Klasifikacija HCC ćelija

Seminarski rad u okviru kursa Istraživanje podataka 2
Matematički fakultet

Jovana Nikolić

jovananiki7@gmail.com

2. septembar 2019

Sadržaj

1 Uvod

Istraživanje podataka je sve prisutnija oblast u raznim naukama. Usled velikog priliva podataka neophodno je bilo razviti algoritme koji u prihvatljivom vremenu obrađuje date podatke. Samo neke od oblasti gde se primenjuje su medicina, biologija, statistika, veštačka inteligencija...[5]

U ovom seminarskom radu biće predstavljen detaljan opis postupka klasifikacije ćelija i prikazani rezultati iste. Pre pocetka i primene algoritama potrebno je upoznati se samim podacima sto je opisano u prvom delu. Tema predprocesiranja je opisana u narednom poglavlju, dok su algoritmi klasifikacije i rezultati glavna tema ovog rada. Na samom kraju izvršena je analiza dobijenih rezultata. Za sve primenjene postupke koričćen je programski jezik Python3. Uz standardnu biblioteku koju nudi Python upotrebljivane su još i biblioteke panda, numpy, sklearn i matplotlib (Listing 1)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

Listing 1: Import biblioteka

2 Skup podataka

Podaci koji se klasifikuju su dobijeni iz perifernih mononuklearnih krvnih ćelija (Peripheral blood mononuclear cells, PBMCs) PBMC tipa HCC. Ove celije se koriste u istraživanju u različitim oblastima biomedicine, poput infektivnih bolesti, imunologije, maligniteta, razvoja vakcina i slično. Glavna funkcija PBMC ćelija je imuna odbrana organizma. Svaki tip ćelije ima karakteristicne obrasce ('mustre') proteina i gena koje ih međusobno razlikuju i mogu da se koriste za podelu prema njihovom tipu. [6]

Sama struktura podaka nad kojima se vrše analize se sastoje od tri zasebna .csv fajla. Svaki fajl predstavlja matricu podataka gde prvi red sadrži redni broj ćelije koja je ispitivana, a prva kolona identifikaciju gena. Vrednosti u matrici sadrže broj transkripta gena u ćeliji. Takodje

svaka datoteka predstavlja jednu klasu. Na Slici 1 dat je isečak iz date matrice zbog sticanja uvida o izgledu samih fajlova.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	hg38_A1BG	0	0	0	3	1	0	2	0	0	0	0	0	0	0	0	0
3	hg38_A1BG-AS1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	hg38_A1CF	2	4	0	2	3	3	1	0	5	1	2	3	1	3	1	2
5	hg38_A2M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	hg38_A2M-AS1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	hg38_A2ML1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	hg38_A2ML1-AS1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	hg38_A2ML1-AS2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	hg38_A3GALT2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	hg38_A4GALT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	hg38_A4GNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	hg38_AAAS	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1
14	hg38_AACS	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
15	hg38_AADAC	0	2	0	4	0	0	0	1	1	0	0	0	0	0	1	4

Slika 1: Fajl za obradu

Radi lakšeg snalaženja dati fajlove imenujemo sa HCC01.csv, HCC02.csv i HCC03.csv. U Tabeli 1 mogu se videti poređenja veličine i broja kolona sva tri fajla, dok svi fajlovi imaju 31221 red.

Tabela 1: Poređenja velicina fajlova

naziv	broj kolona	velicina
HCC01.csv	895	56.5MB
HCC02.csv	2083	131.1MB
HCC03.csv	684	42.1MB

Na samom početku učitavamo podatke korišćenjem funkcije pd.read_csv. Kada su podaci učitani mozemo započeti analizu, obradu i klasifikaciju datih podataka što sledi u nastavku.

3 Predprocesiranje

Nakon što smo se upoznali sa podacima i učitali date fajlove u ovom poglavlju opisaćemo procees predprocesiranja. Struktura podataka je

takva da postoji primetno veliki broj redova koji sadrže isključivo nule, ili samo jednu vrednost različitu od nule. Takvi redovi nemaju značaj za klasifikaciju tako da ćemo ih eliminisati. Programski kod kojim smo ovo postigli je prikazan u nastavku (Listing 2).

```
tabela1.drop(tabela1.columns[[0]], axis=1, inplace=True)
tabela10 = tabela1[(tabela1.T != 0).any()]
i=1
j=0
brojac=0
for i in range(1, redovi):
  if brojac==1:
    tabela10.drop(tabela10.index[i])
  for j in range(0,kolone):
    if tabela10.iloc[i,j]!=0:
      brojac=brojac+1
    if brojac>1:
      brojac=0
      break
redovi = tabela10.shape[0]
kolone = tabela10.shape[1]
print(redovi,kolone)
```

Listing 2: Redukcija redova

Radi smanjenja vremena izvršavanja, nad svakim fajlom posebno je primenjen ovaj postupak i u nastavku će biti učitani novonastali fajlovi. U Tabeli 2 prikazana je odnos broja redova pre i posle uklanjanja nepotrebnih instanci.

Tabela 2: Broj redova pre i nakon uklanjanja 0 redova

naziv	ukupan broj redova	broj redova bez 0 redova
HCC01.csv	31221	15584
HCC02.csv	31221	18093
HCC03.csv	31221	14085

Sledeći korak pre same klasifikacije jeste spajanje tabela. Međutim, želimo da imamo i vizuelizaciju podataka za svaku klasu. Da bi smo to izvrsili potrebno je normalizovati podatke. Nakon toga izvršen je PCA(Principal component analysis) algoritam kojima je smanjena dimenzionalnost matrice na 3 dimenzije da bi mogli prikazati podatke na 3D dijagramu. Ovi koraci prikazani su u Listingu 3.

```
x = tabela10.values
standard_scaler = StandardScaler()
x_standardized = standard_scaler.fit_transform(x)
tabela10=pd.DataFrame(x_standardized, index=tabela10.index
    .values)

pca = PCA(3)
x = tabela10.values
x = pca.fit_transform(x)
tabela10=pd.DataFrame(x, index=tabela10.index.values)
tabela10.to_csv("fajl3.csv", encoding='utf-8', index=False
    )
```

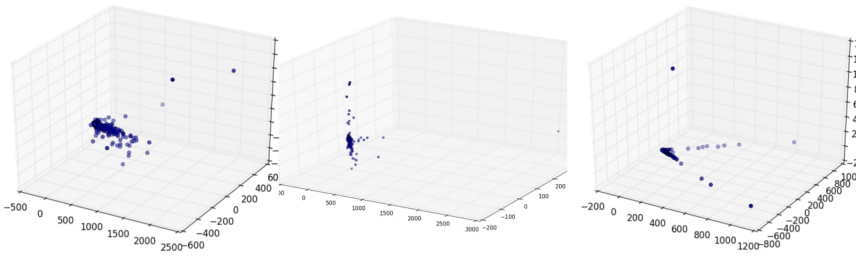
Listing 3: Normalizacija i PCA

Dimenzionom redukciom dobili smo padatke koje je moguće predsaviti na 3D dijagramima (Slika 2) koji su generisani propratnim kodom (Listing 4). Moze se primetiti da su geni iz jedne klase grupisani oko jednog polja dijagrama.

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x[:, 0], x[:, 1], x[:, 2])
plt.show()
```

Listing 4: Vizuelizacija

Konačno prelazimo na spajanje naših fajlova u jedan fajl. Funkcije kojima smo ovo postigli su navedene u Listingu 4. Nakon učitavanja fajlova, svaki fajl je transponovan, uklonjene su 0 kolone i dodata mu je numerička oznaka klase, a zatim je spojen u jedan fajl nad kojim je



Slika 2: 3D plot

vršen proces klasifikacije u nastavku ovog rada. [2][3]

```
0
import pandas as pd
2 import numpy as np
import gc
4
#citanje fajlova
6 fajl1=pd.read_csv("007_HCC_cells_csv.csv", index_col =
    False)
fajl2=pd.read_csv("008_HCC_cells_csv.csv", index_col =
    False)
8 fajl3=pd.read_csv("009_HCC_cells_csv.csv", index_col =
    False)

10 #transponovanje i brisanje prve kolone koja oznacava redni
     broj
fajl1 = fajl1.T
12 kolone1 = fajl1.iloc[0, :]
fajl1 = fajl1.iloc[1:, :]
14 fajl1.columns = kolone1
gc.collect()
16
fajl2 = fajl2.T
18 kolone2 = fajl2.iloc[0, :]
fajl2 = fajl2.iloc[1:, :]
```

```
fajl2.columns = kolone2
gc.collect()

fajl3 = fajl3.T
kolone3 = fajl3.iloc[0, :]
fajl3 = fajl3.iloc[1:, :]
fajl3.columns = kolone3
gc.collect()

#dodeljivanje klase
fajl1['class'] = 0
fajl2['class'] = 1
fajl3['class'] = 2
gc.collect()

#spajanje i brisanje nula kolona
spojeni = pd.concat([fajl1,fajl2,fajl3], axis = 0,
   ignore_index = True)
spojeni.loc[:, (spojeni != 0).any(axis=0)]
spojeni.to_csv('./spojeni.csv', index = False)
```

Listing 5: Spajanje

4 Proces klasifikacije

Nakon što smo podatke sredili i doveli u oblik pogodan za klasifikaciju prelazimo na primenu raznih oblika klasifikacije. Naš cilj je da na osnovu dobijenih podataka utvrdimo zakonitosti i na osnovu njih istreniramo naš model za što bolju klasifikaciju. Biće primenjene metode K najbližih suseda, SVM algoritam, Drveta odlučivanja, Bajesov algoritam i neuronske mreže. U poglavlju koje sledi za svaku vrstu klasifikacije biće menjani parametri i prikazivani rezultati. Pre primene određenog postupka potrebno je da skup podataka podelimo na trening i test skup.Podela je izvršena tako da trening skup sadrži 70% početnih podataka. Zatim su nad trening podacima sprovedeni navedeni algoritmi i dobijeni određeni

zaključi pomoću test skupa. Funkcije kojima smo ovo postigli i zatim kao početni primer primenili K najbližih suseda su prikazane u Listingu 6. [7] [9]

```
0
    #citanje podataka
2   df = pd.read_csv('./tabela.csv')
    #X skup na osnovu koga klasifikujemp
4   X = df.loc[:, df.columns != 'class']
    #y skup vrednosti koji oznacavaju klasu
6   y = df[['class']]

8   #podela na test i trening skup
    X_train, X_test, y_train, y_test = train_test_split(X,
    y, test_size=0.3)
10
    #clf menjamo u zavisnosti od metode koju koristimo
12  #u nastavku rada su navedeni svi pozivi funkcija koje
smo koristili
    clf = KNeighborsClassifier(n_neighbors=5, weights='
distance')
14  clf.fit(X_train, y_train.values.ravel())

16  #vrsimo predikciju
    y_test_predicted = clf.predict(X_test)
18  y_train_predicted = clf.predict(X_train)

20  #racunamo rezultate
    train_acc = clf.score(X_train, y_train)
22  test_acc = clf.score(X_test, y_test)
    train_conf = sklearn.metrics.confusion_matrix(y_train,
    y_train_predicted)
24  test_conf = sklearn.metrics.confusion_matrix(y_test,
y_test_predicted)

26  #stampanje
    print('Preciznost trening skupa: {}'.format(train_acc)
)
```

```
28      print("Matrica konfuzije:\n{}".format(train_conf))
        print('Preciznost test skupa: {}'.format(test_acc))
30      print("Matrica konfuzije:\n{}".format(test_conf))
```

Listing 6: Klasifikacija

5 Rezultati

Rezultate prikazujemo za svaki od algoritama u vidu matrica konfuzije i preciznosti. Ako je preciznost 1 naš model nije dobar jer dolazi do preprilagodjavanja što govori da se model previše oslanja na podatke iz trening skupa. Mala preciznost takođe govori da je model loš. Podrazumevano je da čitalac poseduje predznanje vezano za korišcene metode tako da principi rada algoritama nisu detaljno opisivani. [8] [4]

5.1 K najbližih suseda

Ovaj algoritam se zasniva na nalaženju najbližih suseda u svakoj iteraciji za svaki podatak. Kao parametari zadaje se vrednost k koja govori koliko suseda uzimamo u obzir. Veća vrednost parametra je pogodnija za rad sa šumovima. Kao mera rastojanja najčesće se koristi Euklidsko rastojanje ali se mogu koristiti i druge metrike za zadavanje težine čvorova poput uniforme težine. Parametri se zadaju u pozivu funkicje prikazanom u Listingu 7.

```
0
clf =  KNeighborsClassifier(n_neighbors=3, weights='
   uniform')
2 clf =  KNeighborsClassifier(n_neighbors=10, weights='
   uniform')

4 clf =  KNeighborsClassifier(n_neighbors=3, weights='
   distance')
```

```
clf =  KNeighborsClassifier(n_neighbors=10, weights='
   distance')
```

Listing 7: Funkcije KNN

Listing 8 prikazuje izlaze iz programa sa različitim vrednostima parametara koji su menjani zarad dobijanja boljeg modela. Za vrdnost k testirana je vrednost 3 i 10, dok je za metriku birano Euklidsko rastojanje i unifomna dodela težina. Iz rezultata se vidi da uniformna raspodela tezina daje ne tako dobar model sa nedovoljno dobrom preciznosti i za trening i za test skup, gde je npr za drugu klasu pogrešno klasifikovao skoro petinu instanci. Ako se posmatra distanca situacija se menja i za trening skup dolazi do dobrog klasifikovanja celog skupa dok je za test skup preciznost jako blizu jedinici. Sa porastom vrednosti k dobijamo lošije modele za uniformnu raspodelu dok se za rastojanje situacija znatno ne menja.

```

**knn 3 uniform**
Preciznost trening skupa: 0.88259
Matrica konfuzije:
[[ 613    4    9]
 [   4 1229    225]
 [   3    65  411]]

Preciznost test skupa: 0.81908
Matrica konfuzije:
[[232   12   25]
 [  5 520   100]
 [  5   62 138]]

**knn 10 uniform**
Preciznost trening skupa:  0.84294
Matrica konfuzije:
[[ 599    7    20]
 [   8 1153    297]
 [   3    75  401]]
```

```
20
Preciznost test skupa: : 0.80086
22 Matrica konfuzije:
[[212   15   42]
24  [  8 514   103]
 [  8   44 153]]
26
**knn 3 distance**
28 Preciznost trening skupa: 1.0
Matrica konfuzije:
30 [[ 626    0    0]
 [   0 1457    0]
32  [   0    0  471]]

34 Preciznost test skupa: 0.9990
Matrica konfuzije:
36 [[268   0   0]
 [  1 624   0]
38  [  0   0 202]]

40 **knn 10 distance**
Preciznost trening skupa: 1.0
42 Matrica konfuzije:
[[ 626    0    0]
44  [   0 1457    0]
 [   0    0  471]]
46
Preciznost test skupa: 0.9972
48 Matrica konfuzije:
[[266   0   2]
50  [  1 624   0]
 [  0   0 202]]
```

Listing 8: Rezultati KNN

5.2 SVM

SVM ili Support vector machines je metoda potpornih vektora koja se koristi u raznim oblastima klasifikacije podataka. Metod radi na principu pronalaženja hiperravni koja deli prostor podataka uzimajući u obzir da podaci moraju biti linearno razdvojivi. Listing 9 prikazuje poziv funkcije kojim smo izvrsili ovu klasifikaciju. Vidimo da se kao parametar zadaje karnel, odnosno vrsta SVM algortma. Moguce vrednosti ovog parametra su 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'.. Mi smo testirali linear, sigmoid i rbf kernel. Kao posebna prednost ovog algortima u literaturi se navodi njegova mala memorijska zahtevnost.

```
0
clf = svm.SVC(kernel = "linear")
2
clf = svm.SVC(kernel = 'sigmoid)
4
clf = svm.SVC(kernel = "rbf")
```

Listing 9: Funkicje SVM

U nastavku Listing 10 prikazuje dobijene rezultate u zavisnosti od parametara. Linearni model se pokazao kao dobar jer je i na trening a i na test skupu pokažao maksimalnu tacnost. Sigmoidni model daje veoma loše rezultate pogotovo za klasifikaciju prve i trece klase sto se moze videti iz matrice konfuzije. Rbf karnel trening skup klasifikuje sa preciznoscu 1, dok preciznost test skupa iznosi 0.820 i vidimo da je druga klasa klasifikovana bez greske a prva i treća su te koje negativno u utiču na preciznost.

```
0
**svm linear**
2 Preciznost trening skupa: 1.0
Matrica konfuzije:
4 [[ 626    0    0]
  [   0 1457    0]
6  [   0    0  471]]
```

```
Preciznost test skupa: 1.0
Matrica konfuzije:
[[268   0   0]
 [  0 625   0]
 [  0   0 202]]

 **svm sigmoid**
Preciznost trening skupa: 0.3962
Matrica konfuzije:
[[ 19 345 262]
 [ 87 966 404]
 [439   5  27]]

Preciznost test skupa: 0.4173
Matrica konfuzije:
[[  8 139 121]
 [ 35 434 156]
 [183   4  15]]

**svm rbf**
Preciznost trening skupa: 1.0
Matrica konfuzije:
[[ 626    0    0]
 [   0 1457    0]
 [   0    0  471]]

Preciznost test skupa: 0.82009
Matrica konfuzije:
[[118 150   0]
 [  0 625   0]
 [  0  47 155]]
```

Listing 10: Rezultati SVM

5.3 Drvo odlučivanja

Ova metoda je zasnovana na skupu jednostanih pravila zaključivanja pomoću kojih se formira stablo na osnovu koga se vrši neparametarska klasifikacija. Sam algoritam koristi razlicite metode poput CART, ID3, C5.0. Logoritamska slozenost u odnosu na broj podataka je glavna prednost ovih metoda. Kao i do sada radi dobijanja sto boljih pozivamo funkcije sa različitim parametrima(Listing 11). Klasifikacija je vršena na osnovu zadatih kriterijuma(Gini indeks ili entropija) i zadate dubine stabla od 5 ili 10.

```

clf =  DecisionTreeClassifier(max_depth=5, criterion = 'entropy')
clf =  DecisionTreeClassifier(max_depth=10, criterion = 'entropy')

clf =  DecisionTreeClassifier(max_depth=5, criterion = 'gini')
clf =  DecisionTreeClassifier(max_depth=10, criterion = 'gini')
```

Listing 11: Funkcije drveta odlučivanja

Ovoga puta dobijeni su sledeci rezultati(Listing 12). Prvo smo kao meru koristili entropiju i dobijen za rezultat sa minimalnom greskom kao i kod primene Gini indeksa gde su dobijeni slični rezultati koji za test skup daju jako dobre rezulatate bez obzira koji kriterijum koristimo.

```
**tree 5 entropy**
Preciznost trening skupa: 1.0
Matrica konfuzije:
[[ 626    0    0]
 [   0 1457    0]
 [   0    0  471]]
Preciznost test skupa: 0.9945205479452055
Matrica konfuzije:
[[267   1   0]
```

```
 [  2 621   2]
 [  0   1 201]]

**tree 10 entropy**
Preciznost trening skupa: 1.0
Matrica konfuzije:
[[ 626    0    0]
 [   0 1457    0]
 [   0    0  471]]
Preciznost test skupa: 0.9972602739726028
Matrica konfuzije:
[[268   0   0]
 [  0 623   2]
 [  0   1 201]]

**tree 5 gini**
Preciznost trening skupa: 1.0
Matrica konfuzije:
[[ 626    0    0]
 [   0 1457    0]
 [   0    0  471]]
Preciznost test skupa: 0.993607305936073
Matrica konfuzije:
[[267   1   0]
 [  2 621   2]
 [  0   2 200]]

**tree 10 gini**
Preciznost trening skupa: 1.0
Matrica konfuzije:
[[ 626    0    0]
 [   0 1457    0]
 [   0    0  471]]
Preciznost test skupa: 0.9972602739726028
Matrica konfuzije:
[[268   0   0]
 [  0 623   2]
 [  0   1 201]]
```

Listing 12: Rezultati drveta odlučivanja

5.4 Bajesov algoritam

Ova vrsta klasifikacije se razlikuje od drugih jer se zasniva na verovatnoći događaja. Glavna predpostavka je da su podaci iz uzorka nezavisni. I ovaj algoritam ima vise verzija a u zavisnosti od parametara, mi smo koristili osnovnu verziju prikazanu u Listingu 13. Iako ovaj metod nije deterministički glavna prednost u odnosu na ostale je samo vreme izvršavanja.

```
0
clf = GaussianNB()
```

Listing 13: Funkicje Bajesov algoritam

Model koji je dobijen je prikazan u Listingu 14. Dogodila se situacija da je i trening i test skup klasifikovao bez greške tako da zaključujemo da je ovaj metod jako dobar za klasifikaciju naših podataka. Takođe pošto je ova metoda zasnovana na činjenici da su podaci nezavisni možemo zaključiti da je to slučaj i sa našim skupom.

```
0
Preciznost trening skupa: 1.0
2 Matrica konfuzije:
[[ 626    0    0]
4  [   0 1457    0]
   [   0    0  471]]
6 Preciznost test skupa: 1.0
Matrica konfuzije:
8 [[268   0   0]
   [  0 625   0]
10  [  0   0 202]]
```

Listing 14: Rezultati Bajesov algoritam

5.5 Neuronske mreže

Neuronske mreže su u informatičkom svetu sve popularniji metod za klasifikaciju koji se zasnivaju na nekim potpuno novim principima. Da bi neuronska mreža dobro radila bitno je podesiti mnoge parametre poput broja neurona, dubine skrivenog sloja, aktivacionu funkciju i slicno. Radi lakse upotrebe korisceni su gotovi rešavači(Solveri) i to lbfgs- tehnika zasnovana na kvazi Njutnovim metodama i sgd- konkretno ažuriranje Stohastičkim gradijentnim spustom(Listing 15)

```
0
clf = MLPClassifier(solver = 'lbfgs')
2
clf = MLPClassifier(solver = 'sgd')
```

Listing 15: Funkcije neuronske mreže

Iz rezultata zaključujemo da Njutnova metoda ne daje nikakve značajne rezultate i preciznost je veoma mala. Rezultati Stohastičkog gradijentnog spusta daje totalno suprotne rezultate tj gotovo idealno je klasifikovan test skup.

```
0 **Njutnova metoda**
Preciznost trening skupa: 0.37588097102584184
2 Matrica konfuzije:
[[ 544   79    3]
4  [1250  205    2]
  [ 205   55  211]]
6 Preciznost test skupa: 0.382648401826484
Matrica konfuzije:
8 [[239  28   1]
  [536  87   2]
10  [ 90  19  93]]

12 **Stohasticki gradijentni spust**
Preciznost trening skupa: 1.0
14 Matrica konfuzije:
[[ 626    0    0]
```

```
 [   0 1457    0]
 [   0    0  471]]
Preciznost test skupa: 0.9990867579908675
Matrica konfuzije:
[[268   0   0]
 [  0 624   1]
 [  0   0 202]]
```

Listing 16: Rezultati neuronske mreže

6 Klasifikacija prve i treće klase

Da bi dodatno testirali naše algoritme ponovićemo ih na skupu od dve klase. Vođeni pretodnim rezultatima spajamo prvu i treću klasu u jedan fajl na isti način kao i do sada. Primenjene su metode koje su dale najlošije rezultate u klasifikaciji tri klase a to su SVM Sigmoid i lbfgs Solver neuronskih mreža, kao i metode koje su se dobro pokazale a to su Bajesov pristup i SGD neuronska mreža. Ovoga puta preciznost svih metoda je bila maksimalna, osim SVM sigmoida koja se pokazala lošija nego u prošlom slučaju. Možemo zaključiti da našim podacima ne odgovara ovaj algoritam. Listing 17 prikazuje rezulatate SVM algoritma i LBFGS Solvera koji je u ovom slučaju dao mnogo bolje rezultate.

```
**lbfgs**
Preciznost trening skupa: 1.0
Matrica konfuzije:
[[625   0]
 [  0 471]]
Preciznost test skupa: 1.0
Matrica konfuzije:
[[269   0]
 [  0 202]]

**SVM sigmoid**
```

```
Preciznost trening skupa: 0.23996350364963503
Matrica konfuzije:
[[211 414]
 [419  52]]
Preciznost test skupa: 0.2208067940552017
Matrica konfuzije:
[[ 80 189]
 [178  24]]
```

Listing 17: Rezultati klasifikacije dve klase

7 Zaključak

Klasifikacija velikih podataka je i memorijski i vremenski zahtevan zadatak. Standardni računari često nemaju odgovarajuće performanse za tako velike poslove i iz tog razloga su svi algoritmi primenjeni u okviru platforme Google colab research koja obezbedjuje 12GB ram memorije i razvojno okruzenje za pisanje Python3 koda.
Primenjeni su različiti algoritmi za klasifikaciju. Neki od njih su bili vremenski zahtevniji ali i pored toga nisu davali upotrebljive rezultate dok su se neki bolje pokazali. Za dalja istrazivanja neophodno je detaljno se upoznati sa podacima u biološkom smislu radi boljeg sticanja uvida u kontrolu parametara kao i upotreba jačih racunara koji bi bili sposobni da omoguće prostornu i vremensku efikasnost. [1]

Literatura

[1] *Google colab research.* https://colab.research.google.com/.

[2] *Pandas.* https://pandas.pydata.org/pandas-docs/.

[3] *Scikit-learn.* . https://scikit-learn.org/.

[4] Mehmed Kantardžić. *Data mining: Concepts, Models, Methods, and Algorithms, 2nd. ed., John Wiley & Sons 2011* .

[5] Nenad Mitić. *Istraživanje podataka 1*. http://www.matf.bg.ac.rs/~nenad/ip1.html.

[6] Nenad Mitić. *Istraživanje podataka 2*. http://www.matf.bg.ac.rs/~nenad/ip2.html.

[7] Nemanja Mićović. *Vestacka inteligencija*. http://poincare.matf.bg.ac.rs/~nemanja_micovic/vi.html.

[8] A. Karpatne P.N.Tan, M. Steinbach. *Introduction to Data Mining, 2nd ed, Pearson Education, 2019* .

[9] Mladen Nikolić Predrag Janičić. *Veštačka inteligencija*. http://poincare.matf.bg.ac.rs/~janicic\courses/vi.pdf.