9. Gyakorlat

Python kurzus

Bevezetés a Big Data adatelemzésbe

NumPy könyvtár – Numerikus számításokhoz

- Tömbök használata, amelyek különböző dimenziókat támogatnak (egy-, két- vagy többdimenziós).
- Hatékony matematikai műveletek tömbökkel.
- Beépített függvények pl. statisztikai számításokra, stb.

NumPy példa:

```
import numpy as np
# Egydimenziós tömb létrehozása
arr = np.array([1, 2, 3, 4, 5])
print("Egydimenziós tömb:", arr)
# Többdimenziós (mátrix) tömb
matrix = np.array([[1, 2, 3], [4, 5, 6]])
print("Mátrix:\n", matrix)
```

```
# Alapvető műveletek: összeg, átlag
arr_sum = np.sum(arr)
arr_mean = np.mean(arr)
print("Összeg:", arr_sum)
print("Átlag:", arr_mean)
```

Pandas könyvtár – Series: egydimenziós indexelt adattömb

```
import pandas as pd
# Series létrehozása
series = pd.Series([10, 20, 30, 40], index=['a', 'b', 'c', 'd'])
print("Series példa:\n", series)
```

```
Series példa:
a    10
b    20
c    30
d    40
dtype: int64
```

Pandas könyvtár – DataFrame: kétdimenziós adattömb

```
# DataFrame létrehozása
data = {
   'Nev': ['Anna', 'Béla', 'Cecília'],
   'Kor': [29, 34, 22],
   'Varos': ['Budapest', 'Debrecen', 'Szeged']
}
df = pd.DataFrame(data)
print("\nDataFrame példa:\n", df)
# Adatokhoz való hozzáférés
print("\nElső sor adatai:")
print(df.iloc[0])
print("\nÉletkor oszlop adatai:")
print(df['Kor'])
```

```
DataFrame példa:
       Nev  Kor      Varos
0     Anna   29  Budapest
1     Béla   34  Debrecen
2  Cecília   22    Szeged

Első sor adatai:
Nev        Anna
Kor          29
Varos    Budapest
Name: 0, dtype: object

Életkor oszlop adatai:
0    29
1    34
2    22
Name: Kor, dtype: int64
```

Alapvető műveletek a DataFrame-ekkel:

- **Adatok importálása és exportálása:**
  - CSV beolvasás: pd.read_csv('fajl_neve.csv')
  - CSV exportálás: df.to_csv('output.csv', index=False)
- **Alapvető adatvizsgálat:**
  - Az első néhány sor megtekintése: df.head()
  - Adatstatisztika: df.describe()
  - Adattípusok ellenőrzése: df.dtypes
- **Adatok manipulálása:**
  - Sorok és oszlopok kiválasztása: df[['Nev', 'Kor']]
  - Adatok szűrése: df[df['Kor'] > 30]
  - Oszlopok hozzáadása vagy módosítása: df['Uj_Oszlop'] = df['Kor'] * 2

Pandas példa: adatok beolvasása fájlból és fájlba írása

```
df = pd.read_csv('fajl_neve.csv')
df.head()        # DataFrame megjelenítése
df.describe()    # Statisztikai jellemzők
df.dtypes          #Oszlopok típusai
df[['Nev', 'Kor']]
df[df['Kor'] > 25]

# Új oszlop hozzáadása
df['Kor x2'] = df['Kor'] * 2

# Szűrés: csak a 'Debrecen'-ben élők
debrecen_df = df[df['Varos'] == 'Debrecen']

# Átlagéletkor kiszámítása
atlag_eletkor = df['Kor'].mean()
print("\nÁtlagéletkor:", atlag_eletkor)

df.to_csv('kimenet.csv', index=False)
```

Az index=False eredménye, hogy az output.csv fájlban nem lesz benne az első index oszlop.

További DataFrame műveletek

Új oszlop hozzáadása:

```
# Új oszlop hozzáadása a df DataFrame-hez
df['jegyek'] = [4.5, 3.8, 4.0]
print("\nÚj oszloppal bővített DataFrame:\n", df)
```

Adatok rendezése:

```
# Adatok rendezése ‚Kor' szerint csökkenő sorrendben
rendezett_df = df.sort_values(by='Kor', ascending=False)
print("\nRendezett DataFrame:\n", rendezett_df)
```

Átlag és maximum:

```
# Átlagéletkor kiszámítása
atlag_kor = df['Kor'].mean()
print("\nÁtlagéletkor:", atlag_kor)

# Legmagasabb életkor megkeresése
max_kor = df['Kor'].max()
print("Legidősebb diák életkora:", max_kor)
```

Eredmények:

Új oszloppal bővített DataFrame:

	Nev	Kor	Varos	jegyek
0	Anna	29	Budapest	4.5
1	Béla	34	Debrecen	3.8
2	Cecília	22	Szeged	4.0

Rendezett DataFrame:

	Nev	Kor	Varos	jegyek
1	Béla	34	Debrecen	3.8
0	Anna	29	Budapest	4.5
2	Cecília	22	Szeged	4.0

Átlagéletkor: 28.333333333333332
Legidősebb életkor: 34

A **Pandas** könyvtár számos módszert biztosít különböző formátumú adatok importálására és azok előkészítésére az elemzéshez.

1. lépés: Adatok beolvasása

A leggyakoribb adatforrások közé tartoznak a **CSV**, **Excel** és **SQL** adatbázisok. Ezek beolvasására a Pandas különböző függvényeket biztosít:

CSV fájl beolvasása:

```
import pandas as pd

# CSV fájl beolvasása
adatok_df = pd.read_csv('pelda_adatok.csv')
print("Beolvasott adatok:\n", adatok_df.head())
```

2. lépés: Adatok előkészítése és tisztítása

Hiányzó adatok eltávolítása és hiányzó adatok kitöltése:

```
# Sorok eltávolítása, ahol hiányzó adat van
adatok_df = adatok_df.dropna()
print("Hiányzó adatok eltávolítása után:\n", adatok_df.head())

# Hiányzó értékek kitöltése egy adott értékkel
adatok_df['kor'] = adatok_df['kor'].fillna(30)

# Hiányzó értékek kitöltése az oszlop átlagával
adatok_df['jegyek'] = adatok_df['jegyek'].fillna(adatok_df['jegyek'].mean())
print("Hiányzó értékek kitöltése átlaggal:\n", adatok_df.head())
```

Adatok típusának konverziója: Az adatelemzési folyamat során ha szükséges, az adatok típusát konvertálni lehet, például szöveget számmá vagy dátumformátummá.

Típuskonverzió egész számokká és dátumformátum konverzió:

```
# 'kor' oszlop konvertálása egész számokká
adatok_df['kor'] = adatok_df['kor'].astype(int)
print("Adatok típusa konvertálás után:\n", adatok_df.dtypes)

# 'datum' oszlop konvertálása dátumformátummá
adatok_df['datum'] = pd.to_datetime(adatok_df['datum'])
```

Duplikált sorok eltávolítása:

```
# Duplikált sorok eltávolítása
adatok_df = adatok_df.drop_duplicates()
```

3. lépés: Adatmanipulációk: szűrés, rendezés, aggregálás

Az adatmanipuláció a Pandas egyik legerősebb funkciója, amely lehetővé teszi az adatok hatékony feldolgozását, szűrését, rendezését és aggregálását. Ez a lépés kulcsfontosságú az adatok előkészítésében és az elemzési folyamat során.

Szűrés: Az adatok szűrése lehetővé teszi, hogy csak a megadott feltételnek megfelelő sorokat tartsuk meg egy új DataFrame-ben.

Rendezés: A rendezés lehetővé teszi az adatok sorainak újra sorrendezését egy vagy több oszlop alapján.

Aggregálás: az adatok **összesítése** különböző metrikák segítségével, például **átlag, összeg, minimum, maximum**.

Összetett példa

```
# Adatok szűrése, ahol a 'kor' nagyobb 25-nél, majd rendezés 'név' szerint
szurt_rendezett_diakok = diákok_df[diakok_df['kor'] > 25].sort_values(by='nev')
print("Szűrt és név szerint rendezett diákok:\n", szurt_rendezett_diakok)

# Csoportosítás város szerint, majd a diákok számának megszámolása
varosi_diak_szam = dakok_df.groupby('varos').size()
print("Diákok száma városonként:\n", varosi_diak_szam)
```

```
Szűrt és rendezett diákok:
      nev  kor      varos
0   Anna    29  Budapest
1    Béla    34  Debrecen
3  Dávid    30      Pécs

Diákok száma városonként:
 varos
 Budapest   1
 Debrecen   1
 Miskolc    1
 Pécs       1
 Szeged     1
```

4. lépés: Adatfeldolgozási feladatok

A **Pandas** könyvtár segítségével, az adatokat **előkészíthetjük, módosíthatjuk**, és **információkat nyerhetünk ki** belőlük.

4.1. Adattranszformációk

Például új oszlopok hozzáadását és a meglévő oszlopok értékeinek módosítását végezzük el.

Új oszlop létrehozása:

```
import pandas as pd
# Minta DataFrame létrehozása
adatok = {
   'nev': ['Anna', 'Béla', 'Cecília', 'Dávid', 'Emma'],
   'kor': [29, 34, 22, 30, 25],
   'varos': ['Budapest', 'Debrecen', 'Szeged', 'Pécs', 'Miskolc']
}
diakok_df = pd.DataFrame(adatok)
# Új oszlop hozzáadása, amely a kor kétszerese
diakok_df['kor_ketszer'] = diakok_df['kor'] * 2
print("Új oszloppal bővített DataFrame:\n", diakok_df)
```

	nev	kor	varos	kor_ketszer
0	Anna	29	Budapest	58
1	Béla	34	Debrecen	68
2	Cecília	22	Szeged	44
3	Dávid	30	Pécs	60
4	Emma	25	Miskolc	50

4.2. Adatok normalizálása

Az adatok normalizálása során az értékeket egy közös skálára helyezzük, ami segíthet az adatok összehasonlíthatóságában.

Értékek skálázása 0 és 1 közé:

```
diakok_df['kor_normalizalt'] = (diakok_df['kor'] - diakok_df['kor'].min()) /
(diakok_df['kor'].max() - diakok_df['kor'].min())
print("Normalizált 'kor' oszlop:\n", diakok_df)
```

Normalizált 'kor' oszlop :

	nev	kor	varos	kor_ketszer	kor_normalizalt
0	Anna	29	Budapest	58	0.636364
1	Béla	34	Debrecen	68	1.000000
2	Cecília	22	Szeged	44	0.000000
3	Dávid	30	Pécs	60	0.727273
4	Emma	25	Miskolc	50	0.272727

4.3. Feltételes adatmanipuláció

Az adatok manipulációja során gyakran szükséges egyes sorokat vagy oszlopokat feltételek alapján módosítani.

Új oszlop létrehozása feltétel vizsgálattal:

```
# Új oszlop, amely megjelöli, ha a kor 30 év felett van
diakok_df['idos_e'] = diakok_df['kor'].apply(lambda x: 'Igen' if x > 30 else 'Nem')
print("Feltételes oszloppal bővített DataFrame:\n", diakok_df)
```

	nev	kor	varos	kor_ketszer	kor_normalizalt	idos_e
0	Anna	29	Budapest	58	0.636364	Nem
1	Béla	34	Debrecen	68	1.000000	Igen
2	Cecília	22	Szeged	44	0.000000	Nem
3	Dávid	30	Pécs	60	0.727273	Nem
4	Emma	25	Miskolc	50	0.272727	Nem

Találkozunk az előadáson!