Seminarski rad iz predmeta

Istraživanje podataka 2

Klasifikacija PMBC ćelija

Marko Veljković 43/2015

prof. dr Nenad Mitić

Aleksandar Veljković

Sadržaj

1 Uvod

//za sada

Ulazne datoteke sadrže podatke dobijene iz perifernih mononukleoznih krvnih ćelija (PBMCs). PBMC ćelije uključuju ćelije različitih tipova: limfocite, monocite i dendritske ćelije. PBMC ćelije se koriste u istraživanju u različitim oblastima biomedicine, uključujući infektivne bolesti, imunologiju, malignitet, transplantacionu imunologiju, razvoj vakcina, i skrining. Mada mogu da imaju različite funkcije, glavna funkcija PBMC ćelija je imuna odbrana organizma.

Svaki tip ćelije ima karakteristične obrasce proteina i gena koje ih međusobno razlikuju i mogu da se koriste za podelu prema njihovom tipu.

Svaka od ulaznih datoteka sadrži podatke vezane za ekspresiju 31221 gena na skupu od 742 i 714 PBMC ćelija. Skup gena koji su posmatrani je identičan u obe datoteke i uređen po istom redosledu.

Nazivi svih gena imaju prefiks hg38 koji označava da su podaci vezani za verziju 38. humanog genoma.

Datoteke predstavljaju pluripotentne[1] matične ćelije, tj. ćelije koje grade epitelno tkivo disajnih puteva. U prvoj datoteci se nalaze podaci o matičnim ćelijama u gornjim disajnim putevima, dok su u drugoj, podaci o matičnim ćelijama u donjim disajnim putevima.

Ulazna datoteka sadrži podatke u CSV formatu u obliku ukrštene tabele. Svaka datoteka ima 31222 reda, pri čemu prvi red sadrži redni broj ćelije za koju je vršeno istraživanje, a prva kolona sadrži identifikaciju gena. Polje u preseku prvog reda i prve kolone je inicijalno bilo prazno, a sada je ubačeno ime ’geni’ radi lakšeg rukovanja sa imenima gena.

Vrednosti u matrici (koje ne pripadaju prvom redu/koloni - 313221xN vrednosti) sadrže broj transkripta gena (navedenog u prvoj koloni) u ćeliji.

[1] Postojanje neprepoznatih matičnih ćelija pokazano je indirektnim zapažanjima, a da nije utvrđen tačan izgled i građa tih ćelija. Pluripotentne matične ćelije su jedna od podvrsta neprepoznatih matičnih ćelija

2 Analiza i pretprocesiranje podataka

2.1 Analiza podataka

```
Dimenzija podataka prvog fajla: (31221, 742)
Dimenzija podataka drugog fajla: (31221, 714)
```

Slika 1: dimenzije ulaznih podataka

Mali deo podataka prvog fajla:

	1	2	3	4	5	6	7	...	736	737	738	739	740	741	742
geni								...							
hg38_A1BG	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0
hg38_A1BG-AS1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A1CF	1	0	6	0	2	0	0	...	0	0	4	5	0	0	0
hg38_A2M	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A2M-AS1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A2ML1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A2ML1-AS1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A2ML1-AS2	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A3GALT2	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A4GALT	0	1	0	1	0	0	0	...	0	0	0	0	0	0	0
hg38_A4GNT	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_AAAS	0	1	0	1	0	0	0	...	0	0	0	4	0	1	0
hg38_AACS	0	0	0	0	1	0	0	...	0	0	0	1	0	0	0
hg38_AADAC	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_AADACL2	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_AADACL2-AS1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_AADACL3	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_AADACL4	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_AADAT	1	0	0	0	0	0	1	...	0	0	0	0	1	0	0
hg38_AAED1	0	0	0	0	0	0	0	...	1	0	0	1	0	1	0

Slika 2: podaci iz prvog fajla

Mali deo podataka drugog fajla:

	1	2	3	4	5	6	7	...	708	709	710	711	712	713	714
geni								...							
hg38_A1BG	0	0	1	0	0	1	2	...	0	1	2	2	0	0	0
hg38_A1BG-AS1	0	0	0	0	0	0	0	...	0	0	0	0	1	0	0
hg38_A1CF	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A2M	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A2M-AS1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A2ML1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A2ML1-AS1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A2ML1-AS2	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A3GALT2	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_A4GALT	1	0	0	0	0	0	0	...	0	0	0	1	0	0	0
hg38_A4GNT	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_AAAS	0	0	2	1	0	0	0	...	0	0	0	1	0	0	0
hg38_AACS	0	0	0	0	0	0	1	...	0	0	2	1	1	0	0
hg38_AADAC	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_AADACL2	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_AADACL2-AS1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_AADACL3	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_AADACL4	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
hg38_AADAT	0	0	1	1	0	0	0	...	0	0	1	0	1	0	0
hg38_AAED1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

Slika 3: podaci iz drugog fajla

2.2 Pretprocesiranje

Pre početka klasifikacije, moramo obraditi i pripremiti sirove podatke za rad sa njima. Korišćene metode su provera i umetanje nepostojećih vrednosti, uklanjanje nula redova i kolona, detekcija i eliminacija elemenata van granica, završna obrada podataka.

2.2.1 Provera i umetanje nepostojećih vrednosti

Izvršavanjem jednostavne funkcije df.isnull().values.any() nad oba skupa podataka, utvrđujemo da se ni u jednom skupu ne nalaze nepostojeće (NaN) vrednosti.

2.2.2 Uklanjanje nula redova i kolona

Razlog zbog kojeg uklanjamo nula redove i kolone je taj što takvi redovi, odnosno kolone nisu značajni za generisanje modela klasifikacije, a njihovim uklanjanjem podižemo efikasnost izvršavanja programa. Eksperimentalnim putem je utvrđeno da u oba fajla ne postoje nula kolone, tako da one nisu posebno obrađivane.

```
def remove_zero_rows(df):
    izbaceni = df[df.sum(axis=1) == 0]
    df.drop(df[df.sum(axis=1) == 0].index, inplace=True)
    return df, izbaceni
```

Slika 4: Funkcija koja vraća skup podataka bez nula redova

```
Dimenzija podataka prvog fajla nakon uklanjanja 0 redova i kolona: (19937, 742)
Dimenzija podataka drugog fajla nakon uklanjanja 0 redova i kolona: (19036, 714)
```

Slika 5: dimenzija podataka nakon uklanjanja nula redova

Vidimo da je iz prvog skupa izbačeno 11284 reda, a iz drugog 12185.

Da bi skupovi i dalje imali iste gene, sada iz oba skupa izbacujemo one gene koje smo izbacili iz jednog skupa, a nismo iz drugog.

```
df_first.drop(zero_rows_second.index.difference(zero_rows_first.index).values, inplace=True)
df_second.drop(zero_rows_first.index.difference(zero_rows_second.index).values, inplace=True)
```

Slika 6: funkcija koja uklanja višak redova

```
Dimenzija podataka prvog fajla nakon uklanjanja 0 redova drugog fajla: (17699, 742)
Dimenzija podataka drugog fajla nakon uklanjanja 0 redova prvog fajla: (17699, 714)
```

Slika 7: dimenzija podataka nakon uklanjanja viška redova

Na osnovu dobijenih rezultata, zaključujemo da je 9947 gena imalo nula redove u oba fajla.

2.2.3 Uklanjanje elemenata van granice

Sledeći korak u pretprocesiranju podataka je otkrivanje i uklanjanje elemenata van granica, odnosno elemenata čije se vrednosti značajno razlikuju od vrednosti ostalih podataka u posmatranom skupu. Razlog uklanjanja je njihov moguć uticaj na generisanje nekih modela za klasifikaciju, čime bi doprineli da model ne funkcioniše na najbolji mogući način.

U nastavku je data implementacija funkcije, koja korišćenjem Z-vrednosti, pronalazi i izdvaja elemente van granica iz trenutnog skupa podataka.

Z-vrednost svakog podatka X_{ij} se računa po formuli $Z_{ij} = \frac{X_{ij} - \mu_j}{\sigma_j}$, gde X_{ij} predstavlja vrednost trenutno posmatranog elementa (i-ti red, j-ta kolona), μ_j srednju vrednost j-te kolone, a σ_j standardnu devijaciju elemenata j-te kolone.

```
def detect_and_exclude_outliers(df):
    ouliers = df[(np.abs(stats.zscore(df)) >= 3).any(axis=1)]
    df_without_outliers = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]
    return ouliers, df_without_outliers
```

Slika 8: funkcija koja otkriva i eleminiše anomalije

Prikaz nekih od elemenata van granica, koji su izbačeni iz prvog, odnosno drugog skupa podataka.

	1	2	3	4	5	6	...	737	738	739	740	741	742
geni							...						
hg38_ABRACL	0	25	6	6	0	2	...	5	0	2	2	0	2
hg38_ACTB	55	135	190	78	33	9	...	21	75	83	16	126	23
hg38_ACTG1	87	166	236	186	66	16	...	50	108	126	26	166	50
hg38_AFP	201	6	392	2	62	4	...	6	120	153	4	394	2
hg38_AGR2	0	49	2	7	1	1	...	0	0	1	0	4	0
hg38_AGR3	0	1	0	0	1	6	...	0	0	1	3	0	1
hg38_AGT	22	0	43	1	18	0	...	0	17	9	0	2	0
hg38_AKR1B10	4	0	138	0	0	0	...	0	2	7	0	0	0
hg38_AKR1C2	4	31	23	6	3	3	...	6	0	0	0	1	7
hg38_AL365500.1	0	0	0	0	0	0	...	0	0	0	0	0	0

Slika 9: elementi van granice prvog skupa

	1	2	3	4	5	6	7	...	708	709	710	711	712	713	714
geni								...							
hg38_ACTB	17	3	45	52	2	33	15	...	8	10	88	89	69	21	1
hg38_ACTG1	55	77	80	90	7	110	102	...	24	26	198	384	198	109	36
hg38_ADI1	0	0	0	2	0	2	0	...	0	2	0	0	0	0	1
hg38_AGR2	0	0	0	18	6	1	16	...	1	1	1	15	1	0	8
hg38_ALDH1A1	1	0	0	0	1	0	0	...	0	0	0	0	1	0	0
hg38_ALDOA	16	18	20	18	0	33	75	...	8	13	13	106	30	18	6
hg38_ANXA2	16	7	15	16	1	10	40	...	15	3	31	59	19	25	0
hg38_ARL6IP1	2	1	1	1	1	0	4	...	0	2	7	7	7	2	1
hg38_ATP5F1B	10	0	3	8	1	4	2	...	2	3	6	24	16	2	2
hg38_ATP5F1E	13	8	34	33	2	23	45	...	11	5	25	79	80	15	10

Slika 10: elementi van granice drugog skupa

```
Dimenzija podataka prvog fajla nakon uklanjanja elemenata van granica: (17232, 742)
Dimenzija podataka drugog fajla nakon uklanjanja elemenata van granica: (17427, 714)
```

Slika 11: dimenzija podataka nakon uklanjanja anomalija

Ponovo da bi skupovi i dalje imali iste gene, iz oba skupa izbacujemo one gene koje smo izbacili iz jednog skupa, a nismo iz drugog.

```
Dimenzija podataka prvog fajla nakon uklanjanja elemenata van granica drugog fajla: (17196, 742)
Dimenzija podataka drugog fajla nakon uklanjanja elemenata van granica prvog fajla: (17196, 714)
```

Slika 12: dimenzija podataka nakon ponovnog uklanjanja viška redova

Zaključujemo da je 236 elemenata van granice bilo zajedničko za oba skupa podataka.

Spisak nekolicine gena koji su predstavljali elemente van granice u oba skupa:

'hg38_ACTB', 'hg38_ACTG1', 'hg38_AGR2', 'hg38_ALDH1A1', 'hg38_ALDOA', 'hg38_ANXA2', 'hg38_ATP5F1B', 'hg38_ATP5F1E', 'hg38_ATP5MC2', 'hg38_ATP5MC3', ... 'hg38_UBC', 'hg38_UBE2C', 'hg38_UQCRB', 'hg38_UQCRH', 'hg38_UQCRQ', 'hg38_VIM', 'hg38_WFDC2', 'hg38_YBX1', 'hg38_ZFAS1'.

2.2.4 Završna obrada

Skupovi sada imaju iste gene, ali se i dalje razlikuju po broju ćelija. Postoje dva načina da se ovaj problem razreši. Prvi je da se ’višak’ kolona iz prvog skupa prekopira u drugi, a drugi način je da se ’višak’ kolona jednostavno ukloni iz prvog fajla, zašta je se u ovom radu i opredeljeno, jednostavno zbog veće efikasnosti.

```
df_first.drop([str(i) for i in range(715, 743)], axis=1, inplace=True)
```

Slika 13: funkcija koja uklanja 'višak' kolona iz prvog skupa

Da bi smo mogli da vršimo klasifikovanje i po ćelijama i po genima, pravimo još 2 skupa podataka, koji zapravo predstavljaju samo transponovane podatke iz prva 2 skupa. U nastavku ću govoriti samo o prva dva skupa podatka, zato što su svi postupci identično ponavljani i na druga dva skupa.

Sada u oba skupa podataka dodajemo kolonu ’class’ koja predstavlja klasu kojoj podaci pripadaju, sa vrednostima 1 za podatke iz prvog skupa, odnosno 2 za podatke iz drugog skupa.

Na kraju spajamo podatke iz oba skupa u jednu matricu, nad kojom ćemo vršiti klasifikaciju.

	1	2	3	4	5	6	7	8	...	708	709	710	711	712	713	714	class
geni									...								
hg38_A1BG	0	0	0	0	0	0	0	0	...	0	1	0	0	1	0	0	1
hg38_A1BG-AS1	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	1
hg38_A1CF	1	0	6	0	2	0	0	2	...	0	0	0	0	0	0	0	1
hg38_A2M	0	0	0	0	0	0	0	7	...	0	0	0	0	0	0	0	1
hg38_A2M-AS1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1
hg38_A4GALT	0	1	0	1	0	0	0	0	...	3	0	0	1	0	3	0	1
...	..	..	..	..	..	..	..	..	...	...	...	...	...	...	...	...	...
hg38_ZXDB	1	0	0	1	0	0	0	0	...	0	0	0	1	0	0	0	2
hg38_ZXDC	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	2
hg38_ZYG11A	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	2
hg38_ZYG11B	0	0	0	0	0	1	1	0	...	2	0	0	0	2	0	0	2
hg38_ZYX	1	0	0	0	0	1	2	1	...	0	0	0	2	2	0	0	2
hg38_ZZEF1	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	2

Slika 14: prikaz spojednih podataka sa dodatim klasama

```
Dimenzija podataka za klasifikaciju po genima(34392, 715)
Dimenzija podataka za klasifikaciju po celijama(1428, 17197)
```

Slika 15: dimenzija dva skupa podataka pred klasifikaciju

3 Klasifikacija

Sada kada smo pripremili podatke, možemo početi sa generisanjem modela za klasifikaciju.

Skup podataka delimo na 2 dela, prvi predstavlja matricu sa svim podacima bez kolone klasa, a drugi vektor-kolonu koja sadrži klase kojima redovi prve matrice pripadaju.

Oba dela zatim šaljemo kao parametre funkciji sklearn.model_selection.train_test_split koja deli skup podataka na trening i test skup. Kao treći parametar se prosleđuje broj u opsegu [0,1], koji označava deo podataka koji će biti izdvojen za testiranje, u ovom slučaju to je 0.3.

Za klasifikaciju podataka korišćene su:

- Jednostavne metode
  - K najbližih suseda
  - Drvo odlučivanja
  - Mašine sa potpornim vektorima
- Ansambl metode[2]
  - Nasumična šuma (eng. Random forest)
  - Pakovanje (eng. Bagging)
  - Pojačavanje (eng. Boosting)
  - Glasanje (eng. Voting)

Dve funkcije korišćene u svim metodama, radi lakšeg testiranja modela i ispisivanja dobijenih rezultata.

```
def fit_model(model, x_train, x_test, y_train, y_test):
    model.fit(x_train, y_train.ravel())
    print("Rezultat trening skupa: " + str(model.score(x_train, y_train)))
    print("Rezultat test skupa: " + str(model.score(x_test, y_test)))
```

Slika 16: funkcija za treniranje modela I ispisivanje rezultata

```
def prediction(model, x_train, x_test, y_train, y_test):
    y_train_predicted = model.predict(x_train)
    y_test_predicted = model.predict(x_test)
    print("Matrica kofuzije trening skupa:\n" + str(confusion_matrix(y_train, y_train_predicted)))
    print("Matrica kofuzije test skupa:\n" + str(confusion_matrix(y_test, y_test_predicted)))
```

Slika 17: funkcija za predviđanje i ispisivanje matrice konfuzije

[2] Metode koje kombinuju rezultate nekoliko, ne nužno različitih, modela dobijenih nad trening podacima

3.1 Jednostavne klasifikacione metode

3.1.1 K najbližih suseda

Osnovna ideja je da na osnovu k najbližih suseda datog sloga, odredimo kojoj klasi on pripada. Vrši se određivanje najbližih suseda, zatim prebrojavanje koliko suseda pripada kojoj klasi. Ona klasa kojoj pripada najviše suseda je dodeljena posmatranom slogu.

```
def knn(x_train, x_test, y_train, y_test, non):
    print("-----K najblizih suseda:-----")
    start = time.time()
    model = KNeighborsClassifier(n_neighbors=non, weights='uniform')
    fit_model(model, x_train, x_test, y_train, y_test)
    prediction(model, x_train, x_test, y_train, y_test)
    print("Vreme izvrsavanja: " + str(time.time()-start) + '\n')
```

Slika 18: funkcija k najbližih suseda

```
-----K najblizih suseda:-----
Rezultat trening skupa: 0.9226136080418709
Rezultat test skupa: 0.8356270595076565
Matrica kofuzije trening skupa:
[[10945  1069]
 [  794 11266]]
Matrica kofuzije test skupa:
[[4256  926]
 [ 770 4366]]
Vreme izvrsavanja: 1940.3197627067566
```

Slika 19: knn klasifikacija po genima

```
-----K najblizih suseda:-----
Rezultat trening skupa: 0.982982982982983
Rezultat test skupa: 0.9557109557109557
Matrica kofuzije trening skupa:
[[489  16]
 [  1 493]]
Matrica kofuzije test skupa:
[[191  18]
 [  1 219]]
Vreme izvrsavanja: 106.51161241531372
```

Slika 20: knn klasifikacija po ćelijama

Prikazani su rezultati generisanja modela sa parametrom non=3, takođe je pokušano sa 5 i 10 suseda, ali je sa 3 pokazalo najbolje rezultate. U odnosu na 5 suseda je imalo sveobuhvatno bolje rezultate, dok je u odnosu na 10 dalo sličan rezultat, ali je vremenski bio dosta efikasniji.

3.1.2 Drvo odlučivanja

Problem klasifikacije rešavamo postavljanjem pažljivo sastavljenih pitanja o atributima iz test skupa. Svaki put kada dobijemo odgovor postavljamo novo pitanje, dok ne dođemo do zaključka o klasi posmatranog sloga.

```
def dtc(x_train, x_test, y_train, y_test):
    print("-----Drvo odlucivanja:-----")
    start = time.time()
    model = DecisionTreeClassifier()
    fit_model(model, x_train, x_test, y_train, y_test)
    prediction(model, x_train, x_test, y_train, y_test)
    print("Vreme izvrsavanja: " + str(time.time()-start) + '\n')
```

Slika 21: funkcija drvo odlučivanja

```
-----Drvo odlucivanja:-----
Rezultat trening skupa: 0.992606131095788
Rezultat test skupa: 0.8558829230471021
Matrica kofuzije trening skupa:
[[11949    65]
 [  113 11947]]
Matrica kofuzije test skupa:
[[4454  728]
 [ 759 4377]]
Vreme izvrsavanja: 11.485100507736206
```

Slika 22: klasifikacija drvetom odlučivanja po genima

```
-----Drvo odlucivanja:-----
Rezultat trening skupa: 1.0
Rezultat test skupa: 0.9487179487179487
Matrica kofuzije trening skupa:
[[505   0]
 [  0 494]]
Matrica kofuzije test skupa:
[[204   5]
 [ 17 203]]
Vreme izvrsavanja: 1.8750097751617432
```

Slika 23: klasifikacija drvetom odlučivanja po ćelijama

Kao mera nečistoće u prikazanom algoritmu, korišćen je Ginijev indeks, dok nije bilo ograničenja maksimalne dubine stabla.

3.1.3 Mašine sa potpornim vektorima

Metod binarne klasifikacije, koji koristi linearni model $f_{w,w_0}(x) = w * x + w_0$. Dobijamo hiperravan koja deli podatke u dve klase. Neke podatke ne možemo razdvojiti pomoću hiperravni, tako da podatke moramo preslikati u visokodimenzioni prostor i onda ih razdvojiti pomoću hiperravni, koja će u dvodimenzionom prostoru biti odgovarajuća kriva. Funkcija koja radi preslikavanje instanci naziva se kernel.

```
def svm(x_train, x_test, y_train, y_test, c, kernel, gamma):
    print("-----SVM:-----")
    start = time.time()
    model = SVC(C=c, kernel=kernel, gamma=gamma)
    # noinspection PyTypeChecker
    fit_model(model, x_train, x_test, y_train, y_test)
    print("Vreme izvrsavanja: " + str(time.time()-start) + '\n')
```

Slika 24: funkcija svm

```
-----SVM:-----
Rezultat trening skupa: 0.9521890836587189
Rezultat test skupa: 0.9247916262841636
Vreme izvrsavanja: 822.3961896896362
```

Slika 25: klasifikacija svm-om po genima

```
-----SVM:-----
Rezultat trening skupa: 1.0
Rezultat test skupa: 0.9953379953379954
Vreme izvrsavanja: 9.582205057144165
```

Slika 26: klasifikacija svm-om po ćelijama

Linearni kernel je dao najbolje, gore prikazane, rezultate. Polinomijalni kernel sa stepenom polinoma 2 je dao slične rezultate, dok se Gausov odlično pokazao na trening podacima (1.0), dok je dosta loše klasifikovao test podatke (0.49), odakle možemo zaključiti da je došlo do preprilagođavanja.

Iako su mašine sa potpornim vektorom dale najbolje rezultate, ako pogledamo i vreme izvršavanja, možemo ustanoviti da rezultati koje je dao model drveta odlučivanja nisu zanemarljivi, tako da ćemo drvo odlučivanja koristi kao osnovni model u nastavku.

3.2 Ansambl klasifikacione metode

Kombinuju rezultate nekoliko modela klasifikacije (istih ili različitih), u cilju smanjenja nivoa greške.

3.2.1 Nasumična šuma

Deli skup podataka na komplementarne podskupove i za svaki od podskupova, generiše zaseban model drveta odlučivanja. Krajnji model predstavlja srednju vrednost rezultata dobijenih iz generisanih modela.

```
def rfc(x_train, x_test, y_train, y_test, n_est):
    print("-----Nasumicna suma:-----")
    start = time.time()
    model = RandomForestClassifier(n_estimators=n_est)
    # noinspection PyTypeChecker
    fit_model(model, x_train, x_test, y_train, y_test)
    prediction(model, x_train, x_test, y_train, y_test)
    print("Vreme izvrsavanja: " + str(time.time()-start) + '\n')
```

Slika 27: funkcija nasumična šuma

```
-----Nasumicna suma:-----
Rezultat trening skupa: 0.9924815153277394
Rezultat test skupa: 0.9108354332234929
Matrica kofuzije trening skupa:
[[11927    87]
 [   94 11966]]
Matrica kofuzije test skupa:
[[4737  445]
 [ 475 4661]]
Vreme izvrsavanja: 14.566174745559692
```

Slika 28: rezultat klasifikacije nasumičnom šumom po genima

```
-----Nasumicna suma:-----
Rezultat trening skupa: 1.0
Rezultat test skupa: 0.9906759906759907
Matrica kofuzije trening skupa:
[[505   0]
 [  0 494]]
Matrica kofuzije test skupa:
[[207   2]
 [  2 218]]
Vreme izvrsavanja: 1.1130309104919434
```

Slika 29: rezultat klasifikacije nasumičnom šumom po ćelijama

Kao mera nečistoće, korišćen je Ginijev indeks, a korišćeno je 50 različitih modela.

3.2.2 Pakovanje

Metoda koja deli ulazni skup podataka na podskupove u kojima se elementi mogu ponavljati. Model se formira za svaki ulazni skup zasebno.

```
def bag(x_train, x_test, y_train, y_test, n_est):
    print("-----Bagging:-----")
    start = time.time()
    est_model = DecisionTreeClassifier()
    model = BaggingClassifier(base_estimator=est_model, n_estimators=n_est)
    fit_model(model, x_train, x_test, y_train, y_test)
    prediction(model, x_train, x_test, y_train, y_test)
    print("Vreme izvrsavanja: " + str(time.time()-start) + '\n')
```

Slika 30: funkcija bagging

```
-----Bagging:-----
Rezultat trening skupa: 0.9904461244496137
Rezultat test skupa: 0.8910641597208762
Matrica kofuzije trening skupa:
[[11912   102]
 [  128 11932]]
Matrica kofuzije test skupa:
[[4707  475]
 [ 649 4487]]
Vreme izvrsavanja: 132.42388820648193
```

Slika 31: rezultat klasifikacije bagging modelom po genima

```
-----Bagging:-----
Rezultat trening skupa: 1.0
Rezultat test skupa: 0.9696969696969697
Matrica kofuzije trening skupa:
[[505   0]
 [  0 494]]
Matrica kofuzije test skupa:
[[207   2]
 [ 11 209]]
Vreme izvrsavanja: 28.544705390930176
```

Slika 32: rezultat klasifikacije bagging modelom po ćelijama

Kao primarni model korišćena su 20 drveta odlučivanja.

3.2.3 Pojačavanje

Na početku se formira loš klasifikator i svim slogovima se dodaju jednake težine. Kroz iteracije se vrši prepravka težina na osnovu rezultata iz prethodne iteracije. Ako je podatak tačno klasifikovan, težina sloga se smanjuje, dok ako je klasifikovan pogrešno, težina se povećava.

```
def boost(x_train, x_test, y_train, y_test, n_est):
    print("-----Boosting:-----")
    start = time.time()
    est_model = DecisionTreeClassifier()
    model = AdaBoostClassifier(base_estimator=est_model, n_estimators=n_est)
    fit_model(model, x_train, x_test, y_train, y_test)
    prediction(model, x_train, x_test, y_train, y_test)
    print("Vreme izvrsavanja: " + str(time.time()-start) + '\n')
```

Slika 33: funkcija boosting (AdaBoost)

```
-----Boosting:-----
Rezultat trening skupa: 0.992606131095788
Rezultat test skupa: 0.8582089552238806
Matrica kofuzije trening skupa:
[[11949    65]
 [  113 11947]]
Matrica kofuzije test skupa:
[[4553  629]
 [ 834 4302]]
Vreme izvrsavanja: 41.751368284225464
```

Slika 34: rezultati klasifikacije boosting modelom nad genima

```
-----Boosting:-----
Rezultat trening skupa: 1.0
Rezultat test skupa: 0.9557109557109557
Matrica kofuzije trening skupa:
[[505   0]
 [  0 494]]
Matrica kofuzije test skupa:
[[205   4]
 [ 15 205]]
Vreme izvrsavanja: 1.513981580734253
```

Slika 35: rezultat klasifikacije boosting modelom nad ćelijama

Kao i u prethodnom primeru, i u ovoj ansambl metodi, glavni primarni metod su bila 20 drveta odlučivanja.

3.2.4 Glasanje

Svaki ulazni slog se klasifikuje svim prosleđenim osnovnim modelima i na osnovu dobijenih rezultata određuje se klasa svakog sloga. //prepraviti sliku i dobijene rezultate

```
def vot(x_train, x_test, y_train, y_test, power, mode, rtype):
    print("-----Voting:-----")
    start = time.time()
    est_model1 = RandomForestClassifier(n_estimators=50)
    est_model2 = SVC(C=100, kernel='linear')
    est_model3 = KNeighborsClassifier(n_neighbors=3)
    model = VotingClassifier(estimators=[('dtc', est_model1), ('svc', est_model2), ('knn', est_model3)],
                             voting=power)
    est_model1.fit(x_train, y_train.ravel())
    est_model2.fit(x_train, y_train.ravel())
    est_model3.fit(x_train, y_train.ravel())

    fit_model(model, x_train, x_test, y_train, y_test, mode, 'voting_' + rtype)
    prediction(model, x_train, x_test, y_train, y_test)
    print("Vreme izvrsavanja: " + str(time.time()-start) + '\n')
```

Slika 36: funkcija voting

```
-----Voting:-----
Rezultat trening skupa: 0.9776522389299659
Rezultat test skupa: 0.9256638883504555
Matrica kofuzije trening skupa:
[[11732   296]
 [  242 11804]]
Matrica kofuzije test skupa:
[[4784  384]
 [ 383 4767]]
Vreme izvrsavanja: 3965.2135984897614
```

Slika 37: rezultat klasifikacije voting modelom nad genima

```
-----Voting:-----
Rezultat trening skupa: 1.0
Rezultat test skupa: 0.9976689976689976
Matrica kofuzije trening skupa:
[[489   0]
 [  0 510]]
Matrica kofuzije test skupa:
[[224   1]
 [  0 204]]
Vreme izvrsavanja: 119.12083983421326
```

Slika 38: rezultat klasifikacije voting modelom nad ćelijama

Korišćena su tri osnovna modela: mašine sa potpornim vektorima uz korišćenje linearnog kernela, nasumična šuma sa 50 drveta odlučivanja i metod k najbližih suseda sa posmatranjem 3 najbliža suseda. Način glasanja je 'tvrd' što znači da metod samo prebrojava glasove i određuje klasu trenutnog sloga.

4 Analiza i poređenje dobijenih rezultata

U prethodnom delu, nakon rezultata izvršavanja svakog od modela klasifikovanja, navedeni su kratki opisi parametara modela i njihov uticaj na generisanje modela. Ovde ćemo dodatno izanalizirati dobijene rezultate i prodiskutovati o njihovom značenju.

Prva i osnovna stvar koju možemo primetiti u dobijenim rezultatima kako jednostavnih, tako i ansambl metoda je ta da modeli klasifikovanja nad ćelijama rade i po nekoliko desetina puta brže od modela nad genima. Osnovni razlog takvih rezultata je broj slogova podataka, odnosno kada klasifikujemo po genima, podaci sadrže 24 puta više slogova nego kada klasifikujemo po ćelijama.

Sada ćemo ponoviti osnovna zapažanja koja smo naveli u prethodnom delu, ali malo detaljnije objašnjena i analizirati dodatne aspekte izvršavanja algoritama i njihove rezultate. Zatim ćemo uporediti rezultate metoda iz istih kategorija, a na kraju objediniti sve to u jedan celovit zaključak, kojim ovaj rad privodimo kraju.

4.1 K najbližih suseda

Atributi	Broj suseda	Rezultat trening skupa	Rezultat test skupa	Vreme izvršavanja
Ćelije	3	0.9189997507684639	0.8347547974413646	1681.6338067054749
Ćelije	5	0.9027997009221567	0.8436712541190153	1719.6496942043304
Ćelije	10	0.8807011713882197	0.8471603023841829	1785.9461197853088
Geni	3	0.974974974974975	0.9627039627039627	94.60397911071777
Geni	5	0.9669669669669669	0.9627039627039627	97.43765449523926
Geni	10	0.9619619619619619	0.9603729603729604	100.50979804992676

Tabela 1: rezultati metoda k najbližih suseda

Možemo primetiti da se povećanjem broja suseda, rezultati klasifikovanja polako, ali sigurno pogoršavaju, kao i da se vreme generisanja modela povećava. Razlog slabljenja rezultata je nepostojanje zajedničke karakteristike većeg broja slogova u n-dimenzionom prostoru, koja bi odvojila dve klase na precizniji način. Iz ovih rezultata zaključujemo da je od ponuđenih najbolja opcija, generisanje modela klasifikacije sa 3 suseda. Pored toga što se modeli u kojima geni predstavljaju atribute generišu u proseku 17.82 puta brže, oni takođe daju 7% i 15% bolje rezultate na trening i test skupu, respektivno.

4.2 Drvo odlučivanja

Atributi	Mera nečistoće	Rezultat trening skupa	Rezultat test skupa	Vreme izvršavanja
Ćelije	Gini	0.9922322837916424	0.8592750533049041	10.478996992111206
Ćelije	Entropija	0.9916507435407493	0.8647024617173871	9.101840257644653
Geni	Gini	1.0	0.9603729603729604	1.9918508529663086
Geni	Entropija	1.0	0.9696969696969697	1.8117952346801758

Tabela 2: rezultati metoda drvo odlučivanja

Metod je bez obzira na veličinu podataka, dao veoma zapažene rezultate i to za izuzetno kratko vreme. Možemo primetiti da smo uz korišćenje entropije kao mere nečistoće dobili za nijansu bolje rezultate uz kraće vreme generisanja modela. Što se statističkih parametara tiče, generisanje modela nad ćelijama se izvršava 5.15 puta brže, uz skoro identične rezultate na trening skupu i 12.5% bolje rezultate na test skupu.

4.3 Mašine sa potpornim vektorima

Atributi	Kernel	Rezultat trening skupa	Rezultat test skupa	Vreme izvršavanja
Ćelije	Gausov	0.9916922821300989	0.5059119984493119	3970.372884750366
Ćelije	Polinomijalan	0.9915676663620503	0.873522000387672	706.1637718677521
Ćelije	Linearan	0.9518152363545734	0.9240162822252375	917.7920296192169
Geni	Gausov	1.0	0.4988344988344988	89.77373218536377
Geni	Polinomijalan	1.0	0.9883449883449883	14.012723445892334
Geni	Linearan	1.0	0.9976689976689976	13.808668851852417

Tabela 3: rezultati metoda mašine sa potpornim vektorima

Prvu stvar koju možemo primetiti je nekoliko puta sporije generisanje modela sa Gausovim krenelom, u odnosu na one sa polinomijalnim, sa stepenom polinoma 2, i linearnim. Razlog tome je način rada tih funkcija. Dok linearna i polinomijalna funkcija dele prostor na 2 dela, Gausova funkcija se prilagođava podacima što je više moguće, što je na kraju i dovelo da se u oba naša primera javi preprilagođavanje trening podacima i otud tako slabi rezultati nad test skupom. Što se preostala dva kernela tiče, vidimo da je linearni postigao bolje rezultate, kao i da je bilo 10% manje razlike između rezultata trening i test skupa, ako posmatramo modele generisane nad genima. Modeli su se u proseku generisali čak 53.37 puta brže nad ćelijama u odnosu na gene i ostvarili 11% bolje rezultate.

4.4 Nasumična šuma

Atributi	Broj modela	Rezultat trening skupa	Rezultat test skupa	Vreme izvršavanja
Ćelije	20	0.9906538173963613	0.9138398914518318	5.800857067108154
Ćelije	50	0.9916922821300989	0.9196549718937779	13.877151250839233
Ćelije	100	0.9917753593087979	0.9185888738127544	28.33163285255432
Geni	20	0.998998998998999	0.9836829836829837	0.6175384521484375
Geni	50	1.0	0.993006993006993	1.1140472888946533
Geni	100	1.0	0.993006993006993	1.9284465312957764

Tabela 4: rezultati metoda nasumična šuma

Osnovnu stvar koju možemo zapaziti su odlični rezultati svih modela, nad trening podacima je uvek bilo preko 99% preciznosti, dok je nad test podacima ta brojka uvek premašivala 91%. Takođe možemo zapaziti slične, skoro identične, rezultate ako posmatramo odvojeno modele generisane nad genima i nad ćelijama. Ono što se najviše razlikuje je vreme izvršavanja algoritama, vidimo da direktno zavisi od broja primarnih modela. S obzirom da sveukupno vreme izvršavanja algoritma nije predugačko, kod određivanja najboljeg modela, možemo posmatrati finije razlike u rezultatima izvršavanja, tako da za najbolji model ove grupe uzimamo šumu sa 50 drveta. Modeli koji su uzimali gene kao atribute su se generisali 12.33 puta brže i dali 1% i 7% bolje rezultate nad trening i test skupovima.

4.5 Pakovanje

Atributi	Broj modela	Rezultat trening skupa	Rezultat test skupa	Vreme izvršavanja
Ćelije	20	0.9891999667691285	0.8947470440007753	109.87931990623474
Ćelije	50	0.9912353576472543	0.9001744524132583	273.3827335834503
Ćelije	100	0.9915676663620503	0.8999806163985269	523.2003977298737
Geni	20	1.0	0.9836829836829837	24.6112642288208
Geni	50	1.0	0.986013986013986	58.814847469329834
Geni	100	1.0	0.9836829836829837	114.72934985160828

Tabela 5: rezultati metoda pakovanje

Ako posmatramo samo rezultate modela, možemo zapaziti da su u velikoj meri slični rezultatima dobijenih nasumičnom šumom, kao i da vreme izvršavanja direktno zavisi od broja korišćenih primarnih modela. Zbog veoma male razlike u dobijenim rezultatima, ali i ne tako male razlike u vremenima izvršavanja, kao najbolje modele iz ove grupe uzimamo pakovanja sa 20 primarnih modela. Generisanje nad ćelijama 4.61 puta brže sa 1% odnosno 10% bolje postignutim rezultatima.

4.6 Pojačavanje

Atributi	Broj modela	Rezultat trening skupa	Rezultat test skupa	Vreme izvršavanja
Ćelije	20	0.9919415136661959	0.8813723589842993	60.3197181224823
Ćelije	50	0.9919415136661959	0.8860244233378561	89.32315754890442
Ćelije	100	0.9919415136661959	0.8825353750726885	122.7359528541565
Geni	20	1.0	0.9627039627039627	1.4427461624145508
Geni	50	1.0	0.965034965034965	1.4064133167266846
Geni	100	1.0	0.9696969696969697	1.3581182956695557

Tabela 6: rezultati metoda pojačavanje

Prvu zanimljivu stvar koju možemo primetiti je da su modeli dali identične rezultate nad trening podacima. Druga zanimljivost je ta da su vremena generisanja modela nad ćelijama u opadajućem redosledu, što je posledica dodatnih izračunavanja u algoritmu, kao što je određivanje matrica konfuzije. Zbog delimično boljih rezultata i ne preterane razlike u vremenu izvršavanja, kao najbolji model uzećemo metod sa 50 primarnih modela. Dobijamo da se modeli nad ćelijama prosečno generišu čak 65.67 puta brže i da daju 1% bolje rezultate nad trening i 9% nad test podacima.

4.7 Glasanje

Što se ovog metoda tiče, rezultate smo već prikazali i prethodnom delu i nemamo više ništa dodati u ovde. Upoređivanje metode glasanja sa ostalim ansambl metodama ćemo uraditi u odeljku 4.9. Ovde ćemo samo uktrakto još pomenuti statističke parametre kao i u prošlim metodama. Generisanje metoda nad genima trajalo je 33.29 puta duže od generisanja nad ćelijama, dok su modeli nad ćelijama dali 2% i 8% bolje rezultate na trening i test skupu.

4.8 Jednostavne metode

Nakon što smo završili analizu svih metoda pojedinačno, sada ćemo analizirati grupe metoda, da bi smo videli koje su se najbolje pokazale nad našim podacima. Za svaku metodu uzimamo parametre sa kojima se najbolje pokazala u prethodnom odeljku.
Odnosno, za k najbližih suseda uzimamo 3 najbliža suseda, što se tiče drveta odlučivanja, uzimamo entropiju kao meru nečistoće i ne ograničavamo drvo maksimalnom dubinom,
dok kod mašina sa potpornim vektorima koristimo linearni kernel.

Metoda	Atributi	Rezultat trening skupa	Rezultat test skupa	Vreme izvršavanja	Tačno klasifikovani	Pogrešno klasifikovani
Knn	Ćelije	0.925	0.845	1790.628	22265 + 8718	1809 + 1600
Knn	Geni	0.976	0.956	85.444	975 + 410	24 + 19
Drvo odlučivanja	Ćelije	0.992	0.871	8.273	23880 + 8991	194 + 1327
Drvo odlučivanja	Geni	1.000	0.953	1.244	999 + 409	0 + 20
Svm	Ćelije	0.951	0.928	1003.812	22889 + 9576	1185 + 742
Svm	Geni	1.000	0.995	11.938	999 + 427	0 + 2

Tabela 7: rezultati jednostavnih metoda

U tabeli su posebno označeni metodi koji su dali sveukupno najbolje rezultate – plavom bojom i oni koji su dali najbolje rezultate u odnosu na vreme generisanja – zelenom bojom.

Primećujemo izuzetno dobre i prihvatljive rezultate kod metoda k najbližih suseda, međutim u poređenju sa rezultatima drugih metodama ti rezultati su ipak ne zadovoljavajući, kako u pogledu rezultata tako i u poređenju vremena generisanja.

Što se tiče drveta odlučivanja vidimo da je opet ekspresno završilo sa radom u oba slučaja, ali takođe možemo uočiti blago preprilagođavanje, pogotovo kod metoda zasnovanog na genima, gde su rezultati sa treninga skupa čak 14% bolji od onih sa test skupa.

Kod rezultata svm metoda vidimo veoma malu razliku između trening i test skupa, tako da možemo tvrditi da je generisan odličan model i da nismo pogrešili izborom linearnog jezgra. Jedina mana ovog metoda je nešto duže vreme generisanja modela u odnosu na drvo odlučivanja.

Kada gledamo statističke parametre vidimo da je nad trening podacima, drvo odlučivanja u proseku dalo 5% bolje rezultate nego knn, dok nad test podacima kada gledamo gene iznosi 3%, a nad ćelijama možemo primetiti da je knn dao bolje rezultate, što samo potvrđuje naše ranije navode o preprilagođavanju podataka kod stabla odlučivanja. Drvo odlučivanja se generisalo 143 puta brže od knn-a, a 65 puta brže u odnosu na svm metod. Najbolji rezultat uočavamo kod svm modela nad ćelijama, u kojim je rezultat test skupa neverovatnih 0.995, odnosno samo 2 pogrešno klasifikovana podatka od ukupno 429.

4.9 Ansambl metode

Kao i u delu 4.8 i ovde upoređujemo rezultate pojedinačnih metoda, sada iz ansambl grupe. Takođe uzimamo samo po jedan model sa parametrima koji su dali najbolje rezultate. Nasumičnu šumu generišemo sa 50 stabala bez dubinskog ograničenja, metod pakovanja izvršavamo pomoću 20 drveta odlučivanja, dok kod pojačavanja to radimo sa 50. Na kraju generisanje modela metoda glasanja postižemo pomoću 3 različita metoda, a to su nasumična šuma (50 stabala), k najbližih suseda (3 suseda) i mašine sa potpornim vektorima (linearni kernel).

Metoda	Atributi	Rezultat trening skupa	Rezultat test skupa	Vreme izvršavanja	Tačno klasifikovani	Pogrešno klasifikovani
Nasumična šuma	Ćelije	0.992	0.922	11.584	23878 + 9513	196 + 805
Nasumična šuma	Geni	1.000	0.993	0.9	999 + 426	0 + 3
Pakovanje	Ćelije	0.990	0.893	108.272	23825 + 9218	249 + 1100
Pakovanje	Geni	1.000	0.995	23.369	999 + 427	0 + 2
Pojačavanje	Ćelije	0.992	0.886	82.993	23882 + 9140	192 + 1178
Pojačavanje	Geni	1.000	0.958	1.765	999 + 411	0 + 18
Glasanje	Ćelije	0.979	0.924	3582.169	23559 + 9534	515 + 784
Glasanje	Geni	1.000	0.998	130.177	999 + 428	0 + 1

Tabela 8: rezultati ansambl metoda

Kao što je i očekivano, kod ansambl metoda dobili smo mnogo bolje rezultate u poređenju sa jednostavnim metodama, razlog tome je generisanje većeg broja jednostavnih ili čak ansambl modela radi dobijanja što preciznijih podataka, kako na trening tako i na test skupu. Možemo reći da se metod nasumične šume odlično pokazao i među ovim metodama ima druge po redu najbolje rezultate. Ova činjenica je još fascinantnija ako uzmemo u obzir vreme generisanja modela, koje kod metoda nad ćelijama iznosi samo 0.9s što je brže od bilo koje jednostavne ili ansambl metode koju smo do sada proučavali.

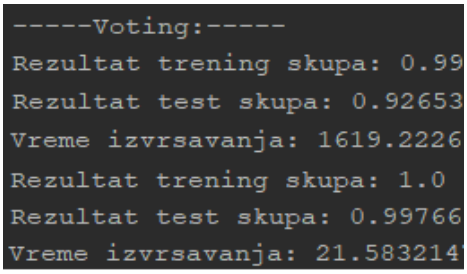
Rezultate i vremena izvršavanja metoda pakovanja i pojačavanja su dobra, ali lošija u poređenju sa nasumičnom šumom i glasanjem, tako da ćemo njih koristiti samo za poređenje i nećemo nadalje posebno obrađivati.

Vidimo da je metod glasanja uopšteno gledano dao najbolje rezultate, ali je cena plaćena u vremenu generisanja metoda. Jedan od glavnih razloga tolikog vremena izvršavanja je generisanje metoda k najbližih suseda i mašina sa potpornim vektorima.

//O statističkim parametrima smo diskutovali u prethodnim primerima, pa ovaj prepuštam čitaocu da sam odradi.

5 Zaključak

Jedna od osnovnih stvari koju možemo primetiti u prethodnim analizama je vreme potrebno da se generiše model metoda glasanja. Osnovni razlog toga je generisanje modela k najbližih suseda i mašina sa potpornim vektorima. Videli smo da metod k najbližih suseda daje prihvatljive, ali ne dovoljno dobre rezultate. Iz tog razloga je odlučeno da ponovo generišemo model glasanja, ali ovaj put kao jedan od tri metoda nije korišćen k najbližih suseda, već drvo odlučivanja. Dobijeni su sledeći rezultati:



Možemo primetiti da su rezultati delimično poboljšani, ali ono što se itekako popravilo je vreme generisanja modela i to nad genima 2.21 puta, a nad ćelijama 6 puta.

Kada generalno upoređujemo dve velike grupe metoda (jednostavne i ansambl) možemo uvideti da smo dobili očekivane odnose rezultate, odnosno da su rezultati ansambl metoda (nasumične šume, pakovanja i pojačavanja) za nijansu bolji od jednostavnih metoda, pre svega drveta odlučivanja koji je korišćen kao osnovni metod, takođe vidimo da dobijamo bolje rezultate u odnosu na k najbližih suseda, dok je kod mašina sa potpornim vektorima drugačija situacija i od ansambl metoda jedino glasanje ima bolje rezultate.

Uopštenom analizom metoda u kojima su atribute predstavljale ćelije vidimo da je stalno dolazilo do blagog preprilagođavanja, što je posledica velikog broja slogova nad kojim su građeni modeli. Kod metoda u kojima su atribute predstavljali geni imali smo mnogo manje slogova, samo 1428 naspram 34392 sloga u prethodnim navodima, uz to kod ovih metoda smo imali preko 17000 hiljada atributa, pa su metode mogle jasnije razgraničiti pripadnost jedno od dveju grupa. Iz rezultata zaključujemo da su metodi sa genima kao atributima u proseku davali 2.28% odnosno 8% bolje rezultate nad trening i test skupovima i uz to generisali modele 35.24 puta brže.

Na kraju možemo tvrditi da smo imali dobre podatke za klasifikaciju, da je pretprocesiranje urađeno na odgovarajući način, a što se metoda tiče, najbolji odnos rezultata i vremena dobili smo nasumičnom šumom, a uopšteno najbolje rezultate metodom glasanja.