Podatki

Podatki v tabeli

Pričnemo s primerom tabelaričnih podatkov, ki predstavlja nabor podatkov o pacientih z nekaterimi osnovnimi zdravstvenimi značilnostmi:

ID pacienta	Starost	Spol	Krvni tlak (mmHg)	Holesterol (mg/dL)	Sladkorna bolezen (Da/Ne)
P001	45	M	130/85	210	Ne
P002	52	Ž	140/90	250	Da
P003	37	M	125/80	190	Ne
P004	60	Ž	160/95	280	Da

Razlaga:

- Vsaka vrstica (razen glave) predstavlja podatkovni primer (posameznega pacienta).
- Vsak stolpec predstavlja značilko (Starost, Spol, Krvni tlak itd.).
- Značilke so lahko numerične (Starost, Holesterol) ali kategorične (Spol, Ima sladkorno bolezen).
- Nekatere značilke (npr. Krvni tlak) lahko zahtevajo preoblikovanje, da so primernejše za strojno učenje.

Ta preprosta tabela pomaga pri uvodu v koncept strukturiranih podatkov, preden se lotimo preoblikovanj, kot so kontinuacija, diskretizacija in vgraditve (embeddings).

Tabelarične podatke lahko formalno predstavimo z matrično notacijo.

Notacija za podatke v tabeli (matriki)

Matematična predstavitev tabelaričnih podatkov

Podatke v tabelarni obliki lahko formalno predstavimo z matrično notacijo.

Matematični formalizem tabelaričnih podatkov

Naj bo:

- X podatkovna množica, predstavljena kot matrika dimenzij $n \times m$, kjer:
    - n je število podatkovnih primerov (vrstic),
    - m je število značilk (stolpcev).
- $X_{i,j}$ predstavlja vrednost j-te značilke za i-ti primer.
- $\mathbf{x}_i$ je vektorska predstavitev značilk (vrstica) za i-ti primer:

$$\mathbf{x}_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,m})$$

- $X_{\cdot,j}$ predstavlja stolpec j-te značilke (vse vrednosti te značilke za vse primere).

Če podatkovna množica vsebuje odvisno spremenljivko (ciljno vrednost), jo označimo kot:

- y je vektor dolžine n (ena vrednost na primer), ki predstavlja odvisno spremenljivko:

$$y = (y_1, y_2, \ldots, y_n)^T$$

  kjer y_i označuje ciljno vrednost za i-ti primer.

Matrična predstavitev

Za naš primer podatkovne množice:

$$X = \begin{bmatrix} 45 & \text{M} & 130/85 & 210 \\ 52 & \check{\text{Z}} & 140/90 & 250 \\ 37 & \text{M} & 125/80 & 190 \\ 60 & \check{\text{Z}} & 160/95 & 280 \end{bmatrix}$$

Če je ima sladkorno bolezen odvisna spremenljivka y, jo predstavimo kot:

$$y = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

kjer:

- $y_1 = 0$ (Brez sladkorne bolezni pri pacientu 1),
- $y_2 = 1$ (Sladkorna bolezen pri pacientu 2),
- in tako naprej.

Evklidska razdalja med dvema podatkovnima primeroma $\mathbf{x}_i$ in $\mathbf{x}_j$ v m-dimenzionalnem prostoru značilk je podana z:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{m} (X_{i,k} - X_{j,k})^2}$$

Razlaga:

- $X_{i,k}$ je vrednost k-te značilke za primer i.
- $X_{j,k}$ je vrednost k-te značilke za primer j.
- Formula izračuna kvadratni koren vsote kvadratov razlik vseh značilk.

Matrična notacija (vektorska oblika)

Alternativno, z vektorsko notacijo:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$$

kjer $\|\cdot\|_2$ označuje Evklidsko normo.

Tipi spremenljivk

V strojnem učenju lahko spremenljivke (ali značilke) razvrstimo na dva načina:

1. Glede na njihovo vlogo v podatkovni množici (vhodne vs. izhodne, razredne vs. meta spremenljivke)
2. Glede na tip podatkov (realne vrednosti, diskretne, kategorične itd.)

1. Glede na vlogo v podatkovni množici

1.1. Vhodne vs. izhodne spremenljivke

- Vhodne spremenljivke (Značilke, Kovariate, Napovedniki, Neodvisne spremenljivke)
    - Uporabljene kot vhodni podatki za model.
    - Označene kot X, kjer $X_{i,j}$ predstavlja vrednost značilke j za primer i.
- Izhodna spremenljivka (Ciljna spremenljivka, Odziv, Odvisna spremenljivka, Oznaka)
    - Spremenljivka, ki jo model napoveduje.
    - Označena kot y, kjer y_i predstavlja ciljno vrednost za primer i.
    - Pri klasifikaciji je y kategorična (npr. razredne oznake).
    - Pri regresiji je y numerična (realna vrednost).

1.2. Razredne vs. meta spremenljivke

- Razredna spremenljivka (Ciljna spremenljivka, Oznaka)
    - Uporablja se pri nadzorovanem učenju (klasifikacija ali regresija).
    - Primer: "Ima sladkorno bolezen" (Da/Ne) v medicinskem naboru podatkov.
- Meta spremenljivke (Metapodatki, Kontekstne spremenljivke)
    - Niso neposredno uporabljene za učenje modela, vendar nudijo kontekst.
    - Primer: "ID pacienta" ali "Časovni žig"—pomembni za sledenje, a ne za modeliranje.

2. Glede na tip podatkov

2.1. Numerične (kvantitativne) spremenljivke

- Realne (neprekinjene) spremenljivke

        - Lahko zavzamejo katerokoli realno vrednost v določenem območju.
        - Primer: Višina, Teža, Temperatura.
        - Običajno obdelane z normalizacijo ali standardizacijo.
    - Diskretne (celoštevilske) spremenljivke
        - Zavzamejo le celoštevilske vrednosti.
        - Primer: Število obiskov bolnišnice, Starost (če jo obravnavamo kot celo število).
        - Pogosto obravnavane podobno kot realne vrednosti.

2.2. Kategorične (nominalne in ordinalne) spremenljivke

- Nominalne (kategorične, neurejene) spremenljivke
    - Predstavljajo različne kategorije brez naravnega vrstnega reda.
    - Primer: "Moški"/"Ženska", "Rdeča"/"Modra"/"Zelena".
    - Običajno kodirane kot one-hot vektorji ali vgraditve (embeddings).
- Ordinalne spremenljivke
    - Kategorične spremenljivke z določenim vrstnim redom, vendar brez enakih presledkov med vrednostmi.
    - Primer: "Nizko", "Srednje", "Visoko" tveganje.
    - Pogosto kodirane kot cela števila ali preslikane v numerične vrednosti.

2.3. Besedilne in nizovne spremenljivke

- Prosto besedilo (nestrukturirani podatki)
    - Primer: Mnenja strank, zdravniški zapisi.
    - Zahteva NLP-tehnike, kot so vgraditve (npr. word2vec, BERT).
- Strukturirani nizi
    - Primer: Poštne številke, Identifikacijske kode izdelkov.
    - Lahko obravnavane kot kategorične spremenljivke ali pretvorjene v numerične predstavitve.

Kodiranje kategoričnih spremenljivk

Za pretvorbo diskretnih podatkov (npr. kategorij, besedila, logičnih vrednosti) v številčne

vrednosti, ki jih lahko uporabimo v strojnih učnih algoritmih, se uporabljajo naslednje metode:

1.1. One-hot kodiranje

- Vsaka kategorija se pretvori v binarni vektor, kjer ima ena sama komponenta vrednost 1, ostale pa 0.
- Uporablja se pri nominalnih (neurejenih) spremenljivkah.
- Primer:
    - Kategorija "rdeča", "modra", "zelena" → vektorji:
        - rdeča: (1, 0, 0)
        - modra: (0, 1, 0)
        - zelena: (0, 0, 1)
- Pomanjkljivosti: uvedba številnih novih soodvisnih spremenljivk (npr. kodiranje poštnih številk)

1.2. Ordinalno kodiranje

- Vsaki kategoriji se dodeli cela števila.
- Primerno za ordinalne spremenljivke (kjer vrstni red pomeni nekaj).
- Primer: "nizek", "srednji", "visok" → 1, 2, 3
- Pomanjkljivost: Model lahko napačno interpretira razlike kot linearne.

1.3. Domensko kodiranje

- Kombinacija zgornjih, le da se namesto številnih spremenljivk uvedejo kategorije in podkategorije
- Primer: kodiranje poštnih številk z nazivi pokrajin, občin, mest in potem številčno kodiranje teh

V nekaterih primerih je treba številčne (numerične) podatke pretvoriti v diskretne kategorije. To je koristno, kadar:

- Model ali algoritem bolje deluje s kategoričnimi podatki (npr. odločitvena drevesa, pravila, nekateri modeli razlage).
- Diskretizacija omogoča boljšo interpretacijo podatkov in razlage modelov.
- Podatki so naravno razdeljeni v skupine (npr. starostne skupine, temperature v območjih).

- Želimo zmanjšati občutljivost modela na manjše numerične spremembe (npr. pri zaokroževanju).

Diskretizacija numeričnih podatkov

- Lažja interpretacija: Namesto številk, ki so lahko abstraktne, uporabimo opisne kategorije.
- Manjša občutljivost na šum: Diskretizacija zmanjšuje vpliv manjših sprememb v podatkih.
- Podpora pravilnim modelom: Diskretni podatki omogočajo lažjo uporabo metod, kot so pravila odločanja, odločitvena drevesa, Bayesove mreže.

1. Enakomerno razdeljeni intervali (Equal-width binning)

- Razdelimo podatke na fiksne intervale enake širine.
- Primer: Starost (realna vrednost) → Starostne skupine (diskretna vrednost)
    - 0–18 let → "mlad"
    - 19–60 let → "odrasel"
    - 61+ let → "starejši"
- Matematika: Če imamo vrednosti x v intervalu $[x_{\min}, x_{\max}]$ in jih razdelimo na k enako širokih intervalov, potem širina posameznega intervala je:

$$w = \frac{x_{\max} - x_{\min}}{k}$$

in vsak primer x_i pripada razredu:

$$\text{razred}(x_i) = \lfloor \frac{x_i - x_{\min}}{w} \rfloor$$

2. Enako število primerov v vsakem intervalu (Equal-frequency binning)

- Razdelimo podatke tako, da je približno enako število vrednosti v vsaki skupini.
- Uporabno pri nelinearno porazdeljenih podatkih.
- Primer: Razvrščanje dohodkov v kvartile (npr. najnižjih 25 %, srednjih 50 %,

najvišjih 25 %).

3. Diskretizacija na podlagi metodami gručenja

- Gručenje poišče naravne skupine numeričnih podatkov in jih označi kot diskretne kategorije.
- Uporabno, ko imajo podatki nelinearne porazdelitve ali naravne gruče.

4. Uporaba domenskega znanja (Custom binning)

- Razredi so ročno določeni glede na strokovno znanje ali predhodno analizo.
- Primer: Temperaturna lestvica
    - < 0°C → "mrzlo"
    - 0–20°C → "zmerno"
    - 20+ °C → "toplo"

5. Uporaba ciljne spremenljivke

- Diskretizacija z uporabo entropije
- Klasifikacijska drevesa lahko naučimo poiskati optimalne točke razreza numeričnih podatkov glede na odvisno spremenljivko.

Natabelarični podatki

V strojnem učenju poleg tabelaričnih (strukturiranih) podatkov pogosto obdelujemo tudi neenotne, nestrukturirane ali polstrukturirane podatke. Tukaj so glavne vrste podatkov in njihov pomen v strojnem učenju:

1. Besedilni podatki (Text Data)

- Primeri:
    - Novice, članki, e-poštna sporočila, transkripti, čustveni komentarji, spletni pregledi, medicinska poročila.
- Izzivi:
    - Besedila nimajo fiksne strukture (dolžina se spreminja).

        - Interpretacija zahteva semantično razumevanje.
- Pretvorba v numerično obliko:
    - Bag-of-Words (BOW): Besedilo kot vektor pojavitev besed.
    - TF-IDF: Upodobi pomembnost besed v dokumentu.
    - Word Embeddings (Word2Vec, GloVe, BERT): Predstavitev besed v vektorskem prostoru.
    - Transformer modeli (GPT, BERT, T5): Napredno razumevanje besedila.
- Uporaba:
    - Strojno prevajanje, analiza sentimenta, klepetalni boti, iskalni algoritmi.

2. Slikovni podatki (Image Data)

- Primeri:
    - Fotografije, rentgenski posnetki, satelitske slike, mikroskopski posnetki.
- Izzivi:
    - Slike so običajno visokodimenzionalne matrike pikslov.
    - Pomembne značilnosti so lahko kompleksne (teksture, oblike, vzorci).
- Pretvorba v numerično obliko:
    - RGB vrednosti: Vsaka slika kot matrika števil (za vsako barvno komponento R, G, B).
    - Feature extraction (SIFT, HOG): Ročno določene značilke.
    - CNN (Convolutional Neural Networks): Samodejno učenje značilk.
- Uporaba:
    - Prepoznavanje obrazov, medicinska diagnostika, samovozeča vozila, analiza satelitskih posnetkov.

3. Omrežja

- Primeri:
    - Socialna omrežja (vozlišča = uporabniki, povezave = prijateljstva).
    - Molekularne strukture (vozlišča = atomi, povezave = vezi).
    - Povezljivost v transportnih mrežah.
- Izzivi:

    - Klasične metode (npr. linearna algebra) ne delujejo neposredno na grafih.
- Pretvorba v numerično obliko:
    - Adjacency matrix: Graf predstavimo kot matriko povezav.
    - Graph embeddings (Node2Vec, GraphSAGE, GCN): Zmanjšanje dimenzije in vektorska predstavitev vozlišč.
- Uporaba:
    - Priporočilni sistemi, analiza omrežij, bioinformatika (genske interakcije).

4. Nizi (Sequence Data)

- Primeri:
    - Časovne vrste (ekonomski podatki, vremenske meritve, EEG signali).
    - Biološki zapisi (DNA sekvence, proteinske sekvence).
- Izzivi:
    - Odvisnosti med preteklimi in prihodnjimi vrednostmi.
    - Variabilna dolžina podatkov.
- Pretvorba v numerično obliko:
    - Časovne značilke (lag features, Fourierjeva transformacija).
    - Recurrent Neural Networks (RNN, LSTM, GRU): Učijo odvisnosti v časovnih podatkih.
    - Transformers (GPT-3, BERT za časovne podatke): Učinkovito modeliranje sekvenc.
- Uporaba:
    - Napovedovanje zalog, analiza srčnega utripa, analiza vremenskih vzorcev.

5. Prostorski podatki (Spatial Data)

- Primeri:
    - Geolokacijski podatki, kartografija, lidar skeni.
- Izzivi:
    - Visoka dimenzionalnost, kompleksni odnosi.
- Pretvorba v numerično obliko:
    - Koordinatne pretvorbe (GPS → UTM projekcija).

    - Spatial embeddings (GeoVectors, kriging interpolacija).
- Uporaba:
    - Analiza mobilnosti, urbanistično načrtovanje, satelitsko kartiranje.

6. Zvok in signalni podatki (Audio & Signal Data)

- Primeri:
    - Glasovni posnetki, senzorni podatki, EEG signali.
- Izzivi:
    - Kompleksna frekvenčna analiza.
- Pretvorba v numerično obliko:
    - Fourierjeva transformacija, MFCC (Mel-Frequency Cepstral Coefficients).
    - WaveNet in CNN za zvok.
- Uporaba:
    - Prepoznavanje govora, diagnostika strojev, EEG analiza.

Vsaka vrsta podatkov zahteva prilagojene metode pretvorbe v numerično obliko, saj večina algoritmov strojnega učenja deluje s številskimi podatki. Razumevanje podatkovne strukture je ključno za izbiro pravilnega modela in učinkovito interpretacijo rezultatov.

Primer DNA niza in njegova pretvorba v numerično predstavitev s k-meri

1. Primer DNA niza

DNA zaporedje je niz nukleotidov (A, C, G, T). Predstavljajmo si naslednje zaporedje:

$$\text{DNA} = \text{"ATGCGATGACCTGACT"}$$

2. Razbitje niza na k-mere

K-mer je podniz dolžine k, ki drsi skozi zaporedje. Če izberemo $k = 3$ (trimeri), dobimo:

Pozicija	3-mer
1–3	ATG
2–4	TGC
3–5	GCG
4–6	CGA
5–7	GAT
6–8	ATG
7–9	TGA
8–10	GAC
9–11	ACC
10–12	CCT
11–13	CTG
12–14	TGA
13–15	GAC
14–16	ACT

3. Kodiranje k-merov v numerični obliko

Metode za pretvorbo k-merov v številke:

(a) One-hot kodiranje k-merov

- Za vsak možen 3-mer ustvarimo binarni vektor dolžine $4^3 = 64$.
- Vsak 3-mer ima 1 na svoji poziciji, ostalo so 0.
- Primer: ATG → (1, 0, 0, ..., 0)

(b) Pogostost k-merov (Bag-of-k-mers)

- Štejemo, kolikokrat se vsak možen 3-mer pojavi v nizu.
- Primer:

    - ATG: 2
    - TGC: 1
    - GCG: 1
    - CGA: 1
    - GAT: 1
    - itd.
- Predstavimo kot vektor:

$$\mathbf{x} = (2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...)$$

(c) Embedding model (DNA2Vec za k-mere)

- Namesto enostavne frekvence model naučimo vektorske predstavitve k-merov z Word2Vec, kjer podobni k-meri dobijo podobne vektorje.
- Primerno za analizo zaporedij s strojno učenimi modeli.

4. Uporaba numerične predstavitve

- Strojno učenje: Razlikovanje med zdravimi in bolnimi genskimi sekvencami.
- Klasifikacija: Napovedovanje funkcije DNA zaporedij.
- Genomska analiza: Iskanje podobnosti med vrstami.

Odprto problemi: kakšen naj bo k?

Diverzija: učenje embedding modelov za (DNA) nize

Continuous Bag-of-Words (CBOW) in Skip-gram pristop

CBOW in Skip-gram sta dve glavni metodi učenja vektorskih predstavitev (embeddingov) besed ali k-merov, uporabljeni v Word2Vec, DNA2Vec in podobnih modelih. Njuna glavna razlika je v tem, kako uporabljata kontekstne informacije.

1. Continuous Bag-of-Words (CBOW)

Napoveduje ciljno besedo (ali k-mer) iz okoliških besed.
Primer (za besedilo):

- V stavku "*Maček spi na okenski polici*" model dobi kontekst (npr. "Maček", "na", "okenski", "polici") in poskuša napovedati manjkajočo besedo "spi".

Matematično:
Če imamo zaporedje besed $w_1, w_2, ..., w_T$, potem CBOW maksimizira verjetnost ciljne besede w_t, danih sosednjih besed (konteksta):

$$P(w_t|w_{t-n}, ..., w_{t-1}, w_{t+1}, ..., w_{t+n})$$

Primer za DNA:
Če imamo DNA niz:

$$\text{”ATG CGA TGA ACC”}$$

in če uporabimo CBOW z oknom 2, bo model, na primer, iz konteksta ("ATG", "TGA") napovedoval srednji k-mer "CGA".

Prednosti CBOW:

- Hitrejši od Skip-gram.
- Dobro deluje pri velikih količinah podatkov.

Slabosti CBOW:

- Slabše pri redkih k-merih (manj natančne reprezentacije).

---

2. Skip-gram pristop

Napoveduje okoliške besede iz ciljne besede.
Primer (za besedilo):

- V istem stavku "*Maček spi na okenski polici*" model dobi ciljno besedo "spi" in poskuša napovedati, katere besede so v njenem kontekstu (npr. "Maček", "na", "okenski", "polici").

Matematično:
Verjetnost kontekstnih besed $w_{t-n}, ..., w_{t-1}, w_{t+1}, ..., w_{t+n}$, danih ciljni besedi w_t:

$$P(w_{t-n}, ..., w_{t+n}|w_t)$$

Primer za DNA:
Če imamo niz:

$$\text{”ATG CGA TGA ACC”}$$

in uporabimo Skip-gram z oknom 2, bo model iz ciljne besede "CGA" napovedoval verjetne okoliške k-mere ("ATG", "TGA").

Prednosti Skip-gram:

- Boljše deluje pri manjših podatkovnih zbirkah.
- Učinkovitejši pri redkih k-merih.

Slabosti Skip-gram:

- Počasnejši, ker modelira več napovedi.

Ključna razlika med CBOW in Skip-gram

Metoda	Napoveduje?	Hitrost	Primeren za?
CBOW	Ciljno besedo iz konteksta	Hitrejši	Velike podatkovne zbirke
Skip-gram	Kontekst iz ciljne besede	Počasnejši	Redke besede (ali k-mere)

Zaključek

- CBOW je hiter in primeren za velike genske zbirke.
- Skip-gram je boljši, če želimo boljše vektorske predstavitve za redke k-mere ali mutacije.

Ali želiš praktičen Python primer teh metod za DNA? 🚀

Kemijske strukture

SMILES (*Simplified Molecular Input Line Entry System*) je linearni zapis kemijskih struktur s pomočjo enostavne besedilne notacije. Vsak molekularni del je predstavljen s simboli elementov, vezmi in posebnimi oznakami za funkcionalne skupine.

Primer za aspirin (*acetilsalicilna kislina*):
SMILES zapis:

```
CC(=O)Oc1ccccc1C(=O)O
```

Razlaga zapisa:

- `CC(=O)O` → acetoksilna skupina (*-$OCOCH_3$*)
- `c1ccccc1` → benzenski obroč
- `C(=O)O` → karboksilna skupina (*-COOH*)

SMILES je pogosto uporaben v kemijski informatiki, saj omogoča računalniško shranjevanje in obdelavo molekulskih podatkov.

Prevod SMILES zapisa v numerični zapis se običajno izvaja z različnimi tehnikami v kemijski informatiki in strojnem učenju. Ključni pristopi vključujejo:

1. One-hot enkodiranje

- Vsak možen znak v SMILES zapisu (C, O, =, -, (,), …) se predstavi kot binarni vektor.
- Primer: Če imamo 20 različnih simbolov, se vsaka molekula predstavi z matriko velikosti dolžina SMILES × 20.

2. Molekularni prstni odtisi (Fingerprints)

- Pretvarjanje v bitne vektorje s fiksno dolžino, kjer vsak bit označuje prisotnost

določene podstrukture ali kemijske lastnosti.

- Pogosti algoritmi:
    - Morgan fingerprint (ECFP): Razširja molekulo iz atomskih jeder in kodira s krožnim pristopom.
    - MACCS ključne lastnosti: Vnaprej definirane podstrukture in kemijski vzorci (166-bitni vektor).

3. Graph Neural Networks (GNNs)

- SMILES se najprej pretvori v graf molekule (vozlišča = atomi, povezave = vezi).
- GNN nato obdela graf in iz njega izračuna numerične vektorje.

4. Sekvenčno enkodiranje (RNN/LSTM/Transformer)

- SMILES obravnavamo kot zaporedje znakov in uporabimo Word2Vec, LSTM ali Transformerje za vektorizacijo.

Primer pretvorbe za aspirin (`CC(=O)Oc1ccccc1C(=O)O`):

1. One-hot matrika (vsaka črka pretvorjena v vektor)
2. ECFP fingerprint (npr. 1024-bitni vektor)
3. GNN reprezentacija (vektorski zapis grafa molekule)

Vsak pristop ima svoje prednosti, odvisno od uporabe – one-hot je preprost, fingerprints so kompaktni, GNN pa omogoča boljšo generalizacijo na nevidene molekule.

Morgan Fingerprint (ECFP) je krožni prstni odtis molekule, kjer se podstrukture kodirajo v bitni vektor s fiksno dolžino (npr. 1024 bitov). Vsak bit označuje prisotnost določene kemične podstrukture. Spodaj je nekaj primerov značilnih fragmentov, ki bi se lahko pojavili v aspirinu in drugih molekulah:

Primeri fragmentov v Morgan Fingerprint za aspirin (`CC(=O)Oc1ccccc1C(=O)O`)

1. Aromatski obroč (benzen)
    - Oznaka: `c1ccccc1`
    - Prisoten v molekuli aspirina.

2. Karboksilna skupina (-COOH)
    - Oznaka: `C(=O)O`
    - Kritična za kisle lastnosti aspirina.
3. Esterna skupina (-COO-)
    - Oznaka: `CC(=O)O`
    - Nastane zaradi acetilacije salicilne kisline.
4. Etilna skupina (-CH3)
    - Oznaka: `CC`
    - Prisotna v acetilni skupini (`-OCOCH3`).
5. Keton (-C=O)
    - Oznaka: `C(=O)`
    - Del esterne in karboksilne skupine.

Kako izgleda Morgan Fingerprint?

- Morgan Fingerprint za aspirin bi bil 1024-bitni vektor, kjer so določeni biti *1*, če je prisoten določen fragment, in *0*, če ga ni.
- Na primer:

```
00001000100001000010001000000001000000000... (1024 bitov)
```

    - Vsak 1 označuje prisotnost specifične kemične podstrukture.

Zakaj uporabiti Morgan Fingerprint?

- Omogoča iskanje podobnosti med molekulami.
- Boljši kot *one-hot encoding*, saj upošteva kemijsko bližino atomov.
- Široko uporabljen v strojnih modelih za napovedovanje lastnosti spojin.

Če želiš generirati Morgan Fingerprint za določeno molekulo (npr. aspirin) v Pythonu, lahko uporabiš RDKit:

```python
from rdkit import Chem
from rdkit.Chem import AllChem

smiles = "CC(=O)Oc1ccccc1C(=O)O"  # Aspirin
mol = Chem.MolFromSmiles(smiles)
fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius=2, nBits=1024)

print(fp.ToBitString())  # Prikaz 1024-bitnega vektorja
```

To bo ustvarilo binaren odtis, ki ga lahko uporabiš za kemijsko analizo!

Slike

Numerična predstavitev slik temelji na matrični in vektorski obliki, kjer se barvne ali sivinske vrednosti pikslov pretvorijo v številčne vrednosti.

1. Matrična predstavitev (surovi podatki)

- Sivinska slika ($H \times W$): Matrika vrednosti I_{ij}, kjer je vsak piksel število v območju [0, 255] (če je 8-bitna slika).

$$I = \begin{bmatrix} 12 & 45 & 78 & 200 \\ 34 & 89 & 120 & 255 \\ 100 & 140 & 190 & 220 \end{bmatrix}$$

- Barvna slika ($H \times W \times 3$): Tridimenzionalna matrika, kjer vsaka plast (R, G, B) vsebuje vrednosti od 0 do 255.

2. Vektorska predstavitev

- Slike lahko pretvorimo v dolge vektorje (flattening).
    - Če imamo sliko 28×28 pikslov, jo lahko predstavimo kot vektor dolžine 784.

3. Predstavitev z značilkami

Namesto surovih podatkov lahko slike predstavimo s pomembnimi značilkami:

- Histogram orientiranih gradientov (HOG) – uporablja robne značilnosti.
- SIFT (Scale-Invariant Feature Transform) – zajema ključne točke.
- CNN (Convolutional Neural Networks) – nauči globoke reprezentacije, kjer vsaka plast zajema abstraktne lastnosti slike.

4. Vlozitve v vektorske prostore

- Autoencoderji: Stisnejo sliko v manjdimenzionalni vektor, ki ohranja ključne informacije.
- ResNet/VGG: Uporabijo se predtrenirani modeli, ki slike pretvorijo v visokodimenzionalne vektorje (npr. vektor dolžine 512).

Besedila

Numerična predstavitev tekstovnih podatkov

Besedilni podatki so nestrukturirani, zato jih moramo pretvoriti v numerično obliko, da jih lahko obdelujemo s strojnim učenjem. Obstaja več metod:

1. One-hot kodiranje

- Vsaki besedi dodelimo binarni vektor, kjer ima ena komponenta vrednost 1, ostale pa 0.
- Primer za besede v besedilu "*Maček spi na polici*":

Beseda	Maček	spi	na	polici
Maček	1	0	0	0
spi	0	1	0	0
na	0	0	1	0

Beseda	Maček	spi	na	polici
polici	0	0	0	1

⚠ Težava: Velikost vektorja hitro narašča z velikostjo besednega zaklada.

2. Bag-of-Words (BOW)

- Pretvori besedilo v vektor frekvenc posameznih besed.
- Primer:
    - Besedilo: "*Maček spi. Maček prede.*"
    - Besednjak: ["Maček", "spi", "prede"]
    - Vektor: (2, 1, 1) (Maček = 2x, spi = 1x, prede = 1x)

⚠ Težava: Ne upošteva vrstnega reda besed.

3. TF-IDF (Term Frequency - Inverse Document Frequency)

- Dodeli večjo težo pomembnim besedam in zmanjša vpliv pogostih, manj informativnih besed.
- Izračuna se kot:

$$\text{TF-IDF} = \text{TF} \cdot \text{IDF}$$

kjer je:

- TF (Term Frequency) = $\frac{\text{št. pojavitev besede}}{\text{št. vseh besed v dokumentu}}$
- IDF (Inverse Document Frequency) = $\log \frac{\text{št. vseh dokumentov}}{\text{št. dokumentov, ki vsebujejo besedo}}$

Prednost: Boljša od BOW, ker izloči pogoste besede (kot "in", "je", "na").

4. Word Embeddings (Word2Vec, FastText, GloVe)

- Besede pretvori v goste vektorske predstavitve, kjer so podobne besede bližje v vektorskem prostoru.
- Modeli:
    - Word2Vec (CBOW, Skip-gram)
    - FastText (dela s podbesedami)

    - GloVe (temelji na so-pojavnosti besed)

Prednost: Ohranja semantične odnose, npr.

- "kralj" - "moški" + "ženska" ≈ "kraljica"

5. Kontekstualni vektorski modeli (BERT, GPT, T5)

- Uporabljajo transformer arhitekturo, kjer besede dobijo kontekstno odvisne vektorje.
- Primer: "bank" (banka) ima drugačen embedding v stavku "*Odšel sem v banko.*" kot v "*Ribič sedi na bregu banke.*"

Prednost: Razume pomen besed v različnih kontekstih.

Zaključek

- Preproste metode: One-hot, BOW, TF-IDF.
- Napredne metode: Word2Vec, FastText, GloVe.
- Sodobni standard: BERT, GPT za kontekstualno razumevanje.

Ko so podatkovni primeri predstavljeni kot vektorji, lahko za ocenjevanje razdalje med njimi uporabimo različne meritve razdalje in podobnosti. Izbira ustrezne metode je odvisna od vrste podatkov in aplikacije.

Merjenje razdalj

1. Evklidska razdalja (Euclidean Distance)

- Formula:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{m} (X_{i,k} - X_{j,k})^2}$$

- Opis: Mera "fizične" razdalje med dvema točkama v prostoru.
- Uporaba: Klasična metrika za kontinuirane (realne) podatke.
- Slabost: Občutljiva na različne lestvice vrednosti (potrebna je normalizacija).

2. Manhattan razdalja (Taxicab Distance, L1 norm)

- Formula:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{m} |X_{i,k} - X_{j,k}|$$

- Opis: Sešteje absolutne razlike med koordinatami.
- Uporaba: Primerna za redke vektorje (npr. v besedilni analizi).
- Slabost: Manj občutljiva na večje razlike v posameznih dimenzijah kot Evklidska razdalja.

3. Kosinusna podobnost (Cosine Similarity)

- Formula:

$$S(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}$$

  kjer je $\mathbf{x}_i \cdot \mathbf{x}_j$ skalarni produkt, $\|\mathbf{x}_i\|$ in $\|\mathbf{x}_j\|$ pa dolžini vektorjev.
- Opis: Meri kot med dvema vektorjema, ne pa absolutne razdalje.
- Uporaba: Zelo učinkovita pri besedilni analizi (TF-IDF, Word2Vec), kjer nas zanima podobnost ne glede na dolžino vektorjev.
- Slabost: Ne upošteva velikosti razlik v vektorjih, temveč samo njihov kot.

4. Jaccardova razdalja (Jaccard Distance)

- Formula (za množice):

$$d_J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

- Opis: Določa podobnost med kategoričnimi podatki ali binarnimi nizi.
- Uporaba: Pogosto pri analizi dokumentov, kjer primerjamo nize besed (npr. k-mer

reprezentacije DNA sekvenc).

- Slabost: Ne deluje dobro pri podatkih z veliko skupnimi elementi (majhne razlike postanejo neizrazite).

5. Mahalanobisova razdalja

Mahalanobisova razdalja je metrika, ki meri razdaljo med dvema podatkovnima točkama ob upoštevanju korelacij med spremenljivkami. Za razliko od Evklidske razdalje, ki predpostavlja, da so vse dimenzije neodvisne in enako pomembne, Mahalanobisova razdalja normalizira podatke glede na njihovo kovariančno matriko.

Za dve podatkovni točki $\mathbf{x}_i$ in $\mathbf{x}_j$ v večdimenzijskem prostoru je Mahalanobisova razdalja definirana kot:

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T S^{-1} (\mathbf{x}_i - \mathbf{x}_j)}$$

kjer:

- $\mathbf{x}_i, \mathbf{x}_j$ sta vektorja podatkovnih točk,
- S je kovariančna matrika podatkov (dimenzije $m \times m$),
- S^{-1} je inverzna kovariančna matrika,
- $(\mathbf{x}_i - \mathbf{x}_j)$ je vektorska razlika med podatkovnima točkama.

Če je kovariančna matrika enotska (identitetna), Mahalanobisova razdalja postane enaka Evklidski razdalji.

Prednosti Mahalanobisove razdalje

1. Neobčutljivost na različne lestvice podatkov
    - Če imamo spremenljivke z različnimi enotami (npr. višina v cm, teža v kg), Mahalanobisova razdalja odstrani vpliv enot, saj upošteva varianco in kovarianco podatkov.
2. Občutljivost na korelacije med spremenljivkami
    - Če sta dve spremenljivki močno korelirani, Evklidska razdalja lahko preceni njihovo pomembnost. Mahalanobisova razdalja to popravi s tem, da upošteva kovariančno strukturo podatkov.
3. Uporabnost pri odkrivanju anomalij

- Točke, ki so daleč od povprečja glede na kovariančno matriko, imajo visoko Mahalanobisovo razdaljo. To je koristno pri odkrivanju izjem (outlier detection).

Geometrijska interpretacija

- Pri Evklidski razdalji so vse dimenzije obravnavane kot neodvisne, zato so izolinije razdalj koncentrične krogle.
- Pri Mahalanobisovi razdalji pa upoštevamo kovariančno matriko podatkov, kar vodi do eliptičnih izolinij razdalj, ki so usmerjene glede na porazdelitev podatkov.

Zaključek

- Evklidska in Manhattan razdalja sta osnovni, vendar sta občutljivi na lestvico podatkov.
- Kosinusna podobnost je uporabna za besedila in embeddinge.
- Mahalanobisova razdalja upošteva porazdelitev podatkov, a je računanje drago.
- Jaccardova in Hammingova razdalja sta uporabni za kategorične in binarne podatke.

Prekletstvo visokih dimenzij

1. Težava Evklidske razdalje v visokih dimenzijah (prekletstvo dimenzionalnosti)

Evklidska razdalja je uporabna v nizkodimenzijskih prostorih, kjer so podatkovne točke dobro ločene. V visokih dimenzijah pa ima več resnih težav:

1. Razdalje med podatkovnimi točkami postanejo skoraj enake
    - V večdimenzijskem prostoru podatki pogosto lebdijo v "praznini" in so vse točke približno enako oddaljene.
    - Razlika med najbližjo in najbolj oddaljeno točko se zmanjša, kar oteži ločevanje podatkov.
2. Dimenzije prispevajo šum, ki zamegli razdaljo
    - Vsaka nova dimenzija doda neodvisno variabilnost, kar povzroči, da Evklidska razdalja narašča nepredvidljivo.

3. Potreba po normalizaciji podatkov
    - Če imajo značilke zelo različne lestvice vrednosti, ena dimenzija dominira nad ostalimi, kar izkrivi izračun Evklidske razdalje.

2. Kosinusna razdalja kot alternativa

Ko Evklidska razdalja postane neuporabna, pogosto uporabljamo Kosinusno razdaljo, ki meri kot med vektorji in ne absolutne razdalje med njimi.

1. Ne občuti vpliva visokih dimenzij tako močno kot Evklidska razdalja
    - Kosinusna podobnost gleda samo kot med vektorji, ne pa absolutne razlike med njimi.
    - V visokih dimenzijah so vektorji pogosto redki, vendar so njihovi koti še vedno informativni.
2. Ni občutljiva na velikost vektorjev
    - Evklidska razdalja močno vpliva, če so vektorji različnih velikosti, medtem ko Kosinusna razdalja deluje neodvisno od tega.
3. Primerna za besedilne podatke in embeddinge
    - Pogosto se uporablja pri TF-IDF, Word2Vec, BERT embeddingih, kjer besede ali dokumenti živijo v visokodimenzijskih prostorih.

Primer uporabe: iskanje osamelcev

Ko imamo podatke predstavljene kot vektorje in lahko računamo razdalje med primeri, lahko osamelce (outlierje) identificiramo z različnimi metodami. Glavni pristopi temeljijo na razdalji, gostoti ali modelskih pristopih.

---

1. Identifikacija osamelcev na podlagi razdalje

1.1. Najbližji sosedje (k-Nearest Neighbors, k-NN)

- Osamelci so točke, katerih povprečna razdalja do njihovih k najbližjih sosedov je velika.

- Postopek:
    i. Za vsak primer $\mathbf{x}_i$ izračunamo razdalje do njegovih k najbližjih sosedov.
    ii. Povprečimo te razdalje.
    iii. Točke z najvišjimi povprečnimi razdaljami so osamelci.
- Metrike razdalje:
    - Evklidska razdalja (za nizke dimenzije).
    - Mahalanobisova razdalja (če podatki niso neodvisni).
    - Kosinusna razdalja (za visoke dimenzije).

1.2. Lokalni faktor osamelcev (Local Outlier Factor, LOF)

- Osamelci imajo nizko gostoto glede na svoje sosede.
- Formula:

$$LOF(\mathbf{x}_i) = \frac{\sum_{j \in kNN(\mathbf{x}_i)} \frac{lrd(\mathbf{x}_j)}{lrd(\mathbf{x}_i)}}{|kNN(\mathbf{x}_i)|}$$

  kjer je lrd (lokalna gostota) definirana kot inverzna povprečna razdalja do k sosedov.
- Če je $LOF(\mathbf{x}_i) > 1$, je točka verjetno osamelec.

Lokalna gostota (Local Reachability Density, lrd) v algoritmu Local Outlier Factor (LOF) je definirana kot inverzna povprečna dosegljiva razdalja (reachability distance) do k najbližjih sosedov.

Za podatkovno točko $\mathbf{x}_i$ je lokalna gostota definirana kot:

$$lrd_k(\mathbf{x}_i) = \frac{|kNN(\mathbf{x}_i)|}{\sum_{\mathbf{x}_j \in kNN(\mathbf{x}_i)} \text{reach-dist}_k(\mathbf{x}_i, \mathbf{x}_j)}$$

kjer je:

- $kNN(\mathbf{x}_i)$ množica k najbližjih sosedov točke $\mathbf{x}_i$,
- $|kNN(\mathbf{x}_i)|$ število teh sosedov (tj. k),

- $\text{reach-dist}_k(\mathbf{x}_i, \mathbf{x}_j)$ dosegljiva razdalja med $\mathbf{x}_i$ in njenim sosedom $\mathbf{x}_j$, ki je definirana kot:

$$\text{reach-dist}_k(\mathbf{x}_i, \mathbf{x}_j) = \max\{\text{k-distance}(\mathbf{x}_j), d(\mathbf{x}_i, \mathbf{x}_j)\}$$

kjer je:

- $\text{k-distance}(\mathbf{x}_j)$ razdalja do k-tega najbližjega soseda točke $\mathbf{x}_j$,
- $d(\mathbf{x}_i, \mathbf{x}_j)$ dejanska razdalja med $\mathbf{x}_i$ in $\mathbf{x}_j$.

Interpretacija

- Nižja vrednost lrd pomeni, da je točka v redko poseljeni regiji, kar pomeni, da bi lahko bila osamelec.
- Višja vrednost lrd pomeni, da je točka v gosto poseljeni regiji, kar pomeni, da je verjetno normalna točka.

To vrednost nato uporabimo pri izračunu Local Outlier Factor (LOF), kjer primerjamo razmerje gostote točke in njenih sosedov.

Ali želiš nadaljevanje s formulo za LOF?

- Prednosti:
    - Prilagodi se lokalni gostoti, zato je robustnejši od preproste razdalje.
    - Dobro deluje v heterogenih podatkih, kjer imajo različni deli podatkovne množice različno gostoto.

Primeri enostavne uporabe numeričnih zapisov podatkov in iskanje podobnosti

- drzave
- slike
- besede
- besedila

Iskanje osamelcev

Če imamo dane skupine, izračun silhuete, na primer za:

- zivali

Klasifikacija dodatnih domenskih podatkov in njihova uporabnost pri razlagi modelov

Dodatni domenski podatki lahko vključujejo ontologije, besedne opise, slikovne opise in druge oblike kontekstualnega znanja, ki izboljšajo razumljivost in interpretacijo modelov strojnega učenja. Ti podatki lahko služijo kot razlaga rezultatov, dodatne značilke v modelih ali omejitve za boljše napovedovanje.

1. Strukturni domenski podatki (Ontologije in taksonomije)

- Primeri:
    - Biomedicinske ontologije (Gene Ontology, SNOMED CT)
    - Taksonomije za razvrščanje vrst (npr. Linnejeva klasifikacija)
    - Ontologije znanja (npr. WordNet, ConceptNet)
- Uporabnost pri razlagi:
    - Hierarhično razumevanje značilk: Ontologije omogočajo semantično interpretacijo rezultatov (npr. gen A vpliva na proces B).
    - Združevanje podobnih entitet: Omogoča združevanje podatkov glede na konceptualno sorodnost.
    - Pravila in omejitve: Ontološka pravila lahko pomagajo filtrirati napačne napovedi ali jih pojasniti.

2. Besedni opisi značilk in primerov

- Primeri:
    - Medicinske diagnoze s tekstovnimi opisi simptomov.
    - Razlaga spremenljivk v podatkovnem naboru (metapodatki).
    - Strnjene opombe ali povzetki modela v naravnem jeziku.

- Uporabnost pri razlagi:
    - Samorazlagajoči modeli: LLM (npr. GPT) lahko uporabijo besedne opise za generiranje tekstovne razlage modela.
    - Večja transparentnost modela: Če model uporablja človeško berljive značilke, jih je lažje interpretirati.
    - Dopolnitev surovih številčnih podatkov: Besedni opisi lahko pomagajo premostiti vrzel med numeričnimi podatki in človekovim razumevanjem.

3. Slikovni in vizualni opisi primerov

- Primeri:
    - Mikroskopske slike celic pri medicinski diagnostiki.
    - Satelitski posnetki za prepoznavo geografskih vzorcev.
    - Vizualizacije podatkovnih vzorcev za izboljšano interpretacijo modelov.
- Uporabnost pri razlagi:
    - Povezava med vizualnimi vzorci in modelom: Modeli lahko razložijo, katera področja slike so bila ključna za odločitev.
    - Uporaba tehnike Grad-CAM v CNN: Vizualizacija, ki prikazuje, kateri deli slike so vplivali na napoved.
    - Uporaba kombinacije slik in numeričnih podatkov: Multimodalni modeli lahko združijo vizualne in numerične značilke za bogatejšo razlago.

4. Pravilne omejitve in logična pravila

- Primeri:
    - Pravila v ekspertskih sistemih (če ima pacient visok krvni tlak, potem je tveganje za srčni infarkt večje).
    - Logične izjave v Prologu ali OWL (Web Ontology Language).
    - Omejitve, ki zagotavljajo, da model ne krši osnovnih fizikalnih zakonov.
- Uporabnost pri razlagi:
    - Zagotavljanje razložljivosti modela: Model je lahko dopolnjen z logičnimi pravili, ki pomagajo razložiti odločitve.

- Zagotavljanje skladnosti s fizikalnimi ali medicinskimi zakonitostmi: Model se izogiba nesmiselnih napovedi.
- Boljša interpretacija rezultatov: Razlaga v obliki pravila (IF-THEN) je pogosto bolj razumljiva kot številčna verjetnostna ocena.

Dodatni domenski podatki bistveno izboljšajo interpretacijo modelov, še posebej v kritičnih področjih, kot so medicina, biološke raziskave in razumevanje kompleksnih sistemov.