13E053SOM - Projektni zadatak iz predmeta Sistemi odlučivanja u medicini

Aleksa Janjić 2019/0021

Septembar 2022.

Dataset 20: COVID-19 *presence*

A 237366

Образац бр. 1

ЈМБГ 1502001773622

Алекса Јањић

(име и презиме - штампаним словима)

(својеручни потпис студента)

* **Врста студија:** основне академске, основне струковне, специјалистичке струковне студије, интегрисане основне и мастер академске, мастер академске, мастер струковне, специјалистичке академске, докторске академске.

** **Степен студија:** први, други, трећи.

A 237366

РЕПУБЛИКА СРБИЈА

Универзитет у Београду

(назив и седиште самосталне високошколске установе)

Електротехнички факултет

(назив и седиште високошколске установе)

Број индекса 0021 / 2019

(број) (година уписа)

ОС

СТУДЕНТСКА КЊИЖИЦА

ИНДЕКС

Алекса Јањић

(име и презиме)

Раде — 15.02.2001.

(име једног родитеља) (датум рођења)

Лозница — Лозница

(место рођења) (општина рођења)

Република Србија — Републике Србије

(држава рођења) (држављанство)

уписан-а је школске 2018 /2020 .године на

основне академске — први

(врста студија)* (степен студија)**

електротехника и рачунарство

(назив студијског програма)

240/4 године

(број изражен у ЕСПБ и време трајања студијског програма)

Овлашћено лице

(М.П.)

(датум уписа)

Sadržaj

1 Uvod **3**

2 Analiza skupa podataka **4**

3 Korelacija između obeležja i *Information Gain* **5**

4 Redukcija dimenzionalnosti - LDA i PCA metodi **6**

5 Parametarski klasifikator **7**

6 Neuralna mreža **8**
6.1 Različite strukture neuralne mreže 8
6.2 Zaštita od preobučavanja - regularizacija 11
6.3 Zaštita od preobučavanja - rano zaustavljanje 11

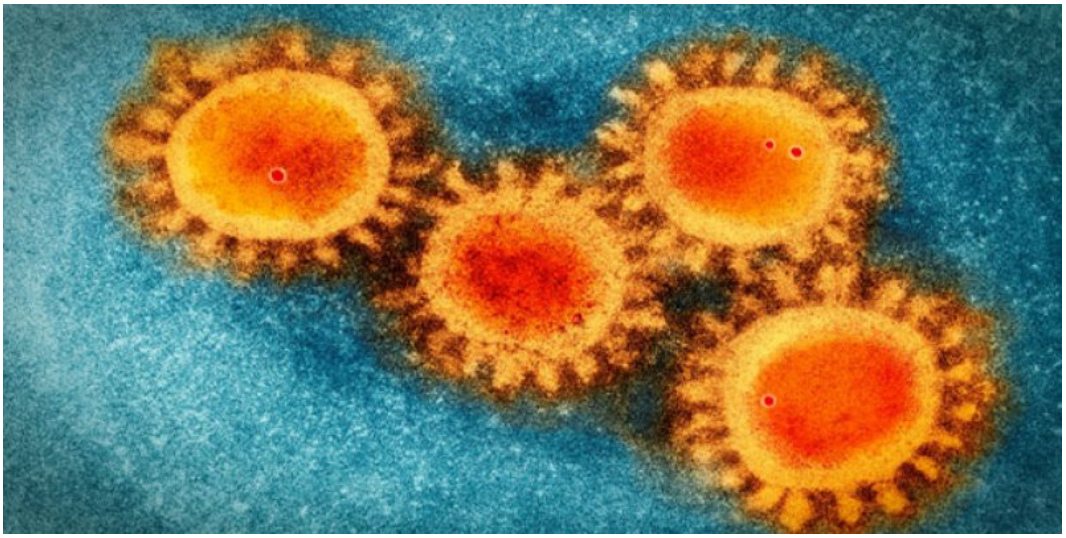
7 Programski kod u programskom jeziku *Python* **12**

1 Uvod

U proteklih nekoliko godina svima nama je termin **COVID-19** savršeno poznat, pošto je svima nama u proteklom periodu život u manjoj ili većoj meri promenjen zbog tog termina. **COVID-19** ili koronavirusna bolest 2019 (engl. *Coronavirus disease 2019*) je zarazna bolest uzrokovana teškim akutnim respiratornim sindromom virus korona 2 (SARS-CoV-2) koja je 2019. godine proglašena pandemijom.

Infekcija se širi od jedne do druge osobe kapljičnim putem - respiratornim kapljicama nastalim tokom kašljanja. Vreme inkubacije do pojave prvih simptoma je obično između 2 i 14 dana, a u proseku oko 5 dana. Uobičajeni simptomi su telesna temperature, kašalj i otežano disanje, dok su nešto ređi simptomi gubitak čula mirisa i ukusa, kao i bol u mišićima i grlobolja.

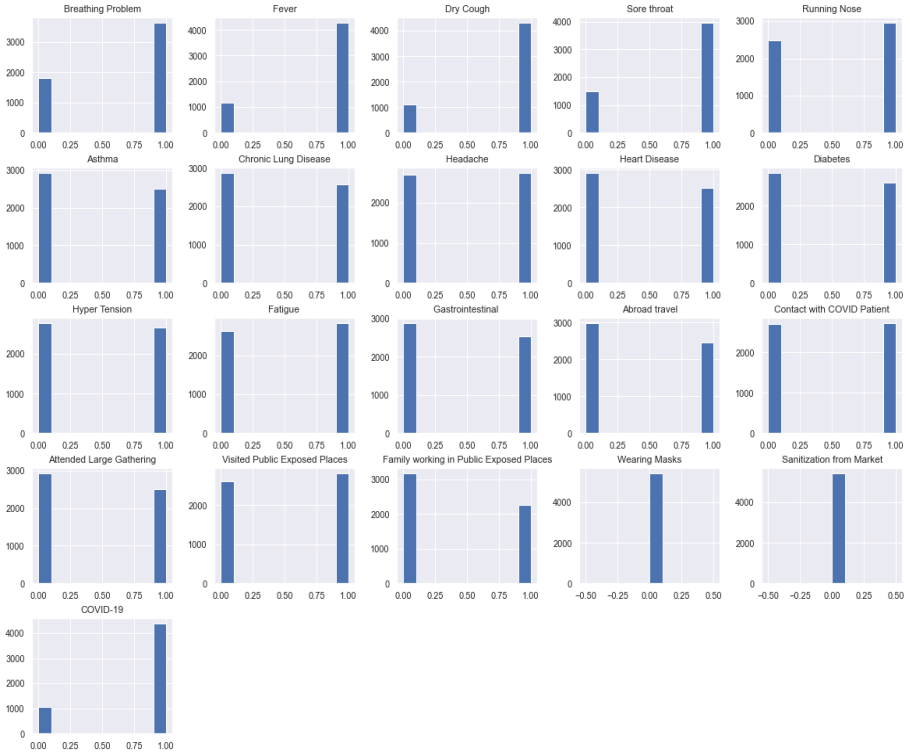
Standardan metod dijagnoze je lančana reakcija polimeraznom rezervnom transkripcijom (rRT-PCR) iz nazofaringealnog brisa ili uzorka ispljuvaka, a infekcija se može registrovati i ispitivanjem prisutnosti antitela iz uzorka krvnog seruma ili iz kombinacije simptoma, faktora rizika zaraze i CT-a prsnog koša koji pokazuje karakteristike upale pluća. Usled same činjenice da se radi o globalnoj pandemiji, a imajući u vidu da se i sama zaraza vrlo lako i brzo širi, broj obolelih ljudi je iz dana u dan vrtoglavo rastao, samim tim i mnogi zdravstveni sistemi su bili preopterećeni i nisu bili spremni da logistički, kadrovski i medicinski odgovore na ovu pandemiju. Sam postupak detekcije prisustva **COVID-19** u telu osobe zahteva određeno vreme koje je značajno veće u odnosu na vreme koje je neophodno da se infekcija proširi između dve ili više osoba, stoga se tragalo za rešenjem koje bi bilo moguće da instantivno odgovori na pitanje da li osoba je zaražena virusom ili nije. U tu svrhu, u ovom radu će se na osnovu dostupnog seta podataka ispitivati određeni simptomi, porediti ih na odgovarajući način i pokušati da pojednostavimo podatke i vezu između njih, samim tim i posmatrani problem, tako da možemo finalno izvršiti odgovarajuću klasifikaciju i izvesti generalizovani zaključak po kom će se donositi odluka da li je osoba zaražena ili ne.



Slika 1: Mikroskopski prikaz virusa SARS-CoV-2

2 Analiza skupa podataka

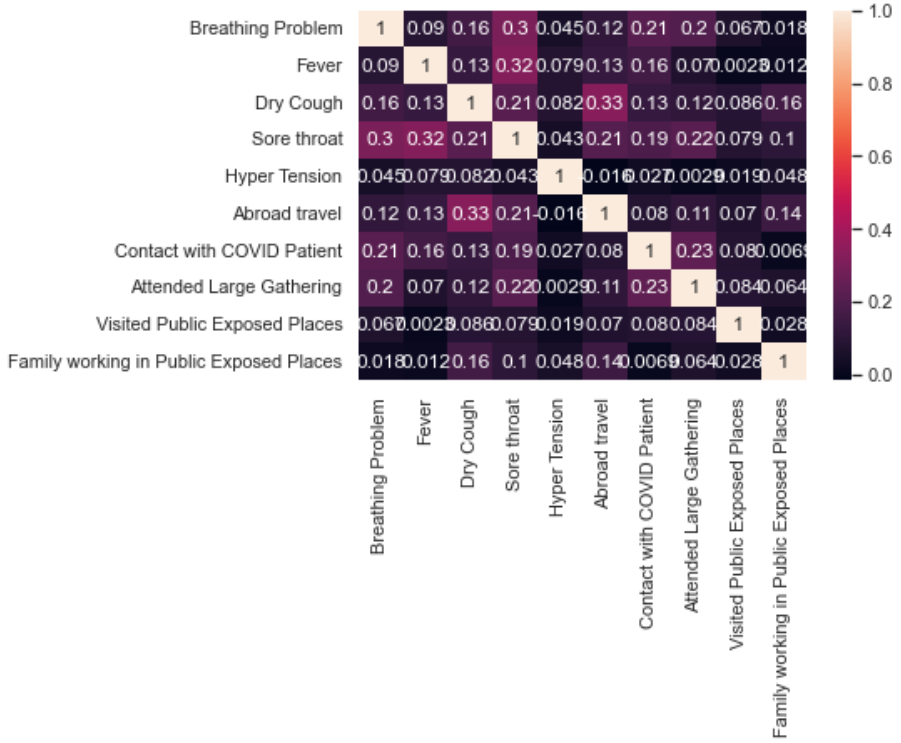
U posmatranom skupu podataka postoji 20 obeležja koji su razvrstani po kolonama, i poslednja kolona koja predstavlja izlazne podatke o tome da li je osoba sa prethodnim vrednostima atributa zaražena ili ne. Svi podaci su DA/NE tipa, stoga je izvršena konverzija stringova DA (u podacima označenih sa *'Yes'*) u celobrojnu vrednost 1, a stringovi NE (u podacima označeni sa *'No'*) u celobrojnu vrednost 0. Naknadnom proverom uočavamo da ne postoji nijedna nedostajuća vrednost, stoga nije potrebno dalje modifikovati podatke u smislu zamenjivanjem očekivanom vrednošću ili izbacivanjem odbirka. Na slici 2 primećujemo da imamo dva neinformativna obeležja - '*Wearing masks*' i '*Sanitization from Market*', pošto za svaki podatak oni imaju uvek istu vrednost, stoga u startu odmah odbacujemo ova dva obeležja.



Slika 2: Histogram obeležja i izlaza nakon adekvatnog kodovanja vrednosti

3 Korelacija između obeležja i *Information Gain*

Prvo ćemo posmatrati korelaciju sa klasom preostalih 18 obeležja uz pomoć ugrađene funkcije ***corr*** (*Pearson*-ov metod), zatim da te vrednosti sortiramo, a onda da 10 obeležja sa najvećom (apsolutnom) vrednošću koeficijenta korelacije tog obeležja sa klasom sačuvamo, a ostale odbacimo. U nastavku je na slici 3 priložen grafički prikaz korelacione matrice izdvojenih 10 obeležja *Pearson*-ovim metodom, dok *Spearman*-ovim metodom je u programsku kodu izvršena, uslovno rečeno verifikacija *Pearson*-ovog metoda, radi ispitivanja postojanja eventualne razlike u pozivanju funkcije ***corr*** za svaki metod. Oba metoda daju slične rezultate.



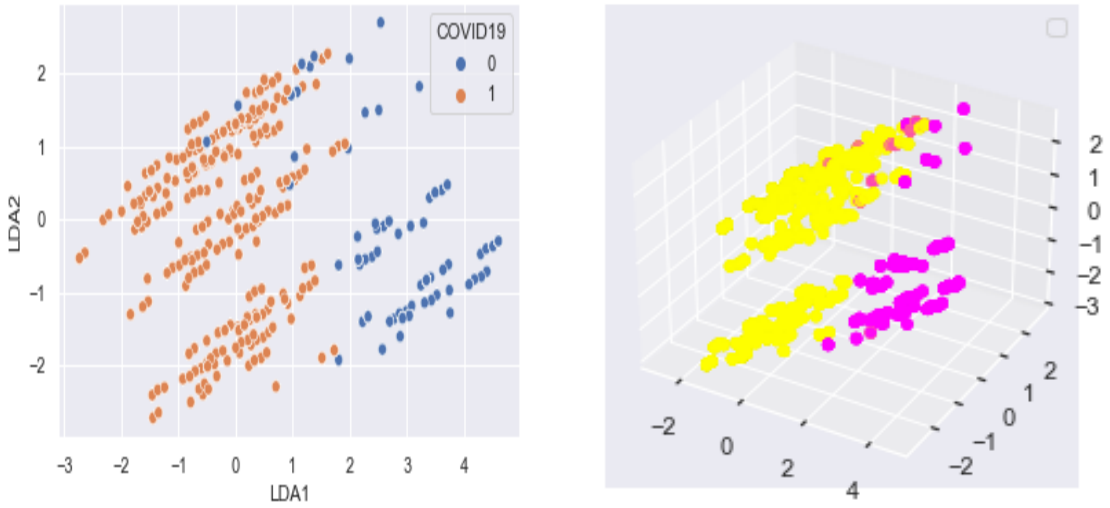
Slika 3: Ispitivanje korelacije između obeležja *Pearson*-ovim metodom

Analizom dobijene korelacione matrice primećujemo da je najveći koeficijent korelacije između dva atributa iznosi 0.33, pa s obzirom na to da ova vrednost nije velika, atributi su dobro odabrani. U nastavku je prikaz tabele koja sadrži koeficijente korelacije između datog obeležja i klase, kao i vrednost IG (*Information Gain*) parametra tog obeležja.

	Koeficijent korelacije sa klasom	IG parametar
Sore throat	0.502848	0.16509021190420425
Dry Cough	0.464292	0.13203769535628973
Abroad travel	0.443875	0.19466403318778536
Breathing Problem	0.443764	0.1359714631581188
Attended Large Gathering	0.390145	0.1299523075927551
Contact with COVID Patient	0.357122	0.10050174342266127
Fever	0.352891	0.07787631667975581
Family working in Public Exposed Places	0.160208	0.01939488782649168
Visited Public Exposed Places	0.119755	0.01038130864329434
Hyper Tension	0.102575	0.007652495823298078

Tabela 1: Koeficijenti korelacije sa klasom i vrednosti IG parametara

4 Redukcija dimenzionalnosti - LDA i PCA metodi



Slika 4: Grafik raspodele odbiraka po klasama nakon redukcije **LDA** metodom na 2 i 3 dimenzije respektivno

Na slici 4 prikazani su grafici raspodele odbiraka po klasama nakon redukcije na 2 i 3 dimenzije **LDA** (engl. *Linear Discriminant Analysis*) metodom i primećujemo da nijedan od prethodnih načina ne separatiše odbirke stoprocentno dobro, ali stiče se utisak da **LDA** redukcija na 2 dimenzije pruža bolju separabilnost od **LDA** redukcije na 3 dimenzije, stoga ćemo u nastavku projekta smatrati da **LDA** redukcija na 2 dimenzije pruža dovoljno zadovoljavajuću separabilnost podataka.

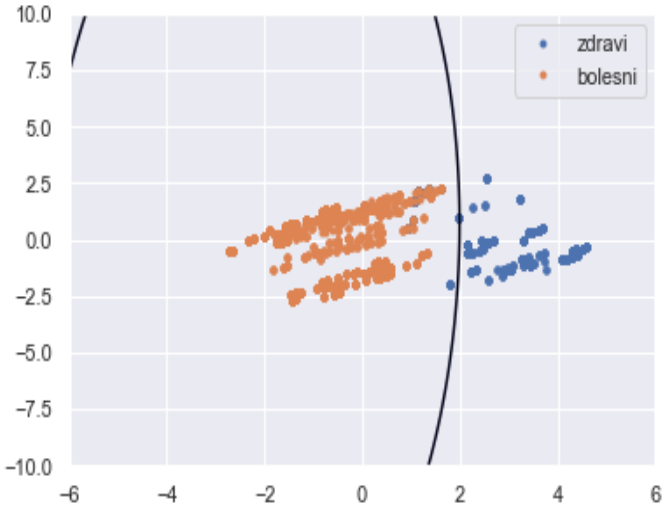
Slika 5: Grafik raspodele odbiraka po klasama nakon redukcije **LDA** metodom na jednu dimenziju i **PCA** metodom na 2 dimenzije respektivno

Imajući u vidu da je skup podataka podeljen na dve klase, potrebno je uraditi još dva načina redukcije dimenzionalnosti- **LDA** redukcija na jednu dimenziju i **PCA** (engl. *Principal Component Analysis*) redukcija na 2 dimenzije. Sa slika 6 i 7 primećujemo da nijedan od ovih načina ne pruža zadovoljavajuću separabilnost, stoga zaključujemo da najbolju separabilnost sadrži **LDA** redukcija na 2 dimenzije.

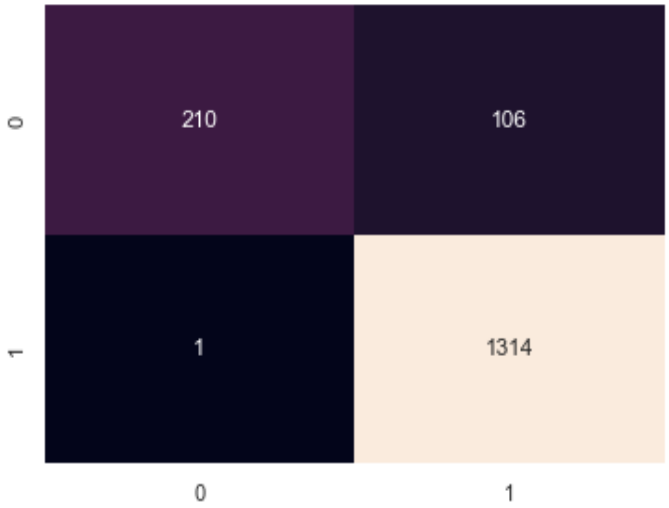
Bitno je napomenuti da na prethodnim graficima u legendama plave tačke (vrednost 0) označavaju negativne (zdrave) pojedince, a narandžaste tačke (vrednost 1) označavaju pozitivne (bolesne) pojedince.

5 Parametarski klasifikator

U prethodnoj tački diskutovali smo o separabilnosti koja se dobija raznim načinima i zaključili smo da najbolju separabilnost pruža **LDA** metod redukcije na 2 dimenzije, pa je sledeći korak projektovanje odgovarajućeg parametarskog klasifikatora po slobodnom izboru. U ovoj tački biće projektovan kvadratni klasifikator na bazi željenog izlaza, sa podelom na trenirajući i testirajući skup u razmeri **70:30**.



Slika 6: Grafički prikaz raspodele odbiraka trenirajućeg skupa i diskriminacione krive u ravni



Slika 7: Grafički prikaz konfuzione matrice izvedene iz testirajućeg skupa

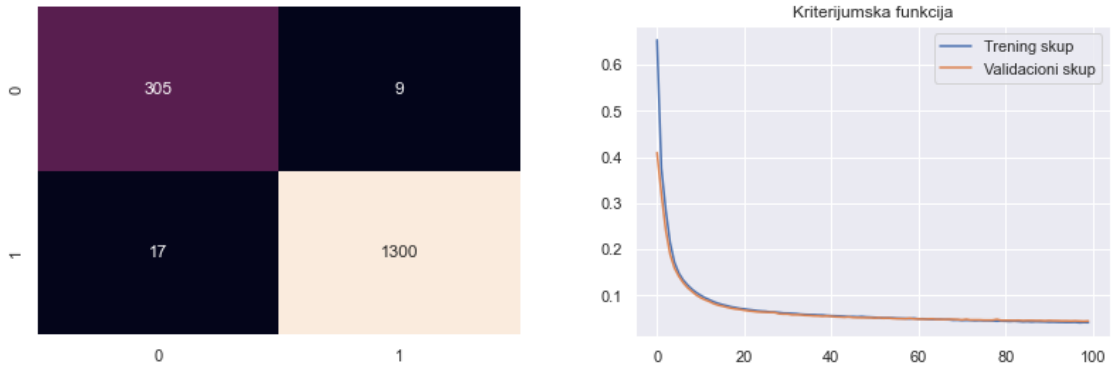
Projektovani klasifikator ima veoma veliku tačnost (93.44%) i ima gotovo zanemarljiv procenat *lažnih alarma*, međutim, jedina mana ovog klasifikatora jeste procenat *propuštenih detekcija* koji iznosi 6.5%.

6 Neuralna mreža

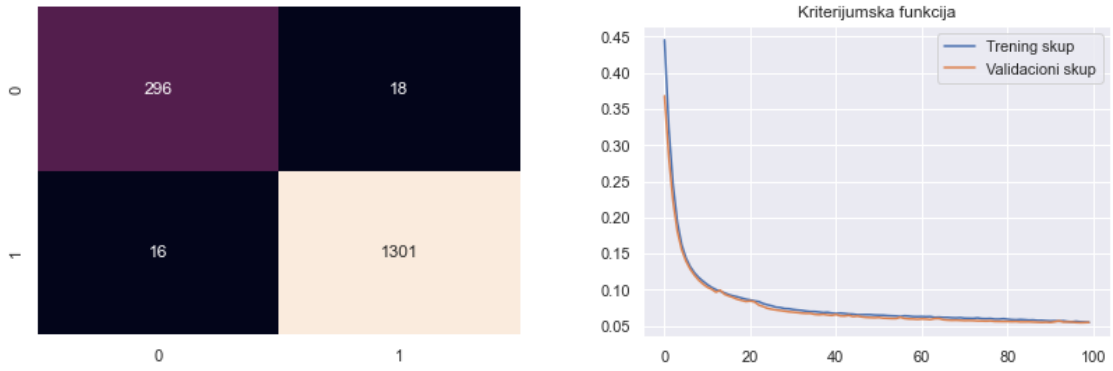
6.1 Različite strukture neuralne mreže

Poslednji deo projekta podrazumeva treniranje i testiranje različitih struktura *feedforward* neuralne mreže sa jednim ili više skrivenih slojeva, kao i sa različitim brojem neurona u slojevima. U nastavku će kao rezultati testiranja biti prikazane konfuzione matrice za svaku strukturu neuralne mreže. Takođe, treba napomenuti da je svaka neuralna mreža pokrenuta u **100 epoha**, kao i da je zadržana razmera **70:30** trenirajućeg i testirajućeg skupa.

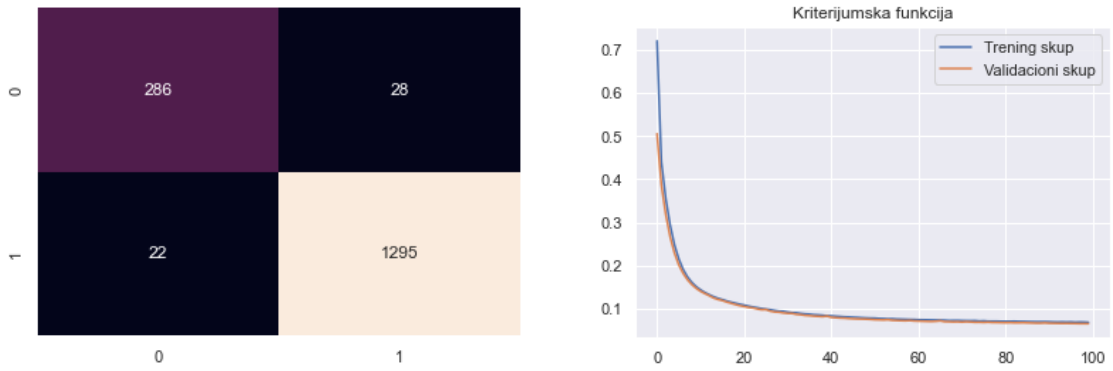
Primetićemo da svaka od struktura daje veoma zadovoljavajuće tačnosti, međutim, što je struktura kompleksnija, to je vreme izvršavanja programa duže, a takođe je dinamika kriterijumskih funkcija veća. Kod struktura sa jednim skrivenim slojem primećujemo da je sa smanjenjem broja neurona u skrivenom sloju manja i tačnost, što znači da nisu adekvatne ni suviše jednostavne strukture neuralne mreže. Najbolje rezultate dale su neuralne mreže sa **jednim** skrivenim slojem od **20** neurona (tačnost **98.406%**), kao i neuralna mreža sa **dva** skrivena sloja od **20** i **5** neurona respektivno (tačnost **98.283%**).



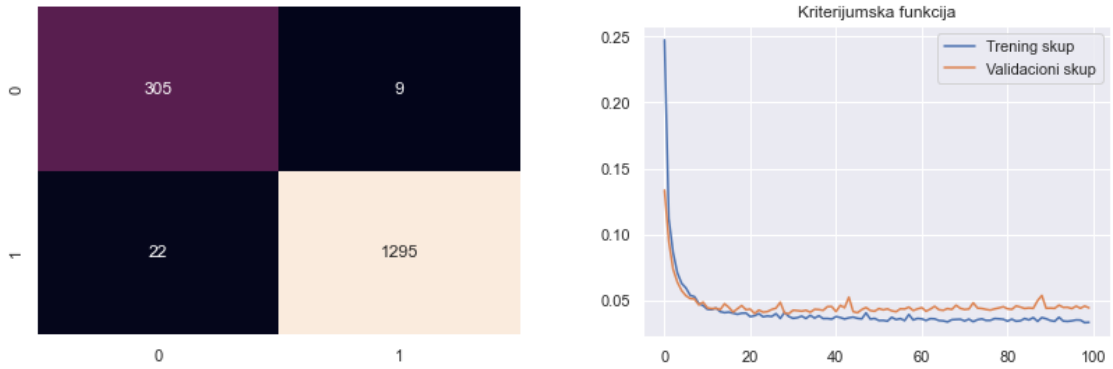
Slika 8: Grafički prikaz konfuzione matrice i kriterijumske funkcije na trening i validacionom skupu za neuralnu mrežu sa **jednim** skrivenim slojem sa **20** neurona (tačnost **98.406%**)



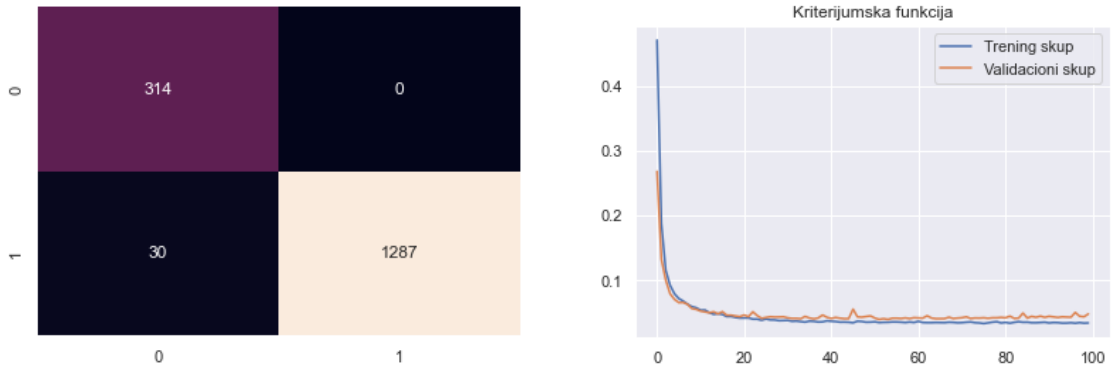
Slika 9: Grafički prikaz konfuzione matrice i kriterijumske funkcije na trening i validacionom skupu za neuralnu mrežu sa **jednim** skrivenim slojem sa **10** neurona (tačnost **97.915%**)



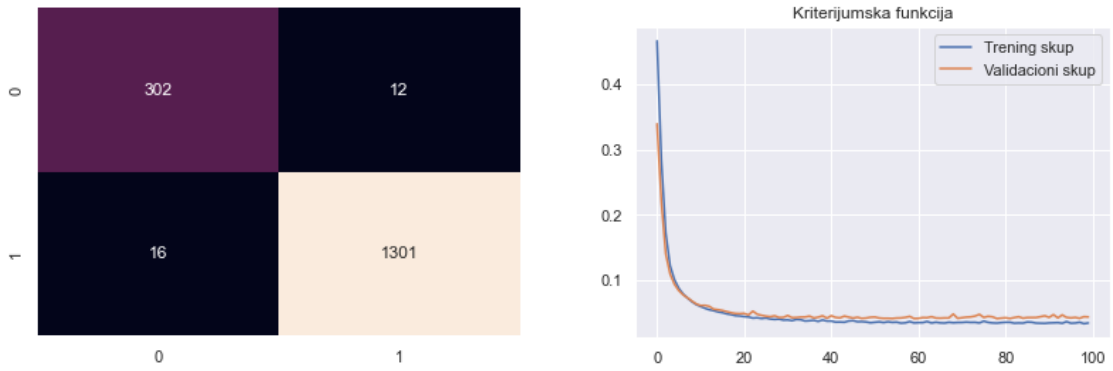
Slika 10: Grafički prikaz konfuzione matrice i kriterijumske funkcije na trening i validacionom skupu za neuralnu mrežu sa **jednim** skrivenim slojem sa **3** neurona (tačnost **96.934%**)



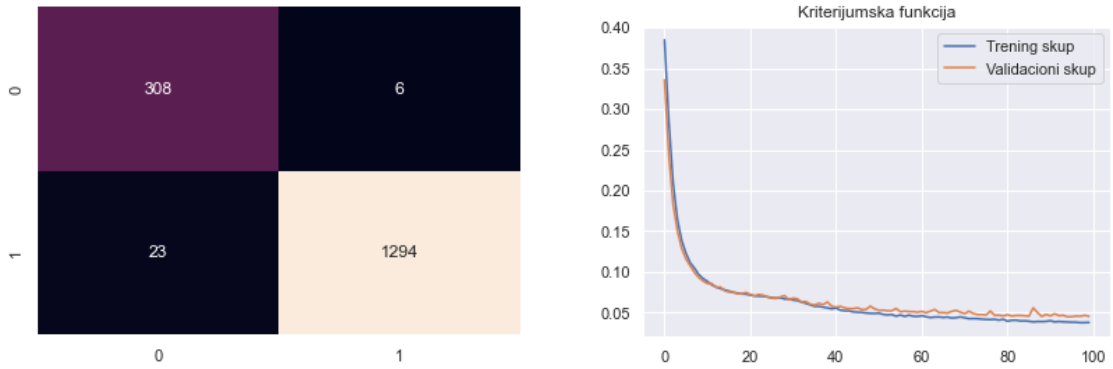
Slika 11: Grafički prikaz konfuzione matrice i kriterijumske funkcije na trening i validacionom skupu za neuralnu mrežu sa **jednim** skrivenim slojem sa **500** neurona (tačnost **98.099%**)



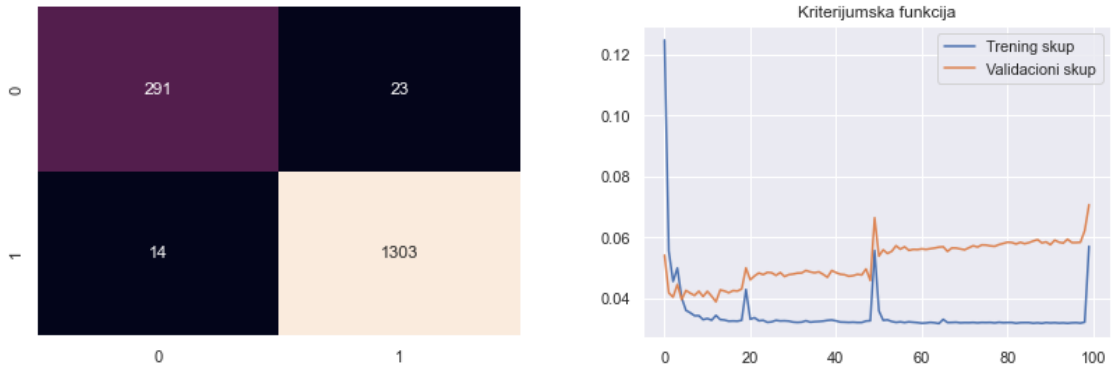
Slika 12: Grafički prikaz konfuzione matrice i kriterijumske funkcije na trening i validacionom skupu za neuralnu mrežu sa **dva** skrivena sloja sa **20** i **15** neurona (tačnost **98.161%**)



Slika 13: Grafički prikaz konfuzione matrice i kriterijumske funkcije na trening i validacionom skupu za neuralnu mrežu sa **dva** skrivena sloja sa **20** i **5** neurona (tačnost **98.283%**)



Slika 14: Grafički prikaz konfuzione matrice i kriterijumske funkcije na trening i validacionom skupu za neuralnu mrežu sa **dva** skrivena sloja sa **10** i **3** neurona (tačnost **98.222%**)

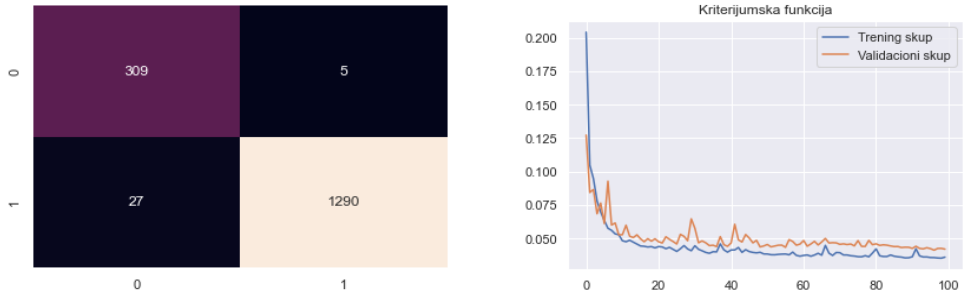


Slika 15: Grafički prikaz konfuzione matrice i kriterijumske funkcije na trening i validacionom skupu za neuralnu mrežu sa **dva** skrivena sloja sa **1000** i **1000** neurona (tačnost **97.731%**)

6.2 Zaštita od preobučavanja - regularizacija

Do preobučavanja neuralne mreže dolazi zbog prevelike kompleksnosti neuralne mreže i tada neuralna mreža prilikom treniranja, uslovno govoreći "uči napamet" na osnovu trenirajućeg seta podataka. Posledica toga su loši rezultati na novim podacima, to jest, na testirajućem skupu podataka. Za potrebe prikazivanja efekta preobučavanja, koristićemo neuralnu mrežu sa dva skrivena sloja od po 1000 neurona u svakom (o kojoj je bilo reči u prethodnoj tački), jer je ona dovoljno kompleksna. U prethodnom primeru su se dobili veoma dobri rezultati ove mreže, ali probaćemo da dobijemo još bolje uvođenjem tehnike **regularizacije** i **ranog zaustavljanja**.

Regularizacija predstavlja tehniku kojom se sprečava preobučavanje tako što se standardna kriterijumska funkcija (obično srednje-kvadratna greška) proširuje dodatnim težinskim članom (engl. *weight decay*) kako bi se sprečilo da velike težine dovedu do predela prezasićenja.

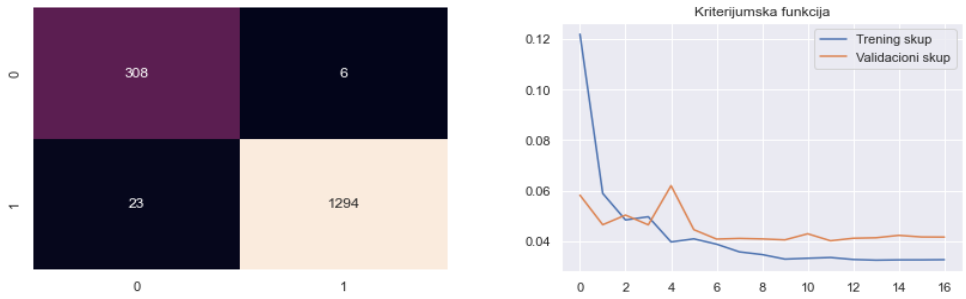


Slika 16: Grafički prikaz konfuzione matrice i kriterijumske funkcije na trening i validacionom skupu za neuralnu mrežu sa **dva** skrivena sloja sa **1000** i **1000** neurona nakon uvođenja **regularizacije** (tačnost **98.038%**)

Uvođenjem regularizacije smo postigli još bolju tačnost naše kompleksne neuralne mreže (poboljšanje od oko **0.30%**), međutim, kao što možemo primetiti na slici 16, kriterijumska funkcija ima značajno manju dinamiku ukoliko se uvede **regularizacija** u odnosu na dinamiku kriterijumske funkcije sa slike 15. Takođe, broj *propuštenih detekcija* je u ovom slučaju smanjen.

6.3 Zaštita od preobučavanja - rano zaustavljanje

Tokom treniranja neuralne mreže, greška na trening setu bi trebala da opada iz epohe u epohu, dok greška na validacionom skupu će opadati sve do nekog trenutka kada počinje da raste i u tom trenutku se čuvaju trenutne težine i dolazi do preobučavanja. Imajući ovo u vidu, kao i činjenicu da simulacija traje duže ukoliko je neuralna mreža kompleksnija, koristimo tehniku **ranog zaustavljanja**, čije rezultate prilažemo u nastavku. Nakon **16** epoha, program se zaustavio na početku 17. epohe, a u tom trenutku tačnost dostiže najveću vrednost od **98.222%**, ali i veći broj *propuštenih detekcija* u odnosu na tehniku **regularizacije**.



Slika 17: Grafički prikaz konfuzione matrice i kriterijumske funkcije na trening i validacionom skupu za neuralnu mrežu sa **dva** skrivena sloja sa **1000** i **1000** neurona nakon uvođenja **ranog zaustavljanja** (tačnost **98.222%**)

7 Programski kod u programskom jeziku *Python*

```
1  import numpy as np
2  import pandas as pd
3  import seaborn as sb
4  import matplotlib.pyplot as plt
5  from sklearn.model_selection import GridSearchCV
6  from keras.models import Sequential
7  from keras.layers import Dense
8  from keras.callbacks import EarlyStopping
9  from keras.wrappers.scikit_learn import KerasClassifier
10 from sklearn.model_selection import train_test_split
11 from keras.regularizers import l2
12
13 np.set_printoptions(precision = 4, suppress=True)
14
15 data = pd.read_csv(r"C:\Users\aleks\OneDrive\Documents\ETF\03 - treca godina\sesti ser
16
17 data.info(verbose = True)
18 #c data.head()
19
20 #kodovanje 'Yes' u 1 i 'No' u 0
21 data = data.replace(to_replace = 'Yes',value = 1)
22 data = data.replace(to_replace = 'No',value = 0)
23
24 data.info(verbose = True)
25 #c data.describe()
26 #c data.isnull().any() - provera da li je ostala neka null vrednost
27
28 data.hist(figsize=(20,15))
29 plt.show()
30
31 #vidimo da su dva atributa skroz neinformativna (Wearing Masks)
32 #i (Sanitization from Market) posto imaju uvek istu vrednost, pa cemo ih odmah
33 #izbaciti
34
35 covid = data.copy(deep = True)
36
37 covid.pop('Wearing Masks')
38 covid.pop('Sanitization from Market')
39
40 #ostala neinformativna obelezja cemo izbaciti tako sto cemo posmatrati
41 #korelaciju (Pearson-ov metod) svakog obelezja sa poslednjom kolonom iz dataseta
42 #koja nam oznacava da li je osoba pozitivna ili negativna i zadrzacemo onih 10
43 #obelezja sa najvecim koeficijentom korelacije
44
45 #%% korelacija svih obelezja
46 pearson_R = covid.corr(method='pearson') #neophodno da prvo odredimo korelaciju
47 #preostalih obelezja, zatim da te vrednosti sortiramo, a onda da 10 obelezja sa
48 #najvecom korelacijom sacuvamo, a ostale odbacimo
49
50 correlation = pd.Series(pearson_R.iloc[:,-1])
51 correlation = correlation.sort_values(ascending=True)
52 print(correlation)
53
54 #izbacivanje obelezja sa najmanjom korelacijom
55
56 for i in range(0,8):
```

```
57      covid.pop(correlation.index[i])
58
59  covid_copy = covid.copy(deep = True)
60  covid_copy.pop('COVID-19')
61
62  pearson_R = covid_copy.corr(method='pearson')
63  plt.figure()
64  sb.heatmap(pearson_R, annot=True) #prikaz 10 obelezja
65  plt.show()
66
67  spearman_R = covid_copy.corr(method='spearman')
68  plt.figure()
69  sb.heatmap(spearman_R, annot=True) #prikaz 10 obelezja
70  plt.show()
71
72  #%% Information Gain
73
74  def calculateInfoD(col):
75      un = np.unique(col)
76      infoD = 0
77      for u in un:
78          p = sum(col == u)/len(col)
79          infoD -= p*np.log2(p)
80      return infoD
81
82  klasa = covid.iloc[:,-1]
83
84  infoD = calculateInfoD(klasa)
85  print('Info(D) = ' + str(infoD))
86
87  feature_names = covid.columns
88
89  IG = np.zeros((covid.shape[1]-1, 2))
90  for ob in range(covid.shape[1]-1):
91      f = np.unique(covid.iloc[:, ob])
92
93      infoDA = 0
94      for i in f:
95          temp = klasa[covid.iloc[:, ob] == i]
96
97          infoDi = calculateInfoD(temp)
98          Di = sum(covid.iloc[:, ob] == i)
99          D = len(covid.iloc[:, ob])
100
101         infoDA += Di*infoDi/D
102
103     IG[ob, 0] = ob+1
104     IG[ob, 1] = infoD - infoDA
105
106     print(str(feature_names[ob]) + ' IG = ' + str(IG[ob,1]))
107     print('------')
108
109 print('IG = \n' + str(IG))
110 IGsorted = IG[IG[:, 1].argsort()]
111 print('Sortirano IG = \n' + str(IGsorted))
112
113 #%% LDA
114
```

```
115 sb.set(style = "darkgrid")
116
117 X_pom = covid.iloc[:,:-1]
118 X = X_pom - np.mean(X_pom, axis = 0)
119 X /= np.max(X, axis = 0)
120
121 y = covid.iloc[:,-1]
122
123 X0 = X.loc[y==0,:]
124 X1 = X.loc[y==1,:]
125 M0 = X0.mean().values.reshape(X0.shape[1],1)
126 M1 = X1.mean().values.reshape(X1.shape[1],1)
127 Sx0 = X0.cov()
128 Sx1 = X1.cov()
129 p0 = X0.shape[0]/X.shape[0]
130 p1 = X1.shape[0]/X.shape[0]
131
132 Sw = p0*Sx0 + p1*Sx1
133 M = p0*M0 + p1*M1
134 Sb = p0*(M0-M)@(M0-M).T + p1*(M1-M)@(M1-M).T
135 Sm = Sw + Sb
136
137 S1 = Sw
138 S2 = Sb
139
140 S = np.linalg.inv(S1)@S2
141 [eigval,eigvec] = np.linalg.eig(S)
142
143 ind = np.argsort(eigval)[::-1]
144 eigval_sort = eigval[ind]
145 eigvec_sort = eigvec[:,ind]
146
147 #%% LDA - 2 dimenzije
148
149 A = eigvec_sort[:,:2]
150 Y = A.T@X.T
151
152 data_LDA_2 = pd.DataFrame(data = Y.T)
153 data_LDA_2 = pd.concat([data_LDA_2,y],axis = 1)
154
155 data_LDA_2.columns = ['LDA1', 'LDA2', 'COVID19']
156
157 plt.figure()
158 sb.scatterplot(data = data_LDA_2, x = 'LDA1', y = 'LDA2' , hue = 'COVID19')
159 plt.show()
160
161 #%% LDA - 3 dimenzije
162
163 A = eigvec_sort[:,:3]
164 Y = A.T@X.T
165
166 data_LDA_3 = pd.DataFrame(data = Y.T)
167 data_LDA_3 = pd.concat([data_LDA_3,y],axis = 1)
168
169 data_LDA_3.columns = ['LDA1', 'LDA2', 'LDA3', 'COVID19']
170
171 fig = plt.figure()
172 ax = fig.add_subplot(111, projection='3d')
```

```
173 color_map = plt.get_cmap('spring')
174 ax.scatter3D(data_LDA_3.LDA1, data_LDA_3.LDA2, data_LDA_3.LDA3,
175                c=data_LDA_3.COVID19, cmap = color_map)
176 ax.legend()
177 plt.show()
178
179 #%% LDA - 1 dimenzija
180
181 A = eigvec_sort[:,:1]
182 Y = A.T@X.T
183
184 data_LDA_1 = pd.DataFrame(data = Y.T)
185 data_LDA_1 = pd.concat([data_LDA_1,y],axis = 1)
186
187 data_LDA_1.columns = ['LDA1', 'COVID19']
188
189 plt.figure()
190 sb.scatterplot(data = data_LDA_1, x = 'LDA1', y = 0 , hue = 'COVID19')
191 plt.show()
192
193 #%% PCA - 2 dimenzije
194
195 Sx = np.cov(X.T)
196
197 [eigval, eigvec] = np.linalg.eig(Sx)
198 ind = np.argsort(eigval)[::-1]
199 eigval = eigval[ind]
200 eigvec = eigvec[:, ind]
201
202 A = eigvec[:, :2]
203 Y = A.T @ X.T
204 Y = Y.T
205 PCA_2 = pd.concat([Y , y] , axis = 1)
206 PCA_2.columns = ['PCA1', 'PCA2', 'COVID19']
207
208 plt.figure()
209 sb.scatterplot(data = PCA_2, x = 'PCA1',y = 'PCA2' , hue = 'COVID19')
210 plt.show()
211
212 #%% Klasifikacija
213
214 data_xy = data_LDA_2.iloc[:,:-1]
215 result = data_LDA_2.iloc[:,-1] # COVID-19 DA/NE
216 K1 = data_xy.loc[result == 0, :] #prva klasa -> zdravi
217 K2 = data_xy.loc[result == 1, :] #druga klasa -> bolesni
218 N1 = len(K1)
219 N2 = len(K2)
220
221 x11_ = np.array(K1.LDA1)
222 x12_ = np.array(K1.LDA2)
223 x21_ = np.array(K2.LDA1)
224 x22_ = np.array(K2.LDA2)
225
226 N1_training = int(0.7*N1)
227 N2_training = int(0.7*N2)
228 N1_test = N1 - N1_training
229 N2_test = N2 - N2_training
230
```

```
231 #%% obucavanje/treniranje
232
233 x11 = np.zeros((N1_training,1))
234 x12 = np.zeros((N1_training,1))
235 x21 = np.zeros((N2_training,1))
236 x22 = np.zeros((N2_training,1))
237
238 for i in range(0,N1_training):
239     x11[i] = x11_[i].real
240     x12[i] = x12_[i].real
241
242 for i in range(0,N2_training):
243     x21[i] = x21_[i].real
244     x22[i] = x22_[i].real
245
246 Z1 = np.concatenate((-x11**2, -x11*x12, -x12**2, -x11, -x12,
247                      -np.ones((N1_training, 1))), axis=1)
248 Z2 = np.concatenate((x21**2, x21*x22, x22**2, x21, x22,
249                      np.ones((N2_training, 1))), axis=1)
250 U = np.concatenate((Z1, Z2), axis=0).T
251
252 #Gama = np.ones(((N1_training+N2_training), 1))
253 Gama = np.append(1*np.ones((N1_training, 1)),
254                  1*np.ones((N2_training, 1)), axis=0)
255
256 W = np.linalg.inv(U@U.T)@U@Gama
257
258 V0 = W[-1]
259 V = np.array([W[3], W[4]])
260 Q = np.array([[W[0], W[1]], [W[1], W[2]]])
261
262 xrange = np.linspace(-6.0, 6.0, 100)
263 yrange = np.linspace(-10.0, 10.0, 100)
264 x,y = np.meshgrid(xrange,yrange)
265
266 equation = V0 + V[0]*x + Q[0,0]*x**2 + 2*Q[0,1]*x*y + Q[1,1]*y**2
267
268 plt.figure()
269 plt.plot(x11, x12, '.', label = 'zdravi')
270 plt.plot(x21, x22, '.', label = 'bolesni')
271 plt.legend()
272 plt.contour(x,y,equation,[0])
273 plt.show()
274
275 #%% testiranje
276
277 x11_t = np.zeros((N1_test,1))
278 x12_t = np.zeros((N1_test,1))
279 x21_t = np.zeros((N2_test,1))
280 x22_t = np.zeros((N2_test,1))
281
282 for i in range(0,N1_test):
283     x11_t[i] = x11_[N1_training + i].real
284     x12_t[i] = x12_[N1_training + i].real
285
286 for i in range(0,N2_test):
287     x21_t[i] = x21_[N2_training + i].real
288     x22_t[i] = x22_[N2_training + i].real
```

```
289
290 decision = np.zeros(((N1_test + N2_test), 1))
291 conf_mat = np.zeros((2,2))
292
293 for i in range(N1_test):
294     h = V0 + V[0]*x11_t[i] + Q[0,0]*x11_t[i]**2 + 2*Q[0,1]*x11_t[i]*x12_t[i]
295     + Q[1,1]*x12_t[i]**2
296     if h < 0:
297         decision[i] = 0
298     else:
299         decision[i] = 1
300
301 for i in range(N2_test):
302     h = V0 + V[0]*x21_t[i] + Q[0,0]*x21_t[i]**2 + 2*Q[0,1]*x21_t[i]*x22_t[i]
303     + Q[1,1]*x22_t[i]**2
304     if h < 0:
305         decision[N1_test + i] = 0
306     else:
307         decision[N1_test + i] = 1
308
309 #%% konfuziona matrica
310
311 #Xtest = np.append(X1_test, X2_test, axis=0)
312 #Ytest = np.append(np.zeros((400, 1)), np.ones((400, 1)))
313 Ytest = np.append(np.zeros((N1_test,1)),np.ones((N2_test,1)), axis = 0)
314
315 from sklearn.metrics import confusion_matrix
316 conf_mat = confusion_matrix(Ytest, decision)
317
318 plt.figure()
319 sb.heatmap(conf_mat, annot=True, fmt='g', cbar=False)
320 plt.show()
321
322 acc = np.trace(conf_mat)/np.sum(conf_mat)*100
323 print('Tacnost klasifikatora iznosi:' + str(acc) + '%')
324
325 #%% Neuralne mreze
326
327 X = data.drop('COVID-19',axis = 1)
328 Y = data.iloc[:,-1]
329
330 X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
331                                                     train_size = 0.7,
332                                                     random_state = 2)
333
334 #%% jedan skriveni sloj sa 20 neurona
335 model = Sequential()
336 model.add(Dense(20, input_dim = 20, activation = 'relu'))
337 model.add(Dense(1,activation = 'sigmoid'))
338
339 model.compile(loss = 'binary_crossentropy',
340               optimizer = 'adam',metrics = ['accuracy'])
341
342 history = model.fit(X_train,Y_train,validation_data = (X_test, Y_test),
343                     epochs = 100, verbose = 0)
344
345 _, train_acc = model.evaluate(X_train,Y_train, verbose = 0)
346 print('Train acc = ' + str(train_acc*100) + '%')
```

```
347
348 _, test_acc = model.evaluate(X_test,Y_test, verbose = 0)
349 print('Test acc = ' + str(test_acc*100) + '%')
350
351 Y_pom = model.predict(X_test)
352 Y_pred = 1*(Y_pom > 0.5)
353 conf_mat = confusion_matrix(Y_test, Y_pred)
354 sb.heatmap(conf_mat, annot=True, fmt='g', cbar=False)
355 plt.show()
356
357 plt.figure()
358 plt.plot(history.history['loss'])
359 plt.plot(history.history['val_loss'])
360 plt.legend(["Trening skup", "Validacioni skup"])
361 plt.title('Kriterijumska funkcija')
362 plt.show()
363
364 #%% jedan skriveni sloj sa 10 neurona
365 model = Sequential()
366 model.add(Dense(10, input_dim = 20, activation = 'relu'))
367 model.add(Dense(1,activation = 'sigmoid'))
368
369 model.compile(loss = 'binary_crossentropy',
370               optimizer = 'adam',metrics = ['accuracy'])
371
372 history = model.fit(X_train,Y_train,validation_data = (X_test, Y_test),
373                     epochs = 100, verbose = 0)
374
375 _, train_acc = model.evaluate(X_train,Y_train, verbose = 0)
376 print('Train acc = ' + str(train_acc*100) + '%')
377
378 _, test_acc = model.evaluate(X_test,Y_test, verbose = 0)
379 print('Test acc = ' + str(test_acc*100) + '%')
380
381 Y_pom = model.predict(X_test)
382 Y_pred = 1*(Y_pom > 0.5)
383 conf_mat = confusion_matrix(Y_test, Y_pred)
384 sb.heatmap(conf_mat, annot=True, fmt='g', cbar=False)
385 plt.show()
386
387 plt.figure()
388 plt.plot(history.history['loss'])
389 plt.plot(history.history['val_loss'])
390 plt.legend(["Trening skup", "Validacioni skup"])
391 plt.title('Kriterijumska funkcija')
392 plt.show()
393
394 #%% jedan skriveni sloj sa 3 neurona
395 model = Sequential()
396 model.add(Dense(3, input_dim = 20, activation = 'relu'))
397 model.add(Dense(1,activation = 'sigmoid'))
398
399 model.compile(loss = 'binary_crossentropy',
400               optimizer = 'adam',metrics = ['accuracy'])
401
402 history = model.fit(X_train,Y_train,validation_data = (X_test, Y_test),
403                     epochs = 100, verbose = 0)
404
```

```
405 _, train_acc = model.evaluate(X_train,Y_train, verbose = 0)
406 print('Train acc = ' + str(train_acc*100) + '%')
407
408 _, test_acc = model.evaluate(X_test,Y_test, verbose = 0)
409 print('Test acc = ' + str(test_acc*100) + '%')
410
411 Y_pom = model.predict(X_test)
412 Y_pred = 1*(Y_pom > 0.5)
413 conf_mat = confusion_matrix(Y_test, Y_pred)
414 sb.heatmap(conf_mat, annot=True, fmt='g', cbar=False)
415 plt.show()
416
417 plt.figure()
418 plt.plot(history.history['loss'])
419 plt.plot(history.history['val_loss'])
420 plt.legend(["Trening skup", "Validacioni skup"])
421 plt.title('Kriterijumska funkcija')
422 plt.show()
423
424 #%% jedan skriveni sloj sa 500 neurona
425 model = Sequential()
426 model.add(Dense(500, input_dim = 20, activation = 'relu'))
427 model.add(Dense(1,activation = 'sigmoid'))
428
429 model.compile(loss = 'binary_crossentropy',
430               optimizer = 'adam',metrics = ['accuracy'])
431
432 history = model.fit(X_train,Y_train,validation_data = (X_test, Y_test),
433                     epochs = 100, verbose = 0)
434
435 _, train_acc = model.evaluate(X_train,Y_train, verbose = 0)
436 print('Train acc = ' + str(train_acc*100) + '%')
437
438 _, test_acc = model.evaluate(X_test,Y_test, verbose = 0)
439 print('Test acc = ' + str(test_acc*100) + '%')
440
441 Y_pom = model.predict(X_test)
442 Y_pred = 1*(Y_pom > 0.5)
443 conf_mat = confusion_matrix(Y_test, Y_pred)
444 sb.heatmap(conf_mat, annot=True, fmt='g', cbar=False)
445 plt.show()
446
447 plt.figure()
448 plt.plot(history.history['loss'])
449 plt.plot(history.history['val_loss'])
450 plt.legend(["Trening skup", "Validacioni skup"])
451 plt.title('Kriterijumska funkcija')
452 plt.show()
453
454 #%% dva skrivena sloja sa 20 i 15 neurona
455
456 model = Sequential()
457 model.add(Dense(20, input_dim = 20, activation = 'relu'))
458 model.add(Dense(15, activation = 'relu'))
459 model.add(Dense(1, activation = 'sigmoid'))
460
461 model.compile(loss = 'binary_crossentropy',
462               optimizer = 'adam',metrics = ['accuracy'])
```

```
463
464 history = model.fit(X_train,Y_train,validation_data = (X_test, Y_test),
465                      epochs = 100, verbose = 0)
466
467 _, train_acc = model.evaluate(X_train,Y_train, verbose = 0)
468 print('Train acc = ' + str(train_acc*100) + '%')
469
470 _, test_acc = model.evaluate(X_test,Y_test, verbose = 0)
471 print('Test acc = ' + str(test_acc*100) + '%')
472
473 Y_pom = model.predict(X_test)
474 Y_pred = 1*(Y_pom > 0.5)
475 conf_mat = confusion_matrix(Y_test, Y_pred)
476 sb.heatmap(conf_mat, annot=True, fmt='g', cbar=False)
477 plt.show()
478
479 plt.figure()
480 plt.plot(history.history['loss'])
481 plt.plot(history.history['val_loss'])
482 plt.legend(["Trening skup", "Validacioni skup"])
483 plt.title('Kriterijumska funkcija')
484 plt.show()
485
486 #%% dva skrivena sloja sa 20 i 10 neurona
487
488 model = Sequential()
489 model.add(Dense(20, input_dim = 20, activation = 'relu'))
490 model.add(Dense(10, activation = 'relu'))
491 model.add(Dense(1, activation = 'sigmoid'))
492
493 model.compile(loss = 'binary_crossentropy',
494               optimizer = 'adam',metrics = ['accuracy'])
495
496 history = model.fit(X_train,Y_train,validation_data = (X_test, Y_test),
497                      epochs = 100, verbose = 0)
498
499 _, train_acc = model.evaluate(X_train,Y_train, verbose = 0)
500 print('Train acc = ' + str(train_acc*100) + '%')
501
502 _, test_acc = model.evaluate(X_test,Y_test, verbose = 0)
503 print('Test acc = ' + str(test_acc*100) + '%')
504
505 Y_pom = model.predict(X_test)
506 Y_pred = 1*(Y_pom > 0.5)
507 conf_mat = confusion_matrix(Y_test, Y_pred)
508 sb.heatmap(conf_mat, annot=True, fmt='g', cbar=False)
509 plt.show()
510
511 plt.figure()
512 plt.plot(history.history['loss'])
513 plt.plot(history.history['val_loss'])
514 plt.legend(["Trening skup", "Validacioni skup"])
515 plt.title('Kriterijumska funkcija')
516 plt.show()
517
518 #%% dva skrivena sloja sa 20 i 5 neurona
519
520 model = Sequential()
```

```
521 model.add(Dense(20, input_dim = 20, activation = 'relu'))
522 model.add(Dense(5, activation = 'relu'))
523 model.add(Dense(1, activation = 'sigmoid'))
524
525 model.compile(loss = 'binary_crossentropy',
526               optimizer = 'adam',metrics = ['accuracy'])
527
528 history = model.fit(X_train,Y_train,validation_data = (X_test, Y_test),
529                     epochs = 100, verbose = 0)
530
531 _, train_acc = model.evaluate(X_train,Y_train, verbose = 0)
532 print('Train acc = ' + str(train_acc*100) + '%')
533
534 _, test_acc = model.evaluate(X_test,Y_test, verbose = 0)
535 print('Test acc = ' + str(test_acc*100) + '%')
536
537 Y_pom = model.predict(X_test)
538 Y_pred = 1*(Y_pom > 0.5)
539 conf_mat = confusion_matrix(Y_test, Y_pred)
540 sb.heatmap(conf_mat, annot=True, fmt='g', cbar=False)
541 plt.show()
542
543 plt.figure()
544 plt.plot(history.history['loss'])
545 plt.plot(history.history['val_loss'])
546 plt.legend(["Trening skup", "Validacioni skup"])
547 plt.title('Kriterijumska funkcija')
548 plt.show()
549
550 #%% dva skrivena sloja sa 10 i 3 neurona
551
552 model = Sequential()
553 model.add(Dense(10, input_dim = 20, activation = 'relu'))
554 model.add(Dense(3, activation = 'relu'))
555 model.add(Dense(1, activation = 'sigmoid'))
556
557 model.compile(loss = 'binary_crossentropy',
558               optimizer = 'adam',metrics = ['accuracy'])
559
560 history = model.fit(X_train,Y_train,validation_data = (X_test, Y_test),
561                     epochs = 100, verbose = 0)
562
563 _, train_acc = model.evaluate(X_train,Y_train, verbose = 0)
564 print('Train acc = ' + str(train_acc*100) + '%')
565
566 _, test_acc = model.evaluate(X_test,Y_test, verbose = 0)
567 print('Test acc = ' + str(test_acc*100) + '%')
568
569 Y_pom = model.predict(X_test)
570 Y_pred = 1*(Y_pom > 0.5)
571 conf_mat = confusion_matrix(Y_test, Y_pred)
572 sb.heatmap(conf_mat, annot=True, fmt='g', cbar=False)
573 plt.show()
574
575 plt.figure()
576 plt.plot(history.history['loss'])
577 plt.plot(history.history['val_loss'])
578 plt.legend(["Trening skup", "Validacioni skup"])
```

```
579 plt.title('Kriterijumska funkcija')
580 plt.show()
581
582 #%% dva skrivena sloja sa 1000 i 1000 neurona
583
584 model = Sequential()
585 model.add(Dense(1000, input_dim = 20, activation = 'relu'))
586 model.add(Dense(1000, activation = 'relu'))
587 model.add(Dense(1, activation = 'sigmoid'))
588
589 model.compile(loss = 'binary_crossentropy',
590               optimizer = 'adam',metrics = ['accuracy'])
591
592 history = model.fit(X_train,Y_train,validation_data = (X_test, Y_test),
593                     epochs = 100, verbose = 0)
594
595 _, train_acc = model.evaluate(X_train,Y_train, verbose = 0)
596 print('Train acc = ' + str(train_acc*100) + '%')
597
598 _, test_acc = model.evaluate(X_test,Y_test, verbose = 0)
599 print('Test acc = ' + str(test_acc*100) + '%')
600
601 Y_pom = model.predict(X_test)
602 Y_pred = 1*(Y_pom > 0.5)
603 conf_mat = confusion_matrix(Y_test, Y_pred)
604 sb.heatmap(conf_mat, annot=True, fmt='g', cbar=False)
605 plt.show()
606
607 plt.figure()
608 plt.plot(history.history['loss'])
609 plt.plot(history.history['val_loss'])
610 plt.legend(["Trening skup", "Validacioni skup"])
611 plt.title('Kriterijumska funkcija')
612 plt.show()
613
614 #%% zastita od preobucavanja - regularizacija
615
616 model = Sequential()
617 model.add(Dense(1000, input_dim = 20, activation = 'relu',
618                 kernel_regularizer=l2(0.0001)))
619 model.add(Dense(1000, activation = 'relu',
620                 kernel_regularizer=l2(0.0001)))
621 model.add(Dense(1, activation = 'sigmoid'))
622
623 model.compile(loss = 'binary_crossentropy',
624               optimizer = 'adam', metrics = ['accuracy'])
625
626 history = model.fit(X_train, Y_train, validation_data = (X_test, Y_test),
627                     epochs = 100, verbose = 0)
628
629 _, train_acc = model.evaluate(X_train,Y_train, verbose = 0)
630 print('Train acc = ' + str(train_acc*100) + '%')
631
632 _, test_acc = model.evaluate(X_test,Y_test, verbose = 0)
633 print('Test acc = ' + str(test_acc*100) + '%')
634
635 plt.figure()
636 Y_pom = model.predict(X_test)
```

```
637 Y_pred = 1*(Y_pom > 0.5)
638 conf_mat = confusion_matrix(Y_test, Y_pred)
639 sb.heatmap(conf_mat, annot=True, fmt='g', cbar=False)
640 plt.show()
641
642 plt.figure()
643 plt.plot(history.history['loss'])
644 plt.plot(history.history['val_loss'])
645 plt.legend(["Trening skup", "Validacioni skup"])
646 plt.title('Kriterijumska funkcija')
647 plt.show()
648
649 #%% zastita od preobucavanja - rano zaustavljanje
650
651 model = Sequential()
652 model.add(Dense(1000, input_dim = 20, activation = 'relu'))
653 model.add(Dense(1000, activation = 'relu'))
654 model.add(Dense(1, activation = 'sigmoid'))
655
656 model.compile(loss = 'binary_crossentropy',
657               optimizer = 'adam', metrics = ['accuracy'])
658
659 es = EarlyStopping(monitor='val_loss', mode='min', patience=5, verbose=1)
660
661 history = model.fit(X_train, Y_train, validation_data = (X_test, Y_test),
662                     callbacks=[es], epochs = 200, verbose = 1)
663
664 _, train_acc = model.evaluate(X_train,Y_train, verbose = 0)
665 print('Train acc = ' + str(train_acc*100) + '%')
666
667 _, test_acc = model.evaluate(X_test,Y_test, verbose = 0)
668 print('Test acc = ' + str(test_acc*100) + '%')
669
670 plt.figure()
671 Y_pom = model.predict(X_test)
672 Y_pred = 1*(Y_pom > 0.5)
673 conf_mat = confusion_matrix(Y_test, Y_pred)
674 sb.heatmap(conf_mat, annot=True, fmt='g', cbar=False)
675 plt.show()
676
677 plt.figure()
678 plt.plot(history.history['loss'])
679 plt.plot(history.history['val_loss'])
680 plt.legend(["Trening skup", "Validacioni skup"])
681 plt.title('Kriterijumska funkcija')
682 plt.show()
```