VISOKA ŠKOLA ELEKTROTEHNIKE I RAČUNARSTVA STRUKOVNIH STUDIJA

BEOGRAD

PROJEKTNI ZADATAK

DRUGI DEO

Predmet: Veštačka inteligencija

Profesor: Emilija Kisić

Datum: *6.6.2021*

Student:

Aleksa Aleksić

RT-9/19

Grupa:

Danijel Stokić RT-74/19

Zadatak 1

(Klasterizacija Kmeans)

Klasterizacija nam služi da grupišemo podatke u neke grupe. Može da se naglasi da je klasterizacija jedan od oblika nenadledanog učenja, takodje, ispravno rešenje nije tačno definisano. Nemamo obučavajući skup kao kod nadgledanog učenja već algoritam sam uči kako da podeli podatke u grupe (klastere).

Za početak, što se tiče našeg zadatka, prikazaćemo podatke sa kojima raspolažemo.

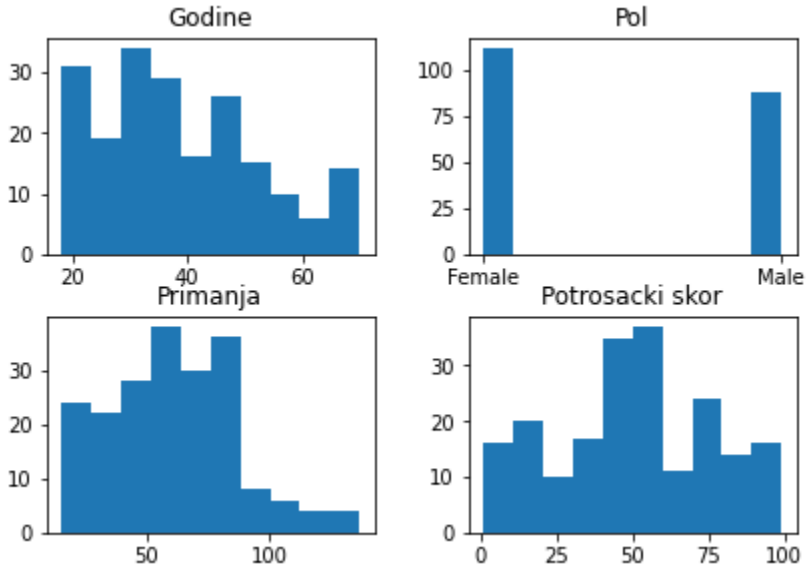
	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
114	115	Female	18	65	48
91	92	Male	18	59	41
65	66	Male	18	48	59
33	34	Male	18	33	92
0	1	Male	19	15	39
..	...	...	...	...	...
90	91	Female	68	59	55
108	109	Male	68	63	43
57	58	Male	69	44	46
70	71	Male	70	49	55
60	61	Male	70	46	56

Slika 1. Podaci iz dataseta tržni centar

Naglasićemo odmah da podatak CostumerID nije od značaja za nas, tako da ćemo ga odmah ukloniti radi lakšeg pozicioniranja po skupu.

Pošto nam je krajnji cilj da izaberemo klaster (grupu) sa mlađom populacijom, odmah smo soritirali podatke po godinama tako da bi mogli „grubo“ da prikažemo grupe, u cilju lakšeg prikaza na grafiku.

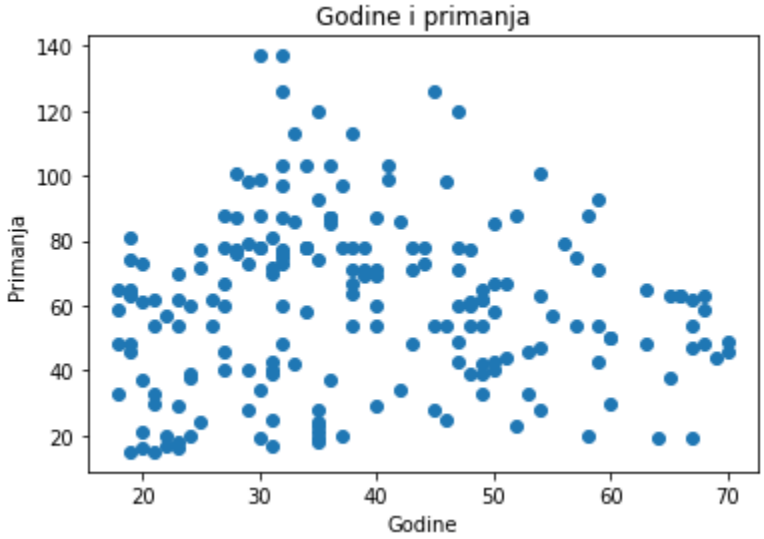
Što se tiče analize skupa izlistaćemo nekoliko histograma za početak.



Slika 2. Histogrami atributa

Ovde možemo da vidimo koliko brojno odnos između atributa. Na primer možemo da primetimo da imamo najviše ljudi sa potrošačkim sorom oko 50, da imamo više žena od muškaraca i da imamo više podataka o ljudima ispod 40 godina nego preko.

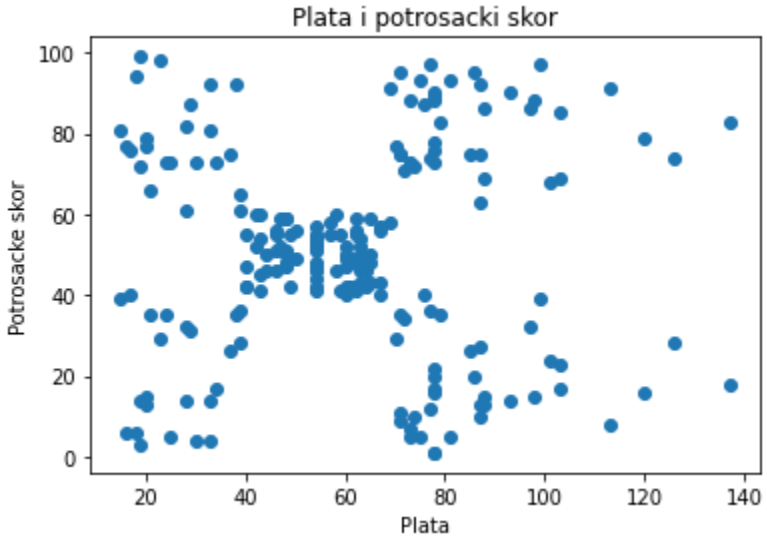
Prikazaćemo i nekoliko rasutih dijagrama da bi mogli da primetimo neki odnos između određenih atributa.



Slika 3. Odnos godina i primanja

Vidimo na slici 3 da i ne možemo da primetimo neku posebnu vezu. Ako je čovek stariji ne znači da ima veću platu ili obrnuto. Jedino što možemo da zaključimo iz prethodnog dijagrama jeste da su maksimumi u platama (preko 100) raspoređeni po ljudima između 30 i 50 godina. Pošto nama treba mlađi deo populacije ova činjenica nam nije od posebnog značaja.

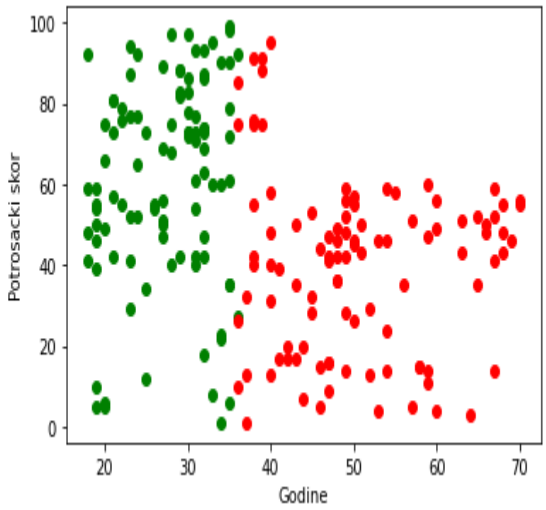
Sada čemo proveriti predpostavku da ako neko ima visoku platu takođe će imati i visok potrošačni skor.



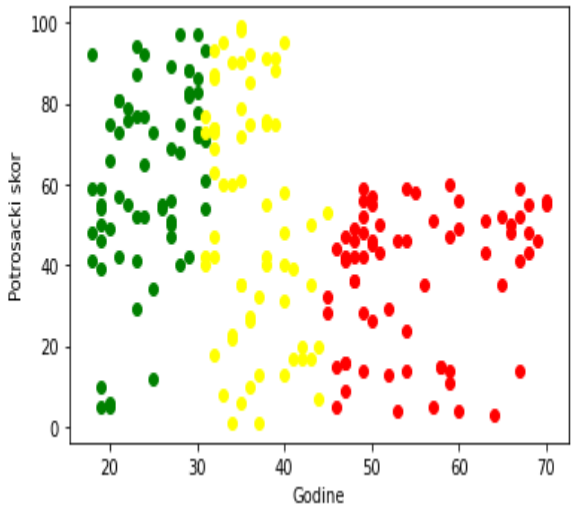
Slika 4. Odnos plate i potrošačnog skora

Vidimo da i ova predpostavka pada u vodu. Podaci se veoma razbacani, jedino što možemo da vidimo jeste da ljudi sa platom između 40 i 70 imaju potrošački skor između 40 i 60. Što se tiče ostalih podataka, neko sa platom manjom od 40 može da ima veći potrošačni skor od nekoga kome je plata preko 100 i obrnuto.

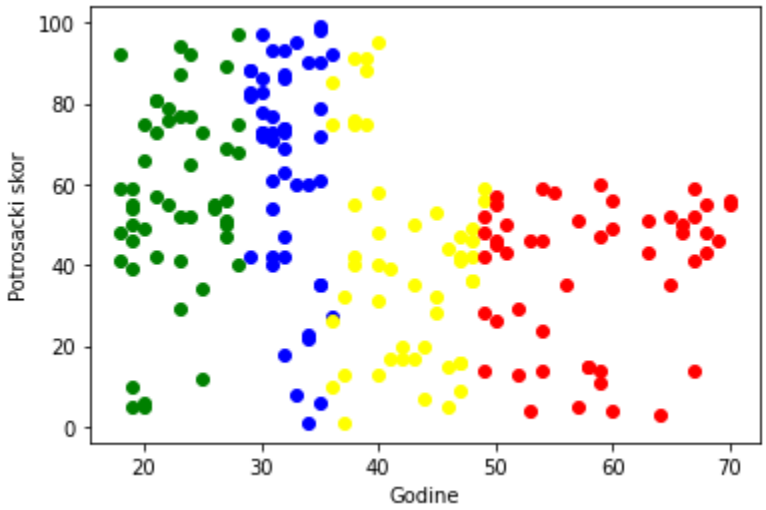
Odlučujemo da koristimo atribute godine i potrošačni skor koji nam bar mogu dati sliku o tome gde nam se nalaze grupe mlađih godina. Sortirali smo skup po godinama tako da će nam boje izdeliti skupove (grubo) po tom atributu. Takođe, izlistaćemo podatke podeljene na dve, tri i četiri grupe, da bi mogli da nađemo najbolji klaster koji bi koristili.



Slika 5. Podaci izdeljeni grubo u dve klase



Slika 6. Podaci izdeljeni grubo u tri klase

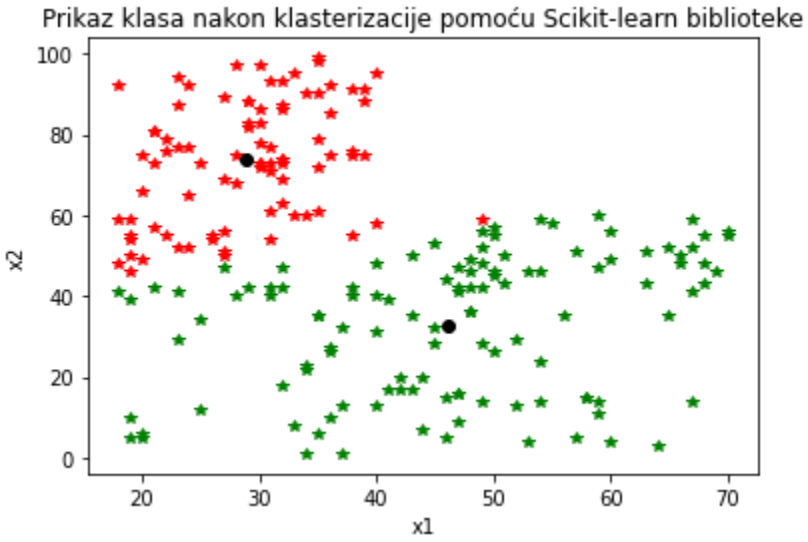


Slika 7 Podaci izdeljeni grubo u ćetiri klase

Prethodno navedene slike ne trebamo da gledamo kao klastere već samo kao obojene delove rasutog dijagrama da bi lakše zaključili koji deo nam je potreban za ciljnu grupu. Konkretno zeleni deo bi bio mlađi deo populacije.

Možemo da primetimo i mali uspon na grafu što se tiče dela na X osi ispod 40. Po tome vidimo da grupa ljudi mlađa od 40 godina troši više od starijih što nismo mogli da zaključimo is prethodnih grafova. Baš taj deo nam je od velikog značaja i pokušaćemo da izaberemo klaster koji sadrži najviše podataka tog dela nakon izvršenja algoritma.

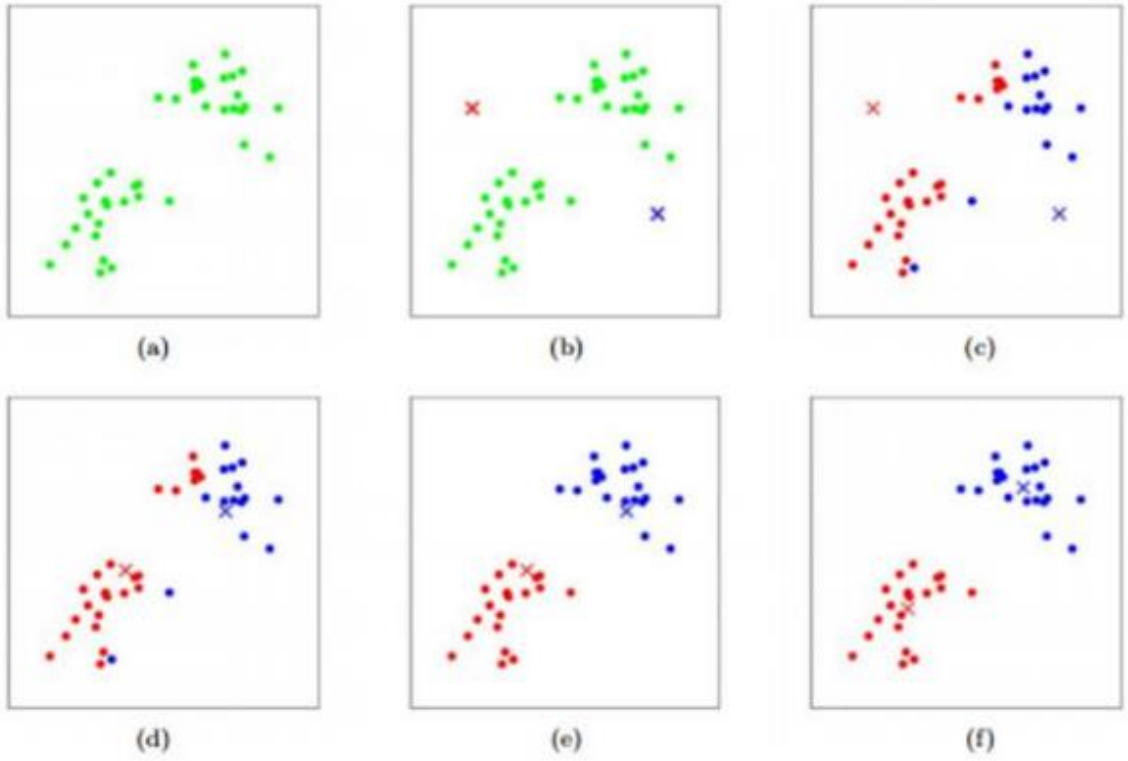
Što se tiče izbora klastera uradićemo klasterizaciju sa 2, 3 i 4 klastera i odabrati klaster koji nam najbolje govori da su podaci u njemu mladji deo populacije i takodje ima visok potrošačni skor.



Slika 8 Klasterizacija sa dva centroida

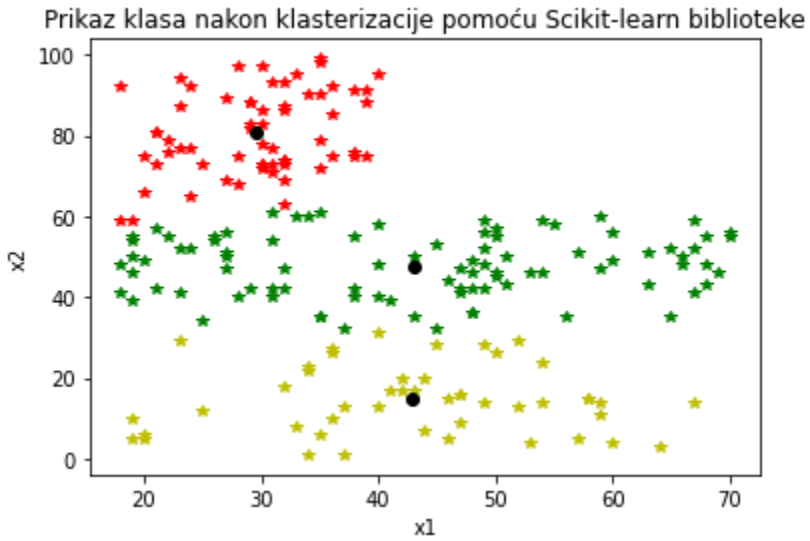
Odmah na osmoj slici možemo da uočimo gore pomenuti klaster koji je obojen crvenom bojom. To su podaci od kojih imamo dosta ljudi mlađe populacije (gledaćemo to kao manje od 30), ali takođe i visok potrošačni skor.

- *K-means algoritam funkcioniše kao zarazni virus (mogli bi se reći i zombi apokalipsa). Odabran broj centroida, ili prvenstveno zaraženih kreće se ka grupama koji za sada nisu zaraženi (klasifikovani). U nekom odredjenom krugu zaražava podatke koji su oko njega (boji ih, klasterizuje) i kreće se tamo gde su najviše zaraženi (najgušći podaci). Kada dodje do centra, i svi u odredjenom krugu budu iste boje, algoritam se završava. U zavisnosti koliko odaberemo klastera toliko ćemo imati i centroida.*



Slika 9 Ilustracija K-means algoritma kroz iteracije

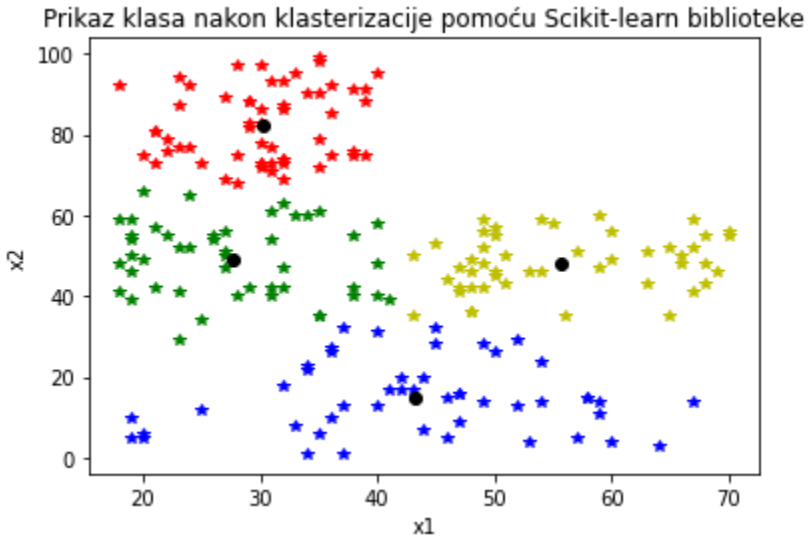
Nastavićemo sa daljom klasterizacijom, pokušaćemo da izdelimo naše podatke na 3 klastera.



Slika 10 Klasterizacija sa tri centroida

Ciljani klaster (crveni) ostaje isti, međutim imamo zeleni koji pokriva sve ljude koji imaju potrošački skor između 40 i 60 ali imamo problem što se tu nalaze ljudi svih uzrasta.

Uradjićemo konačno i klasterizaciju sa 4 klastera.



Slika 11 Klasterizacija sa četiri centroida

Ponovo, crveni klaster ostaje isti, međutim prethodni zeleni klaster izdvojen je na dva dela, mlađe i starije. Sada nam i zeleni klaster može biti od značaja.

Ako se radi o mađem delu populacije mogli bi da se odlučimo za jedan od dva klastera (crveni ili zeleni). Crveni nam daje prednost jer je potrošačni skor dosta visok tako da je najbolje da koristimo taj klaster za plasiranje proizvoda koji je popularan kod mlađeg dela populacije.

Zadatak 2

(Neuronska mreža)

Cilj ovog zadatke jeste da obučimo neuronsku mrežu tako da u odnosi na parametre može da predpostavi da li je odredjena osoba imala šlog ili nije. Model je veoma sličan poznatom problemu titanik skupa pa ćemo se držati metodologije korišćene u rešavanju istog.

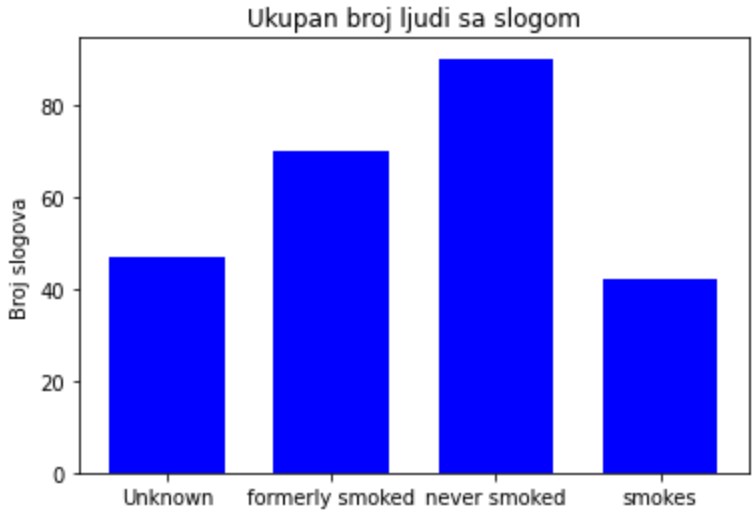
Za početak možemo da prikažemo podatke sa kojima raspolažemo.

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
5	56669	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
6	53882	Male	74.0	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
7	10434	Female	69.0	0	0	No	Private	Urban	94.39	22.8	never smoked	1
8	27419	Female	59.0	0	0	Yes	Private	Rural	76.15	NaN	Unknown	1
9	60491	Female	78.0	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1

Slika 12 Podaci iz skupa za predikciju šloga

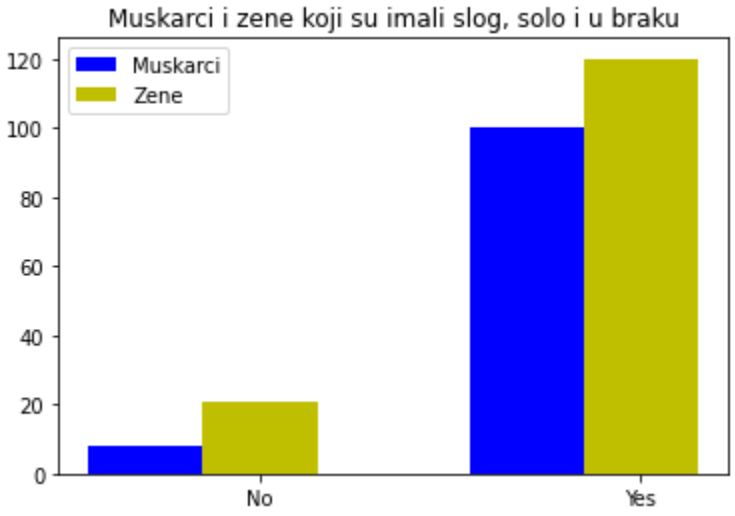
Odmah vidimo da ćemo imati više problema sa inicjalnim skupom. Na primer, imamo Nan vrednosti u koloni bml a takodje imamo i dosta vrednosti koje su tipa string dok nama trebaju numerični tipovi za neuronsku mrežu.

Ali, za početak, analiziraćemo nekoliko interesantnih podataka.



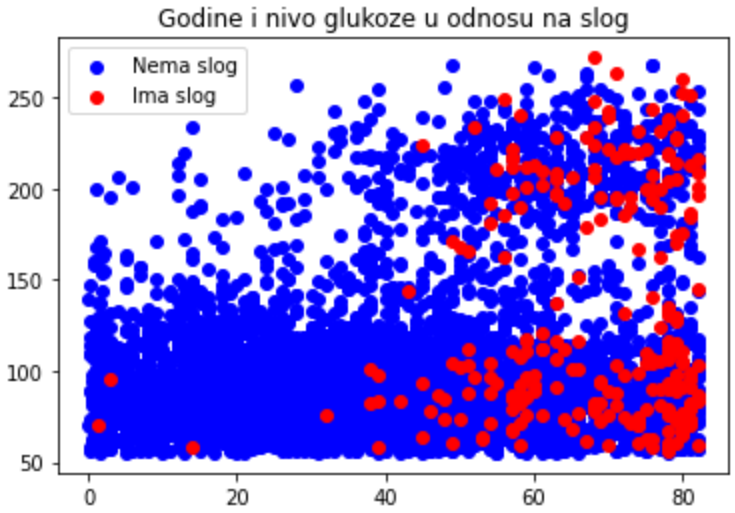
Slika 13 Podaci iz skupa za predikciju šloga

Prikazali smo pušački status i broj ljudi sa šlogom. Vidimo da je najveći broj ljudi sa šlogom dodeljen nepušačima. Takodje, imamo kolonu Unknown. Za ove podatke nije poznato da li su pušili ili nisu. Mogli bi da izdelimo kolonu proporcionalno na ostale ili da je uklonimo, međutim, smatram da je bitan faktor pa ćemo je ostaviti ovakvom kakva jeste.



Slika 14 Muškarci i žene koji su imali šlog i koji nisu u odnosu na bračni status

Iz prethodne slike možemo jasno da vidimo da je mnogo veči broj ljudi koji su u braku imalo šlog od ljudi koji nisu u braku. Inicijalno kolonu koju smo mogli da uklonimo, zbog ove slike, proglasićemo kao značajnu.



Slika 15 Godine i novi glukoze u odnosu na šlog

Prikazan je rasuti dijagram odnosa godina i nivoa glukoze. Jasno možemo da vidimo da što je osoba starija, nivo glukoze raste, međutim, naravno, imamo izuzetaka. Crvenom bojom su obojeni ljudi koji su imali šlog a plavom ljudi koji nisu. Vidimo da je crveni deo dosta zbijen uz kraj X ose gde nam se nalazi stariji deo populacije što je i logično. Godine i novo glukoze su nam veoma bitan faktor u obučavanju ove mreže.

Počećemo sa sređivanjem našeg skupa tako da možemo da ga koristimo za obučavanje mreže. Za početak, proverićemo gore pomenute Nan vrednosti.

	Total	%
bmi	201	3.9
stroke	0	0.0
smoking_status	0	0.0
avg_glucose_level	0	0.0
Residence_type	0	0.0
work_type	0	0.0
ever_married	0	0.0
heart_disease	0	0.0
hypertension	0	0.0
age	0	0.0
gender	0	0.0
id	0	0.0

Slika 16 Vrednosti koje nedostaju u skupu u procenat u odnosu na ukupan broj vrednosti

Vidimo da jedina kolona u kojoj nedostaju vrednosti jeste bmi. Imamo 201 nedostajuću vrednost što je samo 3.9% celog skupa. Iz tog razloga, nećemo ukloniti kolonu već ćemo dopuniti vrednosti sa nasumičnim brojem između vrednosti dobijene razlikom srednje vrednosti i standardne devijacije i zbira standardne devijacija i srednje vrednosti.

```
random_bmi = np.random.randint(mean -std, mean+std, size = is_null)
```

Slika 17. Zamena nedostajućih vrednosti

Kao što smo već rekli, da smo naišli na veliki broj nedostajućih podataka više ova tehnika ne bi bila validna jer bi ovom metodom dopunili podatke i više skup ne bi bio validan (prirodan). Tada bi morali da uklonimo kompletnu kolonu.

Pretvorićemo vrednosti koje su opisane stringom u numeričke i takodje float vrednosti čemo pretvoriti u int.

```
for i in data:
  data['age'] = data['age'].astype(int)
  data['avg_glucose_level'] = data['avg_glucose_level'].astype(int)
  data['bmi'] = data['bmi'].astype(int)
  data['ever_married'] = data['ever_married'].replace('Yes', 1).replace('No', 0)
  data['gender'] = data['gender'].replace('Male', 0).replace('Female', 1).replace('Other',2)
  data['work_type'] = data['work_type'].replace('Private', 1).replace('Self-employed', 0).replace('Govt_job',2).replace('children', 3).replace('Never_worked', 4)
  data['Residence_type'] = data['Residence_type'].replace('Urban', 1).replace('Rural', 0)
  data['smoking_status'] = data['smoking_status'].replace('Unknown', 0).replace('never smoked', 1).replace('formerly smoked', 2).replace('smokes', 3)
```

Slika 18.Popravljanje tipova vrednosti

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	0	67	0	1	1	1	1	228	36	2	1
1	51676	1	61	0	0	1	0	0	202	28	1	1
2	31112	0	80	0	1	1	1	0	105	32	1	1
3	60182	1	49	0	0	1	1	1	171	34	3	1
4	1665	1	79	1	0	1	0	0	174	24	1	1
5	56669	0	81	0	0	1	1	1	186	29	2	1
6	53882	0	74	1	1	1	1	0	70	27	1	1
7	10434	1	69	0	0	0	1	1	94	22	1	1
8	27419	1	59	0	0	1	1	0	76	34	0	1
9	60491	1	78	0	0	1	1	1	58	24	0	1

Slika 19.Skup nakon popravljanja vrednosti

Da bi još bolje pripremili skup za obuku, skaliraćemo odredjene opsege godina na manje brojeve. Isto ćemo uraditi i za kolonu avg_glucose_level i bmi.

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	0	5	0	1	1	1	1	5	4	2	1
1	51676	1	5	0	0	1	0	0	4	2	1	1
2	31112	0	6	0	1	1	1	0	2	3	1	1
3	60182	1	3	0	0	1	1	1	3	4	3	1
4	1665	1	6	1	0	1	0	0	3	2	1	1
5	56669	0	6	0	0	1	1	1	4	3	2	1
6	53882	0	6	1	1	1	1	0	1	2	1	1
7	10434	1	5	0	0	0	1	1	1	1	1	1
8	27419	1	4	0	0	1	1	0	1	4	0	1
9	60491	1	6	0	0	1	1	1	0	2	0	1
10	12109	1	6	1	0	1	1	0	1	3	1	1
11	12095	1	5	0	1	1	2	0	2	4	3	1
12	12175	1	4	0	0	1	1	1	2	2	3	1
13	8213	0	6	0	1	1	1	1	4	1	0	1
14	5317	1	6	0	1	1	1	1	4	2	1	1
15	58202	1	3	1	0	1	0	0	3	3	1	1
16	56112	0	5	0	1	1	1	1	4	4	3	1
17	34120	0	6	1	0	1	1	1	5	2	3	1
18	27458	1	4	0	0	0	1	1	1	4	1	1
19	25226	0	4	0	1	0	2	1	4	3	0	1

Slika 20.Skup nakon skaliranja vrednostii

Proverićemo i da li je naše skaliranje validno

```
Skalirane godine
0    1025
4     823
3     739
6     710
2     674
5     594
1     545
Name: age, dtype: int64
Skaliran prosecan nivo glukoze
1    2919
2    1111
4     381
0     270
3     232
5     197
Name: avg_glucose_level, dtype: int64
Skaliran indeks telesne mase
2    1733
3    1167
4     756
1     701
0     404
5     284
6      61
7       4
Name: bmi, dtype: int64
```

Slika 21.Provera validnosti skaliranja

Na slici 21 prikazan je broj ljudi koji pripadaju novim klasama (opsezima) koje smo izabrali za skaliranje. Primetićemo da se ni u jednoj grupi ne nalazi 80 procenata skupa tako da je sklairanje regularno.

Nakon što smo sredili podatke možemo da odlučimo koji atributi su nam od značaja za obuku. Odabrali smo sve atribute osim id koji nam nije od značaja.

Y kolonu koristimo kao target vrednost dok je X skup svih atributa sa njihovim respektivnim vrednostima.

Skupove delimo na obučavajući koji je 80% celog skupa, dok skup za testiranje opet delimo na dva dela. Jedan veliki deo 99.3% koristimo za testiranje modela, dok nam je preostalih 0.7% za predikciju (samo 8 instanci)

```
from sklearn.model_selection import train_test_split
X_obucavajuci, X_testirajuci, Y_obucavajuci, Y_testirajuci= train_test_split(X, Y, test_size=0.20, random_state=1)

broj_featuresa = X_obucavajuci.shape[1]
broj_featuresa

X_testirajuci1, X_testMali, Y_testirajuci1, Y_testMali = train_test_split(X_testirajuci, Y_testirajuci, test_size=0.007, random_state=1)

X_testirajuci = X_testirajuci1
Y_testirajuci = Y_testirajuci1
print('Obucavajuci X i : ',X_obucavajuci.shape,Y_obucavajuci.shape)
print('Testirajuci X i Y: ',X_testirajuci.shape,Y_testirajuci.shape)
print('Mali deo za predikciju X i Y', X_testMali.shape, Y_testMali.shape)
```

```
Obucavajuci X i :  (4088, 10) (4088,)
Testirajuci X i Y:  (1014, 10) (1014,)
Mali deo za predikciju X i Y (8, 10) (8,)
```

Slika 22. Deljenje skupova i njihove dimenzije nakon podele

Sada konačno prelazi i na deo sa neuronskom mrežom. Za početak trebamo da shvatimo arhitekturu mreže. Konkretno za naš problem biramo ulazni sloj sa 10 naurona koji će primiti svih 10 atributa koje smo izabrali kao atribute od značaja. Izlazni sloj sadrži samo jedan neuron koji može dati vrednost 0 ili 1 (nije imao šlog ili jeste).

Najteži deo kod neuronske mreže jeste odabir broja skrivenih slojeva (skrivenim slojem se smatra svaki sloj koji se nalazi između ulaznog i izlaznog) i takodje neurona u samom sloju. Neko pravilo je da se krene od malog broja, proverava tačnost i zatim povećava. Ulazni sloj nam dale bolju obradu podataka, imamo više sinapsi (težinskih veza) koje čuvaju znanje. Tako će od ulaza, neki podatak biti poslat kroz skriveni sloj pa sve do izlaz.

Epohe ili iteracije obučavanja trebamo da biramo takođe testiranjem. Po pravilu, obučavanje ćemo prekinuti u trenutnu kada greška na o testirajućem skupu počne da raste. Ova vrednost nam takođe dosta govori o tome koliko će dugo obuka trajati.

U našem zadatku odlučujemo se za samo jedan skriveni sloj od 5 neurona jer je odnos vremena obučavanje i greske odličan. Što se tiče epoha uzeli smo 50.

```
model.add(Dense(10, activation='relu', kernel_initializer= 'uniform', input_shape=(broj_featuresa,)))
model.add(Dense(5, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Slika 23. Arhitektura neuronske mreže

Testiraćemo model da bi proverili tačnost:

```
Tacnost modela je: 0.94181
```

Slika 24. Tačnost modela

Proverićemo i tačnost predikcije nakon obučavanja:

```
1/1 - 0s - loss: 0.4669 - accuracy: 0.8750
Tacnost predikcije je: 0.87500
```

Slika 25. Tačnost predikcije

Rezultati deluju odlično, model ima tačnost od 94 procenta a pogodili smo 7 od 8 predikcija.

Problem se javlja kada pogledamo skup podataka i zaključimo da su nam samo 250 iteracija ljudi koji su imali šlog a svi ostali nisu (oko 4800). Da bi sa sigurnošću znali da li model radi doslednu predikciju morali bi da uvedemo još podataka u skup sa ljudima koji su imali šlog tako da neuronska mreža ima evidenciju o atributima koji bi mogli da utičnu na dalje odlučivanje.

Možemo još i da pokušamo sa dosta manjim skupom, prvih 600 iteracija od kojih nam je 250 imalo šlog a ostatak nije. Arhitekturu mreže ostavljamo nepromenjenu, a kada testiramo više puta zaključujemo da povećanjem iteracija preko 50 greška počinje da raste, i ostavljamo i broj iteracija na 50

```
X = X11
Y = Y11

from sklearn.model_selection import train_test_split
X_obucavajuci, X_testirajuci, Y_obucavajuci, Y_testirajuci= train_test_split(X, Y, test_size=0.20, random_state=1)

broj_featuresa = X_obucavajuci.shape[1]
broj_featuresa

X_testirajuci1, X_testMali, Y_testirajuci1, Y_testMali = train_test_split(X_testirajuci, Y_testirajuci, test_size=0.007, random_state=1)

X_testirajuci = X_testirajuci1
Y_testirajuci = Y_testirajuci1
print('Obucavajuci X i : ',X_obucavajuci.shape,Y_obucavajuci.shape)
print('Testirajuci X i Y: ',X_testirajuci.shape,Y_testirajuci.shape)
print('Mali deo za predikciju X i Y', X_testMali.shape, Y_testMali.shape)
```

```
Obucavajuci X i :  (480, 10) (480,)
Testirajuci X i Y:  (119, 10) (119,)
Mali deo za predikciju X i Y (1, 10) (1,)
```

Slika 26. Podela za manji deo skupa

```
Tacnost modela je: 0.73109
```

Slika 27. Tačnost na manjem testirajućem skupu

Dobijena tačnost nije zavidna. Trebali bi da imamo dosta više podataka za ljude koji su imali šlog tako da bi mreža mogla da rapsolaže sa više različitih podataka i samim time povećali bi tačnost predikcije.

Kod korišćen za izradu zadataka

- *Kompletan kod je pisan u Google Colab-u zbog toga je blok znakova '#' postavljen tamo gde je poseban blok koda.*

################################ ZADATAK 1 ################################

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt


dataset = pd.read_csv('/content/Trzni_centar.csv')

dataset = dataset.drop('CustomerID', axis=1)

dataset = dataset.sort_values(by = 'Age')

print(dataset)

#Izbacujemo CostumerID jer nije podatak od znacaja
```

####################################

```
fig, axs = plt.subplots(2, 2)

fig.tight_layout(pad=1.0)

axs[0, 0].hist(dataset['Age'])

axs[0, 0].set_title('Godine')
```

```
axs[0, 1].hist(dataset['Gender'])
axs[0, 1].set_title('Pol')
axs[1, 0].hist(dataset['Annual Income (k$)'])
axs[1, 0].set_title('Primanja')
axs[1, 1].hist(dataset['Spending Score (1-100)'])
axs[1, 1].set_title('Potrosacki skor')

plt.show()
```

```
#####################################
```

```
plt.scatter(dataset['Age'], dataset['Annual Income (k$)'])
plt.title('Godine i primanja')
plt.xlabel('Godine')
plt.ylabel('Primanja')

plt.show()
```

```
#####################################
```

```
plt.scatter(dataset['Annual Income (k$)'], dataset['Spending Score (1-100)'])
plt.title('Plata i potrosacki skor')
plt.xlabel('Plata')
plt.ylabel('Potrosacke skor')

plt.show()
```

```
#####################################
```

```
dataset = dataset.to_numpy()

X1 = dataset[:,1]
X2 = dataset[:,3]

x11 = X1[0:100]
x12 = X2[0:100]
x21 = X1[100:]
x22 = X2[100:]
```

```
x11=x11.reshape(-1,1)
x12=x12.reshape(-1,1)
x21=x21.reshape(-1,1)
x22=x22.reshape(-1,1)

X1 = X1.reshape(-1,1)
X2 = X2.reshape(-1,1)



plt.scatter(x11[:50], x12[:50], color = 'green')
plt.scatter(x11[50:100], x12[50:100], color = 'blue')
plt.scatter(x21[:50], x22[:50], color = 'yellow')
plt.scatter(x21[50:100], x22[50:100], color = 'red')

#plt.scatter(X1[:66], X2[:66], color = 'green')
#plt.scatter(X1[66:132], X2[66:132], color = 'yellow')
#plt.scatter(X1[132:], X2[132:], color='red')



#plt.scatter(x11, x12, color = 'green')
#plt.scatter(x21, x22, color = 'red')
```

```
plt.xlabel('Godine')
plt.ylabel('Potrosacki skor')
plt.show()

#####################################

plt.scatter(X1[:66], X2[:66], color = 'green')
plt.scatter(X1[66:132], X2[66:132], color = 'yellow')
plt.scatter(X1[132:], X2[132:], color='red')

plt.xlabel('Godine')
plt.ylabel('Potrosacki skor')

plt.show()

#####################################

plt.scatter(x11, x12, color = 'green')
plt.scatter(x21, x22, color = 'red')
```

```
plt.xlabel('Godine')
plt.ylabel('Potrosacki skor')

plt.show()

#####################################


import sklearn
from sklearn.cluster import KMeans

X = np.concatenate((x11,x21))
Y = np.concatenate((x12,x22))
XY = np.concatenate((X,Y), axis=1)

kmeans = KMeans(n_clusters=2, random_state=0)
kmeans = kmeans.fit(XY)
labels = kmeans.predict(XY)
centroidi = kmeans.cluster_centers_


Prvaklasax=[]
Prvaklasay=[]
```

```
Drugaklasax=[]
Drugaklasay=[]

for i in range(0,len(labels)):
  if (labels[i]==0):
    Prvaklasax.append(XY[i,0])
    Prvaklasay.append(XY[i,1])
  elif (labels[i]==1):
    Drugaklasax.append(XY[i,0])
    Drugaklasay.append(XY[i,1])

plt.plot(Prvaklasax,Prvaklasay,'g*')
plt.plot(Drugaklasax,Drugaklasay,'r*')

plt.plot(centroidi[0,0],centroidi[0,1],'ko',linewidth=3)
plt.plot(centroidi[1,0],centroidi[1,1],'ko',linewidth=3)

plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Prikaz klasa nakon klasterizacije pomoću Scikit-learn biblioteke')
plt.show()



####################################
```

```
X = np.concatenate((x11,x21))
Y = np.concatenate((x12,x22))
XY = np.concatenate((X,Y), axis=1)

kmeans = KMeans(n_clusters=3, random_state=0)
kmeans = kmeans.fit(XY)
labels = kmeans.predict(XY)
centroidi = kmeans.cluster_centers_


Prvaklasax=[]
Prvaklasay=[]
Drugaklasax=[]
Drugaklasay=[]
Trecaklasax=[]
Trecaklasay=[]


for i in range(0,len(labels)):
  if (labels[i]==0):
    Prvaklasax.append(XY[i,0])
    Prvaklasay.append(XY[i,1])
  elif (labels[i]==1):
```

```
    Drugaklasax.append(XY[i,0])
    Drugaklasay.append(XY[i,1])
  elif (labels[i]==2):
    Trecaklasax.append(XY[i,0])
    Trecaklasay.append(XY[i,1])

plt.plot(Prvaklasax,Prvaklasay,'g*')
plt.plot(Drugaklasax,Drugaklasay,'r*')
plt.plot(Trecaklasax,Trecaklasay,'y*')
plt.plot(centroidi[0,0],centroidi[0,1],'ko',linewidth=3)
plt.plot(centroidi[1,0],centroidi[1,1],'ko',linewidth=3)
plt.plot(centroidi[2,0],centroidi[2,1],'ko',linewidth=3)

plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Prikaz klasa nakon klasterizacije pomoću Scikit-learn biblioteke')
plt.show()



#####################################


X = np.concatenate((x11,x21))
Y = np.concatenate((x12,x22))
```

```
XY = np.concatenate((X,Y), axis=1)


kmeans = KMeans(n_clusters=4, random_state=0)
kmeans = kmeans.fit(XY)
labels = kmeans.predict(XY)
centroidi = kmeans.cluster_centers_



Prvaklasax=[]
Prvaklasay=[]
Drugaklasax=[]
Drugaklasay=[]
Trecaklasax=[]
Trecaklasay=[]
Cetvrtaklasax=[]
Cetvrtaklasay=[]
for i in range(0,len(labels)):
  if (labels[i]==0):
    Prvaklasax.append(XY[i,0])
    Prvaklasay.append(XY[i,1])
  elif (labels[i]==1):
    Drugaklasax.append(XY[i,0])
    Drugaklasay.append(XY[i,1])
  elif (labels[i]==2):
    Trecaklasax.append(XY[i,0])
```

```
    Trecaklasay.append(XY[i,1])
  elif (labels[i]==3):
    Cetvrtaklasax.append(XY[i,0])
    Cetvrtaklasay.append(XY[i,1])
#####################################



plt.plot(Prvaklasax,Prvaklasay,'g*')
plt.plot(Drugaklasax,Drugaklasay,'b*')
plt.plot(Trecaklasax,Trecaklasay,'r*')
plt.plot(Cetvrtaklasax,Cetvrtaklasay,'y*')
plt.plot(centroidi[0,0],centroidi[0,1],'ko',linewidth=3)
plt.plot(centroidi[1,0],centroidi[1,1],'ko',linewidth=3)
plt.plot(centroidi[2,0],centroidi[2,1],'ko',linewidth=3)
plt.plot(centroidi[3,0],centroidi[3,1],'ko',linewidth=3)
plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Prikaz klasa nakon klasterizacije pomoću Scikit-learn biblioteke')
plt.show()
```

#Ako se radi o mladjoj populaciji, birali bi klaster koja je obelezen crvenom bojom.

#Ako odaberemo zeleni klaster, imamo ljude koji su ispod 30 godina, a takodje imaju prosecan potrošački skor (izmedju 40 i 60)

#Zuti i plavi klasteri nam nisu od preteranog značaja, jer se ili radi o ljudima starijih godina ili je potrošački skor nizak

############################ ZADATAK 2 ############################

```
import pandas as pd
import numpy as np
import pylab as plt

data = pd.read_csv('/content/Detekcija_sloga.csv')

data.head(10)

########################################

#Prvo da vidimo za pusace

pacijenti = data.groupby('smoking_status')['stroke'].agg(sum)

fig = plt.figure()
ax = fig.add_subplot(111)
rect = ax.bar(pacijenti.index.values.tolist(), pacijenti, color = 'blue', width = 0.7)
ax.set_ylabel('Broj slogova')
ax.set_title('Ukupan broj ljudi sa slogom')
xSmokingState = pacijenti.index.values.tolist()
ax.set_xticks(pacijenti.index.values.tolist())
xSmokingStatus = ax.set_xticklabels(xSmokingState)
plt.setp(xSmokingStatus, fontsize = 10)
```

```
plt.show()


########################################

muskarci_slog = data[data['gender'] == 'Male'].groupby('ever_married')['stroke'].agg(sum)
zene_slog = data[data['gender'] == 'Female'].groupby('ever_married')['stroke'].agg(sum)
bar_width = 0.3


fig = plt.figure()

ax = fig.add_subplot(111)
index = np.arange(muskarci_slog.count())
rect1 = ax.bar(index, muskarci_slog, bar_width ,color='blue', label = 'Muskarci')
rect2 = ax.bar(index + bar_width,zene_slog, bar_width, color='y', label = 'Zene')
xMarried = muskarci_slog.index.values.tolist()
ax.set_xticks(index + bar_width)
xMarredStatus = ax.set_xticklabels(xMarried)
plt.setp(xMarredStatus)
plt.legend()
plt.title('Muskarci i zene koji su imali slog, solo i u braku')
plt.show()
```

```
########################################

data1 = data.to_numpy()

X1 = data1[:251,2]
X2 = data1[:251,8]

Y1 = data1[251:,2]
Y2 = data1[251:,8]

plt.scatter(Y1, Y2, color = 'blue',label = 'Nema slog')
plt.scatter(X1, X2, color = 'red',label = 'Ima slog')
plt.legend()
plt.title('Godine i nivo glukoze u odnosu na slog')
plt.show()

########################################

#Brojim nepostojece vrednosti:
total = data.isnull().sum().sort_values(ascending=False)
percent_1 = data.isnull().sum()/data.isnull().count()*100
percent_2 = (round(percent_1, 1)).sort_values(ascending=False)
missing = pd.concat([total, percent_2], axis=1, keys = ['Total', '%'])
```

```
print(missing)



#Samo 3.9% nedostaje tako da cemo zameniti Nan sa srednjom vrednosti (ne moramo da izbacimo kolonu)


########################################


for i in data:
  mean = data['bmi'].mean()
  std = data['bmi'].std()
  is_null = data['bmi'].isnull().sum()
  random_bmi = np.random.randint(mean -std, mean+std, size = is_null)
  bmi_s = data['bmi'].copy()
  bmi_s[np.isnan(bmi_s)] = random_bmi
  data['bmi'] = bmi_s
data['bmi'].isnull().sum()


#Mozemo da ponovimo kod iznad tako cemo znati da nam sigurno nema 0 a takodje, na kraju smo ispisali koliko Nan vrednosti ima i dobili smo 0


########################################



for i in data:
  data['age'] = data['age'].astype(int)
```

```
  data['avg_glucose_level'] = data['avg_glucose_level'].astype(int)

  data['bmi'] = data['bmi'].astype(int)

  data['ever_married'] = data['ever_married'].replace('Yes', 1).replace('No', 0)

  data['gender'] = data['gender'].replace('Male', 0).replace('Female', 1).replace('Other',2)

  data['work_type'] = data['work_type'].replace('Private', 1).replace('Self-employed',
0).replace('Govt_job',2).replace('children', 3).replace('Never_worked', 4)

  data['Residence_type'] = data['Residence_type'].replace('Urban', 1).replace('Rural', 0)

  data['smoking_status'] = data['smoking_status'].replace('Unknown', 0).replace('never smoked',
1).replace('formerly smoked', 2).replace('smokes', 3)



data.head(10)

#Zaminili smo sve vrednosti sa numerickim


##########################################


data1 = [data]

for dataset in data1:

  dataset.loc[dataset['age'] <=20, 'age'] = 0

  dataset.loc[(dataset['age'] > 20) & (dataset['age'] <=30), 'age'] = 1

  dataset.loc[(dataset['age'] > 30) & (dataset['age'] <=40), 'age'] = 2

  dataset.loc[(dataset['age'] > 40) & (dataset['age'] <=50), 'age'] = 3

  dataset.loc[(dataset['age'] > 50) & (dataset['age'] <=60), 'age'] = 4

  dataset.loc[(dataset['age'] > 60) & (dataset['age'] <=70), 'age'] = 5

  dataset.loc[dataset['age'] > 70, 'age'] = 6
```

```
data.head(20)

##########################################

data1 = [data]

for dataset in data1:

  dataset.loc[dataset['avg_glucose_level'] <=60, 'avg_glucose_level'] = 0

  dataset.loc[(dataset['avg_glucose_level'] > 60) & (dataset['avg_glucose_level'] <=100), 'avg_glucose_level'] = 1

  dataset.loc[(dataset['avg_glucose_level'] > 100) & (dataset['avg_glucose_level'] <=140), 'avg_glucose_level'] = 2

  dataset.loc[(dataset['avg_glucose_level'] > 140) & (dataset['avg_glucose_level'] <=180), 'avg_glucose_level'] = 3

  dataset.loc[(dataset['avg_glucose_level'] > 180) & (dataset['avg_glucose_level'] <=220), 'avg_glucose_level'] = 4

  dataset.loc[dataset['avg_glucose_level'] > 220, 'avg_glucose_level'] = 5



data.head(20)

##########################################

data1 = [data]

for dataset in data1:

  dataset.loc[dataset['bmi'] <=18, 'bmi'] = 0

  dataset.loc[(dataset['bmi'] > 18) & (dataset['bmi'] <=22), 'bmi'] = 1

  dataset.loc[(dataset['bmi'] > 22) & (dataset['bmi'] <=28), 'bmi'] = 2
```

```
  dataset.loc[(dataset['bmi'] > 28) & (dataset['bmi'] <=33), 'bmi'] = 3
  dataset.loc[(dataset['bmi'] > 33) & (dataset['bmi'] <=40), 'bmi'] = 4
  dataset.loc[(dataset['bmi'] > 40) & (dataset['bmi'] <=50), 'bmi'] = 5
  dataset.loc[(dataset['bmi'] > 50) & (dataset['bmi'] <=70), 'bmi'] = 6
  dataset.loc[dataset['bmi'] > 70, 'bmi'] = 7


data.head(20)

##########################################

target = data['stroke']
data = data.drop('stroke', axis = 1)
data.head()
print('Skalirane godine')
print(data['age'].value_counts())
print('Skaliran prosecan nivo glukoze')
print(data['avg_glucose_level'].value_counts())
print('Skaliran indeks telesne mase')
print(data['bmi'].value_counts())

##########################################

Y = target.values
```

```
X =
data[["gender","age","hypertension","heart_disease","ever_married","work_type","Residence_type","avg_glucose_level","bmi","smoking_status"]].values


Y11 = Y[:600]

X11 = X[:600]


########################################


from sklearn.model_selection import train_test_split

X_obucavajuci, X_testirajuci, Y_obucavajuci, Y_testirajuci= train_test_split(X, Y, test_size=0.20, random_state=1)


broj_featuresa = X_obucavajuci.shape[1]

broj_featuresa


X_testirajuci1, X_testMali, Y_testirajuci1, Y_testMali = train_test_split(X_testirajuci, Y_testirajuci, test_size=0.007, random_state=1)


X_testirajuci = X_testirajuci1

Y_testirajuci = Y_testirajuci1

print('Obucavajuci X i : ',X_obucavajuci.shape,Y_obucavajuci.shape)

print('Testirajuci X i Y: ',X_testirajuci.shape,Y_testirajuci.shape)

print('Mali deo za predikciju X i Y', X_testMali.shape, Y_testMali.shape)
```

```
########################################

import tensorflow as tf

import tensorflow.keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

model = Sequential()

model.add(Dense(10, activation='relu', kernel_initializer= 'uniform',
input_shape=(broj_featuresa,)))

model.add(Dense(5, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

########################################

model.fit(X_obucavajuci, Y_obucavajuci, epochs=50, batch_size=32, verbose=0)

########################################

#Evaluacija modela na celom testirajucem skupu

loss, acc = model.evaluate(X_testirajuci, Y_testirajuci, verbose = 0)

print('Tacnost modela je: %.5f' % acc)
```

```
predikcija = model.predict(X_testirajuci)

#Model ima tacnost 94%!

##########################################

#Evaluacija na malom delu skupa (8 instanci)

X_testirajuci = X_testMali
Y_testirajuci = Y_testMali

loss, acc = model.evaluate(X_testirajuci, Y_testirajuci, verbose = 2)
print('Tacnost predikcije je: %.5f' % acc)
predikcija = model.predict(X_testirajuci)

#7 od 8 instanci je tačno prediktovano.

##########################################

X = X11
Y = Y11

from sklearn.model_selection import train_test_split
```

```
X_obucavajuci, X_testirajuci, Y_obucavajuci, Y_testirajuci= train_test_split(X, Y, test_size=0.20, random_state=1)


broj_featuresa = X_obucavajuci.shape[1]

broj_featuresa


X_testirajuci1, X_testMali, Y_testirajuci1, Y_testMali = train_test_split(X_testirajuci, Y_testirajuci, test_size=0.007, random_state=1)


X_testirajuci = X_testirajuci1

Y_testirajuci = Y_testirajuci1

print('Obucavajuci X i : ',X_obucavajuci.shape,Y_obucavajuci.shape)

print('Testirajuci X i Y: ',X_testirajuci.shape,Y_testirajuci.shape)

print('Mali deo za predikciju X i Y', X_testMali.shape, Y_testMali.shape)




########################################


import tensorflow as tf

import tensorflow.keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense


model = Sequential()
```

```
model.add(Dense(10, activation='relu', kernel_initializer= 'uniform',
input_shape=(broj_featuresa,)))

model.add(Dense(5, activation='relu'))

model.add(Dense(1, activation='sigmoid'))


model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])


##########################################


model.fit(X_obucavajuci, Y_obucavajuci, epochs=50, batch_size=32, verbose=0)


##########################################


loss, acc = model.evaluate(X_testirajuci, Y_testirajuci, verbose = 0)

print('Tacnost modela je: %.5f' % acc)

predikcija = model.predict(X_testirajuci)


##########################################
```