Pandas i NumPy biblioteke

Sledeći kod je većinski preuzet od asistenata Anđelke Zečević i Milana Čugurovića sa časova vežbi iz Mašinskog učenja.

In [4]:
```
import pandas as pd
# prethodno instalirati biblioteku, npr. komandom !pip install pandas
```

In [5]:
```
import numpy as np # trebalo bi da je NumPy biblioteka sadržana u okviru prethodnog,
# u suprotnom je instalirati komandom !pip install numpy
```

NumPy

NumPy biblioteka nudi mnoštvo olakšica u radu sa nizovima i matricama (pa i višedimenzionalnim nizovima). Način na koji se sprovode operacije nad njima će nas podsetiti na rad u R-u, pa ne bi trebalo da bude poteškoća u snalaženju.

In [10]:
```
a = np.array([1, 2, 3]) # niz od 3 elementa
M = np.array([[1, 2, 3], [4, 5, 6]], dtype='float32') # matrica od 6 elemenata
# koji se redjaju u matricu redom po vrstama
# dtype() možemo koristiti da promenimo tip elemenata

print('Niz a:')
print(a)
print('shape =', a.shape)
print('size =', a.size)
print('ndim =', a.ndim)
print('dtype =', a.dtype, end='\n\n')

print('Matrica M:')
print(M)
print('shape =', M.shape) # dimenzije matrice
print('size =', M.size)
print('ndim =', M.ndim)
print('dtype =', M.dtype)
```

```
Niz a:
[1 2 3]
shape = (3,)
size = 3
ndim = 1
dtype = int64

Matrica M:
[[1. 2. 3.]
 [4. 5. 6.]]
shape = (2, 3)
size = 6
ndim = 2
dtype = float32
```

In [11]:
```
a1 = np.arange(1, 10, 2) # ekvivalent pozivu u R-u: seq(1,10,2)
a2 = np.linspace(1, 3, 21) # ekvivalent pozivu u R-u: seq(1,3,length.out = 21)
a3 = np.zeros(6) # ekvivalent pozivu u R-u: rep(0,6)
a4 = np.random.uniform(0, 1, 5) # ekvivalent pozivu u R-u: runif(5,0,1)

print('a1 =', a1)
```

```
print('a2 =', a2)
print('a3 =', a3)
print('a4 =', a4)

M1 = np.ones((2, 3)) # ekvivalent pozivu u R-u: matrix(rep(1,6), nrow =2)
M2 = np.eye(3) # jedinična matrica dimenzije 3x3
M3 = np.diag([4, 3, 2, 1]) # dijagonalna matrica sa datim elementima
M4 = np.random.randn(4, 3) # matrica dimenzije 4x3 sa elementima iz N(0,1) raspodele.

print('M1 =\n', M1)
print('M2 =\n', M2)
print('M3 =\n', M3)
print('M4 =\n', M4)
```

```
a1 = [1 3 5 7 9]
a2 = [1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7
 2.8 2.9 3. ]
a3 = [0. 0. 0. 0. 0. 0.]
a4 = [0.25392075 0.75204002 0.2329427  0.34460522 0.23777922]
M1 =
 [[1. 1. 1.]
 [1. 1. 1.]]
M2 =
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
M3 =
 [[4 0 0 0]
 [0 3 0 0]
 [0 0 2 0]
 [0 0 0 1]]
M4 =
 [[-1.81049194 -1.04521863  1.83621478]
 [ 1.25854907  0.62054228 -0.22003193]
 [-0.21100287  0.25734473  0.55611885]
 [ 0.69452773 -0.59026969  0.20072819]]
```

In [12]:
```
M1 = np.array([[1, 3, 5], [7, 9, 11], [13, 15, 17]])
M2 = np.array([[2, 4, 6], [8, 10, 12], [14, 16, 18]])
print('M1 + M2 =\n', M1 + M2) # zbir matrica
print('M1 * M2 =\n', M1 * M2) # koordinatno množenje matrica
print('M1 x M2 =\n', M1.dot(M2)) # matrično množenje (alternativa: np.dot(M1, M2))
print('transpose(M1) =\n', M1.T, '=\n', M1.transpose()) # transponat matrice
print('sin(M1) =\n', np.sin(M1)) # primena funkcije sin() na svaki element matrice M1
```

```
M1 + M2 =
 [[ 3  7 11]
 [15 19 23]
 [27 31 35]]
M1 * M2 =
 [[  2  12  30]
 [ 56  90 132]
 [182 240 306]]
M1 x M2 =
 [[ 96 114 132]
 [240 294 348]
 [384 474 564]]
transpose(M1) =
 [[ 1  7 13]
 [ 3  9 15]
 [ 5 11 17]] =
 [[ 1  7 13]
 [ 3  9 15]
 [ 5 11 17]]
sin(M1) =
 [[ 0.84147098  0.14112001 -0.95892427]
```

```
 [ 0.6569866   0.41211849 -0.99999021]
 [ 0.42016704  0.65028784 -0.96139749]]
```

In [13]:
```
x = np.array([1, 2, 3])
y = np.array([4, 5, 6])
# vertikalno nadovezivanje kako bi se dobila matrica sa vrstama x i y
print(np.vstack((x, y)))
# horiznotalno nadovezivanje (u R-u: c(x,y))
print(np.hstack((x, y)))
# skalarni proizvod
print(np.sum(x * y))
# minimalni element niza x
print(np.min(x))
# maksimalni element niza y
print(y.max())
#elementi niza x koji su veci od 1.5
print(x[x > 1.5])
```

```
[[1 2 3]
 [4 5 6]]
[1 2 3 4 5 6]
32
1
6
[2 3]
```

In [14]:
```
A = np.array([[1, 2, 3], [4, 5, 6]])
B = np.array([[1, 2, 3], [4, 6, 6]])

# jednakost dve matrice
print(np.array_equal(A, B))

# jednakost po koordinatama
print(A == B)

# matrica 3x2 od matrice 2x3
print(A.reshape(3, 2))

# suma elemenata koji su jednaki 6
print(np.sum(B[B == 6]))
```

```
False
[[ True  True  True]
 [ True False  True]]
[[1 2]
 [3 4]
 [5 6]]
12
```

In [16]:
```
M = np.array([[1, 2, 3], [4, 5, 6]])
x = np.array([10, 20, 30])
print(2*M) # množenje matrice skalarom
print(x - 3) # koordinatno oduzimanje trojke od elemenata niza
print(M + x) # sabiranje matrice i niza se vrši tako što se niz
# ciklično proširi do matrice odgovarajuće dimenzije
```

```
[[ 2  4  6]
 [ 8 10 12]]
[ 7 17 27]
[[11 22 33]
 [14 25 36]]
```

In [19]:
```
M = np.arange(9).reshape(3, 3) # matrica
# prvih 9 brojeva (počevši od NULE) dimenzije 3x3
print('M =\n', M)
average = np.average(M) # prosek elemenata matrice
```

```
print(average)
print(np.sum(M[M > average])) # suma elemenata matrice većih od proseka
```

```
M =
 [[0 1 2]
 [3 4 5]
 [6 7 8]]
4.0
26
```

In [23]:
```
M = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
# pristup elementima matrice:
print(M)
print('Pojedinačni element:', M[1,2]) # (voditi računa o tome da numeracija počinje od 0
print('Elementi na pozicijama (2,3) i (2,4):', M[(1,1), (2,3)])
print('Druga vrsta:', M[1])
print('Prva kolona:', M[:,0])
print('Svaki drugi element poslednje kolone:', M[::2, -1])
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
Pojedinacni element: 7
Elementi na pozicijama (1,2) i (1,3): [7 8]
Druga vrsta: [5 6 7 8]
Prva kolona: [1 5 9]
Svaki drugi element poslednje kolone: [ 4 12]
```

In [26]:
```
M = np.arange(25).reshape(5, 5) + 1 # dodajemo 1 ako želimo da se elementi
# redaju od 1 do 25
print(M)
print('Vrednosti od 3. do 4. elementa u nultom redu:', M[0,3:5])
print('Kvadratna podmatrica reda 2 u donjem desnom uglu:\n', M[-2:,-2:])
print('Elementi svakog drugog reda počev od 2. i elementi svake druge kolone:\n',
      M[2::2,::2])
```

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]
 [21 22 23 24 25]]
Vrednosti od 3. do 4. elementa u nultom redu: [4 5]
Kvadratna podmatrica reda 2 u donjem desnom uglu:
 [[19 20]
 [24 25]]
Elementi svakog drugog reda počev od 2. i elementi svake druge kolone:
 [[11 13 15]
 [21 23 25]]
```

Pandas

Pandas biblioteku je preporučljivo koristiti u radu sa podacima. Bavićemo se dvema strukturama - series i data frame.

Serije

Serije odgovaraju jednodimenzionim nizovima.

In [28]:
```
s = pd.Series([2, 3, 4, 53, -3, 12]) # zadavanje serije
print(s) # prva kolona predstavlja indekse vektora, a druga vrednosti
print('Elementi: {}'.format(s.values))
print('Indeksi: {}'.format(s.index))
```

```
0     2
1     3
2     4
3    53
4    -3
5    12
dtype: int64
Elementi: [ 2  3  4 53 -3 12]
Indeksi: RangeIndex(start=0, stop=6, step=1)
```

In [29]:
```
s = pd.Series([2, 3, 5, 7, 12], index = ['a','b','c', 'd', 'e']) # indeksi
# se mogu i eksplicitno zadati
print(s)
```

```
a     2
b     3
c     5
d     7
e    12
dtype: int64
```

In [30]:
```
s = pd.Series([1, 2, np.nan, np.nan, 45, -4, np.nan, -9]) # serija sa NA vrednostima
print(s)
```

```
0     1.0
1     2.0
2     NaN
3     NaN
4    45.0
5    -4.0
6     NaN
7    -9.0
dtype: float64
```

In [31]:
```
s = pd.Series(5, index=[0, 1, 2, 3]) # serija od 4 petice
print(s)
```

```
0    5
1    5
2    5
3    5
dtype: int64
```

In [33]:
```
data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd.Series(data) # još jedan način zadavanja serije
print(s)
```

```
a    0.0
b    1.0
c    2.0
dtype: float64
```

In [35]:
```
data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd.Series(data,index=['b','c','a']) # prethodna serija sa promenjenim indeksima
print(s)
```

```
b    1.0
c    2.0
a    0.0
dtype: float64
```

In [40]:
```
print('Element sa indeksom "b":', s['b']) # pristup elementima serije
print('Element sa indeksom "c":', s['c'])
```

```
Element sa indeksom "b": 2
Element sa indeksom "c": 3
```

In [41]:
```
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
print(s[['a','c','d']]) # pristup većem broju elemenata serije
```

```
a    1
c    3
d    4
dtype: int64
```

In [42]:
```
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
print(s[0]) # pristup elementima serije preko rednih brojeva
```

```
1
```

In [43]:
```
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
print(s[3:]) # pristup većem broju elemenata preko rednih brojeva
```

```
d    4
e    5
dtype: int64
```

Data frame

Strukture za tabelarni prikaz podataka koje se sastoje iz serija.

In [44]:
```
df = pd.DataFrame({ # zadavanje data frame-a
    'ime': ['Ana', 'Laza', 'Maja', 'Goran'],
    'godine': [20, 22, 30, 27],
    'grad': ['BG', 'NS', 'NS', 'NI']
})
print(df)
```

```
     ime  godine grad
0    Ana      20   BG
1   Laza      22   NS
2   Maja      30   NS
3  Goran      27   NI
```

In [45]:
```
df # lepši ispis
```

Out[45]:

	ime	godine	grad
0	Ana	20	BG
1	Laza	22	NS
2	Maja	30	NS
3	Goran	27	NI

In [47]:
```
df.head(2) # prikaz prve dve vrste data frame-a
```

Out[47]:

	ime	godine	grad
0	Ana	20	BG
1	Laza	22	NS

In [54]:
```
print('Imena:')
df['ime'] # kolona "ime"
```

Out[54]:
```
Imena:
0      Ana
1     Laza
2     Maja
```

```
3    Goran
Name: ime, dtype: object
```

In [55]:
```
df[['ime', 'godine']] # kolone "ime" i "godine"
```

Out[55]:

	ime	godine
0	Ana	20
1	Laza	22
2	Maja	30
3	Goran	27

In [56]:
```
df.loc[2] # druga vrsta
```

Out[56]:
```
ime        Maja
godine       30
grad         NS
Name: 2, dtype: object
```

In [57]:
```
df.loc[1:2] # više vrsta
```

Out[57]:

	ime	godine	grad
1	Laza	22	NS
2	Maja	30	NS

In [58]:
```
df.loc[2]['ime'] # vrednost jednog polja
```

Out[58]:
```
'Maja'
```

In [63]:
```
df = pd.DataFrame(np.random.randn(6, 4), index=datumi, columns=list("ABCD"))
df
```

Out[63]:

	A	B	C	D
2013-01-01	1.016009	-0.491092	0.389737	0.470957
2013-01-02	-1.403572	1.318029	-0.131152	0.580054
2013-01-03	1.994984	1.564252	1.660782	-0.236695
2013-01-04	0.440024	-0.843418	1.619495	0.742567
2013-01-05	-0.581777	0.761111	0.830550	-0.117320
2013-01-06	1.444685	0.320624	-0.118342	0.206957

In [66]:
```
df.describe() # ispis nekih poznatih statistika
```

Out[66]:

	A	B	C	D
count	6.000000	6.000000	6.000000	6.000000
mean	0.485059	0.438251	0.708512	0.274420
std	1.279825	0.966055	0.805226	0.392543
min	-1.403572	-0.843418	-0.131152	-0.236695
25%	-0.326327	-0.288163	0.008678	-0.036251
50%	0.728017	0.540867	0.610143	0.338957

75%	1.337516	1.178799	1.422259	0.552780
max	1.994984	1.564252	1.660782	0.742567

In [67]:
```
df.sort_values(by="B") # rastuće sortiranje vrsta po koloni "B"
```

Out[67]:

	A	B	C	D
2013-01-04	0.440024	-0.843418	1.619495	0.742567
2013-01-01	1.016009	-0.491092	0.389737	0.470957
2013-01-06	1.444685	0.320624	-0.118342	0.206957
2013-01-05	-0.581777	0.761111	0.830550	-0.117320
2013-01-02	-1.403572	1.318029	-0.131152	0.580054
2013-01-03	1.994984	1.564252	1.660782	-0.236695

In [69]:
```
df[df["A"] > 0] # vrste sa pozitivnim elementima prve kolone
```

Out[69]:

	A	B	C	D
2013-01-01	1.016009	-0.491092	0.389737	0.470957
2013-01-03	1.994984	1.564252	1.660782	-0.236695
2013-01-04	0.440024	-0.843418	1.619495	0.742567
2013-01-06	1.444685	0.320624	-0.118342	0.206957

In [70]:
```
df > 0 # koordinatno poređenje
```

Out[70]:

	A	B	C	D
2013-01-01	True	False	True	True
2013-01-02	False	True	False	True
2013-01-03	True	True	True	False
2013-01-04	True	False	True	True
2013-01-05	False	True	True	False
2013-01-06	True	True	False	True

In [72]:
```
df["E"] = ["one", "two", "three", "four", "five", "six"] # dodavanje kolone
df
```

Out[72]:

	A	B	C	D	E
2013-01-01	1.016009	-0.491092	0.389737	0.470957	one
2013-01-02	-1.403572	1.318029	-0.131152	0.580054	two
2013-01-03	1.994984	1.564252	1.660782	-0.236695	three
2013-01-04	0.440024	-0.843418	1.619495	0.742567	four
2013-01-05	-0.581777	0.761111	0.830550	-0.117320	five
2013-01-06	1.444685	0.320624	-0.118342	0.206957	six