ANALIZA VELIKIH PODATAKA

školska 2024/2025 godina

Vežba 4: Čišćenje i priprema podataka za analizu

Kada radimo sa realnim podacima, oni često nisu savršeno strukturirani niti spremni za analizu. Podaci mogu doći iz različitih izvora (npr. senzori, ankete, baze podataka) i gotovo uvek sadrže određene nepravilnosti. Te nepravilnosti uključuju:

- **Nedostajuće vrednosti** – prazne ćelije koje onemogućavaju precizne analize
- **Duplikate** – isti podaci ponovljeni više puta, što narušava rezultate
- **Pogrešne tipove podataka** – brojevi kao tekst, datumi kao običan string
- **Tekstualne podatke koje je potrebno pretvoriti u brojeve**
- **Vremenske vrednosti koje nisu u ispravnom formatu**

⚠️ Loše pripremljeni podaci mogu direktno uticati na tačnost modela i zaključaka. Zato je kvalitetna priprema podataka **neizostavan korak** u svakom procesu obrade podataka ili izgradnje modela. U ovoj vežbi ćemo pokazati neke od najpoznatijih i najefikasnih funkcija iz `numpy i pandas` biblioteka koje možete primeniti nad svojim podacima iz DataFrame-a.

1. Obrada nedostajućih vrednosti

Nedostajuće vrednosti (npr. NaN) su vrlo česte i moraju se pažljivo tretirati. Ako ih ignorišemo, mogu uzrokovati greške ili dati iskrivljene rezultate. Postoji više strategija za njihovo rešavanje:

- **Uklanjanje redova ili kolona** – kada je procenat NaN nizak i ne utiče značajno na skup podataka.
- **Imputacija (popunjavanje)**:
    - Srednjom vrednošću (mean) – korisno kada je raspodela normalna.
    - Medijanom (median) – koristi se kada postoje ekstremi (outlieri).
    - Najčešćom vrednošću (mode) – pogodna za kategorijske podatke.

Primer koda:

U prikazanom primeru, prvo identifikujemo kolone koje sadrže nedostajuće vrednosti pomoću `isnull().sum()`. Nakon toga, kolona *Godine* se popunjava (imputuje) pomoću dve različite metode: prosečnom vrednošću (`mean`) i medijanom (`median`). Ove nove kolone mogu nam pomoći da uporedimo efekte različitih metoda popunjavanja.

```
data = {
    'Ime': ['Ana', 'Marko', 'Jelena', 'Petar'],
    'Godine': [25, np.nan, 30, np.nan],
    'Grad': ['Beograd', 'Novi Sad', None, 'Niš']
}

df = pd.DataFrame(data)

# Prikaz nedostajućih vrednosti
print("Broj NaN vrednosti po koloni:")
print(df.isnull().sum())

# Imputacija srednjom i medijanom
df['Godine_mean'] = df['Godine'].fillna(df['Godine'].mean())
df['Godine_median'] = df['Godine'].fillna(df['Godine'].median())
```

2. Uklanjanje duplikata

Duplikati su identični redovi koji se ponavljaju. Ako ih ne uklonimo, mogu negativno uticati na statistiku i dovesti do pogrešnih interpretacija. Najčešći uzroci su greške pri unosu podataka ili njihovo višestruko preuzimanje.

Primer koda:

U narednom kodu najpre kreiramo DataFrame sa namerno ponovljenim redom. Zatim pomoću `duplicated()` proveravamo koji redovi su označeni kao duplikati (True/False), a sa `drop_duplicates()` uklanjamo ih, ostavljajući samo jedinstvene zapise. Ova metoda čuva prvi red gde se pojavljuju podaci i uklanja sve ostale kopije.

```
df_dups = pd.DataFrame({
    'Ime': ['Ana', 'Marko', 'Ana'],
    'Godine': [25, 22, 25],
    'Grad': ['Beograd', 'Niš', 'Beograd']
})

# Identifikacija i uklanjanje duplikata
print(df_dups.duplicated())
df_unique = df_dups.drop_duplicates()
```

3. Konverzija tipova podataka

Automatski učitani podaci mogu biti u pogrešnom formatu. Na primer, brojevi se često učitaju kao stringovi (tekst), što može izazvati probleme ako pokušamo da radimo aritmetičke operacije. Isto važi za datume – ako ostanu kao tekst, ne možemo ih filtrirati, sortirati po vremenu ili izdvajati delove datuma.

Zato je važno proveriti i po potrebi konvertovati tipove podataka u one koji odgovaraju analizi: numerički za brojeve, datetime za datume i sl.

Primer koda:

U ovom primeru kolona *Godine* se originalno sastoji od tekstualnih vrednosti, ali se pomoću `pd.to_numeric()` konvertuje u brojčane vrednosti. Slično tome, kolona *Datum* se konvertuje u pravi `datetime` tip pomoću `pd.to_datetime()`, što omogućava dalje vremenske analize.

```
data = {
    'Godine': ['25', '30', '28'],
    'Datum': ['2024-04-21', '2023-05-15', '2022-10-10']
}

df = pd.DataFrame(data)

# Konverzija u odgovarajuće tipove
df['Godine'] = pd.to_numeric(df['Godine'])
df['Datum'] = pd.to_datetime(df['Datum'])
```

4. Rad sa vremenskim podacima

Vremenski podaci pružaju dosta informacija, ali samo ako su u odgovarajućem formatu. Kada su pravilno konvertovani u datetime, možemo iz njih automatski izvući godinu, mesec, dan, pa čak i dan u nedelji, kvartal i druge komponente koje pomažu u vremenskoj analizi.

Ova obrada podataka je korisna kada želimo da analiziramo sezonske trendove, poređenja po mesecima ili praćenje promena kroz vreme.

Primer koda:

U datom kodu nakon što je kolona *Datum* konvertovana u `datetime`, pomoću `.dt` pristupamo njenim komponentama – godinu izdvajamo sa `.dt.year`, mesec sa `.dt.month`, a dan sa `.dt.day`. Ove nove kolone se mogu koristiti u grupisanju, filtriranju ili za vizualizaciju sezonskih obrazaca.

```
df['Godina'] = df['Datum'].dt.year
df['Mesec'] = df['Datum'].dt.month
df['Dan'] = df['Datum'].dt.day
```

5. Kodiranje kategorijskih vrednosti (tekst u broj)

Kada radimo sa stvarnim podacima, često nailazimo na kolone koje sadrže **kategorijske (tekstualne) vrednosti**, kao što su: *Pol*, *Grad*, *Vrsta proizvoda*, *Status narudžbine* itd.

🔍 Problem: Mašinski algoritmi (npr. linearna regresija, stablo odlučivanja, KNN) **ne mogu direktno raditi sa tekstom**, jer očekuju numeričke vrednosti. Zbog toga moramo pretvoriti te tekstualne vrednosti u brojeve.

Postoje **dva glavna pristupa** za kodiranje kategorijskih podataka:

1) Label Encoding

- Svakoj kategoriji dodeljuje se jedinstven broj (npr. *Muško* = 1, *Žensko* = 0).
- Jednostavno i brzo.
- Dobro za **ordinalne vrednosti** (koje imaju redosled), kao što su: *nizak, srednji, visok*.

⚠️ Nije idealno za **nominalne vrednosti** (bez redosleda), jer brojevi mogu sugerisati lažnu hijerarhiju.

2) One-hot Encoding

- Svaka kategorija postaje posebna kolona.
- Ako imamo vrednosti: *Muško*, *Žensko* – dobijamo dve nove kolone: Pol_Muško i Pol_Žensko.
- Svaka red dobija 1 u odgovarajućoj koloni, a 0 u ostalim.
- Idealno za **nominalne vrednosti**, jer izbegava veštačku relaciju između kategorija.

Primer koda:

Imamo kolonu *Pol* sa tekstualnim vrednostima. Primenićemo obe metode.

```
from sklearn.preprocessing import LabelEncoder
import pandas as pd

# Ulazni podaci
df = pd.DataFrame({'Pol': ['Muško', 'Žensko', 'Žensko', 'Muško']})

# Label Encoding – svaka kategorija dobija broj (Muško=1, Žensko=0)
le = LabelEncoder()
df['Pol_encoded'] = le.fit_transform(df['Pol'])

# One-hot Encoding – svaka kategorija dobija svoju kolonu (binarnu)
df_onehot = pd.get_dummies(df['Pol'], prefix='Pol')
```

Na kraju:

- `df` sadrži originalnu kolonu i dodatnu kolonu `Pol_encoded` sa numeričkim oznakama.
- `df_onehot` sadrži dve kolone: `Pol_Muško i Pol_Žensko`, koje mogu direktno da se koriste u modelima.

Čišćenje i priprema podataka su **najvažniji i najzahtevniji deo analize podataka**. Ako preskočimo ovaj korak ili ga površno odradimo, kasnije faze analize mogu biti potpuno beskorisne. Kroz pet ključnih koraka – od nedostajućih vrednosti do kodiranja kategorija – postavljamo temelj za kvalitetnu, tačnu i pouzdanu analizu.

💡 **Ne zaboravite da ispratite i Google Colab notebook sa časa:**
https://colab.research.google.com/drive/1_MfGpAByHxQh3haKtl7Ugu31mPydcBJw?usp=sharing