Uvodni primjeri za upoznavanje sa novim funkcionalnostima i bibliotekama

Pandas

```
1 import pandas as pd # ucitaj Pandas
2 import numpy as np # ucitaj NumPy
3 data = pd.read_csv('dataset.csv') # ucitaj podatke
4 data
5
```

```
1 data.head() #uzimamo prvih 5 elemenata po defaultu
```

```
1 data.tail(2) #uzimamo zadnja dva reda
```

```
1 data_transponovano = data.T
2 data_transponovano #transponovanje podataka
```

```
1 data["Indeks"] #odabir jedne kolone
```

```
0     94-ST
1     77-ST
2     69-ST
3     79-ST
4     89-ST
5     78-ST
```

```
6     93-ST
7     68-ST
8     83-ST
9     92-ST
10    65-ST
11    70-ST
12    90-ST
13    97-ST
14    84-ST
15    98-ST
16    91-ST
17    87-ST
18    76-ST
19    96-ST
20    66-ST
21    62-ST
22    74-ST
23    86-ST
24    73-ST
25    80-ST
26    63-ST
27    72-ST
28    95-ST
29    85-ST
30    99-ST
31    41-ST
32    17-ST
33    82-ST
34    15-ST
35    64-ST
36    44-ST
37    71-ST
38    54-ST
39    60-ST
40    48-ST
41    61-ST
42     1-ST
43    88-ST
44    75-ST
45    67-ST
46    81-ST
Name: Indeks, dtype: object
```

```
1 data[3:8] #uzimanje podsekvence redova
```

```
# nadji indeks studenta u 5. redu
data.loc[5, 'Indeks']
```

```
# svi studenti koji imaju ukupno preko 43 boda
data.loc[data['UKUPNO'] > 43]
```

```
# izlistaj indeks i prisustvo za trećeg,četvrtog i petog studenta
data.loc[2:4, ['Indeks', 'Prisustvo']]
```

```
df = pd.DataFrame(np.arange(25).reshape(5,5),
index=[5,4,3,2,1],
columns=['x', 'y', 'z', 'v', 'i'])

df
```

```
# loc radi sa kljucem (index) reda,
# i sa nazivima kolona
df.loc[1, 'x'] #uzimamo element sto se nalazi u redu sa id 1 i koloni x
```

```
20
```

```
# iloc radi isključivo sa stvarnim (integer) indeksima
# (kao standardne matrice)
df.iloc[1, 0] #iloc posmatra dataframe objekat kao stvarnu matricu, uzimamo element na inc
```

```
5
```

```
data.replace('/', np.nan, inplace=True)
data.head() # svaka pojava / zamijenjena sa NaN
```

```
data.head()
```

```
data.drop([2, 3, 4], inplace=True) # izbaci redove na indeksima 2, 3, i 4
data.head()
```

```
# izbaci kolonu ispit2
data.drop(columns=['Ispit2'], inplace=True)
data.head()
```

```
data
```

Primjeri koristenja Scikit-Learn (sklearn) biblioteke

```
1 data = pd.read_csv('dataset.csv') # ucitaj podatke
2 data.replace('?', np.nan, inplace=True)
```

```
data.replace('?', np.nan, inplace=True)
data
```

```
from sklearn.impute import SimpleImputer
si = SimpleImputer(strategy='mean')
print('Prije zamjene: ')
print(data.loc[:5, 'Ispit2'])
data.loc[:, 'Ispit2'] = si.fit_transform(data.loc[:, 'Ispit2'].values.reshape(-1,1))
print('Nakon zamjene: ')
print(data.loc[:5, 'Ispit2'])
```

```
Prije zamjene:
0     7.5
1     NaN
2     NaN
3     4.8
4    12.5
5     NaN
Name: Ispit2, dtype: object
Nakon zamjene:
0     7.500000
1     9.323529
```

```
2     9.323529
3     4.800000
4    12.500000
5     9.323529
Name: Ispit2, dtype: float64
```

```
from sklearn.preprocessing import scale
data['Ispit2'] = scale(data['Ispit2']) # Z score
data.loc[:5, 'Ispit2']
```

```
0   -7.156096e-01
1    6.970976e-16
2    6.970976e-16
3   -1.775173e+00
4    1.246546e+00
5    6.970976e-16
Name: Ispit2, dtype: float64
```

```
from sklearn.preprocessing import MinMaxScaler
mm = MinMaxScaler()
data['Ispit2'] = mm.fit_transform(data['Ispit2'].values.reshape(-1,1))
data.loc[:5, 'Ispit2']
```

```
0    0.379747
1    0.495160
2    0.495160
3    0.208861
4    0.696203
5    0.495160
Name: Ispit2, dtype: float64
```

```
import numpy as np
from sklearn.model_selection import train_test_split

X, Y = np.arange(10).reshape((5, 2)), range(5)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.33,
random_state=42)
X,Y
```

```
(array([[0, 1],
        [2, 3],
        [4, 5],
        [6, 7],
        [8, 9]]), range(0, 5))
```

Primjer koristenja Pickle datoteke

```
import pickle
import os
import numpy as np
niz = np.arange(1000000)
print("Zauzima {0} kB".format(niz.nbytes/1000))
pickle.dump(niz, open( "datoteka.p", "wb" ), protocol=pickle.HIGHEST_PROTOCOL)
velicina = os.path.getsize('datoteka.p')
print("Datoteka zauzima {0} kB".format(velicina/1000))
niz2 = pickle.load(open("datoteka.p", 'rb'))
if (niz2 == niz).all():
  print("Nizovi su identicni!")
```

```
Zauzima 8000.0 kB
Datoteka zauzima 8000.162 kB
Nizovi su identicni!
```

3 Zadaci za rad u laboratoriji

Predviđeno je da se svi zadaci u nastavku rade u sklopu Jupyter Notebook okruženja. Svaki podzadatak treba biti zasebna Jupyter ćelija.

Zadatak 1 - Obrada podataka kroz Pandas

Uz vježbu ste dobili datoteku izvjestaj.csv. Potrebno je, koristeći Pandas:

a) Učitati datoteku kao Pandas DataFrame, te prikazati prvih 5 i posljednjih 10 unosa u istoj

```
#Priprema podataka za prvi zadatak
import numpy as np # ucitaj NumPy
import pandas as pd # ucitaj Pandas

data = pd.read_csv('dataset.csv') # ucitaj podatke

data.head() #prvih 5
data.tail(10) #zadnjih 10
```

b) Ispisati samo podatke vezane za redovne parcijalne ispite (kolone Ispit1 i Ispit2);

```
1 data.loc[:, ['Ispit1', 'Ispit2']]
```

c) Ispisati sve studente koji su izgubili prisustvo

```
data.loc[data['Prisustvo'] == 0]
```

c) Ispisati indeks, ukupne bodove, i ocjenu, za sve studente koji su upisali ocjenu 8 ili više;

```
1 data.loc[data['Ocjena'] >= '8', ['Indeks', 'UKUPNO', 'Ocjena']]
```

e) Sve nedostajuće vrijednosti (označene sa simbolom /) zamijeniti sa vrijednošću np.nan

```
1 data.replace('/', np.nan, inplace=True)
2 data
```

f) Iz skupa podataka odbaciti sve studente koji nemaju upisanu ocjenu

```
1 data.dropna(subset=['Ocjena'])
```

g) Kreirati novu kolonu Ispit1_final u koju ćete za svakog studenta upisati onaj rezultat ispita koji je bolji (na primjer, ako je student bolje uradio popravni ispit, onda tu pišete bodove sa popravnog, a u protivnom sa redovnog);

```
1 ispitiRedovni = data["Ispit1"]
2 ispitiPopravni = data["Ispit1_popravni"]
```

```
ispitiFinal = []
for i in range(0,len(ispitiRedovni)):
  if(isinstance(ispitiRedovni[i],str) != True and isinstance(ispitiPopravni[i],str)!=True)
    ispitiFinal.append(ispitiRedovni[i])
  elif (isinstance(ispitiRedovni[i],str) != True ): #Ako nije izlazio na redovni, automats
    ispitiFinal.append(ispitiPopravni[i])
  elif (isinstance(ispitiPopravni[i],str) != True ): #Ako nije izlazio na popravni, automa
    ispitiFinal.append(ispitiRedovni[i])
  elif (float(ispitiRedovni[i])>=float(ispitiPopravni[i])): #Ako je vise imao na redovnom
    ispitiFinal.append(ispitiRedovni[i])
  else: #U ostalim slucajevima upisujemo popravni
    ispitiFinal.append(ispitiPopravni[i])

data['Ispit1_final']=ispitiFinal
data

```

h) Ponoviti postupak za Ispit2

```
ispitiRedovni = data["Ispit2"]
ispitiPopravni = data["Ispit2_popravni"]
```

```
ispitiFinal = []
for i in range(0,len(ispitiRedovni)):
  if(isinstance(ispitiRedovni[i],str) != True and isinstance(ispitiPopravni[i],str)!=True)
    ispitiFinal.append(ispitiRedovni[i])
  elif (isinstance(ispitiRedovni[i],str) != True ): #Ako nije izlazio na redovni, automats
    ispitiFinal.append(ispitiPopravni[i])
  elif (isinstance(ispitiPopravni[i],str) != True ): #Ako nije izlazio na popravni, automa
    ispitiFinal.append(ispitiRedovni[i])
  elif (float(ispitiRedovni[i])>=float(ispitiPopravni[i])): #Ako je vise imao na redovnom
    ispitiFinal.append(ispitiRedovni[i])
  else: #U ostalim slucajevima upisujemo popravni
    ispitiFinal.append(ispitiPopravni[i])

data['Ispit2_final']=ispitiFinal
data
```

i) Odbaciti četiri stare kolone za ispite

```
data.drop(columns=['Ispit1','Ispit2','Ispit1_popravni', 'Ispit2_popravni'], inplace=True)
data
```

j) Skup podataka sačuvati kao CSV datoteku pod nazivom izvjestaj_modificirano.csv. Koristiti znak tačka-zarez kao separator

```
1 data.to_csv('izvjestaj_modificirano.csv',index=False)
2
3
```

k) Skup podataka sačuvati kao Pickle datoteku pod nazivom izvjestaj_modificirano_pickle.p.

```
1 import pickle
2 pickle.dump(data, open( "izvjestaj_modificirano.p", "wb" ), protocol=pickle.HIGHEST_PROTOC
```

Zadatak 2 - Normalizacija podataka pomoću Sklearn

a) Učitati datoteku izvjestaj.csv kao Pandas DataFrame;

```
1 import pandas as pd
2 import numpy as np # ucitaj NumPy
3 data = pd.read_csv('dataset.csv') # ucitaj podatke
4 data.replace('/', np.nan, inplace=True)
5 data
```

b) Za kolone koje predstavljaju redovne ispite, izvršiti zamjenu nedostajućih vrijednosti strategijom median

```
from sklearn.impute import SimpleImputer
si = SimpleImputer(strategy='median')

print('Prije zamjene: ')
print(data.loc[:5, 'Ispit1'])
print(data.loc[:5, 'Ispit2'])
data.loc[:, 'Ispit1'] = si.fit_transform(data.loc[:, 'Ispit1'].values.reshape(-1,1))
data.loc[:, 'Ispit2'] = si.fit_transform(data.loc[:, 'Ispit2'].values.reshape(-1,1))
print('Nakon zamjene: ')
print(data.loc[:5, 'Ispit1'])
print(data.loc[:5, 'Ispit2'])
```

```
Prije zamjene:
0     9.4
1     8.5
2      17
3     7.6
4    10.6
5     NaN
Name: Ispit1, dtype: object
0     7.5
1     NaN
2     NaN
3     4.8
4    12.5
5     NaN
Name: Ispit2, dtype: object
Nakon zamjene:
0     9.4
1     8.5
2    17.0
3     7.6
4    10.6
5    12.5
Name: Ispit1, dtype: float64
0     7.5
1     8.7
2     8.7
3     4.8
4    12.5
5     8.7
Name: Ispit2, dtype: float64
```

c) Za kolone koje predstavljaju popravne ispite, izvršiti zamjenu nedostajućih vrijednosti pomoću strategije mean

```
from sklearn.impute import SimpleImputer
si = SimpleImputer(strategy='mean')

print('Prije zamjene: ')
print(data.loc[:5, 'Ispit1_popravni'])
print(data.loc[:5, 'Ispit2_popravni'])
data.loc[:, 'Ispit1_popravni'] = si.fit_transform(data.loc[:, 'Ispit1_popravni'].values.re
data.loc[:, 'Ispit2_popravni'] = si.fit_transform(data.loc[:, 'Ispit2_popravni'].values.re
print('Nakon zamjene: ')
print(data.loc[:5, 'Ispit1_popravni'])
print(data.loc[:5, 'Ispit2_popravni'])
```

```
Prije zamjene:
0    12.08
1      NaN
2      NaN
3      NaN
4    12.58
```

```
5    13.08
Name: Ispit1_popravni, dtype: object
0    13.73
1      NaN
2      NaN
3      NaN
4      NaN
5    12.83
Name: Ispit2_popravni, dtype: object
Nakon zamjene:
0    12.080
1    10.657
2    10.657
3    10.657
4    12.580
5    13.080
Name: Ispit1_popravni, dtype: float64
0    13.730000
1    14.662727
2    14.662727
3    14.662727
4    14.662727
5    12.830000
Name: Ispit2_popravni, dtype: float64
```

d) Izvršiti Z-score normalizaciju vrijednosti za redovne parcijalne ispite

```
from sklearn.preprocessing import scale
print("Ispit 1 prije normalizacije")
print(data.loc[:5,'Ispit1'])
data['Ispit1'] = scale(data['Ispit1'])
print("Ispit 1 poslije normalizacije")
print(data.loc[:5, 'Ispit1'])

print("Ispit 2 prije normalizacije")
print(data.loc[:5,'Ispit2'])
data['Ispit2'] = scale(data['Ispit2'])
print("Ispit 2 poslije normalizacije")
print(data.loc[:5, 'Ispit2'])
```

```
Ispit 1 prije normalizacije
0     9.4
1     8.5
2    17.0
3     7.6
4    10.6
5    12.5
Name: Ispit1, dtype: float64
Ispit 1 poslije normalizacije
0   -1.561766
1   -2.029964
2    2.391904
3   -2.498161
```

```
4   -0.937502
5    0.050915
Name: Ispit1, dtype: float64
Ispit 2 prije normalizacije
0     7.5
1     8.7
2     8.7
3     4.8
4    12.5
5     8.7
Name: Ispit2, dtype: float64
Ispit 2 poslije normalizacije
0   -0.555596
1   -0.087900
2   -0.087900
3   -1.607912
4    1.393136
5   -0.087900
Name: Ispit2, dtype: float64
```

e) Izvršiti MinMax normalizaciju vrijednosti za popravne parcijalne ispite

```
from sklearn.preprocessing import MinMaxScaler
mm = MinMaxScaler()

print("Ispit1_popravni prije normalizacije")
print(data.loc[:5,'Ispit1_popravni'])
data['Ispit1_popravni'] = mm.fit_transform(data['Ispit1_popravni'].values.reshape(-1,1))
print("Ispit1_popravni poslije normalizacije")
print(data.loc[:5, 'Ispit1_popravni'])

print("Ispit2_popravni prije normalizacije")
print(data.loc[:5,'Ispit2_popravni'])
data['Ispit2_popravni'] = mm.fit_transform(data['Ispit2_popravni'].values.reshape(-1,1))
print("Ispit2_popravni poslije normalizacije")
print(data.loc[:5, 'Ispit2_popravni'])
```

```
Ispit1_popravni prije normalizacije
0    12.080
1    10.657
2    10.657
3    10.657
4    12.580
5    13.080
Name: Ispit1_popravni, dtype: float64
Ispit1_popravni poslije normalizacije
0    0.795918
1    0.679755
2    0.679755
3    0.679755
4    0.836735
5    0.877551
Name: Ispit1_popravni, dtype: float64
```

```
Ispit2_popravni prije normalizacije
0    13.730000
1    14.662727
2    14.662727
3    14.662727
4    14.662727
5    12.830000
Name: Ispit2_popravni, dtype: float64
Ispit2_popravni poslije normalizacije
0    0.401146
1    0.490232
2    0.490232
3    0.490232
4    0.490232
5    0.315186
Name: Ispit2_popravni, dtype: float64
```

f) Sve ostale nedostajuće vrijednosti u skupu podataka zamijeniti sa nulama

```
1 data.replace(np.nan, '0', inplace=True)
2 data
```

g) Izdvojiti kolonu Ocjena u posebnu varijablu, na način da sada u originalnom skupu podataka ta kolona više ne postoji

```
print("Prije izdvajanja kolone Ocjena: ")
print(data)
ocjene = data.pop("Ocjena")
print("Poslije izdvajanja kolone Ocjena: ")
print(data)
print("Ocjene: ")
print(ocjene)
```

```
Prije izdvajanja kolone Ocjena:
    id Indeks  Prisustvo    Ispit1    Ispit2  Ispit1_popravni  \
0    1  94-ST         10 -1.561766 -0.555596         0.795918
1    2  77-ST         10 -2.029964 -0.087900         0.679755
2    3  69-ST         10  2.391904 -0.087900         0.679755
3    4  79-ST          0 -2.498161 -1.607912         0.679755
4    5  89-ST         10 -0.937502  1.393136         0.836735
5    6  78-ST          0  0.050915 -0.087900         0.877551
6    7  93-ST         10  0.050915  1.393136         0.844082
7    8  68-ST         10  0.050915 -0.087900         0.679755
8    9  83-ST          0  0.050915 -0.087900         0.679755
9   10  92-ST         10  0.050915 -0.087900         0.679755
10  11  65-ST         10  0.050915 -0.087900         0.679755
11  12  70-ST         10  0.311025 -0.438672         0.679755
12  13  90-ST         10  1.663596 -0.477647         0.679755
13  14  97-ST         10  0.050915 -0.087900         0.679755
14  15  84-ST         10  0.050915 -0.087900         0.679755
15  16  98-ST         10  0.050915 -0.438672         0.679755
16  17  91-ST         10  0.050915 -0.087900         0.679755
17  18  87-ST         10  0.050915 -0.087900         0.679755
18  19  76-ST         10  0.050915 -0.087900         0.679755
19  20  96-ST         10  0.050915 -0.087900         0.679755
20  21  66-ST         10  0.831245  0.301846         0.679755
21  22  62-ST         10  0.050915 -0.087900         0.679755
22  23  74-ST          0 -0.885480 -0.087900         1.000000
23  24  86-ST         10  0.623157  1.393136         0.679755
24  25  73-ST         10  0.050915 -0.945343         0.679755
25  26  80-ST         10  0.050915 -0.087900         0.679755
26  27  63-ST         10  2.183816 -0.087900         0.679755
27  28  72-ST         10  0.050915 -0.087900         0.679755
28  29  95-ST          0 -0.833458  1.393136         0.401633
29  30  85-ST         10  0.050915  1.393136         0.679755
30  31  99-ST         10  0.050915 -0.087900         0.966531
31  32  41-ST          0  0.050915 -0.087900         0.679755
32  33  17-ST          0 -2.550183 -2.816126         0.000000
33  34  82-ST          0 -1.249634 -2.894075         0.679755
34  35  15-ST          0  0.050915 -0.087900         0.679755
35  36  64-ST         10  0.467091  2.367503         0.679755
36  37  44-ST         10  0.050915 -0.087900         0.462857
37  38  71-ST         10  0.050915 -0.087900         0.679755
38  39  54-ST         10  0.050915 -0.087900         0.679755
39  40  60-ST         10 -1.249634 -0.087900         0.679755
40  41  48-ST         10  0.050915 -0.087900         0.679755
```

```
41  42  61-ST         10  0.050915 -0.087900         0.679755
42  43   1-ST         10  0.050915 -0.087900         0.679755
43  44  88-ST         10  0.050915 -0.087900         0.679755
44  45  75-ST         10  0.050915 -0.087900         0.612245
45  46  67-ST         10  2.131794 -0.087900         0.679755
46  47  81-ST         10  1.715618  3.263919         0.679755

    Ispit2_popravni  UKUPNO Ocjena
0          0.401146   35.81      7
1          0.490232   35.50      7
2          0.490232   39.50      9
3          0.490232   12.40      0
4          0.490232   35.08      6
5          0.315186   25.91      6
6          0.490232   35.17      8
```

h) Konvertovati obje varijable (originalni skup podataka bez kolone Ocjena, kao i posebnu kolonu Ocjena) u NumPy nizove

```
1 print("Prije konvertovanja: ")
2 print(data)
3 print(ocjene)
4 print("Poslije konvertovanja: ")
5 data = data.to_numpy()
6 ocjene = ocjene.to_numpy()
7 print(data)
8 print(ocjene)
```

```
 0.6797551020408164 0.4902318312060433 42.06]
[11 '65-ST' 10 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 43.19]
[12 '70-ST' 10 0.3110249685383584 -0.4386721044704893 0.6797551020408164
 0.646609360076409 39.3]
[13 '90-ST' 10 1.663596184032576 -0.47764675269376516 0.6797551020408164
 0.45558739255014336 39.9]
[14 '97-ST' 10 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 41.05]
[15 '84-ST' 10 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 43.56]
[16 '98-ST' 10 0.050915119404855075 -0.4386721044704893
 0.6797551020408164 0.0 38.5]
[17 '91-ST' 10 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 43.48]
[18 '87-ST' 10 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 35.89]
[19 '76-ST' 10 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 27.18]
[20 '96-ST' 10 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 41.4]
[21 '66-ST' 10 0.8312446668053652 0.30184621177175464 0.6797551020408164
 0.617956064947469 40.0]
[22 '62-ST' 10 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 42.0]
[23 '74-ST' 0 -0.8854803374757575 -0.08790027046100546
 0.9999999999999999 0.20725883476599816 27.08]
```

```
[24 '86-ST' 10 0.6231567874985623 1.3931363620234831 0.6797551020408164
 0.4902318312060433 36.1]
[25 '73-ST' 10 0.050915119404855075 -0.9453425313730773
 0.6797551020408164 0.9111747851002866 41.57]
[26 '80-ST' 10 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 39.45]
[27 '63-ST' 10 2.1838158822995837 -0.08790027046100546
 0.6797551020408164 1.0 46.6]
[28 '72-ST' 10 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 18.41]
[29 '95-ST' 0 -0.833458367649056 1.3931363620234831 0.40163265306122453
 0.4902318312060433 23.5]
[30 '85-ST' 10 0.050915119404855075 1.3931363620234831
 0.6797551020408164 0.4902318312060433 35.25]
[31 '99-ST' 10 0.050915119404855075 -0.08790027046100546
 0.9665306122448979 0.4902318312060433 24.17]
[32 '41-ST' 0 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 5.87]
[33 '17-ST' 0 -2.5501833719301787 -2.8161256460903257 0.0
 0.4902318312060433 9.2]
[34 '82-ST' 0 -1.2496341262626618 -2.8940749425368777 0.6797551020408164
 0.4902318312060433 16.0]
[35 '15-ST' 0 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 0.0]
[36 '64-ST' 10 0.46709087801846083 2.367502567605383 0.6797551020408164
 0.4902318312060433 38.3]
[37 '44-ST' 10 0.050915119404855075 -0.08790027046100546
 0.46285714285714286 0.45845272206303733 32.33]
[38 '71-ST' 10 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 42.16]
[39 '54-ST' 10 0.050915119404855075 -0.08790027046100546
```

i) Ukoliko pretpostavimo da NumPy niz sa ocjenama predstavlja labele, a niz sa ostalim kolonama atribute (značajke), izvršiti podjelu na trening i testni skup podataka, pri čemu za testni skup treba uzeti 20% podataka.

```
1 import numpy as np
2 from sklearn.model_selection import train_test_split
3
4
5 data_train, data_test, ocjene_train, ocjene_test = train_test_split(data, ocjene, test_siz
6 print("Data train: ")
7 print(data_train)
8 print("Data test: ")
9 print(data_test)
10 print("Ocjene train: ")
11 print(ocjene_train)
12 print("Ocjene test: ")
13 print(ocjene_test)
```

```
[35 '15-ST' 0 0.050915119404855075 -0.08790027046100546
 0.6797551020408164 0.4902318312060433 0.0]
```

```
 [2 '77-ST' 10 -2.029963673663172 -0.08790027046100546 0.6797551020408164
  0.4902318312060433 35.5]
 [42 '61-ST' 10 0.050915119404855075 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 23.12]
 [22 '62-ST' 10 0.050915119404855075 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 42.0]
 [3 '69-ST' 10 2.3919037616063856 -0.08790027046100546 0.6797551020408164
  0.4902318312060433 39.5]
 [36 '64-ST' 10 0.46709087801846083 2.367502567605383 0.6797551020408164
  0.4902318312060433 38.3]
 [24 '86-ST' 10 0.6231567874985623 1.3931363620234831 0.6797551020408164
  0.4902318312060433 36.1]
 [38 '71-ST' 10 0.050915119404855075 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 42.16]
 [11 '65-ST' 10 0.050915119404855075 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 43.19]
 [23 '74-ST' 0 -0.8854803374757575 -0.08790027046100546
  0.9999999999999999 0.20725883476599816 27.08]
 [19 '76-ST' 10 0.050915119404855075 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 27.18]
 [47 '81-ST' 10 1.7156181538592763 3.263919476740732 0.6797551020408164
  0.4902318312060433 43.0]
 [21 '66-ST' 10 0.8312446668053652 0.30184621177175464 0.6797551020408164
  0.617956064947469 40.0]
 [8 '68-ST' 10 0.050915119404855075 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 50.0]
 [43 '1-ST' 10 0.050915119404855075 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 10.0]
 [15 '84-ST' 10 0.050915119404855075 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 43.56]
 [29 '95-ST' 0 -0.833458367649056 1.3931363620234831 0.40163265306122453
  0.4902318312060433 23.5]
 [39 '54-ST' 10 0.050915119404855075 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 19.35]]
Data test:
[[28 '72-ST' 10 0.050915119404855075 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 18.41]
 [40 '60-ST' 10 -1.2496341262626618 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 38.5]
 [27 '63-ST' 10 2.1838158822995837 -0.08790027046100546
  0.6797551020408164 1.0 46.6]
 [44 '88-ST' 10 0.050915119404855075 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 37.0]
 [25 '73-ST' 10 0.050915119404855075 -0.9453425313730773
  0.6797551020408164 0.9111747851002866 41.57]
 [37 '44-ST' 10 0.050915119404855075 -0.08790027046100546
  0.46285714285714286 0.45845272206303733 32.33]
 [13 '90-ST' 10 1.663596184032576 -0.47764675269376516 0.6797551020408164
  0.45558739255014336 39.9]
 [20 '96-ST' 10 0.050915119404855075 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 41.4]
 [5 '89-ST' 10 -0.937502307302458 1.3931363620234831 0.836734693877551
  0.4902318312060433 35.08]
 [26 '80-ST' 10 0.050915119404855075 -0.08790027046100546
  0.6797551020408164 0.4902318312060433 39.45]]
Ocjene train:
```

Zadatak 3 - Klasični algoritam mašinskog učenja

U ovom zadatku, upoznat ćemo se sa primjenom Sklearn biblioteke u klasičnim algoritimma mašinskog učenja. Konkretno, radit će se klasifikacija cvijeta iris na tri različite vrste (prikazane na slici u postavci), i to: • Setosa; • Versicolour; • Virginica;

Algoritam će prvo na osnovu postojećeg skupa podataka, koji sadrži značajke i labele, naučiti kako koja značajka djeluje samu vrstu cvijeta, te će biti u stanju da za novi (neviđeni) podatak pretpostavi tačnu klasu kojoj cvijet pripada.

a) Sam skup podataka dolazi spreman uz Sklearn, te ga je potrebno učitati:

```
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data
y = iris.target

feature_names = iris.feature_names
target_names = iris.target_names

print("Nazivi znacajki:", feature_names)
print("Nazivi labela:", target_names)
print("\nPrvih 5 redova X:\n", X[:5])
```

```
Nazivi znacajki: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal
Nazivi labela: ['setosa' 'versicolor' 'virginica']

Prvih 5 redova X:
 [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```

b) Sljedeći korak jeste dijeljenje skupa podataka na dva dijela - trening i testni skup. Trening skup se koristi kako bi algoritam naučio uzorke ponašanja, dok testni skup predstavlja nove podatke, na osnovu kojih se vrši evaluacija algoritma (to jeste evaluacija naučenog).

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state =
```