ANALIZA VELIKIH PODATAKA

školska 2024/2025 godina

Vežba 6: Grupisanje i agregacija podataka u Pandas-u

Grupisanje i agregacija podataka su ključne tehnike u analizi podataka, posebno kada radimo sa velikim skupovima podataka. U Pandas-u, najmoćniji alati za ove operacije su:

- **groupby() metod:** omogućuje grupisanje podataka na osnovu vrednosti u kolonama i primenu agregatnih funkcija na svaku grupu.
- **Pivot tabele**: omogućuju reorganizaciju podataka tako da dobijemo tabelarni prikaz sa agregiranim vrednostima. One su slične groupby() metodu, ali pružaju više kontrole nad redovima i kolonama.
- **Resampling**: koristi se za vremenske serije – podatke koji uključuju vremenske oznake. Pomoću njega možemo menjati frekvenciju (npr. sa dnevne na mesečnu), kao i agregirati po vremenu.

Pored groupby(), pivot_table() i resample(), u analizi vremenskih serija često se koriste i metode **rolling()** i **expanding()**, koje omogućuju dinamičku analizu podataka kroz pokretne ili kumulativne prozore.

Više o njima sledi u nastavku, a sada možete pregledati tabelu za lakše snalaženje i pamćenje ovih metoda:

Tehnika	Kada se koristi	Ključni metod
groupby()	Grupisanje i agregacija po kategorijama	groupby()
Pivot tabela	Organizacija podataka u format redova/kolona	pivot_table()
Resampling	Promena vremenske frekvencije (npr. dnevno → mesečno)	resample()
rolling()	Analiza trendova u pokretnom prozoru	rolling()
expanding()	Kumulativna analiza svih prethodnih vrednosti	expanding()

🔷 Grupisanje podataka pomoću groupby()

Metod groupby() se koristi za **grupisanje redova po vrednostima u jednoj ili više kolona**, kako bi se zatim nad tim grupama primenile **agregatne funkcije** poput sum(), mean(), count(), min(), max(), itd. Ovo je posebno korisno kada želimo analizirati podatke po kategorijama (npr. po zemlji, proizvodu, mesecu...).

Osnovna sintaksa groupby() metoda:

df.groupby('kolona').agregatna_funkcija()

- **kolona**: Kolona na osnovu koje ćemo grupisati podatke.
- **agregatna_funkcija**: Funkcija koja se koristi za agregaciju podataka, kao što su sum(), mean(), count(), itd.

Primer: Grupisanje i agregacija na osnovu jedne kolone

Pretpostavimo da imamo dataset sa podacima o prodaji proizvoda. Grupisaćemo podatke po kategorijama proizvoda i izračunati ukupnu prodaju i prosečnu cenu po kategorijama.

```
import pandas as pd

# Učitavanje podataka
data = {
    'Proizvod': ['Laptop', 'Patike', 'Monitor', 'Jakna', 'Laptop', 'Vino'],
    'Kategorija': ['Elektronika', 'Moda', 'Elektronika',
                   'Moda', 'Elektronika', 'Pice'],
    'Prodaja': [1000, 1500, 1200, 2000, 1300, 1800],
    'Cena': [500, 300, 150, 300, 550, 180]
}

df = pd.DataFrame(data)

grouped = df.groupby('Kategorija').agg({'Prodaja': 'sum', 'Cena': 'mean'})

print(grouped)
```

Grupisali smo podatke prema kategorijama proizvoda ('Kategorija'). Za svaku kategoriju smo izračunali ukupnu prodaju (sum) i prosečnu cenu (mean).

Kategorija	Prodaja	Cena
Elektronika	3500	400.0
Moda	3500	300.0
Pice	1800	180.0

Primer: Grupisanje i agregacija na osnovu više kolona

Ako želimo grupisati podatke po više kolona, možemo koristiti više kolona kao argumente u groupby() metodi. Na primer, možemo grupisati podatke po zemlji i kategoriji proizvoda, i izračunati sumu prodaje i prosečnu cenu.

```
# Dodajemo kolonu 'Zemlja' u dataset
data = {
    'Proizvod': ['Laptop', 'Patike', 'Monitor', 'Jakna', 'Laptop', 'Vino'],
    'Kategorija': ['Elektronika', 'Moda', 'Elektronika', 'Moda',
                   'Elektronika', 'Pice'],
    'Zemlja': ['SAD', 'Kina', 'Nemačka', 'SAD', 'Vijetnam', 'Italija'],
    'Prodaja': [1000, 1500, 1200, 2000, 1300, 1800],
    'Cena': [500, 300, 150, 300, 550, 180]
}

df = pd.DataFrame(data)

# Grupisanje podataka po Zemlji i Kategoriji i agregacija
grouped = df.groupby(['Zemlja', 'Kategorija'])
                                .agg({'Prodaja': 'sum', 'Cena': 'mean'})

print(grouped)
```

```
                     Prodaja   Cena
Zemlja   Kategorija
Italija  Pice           1800  180.0
Kina     Moda           1500  300.0
Nemačka  Elektronika     1200  150.0
SAD      Elektronika     1000  500.0
         Moda            2000  300.0
Vijetnam Elektronika     1300  550.0
```

Još neki primeri sa grupisanjem

Broj recenzija po hotelu:
df.groupby('hotel_name')['review_text'].count()

Za svaki hotel dobijamo prosečnu ocenu, broj recenzija i standardnu devijaciju.
df.groupby('hotel_name')['rating'].agg(['mean', 'count', 'std'])

🔷 Organizacija podataka pomoću pivot_table()

Koncept pivot_table() reorganizuje podatke u **tabelarni format sa redovima i kolonama**, što omogućuje jasniji pregled odnosa između dimenzija. Sličan je groupby() metodu, ali pruža višedimenzionalne analize i agregacije.

Osnovna sintaksa pivot tabele:

df.pivot_table(values='vrednosti', index='redovi', columns='kolone', aggfunc='funkcija')

- **values**: Kolona koja sadrži vrednosti koje se agregiraju.
- **index**: Kolone koje se koriste kao redovi u pivot tabeli.
- **columns**: Kolone koje se koriste kao kolone u pivot tabeli.
- **aggfunc**: Agregatna funkcija koja se primenjuje (npr. sum, mean, count).

Primer: Pivot tabela sa mesecima i kategorijama

Pretpostavimo da imamo podatke o prodaji proizvoda po mesecima i želimo da napravimo pivot tabelu koja prikazuje sumu prodaje po mesecima i kategorijama proizvoda.

```
# Učitavanje podataka sa vremenskom komponentom
data = {
    'Datum': pd.to_datetime(['2021-01-01', '2021-01-05', '2021-02-15',
                             '2021-03-01', '2021-03-05']),
    'Kategorija': ['Elektronika', 'Obuća', 'Elektronika', 'Obuća',
                                                    'Elektronika'],
    'Prodaja': [1000, 1500, 1200, 2000, 1300]
}

df = pd.DataFrame(data)

# Dodajemo mesec kao novu kolonu
df['Mesec'] = df['Datum'].dt.month

# Kreiramo pivot tabelu sa popunjenim nedostajućim vrednostima kao 0
pivot_table = df.pivot_table(
    values='Prodaja',
    index='Mesec',
    columns='Kategorija',
    aggfunc='sum',
    fill_value=0
)

print(pivot_table)
```

Kategorija	Elektronika	Obuća
Mesec		
1	1000	1500
2	1200	0
3	1300	2000

Još neki primeri sa pivot tabelama

Prosečne ocene hotela po nacionalnosti:

pivot = pd.pivot_table(df, values='rating', index='hotel_name', columns='nationality', aggfunc='mean')

Svaka ćelija prikazuje prosečnu ocenu koju je određena nacionalnost dala hotelu.

Kombinovanje više agregatnih funkcija:

pd.pivot_table(df, values='rating', index='hotel_name', aggfunc=['mean', 'count', 'std'])

Na ovaj način, za svaku grupu (hotel) istovremeno izračunamo prosečnu ocenu, broj ocena i odstupanje ocena, što daje detaljniji uvid u podatke bez potrebe za dodatnim analizama.

🔷 **Promena vremenske frekvencije pomoću resample()**

Metod resample() se koristi za **vremenske serije**, gde želimo da promenimo učestalost podataka – npr. agregiramo dnevne podatke u mesečne. Potrebno je da indeks bude u datetime formatu.

Osnovna sintaksa:

```
df.set_index('vremenska_kolona').resample('frekvencija')
                .agg({'kolona1': 'agregatna_funkcija1', 'kolona2': 'agregatna_funkcija2'})
```

- **'vremenska_kolona':** Kolona sa vremenskim podacima, mora biti u datetime formatu i postavljena kao indeks.
- **'frekvencija':** Kod koji definiše novu vremensku učestalost.
- **'kolona1', 'kolona2':** Kolone nad kojima se vrši agregacija.
- **'agregatna_funkcija':** Funkcija poput 'sum', 'mean', 'count', 'max', itd.

Ključne karakteristike:

- resample() funkcioniše **slično kao groupby()**, ali grupiše na osnovu **vremenskog indeksa**, a ne na osnovu kolone sa kategorijama.
- Nakon primene resample(), najčešće se koristi metoda agg() ili direktne agregatne funkcije (sum(), mean(), count(), itd.).

📅 Najčešće korišćeni kodovi za vremensku frekvenciju:

Kod	Opis
'D'	Dnevno
'W'	Nedeljno
'ME'	Mesečno (kraj meseca)
'MS'	Mesečno (početak meseca)
'Q'	Kvartalno
'Y'	Godišnje

Primer: Pretvaramo dnevne podatke o prodaji u mesečne

```
# Učitavanje podataka sa vremenskom komponentom
data = {
    'Datum': pd.to_datetime(['2021-01-01', '2021-01-05', '2021-02-15',
                             '2021-03-01', '2021-03-05']),
    'Prodaja': [1000, 1500, 1200, 2000, 1300]
}

df = pd.DataFrame(data)
df.set_index('Datum', inplace=True)

# Resampling podataka na mesečnom nivou
monthly_sales = df.resample('ME').sum()

print(monthly_sales)
```

```
            Prodaja
Datum
2021-01-31     2500
2021-02-28     1200
2021-03-31     3300
```

Još neki primeri sa resample metodama

⭐ Popunjavanje nedostajućih vrednosti: U slučaju da za neki period nemamo podatke, možemo koristiti funkciju kao što je fillna(0) da popunimo te vrednosti.
monthly_avg = df.set_index('reviewed_at').resample('ME')['rating'].mean().fillna(0)

Mesečni prosek ocena hotela
df['reviewed_at'] = pd.to_datetime(df['reviewed_at'])
monthly_avg = df.set_index('reviewed_at').resample('ME')['rating'].mean()

Broj recenzija hotela po nedeljama
weekly_counts = df.set_index('reviewed_at').resample('W')['review_text'].count()

Slično group_by, i ovde možemo koristiti više agregatnih funkcija u kombinaciji sa agg().

🔁 rolling() – Pokretni prozor

Metoda rolling() omogućuje izračunavanje agregatnih statistika (kao što su prosek, suma, standardna devijacija itd.) u **prozorima fiksne veličine** koji se pomeraju kroz vremensku seriju. Ovo je korisno kada želite da analizirate podatke u kontekstu promena tokom vremena i kako se agregirane vrednosti ponašaju u određenim vremenskim intervalima.

Osnovna sintaksa:

df['column_name'].rolling(window=3).mean()

- **column_name:** Kolona u DataFrame-u na kojoj primenjujemo pokretni prozor.
- **window=3:** Veličina prozora (broj uzoraka u svakom prozoru). Na primer, window=3 znači da će za svaki trenutni podatak biti uzeta tri prethodna podatka za izračunavanje agregatne vrednosti.
- .mean(): Agregatna funkcija koja se koristi na prozoru. Možete koristiti i druge funkcije poput sum(), std(), min(), max(), itd.

Primer: Izračunavanje pokretnog proseka ocena hotela

Zamislite da analizirate ocene hotela koje su ostavili gosti, a želite da dobijete prosečne ocene za svaki period od 3 dana (ili recenzije). U ovom slučaju, koristićemo rolling() metod da izračunamo pokretni prosek ocena.

```
# Primer podataka
data = {
    'Date': pd.to_datetime(['2021-01-01', '2021-01-02', '2021-01-03',
                            '2021-01-04', '2021-01-05']),
    'Rating': [4, 3, 5, 4, 2]
}

df = pd.DataFrame(data)

# Pokretni prosek sa prozorom od 3 dana
df['Rolling_Mean'] = df['Rating'].rolling(window=3).mean()

# Zamena potencijalnih NaN vrednosti sa 0
df['Rolling_Mean'] = df['Rolling_Mean'].fillna(0)

print(df)
```

	Date	Rating	Rolling_Mean
0	2021-01-01	4	0.000000
1	2021-01-02	3	0.000000
2	2021-01-03	5	4.000000
3	2021-01-04	4	4.000000
4	2021-01-05	2	3.666667

📈 expanding() – Kumulativna agregacija

Metoda expanding() u Pandas-u omogućava izračunavanje agregatnih funkcija koje uključuju sve prethodne vrednosti u seriji do trenutnog reda. Za razliku od rolling() metode, koja koristi fiksni prozor podataka, expanding() koristi sve podatke od početka serije pa do trenutnog reda, stvarajući kumulativne statistike. Ovo je korisno kada želite da analizirate kako se podaci akumuliraju tokom vremena ili da pratite kako se agregirani rezultati razvijaju u odnosu na sve prethodne vrednosti.

Osnovna sintaksa:

df['rating'].expanding().mean()

- **column_name:** Kolona u DataFrame-u na kojoj primenjujemo kumulativnu agregaciju.
- **.expanding():** Pokreće kumulativnu funkciju, uzimajući sve prethodne vrednosti do trenutnog reda.
- .mean(): Agregatna funkcija koja se koristi na seriji. Možete koristiti i druge funkcije poput sum(), std(), min(), max(), itd.

Primer: Kumulativni prosek ocena

Zamislite da analizirate ocene proizvoda koje su ostavili korisnici i želite da izračunate kumulativni prosek ocena kako bi videli kako se ocene menjaju tokom vremena.

```
# Primer podataka
data = {
    'Date': pd.to_datetime(['2021-01-01', '2021-01-02', '2021-01-03',
                            '2021-01-04', '2021-01-05']),
    'Rating': [4, 3, 5, 4, 2]
}

df = pd.DataFrame(data)

# Kumulativni prosek sa expanding metodom
df['Expanding_Mean'] = df['Rating'].expanding().mean()

print(df)
```

	Date	Rating	Expanding_Mean
0	2021-01-01	4	4.0
1	2021-01-02	3	3.5
2	2021-01-03	5	4.0
3	2021-01-04	4	4.0
4	2021-01-05	2	3.6