben a lineáris modellek jól teljesítenek, és gyorsan taníthatók (Joachims, 1998; Manning és mtsai., 2008).

A **multinomiális naiv Bayes** (McCallum & Nigam, 1998) a dokumentumot a szavak független húzásainak sorozataként modellezi. A 9.4. alfejezet osztályozójának ez a szövegekre szabott változata:

$$\hat{c} = \arg\max_c \left[\ln P(c) + \sum_t \mathrm{tf}_{t,d} \ln P(t \mid c)\right], \qquad P(t \mid c) = \frac{N_{t,c} + 1}{\sum_{t'} N_{t',c} + n}, \tag{13.3}$$

ahol $N_{t,c}$ a t szó előfordulásainak száma a c osztály tanítódokumentumaiban, és a nevezőben a Laplace-simítás jelenik meg. A logisztikus regresszió (9.2. alfejezet) és a lineáris SVM (9.6. alfejezet) a TF-IDF jellemzőkön rendszerint pontosabb, és az együtthatóik közvetlenül megmutatják, mely szavak szólnak egy-egy osztály mellett vagy ellen.

13.4. Szövegklaszterezés és témamodellek

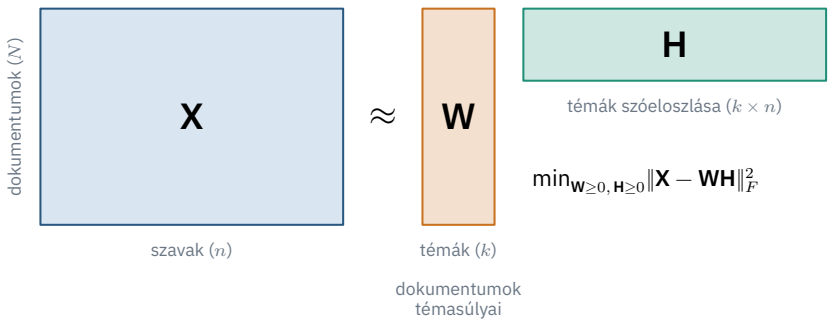
A címkék nélküli korpuszok szerkezetének feltárására két megközelítés terjedt el. A **szövegklaszterezés** minden dokumentumot egyetlen csoportba sorol, például a normált TF-IDF vektorok k-közép klaszterezésével. A **témamodellek** ezzel szemben minden dokumentumot több téma keverékeként írnak le, ahol a téma a szavak egy eloszlása – ez a vegyes tartalmú szövegeknél (egy vélemény a termékről *és* a szállításról is szólhat) természetesebb.

A **látens szemantikus elemzés** (LSA) (Deerwester és mtsai., 1990) a TF-IDF mátrix csonkolt szinguláris érték szerinti felbontása (B. függelék): $\mathbf{X} \approx \mathbf{U}_k \boldsymbol{\Sigma}_k \mathbf{V}_k^\top$. A k dimenziós látens térben az együtt előforduló szavak – és az ilyen szavakat tartalmazó dokumentumok – közel kerülnek egymáshoz, ami a szinonímia hatását is enyhíti. A látens dimenziók azonban negatív súlyokat is tartalmaznak, ezért nehezen értelmezhetők.

A **nemnegatív mátrixfaktorizáció** (NMF) (D. D. Lee & Seung, 1999) a nemnegatív $\mathbf{X}$ mátrixot két nemnegatív mátrix szorzatára bontja (13.3. ábra):

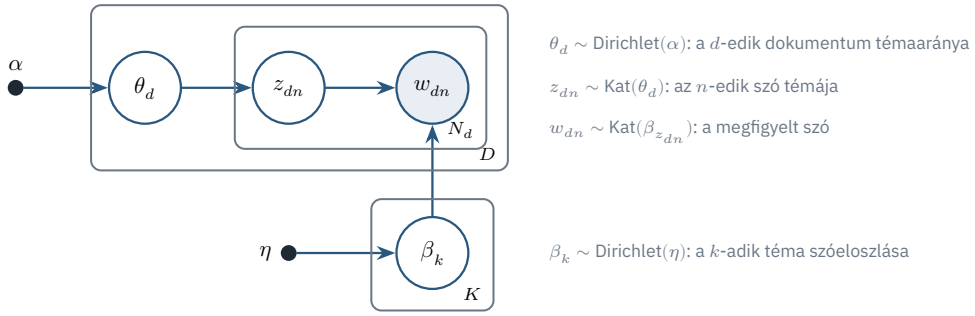
$$\min_{\mathbf{W}\geq 0,\ \mathbf{H}\geq 0} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2. \tag{13.4}$$

A nemnegativitás miatt a dokumentumok a témák *additív* keverékei, és a $\mathbf{H}$ sorai – a témák szósúlyai – közvetlenül értelmezhetők: egy téma a legnagyobb súlyú szavaival jellemezhető.



13.3. ábra. A nemnegatív mátrixfaktorizáció: a dokumentum–szó mátrix a dokumentumok témasúlyainak (**W**) és a témák szósúlyainak (**H**) szorzataként közelíthető.

A **látens Dirichlet-allokáció** (LDA) (Blei és mtsai., 2003) generatív valószínűségi modell (13.4. ábra). Feltételezi, hogy minden téma egy $\boldsymbol{\beta}_k$ szóeloszlás, minden dokumentum egy $\boldsymbol{\theta}_d$ témaeloszlás, és a dokumentum minden szavát úgy „írjuk le”, hogy előbb a $\boldsymbol{\theta}_d$ szerint témát, majd a téma szóeloszlása szerint szót húzunk. A megfigyelt szavakból a rejtett téma- és szóeloszlásokat variációs Bayes-módszerrel vagy Gibbs-mintavétellel becsüljük. A témák számát (k) mindhárom módszernél nekünk kell megadnunk; a választást a témák értelmezhetősége, a visszatartott dokumentumokon mért illeszkedés és a felhasználási cél együtt határozza meg.



13.4. ábra. A látens Dirichlet-allokáció grafikus modellje (lemezjelöléssel). A satírozott csúcs a megfigyelt szó; a lemezek a dokumentumokon (D), a dokumentumok szavain (N_d) és a témákon (K) való ismétlést jelölik.

13.5. Szóbeágyazások

A szózsákmodellben minden szó külön dimenzió: a *drága* és a *túlárazott* ugyanolyan távol van egymástól, mint a *drága* és a *futár*. A **disztribúciós hipotézis** (Z. S. Harris, 1954) szerint a hasonló jelentésű szavak hasonló környezetben fordulnak elő; erre épülnek a **szóbeágyazások**, amelyek minden szót egy sűrű, néhány száz dimenziós vektorral írnak le úgy, hogy a hasonló környezetű szavak vektorai közel legyenek.

Klasszikus változatuk a szó–környezet együttelőfordulási mátrixból indul. A **pozitív pontonkénti kölcsönös információ** (PPMI) azt méri, mennyivel gyakrabban fordul elő együtt két szó, mint függetlenség esetén:

$$\mathrm{PPMI}(w, c) = \max\!\left(0,\ \ln \frac{P(w, c)}{P(w)\,P(c)}\right), \tag{13.5}$$

és a PPMI-mátrix csonkolt SVD-je adja a beágyazást. A **word2vec** (Mikolov és mtsai., 2013) egy sekély neurális hálózattal, a szavak környezetének előrejelzésével tanulja a vektorokat; Levy & Goldberg (2014) megmutatta, hogy negatív mintavétellel ez lényegében egy eltolt PMI-mátrix implicit faktorizációja – a két megközelítés tehát közeli rokon. A szóbeágyazások nagy korpuszt igényelnek; előre tanított magyar vektorok nyilvánosan elérhetők, és a HuSpaCy modelljei is tartalmaznak ilyeneket. A szövegkörnyezettől függő (kontextuális) reprezentációk – a transzformer alapú nyelvi modellek – már túlmutatnak a könyv keretein.

13.6. Kulcsszavak és összefoglalás

A **kulcsszó-kinyerés** legegyszerűbb módja a dokumentum (vagy dokumentumcsoport) legnagyobb TF-IDF súlyú szavainak kiválasztása. A **TextRank** (Mihalcea & Tarau, 2004) gráfalapú módszer: a szavakat (kulcsszó-kinyeréshez) vagy a mondatokat (kivonatoló összefoglaláshoz) egy gráf csúcsaiként, a hasonlóságukat élsúlyként kezeli, és a csúcsok fontosságát a PageRank-algoritmushoz hasonlóan, iteratívan számolja:

$$S(v_i) = (1-d) + d \sum_{v_j \in \mathrm{Be}(v_i)} \frac{w_{ji}}{\sum_{v_k \in \mathrm{Ki}(v_j)} w_{jk}}\, S(v_j), \qquad d = 0{,}85. \tag{13.6}$$

Egy mondat annál fontosabb, minél több más fontos mondathoz hasonlít; a legmagasabb pontszámú mondatok együtt a szöveg(csoport) „legjellemzőbb” kivonatát adják. A kivonatoló összefoglalás mindig a forrás mondatait adja vissza, ezért nem „talál ki” új állításokat – ez a megbízhatóság szempontjából előny.

13.7. Gyakorlat: vásárlói vélemények és műszaknapló

13.7.1. A vélemények előfeldolgozása és szókincse

FUTÓ ESET · PANNONSHOP

A PannonShop 3000 vásárlói véleményt gyűjtött; mindegyikhez csillagos értékelés is tartozik, amelyből hangulatcímkét képzünk (4–5 csillag: pozitív, 3: semleges, 1–2: negatív). A cél kettős: egyrészt egy osztályozó, amely a csillag nélküli szöveges visszajelzéseket (e-maileket, közösségi bejegyzéseket) is hangulat szerint tudja besorolni, másrészt annak feltárása, miről szólnak a vélemények.

```
r = ab.load_shop().reviews
print(r.Hangulat.value_counts().to_dict())
r[["Csillag", "Szoveg"]].sample(5, random_state=3)
```

```
{'pozitív': 1781, 'semleges': 633, 'negatív': 586}
```

	Csillag	Szoveg
1283	2	Rossz minőségű az adventi koszorú. Csalódott vagyok.
250	2	A fazékkészlet egyáltalán nem olyan, mint a leírásban. Túl drága ahhoz képest, amit ny…
2153	4	Átlagos kávébab 1 kg, megfelel a célnak. Másnapra megérkezett a csomag. Öt csillag.
2786	4	A zöld tea 50 filter pontosan olyan, mint a képen. Az ára elfogadható. Biztosan rendel…
632	3	A fekete tea 50 filter két hét után elromlott. Az ügyfélszolgálat gyorsan és kedvesen …

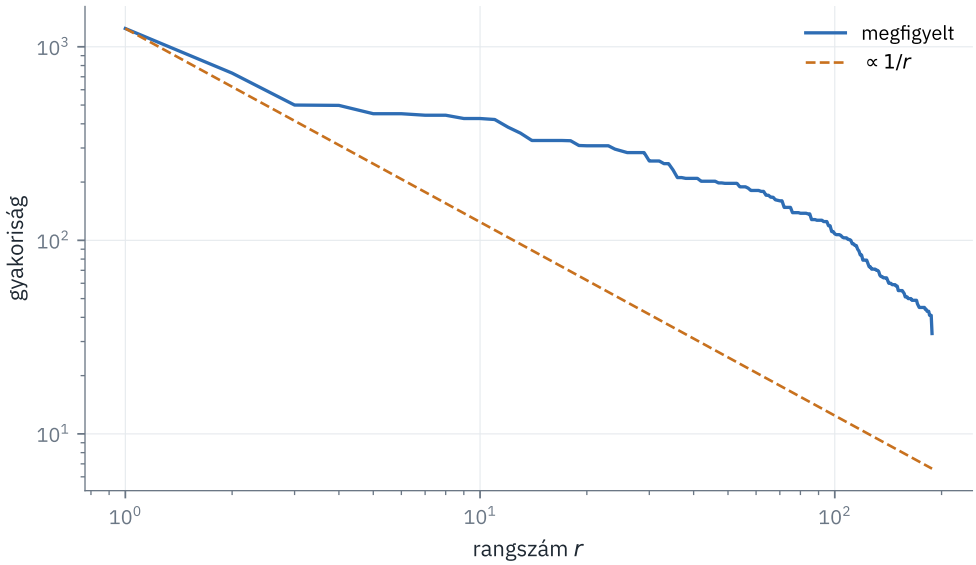
```
STOP = set("a az és is de hogy volt vagyok mint egy meg már csak még ez azt
↪  ami amit".split())

def tokenizal(szoveg):
    """Kisbetűsítés, szavakra bontás és stopszószűrés (a "nem" tagadószót
    ↪  megtartjuk)."""
    return [w for w in re.findall(r"[a-záéíóöőúüű]+", szoveg.lower()) if w
    ↪  not in STOP and len(w) > 1]

from collections import Counter
gyak = Counter(w for t in r.Szoveg for w in tokenizal(t))
print(f"{sum(gyak.values())} szóelőfordulás, {len(gyak)} különböző szó (a
↪  szimulált korpusz sablonos, ezért szűk a szókincse); "
      f"a 10 leggyakoribb: {', '.join(w for w, _ in gyak.most_common(10))}")
```

```
29544 szóelőfordulás, 188 különböző szó (a szimulált korpusz sablonos, ezért szűk
↪  a szókincse); a 10 leggyakoribb: nem, szállítás, minőségű, rendben, biztosan,
↪  rendelek, öt, csillag, ajánlani, tudom
```

```
f = np.array(sorted(gyak.values(), reverse=True))
rang = np.arange(1, len(f) + 1)
fig, ax = plt.subplots()
ax.loglog(rang, f, color=ab.PALETTE[0], lw=1.5, label="megfigyelt")
ax.loglog(rang, f[0] / rang, color=ab.PALETTE[1], ls="--", lw=1.2,
 ↪  label="$\\propto 1/r$")
ax.set_xlabel("rangszám $r$"); ax.set_ylabel("gyakoriság"); ax.legend()
fig.tight_layout()
```



13.5. ábra. A vélemények szógyakoriságai a rangszám függvényében, log–log skálán. A szaggatott vonal az $1/r$-rel arányos Zipf-törvény. A sablonos, kis szókincsű korpuszban az eloszlás vége gyorsabban esik, mint a természetes szövegekben.

13.7.2. Hangulatosztályozás

Három reprezentációt (szó-unigramok, szó-uni- és bigramok, karakter-3–5-gramok) és három osztályozót (multinomiális naiv Bayes, logisztikus regresszió, lineáris SVM) hasonlítunk össze rétegzett ötszörös keresztvalidációval. Mivel a három osztály aránya eltérő, a makró-F1 értéket (az osztályonkénti F1 értékek átlagát) használjuk.

```
from sklearn.feature_extraction.text import TfidfVectorizer,
↪  CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import StratifiedKFold, cross_val_score

X, y = r.Szoveg, r.Hangulat
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
reprezentaciok = {
    "szó (1)": lambda: TfidfVectorizer(tokenizer=tokenizal,
    ↪  token_pattern=None, min_df=2),
    "szó (1–2)": lambda: TfidfVectorizer(tokenizer=tokenizal,
    ↪  token_pattern=None, ngram_range=(1, 2), min_df=2),
    "karakter (3–5)": lambda: TfidfVectorizer(analyzer="char_wb",
    ↪  ngram_range=(3, 5), min_df=2),
}
osztalyozok = {"naiv Bayes": lambda: MultinomialNB(),
               "logisztikus regresszió": lambda:
               ↪  LogisticRegression(max_iter=3000, C=5),
               "lineáris SVM": lambda: LinearSVC(C=0.5)}
pd.DataFrame({onev: {rnev: cross_val_score(make_pipeline(rep(), osz()), X,
↪  y, cv=cv, scoring="f1_macro").mean()
                     for rnev, rep in reprezentaciok.items()} for onev, osz
                     ↪  in osztalyozok.items()}).round(3)
```

	naiv Bayes	logisztikus regresszió	lineáris SVM
szó (1)	0.854	0.903	0.896
szó (1–2)	0.858	0.893	0.888
karakter (3–5)	0.862	0.902	0.898

A lineáris modellek a naiv Bayesnél pontosabbak, a három reprezentáció között viszont nincs érdemi különbség: ebben a gondosan megfogalmazott, helyesen írt korpuszban a szóalapú reprezentáció is jól működik. Melyik osztályt tévesztik leginkább, és mely szavak alapján dönt a modell?

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.3, stratify=y,
↪  random_state=0)
vec = TfidfVectorizer(tokenizer=tokenizal, token_pattern=None, min_df=2)
lr = LogisticRegression(max_iter=3000, C=5).fit(vec.fit_transform(Xtr),
↪  ytr)
print(classification_report(yte, lr.predict(vec.transform(Xte)),
↪  digits=3))
```

```
szavak = np.array(vec.get_feature_names_out())
pd.DataFrame({c: szavak[np.argsort(-lr.coef_[i])[:8]] for i, c in
↪  enumerate(lr.classes_)})
```

```
              precision    recall  f1-score   support

     negatív      0.930     0.903     0.916       176
     pozitív      0.951     0.976     0.963       534
    semleges      0.834     0.795     0.814       190

    accuracy                         0.923       900
   macro avg      0.905     0.891     0.898       900
weighted avg      0.922     0.923     0.922       900
```

	negatív	pozitív	semleges
0	csalódott	csillag	mennyire
1	ajánlom	öt	majd
2	csalódás	tudom	meglátjuk
3	soha	ajánlani	tartós
4	többet	biztosan	közepes
5	nem	rendelek	turmixgép
6	rossz	kiváló	vágódeszka
7	rendesen	hibátlanul	kenyérpirító

FUTÓ ESET · PANNONSHOP

A semleges osztály a legnehezebb: a háromcsillagos vélemények vegyes tartalmúak – dicsérik az árat, de panaszkodnak a szállításra –, és a csillagszám maga is zajos címke. A legfontosabb szavak jól értelmezhetők: a pozitív osztály mellett a lelkes zárómondatok és a minőséget dicsérő szavak, a negatív mellett a csalódást, a meghibásodást és a „nem ajánlom” típusú kifejezéseket jelző szavak szólnak. Az együtthatók vizsgálata egyben ellenőrzés is – és itt rögtön talál is valamit: a semleges osztály mellett terméknevek (*turmixgép*, *vágódeszka*, *kenyérpirító*) is nagy súlyt kapnak. Ezeknek a termékeknek a minősége közepes, ezért a véleményeik gyakrabban semlegesek; a modell tehát részben a termékre, nem a szöveg hangulatára tanult rá. Egy új termékcsaládra ez a tudás nem általánosítható, ezért a terméknevek kiszűrése vagy helyettesítése (például egy közös *TERMÉK* jelölővel) indokolt lehet.

13.7.3. Miről szólnak a vélemények? Témák mondatszinten

Egy vélemény jellemzően több szempontot érint (a termék minőségét, a szállítást, az árat, az ügyfélszolgálatot), ezért a témákat mondatszinten keressük, NMF-fel.

```
from sklearn.decomposition import NMF
mondatok = [m.strip() for t in r.Szoveg for m in re.split(r"(?<=[.!?])\s+",
 ↪  t) if m.strip()]
STOP_T = STOP - {"nem"}
tok_t = lambda s: [w for w in re.findall(r"[a-záéíóöőúüű]+", s.lower()) if
 ↪  w not in STOP_T and len(w) > 1]
vec_t = TfidfVectorizer(tokenizer=tok_t, token_pattern=None, min_df=5,
 ↪  max_df=0.3)
M = vec_t.fit_transform(mondatok)
nmf = NMF(n_components=8, init="nndsvda", random_state=0,
 ↪  max_iter=800).fit(M)
voc = np.array(vec_t.get_feature_names_out())
print(f"{len(mondatok)} mondat, {M.shape[1]} szó")
pd.DataFrame({f"{k + 1}. téma": voc[np.argsort(-c)[:7]] for k, c in
 ↪  enumerate(nmf.components_)}).T \
  .apply(lambda s: ", ".join(s), axis=1).rename("legnagyobb súlyú
 ↪  szavak").to_frame()
```

```
8422 mondat, 188 szó
```

	legnagyobb súlyú szavak
1. téma	biztosan, rendelek, minőségű, nagyon, kiváló, közepes, kedvező
2. téma	csillag, öt, ára, elfogadható, érkezett, gondos, csomagolásban
3. téma	ajánlani, tudom, ára, elfogadható, érkezett, gondos, csomagolásban
4. téma	szállítás, gyors, időn, belül, megígért, megtörtént, hetet
5. téma	nem, olcsó, drága, ajánlom, különleges, van, időben
6. téma	mennyire, majd, meglátjuk, tartós, érkezett, csomagolásban, gondos
7. téma	kihozta, futár, rendben, különleges, van, ki, jött
8. téma	csomag, megérkezett, másnapra, érkezett, gondos, csomagolásban, sérült

FUTÓ ESET · PANNONSHOP

A témák többsége egyértelműen azonosítható szemponthoz köthető: a szállítás gyorsasága és késése, a futár, a csomagolás és a sérült doboz, az ár („drága”, „olcsó”, „megéri”), a tartósság, illetve az ajánló vagy elutasító zárómondatok. Néhány téma keveredik – az NMF a gyakran együtt előforduló mondatrészleteket is egy témába vonhatja –, ezért a témák elnevezése és szükség esetén összevonása szakértői feladat. A témasúlyok (**W** mátrix) alapján ezután például az is követhető, hogyan változik a szállítási panaszok aránya hónapról hónapra.

13.7.4. Szóbeágyazás PPMI-mátrixból

```
from sklearn.preprocessing import normalize
cv1 = CountVectorizer(tokenizer=tok_t, token_pattern=None, min_df=5,
↪  binary=True)
B = cv1.fit_transform(mondatok)                    # mondat–szó előfordulás
↪  (a környezet: a mondat)
v = np.array(cv1.get_feature_names_out())
C = (B.T @ B).toarray().astype(float)
np.fill_diagonal(C, 0)
osszes = C.sum()
p_w = C.sum(axis=1) / osszes
with np.errstate(divide="ignore"):
    pmi = np.log((C / osszes) / np.outer(p_w, p_w))
ppmi = np.where(np.isfinite(pmi) & (pmi > 0), pmi, 0.0)
U, S, _ = np.linalg.svd(ppmi)
E = normalize(U[:, :30] * np.sqrt(S[:30]))         # 30 dimenziós beágyazás
def szomszedok(szo, k=5):
    i = int(np.flatnonzero(v == szo)[0])
    h = E @ E[i]
    legkozelebbi = [j for j in np.argsort(-h) if j != i][:k]
    return ", ".join(f"{v[j]} ({h[j]:.2f})" for j in legkozelebbi)
pd.Series({w: szomszedok(w) for w in ["szállítás", "drága", "kávéfőző",
↪  "futár"]},
          name="legközelebbi szavak (koszinusz)").to_frame()
```

	legközelebbi szavak (koszinusz)
szállítás	belül (1.00), hetet (1.00), időn (1.00), gyors (1.00), megígért (1.00)
drága	nyújt (0.98), képest (0.98), túl (0.98), ahhoz (0.98), olcsó (0.82)
kávéfőző	gép (0.96), kotyogós (0.96), fotóalbum (0.61), tollkészlet (0.55), jegyzettömb (0.42)
futár	időben (0.97), jött (0.97), ki (0.97), kihozta (0.90), rendben (0.50)

A beágyazás jól mutatja a disztribúciós reprezentációk két arcát. A *drága* legközelebbi szomszédai között az *olcsó* is ott van – az ellentétes jelentésű szavak hasonló környezetben fordulnak elő, ezért a disztribúciós hasonlóság nem azonos a szinonímiával –, a *kávéfőző* szomszédai pedig más terméknevek, vagyis a beágyazás a szavak „szerepét” is megtanulta. Másrészt ebben a kis, sablonos korpuszban az ugyanabban a mondatfordulatban szereplő szavak szinte azonos vektort kapnak: valódi jelentésbeli hasonlóságok tanulásához sokkal nagyobb és változatosabb korpusz kell.

13.7.5. A panaszok kivonatoló összefoglalása

```
import networkx as nx
negativ = [m for t in r.loc[r.Hangulat == "negatív", "Szoveg"] for m in
 ↪  re.split(r"(?<=[.!?])\s+", t) if m.strip()]
egyedi = pd.Series(negativ).value_counts()
mondat_lista = egyedi.index.tolist()
V = TfidfVectorizer(analyzer="char_wb", ngram_range=(3,
 ↪  5)).fit_transform(mondat_lista)
S_hasonl = (V @ V.T).toarray()
np.fill_diagonal(S_hasonl, 0)
G = nx.from_numpy_array(S_hasonl * egyedi.to_numpy()[None, :])   # a
 ↪  gyakori mondatok súlyosabbak
pr = nx.pagerank(G, alpha=0.85, weight="weight")
legjobb = sorted(pr, key=pr.get, reverse=True)[:5]
pd.DataFrame({"mondat": [mondat_lista[i] for i in legjobb], "előfordulás":
 ↪  [egyedi.iloc[i] for i in legjobb],
              "TextRank-pontszám": [round(pr[i], 4) for i in legjobb]})
```

	mondat	előfordulás	TextRank-pontszám
0	Csalódott vagyok.	162	0.0309
1	Nem ajánlom.	138	0.0205
2	A futár rendben kihozta.	60	0.0144
3	Nem olcsó, de nem is drága.	55	0.0130
4	Az asztali futó egyáltalán nem olyan, mint a leírásban.	11	0.0100

FUTÓ ESET · PANNONSHOP

A legmagasabb pontszámot az elutasító zárómondatok kapják – ezek a negatív vélemények közös, leggyakoribb elemei –, utánuk semleges szállítási és árral kapcsolatos mondatok, majd egy minőségi kifogás következik. A TextRank a legközpontibb, a többihez leginkább hasonló mondatokat emeli ki, amelyek nem feltétlenül a leginformatívabbak. A gyakorlatban ezért a kivonatot szűrik (például a zárómondatokat vagy a semleges hangulatú mondatokat kizárva), vagy témánként – a fenti NMF-témák szerint csoportosítva – készítik el. Mivel a módszer a forrás mondatait idézi, a kivonat minden állítása visszakereshető.

13.7.6. Szakmai szöveg: a műszaknapló

FUTÓ ESET · PANNON ÜZEM

A Pannon üzem kezelői minden műszakban rövid, szabad szöveges bejegyzéseket írnak a műszaknaplóba. A bejegyzések stílusa tipikus: rövidítések (HV = hűtővíz, betáp, konc.), elírások, hiányzó ékezetek. A szakmai szövegek (karbantartási jegyek, műszaknaplók) elemzése külön kihívás, mert a nyelvi eszközláncokat általános szövegekre fejlesztették; erre a problémára a szakirodalom a *technikai nyelvfeldolgozás* (technical language processing) elnevezést használja (Brundage és mtsai., 2021).

```
naplo = ab.load_shift_log()
print(f"{len(naplo)} bejegyzés; esemény szerinti megoszlás (csak az
↪  értékeléshez): {naplo.esemeny.value_counts().to_dict()}")
naplo[["ido", "muszak", "szoveg"]].sample(6, random_state=5)
```

```
426 bejegyzés; esemény szerinti megoszlás (csak az értékeléshez): {'rutin': 373,
↪  'szenzor_drift': 18, 'szelep_akadas': 12, 'betap_zavar': 8, 'hutoviz_kieses':
↪  8, 'tisztitas': 7}
```

	ido	muszak	szoveg
139	2025-05-03 13:34:00	délelőtt	napi bejárás megvolt, rendben
351	2025-10-29 15:56:00	délután	T mérés gyanús, nyomással nem stimmel
366	2025-11-08 13:18:00	délelőtt	szűrőcsere megtörtént
195	2025-06-12 13:19:00	délelőtt	kenés elvégezve a keverőn
53	2025-02-16 05:11:00	éjszaka	kevero rendben, zaj nincs
348	2025-10-28 16:57:00	délután	hőmérő elmászott? kézi méréssel eltér

Első feladatként keressük meg a szelep akadozására utaló bejegyzéseket egy rövid keresőkifejezés alapján! Az egyik változat szavakra, a másik karakter-n-gramokra épül.

```
kerdes = "szelep akad"
eredm = {}
for nev, vec in [("szó", TfidfVectorizer()), ("karakter (2–4)",
↪  TfidfVectorizer(analyzer="char_wb", ngram_range=(2, 4)))]:
    Xn = vec.fit_transform(naplo.szoveg)
    h = (Xn @ vec.transform([kerdes]).T).toarray().ravel()
    talalat = np.argsort(-h)[:10]
    eredm[nev] = {"találatok (első 10)": "
↪  ".join(naplo.esemeny.iloc[talalat].str.replace("szelep_akadas",
↪  "SZELEP")),
                  "pontosság@10": (naplo.esemeny.iloc[talalat] ==
                   ↪  "szelep_akadas").mean()}
```

```
pd.DataFrame(eredm).T
```

	találatok (első 10)	pontosság@10
szó	SZELEP, SZELEP, SZELEP, SZELEP, SZELEP, SZELEP, SZELEP, rutin, rutin, rutin	0.7
karakter (2–4)	SZELEP, SZELEP, SZELEP, SZELEP, SZELEP, SZELEP, SZELEP, SZELEP, SZELEP, SZELEP	1.0

Második feladatként az első félév bejegyzéseit – amelyeket utólag, a hibajegyek alapján eseménytípus szerint felcímkéztek – tanítóhalmaznak használjuk, és a második félév bejegyzéseit soroljuk be automatikusan.

```
from sklearn.metrics import f1_score
tanit = (naplo.ido < "2025-07-01").to_numpy()
sor = {}
for nev, vec in [("szó", TfidfVectorizer()), ("karakter (2–4)",
↪  TfidfVectorizer(analyzer="char_wb", ngram_range=(2, 4)))]:
    Xn = vec.fit_transform(naplo.szoveg)
    pred = LinearSVC(C=1.0, class_weight="balanced").fit(Xn[tanit],
↪   naplo.esemeny[tanit]).predict(Xn[~tanit])
    valos = naplo.esemeny[~tanit].to_numpy()
    sor[nev] = {"makró-F1": f1_score(valos, pred, average="macro")}
    sor[nev].update({c: np.mean(pred[valos == c] == c) for c in
↪   ["hutoviz_kieses", "szelep_akadas", "betap_zavar", "szenzor_drift",
↪   "tisztitas"]})
pd.DataFrame(sor).T.round(2).rename_axis("2. félév (felidézés)")
```

2. félév (felidézés)	makró-F1	hutoviz_kieses	szelep_akadas	betap_zavar	szenzor_drift	tisztitas
szó	0.64	0.2	0.5	0.25	0.50	0.75
karakter (2–4)	0.81	0.2	1.0	0.75	0.75	1.00

Végül vizsgáljuk meg, felismerhetők-e címkék nélkül az eseménytípusok: a rutin-bejegyzések kiszűrése után klaszterezzük a rendkívüli bejegyzéseket!

```
from sklearn.cluster import KMeans
from sklearn.metrics import adjusted_rand_score
rendkivuli = naplo[naplo.esemeny != "rutin"].reset_index(drop=True)
sor = {}
for nev, vec in [("szó", TfidfVectorizer()), ("karakter (3–5)",
↪  TfidfVectorizer(analyzer="char_wb", ngram_range=(3, 5)))]:
    Xr = vec.fit_transform(rendkivuli.szoveg)
    cimke = KMeans(n_clusters=5, n_init=50, random_state=0).fit_predict(Xr)
    sor[nev] = adjusted_rand_score(rendkivuli.esemeny, cimke)
pd.Series(sor, name="korrigált Rand-index").round(3).to_frame()
```

	korrigált Rand-index
szó	0.079
karakter (3–5)	0.437

FUTÓ ESET · PANNON ÜZEM

A szakmai, „zajos” szövegeken a karakter-n-gramok előnye egyértelmű: a keresés és az osztályozás is jobb velük, mert az ékezet nélküli és a rövidített alakok (*hutoviz – hűtővíz*, *betap – betáplálás*) is sok közös jellemzőn osztoznak. A címke nélküli klaszterezés ezzel szemben a karakter-n-gramokkal is csak részben adja vissza az eseménytípusokat: a rövid bejegyzések kevés szót tartalmaznak, és ugyanazt a jelenséget a kezelők sokféleképpen írják le. A gyakorlatban ezért a két megközelítést kombinálják: a klaszterezés a szakértői címkézést segíti, a felcímkézett adatokon tanított osztályozó pedig a további bejegyzéseket sorolja be. A naplóbejegyzések és a 11. fejezet riasztásai együtt adnak teljes képet egy üzemzavarról – ezt a 15. fejezetben hasznosítjuk.

Interaktív alkalmazás: TF-IDF és koszinusz-hasonlóság: saját dokumentumokkal

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/tfidf.html

Colab-notebook: 13. fejezet – szövegbányászat

13.8. Összefoglalás

- A szövegbányászat a szöveget tokenekre bontja, normalizálja és szűri; a magyar nyelv gazdag toldalékolása miatt a lemmatizálás vagy a karakter-n-gramok használata különösen fontos.
- A vektortér-modellben a dokumentumokat TF-IDF súlyozású, ritka vektorok írják le; hasonlóságukat a koszinusz méri.
- A szövegosztályozásban a lineáris modellek (logisztikus regresszió, lineáris SVM) erős, értelmezhető alapmódszerek; a multinomiális naiv Bayes gyors viszonyítási alap.
- A témamodellek (LSA, NMF, LDA) a dokumentumokat témák keverékeként írják le; a témák értelmezése és elnevezése szakértői feladat.
- A szóbeágyazások a disztribúciós hipotézisre épülnek; a PPMI-mátrix SVD-je és a word2vec közeli rokonok.
- A TextRank gráfalapú kivonatoló összefoglalást ad, amely a forrás mondatait idézi.

- A szakmai, zajos szövegeken (naplók, karbantartási jegyek) a karakterszintű reprezentációk robusztusabbak.

Ellenőrző kérdések

1. Miért nulla a minden dokumentumban előforduló szavak TF-IDF súlya?
2. Miért független a koszinusz-hasonlóság a dokumentumok hosszától?
3. Mi a különbség a szótövezés és a lemmatizálás között, és miért fontos ez a magyar nyelvben?
4. Miért lehet hasznos megtartani a „nem” szót a stopszavak szűrésekor?
5. Miben különbözik a szövegklaszterezés és a témamodellezés kimenete?
6. Miért kerülhetnek közel egymáshoz az ellentétes jelentésű szavak egy disztribúciós beágyazásban?
7. Milyen előnye van a kivonatoló összefoglalásnak a megfogalmazó (absztraktív) összefoglalással szemben?

Feladatok

1. **Kézi TF-IDF.** Számítsa ki kézzel a kézi példa második és ötödik dokumentumának koszinusz-hasonlóságát! Hogyan változik az eredmény, ha a `scikit-learn` simított idf-jét használjuk?
2. **Tagadás.** Készítsen olyan reprezentációt, amely a „nem” szót a következő szóval összevonja (*nem_ajánlom*)! Javul-e a negatív osztály felismerése?
3. **Témaszám.** Illesszen NMF-et és LDA-t $k = 4, 6, 8, 10, 12$ témával a mondatokra, és értékelje a témák értelmezhetőségét! Melyik k mellett különül el legjobban a szállítás, az ár és az ügyfélszolgálat?
4. **Termékszintű elemzés.** Számítsa ki termékenként a negatív vélemények arányát és a minőségi panaszok témasúlyát! Összhangban van-e a két mérőszám?
5. **Napló és riasztások.** Kapcsolja össze a műszaknapló bejegyzéseit a 11. fejezet riasztásaival (a bejegyzés előtti 2 órában fellépő riasztások)! Mely riasztástípusok járnak együtt az egyes bejegyzéstípusokkal?

További olvasmányok

A szöveges információkeresés és -osztályozás klasszikus tankönyve Manning és mtsai. (2008). A témamodellek alapcikkei Deerwester és mtsai. (1990), D. D. Lee & Seung (1999) és Blei és mtsai. (2003); a szóbeágyazásokról Mikolov és mtsai. (2013) és Levy & Goldberg (2014) ajánlott. A magyar nyelvű szövegfeldolgozás

eszközeit Orosz és mtsai. (2023) mutatja be; a szakmai szövegek elemzésének sajátosságairól Brundage és mtsai. (2021) ad áttekintést.

V. Alkalmazások

14. Ügyfélkapcsolat-menedzsment és ügyfélanalitika

Miről szól a fejezet?

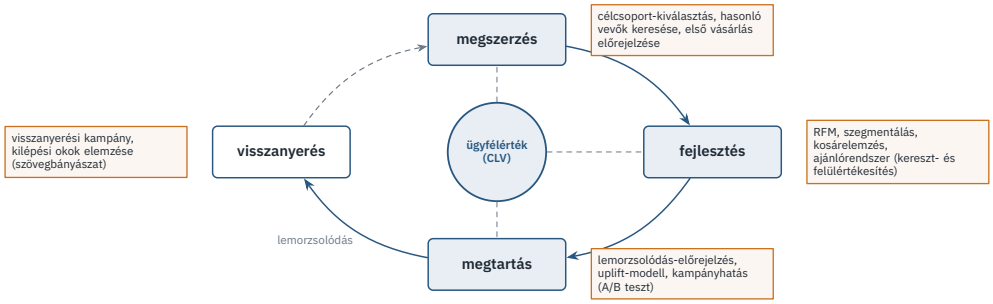
Az ügyfélkapcsolat-menedzsment (customer relationship management, CRM) az adatbányászat egyik legrégebbi és legnagyobb alkalmazási területe. Ez a fejezet az eddigi módszereket – szegmentálás, asszociációs szabályok, osztályozás, értékelés – egy egységes üzleti keretbe illeszti: a vevői életciklus szakaszaihoz rendeli őket, és olyan új eszközökkel egészíti ki, amelyek kifejezetten az ügyfélanalitika kérdéseire válaszolnak. Ezek az RFM-elemzés, az ügyfélérték (CLV) valószínűségi modellezése, az ajánlórendszerek, valamint a kampányok valódi, oksági hatását mérő A/B tesztek és uplift-modellek. A fejezet végén az ügyféladatok elemzésének jogi és etikai kereteit foglaljuk össze.

A fejezet végére az olvasó képes lesz RFM-pontszámokat és -szegmenseket képezni; a BG/NBD és a gamma–gamma modellt illeszteni és az ügyfélértéket előre jelezni; egyszerű kollaboratív szűrést megvalósítani és népszerűségi viszonyítási alappal összevetni; egy randomizált kampánykísérletet kiértékelni; és megérteni, miért a kampány hatására reagáló, nem pedig a legnagyobb kockázatú vevőket kell megcélozni.

14.1. A vevői életciklus és az analitika

A CRM stratégiai keretrendszere (Kumar & Reinartz, 2018; Payne & Frow, 2005) a vevőkapcsolatot életciklusként írja le (14.1. ábra): a vevőt meg kell szerezni, a kapcsolatot fejleszteni (több és értékesebb vásárlás), a vevőt megtartani, és ha mégis távozik, esetleg visszanyerni. Minden szakaszhoz jellegzetes elemzési feladatok tartoznak (Ngai és mtsai., 2009), és mindegyik döntés közös mércéje az **ügyfélérték**: az a jövőbeli nyereség, amelyet a vevő a kapcsolat során várhatóan termel.

Az ügyfélanalitika adatai jellemzően tranzakciós adatok (a 2. fejezet értékesítési ténytáblája), kiegészítve a vevők jellemzőivel, az interakciós adatokkal (webes vi-



14.1. ábra. A vevői életciklus szakaszai és a hozzájuk tartozó elemzési feladatok. Az ügyfélérték (CLV) minden szakasz döntéseinek közös mércéje.

selkedés, kampányreakciók, ügyfélszolgálati megkeresések) és a szöveges visszajelzésekkel (13. fejezet). Az ilyen adatok személyes adatok, ezért kezelésüket a GDPR szabályozza (14.6. alfejezet).

14.2. RFM-elemzés és szegmentálás

Az **RFM-elemzés** (Hughes, 1994) a vevők vásárlási viselkedését három mutatóval írja le: mennyi ideje vásárolt utoljára (*recency*), milyen gyakran vásárol (*frequency*) és mennyit költ (*monetary*). A három mutató egyszerű, könnyen kommunikálható, és sok vizsgálat szerint a jövőbeli viselkedés legerősebb előrejelzői közé tartozik – ezt láttuk a 9. fejezet lemorzsolódási modelljeiben is. A klasszikus eljárás minden mutatót kvintilisekre bont (1–5 pont, ahol 5 a legjobb: a legutóbbi, leggyakoribb, legnagyobb értékű vásárlás), és a pontkombinációk alapján nevesített szegmenseket képez, például „bajnokok” (magas R és F), „veszélyeztetettek” (korábban gyakori, de régóta nem vásárló vevők) vagy „alvók” (alacsony R és F). Rugalmasabb a 5. fejezet klaszterezési módszereivel, az RFM- és egyéb viselkedési jellemzők alapján képzett szegmentálás – ezt a PannonShop vevőin ott már elvégeztük.

Az RFM korlátja, hogy pillanatfelvétel: nem mondja meg, mennyi a vevő *jövőbeli* értéke, és nem veszi figyelembe, hogy egy ritkán vásárló vevőnél egy hosszabb szünet teljesen normális lehet, egy gyakorinál viszont a távozás jele. Erre adnak választ a következő alfejezet valószínűségi modelljei.

14.3. Az ügyfélérték modellezése

14.3.1. Az ügyfélérték fogalma

Az **ügyfélérték** (customer lifetime value, CLV) a vevőtől várható jövőbeli nyereségáramok jelenértéke (Gupta és mtsai., 2006):

$$\text{CLV} = \sum_{t=1}^{H} \frac{m_t \, s_t}{(1+d)^t}, \tag{14.1}$$

ahol m_t a t-edik időszakban várható fedezet (nyereség), s_t annak valószínűsége, hogy a vevő a t-edik időszakban még aktív, d a diszkontráta, H a horizont. Állandó m fedezet és r megtartási ráta mellett végtelen horizonton $s_t = r^t$, és a képlet a $\text{CLV} = m\,r/(1+d-r)$ zárt alakra egyszerűsödik. Ez a „szerződéses” helyzetben (előfizetések, ahol a lemorzsolódás megfigyelhető) jó közelítés. A webáruházak vevői azonban **nem szerződéses** kapcsolatban állnak a céggel: a távozás nem figyelhető meg közvetlenül, csak az, hogy a vevő egy ideje nem vásárolt. Ilyenkor a „még vásárol-e” kérdést valószínűségi modellel kell megválaszolni.

14.3.2. A BG/NBD modell

A „buy till you die” (BTYD) modellcsalád (Fader és mtsai., 2005a; Schmittlein és mtsai., 1987) azt feltételezi, hogy a vevő egy ideig aktív, ezalatt véletlenszerűen vásárol, majd egy nem megfigyelhető időpontban végleg inaktívvá válik (14.2. ábra). A **BG/NBD** modell (Fader és mtsai., 2005a) feltevései:

1. Aktív állapotban a vevő vásárlásai λ intenzitású Poisson-folyamatot követnek.
2. A λ a vevők között $\text{Gamma}(r, \alpha)$ eloszlású (heterogenitás a vásárlási gyakoriságban).
3. Minden vásárlás után a vevő p valószínűséggel inaktívvá válik.
4. A p a vevők között $\text{Béta}(a, b)$ eloszlású (heterogenitás a hűségben).
5. A λ és a p egymástól független.

A modell figyelemre méltó tulajdonsága, hogy egy vevő teljes vásárlási története helyett elegendő három szám: az ismételt vásárlások száma (x), az utolsó vásárlás ideje az első óta (t_x) és a megfigyelés hossza (T) – lényegében az RFM-mutatók. A vevő likelihoodja



14.2. ábra. A BG/NBD modell története egy vevőre. A modell csak a vásárlások számát (x), az utolsó vásárlás idejét (t_x) és a megfigyelés hosszát (T) használja; azt, hogy a vevő mikor vált inaktívvá, nem figyeljük meg.

$$L(r, \alpha, a, b \mid x, t_x, T) = \frac{B(a, b+x)}{B(a,b)} \frac{\Gamma(r+x)\,\alpha^r}{\Gamma(r)\,(\alpha+T)^{r+x}} + \delta_{x>0} \frac{B(a+1, b+x-1)}{B(a,b)} \frac{\Gamma} \tag{14.2}$$

ahol B a béta-, Γ a gammafüggvény, $\delta_{x>0}$ pedig 1, ha volt ismételt vásárlás. Az első tag annak felel meg, hogy a vevő a megfigyelés végén még aktív, a második annak, hogy az utolsó vásárlása után inaktívvá vált. A négy paramétert a vevők log-likelihoodjainak összegét maximalizálva becsüljük, és a modellből zárt alakban számolható annak valószínűsége, hogy a vevő még aktív,

$$P(\text{aktív} \mid x, t_x, T) = \left[1 + \delta_{x>0} \frac{a}{b+x-1} \left(\frac{\alpha+T}{\alpha+t_x}\right)^{r+x}\right]^{-1}, \tag{14.3}$$

valamint a következő t időegységben várható vásárlások száma (lásd a gyakorlatot). Az (14.3) képlet formalizálja az RFM intuícióját: a $T - t_x$ inaktív idő annál inkább a távozás jele, minél gyakrabban vásárolt korábban a vevő (nagy x).

A vásárlások *értékét* a **gamma–gamma modell** (Fader és mtsai., 2005b) írja le: a vevő tranzakcióinak értéke a vevőre jellemző átlag körül gamma-eloszlású, és ez az átlag a vevők között szintén gamma-eloszlású. A modellből a vevő várható átlagos vásárlási értéke a populációs átlag és a vevő saját megfigyelt átlaga ($\bar{z}$) közötti súlyozott átlag:

$$E[Z \mid \bar{z}, n] = \frac{q-1}{pn+q-1} \cdot \frac{\gamma p}{q-1} + \frac{pn}{pn+q-1}\,\bar{z}, \tag{14.4}$$

ahol n a vevő tranzakcióinak száma, p, q, γ a modell paraméterei. Kevés tranzakció esetén a becslés a populációs átlag felé húzódik (zsugorítás), sok tranzakció esetén a vevő saját átlaga dominál. A modell feltételezi, hogy a vásárlások értéke független a gyakoriságuktól – ezt az adatokon ellenőrizni kell.

> **FUTÓ ESET · PANNON ÜZEM**
>
> A BTYD-modellek logikája – egy rejtett „aktív/inaktív” állapot, amelyről csak a megfigyelt események alapján következtethetünk – a mérnöki megbízhatóság-elmélet modelljeire emlékeztet, ahol egy berendezés rejtett állapotára a meghibásodásokból és a karbantartási eseményekből következtetünk. A 15. fejezetben a berendezések állapotfüggő karbantartásánál hasonló gondolatmenettel találkozunk.

14.4. Ajánlórendszerek

A kapcsolat fejlesztésének fő eszköze a **kereszt- és felülértékesítés**: olyan termékek ajánlása, amelyek a vevőt valószínűleg érdeklik. A 6. fejezet asszociációs szabályai („aki kávéfőzőt vesz, az kávébabot is”) az ajánlás legegyszerűbb formái; az **ajánlórendszerek** ennél általánosabbak, és minden vevőnek személyre szabott listát adnak (Aggarwal, 2016).

A **kollaboratív szűrés** a vevők közös vásárlási mintázataiból tanul. A *felhasználóalapú* változat a vevőhöz leginkább hasonló vevőket (szomszédokat) keresi meg, és azokat a termékeket ajánlja, amelyeket a szomszédok vettek, ő még nem. A *tételalapú* változat (Linden és mtsai., 2003; Sarwar és mtsai., 2001) a termékek hasonlóságát számolja ki a közös vásárlók alapján, és a vevő korábbi vásárlásaihoz hasonló termékeket ajánl; nagy vevőszámnál ez hatékonyabb, mert a termék–termék hasonlósági mátrix kisebb és stabilabb. A vevő–termék mátrix bináris (vásárolt-e) vagy értékelési adatokat tartalmaz; a hasonlóság rendszerint koszinusz-hasonlóság (13.2). A **mátrixfaktorizációs** módszerek (Koren és mtsai., 2009) a vevőket és a termékeket egy közös, alacsony dimenziós látens térbe képezik – ez a 13. fejezet NMF- és LSA-módszereinek közeli rokona.

Az ajánlórendszerek értékelése a jövőbeli vásárlások visszatartásával történik: a modellt egy korábbi időszakon tanítjuk, és azt mérjük, hogy a későbbi időszakban ténylegesen megvásárolt termékek közül hány szerepel az ajánlólista első K helyén (találati arány, precision@K, recall@K) (Herlocker és mtsai., 2004). Mindig érdemes egy egyszerű népszerűségi ajánlóval (mindenkinek a legnépszerűbb, még meg nem vásárolt termékeket ajánlja) összevetni: kis termékválaszték mellett ezt meglepően nehéz legyőzni.

14.5. A kampányhatás mérése: A/B teszt és uplift-modellezés

14.5.1. Korreláció helyett oksági hatás

A 9. fejezet lemorzsolódási modellje megmondja, kik a legnagyobb kockázatú vevők. Egy megtartási kampány célzásához azonban nem ez a kérdés, hanem az, hogy *kinél változtat a kampány* a kimeneten. A vevők négy csoportba sorolhatók aszerint, hogyan viselkednének kampánnyal és kampány nélkül (14.3. ábra): a **meggyőzhetők** csak a kampány hatására maradnak, a **biztosak** enélkül is maradnának, az **elveszettek** kampánnyal sem maradnak, az **alvó kutyák** pedig éppen a kampány hatására távoznak (például mert egy kéretlen megkeresés emlékezteti őket arra, hogy lemondják a szolgáltatást). A kockázatalapú célzás az elveszetteket és az alvó kutyákat is megcélozza, a biztosakra pedig fölöslegesen költ – a kampány haszna ezért kisebb lehet a vártnál, vagy akár negatív is.



14.3. ábra. A vevők négy csoportja egy megtartási kampány szempontjából. A kampány csak a meggyőzhetőknél hoz hasznot; az alvó kutyáknál kárt okoz. Egy vevőnél a két lehetséges kimenetel közül mindig csak az egyik figyelhető meg.

14.5.2. A randomizált kísérlet

Egy vevőnél soha nem figyelhetjük meg egyszerre, mi történik kampánnyal és kampány nélkül – ez az oksági következtetés alapproblémája. A **randomizált kontrollált kísérlet** (az online világban A/B teszt) (Kohavi és mtsai., 2020) ezt úgy oldja fel, hogy a vevőket véletlenszerűen osztja kezelt ($T = 1$) és kontroll ($T = 0$) csoportra; így a két csoport minden más szempontból azonos eloszlású, és a kimenetek különbsége a kampány átlagos oksági hatását (average treatment effect, ATE) becsli:

$$\hat{\tau} = \bar{y}_{T=1} - \bar{y}_{T=0}, \qquad \mathrm{SE}(\hat{\tau}) = \sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{n_1} + \frac{\hat{p}_0(1-\hat{p}_0)}{n_0}}, \tag{14.5}$$

ahol bináris kimenetnél $\hat{p}_1$ és $\hat{p}_0$ a két csoport kimeneti aránya, n_1 és n_0 a csoportok mérete.

14.5.3. Uplift-modellek

Az **uplift-modell** az egyéni (feltételes) hatást becsli a vevők jellemzői alapján (Devriendt és mtsai., 2018; Gutierrez & Gérardy, 2017):

$$\tau(\mathbf{x}) = P(Y = 1 \mid \mathbf{x}, T = 0) - P(Y = 1 \mid \mathbf{x}, T = 1), \tag{14.6}$$

ahol $Y = 1$ a lemorzsolódást jelöli, így a pozitív $\tau(\mathbf{x})$ a kampány által megelőzött lemorzsolódás valószínűsége. A legegyszerűbb a **kétmodelles megközelítés** (T-learner) (Künzel és mtsai., 2019): külön osztályozót tanítunk a kezelt és a kontroll csoport adatain, és a becsült valószínűségek különbsége az uplift. Az uplift-modellt nem lehet a szokásos módon, egyedi előrejelzésenként értékelni, hiszen az egyéni hatás nem figyelhető meg. Ehelyett a vevőket a becsült uplift szerint rendezzük, és a lista elejére számoljuk a kezelt és a kontroll csoport kimenetének különbségét; a **Qini-görbe** ezt a kumulált többlethatást mutatja a megcélzott hányad függvényében.

14.6. Jogi és etikai keretek

Az ügyfélanalitika személyes adatokkal dolgozik, ezért a GDPR (Európai Parlament és Tanács, 2016) elvei közvetlenül érintik (1. fejezet). A **célhoz kötöttség** és az **adattakarékosság** elve szerint csak a kitűzött célhoz szükséges adatokat szabad gyűjteni és felhasználni. A vevők RFM-, CLV- vagy lemorzsolódási pontszám szerinti besorolása **profilalkotás**, amelyről a vevőt tájékoztatni kell, és a kizárólag automatizált, a vevőre jelentős hatással járó döntésekkel szemben (22. cikk) a vevőnek joga van emberi beavatkozást kérni. Az ügyfélérték szerinti megkülönböztetés – például eltérő árak vagy kiszolgálási színvonal – etikai kérdéseket is felvet: a modell a múltbeli egyenlőtlenségeket újratermelheti, és a védett tulajdonságokkal korreláló jellemzők (lakóhely, eszköztípus) közvetett diszkriminációhoz vezethetnek (Barocas és mtsai., 2023). A gyakorlatban ezért a modellek fejlesztését dokumentálni, a döntések hatását csoportonként monitorozni, és a vevők felé átláthatóan kommunikálni kell.

14.7. Gyakorlat: a PannonShop vevőinek értéke és megtartása

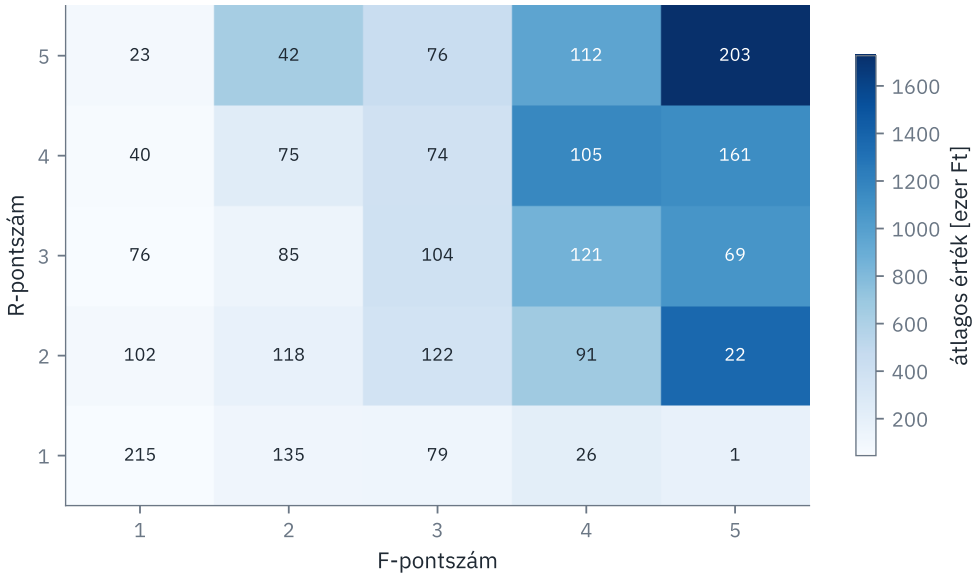
14.7.1. RFM-pontszámok

FUTÓ ESET · PANNONSHOP

A PannonShop marketingcsapata egyszerű, kommunikálható szegmenseket kér a vevőkről a 2025. július 1-jei állapot szerint. Képezzük az RFM-pontokat kvintilisek alapján!

```
vevok = ab.customer_table(ref_date="2025-07-01")
rfm = pd.DataFrame(index=vevok.index)
rfm["R"] = pd.qcut(vevok.R_nap.rank(method="first"), 5, labels=[5, 4, 3, 2,
 ↪  1]).astype(int)
rfm["F"] = pd.qcut(vevok.F_szamla.rank(method="first"), 5, labels=[1, 2, 3,
 ↪  4, 5]).astype(int)
rfm["M"] = pd.qcut(vevok.M_ertek.rank(method="first"), 5, labels=[1, 2, 3,
 ↪  4, 5]).astype(int)
racs = vevok.M_ertek.groupby([rfm.R, rfm.F]).mean().unstack() / 1000
db = rfm.groupby(["R", "F"]).size().unstack()
fig, ax = plt.subplots()
im = ax.imshow(racs.loc[::-1].to_numpy(), cmap="Blues", aspect="auto")
ax.set_xticks(range(5), racs.columns); ax.set_yticks(range(5),
 ↪  racs.index[::-1])
ax.set_xlabel("F-pontszám"); ax.set_ylabel("R-pontszám"); ax.grid(False)
for i, r_ in enumerate(racs.index[::-1]):
    for j, f_ in enumerate(racs.columns):
        n = db.loc[r_, f_] if f_ in db.columns and r_ in db.index else 0
        ax.text(j, i, f"{0 if pd.isna(n) else int(n)}", ha="center",
 ↪  va="center", fontsize=8,
                color="white" if racs.loc[r_, f_] >
 ↪  racs.to_numpy()[~np.isnan(racs.to_numpy())].mean() * 1.6 else
 ↪  "#1F2933")
fig.colorbar(im, ax=ax, shrink=0.8, label="átlagos érték [ezer Ft]")
fig.tight_layout()
```

```
def rfm_szegmens(s):
    if s.R >= 4 and s.F >= 4: return "bajnok"
    if s.R >= 3 and s.F >= 3: return "hűséges"
    if s.R <= 2 and s.F >= 4: return "veszélyeztetett"
    if s.R >= 4 and s.F <= 2: return "új vagy alkalmi"
    if s.R <= 2 and s.F <= 2: return "alvó"
    return "átlagos"
```



14.4. ábra. A vevők átlagos vásárlási értéke (ezer Ft) az R- és az F-pontszám szerint (5 = legutóbbi, illetve leggyakoribb vásárlás). A cellák felirata a vevők száma.

```
rfm["szegmens"] = rfm.apply(rfm_szegmens, axis=1)
valodi = rfm.index.map(ab.load_shop().truth["segment"])
pd.crosstab(rfm.szegmens, valodi, margins=True, margins_name="összesen")
```

col_0 szegmens	akcióvadász	alkalmi	lemorzsolódó	törzsvásárló	viszonteladó	összesen
alvó	105	301	134	16	7	563
bajnok	81	15	15	405	65	581
hűséges	154	96	29	129	36	444
veszélyeztetett	65	9	29	30	7	140
átlagos	100	162	68	23	8	361
új vagy alkalmi	28	90	20	25	13	176
összesen	533	673	295	628	136	2265

FUTÓ ESET · PANNONSHOP

Az RFM-szegmensek jól tükrözik a szimulált „valódi” vevőtípusokat: a bajnokok között a törzsvásárlók, a hűségesek között az akcióvadászok és a törzsvásárlók, az alvók között az alkalmi és a lemorzsolódó vevők vannak túlsúlyban. A „veszélyeztetett” szegmens – korábban gyakran, de régóta nem vásárló vevők, köztük sok akcióvadász – a megtartási kampány kézenfekvő célcsoportja lehetne; hogy valóban az-e, azt a fejezet végén, az uplift-

modellel vizsgáljuk meg.

14.7.2. Ügyfélérték a BG/NBD és a gamma–gamma modellel

A modellt a 2024. január 1. és 2025. június 30. közötti kalibrációs időszakon illesztjük, és a második félévben ellenőrizzük. Az egy napon belüli vásárlásokat egy tranzakcióvá vonjuk össze, az időt hetekben mérjük.

```
from scipy.special import gammaln, betaln, hyp2f1
from scipy.optimize import minimize

tx = ab.shop_clean()
tx = tx[tx.CustomerID.notna()]
napi = (tx.assign(nap=tx.InvoiceDate.dt.normalize())
          .groupby(["CustomerID", "nap"]).Ertek.sum().reset_index())
kal_vege, ell_vege = pd.Timestamp("2025-07-01"), pd.Timestamp("2026-01-01")
kal = napi[napi.nap < kal_vege]
ell = napi[(napi.nap >= kal_vege) & (napi.nap < ell_vege)]
g = kal.groupby("CustomerID")
d = pd.DataFrame({"x": g.size() - 1, "elso": g.nap.min(), "utolso":
 ↪  g.nap.max(), "z_atl": g.Ertek.mean()})
d["t_x"] = (d.utolso - d.elso).dt.days / 7
d["T"] = (kal_vege - d.elso).dt.days / 7
d["y_ell"] = ell.groupby("CustomerID").size().reindex(d.index).fillna(0)
x, t_x, T = d.x.to_numpy(), d.t_x.to_numpy(), d["T"].to_numpy()

def bgnbd_negloglik(log_par):
    r, alfa, a, b = np.exp(log_par)                    # logaritmikus
 ↪  paraméterezés: pozitivitás
    A1 = gammaln(r + x) - gammaln(r) + r * np.log(alfa)
    A2 = betaln(a, b + x) - betaln(a, b)
    A3 = -(r + x) * np.log(alfa + T)
    A4 = np.where(x > 0, np.log(a) - np.log(np.maximum(b + x - 1, 1e-12)) -
 ↪  (r + x) * np.log(alfa + t_x), -np.inf)
    return -np.sum(A1 + A2 + np.logaddexp(A3, A4))      # a (-@eq-bgnbd)
     ↪  log-likelihood

illesztes = minimize(bgnbd_negloglik, np.zeros(4), method="Nelder-Mead",
                     options={"maxiter": 5000, "xatol": 1e-6, "fatol": 1e-6})
r, alfa, a, b = np.exp(illesztes.x)
print(f"r = {r:.3f}, α = {alfa:.2f}, a = {a:.3f}, b = {b:.2f}  (konvergált:
 ↪  {illesztes.success})")
print(f"átlagos heti vásárlási intenzitás aktív vevőnél: r/α = {r /
 ↪  alfa:.3f}; "
```

```
      f"várható lemorzsolódási valószínűség vásárlásonként: a/(a+b) = {a /
      ↪  (a + b):.3f}")
```

```
r = 1.070, α = 12.13, a = 4.402, b = 68.44  (konvergált: True)
átlagos heti vásárlási intenzitás aktív vevőnél: r/α = 0.088; várható
↪  lemorzsolódási valószínűség vásárlásonként: a/(a+b) = 0.060
```

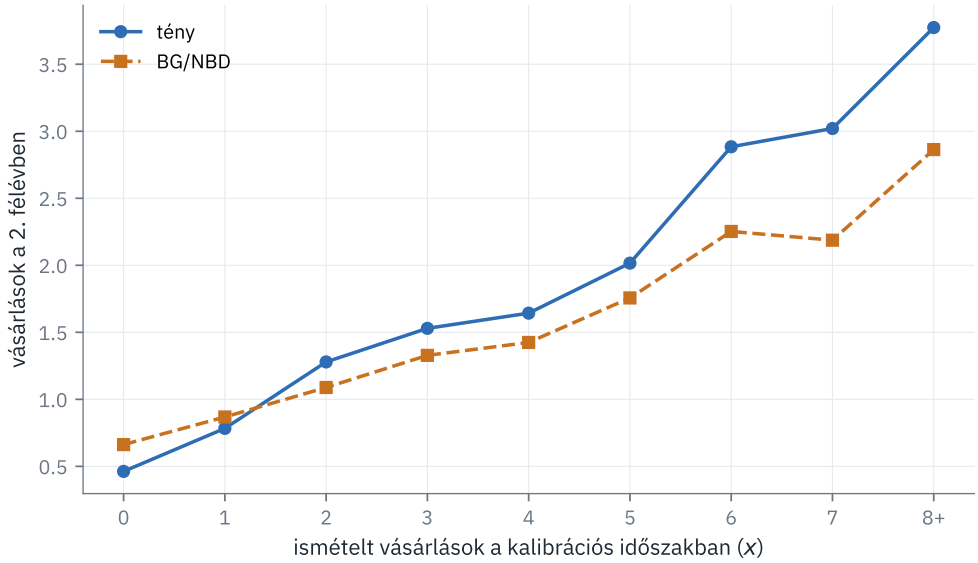
```
def p_aktiv(x, t_x, T):
    return 1 / (1 + (x > 0) * a / (b + x - 1 + (x == 0)) * ((alfa + T) /
    ↪  (alfa + t_x)) ** (r + x))

def varhato_vasarlas(t, x, t_x, T):
    """A következő t hétben várható vásárlások száma (Fader, Hardie és Lee,
    ↪  2005)."""
    hyp = hyp2f1(r + x, b + x, a + b + x - 1, t / (alfa + T + t))
    szamlalo = (a + b + x - 1) / (a - 1) * (1 - ((alfa + T) / (alfa + T +
↪  t)) ** (r + x) * hyp)
    return szamlalo / (1 + (x > 0) * a / (b + x - 1 + (x == 0)) * ((alfa +
    ↪  T) / (alfa + t_x)) ** (r + x))

H = (ell_vege - kal_vege).days / 7
d["p_aktiv"] = p_aktiv(x, t_x, T)
d["y_becsult"] = varhato_vasarlas(H, x, t_x, T)
d["y_naiv"] = (d.x + 1) / d["T"] * H                    # a múltbeli átlagos
↪  gyakoriság kivetítése
print(f"a második félév vásárlásai: tény {d.y_ell.sum():.0f}, BG/NBD
↪  {d.y_becsult.sum():.0f}, naiv {d.y_naiv.sum():.0f}")
print(f"átlagos abszolút hiba vevőnként: BG/NBD {np.mean(np.abs(d.y_ell -
↪  d.y_becsult)):.2f}, "
      f"naiv {np.mean(np.abs(d.y_ell - d.y_naiv)):.2f}")
```

```
a második félév vásárlásai: tény 3656, BG/NBD 3179, naiv 6094
átlagos abszolút hiba vevőnként: BG/NBD 1.26, naiv 1.76
```

```
cs = d.groupby(d.x.clip(upper=8)).agg(teny=("y_ell", "mean"),
↪  becsult=("y_becsult", "mean"))
fig, ax = plt.subplots()
ax.plot(cs.index, cs.teny, "o-", color=ab.PALETTE[0], label="tény")
ax.plot(cs.index, cs.becsult, "s--", color=ab.PALETTE[1], label="BG/NBD")
ax.set_xticks(cs.index, [str(i) for i in cs.index[:-1]] + ["8+"])
ax.set_xlabel("ismételt vásárlások a kalibrációs időszakban ($x$)")
ax.set_ylabel("vásárlások a 2. félévben"); ax.legend()
fig.tight_layout()
```



14.5. ábra. A BG/NBD modell ellenőrzése: a második félévben átlagosan várható és ténylegesen megfigyelt vásárlások száma a kalibrációs időszak ismételt vásárlásainak száma szerint csoportosítva.

FUTÓ ESET · PANNONSHOP

A modell a vevők sorrendjét jól eltalálja, és vevőnként jóval pontosabb, mint a múltbeli gyakoriság egyszerű kivetítése, összességében azonban alulbecsli a második félév vásárlásait. Ennek oka a modell egyik alapfeltevésében keresendő: a BG/NBD időben állandó vásárlási intenzitást feltételez, a második félév viszont a teljes év végi szezonális csúcsot tartalmazza, a kalibrációs időszak pedig másfél évre átlagolja azt – és a forgalom közben nőtt is. A szezonalitás és a növekedés kezelésére a modellt idővel változó intenzitással vagy kovariánsokkal lehet bővíteni; az egyszerű változat ereje a vevők *rangsorolásában* van.

Végül a gamma–gamma modellel a vásárlási értéket, majd a kettő szorzataként az egyéves ügyfélértéket becsüljük (30%-os fedezettel, diszkontálás nélkül):

```
n_tr, z = (d.x + 1).to_numpy(), d.z_atl.to_numpy()
print(f"a gyakoriság és az átlagos érték korrelációja: {np.corrcoef(n_tr,
 ↪  z)[0, 1]:.3f}")

def gg_negloglik(log_par):
```

```
    p, q, gamma = np.exp(log_par)
    ll = (gammaln(p * n_tr + q) - gammaln(p * n_tr) - gammaln(q) + q *
↪   np.log(gamma)
          + (p * n_tr - 1) * np.log(z) + p * n_tr * np.log(n_tr) - (p * n_tr
↪    + q) * np.log(gamma + n_tr * z))
    return -ll.sum()

gg = minimize(gg_negloglik, np.log([1.0, 2.0, 1e4]), method="Nelder-Mead",
↪   options={"maxiter": 5000})
p_, q_, gamma_ = np.exp(gg.x)
d["z_varhato"] = (gamma_ + n_tr * z) * p_ / (p_ * n_tr + q_ - 1)        #
↪   (-@eq-gamma-gamma)
d["CLV_12ho"] = varhato_vasarlas(52, x, t_x, T) * d.z_varhato * 0.30
d["tipus"] = d.index.map(ab.load_shop().truth["segment"])
(d.groupby("tipus")
   .agg(vevok=("x", "size"), p_aktiv=("p_aktiv", "mean"),
↪   CLV_median=("CLV_12ho", "median"),
        CLV_osszes=("CLV_12ho", "sum"))
   .assign(CLV_reszesedes=lambda t: t.CLV_osszes / t.CLV_osszes.sum())
   .drop(columns="CLV_osszes").round({"p_aktiv": 2, "CLV_median": 0,
↪   "CLV_reszesedes": 3}))
```

```
a gyakoriság és az átlagos érték korrelációja: 0.045
```

tipus	vevok	p_aktiv	CLV_median	CLV_reszesedes
akcióvadász	533	0.75	59843.0	0.174
alkalmi	673	0.88	19244.0	0.068
lemorzsolódó	295	0.74	14473.0	0.025
törzsvásárló	628	0.78	59469.0	0.179
viszonteladó	136	0.81	798112.0	0.554

FUTÓ ESET · PANNONSHOP

A gyakoriság és az átlagos vásárlási érték gyakorlatilag független, így a gamma–gamma modell feltevése teljesül. Az ügyfélérték nagyon egyenlőtlenül oszlik meg: a viszonteladók a vevők kis hányadát teszik ki, mégis a várható érték jelentős részét adják, a törzsvásárlók és az akcióvadászok mediánértéke pedig többszöröse az alkalmi és a lemorzsolódó vevőkének. Ez a megoszlás határozza meg, mennyit érdemes egy-egy vevőcsoport megtartására költeni.

14.7.3. Ajánlások a második félévre

FUTÓ ESET · PANNONSHOP

Melyik, a vevő által még meg nem vásárolt termékeket ajánljuk? A kalibrációs időszak vásárlásaiból tanulunk, és azt mérjük, hogy a második félévben a vevők által először megvásárolt termékek közül legalább egy szerepel-e az ötelemű ajánlólistán (top-5 találati arány). Három módszert vetünk össze: a népszerűségi ajánlót, a tételalapú és a felhasználóalapú (50 szomszédos) kollaboratív szűrést.

```
from scipy import sparse
from sklearn.preprocessing import normalize

kal_tx, ell_tx = tx[tx.InvoiceDate < kal_vege], tx[(tx.InvoiceDate >=
 ↪  kal_vege) & (tx.InvoiceDate < ell_vege)]
vevo_idx = {c: i for i, c in enumerate(sorted(kal_tx.CustomerID.unique()))}
termekek = sorted(tx.StockCode.unique()); termek_idx = {p: j for j, p in
 ↪  enumerate(termekek)}
R = sparse.csr_matrix((np.ones(len(kal_tx)),
 ↪  (kal_tx.CustomerID.map(vevo_idx), kal_tx.StockCode.map(termek_idx))),
                      shape=(len(vevo_idx), len(termekek)))
R = (R > 0).astype(float)
Rs = R.toarray()
nepszeruseg = Rs.sum(axis=0)
Rn = normalize(R, axis=0)
S_tetel = (Rn.T @ Rn).toarray(); np.fill_diagonal(S_tetel, 0)       #
 ↪  termék–termék koszinusz
Un = normalize(Rs)                                                   #
 ↪  vevővektorok egységnyi hosszra

def felhasznaloalapu(u, k=50):
    hasonl = Un @ (u / np.linalg.norm(u))
    szomszed = np.argsort(-hasonl)[1:k + 1]
    return hasonl[szomszed] @ Rs[szomszed]

modszerek = {"népszerűség": lambda u: nepszeruseg,
             "tételalapú CF": lambda u: S_tetel @ u,
             "felhasználóalapú CF (k = 50)": felhasznaloalapu}
kesobbi = ell_tx.groupby("CustomerID").StockCode.apply(set)
talalat = {nev: [] for nev in modszerek}
for c in sorted(set(vevo_idx) & set(kesobbi.index)):
    u = Rs[vevo_idx[c]]
    uj = {termek_idx[p] for p in kesobbi[c]} - set(np.flatnonzero(u))
    if not uj:
        continue
```

```
    for nev, f in modszerek.items():
        pont = f(u).astype(float)
        pont[u > 0] = -np.inf                                          # csak
↪   új termékeket ajánlunk
        talalat[nev].append(len(set(np.argsort(-pont)[:5]) & uj) > 0)
pd.Series({nev: np.mean(v) for nev, v in talalat.items()}, name=f"top-5
↪   találati arány ({len(talalat['népszerűség'])}
↪   vevő)").round(3).to_frame()
```

	top-5 találati arány (1121 vevő)
népszerűség	0.709
tételalapú CF	0.675
felhasználóalapú CF (k = 50)	0.782

FUTÓ ESET · PANNONSHOP

A népszerűségi ajánló erős viszonyítási alap: hatvan termék közül a legnépszerűbbek sok vevőnél eltalálják a következő vásárlást. Az egyszerű, összegző tételalapú szűrés ezt nem is éri el – a sok terméket vásárló vevőknél a gyakori termékek hasonlóságai összeadódnak, és a lista a népszerű termékek felé torzul. A felhasználóalapú szűrés viszont jobb a népszerűségi ajánlónál: a hasonló vásárlási mintázatú vevők a szimulációban azonos vevőtípusba tartoznak, és jellemző termékkategóriáik a következő vásárlást is előre jelzik. A tanulság általános: az ajánlórendszer pontosságát mindig népszerűségi viszonyításhoz kell mérni, és a pontozási szabály részletei (normalizálás, szomszédszám) érdemben befolyásolják az eredményt.

14.7.4. Megtartási kampány: kísérlet és uplift-modell

FUTÓ ESET · PANNONSHOP

A PannonShop 2025. július 1-jén megtartási kampányt indít (személyre szóló kupon), és – a helyes gyakorlatnak megfelelően – előbb randomizált kísérletben teszteli: a vevők fele véletlenszerűen kapja meg a kupont. A kísérletet itt szimuláljuk. A kampány hatását vevőtípusonként rögzítjük: az akcióvadászok lemorzsolódását 20, a lemorzsolódókét 10, az alkalmi vevőkét 5 százalékponttal csökkenti, a törzsvásárlókra nincs hatással, a viszonteladók egy részét viszont – akik a kéretlen kuponos megkeresést kéretlen reklámnak tekintik – 6 százalékponttal nagyobb valószínűséggel elriasztja. Az elemző a vevőtípusokat természetesen nem ismeri, csak a jellemzőket és a kísérlet kimenetelét.

```
churn = ab.churn_table(cutoff="2025-07-01", horizon_days=183)
tipus = np.asarray(churn.index.map(ab.load_shop().truth["segment"]
↪ ]).fillna("egyéb"))
hatas = {"akcióvadász": -0.20, "lemorzsolódó": -0.10, "alkalmi": -0.05,
         "törzsvásárló": 0.0, "viszonteladó": 0.06, "egyéb": 0.0}
rng = np.random.default_rng(0)
kezelt = rng.random(len(churn)) < 0.5                # véletlen besorolás
Y0 = churn.lemorzsolodik.to_numpy().astype(int)       # kimenet kampány
↪ nélkül
alap = pd.Series(Y0).groupby(tipus).mean()
tau = np.array([hatas[t] for t in tipus]); p_t = np.array([alap[t] for t in
↪ tipus])
u = rng.random(len(churn))
Y1 = Y0.copy()                                        # kimenet kampánnyal
Y1[(tau < 0) & (Y0 == 1) & (u < -tau / np.maximum(p_t, 1e-9))] = 0
Y1[(tau > 0) & (Y0 == 0) & (u < tau / np.maximum(1 - p_t, 1e-9))] = 1
Y = np.where(kezelt, Y1, Y0)                          # csak ez figyelhető
↪ meg
p1, p0 = Y[kezelt].mean(), Y[~kezelt].mean()
se = np.sqrt(p1 * (1 - p1) / kezelt.sum() + p0 * (1 - p0) / (~kezelt).sum())
print(f"lemorzsolódás: kezelt {p1:.3f}, kontroll {p0:.3f}; becsült hatás
↪ {p1 - p0:+.3f} "
      f"(95% CI: {p1 - p0 - 1.96 * se:+.3f} … {p1 - p0 + 1.96 * se:+.3f});
      ↪ valódi átlagos hatás {np.mean(Y1 - Y0):+.3f}")
```

```
lemorzsolódás: kezelt 0.353, kontroll 0.423; becsült hatás -0.070 (95% CI: -0.113
↪ … -0.027); valódi átlagos hatás -0.075
```

A kísérlet szerint a kampány átlagosan mintegy hét százalékponttal csökkenti a lemorzsolódást. De kiket érdemes megcélozni, ha a költségkeret csak a vevők egy részére elég? Tanítsunk kétmodelles uplift-modellt a kísérlet adatainak 60%-án, és értékeljük a maradék 40%-on, összevetve a lemorzsolódási kockázat szerinti célzással!

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import FunctionTransformer, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression

jell = [c for c in churn.columns if c not in ("lemorzsolodik",
↪ "atl_koztes_nap")]
X = churn[jell].astype(float).to_numpy()
teszt = np.random.default_rng(1).random(len(X)) < 0.4
modell = lambda C=0.5: make_pipeline(SimpleImputer(strategy="median"),
↪ FunctionTransformer(np.log1p),
```

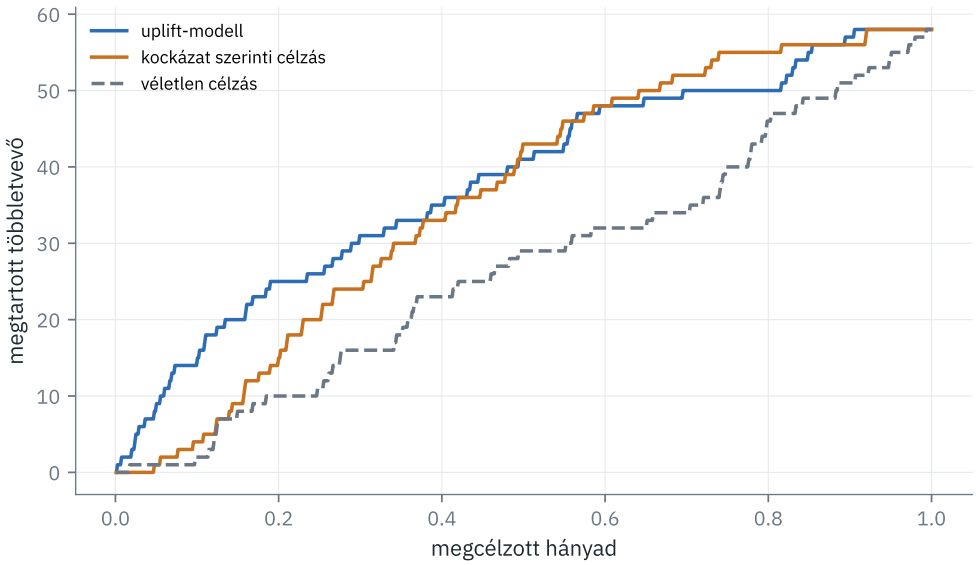
```
                                    StandardScaler(),
    LogisticRegression(max_iter=2000, C=C))
m_kontroll = modell().fit(X[~teszt & ~kezelt], Y[~teszt & ~kezelt])
m_kezelt = modell().fit(X[~teszt & kezelt], Y[~teszt & kezelt])
uplift = m_kontroll.predict_proba(X)[:, 1] - m_kezelt.predict_proba(X)[:,
    1]     # (-@eq-uplift)
kockazat = m_kontroll.predict_proba(X)[:, 1]
    # kampány nélküli kockázat

# A szimulációban mindkét lehetséges kimenetel ismert, így a célzás valódi
    haszna – a megcélzott
# vevők közül a kampány hatására megtartottak száma – közvetlenül
    kiszámítható.
tobblet = (Y0 - Y1)[teszt]
def valodi_haszon(pont):
    sorrend = np.argsort(-pont[teszt])
    return np.cumsum(tobblet[sorrend])

n_teszt = teszt.sum()
gorbek = {"uplift-modell": valodi_haszon(uplift),
          "kockázat szerinti célzás": valodi_haszon(kockazat),
          "véletlen célzás":
              valodi_haszon(np.random.default_rng(5).random(len(X)))}
pd.DataFrame({nev: {f"felső {int(f * 100)}%": g[int(f * n_teszt) - 1] for f
    in (0.1, 0.3, 0.5, 1.0)}
              for nev, g in
                  gorbek.items()}).T.astype(int).rename_axis("megtartott
                  többletvevő (tesztcsoport)")
```

megtartott többletvevő (tesztcsoport)	felső 10%	felső 30%	felső 50%	felső 100%
uplift-modell	14	31	41	58
kockázat szerinti célzás	4	24	43	58
véletlen célzás	2	16	29	58

```
fig, ax = plt.subplots()
for (nev, gorbe), szin, vonal in zip(gorbek.items(), [ab.PALETTE[0],
    ab.PALETTE[1], "#6B7785"], ["-", "-", "--"]):
    ax.plot(np.arange(1, n_teszt + 1) / n_teszt, gorbe, color=szin,
    ls=vonal, lw=1.6, label=nev)
ax.set_xlabel("megcélzott hányad"); ax.set_ylabel("megtartott
    többletvevő"); ax.legend(fontsize=8)
fig.tight_layout()
```



14.6. ábra. A célzási stratégiák valódi haszna a tesztcsoporton (a szimulációban ismert lehetséges kimenetelek alapján): a kampány hatására megtartott vevők kumulált száma a megcélzott hányad függvényében. Az uplift-modell a kampány hatásának nagy részét a vevők kisebb hányadával éri el.

```
pd.DataFrame({"vevők": pd.Series(tipus).value_counts(),
              "valódi hatás [százalékpont]": pd.Series({t: -100 * h + 0.0
               ↪ for t, h in hatas.items()}),
              "becsült uplift [százalékpont]": pd.Series(100 *
               ↪ uplift).groupby(tipus).mean(),
              "kockázat kampány nélkül":
               ↪ pd.Series(kockazat).groupby(tipus).mean()}).drop(index=
               ↪ "egyéb").round(2)
```

14.1. táblázat. A becsült uplift és a lemorzsolódási kockázat vevőtípusonként, a szimulációban rögzített valódi hatással összevetve.

	vevők	valódi hatás [százalékpont]	becsült uplift [százalékpont]	kockázat kampány nélkül
akcióvadász	483	20.0	17.39	0.46
alkalmi	553	5.0	13.08	0.65
lemorzsolódó	187	10.0	7.33	0.65
törzsvásárló	611	0.0	1.76	0.23
viszonteladó	133	-6.0	-4.12	0.12

FUTÓ ESET · PANNONSHOP

A kockázat szerinti célzás a legnagyobb lemorzsolódási valószínűségű vevőket – főként az alkalmi és a lemorzsolódó vevőket – célozza meg, akiknél a kupon csak kis hatású; az uplift-modell ezzel szemben elsősorban az akcióvadászokat sorolja előre, akiknél a kampány valóban változtat. A vevők 30%-ának megcélzásával az uplift-modell a teljes kampányhatás mintegy felét éri el, a kockázatalapú célzás jóval kevesebbet, bár többet, mint a véletlenszerű célzás; ha viszont a vevők felét vagy többet célzunk meg, a különbség eltűnik – a kis költségkeretű kampányoknál számít igazán a jó célzás. (A gyakorlatban a lehetséges kimenetelek nem ismertek; ott a célzás hasznát a kísérlet kezelt és kontroll csoportjának összevetésével, a Qini-görbével kell becsülni, ami ilyen mintaméretnél jelentős véletlen ingadozással jár – lásd a feladatokat.) A becsült egyéni hatások zajosak – a törzsvásárlóknál például a modell pozitív hatást becsül, holott a szimulációban nincs –, ami a kétmodelles megközelítés ismert gyengesége: a két modell hibái összeadódnak. Ezért a gyakorlatban az uplift-modellt mindig egy új, randomizált kísérletben kell ellenőrizni, mielőtt a teljes vevőkörre alkalmaznák.

Interaktív alkalmazás: Ügyfélérték-kalkulátor: megtartási ráta, fedezet, diszkontráta

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/clv.html

Colab-notebook: 14. fejezet – ügyfélanalitika

14.8. Összefoglalás

- Az ügyfélanalitika feladatai a vevői életciklus szakaszaihoz kapcsolódnak; közös mércéjük az ügyfélérték.
- Az RFM-elemzés egyszerű, kommunikálható szegmentálást ad, de pillanatfelvétel.
- Nem szerződéses kapcsolatban a BG/NBD modell becsli, mekkora valószínűséggel aktív még egy vevő, és hányszor fog vásárolni; a gamma–gamma modell a vásárlások értékét. A kettő együtt adja az ügyfélérték előrejelzését.
- Az ajánlórendszerek a vevők közös vásárlási mintázataiból tanulnak; a népszerűségi ajánló erős viszonyítási alap.
- A kampányok oksági hatását randomizált kísérlettel kell mérni; a célzáshoz a lemorzsolódási kockázat helyett a kampány egyéni hatását (uplift) kell becsülni.
- Az ügyfélanalitika profilalkotás: a GDPR elvei és az etikai szempontok (átláthatóság, méltányosság) a módszertan részei.

Ellenőrző kérdések

1. Miben különbözik a szerződéses és a nem szerződéses vevőkapcsolat, és miért kell az utóbbiban valószínűségi modell az ügyfélérték becsléséhez?
2. Mit jelentenek a BG/NBD modell paraméterei, és milyen heterogenitást írnak le?
3. Miért nő egy vevő inaktivitásának valószínűsége az (14.3) szerint gyorsabban, ha korábban gyakran vásárolt?
4. Mit jelent a gamma–gamma modellben a zsugorítás, és mikor erős?
5. Miért nem a legnagyobb lemorzsolódási kockázatú vevőket kell egy megtartási kampányban megcélozni?
6. Miért nem értékelhető egy uplift-modell egyedi előrejelzésenként, és mit mutat a Qini-görbe?
7. Milyen GDPR-kötelezettségek kapcsolódnak a vevők pontozásához?

Feladatok

1. **Megtartási ráta.** Egy előfizetéses szolgáltatás havi fedezete vevőnként 3000 Ft, a havi megtartási ráta 0,9, a havi diszkontráta 1%. Mekkora a végtelen horizontú ügyfélérték? Hogyan változik, ha a megtartási ráta 0,92-re nő?
2. **Aktív valószínűség.** Ábrázolja a BG/NBD modell $P(\text{aktív})$ valószínűségét a $T - t_x$ inaktív idő függvényében $x = 1, 5, 15$ ismételt vásárlás mellett! Értelmezze az eredményt!

3. **Szezonalitás.** Illessze a BG/NBD modellt úgy, hogy a kalibrációs időszak 2024. december 31-én érjen véget, és a következő félévre ellenőrizze! Mennyit változik az összesített előrejelzés hibája?
4. **Ajánló.** Vizsgálja meg a felhasználóalapú szűrés találati arányát a szomszédszám ($k = 10, 25, 50, 100, 200$) függvényében! Készítsen hibrid ajánlót, amely a népszerűségi és a felhasználóalapú pontszámot súlyozza!
5. **Qini-görbe.** Számítsa ki a tesztcsoporton a Qini-görbét – a lista elejére a kontroll és a kezelt csoport lemorzsolódási arányának különbsége szorozva a kezeltek számával –, és vesse össze a valódi haszon görbéjével! Mekkora a véletlen ingadozás? (Segítség: ismételje meg a kísérlet véletlen besorolását több véletlenmaggal.)
6. **Kampányprofit.** Tegyük fel, hogy egy megtartott vevő 20 000 Ft fedezetet hoz, a kupon költsége 1500 Ft. Ábrázolja a kampány nyereségét a megcélzott hányad függvényében az uplift-modell és a kockázatalapú célzás esetén! Mekkora az optimális hányad?

További olvasmányok

A CRM stratégiai kereteiről Payne & Frow (2005) és Kumar & Reinartz (2018), az adatbányászati módszerek CRM-beli alkalmazásairól Ngai és mtsai. (2009) áttekintése ajánlott. Az ügyfélérték modellezésének áttekintése Gupta és mtsai. (2006); a BTYD-modellek alapcikkei Schmittlein és mtsai. (1987), Fader és mtsai. (2005a) és Fader és mtsai. (2005b). Az ajánlórendszerekről Aggarwal (2016) tankönyve, az online kísérletekről Kohavi és mtsai. (2020) könyve, az uplift-modellezésről Gutierrez & Gérardy (2017) és Devriendt és mtsai. (2018) áttekintése ajánlott.

15. Ipari folyamatanalitika

Miről szól a fejezet?

Az ipari folyamatok – vegyipari reaktorok, erőművek, gyártósorok – a legadatgazdagabb rendszerek közé tartoznak: több ezer érzékelő méri őket másodperces vagy perces felbontásban, évtizedekre visszamenőleg. A 14. fejezet üzleti alkalmazásai után ez a fejezet a mérnöki szálat zárja le: a korábbi fejezetekben a Pannon üzem adatain kipróbált módszereket – regresszió, osztályozás, anomália-észlelés, idősorok, szövegbányászat – egy ipari folyamatanalitikai rendszer építőköveiként rendezi egységbe. Sorra vesszük az ipari adatok sajátosságait, a lágy szenzorokat, a több működési állapotú folyamatok felügyeletét, a hibadiagnózist, a riasztáskezelést és a prediktív karbantartást, végül a modellek üzemeltetésének kérdéseit.

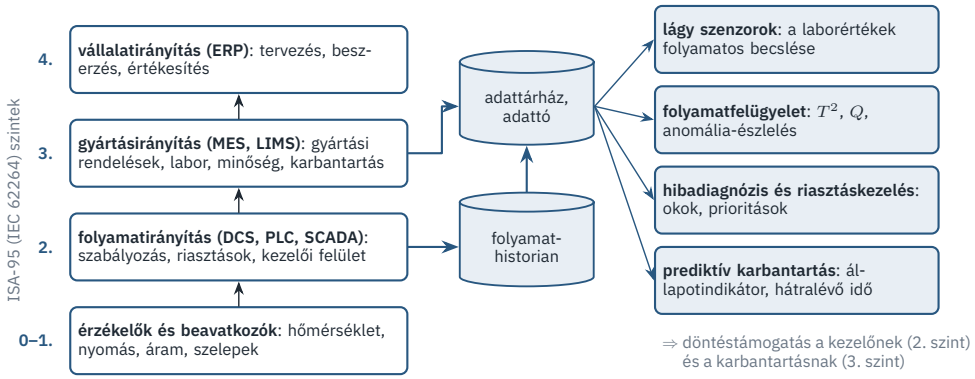
A fejezet végére az olvasó képes lesz lágy szenzort tervezni és értékelni, és a maradékát diagnosztikai célra felhasználni; megítélni, mikor érdemes működési állapotonként külön felügyeleti modellt építeni; a riasztási rendszert a szabványos mutatókkal értékelni; adatalapú állapotindikátort készíteni és a hátralévő időt becsülni; és átlátni, milyen feltételek kellenek az adatalapú modellek tartós ipari üzemeltetéséhez.

15.1. Az ipari adatok sajátosságai

Az ipari adatok a vállalati információs rendszerek hierarchiájának különböző szintjein keletkeznek (15.1. ábra). Az ISA-95 (IEC 62264) modell (IEC, 2013) az érzékelők és beavatkozók szintjétől a folyamatirányításon (DCS, PLC, SCADA) és a gyártásirányításon (MES, laborinformációs rendszer) át a vállalatirányításig (ERP) írja le a szinteket. Az adatelemzés számára a legfontosabb forrás a **folyamat-historian**, amely a mért jeleket időbélyeggel, tömörítve tárolja, kiegészítve a laboreredményekkel, a riasztási és eseménynaplókkal, a karbantartási jegyekkel és a termelési tervvel.

Az ipari adatokat néhány jellegzetesség különbözteti meg az üzleti adatoktól (Ge és mtsai., 2013; Qin, 2014):

- **Dinamika és autokorreláció**: a jelek folytonos fizikai folyamatokat követnek, holtidőkkel és tehetetlenséggel; a minták nem függetlenek (12. fejezet).



15.1. ábra. Az ipari adatok forrásai az ISA-95 szintjein és a folyamatanalitika fő alkalmazásai. Az elemzések eredménye döntéstámogatásként jut vissza a kezelőkhöz és a karbantartáshoz.

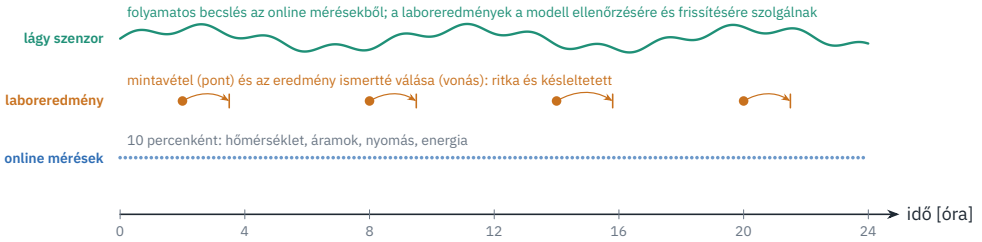
- **Eltérő mintavételezés**: az online mérések percenként, a laboreredmények néhány óránként és késleltetve állnak rendelkezésre.
- **Visszacsatolt szabályozás**: a szabályozók elfedik a zavarokat – a szabályozott változó az alapjelen marad, a zavar a beavatkozó jelben vagy más változókban jelenik meg. Ezért láttuk a szenzor-drift esetén, hogy a hibás mérés maga alig tér el.
- **Több működési állapot**: termékváltások, terhelésváltozások, indítások és leállások más-más összefüggéseket hoznak létre.
- **Lassú változások**: a berendezések elhasználódása, a lerakódás, a katalizátor öregedése és az évszakok miatt a „normál” működés is változik.
- **Kevés címkézett hiba**: a hibák ritkák, sokfélék, és a hibamentes üzem adatai dominálnak (11. fejezet).

Ezek a sajátosságok magyarázzák, miért értékes az ipari elemzésben a folyamatismeret: a fizikai összefüggésekre épülő jellemzők és szürkedobozos modellek a korábbi fejezetekben rendre jobb eredményt adtak, mint a pusztán adatvezérelt megközelítések. A **digitális iker** (Tao és mtsai., 2019) fogalma ennek a gondolatnak a kiteljesedése: a fizikai rendszer folyamatosan frissülő, adatokkal kalibrált modellje, amely felügyeletre, előrejelzésre és optimalizálásra egyaránt használható.

15.2. Lágy szenzorok

A **lágy szenzor** (soft sensor, inferenciális szenzor) olyan modell, amely egy nehezen, drágán, ritkán vagy késleltetve mérhető változót – tipikusan egy laborban

meghatározott minőségi jellemzőt – a folyamatosan mért változókból becsül (Fortuna és mtsai., 2007; Kadlec és mtsai., 2009). Alkalmazásai: a laborértékek közötti időszakok kitöltése (és így a gyorsabb minőségszabályozás), a fizikai műszer ellenőrzése vagy kiváltása, és a „mi lenne, ha” elemzések (15.2. ábra).



15.2. ábra. A lágy szenzor szerepe az eltérő mintavételezésű adatok között. A ritka és késleltetett laboreredmények helyett a modell az online mérésekből folyamatos becslést ad.

A lágy szenzorok modellje lehet első elvekre épülő (fizikai), adatvezérelt vagy a kettőt ötvöző szürkedobozos modell (8. fejezet). Az adatvezérelt modellek közül a vegyiparban a legelterjedtebb a **részleges legkisebb négyzetek** (partial least squares, PLS) regresszió (Wold és mtsai., 2001). A PLS a PCA-hoz hasonlóan kevés látens változóval írja le a sok, egymással erősen korreláló bemenetet, de a látens irányokat nem a bemenetek varianciája, hanem a kimenettel vett kovariancia maximalizálásával választja:

$$\mathbf{w}_1 = \arg\max_{\|\mathbf{w}\|=1} \operatorname{cov}(\mathbf{X}\mathbf{w},\, \mathbf{y})^2, \tag{15.1}$$

majd a bemenetekből és a kimenetből eltávolítja az első látens változó által magyarázott részt, és a további irányokat ugyanígy határozza meg. A látens változók számát keresztvalidációval választjuk. A PLS kezeli a kollinearitást (mint a ridge-regresszió), és látens változói a folyamatfelügyeletben is értelmezhetők.

A lágy szenzor fejlesztésénél a modell *karbantartása* legalább annyira fontos, mint a kezdeti pontosság (Kadlec és mtsai., 2009). A folyamat lassú változásai (lerakódás, katalizátor-öregedés, új alapanyag) miatt a modell pontossága romlik, ezért a modellt rendszeresen frissíteni kell – mozgó ablakkal, rekurzív becsléssel vagy az új laboreredményekhez igazított torzításkorrekcióval. A laboreredmény és a becslés közötti eltérés közben értékes diagnosztikai jel: ha hirtelen megnő, az a modell érvényességének elvesztésére vagy egy mérési hibára utal.

15.3. Több működési állapotú folyamatok felügyelete

A 11. fejezet PCA-alapú felügyelete egyetlen, időben állandó „normál” működést feltételez. A több termékkel, terhelési szinttel vagy évszakfüggő körülményekkel működő folyamatoknál ez a feltevés sérül. A lehetséges stratégiák (Ge és mtsai., 2013): (1) egyetlen globális modell, amely a működési állapotok közötti különbségeket is „normálisnak” tekinti – ez egyszerű, de csökkenti az érzékenységet; (2) állapotonként külön (lokális) modell, az állapotot a termelési tervből vagy klaszterezéssel (például a 5. fejezet Gauss-keverék modelljével) azonosítva; (3) állapotonkénti központosítás és skálázás egy közös modell előtt; (4) adaptív modellek, amelyek a lassú változásokat rekurzív frissítéssel követik (Li és mtsai., 2000). A lokális modellek elvben érzékenyebbek, de mindegyik állapothoz elegendő és reprezentatív normál adat kell – ha egy állapotot csak egy évszakban figyeltünk meg, a modell a másik évszakban téves riasztásokat ad.

15.4. Hibadiagnózis

Az anomália-észlelés azt mondja meg, *hogy* baj van; a **hibadiagnózis** azt, *mi* a baj és *hol*. Venkatasubramanian és mtsai. (2003) a módszereket három csoportba sorolja: kvantitatív modellalapú (a mérések és egy fizikai modell közötti maradékok elemzése, analitikus redundancia), kvalitatív modellalapú (ok-okozati gráfok, hibafák, szakértői rendszerek) és folyamattörténeti (adatvezérelt) módszerek. A könyvben mindhárom típusra láttunk példát: a nyomásmaradék a 9. fejezetben analitikus redundancia volt, a riasztási sorozatok a 6. fejezetben az ok-okozati láncok feltárását segítették, a felügyelt osztályozó és a hozzájárulási ábra pedig adatvezérelt diagnózist adott.

A gyakorlatban a diagnózis több információforrás összevetése: az adatalapú felügyelet riasztása és hozzájárulási ábrája, a hagyományos riasztások sorrendje, a kezelők naplóbejegyzései (13. fejezet) és a karbantartási előzmények együtt adnak teljes képet. Az adatbányászat itt nem a szakértőt helyettesíti, hanem a figyelmét a lényeges jelekre irányítja.

15.5. Riasztáskezelés

A folyamatirányító rendszerek riasztásai a kezelő elsődleges információforrásai a rendellenes helyzetekről. A rosszul tervezett riasztási rendszer azonban több kárt okozhat, mint hasznot: a túl sok, fölösleges, ismétlődő riasztás elvonja a figyelmet, és egy üzemzavar során a „riasztásáradat” a kezelőt cselekvésképtelenné teheti (J. Wang és mtsai., 2016). A riasztáskezelés szabványai – az ANSI/ISA-18.2 (International Society of Automation, 2016) és az EEMUA 191 (EEMUA, 2013) – a riasztási

rendszer teljes életciklusát (filozófia, azonosítás, racionalizálás, tervezés, bevezetés, üzemeltetés, monitorozás, felülvizsgálat) szabályozzák, és mérhető teljesítménymutatókat adnak. Az EEMUA 191 szerint egy kezelőre átlagosan legfeljebb néhány riasztás jusson óránként (10 percenként egy már nehezen kezelhető), 10 perc alatt 10-nél több riasztás már áradatnak számít, és a riasztások nagy részét néhány „rossz szereplő” okozza, amelyeket elsőként kell felülvizsgálni.

Az adatbányászat a riasztáskezelés több lépését támogatja: a gyakori sorozatminták és asszociációs szabályok feltárják az együtt vagy egymás után jelentkező, redundáns riasztásokat, amelyek csoportosíthatók vagy elnyomhatók (Dorgo & Abonyi, 2018); a fecsegő (chattering) riasztások – rövid időn belül sokszor ismétlődő be- és kikapcsolások – egyszerű statisztikákkal azonosíthatók; és az adatalapú felügyelet összevethető a hagyományos, határérték-alapú riasztásokkal.

15.6. Prediktív karbantartás

A karbantartási stratégiák fejlődése a **reaktív** (a meghibásodás utáni javítás) és a **megelőző** (rögzített időközönkénti) karbantartástól az **állapotalapú** és a **prediktív** karbantartás felé halad (Jardine és mtsai., 2006). Ez utóbbiak a berendezés tényleges állapotát egy mérésekből számolt **állapotindikátorral** (health indicator) követik, és a beavatkozást akkor ütemezik, amikor az indikátor – vagy az előrejelzése – egy küszöböt elér (15.3. ábra). A **hátralévő hasznos élettartam** (remaining useful life, RUL) becslése a prognosztika központi feladata (Lei és mtsai., 2018); a becslés mindig bizonytalan, ezért jó gyakorlat egy valószínűségi intervallumot is megadni.



15.3. ábra. Állapotalapú és prediktív karbantartás. Az állapotindikátor a berendezés romlását követi; a jelenlegi trend extrapolációja (bizonytalansági sávval) megadja, mennyi idő van hátra a karbantartási küszöb eléréséig.

Az állapotindikátor lehet közvetlen fizikai mennyiség (rezgésszint, hőmérséklet), egy modell maradéka (a tényleges és a „hibátlan állapotra” jellemző viselkedés eltérése) vagy egy többváltozós statisztika (például a Q). A Pannon üzemben a hűtőköpeny lerakódása a tipikus elhasználódási folyamat: a tisztítás után a hőátadás romlik, a hűtési igény nő, és nyáron a hűtés telítődése minőségi gondokat okoz. Az állapotindikátort a 10. fejezet hűtési igény modelljének mintájára építjük meg.

15.7. Gyakorlat: a Pannon üzem folyamatanalitikai rendszere

15.7.1. Lágy szenzor a termékkoncentrációra

FUTÓ ESET · PANNON ÜZEM

A termék koncentrációját a labor négyóránként méri, az eredmény egy–két óra múlva érkezik. Építsünk lágy szenzort, amely a koncentrációt a 10 perces online mérésekből folyamatosan becsli! Négy modellt hasonlítunk össze: a keresztvalidációval hangolt PLS-t, a ridge-regressziót, a véletlen erdőt és a 8. fejezet szürkedobozos modelljét, amely a betáplált koncentrációt az Arrhenius-kinetikából számolt konverzióval szorozza.

```
from sklearn.cross_decomposition import PLSRegression
from sklearn.linear_model import RidgeCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV, TimeSeriesSplit
from sklearn.metrics import r2_score, mean_absolute_error
from adatbanyaszat.plant import conversion

P = ab.load_plant()
q = ab.quality_table()
q["allapot"] =
↪  P.truth["fault"].reindex(q.ido.dt.floor("10min")).to_numpy()
bemenet = ["T_kornyezet", "F_betap", "c_betap", "T_betap", "T_reaktor",
↪  "T_alapjel", "F_hutoviz",
          "T_hutoviz", "P_reaktor", "P_kevero", "E_fogyasztas"]
tanito = (q.ido < "2025-07-01").to_numpy()
y = q.c_termek.to_numpy()
modellek = {
    "PLS": GridSearchCV(make_pipeline(StandardScaler(), PLSRegression()),
```

```
                        {"plsregression__n_components": [1, 2, 3, 4, 5, 6,
↪  8]}, cv=TimeSeriesSplit(5)),
    "ridge": make_pipeline(StandardScaler(), RidgeCV(alphas=np.logspace(-3,
    ↪  3, 13))),
    "véletlen erdő": RandomForestRegressor(n_estimators=300,
    ↪  min_samples_leaf=3, n_jobs=-1, random_state=0),
}
becsles = {}
for nev, m in modellek.items():
    m.fit(q.loc[tanito, bemenet], y[tanito])
    becsles[nev] = np.ravel(m.predict(q[bemenet]))
becsles["szürkedobozos"] = (q.c_betap * conversion(q.T_reaktor,
↪  q.F_betap)).to_numpy()
print("a PLS látens változóinak száma:",
↪  modellek["PLS"].best_params_["plsregression__n_components"])
teszt = ~tanito
tabla = {}
for nev, p in becsles.items():
    hiba = np.abs(y - p)
    tabla[nev] = {"R² (2. félév)": r2_score(y[teszt], p[teszt]), "MAE (2.
↪  félév)": hiba[teszt].mean()}
    for allapot in ["normál", "szenzor_drift", "betap_zavar",
    ↪  "szelep_akadas", "hutoviz_kieses"]:
        tabla[nev][f"MAE: {allapot}"] = hiba[teszt & (q.allapot ==
↪  allapot).to_numpy()].mean()
pd.DataFrame(tabla).T.round(4)
```

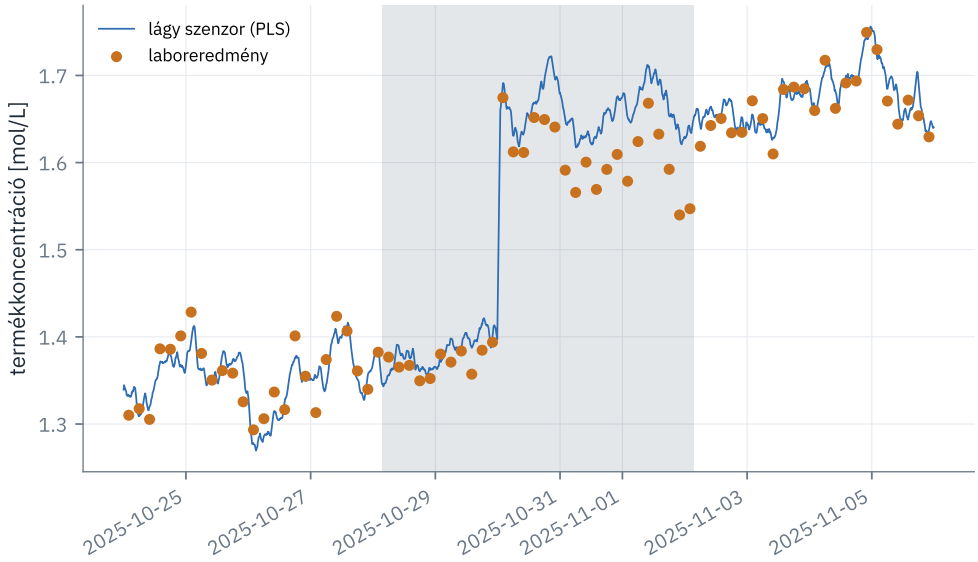
```
a PLS látens változóinak száma: 6
```

	R² (2. félév)	MAE (2. félév)	MAE: normál	MAE: szenzor_drift	MAE: betap_zavar	MAE: szelep_akadas	MAE: hutoviz_kie
PLS	0.9808	0.0155	0.0139	0.0420	0.0189	0.0180	0.0193
ridge	0.9828	0.0151	0.0139	0.0340	0.0189	0.0181	0.0204
véletlen erdő	0.9754	0.0175	0.0154	0.0451	0.0521	0.0177	0.0359
szürkedobozos	0.9790	0.0154	0.0133	0.0532	0.0140	0.0146	0.0164

FUTÓ ESET · PANNON ÜZEM

Mind a négy modell pontos ($R^2 \approx 0{,}98$), a fizikai ismeretre épülő szürkedobozos modell tanítás nélkül is a legjobbak között van – ez az ismert kinetika értéke. A véletlen erdő a legkevésbé pontos, főként a betáplálási zavar idején, amikor a betáplált koncentráció a tanítóadatok tartományán kívülre esik. A legérdekesebb az utolsó oszlopok mintázata: a szenzor-drift idején *minden* modell hibája többszörösére nő. A lágy szenzor a hibás hőmérséklet-mérésből számol, ezért a becslése eltér a laboreredménytől – vagyis a lágy szenzor maradéka a műszerhiba érzékeny jelzője.

```
m10 = ab.plant_clean()
m10 = m10[~m10.leallas].set_index("ido")
ablak = m10["2025-10-24":"2025-11-05"]
folyamatos = np.ravel(modellek["PLS"].predict(ablak[bemenet].rolling(
↪  "60min").mean()))
lab = q[(q.ido >= "2025-10-24") & (q.ido < "2025-11-06")]
drift = P.truth["faults"].query("hiba == 'szenzor_drift' and kezdet >
↪  '2025-10-01'").iloc[0]
fig, ax = plt.subplots()
ax.axvspan(drift.kezdet, drift.veg, color="#6B7785", alpha=0.18, lw=0)
ax.plot(ablak.index, folyamatos, color=ab.PALETTE[0], lw=0.8, label="lágy
↪  szenzor (PLS)")
ax.scatter(lab.ido, lab.c_termek, color=ab.PALETTE[1], s=16, zorder=3,
↪  label="laboreredmény")
ax.set_ylabel("termékkoncentráció [mol/L]"); ax.legend(fontsize=8)
fig.autofmt_xdate()
fig.tight_layout()
```



15.4. ábra. A PLS lágy szenzor folyamatos becslése és a laboreredmények a második szenzor-drift idején (szürke sáv). A drift előrehaladtával a becslés egyre jobban eltér a laboreredményektől.

15.7.2. Globális vagy állapotonkénti felügyeleti modell?

FUTÓ ESET · PANNON ÜZEM

A Pannon üzem három terméket gyárt, eltérő alapjellel és betáplálással. Érdemes-e termékenként külön PCA-modellt építeni? Hasonlítsuk össze a 11. fejezet globális modelljét (empirikus határokkal, háromminás perzisztenciával) a termékenkénti modellekkel és a termékenként központosított, közös modellel!

```
from sklearn.decomposition import PCA

m = ab.plant_clean()
m = m[~m.leallas].copy()
m["hiba"] = P.truth["fault"].reindex(m.ido).to_numpy()
mi = m.set_index("ido")
jelek = ["T_reaktor", "F_hutoviz", "T_hutoviz", "P_reaktor", "F_betap",
 ↪  "c_betap", "T_betap",
         "P_kevero", "E_fogyasztas", "T_kornyezet"]
X = mi[jelek].copy()
X["F_hut_szoras_1h"] = mi.F_hutoviz.rolling("60min").std()
X = X.dropna()
allapot = mi.loc[X.index, "hiba"].to_numpy()
termek = mi.loc[X.index, "termek"].to_numpy()
tr = X.index < "2025-07-01"
nt = tr & (allapot == "normál")

def pca_riasztas(X_tanito, X_minden, q):
    """PCA-felügyelet empirikus (99,5%-os) T² és Q határokkal; logikai
     ↪  riasztásvektort ad."""
    sk = StandardScaler().fit(X_tanito)
    Zt, Z = sk.transform(X_tanito), sk.transform(X_minden)
    pca = PCA(n_components=q).fit(Zt)
    def stat(A):
        T = pca.transform(A)
        return (T ** 2 / pca.explained_variance_).sum(axis=1), ((A -
         ↪  pca.inverse_transform(T)) ** 2).sum(axis=1)
    T2t, Qt = stat(Zt); T2, Q = stat(Z)
    return (T2 > np.quantile(T2t, 0.995)) | (Q > np.quantile(Qt, 0.995))

valtozatok = {"globális modell": pca_riasztas(X[nt], X, 5)}
lokalis = np.zeros(len(X), bool)
kozpontositott = X.copy()
for t in ["A", "B", "C"]:
    s = termek == t
    lokalis[s] = pca_riasztas(X[nt & s], X[s], 4)
    kozpontositott[s] = (X[s] - X[nt & s].mean()) / X[nt & s].std()
```

```
valtozatok["termékenkénti modellek"] = lokalis
valtozatok["termékenként központosított közös modell"] =
↪  pca_riasztas(kozpontositott[nt], kozpontositott, 5)

eredm = {}
for nev, r in valtozatok.items():
    r3 = pd.Series(r, index=X.index).rolling(3).sum().eq(3).to_numpy()
    eredm[nev] = {"téves riasztás (2. félév)": r3[~tr & (allapot ==
↪  "normál")].mean()}
    eredm[nev].update({f: r3[~tr & (allapot == f)].mean() for f in
↪  ["szenzor_drift", "betap_zavar", "szelep_akadas", "hutoviz_kieses"]})
pd.DataFrame(eredm).T.round(3)
```

	téves riasztás (2. félév)	szenzor_drift	betap_zavar	szelep_akadas	hutoviz_kies
globális modell	0.005	0.078	0.993	0.991	0.933
termékenkénti modellek	0.022	0.129	0.167	0.991	0.620
termékenként központosított közös modell	0.015	0.086	0.979	0.991	0.700

```
pd.crosstab(pd.Series(termek, name="termék"), pd.Series(np.where(tr, "1.
↪  félév", "2. félév"), name="időszak"))
```

időszak termék	1. félév	2. félév
A	8496	10414
B	7294	9550
C	10126	6336

FUTÓ ESET · PANNON ÜZEM

Az eredmény elsőre meglepő: a termékenkénti modellek több téves riasztást adnak, és a betáplálási zavart és a hűtővíz-kiesést is rosszabbul ismerik fel, mint a globális modell. A magyarázat a tanítóadatok lefedettségében rejlik. A termékek az év során eltérő arányban és eltérő évszakokban futottak, így egy termékenkénti modell tanítóadatai csak az év egy részének körülményeit (környezeti és hűtővíz-hőmérséklet, lerakódás) fedik le; a második félév más körülményeit a modell rendellenesnek látja, a határai pedig a kevesebb adat miatt bizonytalanabbak. A globális modell az összes termék összes körülményét látta, és empirikus határai a termékek közötti különbségeket is magukba foglalják. A tanulság nem az, hogy a lokális modellek rosszak, hanem az, hogy az állapotfüggő felügyelet csak akkor javít, ha az állapotok összefüggési szerkezete valóban eltér, és mindegyikről elegendő, reprezentatív normál adat áll rendelkezésre; különben a lassú változásokat követő adaptív modellek vagy a globális modell a jobb választás.

15.7.3. A riasztási rendszer értékelése

```
riaszt = P.alarms.copy()
tiz_perc = riaszt.groupby(riaszt.ido.dt.floor("10min")).size()
oras = riaszt.groupby(riaszt.ido.dt.floor("1h")).size()
riaszt = riaszt.sort_values("ido")
riaszt["elozo"] = riaszt.groupby("riasztas").ido.diff()
pd.Series({
    "riasztások száma (1 év)": len(riaszt),
    "átlagos riasztásszám / nap": round(len(riaszt) / 365, 2),
    "legtöbb riasztás 10 perc alatt": int(tiz_perc.max()),
    "áradatos 10 perces ablakok (≥ 10 riasztás)": int((tiz_perc >=
    ↪  10).sum()),
    "legtöbb riasztás egy óra alatt": int(oras.max()),
    "30 percen belül ismétlődő riasztások aránya": round((riaszt.elozo <
    ↪  pd.Timedelta("30min")).mean(), 3),
}, name="érték").to_frame()
```

	érték
riasztások száma (1 év)	605.00
átlagos riasztásszám / nap	1.66
legtöbb riasztás 10 perc alatt	3.00
áradatos 10 perces ablakok (≥ 10 riasztás)	0.00
legtöbb riasztás egy óra alatt	3.00
30 percen belül ismétlődő riasztások aránya	0.00

A riasztási terhelés messze az EEMUA 191 ajánlásai alatt marad – a szimulált üzem riasztási rendszere „racionalizált”. Fontosabb kérdés, hogy a riasztások mennyire relevánsak: hányad részük kapcsolódik valódi üzemzavarhoz, és melyik zavart jelzik?

```
esemenyek = P.truth["faults"]
def esemenyhez(t):
    talalt = esemenyek[(esemenyek.kezdet <= t) & (esemenyek.veg +
↪  pd.Timedelta("2h") >= t)]
    return talalt.hiba.iloc[0] if len(talalt) else "nincs (normál üzem)"
riaszt["esemeny"] = riaszt.ido.apply(esemenyhez)
osszesito = pd.crosstab(riaszt.riasztas, riaszt.esemeny)
osszesito.insert(0, "összesen", osszesito.sum(axis=1))
osszesito["eseményhez kötődő arány"] = (1 - osszesito["nincs (normál
↪  üzem)"] / osszesito["összesen"]).round(2)
osszesito.sort_values("összesen", ascending=False)
```

15.1. táblázat. A riasztástípusok gyakorisága és hibaeseményekhez kötődő aránya (a riasztás egy hibaesemény alatt vagy azt követő két órán belül jelentkeze

esemeny riasztas	összesen	betap_zavar	hutoviz_kieses	nincs (normál üzem)	szelep_akadas	szenzor_drift	ese
MINOSEG_ELTERES	189	25	2	138	5	19	0.1
HUTOVIZ_OSZCILLACIO	103	2	13	78	2	8	0.2
HUTES_TELITETT	92	0	5	79	0	8	0.1
T_MAGAS	73	0	5	65	1	2	0.1
HUTOVIZ_MELEG	72	3	8	60	0	1	0.1
BETAP_KONC_ALACSONY	43	2	0	40	0	1	0.0
NYOMAS_MAGAS	26	0	0	24	1	1	0.0
BETAP_LEALLT	7	0	0	5	0	2	0.2

```
def elso_riasztas(e):
    r = riaszt[(riaszt.ido >= e.kezdet) & (riaszt.ido <= e.veg)]
    return pd.Series({"első riasztás": r.riasztas.iloc[0] if len(r) else
    ↪  "—",
                      "késés [óra]": round((r.ido.iloc[0] -
                      ↪  e.kezdet).total_seconds() / 3600, 1) if len(r)
                      ↪  else np.nan})
esemenyek.join(esemenyek.apply(elso_riasztas, axis=1))[["hiba", "kezdet",
↪  "első riasztás", "késés [óra]"]] \
    .query("kezdet >= '2025-07-01'").reset_index(drop=True)
```

	hiba	kezdet	első riasztás	késés [óra]
0	szenzor_drift	2025-07-10 06:50:00	HUTES_TELITETT	0.5
1	hutoviz_kieses	2025-07-11 02:20:00	HUTOVIZ_MELEG	0.0
2	hutoviz_kieses	2025-07-24 10:20:00	HUTOVIZ_MELEG	0.0
3	hutoviz_kieses	2025-08-05 06:30:00	HUTOVIZ_MELEG	0.0
4	hutoviz_kieses	2025-08-18 15:30:00	HUTES_TELITETT	0.2
5	betap_zavar	2025-08-19 06:30:00	BETAP_KONC_ALACSONY	0.0
6	hutoviz_kieses	2025-09-01 16:20:00	HUTOVIZ_OSZCILLACIO	0.0
7	szelep_akadas	2025-09-23 17:00:00	HUTOVIZ_OSZCILLACIO	0.3
8	szenzor_drift	2025-10-28 03:30:00	HUTOVIZ_OSZCILLACIO	7.3

FUTÓ ESET · PANNON ÜZEM

A leggyakoribb riasztás, a minőségi eltérés (MINOSEG_ELTERES), nagyrészt normál üzemben jelentkezik: ezek a hűtés nyári telítődéséből és a termék célértéke körüli ingadozásból eredő, utólagos jelzések, amelyek a laboreredmény késése miatt órákkal az ok után érkeznek. A hűtővíz felmelegedését, a betáplálási zavart és a szelepakadást a határérték-alapú riasztások gyorsan és a hiba jellegének megfelelő üzenettel jelzik. A szenzor-drift idején fellépő első riasztások (a hűtés telítődése, a hűtővízáram ingadozása) viszont egyidejű, más jelenségekhez kötődnek, és a hiba okára egyáltalán nem utalnak – a hibás hőmérséklet-mérés a határértékek szempontjából „normálisnak” látszik. Az adatalapú felügyelet (11.1. táblázat) a hirtelen hibákat hasonlóan gyorsan észlelte, a driftet pedig 10, illetve 50 óra

után, a sérült összefüggések alapján. A két rendszer tehát kiegészíti egymást: a határérték-alapú riasztások egyszerűek és jól érthetők, az adatalapú felügyelet a változók közötti összefüggések sérülését is észreveszi.

15.7.4. Állapotindikátor és hátralévő idő a hűtőköpeny lerakódására

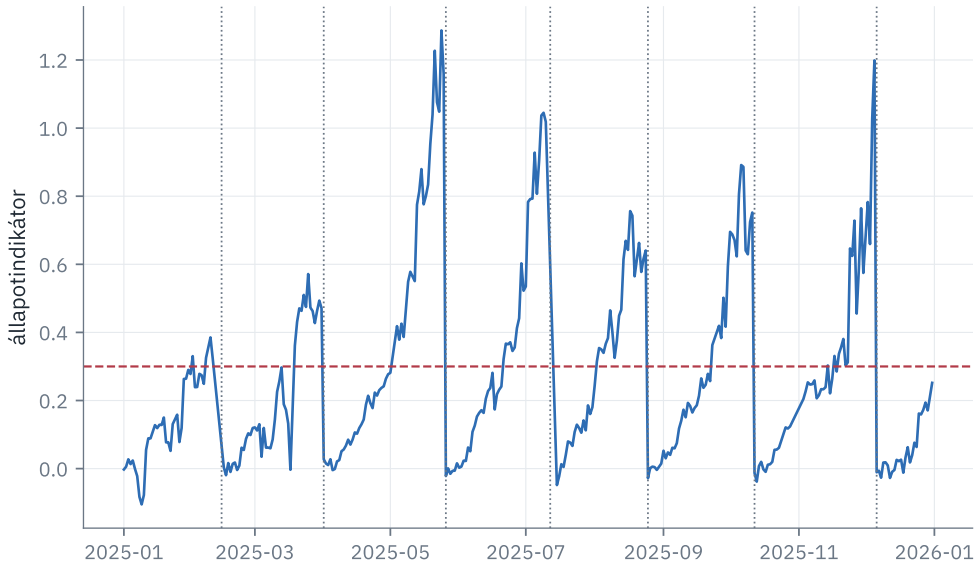
FUTÓ ESET · PANNON ÜZEM

A hűtőköpenyt jelenleg nagyjából hét hetente tisztítják. Készítsünk állapotindikátort, amely a lerakódás mértékét az üzemi adatokból becsli, és vizsgáljuk meg, mennyire jelezhető előre, mikor éri el egy karbantartási küszöböt! Az indikátor a hűtési igény relatív többlete a *tiszta* állapotra jellemző igényhez képest: a hűtési igény modelljét (másodfokú polinomos ridge-regresszió) csak a tisztítások utáni első hét nap adatain tanítjuk, így a modell a tiszta hűtőköpeny viselkedését írja le.

```
from sklearn.preprocessing import PolynomialFeatures
m = ab.plant_clean()
utolso = m.ido.where(m.leallas).ffill()
m["nap_tisztitas_ota"] = ((m.ido - utolso).dt.total_seconds() /
 ↪  86400).fillna((m.ido - m.ido.iloc[0]).dt.total_seconds() / 86400)
m = m[~m.leallas].copy()
m["hiba"] = P.truth["fault"].reindex(m.ido).to_numpy()
be = ["T_reaktor", "T_alapjel", "T_hutoviz", "F_betap", "c_betap",
 ↪  "T_betap", "T_kornyezet"]
h = (m[m.hiba == "normál"].set_index("ido")[be + ["F_hutoviz",
 ↪  "nap_tisztitas_ota"]]
     .resample("1h").mean().dropna())
tiszta = h.nap_tisztitas_ota < 7
modell_tiszta = make_pipeline(StandardScaler(), PolynomialFeatures(2),
 ↪  StandardScaler(),
                              RidgeCV(alphas=np.logspace(-3, 3,
 ↪  13))).fit(h.loc[tiszta, be], h.F_hutoviz[tiszta])
h["HI"] = h.F_hutoviz / modell_tiszta.predict(h[be]) - 1          # relatív
 ↪  többletigény
napi = h[["HI", "nap_tisztitas_ota"]].resample("1D").mean().dropna()
napi["lerakodas"] =
 ↪  P.truth["fouling"].resample("1D").mean().reindex(napi.index)   # csak
 ↪  az ellenőrzéshez
print(f"az állapotindikátor és a (szimulált, a valóságban nem mérhető)
 ↪  lerakódás korrelációja: "
      f"{napi.HI.corr(napi.lerakodas):.3f}")
```

```
az állapotindikátor és a (szimulált, a valóságban nem mérhető) lerakódás
↪  korrelációja: 0.892
```

```
fig, ax = plt.subplots()
ax.plot(napi.index, napi.HI, color=ab.PALETTE[0], lw=1.2)
for d in P.truth["cleanings"]:
    ax.axvline(d, color="#6B7785", lw=0.8, ls=":")
ax.axhline(0.30, color=ab.PALETTE[4], lw=1, ls="--")
ax.set_ylabel("állapotindikátor")
fig.tight_layout()
```



15.5. ábra. A hűtőköpeny állapotindikátora (a hűtési igény relatív többlete a tiszta állapothoz képest, napi átlag) az év során. A függőleges vonalak a tisztítások, a szaggatott vízszintes vonal a 30%-os karbantartási küszöb.

Az indikátor minden tisztítás után nulla közelébe esik, majd egyre gyorsulva nő – a lerakódás rontja a hőátadást, és a szükséges hűtővízáram a hőátadási tényező csökkenésével nemlineárisan nő. A hátralévő idő becsléséhez a korábbi ciklusokból egy közös romlási görbét tanulunk ($\mathrm{HI}(t) \approx \beta_1 t + \beta_2 t^2$), és az aktuális ciklus első húsz napjának adataiból egyetlen skálatényezővel igazítjuk az aktuális ciklushoz; a becsült küszöbátlépés az igazított görbe és a küszöb metszéspontja.

```
kuszob, k = 0.30, 20
napi["ciklus"] = (napi.nap_tisztitas_ota.diff() < -1).cumsum()
```

```
ciklusok = [g.sort_values("nap_tisztitas_ota") for _, g in
↪  napi.groupby("ciklus") if len(g) >= 30]
sorok = []
for i in range(1, len(ciklusok)):
    elozmeny = pd.concat(ciklusok[:i])
    elozmeny = elozmeny[elozmeny.nap_tisztitas_ota <= 45]
    A = np.column_stack([elozmeny.nap_tisztitas_ota,
↪   elozmeny.nap_tisztitas_ota ** 2])
    beta, *_ = np.linalg.lstsq(A, elozmeny.HI.to_numpy(), rcond=None)
    g = ciklusok[i]
    tenyleges = g.nap_tisztitas_ota[g.HI.rolling(3, center=True).mean() >=
↪   kuszob].min()
    s = g[(g.nap_tisztitas_ota >= 2) & (g.nap_tisztitas_ota <= k)]
    alak = beta[0] * s.nap_tisztitas_ota + beta[1] * s.nap_tisztitas_ota **
↪   2
    skala = float(alak @ s.HI / (alak @ alak))
    t = np.linspace(0, 120, 4801)
    becsult = t[np.argmax(skala * (beta[0] * t + beta[1] * t ** 2) >=
↪   kuszob)]
    sorok.append({"ciklus kezdete": g.index[0].date(), "skálatényező":
↪   round(skala, 2),
                  "becsült átlépés [nap]": round(becsult, 1), "tényleges
                   ↪  átlépés [nap]": round(tenyleges, 1),
                  "hiba [nap]": round(becsult - tenyleges, 1)})
rul = pd.DataFrame(sorok)
print(f"átlagos abszolút hiba: {rul['hiba [nap]'].abs().mean():.1f} nap")
rul
```

```
átlagos abszolút hiba: 5.1 nap
```

15.2. táblázat. A 30%-os küszöb átlépésének becsült és tényleges ideje ciklusonként (napokban a tisztítás óta), a ciklus 20. napján készített előrejelzés alapján.

	ciklus kezdete	skálatényező	becsült átlépés [nap]	tényleges átlépés [nap]	hiba [nap]
0	2025-02-15	1.27	31.1	34.2	-3.1
1	2025-04-01	1.66	25.6	29.8	-4.1
2	2025-05-26	2.06	21.8	26.2	-4.3
3	2025-07-15	1.77	22.3	21.2	1.1
4	2025-08-25	1.36	23.8	29.0	-5.2
5	2025-10-12	1.30	24.2	37.2	-13.0

FUTÓ ESET · PANNON ÜZEM

Az állapotindikátor jól követi a lerakódást, a hátralévő idő becslése azonban átlagosan mintegy öt napos, egy ciklusban közel két hetes hibával terhelt; a becslés többnyire korai, vagyis konzervatív. A hibák fő forrása, hogy az indikátor nemcsak a lerakódástól függ: a nyári ciklusokban a melegebb

hűtővíz és a nagyobb terhelés gyorsabb növekedést okoz (a skálatényező is nagyobb), és ezt a korábbi ciklusokból tanult görbe nem tudja előre. A prediktív karbantartás gyakorlatában ezért a becslést valószínűségi intervallummal adják meg, a modellt a körülmények (évszak, terhelés) előrejelzésével bővítik, és a döntést nem egyetlen pontbecslésre, hanem a kockázat és a karbantartási költség mérlegelésére alapozzák. Már a puszta indikátor is jelentős előrelépés a rögzített időközönkénti tisztításhoz képest: megmutatja, mikor valóban szükséges a beavatkozás, és mikor halasztható.

15.8. A modellek üzemeltetése

Egy adatalapú modell értéke csak akkor realizálódik, ha tartósan, megbízhatóan működik az üzemi környezetben. Ez szoftvermérnöki és szervezeti feladat is: a gépi tanulási rendszerek „rejtett technikai adóssága” – az adatfüggőségek, a visszacsatolási hurkok, a konfigurációs bonyolultság – sokszor nagyobb, mint maga a modell (Sculley és mtsai., 2015). A CRISP-ML(Q) folyamatmodell (Studer és mtsai., 2021) ezért a 1. fejezet CRISP-DM modelljét a minőségbiztosítással és a modellek üzemeltetésével (monitorozás, karbantartás) egészíti ki. Az ipari alkalmazások legfontosabb kérdései:

- **Adatminőség és -elérés**: a historian adatainak tömörítése, a hiányzó és hibás mérések kezelése valós időben (3. fejezet).
- **A modell érvényességének felügyelete**: a bemenetek eloszlásának és a modell hibájának folyamatos követése, a koncepciósodródás (concept drift) észlelése (Gama és mtsai., 2014), és a modell újratanításának szabályozott folyamata.
- **Az ember szerepe**: a kezelőnek értenie kell, miért riaszt vagy mit javasol a rendszer; az értelmezhető modellek és a hozzájárulási ábrák ezt szolgálják. A döntés felelőssége az emberé marad.
- **Biztonság és szabályozás**: a folyamatirányításba beavatkozó modellekre a funkcionális biztonsági követelmények, az MI-rendszerekre az európai MI-rendelet (Európai Parlament és Tanács, 2024) előírásai vonatkoznak.

Interaktív alkalmazás: Lágy szenzor: bemenetek, modell és a laboreredmények

A böngészőben futtatható változat: https://complexsystemsintelligence.github.io/adatbanyaszat/apps/lagy-szenzor.html

Colab-notebook: 15. fejezet – folyamatanalitika

15.9. Összefoglalás

- Az ipari adatok dinamikusak, eltérő mintavételezésűek, szabályozott és több működési állapotú folyamatokból származnak; a folyamatismeret ezért az elemzés kulcsa.
- A lágy szenzor a ritkán mért minőségi jellemzőket becsli az online mérésekből; a laboreredmény és a becslés eltérése a modell érvényességét és a műszerhibákat is jelzi.
- A több működési állapotú folyamatok felügyeletében a lokális modellek csak elegendő és reprezentatív állapotonkénti adat mellett előnyösek.
- A hibadiagnózis modellalapú, tudásalapú és adatvezérelt eszközöket és több információforrást kombinál.
- A riasztási rendszert a szabványos mutatókkal (riasztásszám, áradatok, fecsegés, relevancia) kell értékelni; az adatbányászat a redundáns és fölösleges riasztások azonosítását segíti.
- A prediktív karbantartás állapotindikátorra és a hátralévő idő – bizonytalanságot is számszerűsítő – előrejelzésére épül.
- Az adatalapú modellek ipari értéke az üzemeltetésen múlik: adatminőség, monitorozás, újratanítás, átláthatóság és emberi felelősség.

Ellenőrző kérdések

1. Miben különböznek az ipari folyamatadatok a webáruház tranzakciós adataitól?
2. Miért nehéz a szabályozott változó mérési hibáját közvetlenül észlelni?
3. Miben különbözik a PLS a főkomponens-regressziótól?
4. Miért lehet a lágy szenzor maradéka diagnosztikai jel?
5. Mikor érdemes működési állapotonként külön felügyeleti modellt építeni?
6. Milyen mutatókkal értékelhető egy riasztási rendszer az EEMUA 191 szerint?
7. Miért kell a hátralévő hasznos élettartamot intervallummal megadni?

Feladatok

1. **Adaptív lágy szenzor.** Frissítse a PLS lágy szenzort mozgó ablakkal (az utolsó 60 nap laboreredményeivel, hetente)! Javul-e a második félév pontossága? Hogyan hat a frissítés a szenzor-drift észlelhetőségére?
2. **Lágy szenzor mint detektor.** Készítsen EWMA-kártyát a lágy szenzor maradékára (a laboreredmény és a becslés különbségére)! Mennyi idő után jelez a szenzor-drift során?

3. **Állapotazonosítás klaszterezéssel.** Azonosítsa a működési állapotokat a termékazonosító nélkül, Gauss-keverék modellel (5. fejezet), és építsen állapotonkénti felügyeleti modelleket! Hogyan viszonyul az eredmény a termékenkénti modellekhez?
4. **Riasztási sorozatok.** Keressen gyakori riasztási sorozatokat a hibaesemények körül (6. fejezet), és javasoljon egy riasztáscsoportosítást, amely a redundáns riasztásokat egyetlen összefoglaló jelzéssé vonja össze!
5. **Valószínűségi RUL.** A korábbi ciklusok skálatényezőinek eloszlása alapján adjon 80%-os előrejelzési intervallumot a küszöbátlépés idejére! Hány ciklusban esik a tényleges érték az intervallumba?

További olvasmányok

A lágy szenzorokról Fortuna és mtsai. (2007) könyve és Kadlec és mtsai. (2009) áttekintése, a PLS-ről Wold és mtsai. (2001) ajánlott. Az adatalapú folyamatfelügyelet és -diagnosztika áttekintései Venkatasubramanian és mtsai. (2003), Qin (2012) és Ge és mtsai. (2013); a folyamatadatok elemzésének kihívásairól Qin (2014) ír. A riasztáskezelésről J. Wang és mtsai. (2016) áttekintése, valamint az ANSI/ISA-18.2 (International Society of Automation, 2016) és az EEMUA 191 (EEMUA, 2013) szabvány, a prediktív karbantartásról Jardine és mtsai. (2006) és Lei és mtsai. (2018) ajánlott.

Irodalomjegyzék

Abadi, D. J., Madden, S. R., & Hachem, N. (2008). Column-Stores vs. Row-Stores: How Different Are They Really? *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, 967–980.

Abonyi, J. (Szerk.). (2006). *Adatbányászat – a hatékonyság eszköze. Gyakorlati útmutató kezdőknek és haladóknak*. ComputerBooks.

Abonyi, J., & Feil, B. (2007). *Cluster Analysis for Data Mining and System Identification*. Birkhäuser.

Abonyi, J., Feil, B., Németh, S., & Árva, P. (2005). Modified Gath–Geva Clustering for Fuzzy Segmentation of Multivariate Time-Series. *Fuzzy Sets and Systems*, *149*(1), 39–56. https://doi.org/10.1016/j.fss.2004.07.008

Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer. https://doi.org/10.1007/978-3-319-29659-3

Aggarwal, C. C. (2017). *Outlier Analysis* (2. kiad.). Springer. https://doi.org/10.1007/978-3-319-47578-3

Agrawal, R., Faloutsos, C., & Swami, A. (1993). Efficient Similarity Search in Sequence Databases. *Foundations of Data Organization and Algorithms (FODO 1993), Lecture Notes in Computer Science*, *730*, 69–84.

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining Association Rules between Sets of Items in Large Databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 207–216.

Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules. *Proceedings of the 20th International Conference on Very Large Data Bases*, 487–499.

Ambroise, C., & McLachlan, G. J. (2002). Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the National Academy of Sciences*, *99*(10), 6562–6566.

Anscombe, F. J. (1973). Graphs in Statistical Analysis. *The American Statistician*, *27*(1), 17–21.

Armbrust, M., Ghodsi, A., Xin, R., & Zaharia, M. (2021). Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics. *Proceedings of the 11th Conference on Innovative Data Systems Research (CIDR)*.

Arthur, D., & Vassilvitskii, S. (2007). k-means++: The Advantages of Careful Seeding. *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035.

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, *31*(3), 606–660.

Baldi, P., & Hornik, K. (1989). Neural Networks and Principal Component Analysis: Learning from Examples Without Local Minima. *Neural Networks*, *2*(1), 53–58. https://doi.org/10.1016/0893-6080(89)90014-2

Barocas, S., Hardt, M., & Narayanan, A. (2023). *Fairness and Machine Learning: Limitations and Opportunities*. MIT Press.

Bayardo, R. J. (1998). Efficiently Mining Long Patterns from Databases. *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, 85–93.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic Differentiation in Machine Learning: A Survey. *Journal of Machine Learning Research*, *18*(153), 1–43.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. https://doi.org/10.1109/72.279181

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, *191*, 192–213.

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, *13*, 281–305.

Berndt, D. J., & Clifford, J. (1994). Using Dynamic Time Warping to Find Patterns in Time Series. *KDD Workshop (AAAI Technical Report WS-94-03)*, 359–370.

Bezdek, J. C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bishop, C. M., & Bishop, H. (2024). *Deep Learning: Foundations and Concepts*. Springer. https://doi.org/10.1007/978-3-031-45468-4

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Bolton, R. J., & Hand, D. J. (2002). Statistical Fraud Detection: A Review. *Statistical Science*, *17*(3), 235–255. https://doi.org/10.1214/ss/1042727940

Boor, C. de. (2001). *A Practical Guide to Splines* (Revised). Springer.

Box, G. E. P., & Cox, D. R. (1964). An Analysis of Transformations. *Journal of the Royal Statistical Society: Series B*, *26*(2), 211–252.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control* (5. kiad.). Wiley.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying Density-Based Local Outliers. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 93–104.

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, *78*(1), 1–3.

Brin, S., Motwani, R., Ullman, J. D., & Tsur, S. (1997). Dynamic Itemset Counting and Implication Rules for Market Basket Data. *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, 255–264.

Broomhead, D. S., & Lowe, D. (1988). Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems*, *2*, 321–355.

Brundage, M. P., Sexton, T., Hodkiewicz, M., Dima, A., & Lukens, S. (2021). Technical Language Processing: Unlocking Maintenance Knowledge. *Manufacturing Letters*, *27*, 42–46. https://doi.org/10.1016/j.mfglet.2020.11.001

Buuren, S. van, & Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, *45*(3), 1–67.

Caliński, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, *3*(1), 1–27.

Campello, R. J. G. B., Moulavi, D., & Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. *Advances in Knowledge Discovery and Data Mining (PAKDD 2013), Lecture Notes in Computer Science*, *7819*, 160–172.

Campos, G. O., Zimek, A., Sander, J., Campello, R. J. G. B., Micenková, B., Schubert, E., Assent, I., & Houle, M. E. (2016). On the Evaluation of Unsupervised Outlier Detection: Measures, Datasets, and an Empirical Study. *Data Mining and Knowledge Discovery*, *30*(4), 891–927. https://doi.org/10.1007/s10618-015-0444-8

Cavnar, W. B., & Trenkle, J. M. (1994). N-Gram-Based Text Categorization. *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR-94)*, 161–175.

Chan, K.-P., & Fu, A. W.-C. (1999). Efficient Time Series Matching by Wavelets. *Proceedings of the 15th International Conference on Data Engineering*, 126–133.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys*, *41*(3), 15:1–15:58.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0: Step-by-step data mining guide*. SPSS Inc.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Chen, D., Sain, S. L., & Guo, K. (2012). Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining. *Journal of Database Marketing & Customer Strategy Management*, *19*(3), 197–208.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

Cho, K., Merriënboer, B. van, Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. https://doi.org/10.3115/v1/D14-1179

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of Official Statistics*, *6*(1), 3–73.

Cleveland, W. S., & McGill, R. (1984). Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods. *Journal of the American Statistical Association*, *79*(387), 531–554.
Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, *13*(6), 377–387.
Codd, E. F., Codd, S. B., & Salley, C. T. (1993). *Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate*. Codd & Date, Inc.
Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, *20*(3), 273–297.
Cover, T. M., & Hart, P. E. (1967). Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27.
Cox, D. R. (1958). The Regression Analysis of Binary Sequences. *Journal of the Royal Statistical Society: Series B*, *20*(2), 215–242.
Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*(4), 303–314.
Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., & Keogh, E. (2019). The UCR Time Series Archive. *IEEE/CAA Journal of Automatica Sinica*, *6*(6), 1293–1305. https://doi.org/10.1109/JAS.2019.1911747
Daubechies, I. (1992). *Ten Lectures on Wavelets*. SIAM.
Davies, D. L., & Bouldin, D. W. (1979). A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-1*(2), 224–227.
Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407.
Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). *Mathematics for Machine Learning*. Cambridge University Press.
Dempster, A., Petitjean, F., & Webb, G. I. (2020). ROCKET: Exceptionally Fast and Accurate Time Series Classification Using Random Convolutional Kernels. *Data Mining and Knowledge Discovery*, *34*(5), 1454–1495. https://doi.org/10.1007/s10618-020-00701-z
Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society: Series B*, *39*(1), 1–38.
Demšar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, *7*, 1–30.
Devriendt, F., Moldovan, D., & Verbeke, W. (2018). A Literature Survey and Experimental Evaluation of the State-of-the-Art in Uplift Modeling: A Stepping Stone Toward the Development of Prescriptive Analytics. *Big Data*, *6*(1), 13–41. https://doi.org/10.1089/big.2017.0104
Dickey, D. A., & Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Journal of the American Statistical Association*, *74*(366a), 427–431.
Domingos, P., & Pazzani, M. (1997). On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, *29*(2–3), 103–130.

Dorgo, G., & Abonyi, J. (2018). Sequence Mining Based Alarm Suppression. *IEEE Access*, *6*.

Downs, J. J., & Vogel, E. F. (1993). A plant-wide industrial process control problem. *Computers & Chemical Engineering*, *17*(3), 245–255.

Dunn, J. C. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, *3*(3), 32–57.

Eckart, C., & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, *1*(3), 211–218.

EEMUA. (2013). *EEMUA Publication 191: Alarm Systems – A Guide to Design, Management and Procurement* (3. kiad.). Engineering Equipment; Materials Users Association.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 226–231.

Európai Parlament és Tanács. (2016). *Az Európai Parlament és a Tanács (EU) 2016/679 rendelete (2016. április 27.) a természetes személyeknek a személyes adatok kezelése tekintetében történő védelméről és az ilyen adatok szabad áramlásáról (általános adatvédelmi rendelet)*. Az Európai Unió Hivatalos Lapja, L 119, 2016.5.4., 1–88. o.

Európai Parlament és Tanács. (2024). *Az Európai Parlament és a Tanács (EU) 2024/1689 rendelete (2024. június 13.) a mesterséges intelligenciára vonatkozó harmonizált szabályok megállapításáról (a mesterséges intelligenciáról szóló rendelet)*. Az Európai Unió Hivatalos Lapja, L sorozat, 2024/1689, 2024.7.12.

Fader, P. S., Hardie, B. G. S., & Lee, K. L. (2005a). „Counting Your Customers" the Easy Way: An Alternative to the Pareto/NBD Model. *Marketing Science*, *24*(2), 275–284.

Fader, P. S., Hardie, B. G. S., & Lee, K. L. (2005b). RFM and CLV: Using Iso-Value Curves for Customer Base Analysis. *Journal of Marketing Research*, *42*(4), 415–430. https://doi.org/10.1509/jmkr.2005.42.4.415

Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast Subsequence Matching in Time-Series Databases. *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, 419–429.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, *17*(3), 37–54.

Fisher, A., Rudin, C., & Dominici, F. (2019). All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. *Journal of Machine Learning Research*, *20*(177), 1–81.

Fogler, H. S. (2016). *Elements of Chemical Reaction Engineering* (5. kiad.). Prentice Hall.

Fortuna, L., Graziani, S., Rizzo, A., & Xibilia, M. G. (2007). *Soft Sensors for Monitoring and Control of Industrial Processes*. Springer.

Freedman, D., & Diaconis, P. (1981). On the histogram as a density estimator: L_2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, *57*(4), 453–476.

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, *29*(5), 1189–1232.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, *46*(4), 44:1–44:37.

Ge, Z., Song, Z., & Gao, F. (2013). Review of Recent Research on Data-Based Process Monitoring. *Industrial & Engineering Chemistry Research*, *52*(10), 3543–3562.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, *4*(1), 1–58.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 249–256.

Goldstein, M., & Uchida, S. (2016). A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLoS ONE*, *11*(4), e0152173.

Golub, G. H., & Van Loan, C. F. (2013). *Matrix Computations* (4. kiad.). Johns Hopkins University Press.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Gower, J. C. (1971). A General Coefficient of Similarity and Some of Its Properties. *Biometrics*, *27*(4), 857–871.

Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., & Pirahesh, H. (1997). Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Mining and Knowledge Discovery*, *1*(1), 29–53.

Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems 35 (Datasets and Benchmarks Track)*.

Gupta, S., Hanssens, D., Hardie, B., Kahn, W., Kumar, V., Lin, N., Ravishanker, N., & Sriram, S. (2006). Modeling Customer Lifetime Value. *Journal of Service Research*, *9*(2), 139–155.

Gustafson, D. E., & Kessel, W. C. (1979). Fuzzy clustering with a fuzzy covariance matrix. *Proceedings of the IEEE Conference on Decision and Control*, 761–766.

Gutierrez, P., & Gérardy, J.-Y. (2017). Causal Inference and Uplift Modelling: A Review of the Literature. *Proceedings of the 3rd International Conference on Predictive Applications and APIs, Proceedings of Machine Learning Research*, *67*, 1–13.

Guyon, I., & Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Han, J., Pei, J., & Tong, H. (2022). *Data Mining: Concepts and Techniques* (4. kiad.). Morgan Kaufmann.

Han, J., Pei, J., & Yin, Y. (2000). Mining Frequent Patterns without Candidate Generation. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 1–12.

Han, S., Hu, X., Huang, H., Jiang, M., & Zhao, Y. (2022). ADBench: Anomaly Detection Benchmark. *Advances in Neural Information Processing Systems 35 (Datasets and Benchmarks Track)*, 32142–32159.

Harinarayan, V., Rajaraman, A., & Ullman, J. D. (1996). Implementing Data Cubes Efficiently. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, 205–216.

Harris, C. R. és mtsai. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362.

Harris, Z. S. (1954). Distributional Structure. *Word*, *10*(2–3), 146–162. https://doi.org/10.1080/00437956.1954.11659520

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2. kiad.). Springer.

Hawkins, D. M. (1980). *Identification of Outliers*. Chapman; Hall. https://doi.org/10.1007/978-94-015-3994-4

He, H., & Garcia, E. A. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1026–1034.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, *22*(1), 5–53. https://doi.org/10.1145/963770.963772

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, *313*(5786), 504–507.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, *12*(1), 55–67.

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, *20*(1), 5–10.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, *4*(2), 251–257.

Horst, A. M., Hill, A. P., & Gorman, K. B. (2020). *palmerpenguins: Palmer Archipelago (Antarctica) penguin data*.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, *24*(6), 417–441.

Hotelling, H. (1947). Multivariate Quality Control, Illustrated by the Air Testing of Sample Bombsights. In C. Eisenhart, M. W. Hastay, & W. A. Wallis (Szerk.), *Techniques of Statistical Analysis* (o. 111–184). McGraw-Hill.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*(1), 193–218.

Hughes, A. M. (1994). *Strategic Database Marketing*. Probus.

Hundman, K., Constantinou, V., Laporte, C., Colwell, I., & Soderstrom, T. (2018). Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 387–395. https://doi.org/10.1145/3219819.3219845

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95.

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (3. kiad.). OTexts. https://otexts.com/fpp3/

Hyndman, R. J., & Koehler, A. B. (2006). Another Look at Measures of Forecast Accuracy. *International Journal of Forecasting*, *22*(4), 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001

IEC. (2013). *IEC 62264-1:2013 Enterprise-Control System Integration – Part 1: Models and Terminology*. International Electrotechnical Commission.

Inmon, W. H. (2005). *Building the Data Warehouse* (4. kiad.). Wiley.

Inselberg, A. (1985). The plane with parallel coordinates. *The Visual Computer*, *1*(2), 69–91.

International Society of Automation. (2016). *ANSI/ISA-18.2-2016: Management of Alarm Systems for the Process Industries*. ISA.

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the 32nd International Conference on Machine Learning*, 448–456.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep Learning for Time Series Classification: A Review. *Data Mining and Knowledge Discovery*, *33*(4), 917–963. https://doi.org/10.1007/s10618-019-00619-1

Jackson, J. E., & Mudholkar, G. S. (1979). Control Procedures for Residuals Associated with Principal Component Analysis. *Technometrics*, *21*(3), 341–349.

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An Introduction to Statistical Learning with Applications in Python*. Springer. https://doi.org/10.1007/978-3-031-38747-0

Jardine, A. K. S., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, *20*(7), 1483–1510.

Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. *Machine Learning: ECML-98, Lecture Notes in Computer Science*, *1398*, 137–142.

Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A*, *374*(2065), 20150202.

Kadlec, P., Gabrys, B., & Strandt, S. (2009). Data-driven Soft Sensors in the process industry. *Computers & Chemical Engineering*, *33*(4), 795–814.

Kaufman, L., & Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley.

Kaufman, S., Rosset, S., Perlich, C., & Stitelman, O. (2012). Leakage in Data Mining: Formulation, Detection, and Avoidance. *ACM Transactions on Knowledge*

Discovery from Data, *6*(4), 15:1–15:21.
Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems 30*, 3146–3154.
Keogh, E., Chakrabarti, K., Pazzani, M., & Mehrotra, S. (2001). Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems*, *3*(3), 263–286.
Keogh, E., Chu, S., Hart, D., & Pazzani, M. (2004). Segmenting Time Series: A Survey and Novel Approach. In M. Last, A. Kandel, & H. Bunke (Szerk.), *Data Mining in Time Series Databases* (o. 1–21). World Scientific.
Keogh, E., & Ratanamahatana, C. A. (2005). Exact indexing of dynamic time warping. *Knowledge and Information Systems*, *7*(3), 358–386.
Killick, R., Fearnhead, P., & Eckley, I. A. (2012). Optimal Detection of Changepoints With a Linear Computational Cost. *Journal of the American Statistical Association*, *107*(500), 1590–1598. https://doi.org/10.1080/01621459.2012.737745
Kimball, R., & Caserta, J. (2004). *The Data Warehouse ETL Toolkit*. Wiley.
Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (3. kiad.). Wiley.
Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.
Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2021). 1D Convolutional Neural Networks and Applications: A Survey. *Mechanical Systems and Signal Processing*, *151*, 107398. https://doi.org/10.1016/j.ymssp.2020.107398
Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1137–1145.
Kohavi, R., & John, G. H. (1997). Wrappers for Feature Subset Selection. *Artificial Intelligence*, *97*(1–2), 273–324.
Kohavi, R., Tang, D., & Xu, Y. (2020). *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge University Press.
Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*(1), 59–69.
Kohonen, T. (2001). *Self-Organizing Maps* (3. kiad.). Springer.
Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, *42*(8), 30–37. https://doi.org/10.1109/MC.2009.263
Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, *37*(2), 233–243.
Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, *29*(1), 1–27.
Kumar, V., & Reinartz, W. (2018). *Customer Relationship Management: Concept, Strategy, and Tools* (3. kiad.). Springer.
Künzel, S. R., Sekhon, J. S., Bickel, P. J., & Yu, B. (2019). Metalearners for Estimating Heterogeneous Treatment Effects Using Machine Learning. *Proceedings of*

the National Academy of Sciences, *116*(10), 4156–4165. https://doi.org/10.1073/pnas.1804597116
Lance, G. N., & Williams, W. T. (1967). A general theory of classificatory sorting strategies: 1. Hierarchical systems. *The Computer Journal*, *9*(4), 373–380.
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.
LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.
Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*(6755), 788–791.
Lee, G. R., Gommers, R., Waselewski, F., Wohlfahrt, K., & O'Leary, A. (2019). PyWavelets: A Python package for wavelet analysis. *Journal of Open Source Software*, *4*(36), 1237.
Lei, Y., Li, N., Guo, L., Li, N., Yan, T., & Lin, J. (2018). Machinery Health Prognostics: A Systematic Review from Data Acquisition to RUL Prediction. *Mechanical Systems and Signal Processing*, *104*, 799–834. https://doi.org/10.1016/j.ymssp.2017.11.016
Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, *2*(2), 164–168.
Levy, O., & Goldberg, Y. (2014). Neural Word Embedding as Implicit Matrix Factorization. *Advances in Neural Information Processing Systems 27*, 2177–2185.
Leys, C., Ley, C., Klein, O., Bernard, P., & Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, *49*(4), 764–766.
Li, W., Yue, H. H., Valle-Cervantes, S., & Qin, S. J. (2000). Recursive PCA for Adaptive Process Monitoring. *Journal of Process Control*, *10*(5), 471–486. https://doi.org/10.1016/S0959-1524(00)00022-6
Lin, J., Keogh, E., Wei, L., & Lonardi, S. (2007). Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, *15*(2), 107–144.
Linden, G., Smith, B., & York, J. (2003). Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, *7*(1), 76–80. https://doi.org/10.1109/MIC.2003.1167344
Little, R. J. A., & Rubin, D. B. (2019). *Statistical Analysis with Missing Data* (3. kiad.). Wiley.
Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *Proceedings of the 8th IEEE International Conference on Data Mining*, 413–422.
Ljung, G. M., & Box, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika*, *65*(2), 297–303.
Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137.
Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization. *International Conference on Learning Representations (ICLR)*.
Maaten, L. van der, & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605.

MacGregor, J. F., & Kourti, T. (1995). Statistical process control of multivariate processes. *Control Engineering Practice*, *3*(3), 403–414.
MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, *1*, 281–297.
Mahalanobis, P. C. (1936). On the generalised distance in statistics. *Proceedings of the National Institute of Sciences of India*, *2*(1), 49–55.
Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The M4 Competition: Results, Findings, Conclusion and Way Forward. *International Journal of Forecasting*, *34*(4), 802–808. https://doi.org/10.1016/j.ijforecast.2018.06.001
Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
Mariscal, G., Marbán, Ó., & Fernández, C. (2010). A survey of data mining and knowledge discovery process models and methodologies. *The Knowledge Engineering Review*, *25*(2), 137–166.
Marquardt, D. W. (1963). An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, *11*(2), 431–441.
McCallum, A., & Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification. *AAAI-98 Workshop on Learning for Text Categorization*, 41–48.
McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133.
McInnes, L., Healy, J., & Melville, J. (2018). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. arXiv:1802.03426.
McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, 56–61.
McKinney, W. (2022). *Python for Data Analysis* (3. kiad.). O'Reilly.
Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing Order into Texts. *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 404–411.
Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv:1301.3781.
Minsky, M., & Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press.
Molnar, C. (2022). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* (2. kiad.). Independently published. https://christophm.github.io/interpretable-ml-book/
Montgomery, D. C. (2019). *Introduction to Statistical Quality Control* (8. kiad.). Wiley.
Montgomery, D. C., Peck, E. A., & Vining, G. G. (2021). *Introduction to Linear Regression Analysis* (6. kiad.). Wiley.
Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 807–814.
Ngai, E. W. T., Xiu, L., & Chau, D. C. K. (2009). Application of data mining techniques in customer relationship management: A literature review and classifi-

cation. *Expert Systems with Applications*, *36*(2), 2592–2602.
Niculescu-Mizil, A., & Caruana, R. (2005). Predicting Good Probabilities with Supervised Learning. *Proceedings of the 22nd International Conference on Machine Learning*, 625–632.
Orosz, G., Szabó, G., Berkecz, P., Szántó, Z., & Farkas, R. (2023). *Advancing Hungarian Text Processing with HuSpaCy: Efficient and Accurate NLP Pipelines*. arXiv:2308.12635.
Orosz, G., Szántó, Z., Berkecz, P., Szabó, G., & Farkas, R. (2022). *HuSpaCy: an industrial-strength Hungarian natural language processing toolkit*. arXiv:2201.01956.
Page, E. S. (1954). Continuous Inspection Schemes. *Biometrika*, *41*(1/2), 100–115. https://doi.org/10.2307/2333009
Pang, G., Shen, C., Cao, L., & Hengel, A. van den. (2021). Deep Learning for Anomaly Detection: A Review. *ACM Computing Surveys*, *54*(2), 38:1–38:38. https://doi.org/10.1145/3439950
Park, J. S., Chen, M.-S., & Yu, P. S. (1995). An Effective Hash-Based Algorithm for Mining Association Rules. *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, 175–186.
Pasquier, N., Bastide, Y., Taouil, R., & Lakhal, L. (1999). Discovering Frequent Closed Itemsets for Association Rules. *Database Theory – ICDT'99, Lecture Notes in Computer Science*, *1540*, 398–416.
Paszke, A. és mtsai. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems 32*, 8024–8035.
Payne, A., & Frow, P. (2005). A Strategic Framework for Customer Relationship Management. *Journal of Marketing*, *69*(4), 167–176.
Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
Pearson, K. (1901). On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine*, *2*(11), 559–572.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.
Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, M.-C. (2001). PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. *Proceedings of the 17th International Conference on Data Engineering*, 215–224.
Piatetsky-Shapiro, G. (1991). Discovery, Analysis, and Presentation of Strong Rules. In G. Piatetsky-Shapiro & W. J. Frawley (Szerk.), *Knowledge Discovery in Databases* (o. 229–248). AAAI/MIT Press.
Polyak, B. T. (1964). Some Methods of Speeding up the Convergence of Iteration Methods. *USSR Computational Mathematics and Mathematical Physics*, *4*(5), 1–17. https://doi.org/10.1016/0041-5553(64)90137-5
Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.
Prechelt, L. (1998). Early Stopping – But When? In *Neural Networks: Tricks of the*

Trade (Köt. 1524, o. 55–69). Springer.
Prince, S. J. D. (2023). *Understanding Deep Learning*. MIT Press.
Qin, S. J. (2012). Survey on data-driven industrial process monitoring and diagnosis. *Annual Reviews in Control*, *36*(2), 220–234.
Qin, S. J. (2014). Process Data Analytics in the Era of Big Data. *AIChE Journal*, *60*(9), 3092–3100. https://doi.org/10.1002/aic.14523
Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, *1*(1), 81–106.
Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.
Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., & Keogh, E. (2012). Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping. *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 262–270.
Raschka, S. (2018). MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *Journal of Open Source Software*, *3*(24), 638.
Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, *22*(3), 400–407. https://doi.org/10.1214/aoms/1177729586
Roberts, S. W. (1959). Control Chart Tests Based on Geometric Moving Averages. *Technometrics*, *1*(3), 239–250. https://doi.org/10.1080/00401706.1959.10489860
Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408.
Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53–65.
Rousseeuw, P. J., & Van Driessen, K. (1999). A Fast Algorithm for the Minimum Covariance Determinant Estimator. *Technometrics*, *41*(3), 212–223.
Rubin, D. B. (1976). Inference and Missing Data. *Biometrika*, *63*(3), 581–592.
Ruff, L., Kauffmann, J. R., Vandermeulen, R. A., Montavon, G., Samek, W., Kloft, M., Dietterich, T. G., & Müller, K.-R. (2021). A Unifying Review of Deep and Shallow Anomaly Detection. *Proceedings of the IEEE*, *109*(5), 756–795. https://doi.org/10.1109/JPROC.2021.3052449
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.
Saito, N. (1994). *Local Feature Extraction and Its Applications Using a Library of Bases* [Phdthesis]. Yale University.
Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLoS ONE*, *10*(3), e0118432.
Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *26*(1), 43–49.
Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523.

Sammon, J. W. (1969). A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computers*, *C-18*(5), 401–409.
Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-Based Collaborative Filtering Recommendation Algorithms. *Proceedings of the 10th International Conference on World Wide Web*, 285–295. https://doi.org/10.1145/371920.372071
Schmittlein, D. C., Morrison, D. G., & Colombo, R. (1987). Counting Your Customers: Who Are They and What Will They Do Next? *Management Science*, *33*(1), 1–24. https://doi.org/10.1287/mnsc.33.1.1
Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, *13*(7), 1443–1471.
Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., & Dennison, D. (2015). Hidden Technical Debt in Machine Learning Systems. *Advances in Neural Information Processing Systems 28*, 2503–2511.
Seabold, S., & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with Python. *Proceedings of the 9th Python in Science Conference*, 92–96.
Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, *27*(3), 379–423.
Shewhart, W. A. (1931). *Economic Control of Quality of Manufactured Product*. D. Van Nostrand.
Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall.
Spärck Jones, K. (1972). A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation*, *28*(1), 11–21. https://doi.org/10.1108/eb026526
Spearman, C. (1904). The Proof and Measurement of Association between Two Things. *The American Journal of Psychology*, *15*(1), 72–101.
Srikant, R., & Agrawal, R. (1995). Mining Generalized Association Rules. *Proceedings of the 21st International Conference on Very Large Data Bases*, 407–419.
Srikant, R., & Agrawal, R. (1996a). Mining Quantitative Association Rules in Large Relational Tables. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, 1–12.
Srikant, R., & Agrawal, R. (1996b). Mining Sequential Patterns: Generalizations and Performance Improvements. *Advances in Database Technology – EDBT'96, Lecture Notes in Computer Science*, *1057*, 3–17.
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.
Stevens, S. S. (1946). On the Theory of Scales of Measurement. *Science*, *103*(2684), 677–680.
Stone, M. (1974). Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society: Series B*, *36*(2), 111–147.
Stonebraker, M., Abadi, D. J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., Lau, E., Lin, A., Madden, S., O'Neil, E., O'Neil, P., Rasin, A., Tran, N., & Zdonik, S.

(2005). C-Store: A Column-oriented DBMS. *Proceedings of the 31st International Conference on Very Large Data Bases*, 553–564.
Strang, G. (2016). *Introduction to Linear Algebra* (5. kiad.). Wellesley-Cambridge Press.
Strang, G. (2019). *Linear Algebra and Learning from Data*. Wellesley-Cambridge Press.
Studer, S., Bui, T. B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S., & Müller, K.-R. (2021). Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology. *Machine Learning and Knowledge Extraction*, *3*(2), 392–413. https://doi.org/10.3390/make3020020
Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the Importance of Initialization and Momentum in Deep Learning. *Proceedings of the 30th International Conference on Machine Learning (ICML), Proceedings of Machine Learning Research*, *28*, 1139–1147.
Sweeney, L. (2002). k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *10*(5), 557–570.
Tan, P.-N., Kumar, V., & Srivastava, J. (2004). Selecting the right objective measure for association analysis. *Information Systems*, *29*(4), 293–313.
Tan, P.-N., Steinbach, M., Karpatne, A., & Kumar, V. (2019). *Introduction to Data Mining* (2. kiad.). Pearson.
Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2019). Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics*, *15*(4), 2405–2415. https://doi.org/10.1109/TII.2018.2873186
Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B*, *58*(1), 267–288.
Torgerson, W. S. (1952). Multidimensional scaling: I. Theory and method. *Psychometrika*, *17*(4), 401–419.
Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., & Altman, R. B. (2001). Missing value estimation methods for DNA microarrays. *Bioinformatics*, *17*(6), 520–525.
Tufte, E. R. (2001). *The Visual Display of Quantitative Information* (2. kiad.). Graphics Press.
Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley.
VanderPlas, J. (2016). *Python Data Science Handbook*. O'Reilly.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, *30*, 5998–6008.
Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. N., & Yin, K. (2003). A review of process fault detection and diagnosis: Part III: Process history based methods. *Computers & Chemical Engineering*, *27*(3), 327–346.
Virtanen, P. és mtsai. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272.
Wang, J., Yang, F., Chen, T., & Shah, S. L. (2016). An Overview of Industrial Alarm Systems: Main Causes for Alarm Overloading, Research Status, and Open Problems. *IEEE Transactions on Automation Science and Engineering*, *13*(2),

1045–1061.
Wang, R. Y., & Strong, D. M. (1996). Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems*, *12*(4), 5–33.
Wang, Z., Yan, W., & Oates, T. (2017). Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline. *2017 International Joint Conference on Neural Networks (IJCNN)*, 1578–1585. https://doi.org/10.1109/IJCNN.2017.7966039
Ward, J. H. (1963). Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, *58*(301), 236–244.
Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021.
Wasserman, L. (2004). *All of Statistics: A Concise Course in Statistical Inference*. Springer.
Wattenberg, M., Viégas, F., & Johnson, I. (2016). How to Use t-SNE Effectively. *Distill*. https://distill.pub/2016/misread-tsne/
Wickham, H. (2014). Tidy Data. *Journal of Statistical Software*, *59*(10), 1–23.
Wilkinson, L. (2005). *The Grammar of Graphics* (2. kiad.). Springer.
Winkler, R., Klawonn, F., & Kruse, R. (2011). Fuzzy C-Means in High Dimensional Spaces. *International Journal of Fuzzy System Applications*, *1*(1), 1–16.
Winters, P. R. (1960). Forecasting Sales by Exponentially Weighted Moving Averages. *Management Science*, *6*(3), 324–342.
Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a Standard Process Model for Data Mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, 29–39.
Wold, S., Sjöström, M., & Eriksson, L. (2001). PLS-Regression: A Basic Tool of Chemometrics. *Chemometrics and Intelligent Laboratory Systems*, *58*(2), 109–130. https://doi.org/10.1016/S0169-7439(01)00155-1
Xie, X. L., & Beni, G. (1991). A Validity Measure for Fuzzy Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *13*(8), 841–847.
Yeo, I.-K., & Johnson, R. A. (2000). A New Family of Power Transformations to Improve Normality or Symmetry. *Biometrika*, *87*(4), 954–959.
Zaki, M. J. (2000). Scalable Algorithms for Association Mining. *IEEE Transactions on Knowledge and Data Engineering*, *12*(3), 372–390.
Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press.
Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Addison-Wesley.
Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, *67*(2), 301–320.

A. Python-alapok

Miről szól a fejezet?

Ez a függelék a könyv példáinak megértéséhez és továbbfejlesztéséhez szükséges Python- ismereteket foglalja össze, programozási előismeretek nélkül is követhető módon. A nyelv alapelemei után a tudományos számítások négy alapkönyvtárát – a NumPy-t, a pandast, a Matplotlibet és a `scikit-learn`-t – mutatjuk be, majd a könyv saját segédcsomagját és a reprodukálható munka gyakorlatát. A részletesebb elsajátításhoz VanderPlas (2016) és McKinney (2022) könyve ajánlott.

A.1. A munkakörnyezet

A könyv minden példája három módon futtatható:

- **Google Colab** – a böngészőben futó, telepítést nem igénylő Jupyter-környezet. A fejezetek végén található „Colab” gombok a fejezet kódját tartalmazó jegyzetfüzetet nyitják meg; az első cella telepíti a könyv segédcsomagját.
- **Saját gépen, Jupyterrel** – a Python 3.11 vagy újabb változatának telepítése után egy virtuális környezetben:

```
python -m venv .venv
source .venv/bin/activate           # Windows: .venv\Scripts\activate
pip install -r requirements.txt      # a könyv tárolójából
pip install -e .                     # a segédcsomag fejlesztői
↪  telepítése
jupyter lab
```

- **A könyv forrásából** – a tároló klónozása után a `quarto render` parancs a teljes könyvet újrafordítja, és közben minden kódot lefuttat.

A Jupyter-jegyzetfüzet **cellákból** áll: a kódcellák eredménye (szöveg, táblázat, ábra) közvetlenül alattuk jelenik meg. A cellák tetszőleges sorrendben futtathatók,

ami kísérletezéshez kényelmes, de hibaforrás is: a jegyzetfüzetet a végén mindig érdemes „újraindítás és minden futtatása” paranccsal ellenőrizni.

A.2. A nyelv alapelemei

A.2.1. Változók és alapvető típusok

A Python dinamikusan típusos nyelv: a változó típusát az értéke határozza meg.

```
homerseklet = 86.5            # lebegőpontos szám (float)
termekkod = "B"               # karakterlánc (str)
tetelszam = 14                # egész szám (int)
megfelelo = True              # logikai érték (bool)
print(type(homerseklet), type(termekkod), type(tetelszam), type(megfelelo))
print(f"A(z) {termekkod} termék hőmérséklete {homerseklet:.1f} °C,
 ↪  tételszám: {tetelszam}.")
```

```
<class 'float'> <class 'str'> <class 'int'> <class 'bool'>
A(z) B termék hőmérséklete 86.5 °C, tételszám: 14.
```

Az utolsó sor egy **f-string**: a kapcsos zárójelek közötti kifejezés értéke behelyettesítődik, a kettőspont utáni rész (`.1f`) a formázást adja meg. A Python a tizedesjelölésre pontot használ; a könyv szövegében – a magyar helyesírásnak megfelelően – tizedesvesszőt írunk, a kódban és a kimenetekben viszont pont szerepel.

A.2.2. Gyűjtemények

A négy beépített gyűjteménytípus a lista (rendezett, módosítható), a rendezett n-es (tuple, nem módosítható), a szótár (kulcs–érték párok) és a halmaz (egyedi elemek).

```
meresek = [82.1, 85.3, 86.0, 84.7]                    # lista
koordinata = (47.09, 17.91)                           # rendezett n-es
alapjel = {"A": 82, "B": 86, "C": 90}                 # szótár
kosar = {"kávébab", "vízkőoldó", "kávébab"}           # halmaz: a
 ↪  duplikátum eltűnik
meresek.append(87.2)
print(meresek[0], meresek[-1], meresek[1:3])          # indexelés 0-tól,
 ↪  szeletelés
print(alapjel["B"], sorted(kosar), len(kosar))
```

```
82.1 87.2 [85.3, 86.0]
86 ['kávébab', 'vízkőoldó'] 2
```

A.2.3. Vezérlési szerkezetek és függvények

```
def minosit(T, termek, tures=1.5):
    """Megfelel-e a hőmérséklet a termék alapjelének a tűréshatáron
    ↪ belül?"""
    elteres = T - alapjel[termek]
    if abs(elteres) <= tures:
        return "megfelelő"
    elif elteres > 0:
        return "túl meleg"
    else:
        return "túl hideg"

for T in meresek:
    print(T, minosit(T, "B"))
```

```
82.1 túl hideg
85.3 megfelelő
86.0 megfelelő
84.7 megfelelő
87.2 megfelelő
```

A Pythonban a kódblokkokat a **behúzás** jelöli. A függvény elején álló karakterlánc a dokumentációs szöveg (docstring), amelyet a `help(minosit)` jelenít meg. A **listaértelmezés** (list comprehension) tömör és gyors módja egy lista előállításának:

```
elteresek = [round(T - alapjel["B"], 1) for T in meresek if T > 84]
print(elteresek)
```

```
[-0.7, 0.0, -1.3, 1.2]
```

A.3. NumPy: tömbök és vektorizált számítás

A NumPy (C. R. Harris és mtsai., 2020) alapobjektuma az azonos típusú elemekből álló, többdimenziós tömb (`ndarray`). A műveletek elemenként, gyorsan, C nyelven megírt ciklusokkal hajtódnak végre (**vektorizálás**), ezért a nagy adattömbökön végzett számításoknál a Python- ciklusok helyett tömbműveleteket használunk.

```
x = np.array([82.1, 85.3, 86.0, 84.7, 87.2])
A = np.arange(12).reshape(3, 4)                          # 3×4-es mátrix
print(x.mean(), x.std(ddof=1), x.max())                  # ddof=1: korrigált
 ↪ tapasztalati szórás
print(A.shape, A[1, :], A[:, 2], A[A > 6])               # sor, oszlop,
 ↪ logikai indexelés
print(A.sum(axis=0), A.sum(axis=1))                      # oszlop- és
 ↪ sorösszegek
```

```
85.05999999999999 1.8981570008827011 87.2
(3, 4) [4 5 6 7] [ 2  6 10] [ 7  8  9 10 11]
[12 15 18 21] [ 6 22 38]
```

A **broadcasting** szabályai szerint eltérő alakú tömbök is kombinálhatók, ha a dimenzióik összeillenek; így például egy mátrix minden oszlopa egy lépésben standardizálható:

```
rng = np.random.default_rng(2026)                        # reprodukálható
 ↪ véletlenszám-generátor
X = rng.normal(loc=[80, 2.0, 10], scale=[3, 0.1, 1], size=(500, 3))
Z = (X - X.mean(axis=0)) / X.std(axis=0)                 # (500,3) - (3,) →
 ↪ soronkénti kivonás
print(Z.mean(axis=0).round(3), Z.std(axis=0).round(3))
```

```
[ 0. -0. -0.] [1. 1. 1.]
```

A vektorizálás sebességelőnye jelentős:

```
import time
v = rng.normal(size=1_000_000)
t0 = time.perf_counter(); s1 = sum(e * e for e in v); t1 =
 ↪ time.perf_counter()
s2 = v @ v; t2 = time.perf_counter()
print(f"Python-ciklus: {t1 - t0:.3f} s, NumPy: {t2 - t1:.4f} s, eltérés:
 ↪ {abs(s1 - s2):.2e}")
```

```
Python-ciklus: 0.104 s, NumPy: 0.0008 s, eltérés: 1.08e-08
```

A mátrixszorzás operátora a `@`; a lineáris algebra további eszközei (inverz, sajátérték-felbontás, SVD) a `np.linalg` modulban vannak (B. függelék).

A.4. pandas: táblázatos adatok

A pandas (McKinney, 2010) két alapobjektuma a címkézett egydimenziós `Series` és a kétdimenziós, oszloponként eltérő típusú `DataFrame`. Az adatbányászati munka nagy része – beolvasás, tisztítás, szűrés, csoportosítás, összekapcsolás – pandasban történik.

```
tetelek = ab.load_shop().transactions
print(tetelek.shape)
tetelek.head()
```

```
(68990, 8)
```

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	CustomerID	Regio
0	S240018	10032	Pléd	2	2024-01-01 07:21:00	9990.0	12003	Közép-Magyarország
1	S240018	10040	Ajándékutalvány 10 000 Ft	1	2024-01-01 07:21:00	10000.0	12003	Közép-Magyarország
2	S247721	10029	Teamécses 50 db	1	2024-01-01 08:27:00	1990.0	13489	Nyugat-Dunántúl
3	S247721	10031	Díszpárna	1	2024-01-01 08:27:00	5990.0	13489	Nyugat-Dunántúl
4	S242901	10003	Kávékapszula 30 db	1	2024-01-01 09:14:00	3290.0	12566	Közép-Dunántúl

```
# szűrés, új oszlop, csoportosítás és aggregálás
t = tetelek[(tetelek.Quantity > 0) & (tetelek.Price > 0)].copy()
t["Ertek"] = t.Quantity * t.Price
(t.groupby("Regio")
  .agg(szamlak=("Invoice", "nunique"), forgalom=("Ertek", "sum"))
  .sort_values("forgalom", ascending=False)
  .head(5))
```

Regio	szamlak	forgalom
Közép-Magyarország	5220	677311026.0
Észak-Alföld	1717	239874396.0
Dél-Alföld	1627	215667452.0
Dél-Dunántúl	1323	201934582.0
Közép-Dunántúl	1537	195137904.0

```
# kimutatás (pivot) és időbeli újramintavételezés
havi = t.set_index("InvoiceDate").Ertek.resample("MS").sum() / 1e6
print(havi.head(4).round(2))
uzem = ab.load_plant().hourly.set_index("ido")
print(uzem.T_reaktor.rolling("24h").mean().describe().round(2))
```

```
InvoiceDate
2024-01-01    61.01
```

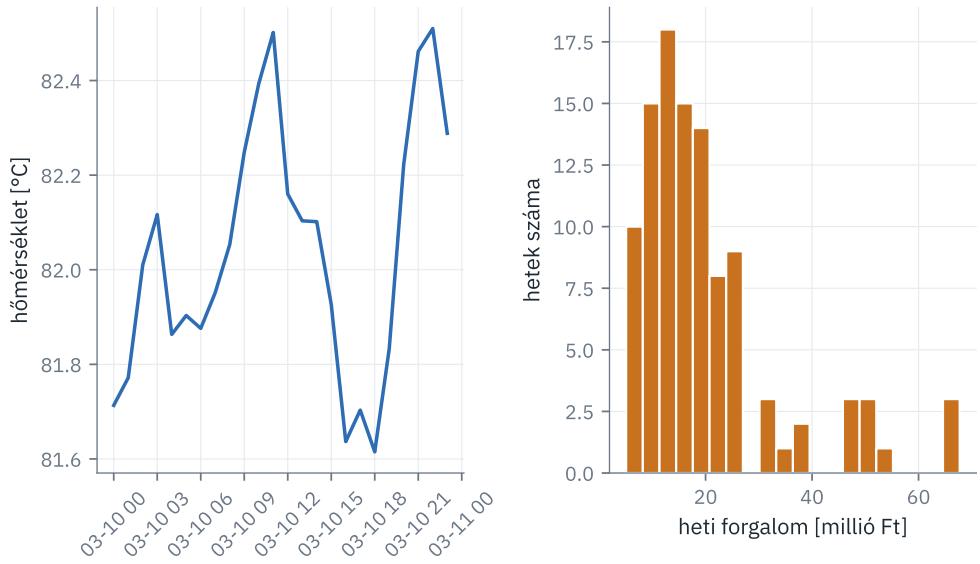
```
2024-02-01    49.65
2024-03-01    71.31
2024-04-01    33.22
Freq: MS, Name: Ertek, dtype: float64
count    8760.00
mean       85.82
std         3.17
min        80.80
25%        82.12
50%        85.96
75%        89.84
max        90.86
Name: T_reaktor, dtype: float64
```

A legfontosabb műveletek: `read_csv`/`read_parquet` (beolvasás), `loc`/`iloc` (címke, illetve pozíció szerinti kiválasztás), `query` (feltételes szűrés), `assign` (új oszlopok), `groupby` + `agg` (csoportos összesítés), `merge` (összekapcsolás, mint az SQL `JOIN`), `pivot_table` (kimutatás), `resample` és `rolling` (idősorok), `isna`/`fillna`/`dropna` (hiányzó értékek). A láncolt írásmód – mint a fenti példában – olvashatóvá teszi az adatfeldolgozási lépések sorozatát.

A.5. Matplotlib: ábrák

A Matplotlib (Hunter, 2007) objektumorientált felülete egy ábrából (`Figure`) és egy vagy több koordináta-rendszerből (`Axes`) áll; a könyv ábrái a segédcsomag `set_style()` függvényével beállított egységes stílust használják. A statisztikai ábrákhoz a seaborn (Waskom, 2021) magasabb szintű függvényeket ad.

```
fig, ax = plt.subplots(1, 2)
nap = uzem.loc["2025-03-10", "T_reaktor"]
ax[0].plot(nap.index, nap.values, color=ab.PALETTE[0])
ax[0].set_ylabel("hőmérséklet [°C]"); ax[0].tick_params(axis="x",
↪  rotation=45)
heti = t.set_index("InvoiceDate").Ertek.resample("W").sum() / 1e6
ax[1].hist(heti, bins=20, color=ab.PALETTE[1], edgecolor="white")
ax[1].set_xlabel("heti forgalom [millió Ft]"); ax[1].set_ylabel("hetek
↪  száma")
fig.tight_layout()
```



A.1. ábra. Egyszerű Matplotlib-ábra két koordináta-rendszerrel: a reaktorhőmérséklet egy napja és a heti forgalom eloszlása.

A.6. scikit-learn: a gépi tanulás egységes felülete

A `scikit-learn` (Pedregosa és mtsai., 2011) minden modellje ugyanazt a felületet követi: a konstruktor a hiperparamétereket kapja, a `fit(X, y)` tanít, a `predict(X)` (osztályozóknál a `predict_proba(X)` is) előrejelez, az előfeldolgozó lépések pedig `fit` és `transform` metódusokkal rendelkeznek. A `Pipeline` a lépéseket egyetlen modellé fűzi, így az előfeldolgozás paraméterei is csak a tanítóadatokból becsülődnek – ez akadályozza meg az információszivárgást (3. fejezet).

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score

churn = ab.churn_table()
X = churn[["R_nap", "F_szamla", "M_ertek"]].astype(float)
y = churn.lemorzsolodik
modell = make_pipeline(StandardScaler(), LogisticRegression())
print(cross_val_score(modell, np.log1p(X), y, cv=5,
 ↪  scoring="roc_auc").round(3))
```

```
[0.776 0.799 0.814 0.756 0.756]
```

A.7. A könyv segédcsomagja

A könyv kódja az `adatbanyaszat` nevű segédcsomagra épül, amelyet a példák elején `import adatbanyaszat as ab` formában töltünk be. A csomag a szimulált futó esetek generátorait (D. függelék), a fejezeteken átívelő előfeldolgozási lépéseket és az ábrák stílusát tartalmazza; így minden fejezet ugyanazokra, a korábbi fejezetekben már megtisztított adatokra építhet.

A.1. táblázat. A segédcsomag legfontosabb függvényei.

Függvény	Tartalom	Bevezetve
`ab.load_shop()`	a PannonShop nyers adatai: tételek, termékek, vevők, vélemények, háttérigazság	1. fejezet
`ab.load_plant()`	a Pannon üzem adatai: 10 perces és órás mérések, riasztások, labor, háttérigazság	1. fejezet
`ab.load_shift_log()`	a Pannon üzem műszaknaplója	13. fejezet
`ab.shop_clean()`	a megtisztított számlatételek (sztornók nélkül)	3. fejezet
`ab.plant_clean()`	a megtisztított 10 perces mérések, leállásjelzővel	3. fejezet
`ab.customer_table(ref_date)`	vevőszintű RFM- és vásárlásiszerkezet-jellemzők	3. fejezet
`ab.churn_table(cutoff, horizon_days)`	lemorzsolódási tábla jellemzőkkel és célváltozóval	7. fejezet
`ab.quality_table()`	a laborminták és a hozzájuk tartozó folyamatjellemzők	7. fejezet
`ab.mini.*`	a kézi számolási példák kis adatsorai	több fejezet
`ab.set_style()`, `ab.PALETTE`	a könyv ábrastílusa és színtévesztés-biztos színpalettája	A.5

A generátorok determinisztikusak: ugyanazzal a véletlenmaggal (alapértelmezés: 2026) mindig ugyanazokat az adatokat állítják elő, ezért a könyv minden eredménye pontosan reprodukálható.

A.8. Jó gyakorlatok

- **Reprodukálhatóság**: rögzítsük a véletlenmagot (`np.random.default_rng(mag)`, `random_state=mag`), a csomagok verzióit (`requirements.txt`), és a jegyzetfüzetet futtassuk végig az elejétől.
- **Verziókezelés**: a kódot és a dokumentációt `git`-tárolóban tartsuk; a nagy adatfájlokat ne, azokat a generáló kód vagy egy adattár hivatkozása írja le.
- **Olvasható kód**: beszédes változónevek, rövid függvények docstringgel, a „varázsszámok” helyett nevesített paraméterek.
- **Tesztelés**: a kézi példák eredményeit (például egy gradiens numerikus ellenőrzését) írjuk meg automatikus tesztként is.

- **Jegyzetfüzet és modul**: a felfedező elemzés jegyzetfüzetben, a többször használt kód modulban (`.py` fájlban) a helyén.

B. Lineáris algebra az adatbányászathoz

Miről szól a fejezet?

Az adatbányászat módszereinek nagy része lineáris algebrai műveletekre épül: az adathalmaz egy mátrix, a regresszió egy vetítés, a főkomponens-elemzés egy sajátérték-feladat, a neurális hálózat mátrixszorzások és nemlineáris függvények sorozata. Ez a függelék a könyvben használt fogalmakat foglalja össze az alapoktól a szinguláris érték szerinti felbontásig és a mátrixos deriválásig, minden fogalomnál megmutatva, hol használjuk és hogyan számolható NumPy-jal. Részletesebb tárgyalás Strang (2016), Strang (2019) és – kifejezetten a gépi tanulás szemszögéből – Deisenroth és mtsai. (2020) könyvében található.

B.1. Vektorok

Egy n-dimenziós **vektor** n valós szám rendezett listája, amelyet oszlopvektorként írunk: $\mathbf{x} = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$. Az adatbányászatban egy objektum (vevő, mérési időpont) jellemzőinek vektora a **jellemzőtér** egy pontja. A vektorok összeadása és számmal való szorzása elemenként történik; geometriailag az összeadás a paralelogramma-szabály, a skalárral szorzás nyújtás vagy zsugorítás.

A két vektor **skaláris szorzata**

$$\mathbf{x}^\top \mathbf{y} = \sum_{i=1}^{n} x_i y_i = \|\mathbf{x}\| \, \|\mathbf{y}\| \cos\theta, \tag{B.1}$$

ahol θ a két vektor által bezárt szög. Ha a skaláris szorzat nulla, a vektorok **merőlegesek** (ortogonálisak). A (B.1) összefüggés adja a koszinusz-hasonlóságot (13. fejezet), és a skaláris szorzat a lineáris modellek ($\mathbf{w}^\top \mathbf{x} + b$) alapművelete.

A vektor hosszát a **norma** méri. A leggyakoribbak az ℓ_p-normák:

$$\|\mathbf{x}\|_1 = \sum_i |x_i|, \qquad \|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2} = \sqrt{\mathbf{x}^\top \mathbf{x}}, \qquad \|\mathbf{x}\|_\infty = \max_i |x_i|. \tag{B.2}$$

Két pont távolsága a különbségvektor normája: az ℓ_2 az euklideszi, az ℓ_1 a Manhattan-, az ℓ_∞ a Csebisev-távolság (5. fejezet). A regularizációban az ℓ_2-norma négyzete a ridge-, az ℓ_1-norma a lasszó-büntetés (8. fejezet).

```
x, y = np.array([3.0, 4.0]), np.array([4.0, 3.0])
print(x @ y, np.linalg.norm(x), np.linalg.norm(x, 1), np.linalg.norm(x,
 ↪  np.inf))
print("koszinusz:", x @ y / (np.linalg.norm(x) * np.linalg.norm(y)))
```

```
24.0 5.0 7.0 4.0
koszinusz: 0.96
```

Az $\mathbf{x}$ vektor $\mathbf{u}$ irányú (egységvektorra) **vetülete** $(\mathbf{u}^\top \mathbf{x})\,\mathbf{u}$: ez az $\mathbf{u}$ egyenesének az $\mathbf{x}$-hez legközelebbi pontja, és a maradék, $\mathbf{x} - (\mathbf{u}^\top \mathbf{x})\mathbf{u}$, merőleges $\mathbf{u}$-ra. A főkomponens-elemzés ilyen vetítésekkel dolgozik.

B.2. Mátrixok és műveleteik

Egy $m \times n$-es **mátrix** m sorba és n oszlopba rendezett szám. Az adatmátrix $\mathbf{X} \in \mathbb{R}^{N \times n}$ sorai az objektumok, oszlopai a jellemzők. A **transzponált** $\mathbf{X}^\top$ a sorokat és oszlopokat felcseréli.

A **mátrix–vektor szorzat** kétféleképpen értelmezhető: $\mathbf{Ax}$ i-edik eleme az $\mathbf{A}$ i-edik sorának és $\mathbf{x}$-nek a skaláris szorzata, ugyanakkor $\mathbf{Ax}$ az $\mathbf{A}$ oszlopainak $\mathbf{x}$ elemeivel súlyozott lineáris kombinációja is:

$$\mathbf{Ax} = x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + \cdots + x_n \mathbf{a}_n. \tag{B.3}$$

A második értelmezés a kulcs a regresszióhoz: a $\mathbf{Xw}$ előrejelzés mindig az $\mathbf{X}$ oszlopai által kifeszített altérben van. A **mátrixszorzat** $\mathbf{C} = \mathbf{AB}$ elemei $c_{ij} = \sum_k a_{ik} b_{kj}$; a szorzás asszociatív, de általában nem kommutatív ($\mathbf{AB} \neq \mathbf{BA}$), és $(\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top$. Egy mátrix lineáris leképezésként is felfogható: a $\mathbf{x} \mapsto \mathbf{Ax}$ leképezés a teret forgatja, nyújtja, vetíti; két leképezés egymás utáni alkalmazása a mátrixok szorzata. A neurális hálózatok rétegei ilyen leképezések nemlineáris függvényekkel váltakozva (10. fejezet).

A **kovarianciamátrix** a centrált adatmátrixból számolható: $\mathbf{S} = \frac{1}{N-1}\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$, ahol $\tilde{\mathbf{X}}$ oszlopai nulla átlagúak. Szimmetrikus, és átlójában a varianciák állnak.

```
rng = np.random.default_rng(0)
X = rng.normal(size=(200, 3)) @ np.array([[1.0, 0.8, 0.0], [0.0, 0.6, 0.0],
 ↪  [0.0, 0.0, 0.5]])
Xc = X - X.mean(axis=0)
S = Xc.T @ Xc / (len(X) - 1)
print(S); print(np.allclose(S, np.cov(X, rowvar=False)))
```

```
[[ 1.033  0.819 -0.047]
 [ 0.819  1.011 -0.046]
 [-0.047 -0.046  0.239]]
True
```

B.3. Lineáris függetlenség, rang, inverz

A $\mathbf{v}_1, \dots, \mathbf{v}_k$ vektorok **lineárisan függetlenek**, ha egyikük sem állítható elő a többiek lineáris kombinációjaként. Egy mátrix **rangja** a lineárisan független oszlopainak (egyben sorainak) maximális száma. Ha egy adatmátrix oszlopai lineárisan összefüggők – például egy jellemző két másik összege, vagy a kategóriaarányok összege 1 –, a mátrix nem teljes rangú, és ez a regresszióban multikollinearitáshoz vezet (8. fejezet).

Egy $n \times n$-es négyzetes mátrix **invertálható**, ha van olyan $\mathbf{A}^{-1}$, amelyre $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$; ez pontosan akkor teljesül, ha a mátrix teljes rangú, vagyis a **determinánsa** nem nulla. A determináns abszolút értéke azt mutatja meg, hányszorosára változtatja a leképezés a térfogatot; a nulla determináns azt jelenti, hogy a leképezés egy alacsonyabb dimenziós altérbe „lapítja” a teret, és így nem fordítható vissza. Numerikusan az inverz kiszámítását kerüljük: a $\mathbf{A}\mathbf{x} = \mathbf{b}$ egyenletrendszert közvetlenül oldjuk meg (`np.linalg.solve`), mert ez gyorsabb és pontosabb. A közel szinguláris mátrixok érzékenységét a **kondíciószám** (a legnagyobb és a legkisebb szinguláris érték hányadosa) méri.

B.4. Legkisebb négyzetek és vetítés

Ha az $\mathbf{X}\mathbf{w} = \mathbf{y}$ egyenletrendszernek nincs pontos megoldása (több egyenlet, mint ismeretlen), a **legkisebb négyzetes** megoldást keressük, amely a $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$ eltérést minimalizálja. Geometriailag ez $\mathbf{y}$ merőleges vetítése az $\mathbf{X}$ oszlopai által kifeszített altérre: a maradék merőleges minden oszlopra, $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}) = \mathbf{0}$, amiből a **normálegyenletek**:

$$\mathbf{X}^\top \mathbf{X}\,\hat{\mathbf{w}} = \mathbf{X}^\top \mathbf{y}, \qquad \hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}, \qquad \hat{\mathbf{y}} = \underbrace{\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top}_{\mathbf{H}}\,\mathbf{y}. \tag{B.4}$$

A **H vetítőmátrix** szimmetrikus és idempotens ($\mathbf{H}^2 = \mathbf{H}$). A gyakorlatban a megoldást QR- vagy SVD-felbontással számoljuk (`np.linalg.lstsq`), mert a $\mathbf{X}^\top\mathbf{X}$ kondíciószáma az $\mathbf{X}$-énak a négyzete.

```
Xd = np.column_stack([np.ones(6), [1, 2, 3, 4, 5, 6]])
yd = np.array([2.1, 3.9, 6.2, 7.8, 10.1, 12.2])
w_normal = np.linalg.solve(Xd.T @ Xd, Xd.T @ yd)
w_lstsq, *_ = np.linalg.lstsq(Xd, yd, rcond=None)
H = Xd @ np.linalg.solve(Xd.T @ Xd, Xd.T)
print(w_normal, w_lstsq, np.allclose(H @ H, H), np.round(Xd.T @ (yd - H @
 ↪  yd), 10))
```

```
[-0.02  2.02] [-0.02  2.02] True [0. 0.]
```

B.5. Sajátértékek és sajátvektorok

Egy négyzetes $\mathbf{A}$ mátrix **sajátvektora** olyan nem nulla $\mathbf{v}$ vektor, amelynek irányát a leképezés nem változtatja meg, csak a hosszát:

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v}, \tag{B.5}$$

ahol λ a hozzá tartozó **sajátérték**. A sajátértékek a $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$ karakterisztikus egyenlet gyökei.

A **szimmetrikus** mátrixok – mint a kovarianciamátrix – különösen fontosak. A **spektráltétel** szerint sajátértékeik valósak, sajátvektoraik páronként merőlegesek, és a mátrix felbontható

$$\mathbf{A} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^\top \tag{B.6}$$

alakban, ahol $\mathbf{V}$ oszlopai az ortonormált sajátvektorok, $\boldsymbol{\Lambda}$ a sajátértékek diagonális mátrixa. A kovarianciamátrix sajátvektorai a **főkomponensek**, sajátértékei a főkomponensek mentén mért varianciák (4. fejezet). Egy szimmetrikus mátrix **pozitív szemidefinit**, ha minden $\mathbf{x}$-re $\mathbf{x}^\top\mathbf{A}\mathbf{x} \geq 0$ – ez pontosan akkor teljesül,

ha minden sajátértéke nemnegatív; minden kovarianciamátrix ilyen. Ha minden sajátérték pozitív, a mátrix **pozitív definit** és invertálható.

Az $\mathbf{x}^\top\mathbf{A}\mathbf{x}$ **kvadratikus forma** szintvonalai pozitív definit $\mathbf{A}$ esetén ellipszisek (ellipszoidok), amelyek tengelyei a sajátvektorok irányába mutatnak, és hossza $1/\sqrt{\lambda_i}$-vel arányos. A Mahalanobis-távolság (11. fejezet) a $\boldsymbol{\Sigma}^{-1}$ mátrixú kvadratikus forma: a $\boldsymbol{\Sigma}$ sajátvektorai mentén az ottani szórással skálázott távolság.

```
lam, V = np.linalg.eigh(S)                    # szimmetrikus mátrixra: eigh
↪  (növekvő sorrend)
print("sajátértékek:", lam)
print("rekonstrukció:", np.allclose(V @ np.diag(lam) @ V.T, S), "|
↪  ortonormált:", np.allclose(V.T @ V, np.eye(3)))
```

```
sajátértékek: [0.202 0.236 1.844]
rekonstrukció: True | ortonormált: True
```

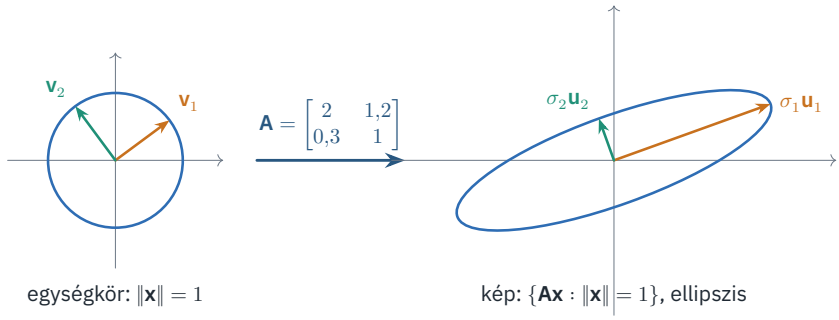
B.6. Szinguláris érték szerinti felbontás

Bármely $m \times n$-es mátrix – nem csak a négyzetes és szimmetrikus – felbontható

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top \tag{B.7}$$

alakban (**szinguláris érték szerinti felbontás**, SVD), ahol $\mathbf{U}$ ($m \times m$) és $\mathbf{V}$ ($n \times n$) ortogonális mátrixok, $\boldsymbol{\Sigma}$ pedig diagonális, a nemnegatív, csökkenő sorrendű $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$ **szinguláris értékekkel**; r a mátrix rangja. Geometriailag minden lineáris leképezés egy forgatás ($\mathbf{V}^\top$), egy tengelyek menti nyújtás ($\boldsymbol{\Sigma}$) és egy újabb forgatás ($\mathbf{U}$) egymásutánja: az egységkört ellipszisbe viszi, amelynek féltengelyei a szinguláris értékek (B.1. ábra). A szinguláris értékek négyzetei az $\mathbf{A}^\top\mathbf{A}$ sajátértékei, $\mathbf{V}$ oszlopai pedig a sajátvektorai.

Az SVD legfontosabb tulajdonsága az **optimális alacsony rangú közelítés**: az Eckart–Young-tétel (Eckart & Young, 1936) szerint az első k tag összege, $\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$, a legjobb k rangú közelítés Frobenius- és spektrálnormában is, és a közelítés hibája $\|\mathbf{A} - \mathbf{A}_k\|_F^2 = \sum_{i>k} \sigma_i^2$. Erre épül a főkomponens-elemzés (a centrált adatmátrix SVD-je: a jobb oldali szinguláris vektorok a főkomponensek, $\sigma_i^2/(N-1)$ a varianciájuk), a látens szemantikus elemzés (13. fejezet), a mátrixfaktorizációs ajánlórendszerek (14. fejezet) és a numerikusan stabil legkisebb négyzetes megoldás. A **pszeudoinverz** $\mathbf{A}^+ = \mathbf{V}\boldsymbol{\Sigma}^+\mathbf{U}^\top$ (ahol $\boldsymbol{\Sigma}^+$ a nem



B.1. ábra. A szinguláris érték szerinti felbontás geometriája. A mátrix az egységkört ellipszisbe képezi; a $\mathbf{v}_i$ jobb oldali szinguláris vektorok képei a $\sigma_i \mathbf{u}_i$ féltengelyek.

nulla szinguláris értékek reciprokait tartalmazza) nem teljes rangú mátrixokra is megadja a legkisebb normájú legkisebb négyzetes megoldást.

```
U, s, Vt = np.linalg.svd(Xc, full_matrices=False)
print("szinguláris értékek:", s)
print("σ²/(N−1) = kovariancia-sajátértékek:", s ** 2 / (len(Xc) - 1))
X1 = s[0] * np.outer(U[:, 0], Vt[0])            # a legjobb 1 rangú
 ↪ közelítés
print("relatív hiba (1 rang):", np.linalg.norm(Xc - X1) /
 ↪ np.linalg.norm(Xc),
      "| elméleti:", np.sqrt((s[1:] ** 2).sum() / (s ** 2).sum()))
```

```
szinguláris értékek: [19.156  6.854  6.345]
σ²/(N−1) = kovariancia-sajátértékek: [1.844 0.236 0.202]
relatív hiba (1 rang): 0.43825376548048633 | elméleti: 0.43825376548048633
```

B.7. Mátrixos deriválás

A gépi tanulás modelljeinek paramétereit egy célfüggvény minimalizálásával határozzuk meg, amihez a célfüggvény paraméterek szerinti **gradiensére** van szükség: egy $f : \mathbb{R}^n \to \mathbb{R}$ függvény gradiense a parciális deriváltak vektora, $\nabla f(\mathbf{x}) = (\partial f/\partial x_1, \dots, \partial f/\partial x_n)^\top$, amely a függvény leggyorsabb növekedésének irányába mutat. A könyvben használt azonosságok ($\mathbf{a}$, $\mathbf{A}$ állandók):

$$\nabla_{\mathbf{x}}(\mathbf{a}^\top \mathbf{x}) = \mathbf{a}, \qquad \nabla_{\mathbf{x}}(\mathbf{x}^\top \mathbf{A}\mathbf{x}) = (\mathbf{A} + \mathbf{A}^\top)\mathbf{x}, \qquad \nabla_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 = -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}). \tag{B.8}$$

Az utolsó kifejezést nullával egyenlővé téve kapjuk a (B.4) normálegyenleteket. Vektorértékű függvények deriváltja a **Jacobi-mátrix** ($J_{ij} = \partial f_i / \partial x_j$); a **láncszabály** összetett függvényekre a Jacobi-mátrixok szorzata, $\partial(\mathbf{f} \circ \mathbf{g})/\partial \mathbf{x} = \mathbf{J_f}\,\mathbf{J_g}$. A neurális hálózatok hibavisszaterjesztése (10. fejezet) ennek a szabálynak a hatékony, a kimenettől a bemenet felé haladó alkalmazása. A másodrendű deriváltak mátrixa, a **Hesse-mátrix**, a konvexitást jellemzi: ha minden pontban pozitív szemidefinit, a függvény konvex, és minden lokális minimuma globális (ilyen a lineáris és a logisztikus regresszió célfüggvénye).

Egy analitikusan levezetett gradiens helyessége **numerikusan ellenőrizhető**:

```
def f(w):
    return np.sum((yd - Xd @ w) ** 2)
w0, h = np.array([0.5, 1.5]), 1e-6
analitikus = -2 * Xd.T @ (yd - Xd @ w0)
numerikus = np.array([(f(w0 + h * e) - f(w0 - h * e)) / (2 * h) for e in
 ↪  np.eye(2)])
print(analitikus, numerikus)
```

```
[-15.6 -72.8] [-15.6 -72.8]
```

B.8. Összefoglaló táblázat

B.1. táblázat. A könyvben használt lineáris algebrai eszközök.

Fogalom	NumPy	Hol használjuk
skaláris szorzat, norma	`x @ y`, `np.linalg.norm`	távolságok, koszinusz-hasonlóság, regularizáció
mátrixszorzat	`A @ B`	lineáris modellek, neurális hálók
egyenletrendszer, legkisebb négyzetek	`np.linalg.solve`, `np.linalg.lstsq`	lineáris regresszió
szimmetrikus sajátérték-felbontás	`np.linalg.eigh`	főkomponens-elemzés, Mahalanobis-távolság
szinguláris érték szerinti felbontás	`np.linalg.svd`	PCA, LSA, ajánlórendszerek, pszeudoinverz
determináns, rang, kondíciószám	`np.linalg.det`, `matrix_rank`, `cond`	multikollinearitás, numerikus stabilitás

C. Valószínűségszámítási és statisztikai alapok

Miről szól a fejezet?

Az adatbányászat módszerei bizonytalan, zajos adatokból következtetnek, ezért nyelvük a valószínűségszámítás és a statisztika. Ez a függelék a könyvben használt fogalmakat foglalja össze: a valószínűségi változókat és eloszlásaikat, a feltételes valószínűséget és a Bayes-tételt, a többdimenziós normális eloszlást, a paraméterbecslést (különösen a maximum-likelihood módszert), valamint a konfidenciaintervallumokat, a hipotézisvizsgálatot és a bootstrapet. Részletesebb, a gépi tanulás szemszögéből írt tárgyalás Wasserman (2004) és Deisenroth és mtsai. (2020) könyvében található.

C.1. Valószínűségi változók és eloszlások

A **valószínűségi változó** egy véletlen kísérlet kimeneteléhez rendelt szám. Diszkrét változónál a $P(X = k)$ valószínűségek, folytonosnál a $p(x)$ sűrűségfüggvény írja le az eloszlást (utóbbinál $P(a \le X \le b) = \int_a^b p(x)\,dx$). A **várható érték** és a **variancia**

$$E[X] = \sum_k k\,P(X = k) \quad \text{vagy} \quad \int x\,p(x)\,dx, \qquad \mathrm{Var}(X) = E\big[(X - E[X])^2\big] = E[X^2] - E[$$

(C.1)

A várható érték lineáris: $E[aX + bY] = aE[X] + bE[Y]$; a variancia független változókra additív: $\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y)$. Két változó **kovarianciája** $\mathrm{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$, a **korrelációs együttható** ennek a szórások szorzatával normált változata, -1 és 1 közötti érték, amely a *lineáris* kapcsolat erősségét méri (4. fejezet).

A könyvben előforduló legfontosabb eloszlások:

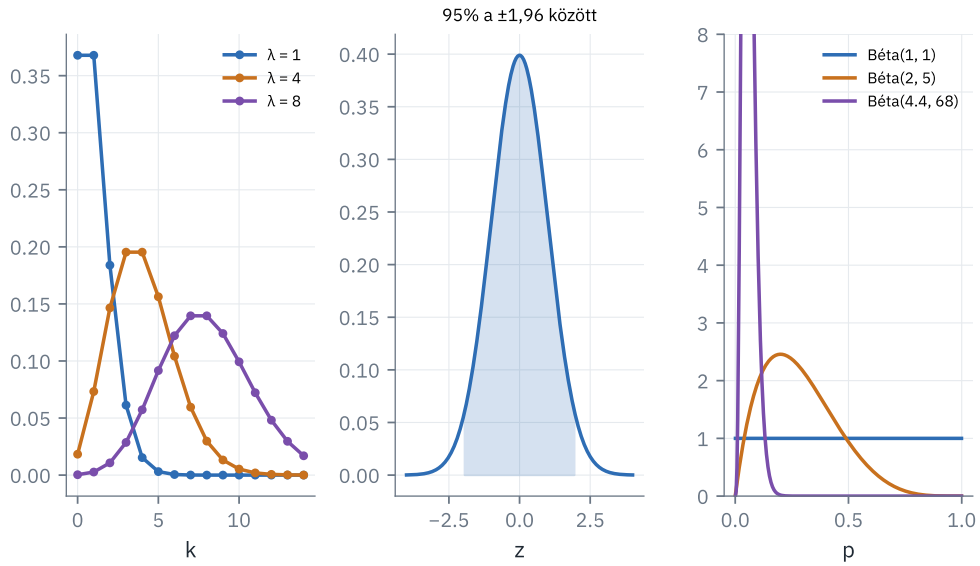
C.1. táblázat. A könyvben használt fontosabb eloszlások.

Eloszlás	Leírás	$E[X]$	$\mathrm{Var}(X)$	Előfordulás a könyvben
Bernoulli(p)	egyetlen igen/nem kimenet	p	$p(1-p)$	bináris osztályozás, lemorzsolódás
binomiális(n, p)	sikerek száma n kísérletben	np	$np(1-p)$	arányok, A/B teszt
Poisson(λ)	események száma egységnyi idő alatt	λ	λ	vásárlások (BG/NBD), riasztások
normális(μ, σ^2)	sok kis hatás összege	μ	σ^2	mérési zaj, regresszió hibatagja
exponenciális(λ)	várakozási idő két esemény között	$1/\lambda$	$1/\lambda^2$	eseményközi idők
gamma(r, α)	pozitív, ferde eloszlás	r/α	r/α^2	vevők közötti heterogenitás
béta(a, b)	valószínűségek eloszlása $[0,1]$-en	$a/(a+b)$	$\frac{ab}{(a+b)^2(a+b+1)}$	lemorzsolódási valószínűség (BG/NBD)
Dirichlet($\boldsymbol{\alpha}$)	valószínűségvektorok eloszlása	$\alpha_k / \sum_j \alpha_j$	–	témamodellek (LDA)

```
fig, ax = plt.subplots(1, 3)
k = np.arange(0, 15)
for lam, c in zip([1, 4, 8], ab.PALETTE):
    ax[0].plot(k, stats.poisson.pmf(k, lam), "o-", ms=3, color=c,
 ↪  label=f"λ = {lam}")
ax[0].legend(fontsize=7); ax[0].set_xlabel("k")
x = np.linspace(-4, 4, 300)
ax[1].plot(x, stats.norm.pdf(x), color=ab.PALETTE[0])
ax[1].fill_between(x, stats.norm.pdf(x), where=np.abs(x) <= 1.96,
 ↪  color=ab.PALETTE[0], alpha=0.2)
ax[1].set_xlabel("z"); ax[1].set_title("95% a ±1,96 között", fontsize=8)
p = np.linspace(0.001, 0.999, 300)
for (a, b), c in zip([(1, 1), (2, 5), (4.4, 68)], ab.PALETTE):
    ax[2].plot(p, stats.beta.pdf(p, a, b), color=c, label=f"Béta({a},
 ↪  {b})")
ax[2].set_ylim(0, 8); ax[2].legend(fontsize=7); ax[2].set_xlabel("p")
fig.tight_layout()
```

C.2. Feltételes valószínűség és a Bayes-tétel

Az A esemény B feltétel melletti **feltételes valószínűsége** $P(A \mid B) = P(A \cap B)/P(B)$. Két esemény **független**, ha $P(A \cap B) = P(A)P(B)$, vagyis ha B bekövetkezése nem változtat A valószínűségén. A **teljes valószínűség tétele** és a **Bayes-tétel**:



C.1. ábra. Néhány eloszlás: Poisson-eloszlás három intenzitással (balra), normális eloszlás sűrűségfüggvénye (középen) és béta-eloszlások (jobbra).

$$P(B) = \sum_i P(B \mid A_i)\,P(A_i), \qquad P(A_i \mid B) = \frac{P(B \mid A_i)\,P(A_i)}{\sum_j P(B \mid A_j)\,P(A_j)}. \tag{C.2}$$

A Bayes-tétel a „fordított” következtetés eszköze: a megfigyelt adatokból (B) a rejtett okokra (A_i) következtet, az a priori valószínűségeket a megfigyelés fényében a posteriori valószínűségekké frissítve. A naiv Bayes-osztályozó (9. fejezet) és a BG/NBD modell aktív-valószínűsége (14. fejezet) is ezen alapul.

Egy klasszikus példa megmutatja, miért fontos az alapgyakoriság. Egy anomáliaészlelő a csalások 95%-át jelzi, a normál rendeléseknek pedig 2%-ánál ad téves riasztást. Ha a rendelések 0,2%-a csalás, mekkora valószínűséggel csalás egy jelzett rendelés?

```
p_csalas, erzekenyseg, teves_arany = 0.002, 0.95, 0.02
p_jelzes = erzekenyseg * p_csalas + teves_arany * (1 - p_csalas)
print(f"P(csalás | jelzés) = {erzekenyseg * p_csalas / p_jelzes:.3f}")
```

```
P(csalás | jelzés) = 0.087
```

A jelzett rendeléseknek mindössze mintegy 9%-a csalás: ritka eseményeknél még egy pontos detektor riasztásainak többsége is téves – ezért a 11. fejezetben az AUC helyett a pontosságot és a lista elején mért találati arányt figyeltük.

C.3. A többdimenziós normális eloszlás

Az n-dimenziós normális eloszlás sűrűségfüggvénye

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}\,|\boldsymbol{\Sigma}|^{1/2}} \exp\!\Big(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big), \tag{C.3}$$

ahol $\boldsymbol{\mu}$ a várhatóérték-vektor, $\boldsymbol{\Sigma}$ a kovarianciamátrix. A kitevőben a Mahalanobis-távolság négyzete áll (11. fejezet); a sűrűség szintvonalai ellipszoidok, amelyek tengelyeit $\boldsymbol{\Sigma}$ sajátvektorai adják (B. függelék). Ha $\mathbf{x}$ n-dimenziós normális eloszlású, d_M^2 χ^2_n eloszlású – ez adja a Mahalanobis-távolságon alapuló küszöböket. A normális eloszlás lineáris transzformációja is normális, peremeloszlásai és feltételes eloszlásai is normálisak; a Gauss-keverék modellek (5. fejezet) ilyen eloszlások súlyozott összegei.

A **nagy számok törvénye** szerint a mintaátlag a várható értékhez tart, a **centrális határeloszlás-tétel** szerint pedig sok független, azonos eloszlású változó átlaga – eloszlásuktól függetlenül – közelítőleg normális eloszlású, $\sigma/\sqrt{N}$ szórással. Ez magyarázza a normális eloszlás kitüntetett szerepét a mérési hibák leírásában és a statisztikai próbákban.

C.4. Paraméterbecslés

C.4.1. Becslők tulajdonságai

Egy $\hat{\theta}$ becslő **torzítatlan**, ha $E[\hat{\theta}] = \theta$, és **konzisztens**, ha a mintaméret növelésével a valódi értékhez tart. A becslő átlagos négyzetes hibája a torzítás négyzetének és a varianciának az összege – ez a 7. fejezet torzítás–variancia felbontásának statisztikai megfelelője. A variancia $\hat{\sigma}^2 = \frac{1}{N}\sum_i (x_i - \bar{x})^2$ becslője torzított, mert a mintaátlag a mintához „igazodik”; a torzítatlan változat $N-1$-gyel oszt (**Bessel-korrekció**):

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2. \tag{C.4}$$

```
rng = np.random.default_rng(1)
minta = rng.normal(0, 1, size=(100_000, 5))              # 100 000 darab 5
↪  elemű minta, σ² = 1
print("1/N-es becslő átlaga:", minta.var(axis=1, ddof=0).mean().round(3),
      "| 1/(N-1)-es becslő átlaga:", minta.var(axis=1,
       ↪  ddof=1).mean().round(3))
```

```
1/N-es becslő átlaga: 0.798 | 1/(N-1)-es becslő átlaga: 0.998
```

A NumPy `var` és `std` függvényei alapértelmezésben N-nel osztanak (`ddof=0`), a pandas megfelelő metódusai $N-1$-gyel (`ddof=1`); a különbség kis mintákon számít.

C.4.2. Maximum-likelihood becslés

A **maximum-likelihood** (ML) becslés azt a paraméterértéket választja, amely mellett a megfigyelt adatok a legvalószínűbbek. Független megfigyeléseknél a **likelihood** a valószínűségek (sűrűségek) szorzata, és a számításokhoz annak logaritmusát használjuk:

$$\hat{\boldsymbol{\theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \; \ell(\boldsymbol{\theta}), \qquad \ell(\boldsymbol{\theta}) = \sum_{i=1}^{N} \ln p(x_i \mid \boldsymbol{\theta}). \tag{C.5}$$

Bernoulli-eloszlásnál (k siker N kísérletből) a $\ell(p) = k\ln p + (N-k)\ln(1-p)$ függvény deriváltját nullával egyenlővé téve $\hat{p} = k/N$; normális eloszlásnál $\hat{\mu} = \bar{x}$ és $\hat{\sigma}^2 = \frac{1}{N}\sum_i (x_i - \bar{x})^2$ – az utóbbi a fenti torzított becslő. A könyv számos módszere ML-becslés: a legkisebb négyzetek módszere normális hibatag esetén, a logisztikus regresszió (kereszt-entrópia), a Gauss-keverék modell EM-algoritmusa, a BG/NBD és a gamma–gamma modell. Ha nincs zárt alakú megoldás, a log-likelihoodot numerikusan maximalizáljuk:

```
from scipy.optimize import minimize_scalar
x =
↪  ab.load_plant().alarms.groupby(ab.load_plant().alarms.ido.dt.date).size()
napi = x.reindex(pd.date_range("2025-01-01", "2025-12-31").date,
↪  fill_value=0).to_numpy()
negloglik = lambda lam: -stats.poisson.logpmf(napi, lam).sum()
ml = minimize_scalar(negloglik, bounds=(0.01, 20), method="bounded")
print(f"a napi riasztásszám Poisson-intenzitásának ML-becslése: {ml.x:.3f}
↪  (a mintaátlag: {napi.mean():.3f})")
print(f"átlag {napi.mean():.2f}, variancia {napi.var(ddof=1):.2f} – a
↪  Poisson-modellben a kettő egyenlő volna")
```

```
a napi riasztásszám Poisson-intenzitásának ML-becslése: 1.658 (a mintaátlag:
↪  1.658)
átlag 1.66, variancia 5.05 – a Poisson-modellben a kettő egyenlő volna
```

A numerikus becslés megegyezik a zárt alakú megoldással (a mintaátlaggal). A variancia azonban jóval nagyobb az átlagnál: a riasztások csoportosan, hibaeseményekhez kötötten jelentkeznek, ezért a napi riasztásszám „túlszórt” – a Poisson-modell itt nem megfelelő, a negatív binomiális (gamma–Poisson keverék) eloszlás jobb leírást adna. A modell feltevéseinek ellenőrzése a becslés elválaszthatatlan része.

C.5. Konfidenciaintervallum, hipotézisvizsgálat, bootstrap

A **konfidenciaintervallum** a becslés bizonytalanságát fejezi ki: a 95%-os intervallum olyan eljárással készül, amely ismételt mintavételnél az esetek 95%-ában tartalmazza a valódi paramétert. Nagy mintán az átlag 95%-os intervalluma $\bar{x} \pm 1{,}96\, s/\sqrt{N}$; két arány különbségére a 14. fejezet (14.5) képletét használtuk.

A **hipotézisvizsgálat** egy nullhipotézist (például „a kampánynak nincs hatása”) vet össze az adatokkal. A **p-érték** annak valószínűsége, hogy a nullhipotézis fennállása esetén a megfigyeltnél legalább ennyire szélsőséges eredményt kapnánk; ha kisebb az előre rögzített szignifikanciaszintnél (rendszerint 0,05), a nullhipotézist elvetjük. A p-érték nem a nullhipotézis valószínűsége, és nem a hatás nagysága: nagy mintán egy gyakorlatilag jelentéktelen különbség is szignifikáns lehet, ezért mindig a hatás nagyságát és a konfidenciaintervallumot is meg kell adni. Sok próba egyidejű elvégzésekor a véletlen „felfedezések” száma nő (**többszörös összehasonlítás**); ilyenkor a szignifikanciaszintet korrigálni kell (például Bonferroni-korrekcióval) – ez az asszociációs szabályok és a modellválasztás értékelésénél is fontos.

A **bootstrap** a becslő eloszlását az adatokból, visszatevéses újramintavételezéssel közelíti: sok bootstrap-mintán kiszámítjuk a statisztikát, és az így kapott értékek eloszlásából (például a 2,5%-os és a 97,5%-os kvantilisből) adunk intervallumot. Előnye, hogy bonyolult statisztikákra (medián, AUC, egy modell hibája) is alkalmazható, zárt alakú képlet nélkül. A 9. fejezet zsákolása (bagging) ugyanezt az újramintavételezést használja.

```
vevok = ab.customer_table(ref_date="2025-07-01")
m = vevok.M_ertek.to_numpy()
boot = np.array([np.median(rng.choice(m, size=len(m), replace=True)) for _
↪  in range(2000)])
print(f"a vevőnkénti költés mediánja: {np.median(m):,.0f} Ft; "
```

```
        f"95%-os bootstrap-intervallum: {np.quantile(boot, 0.025):,.0f} –
        ↪  {np.quantile(boot, 0.975):,.0f} Ft")
```

```
a vevőnkénti költés mediánja: 158,150 Ft; 95%-os bootstrap-intervallum: 139,810 –
↪  174,590 Ft
```

D. Az adatsorok leírása

Miről szól a fejezet?

A könyv két futó esete – a PannonShop webáruház és a Pannon üzem – szimulált adatokra épül. Ez a függelék leírja az adatsorok szerkezetét, a generálásuk elvét és a beléjük épített, ellenőrzésre szolgáló „háttérigazságot”, végül olyan nyilvános valós adatsorokat ajánl, amelyeken a módszerek tovább gyakorolhatók.

D.1. Miért szimulált adatok?

A tananyag céljaihoz a szimulált adatoknak három előnye van. Először, a **háttérigazság ismert**: tudjuk, melyik vevő melyik típusba tartozik, melyik rendelés csalás, mikor milyen hiba lépett fel az üzemben, ezért minden módszer eredménye pontosan ellenőrizhető – ez a valós adatokon ritkán lehetséges. Másodszor, az adatok **szabadon felhasználhatók és reprodukálhatók**: a generátorok egy véletlenmagból mindig ugyanazokat az adatokat állítják elő, nincs adatvédelmi vagy licencakadály. Harmadszor, az adatokba **tudatosan építettük be** azokat a jelenségeket – adatminőségi hibákat, rejtett szegmenseket, asszociációs szabályokat, szezonalitást, lassú romlást, ritka hibákat –, amelyeken a módszerek tanulságai bemutathatók. Az ára az egyszerűsítés: a valós adatok változatosabbak, zajosabbak és meglepőbbek. Ezért a fejezetek a tanulságokat óvatosan fogalmazzák meg, és a függelék végén valós adatsorokat is ajánlunk.

A generátorok a segédcsomag `shop.py` és `plant.py` moduljában találhatók; az adatok CSV formátumban a könyv tárolójának `data/` mappájában is elérhetők (a `scripts/gen_data.py` állítja elő őket).

D.2. PannonShop: online kiskereskedelem

A PannonShop egy kávéval, teával, konyhai eszközökkel, háztartási gépekkel, dekorációs termékekkel, ajándékokkal és írószerekkel kereskedő magyar webáru-

ház. Az adatok szerkezete a nyilvános *Online Retail II* adatsort (D. Chen és mtsai., 2012) követi, így a módszerek közvetlenül átvihetők arra is.

D.1. táblázat. A PannonShop adattáblái (2024. január 1. – 2025. december 31.).

Tábla	Sorok	Fontosabb oszlopok
`transactions` (számlatételek)	68 990	`Invoice`, `StockCode`, `Description`, `Quantity`, `InvoiceDate`, `Price`, `CustomerID`, `Regio`
`products` (termékek)	48	`StockCode`, `Description`, `Kategoria`, `Alkategoria`, `ListaAr`
`customers` (vevők)	3 000	`CustomerID`, `Regio`, `Regisztracio`
`reviews` (vélemények)	3 000	`CustomerID`, `StockCode`, `Datum`, `Csillag`, `Szoveg`, `Hangulat`

A generálás elve a következő. A vevők öt rejtett típusba tartoznak (törzsvásárló, alkalmi vevő, akcióvadász, viszonteladó, lemorzsolódó), amelyek a vásárlási gyakoriságban, a kosárméretben, a kategóriapreferenciákban, az akciókra való érzékenységben és a lemorzsolódás időzítésében különböznek. A vevők egy része már az időszak elején aktív, a többiek fokozatosan regisztrálnak. A kosarak tartalmát a kategóriapreferenciák mellett beültetett asszociációs szabályok alakítják (például *kávéfőző gép* $\Rightarrow$ *kávébab*, *teáskanna* $\Rightarrow$ *zöld tea*). Az év végén szezonális csúcs, a 2., 3., 47. és 48. héten akciós időszak van.

Az adatokba a 3. fejezet gyakorlatához szándékosan adatminőségi hibák kerültek: sztornózott számlák (C-előtagú számlaszám, negatív mennyiség), postaköltség-sorok, ismétlődő tételek, eltérő írásmódú termékmegnevezések, nulla árú és százszorosan elírt árú tételek, valamint vevőazonosító nélküli vendégvásárlások. Huszonöt csalásgyanús rendelés is található bennük (11. fejezet). A vélemények szempontonként (termékminőség, szállítás, ár, ügyfélszolgálat) összeállított mondatokból állnak, és a csillagszám a szempontok hangulatából adódik.

A háttérigazság (`truth`) tartalmazza a vevők valódi típusát, a lemorzsolódás időpontját, a beültetett szabályokat, a csalásgyanús és a sztornózott számlák listáját, valamint az akciós heteket. Ezeket a könyv kizárólag a módszerek *értékeléséhez* használja.

D.3. Pannon üzem: folyamatos vegyipari reaktor

A Pannon üzem egy folyamatos üzemű, kevert tankreaktor (CSTR), amelyben három termékminőséget (A, B, C) gyártanak kampányokban. A szimulátor egyszerűsített, de fizikailag konzisztens modellen alapul (Fogler, 2016). Az elsőrendű, exoterm reakció sebességi állandója az Arrhenius-egyenletet követi, a stacionárius konverzió

$$k(T) = k_0\, e^{-E_a/(RT)}, \qquad X = \frac{k\tau}{1 + k\tau}, \qquad \tau = \frac{V}{F}, \tag{D.1}$$

ahol V a reaktortérfogat, F a betáplálási áram. A reakcióhőt a hűtőköpeny vezeti el, amelynek hőátadási tényezője a lerakódás miatt két tisztítás között fokozatosan romlik; a szükséges hűtővízáramot a hőmérleg határozza meg, és ha az meghaladja a szelep kapacitását, a hűtés telítődik, és a reaktor túlmelegszik. A környezeti és a hűtővíz- hőmérséklet éves és napi ciklust követ.

D.2. táblázat. A Pannon üzem mért változói (10 perces mintavétel, 2025. január 1. – december 31.).

Változó	Jelentés	Mértékegység
`T_kornyezet`	környezeti hőmérséklet	°C
`F_betap`, `c_betap`, `T_betap`	betáplálási áram, koncentráció, hőmérséklet	m³/h, mol/L, °C
`T_reaktor`, `T_alapjel`	mért reaktorhőmérséklet és alapjele	°C
`F_hutoviz`, `T_hutoviz`	hűtővízáram és a hűtővíz hőmérséklete	m³/h, °C
`P_reaktor`, `P_kevero`	reaktornyomás, keverőteljesítmény	bar, kW
`E_fogyasztas`	villamosenergia-fogyasztás	kW
`c_termek_labor`	a termékkoncentráció laborértéke (csak a mintavételkor)	mol/L
`termek`	a gyártott termék (A, B, C)	–

Az adatsor elemei: 52 560 darab 10 perces mérés (`measurements`) és ezek órás átlagai (`hourly`); 2176 laborminta a termékkoncentrációval, a szennyezettséggel és a minősítéssel (`lab`); a riasztási napló (`alarms`), amelyet határérték- és változásalapú szabályok állítanak elő; és a műszaknapló (`load_shift_log()`), a kezelők szabad szöveges bejegyzéseivel. A mérésekben zaj, kiesett mérések és kiugró értékek is vannak.

A beültetett események: hét tisztítási leállás; három szenzor-drift (a hőmérsékletmérés öt nap alatt 4 °C-ig fokozatosan elmászik, a szabályozó pedig a hibás mérést tartja az alapjelen), két betáplálási zavar (a betáplált koncentráció 15%-kal csökken két napra), két szelepakadás (a hűtővízáram háromnapos oszcillációja) és nyolc ötórás hűtővíz-kiesés (a hűtővíz hőmérséklete 12 °C-kal megemelkedik). A háttérigazság a hibák pontos idejét, a lerakódás mértékét, a valódi reaktorhőmérsékletet és termékminőséget is tartalmazza.

D.4. Kis adatsorok a kézi számolásokhoz

A `ab.mini` modul a kézi számolási példák kis adatsorait tartalmazza: a döntési fa és a naiv Bayes-osztályozó tizennégy gyártási tételét (9. fejezet), a gyakori elemhalmazok nyolc kosarát (6. fejezet), a hierarchikus klaszterezés hét pontját (5. fe-

jezet), a regresszió hat pontját (8. fejezet), a DTW két jelalakját (12. fejezet) és a TF-IDF öt dokumentumát (13. fejezet).

D.5. Nyilvános valós adatsorok a további gyakorláshoz

D.3. táblázat. Nyilvános adatsorok a könyv módszereinek gyakorlásához.

Adatsor	Tartalom	Kapcsolódó fejezetek	Forrás, licenc
Online Retail II	egy brit webáruház két évnyi számlatétele	2–6., 14.	UCI Machine Learning Repository (D. Chen és mtsai., 2012), CC BY 4.0
Tennessee Eastman-folyamat	vegyipari referenciafolyamat 21 hibatípussal	11., 15.	Downs & Vogel (1993); számos nyilvános szimulált adatkészlet
Palmer-pingvinek	három pingvinfaj testméretei	4., 5., 9.	Horst és mtsai. (2020), CC0
UCR-idősorarchívum	több mint 120 idősor-osztályozási feladat	12.	Dau és mtsai. (2019)

A Palmer-pingvinek adatsora a segédcsomagból is elérhető (`ab.load_penguins()`). A valós adatsorok használatakor mindig ellenőrizzük a licencfeltételeket, és – személyes adatok esetén – az adatkezelés jogalapját (1. fejezet).

E. Jelölések

A könyv a gépi tanulási szakirodalom elterjedt jelöléseit követi (Bishop, 2006; Hastie és mtsai., 2009). Ahol egy jelölésnek a fejezet szövegkörnyezetében más jelentése van (például T egyszerre jelöli a hőmérsékletet, az idősor hosszát és a kezelési változót), azt a fejezet külön jelzi.

Általános jelölések

Jelölés	Jelentés
x, $\mathbf{x}$, $\mathbf{X}$	skalár, (oszlop)vektor, mátrix
$\mathbf{x}^\top$, $\mathbf{X}^\top$	transzponált
$\mathbf{I}$	egységmátrix
$\mathbf{X} \in \mathbb{R}^{N \times n}$	adatmátrix: N objektum (sor), n jellemző (oszlop)
$\mathbf{x}_k$	a k-adik objektum jellemzővektora (az $\mathbf{X}$ k-adik sora, oszlopvektorként)
x_{kj}	a k-adik objektum j-edik jellemzője
y_k, $\mathbf{y}$	a k-adik objektum célváltozója; a célváltozók vektora
$\hat{y}$, $\hat{\boldsymbol{\theta}}$	előrejelzés, illetve becsült paraméter
C, c	az osztályok száma, egy osztály
K	a klaszterek (komponensek, témák) száma
$\lVert\mathbf{x}\rVert_p$	ℓ_p-norma; index nélkül az euklideszi (ℓ_2) norma
$d(\mathbf{x}, \mathbf{y})$	távolság
$\mathbf{x}^\top \mathbf{y}$	skaláris szorzat
$\odot$	elemenkénti (Hadamard-) szorzat
$\mathrm{tr}(\mathbf{A})$, $\det(\mathbf{A})$, $\mathrm{rank}(\mathbf{A})$	nyom, determináns, rang
∇f, $\partial f / \partial x$	gradiens, parciális derivált
$\arg\min$, $\arg\max$	a minimumot, illetve maximumot adó argumentum
$\mathbb{1}[\cdot]$, $\delta_{x>0}$	indikátorfüggvény (1, ha a feltétel teljesül, egyébként 0)
$\ln$, $\log_2$	természetes, illetve kettes alapú logaritmus

Valószínűség és statisztika

Jelölés	Jelentés
$P(A)$, $P(A \mid B)$	valószínűség, feltételes valószínűség

Jelölés	Jelentés
$p(\mathbf{x})$	sűrűségfüggvény
$E[X]$, $\mathrm{Var}(X)$, $\mathrm{Cov}(X,Y)$	várható érték, variancia, kovariancia
$\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$	várhatóérték-vektor, kovarianciamátrix
$\bar{x}$, s, $\mathbf{S}$	mintaátlag, korrigált tapasztalati szórás, tapasztalati kovarianciamátrix
$\mathcal{N}(\mu, \sigma^2)$	normális eloszlás
$\sigma(z) = 1/(1+e^{-z})$	szigmoid (logisztikus) függvény
$\ell(\boldsymbol{\theta})$	log-likelihood
χ^2_n, $F(n_1, n_2)$	khi-négyzet és F-eloszlás

Modellek és tanulás

Jelölés	Jelentés
$\mathbf{w}$, b	súlyvektor, eltolás (lineáris modellek, neuronok)
$J(\boldsymbol{\theta})$	célfüggvény (veszteség és regularizáció)
$L(y, \hat{y})$, ℓ	veszteségfüggvény
λ, α	regularizációs paraméter (a `scikit-learn`-ben `alpha`, illetve `C = 1/`λ)
η	tanulási ráta
$\mathbf{W}^{(l)}$, $\mathbf{h}^{(l)}$, $\boldsymbol{\delta}^{(l)}$	a neurális hálózat l-edik rétegének súlymátrixa, kimenete, hibajele
φ	aktivációs függvény
$\mathbf{P}$, $\mathbf{t}$, λ_a	főkomponens-irányok mátrixa, főkomponens-értékek, sajátértékek
T^2, Q	Hotelling-féle statisztika, négyzetes előrejelzési hiba (PCA-alapú felügyelet)
u_{ik}, m	fuzzy tagsági érték és fuzzy kitevő (FCM)
$\sigma(X)$, $s(X \Rightarrow Y)$, $c(X \Rightarrow Y)$	támogatottság (darabszám és arány), bizonyosság (asszociációs szabályok)
H, G, IG	entrópia, Gini-index, információnyereség
$\tau(\mathbf{x})$	egyéni (feltételes) kezelési hatás, uplift

Idősorok

Jelölés	Jelentés
y_t, T	az idősor t-edik értéke, az idősor hossza
T_t, S_t, R_t	trend-, szezonális és maradékkomponens
m	szezonális periódus
B, ∇	késleltető operátor ($By_t = y_{t-1}$), differenciaoperátor
r_k	k késleltetésű autokorreláció
$\hat{y}_{T+h\|T}$	a T időpontban készített h lépéses előrejelzés
$D(i,j)$	a DTW kumulált költségmátrixa

Mértékegységek és számírás

A szövegben a magyar helyesírásnak megfelelően tizedesvesszőt és ezres tagolásra szóközt használunk (2 176; 0,95), a programkódban és a kimenetekben – a Python szabályai szerint – tizedespontot. A pénzösszegek forintban, a hőmérsékletek Celsius-fokban, az áramok m³/h-ban, a koncentrációk mol/L-ben értendők.

F. Magyar–angol szójegyzék

A szójegyzék a könyvben használt szakkifejezések angol megfelelőit adja meg, a nemzetközi szakirodalom és a szoftverdokumentációk olvasásának megkönnyítésére. A magyar terminológia a hazai szakirodalom bevett szóhasználatát követi; ahol több változat is elterjedt, zárójelben a másikat is feltüntetjük.

F.1. táblázat. Magyar–angol szakkifejezések.

Magyar	Angol
adatbányászat	data mining
adatelőkészítés	data preparation, preprocessing
adatszivárgás (információszivárgás)	data leakage
adattárház	data warehouse
adattó	data lake
aktivációs függvény	activation function
alapjel	setpoint
állapotindikátor	health indicator
anomália, kiugró érték	anomaly, outlier
asszociációs szabály	association rule
átlagos pontosság	average precision
autokorrelációs függvény	autocorrelation function (ACF)
autoenkóder	autoencoder
bizonyosság (megbízhatóság)	confidence
csoportosítás (klaszterezés)	clustering
dimenziócsökkentés	dimensionality reduction
dinamikus idővetemítés	dynamic time warping (DTW)
döntési fa	decision tree
egyosztályos SVM	one-class SVM
együttes modell	ensemble model
előrejelzés	forecasting, prediction
ellenőrzőkártya	control chart
elenyésző gradiens	vanishing gradient
esélyhányados	odds ratio
exponenciális simítás	exponential smoothing
fedezet	(profit) margin
feltáró adatelemzés	exploratory data analysis (EDA)
felügyelt / felügyelet nélküli tanulás	supervised / unsupervised learning
főkomponens-elemzés	principal component analysis (PCA)
fuzzy c-közép	fuzzy c-means (FCM)
gördülő kezdőpontú értékelés	rolling-origin evaluation
gradiens-boosting	gradient boosting
gyakori elemhalmaz	frequent itemset
hátralévő hasznos élettartam	remaining useful life (RUL)
hibadiagnózis	fault diagnosis
hibavisszaterjesztés	backpropagation
hiperparaméter	hyperparameter
hozzájárulási ábra	contribution plot
információnyereség	information gain

Magyar	Angol
izolációs erdő	isolation forest
jellemző	feature
jellemzőkinyerés, jellemzőtervezés	feature extraction, feature engineering
keresztvalidáció	cross-validation
kereszt-entrópia	cross-entropy (log loss)
kiegyensúlyozatlan osztályok	imbalanced classes
kiejtés (dropout)	dropout
konvolúciós hálózat	convolutional neural network (CNN)
korai leállítás	early stopping
költség–komplexitás metszés	cost-complexity pruning
lágy szenzor	soft sensor
látens szemantikus elemzés	latent semantic analysis (LSA)
lemorzsolódás	churn
lift (javulás)	lift
lokális kiugrási tényező	local outlier factor (LOF)
maradék	residual
margó	margin
maximum-likelihood becslés	maximum likelihood estimation
megtartás	retention
metszés	pruning
mintavételezés	sampling
modellértékelés	model evaluation
naiv Bayes-osztályozó	naive Bayes classifier
nemnegatív mátrixfaktorizáció	non-negative matrix factorization (NMF)
normalizálás, standardizálás	normalization, standardization
nyereségarány	gain ratio
osztályozás	classification
önszerveződő térkép	self-organizing map (SOM)
parciális függés	partial dependence
permutációs fontosság	permutation importance
pontosság (precízió), felidézés	precision, recall
prediktív karbantartás	predictive maintenance
regresszió	regression
regularizáció	regularization
rekurrens hálózat	recurrent neural network (RNN)
riasztásáradat	alarm flood
sajátérték, sajátvektor	eigenvalue, eigenvector
szegmentálás	segmentation
szennyezettség	impurity
szinguláris érték szerinti felbontás	singular value decomposition (SVD)
szóbeágyazás	word embedding
szótövezés, lemmatizálás	stemming, lemmatization
szózsák	bag of words
szürkedobozos modell	grey-box model
támaszvektor-gép	support vector machine (SVM)
támogatottság	support
témamodell	topic model
tévesztési mátrix	confusion matrix
torzítás–variancia felbontás	bias–variance decomposition
túltanulás	overfitting
ügyfélérték	customer lifetime value (CLV)
ügyfélkapcsolat-menedzsment	customer relationship management (CRM)
uplift-modell	uplift model
véletlen erdő	random forest
zsákolás	bagging (bootstrap aggregating)

Adatbányászat – a hatékonyság eszköze

Az adatbányászat az adatokban rejlő, korábban ismeretlen, hasznos és érthető összefüggések feltárása – ma a vállalatirányítás, a mérnöki gyakorlat és a tudományos kutatás mindennapi eszköze. A könyv a terület módszereit a matematikai háttér részletes kifejtésével és futtatható Python-kóddal mutatja be: az adattárházaktól és az adatelőkészítéstől a csoportosításon, az asszociációs szabályokon, a regresszión és az osztályozáson át a neurális hálózatokig, az anomáliák észleléséig, az idősorok és a szövegek elemzéséig.

A módszereket két, fejezetről fejezetre továbbvitt esettanulmány kapcsolja össze: egy webáruház üzleti adatai és egy vegyipari reaktor mérései. Az elméletet böngészőben futó gyakorló alkalmazások és Google Colab-jegyzetfüzetek egészítik ki, így minden eredmény kipróbálható és továbbfejleszthető.

A könyv elsősorban mérnök-, informatikus- és gazdálkodási BSc-hallgatóknak készült; a szükséges Python-, lineáris algebrai és statisztikai alapokat függelékek foglalják össze.

Pannon Egyetem
University of Pannonia

complexsystemsintelligence.github.io/adatbanyaszat

A szöveg és az ábrák CC BY-NC 4.0, a programkód MIT licenc alatt használható fel.