Klasifikacija HEK293T ćelija
Istraživanje podataka 2
Matematički fakultet

David Gavrilović 294/2015

25. maj 2019

Sadržaj

1 Uvod

Zadatak je klasifikovati HEK293T ćelije bubrega. Na početku imamo sedam različitih datoteka od kojih svaka predstavlja zasebnu klasu. Klase, imena i dimenzije datih datoteka su sledeće:

- Klasa jedan, 061_HEK293T_human_embryonic_kidney_csv, dimenzija 31222 i 800
- Klasa dva, 065_HEK293T_human_embryonic_kidney_csv, dimenzija 31222 i 67
- Klasa tri, 066_HEK293T_human_embryonic_kidney_csv, dimenzija 31222 i 581
- Klasa četiri, 067_HEK293T_human_embryonic_kidney_csv, dimenzija 31222 i 4164
- Klasa pet, 068_HEK293T_human_embryonic_kidney_csv, dimenzija 31222 i 7875
- Klasa šest, 073_HEK293T-human_embryonic_kidney_matcsv, dimenzija 31222 i 3066
- Klasa sedam, 074_HEK293T-human_embryonic_kidney_csv, dimenzija 31222 i 526

Sadržaj datoteka se nalazi u CSV formatu. Klasa pet se sastoji od najviše primeraka dok ih klasa dva sadrži ubedljivo najmanje. U svakoj datoteci prva kolona sadrži nazive gena, njih 31222, od kojih je svaki različit. Ostatak kolona su vrednosti koje određuju taj gen.

Prvo ćemo izvršiti preprocesiranje nad datim podacima, nakon čega pristupamo klasifikaciji podataka. Za izvršavanje svih algoritama korišćen je programski jezik Pajton3 (eng. *Python3*) i biblioteke *pandas* [1] za čitanje, pisanje i manipulaciju podataka, *numpy* [2] i *scikit-learn* [3] za detekciju elemenata van granice i za klasifikaciju. Za iscrtavanje grafika korišćena je biblioteka *matplotlib* [4]. Programi su izvršavani na *Google Colaboratory* serveru [5] na kome je dostupno 12,7GB RAM memorije.

2 Preprocesiranje

Preprocesiranje uključuje tri koraka. Prvo spajamo sve datoteke u jednu, dodeljući svakoj datoteci dodatnu kolonu koja označava klasu. Pre samog spajanja potrebno je izvršiti i transponovanje sadržaja. Nakon toga, eliminišemo sve kolone koje imaju samo nule kao vrednost, i na kraju vršimo eliminaciju elemenata van granica.

2.1 Spajanje i eliminacija nula kolona

U ovoj sekciji se prvo bavimo transponovanjem sadržaja datoteka. Kao što smo već pomenuli, prva kolona svake datoteke sadrži nazive gena. Cilj nam je da, za svaku od datoteka, transponujemo sadržaj i da onda te nazive gena postavimo kao nazive atributa za svaku od kolona redom. Funkcija koja izvršava pomenuti proces je prikazana u kodu 1. Funkcija kao argument ima tabelu koju treba izmeniti, a povratna vrednost joj je promenjena tabela.

```
import pandas as pd

def transpose_and_set_column_names(df):
    # transpose
    df = df.T

    # set new column names and remove first column
    hg_names = df.iloc[0, :]
    df = df.iloc[1:, :]
    df.columns = hg_names

    return df
```

Kod 1: Transponovanje

Pokrenućemo datu funkciju za svaku od datoteka. Taj proces je prikazan za datoteku u kojoj se nalazi prva klasa u kodu 2. Rezultat funkcije se za klasu jedan čuva u promenljivoj *df061* nazvanoj po brojevima koji se nalaze na početku naziva datoteke. Analogno ćemo pozvati istu funkciju za ostale datoteke. Rezultat za klasu dva čuvamo u *df065*, za klasu tri u *df066* i tako redom.

```

df061 = pd.read_csv('061_klasa1_putanja.csv',
                                    index_col=False)
df061 = transpose_and_set_column_names(df061)
```

Kod 2: Primer transponovanja

Sledeće što je potrebno uraditi je tako transponovane tabele spojiti. Proces spajanja je prikazan u kodu 3. Prvo transponujemo svaku od tabela i onda izvršimo spajanje. Napomenimo i to da je spajanje moguće zbog toga što će svaka tabela imati isti skup atributa. Pre spajanja je potrebno i dodati dodatnu kolonu u svaku od tabela koja predstavlja klasu. Vrednost klase za prvu klasu će biti *class1*, za klasu dva *class2* i tako analogno za svaku od klasa.

```

# combine files
df = pd.concat([df061, df065, df066, df067,
            df068, df073, df074], axis=0, ignore_index=True)
```

Kod 3: Spajanje

Nakon spajanja tabela mogu se eliminisati nula kolone. Nula kolone su onakve kolone kojima je svaka vrednost nula. Takve kolone se eliminišu zato što

nam nisu od značaja. Nakon eliminacije dobijena tabela je sačuvana kao csv datoteka nazvana *combined_ data.csv*. Ceo proces je prikazan u kodu 4. Dobijena tabela ima 17079 redova i 22251 kolona.

```
# filter zoro columns
df.loc[:, (df != 0).any(axis=0)]

# save data frame
df.to_csv('combined_data.csv', index=False)
```

Kod 4: Eliminacija nula kolona

2.2 Eliminacija elemenata van granica

U ovoj sekciji će biti prikazan proces eliminacije elemenata van granica. Za uklanjanje će biti korišćen *Local outlier factor* ili skraćeno LOF [6]. Vrednost parametra koji predstavlja broj suseda, kao i vrednost faktora koji nam služi za eliminaciju su izabrani demonstrativno, ali na takav način da se za svaku od klasa uklone neki predstavnici. Podaci bez elemenata van granica su sačuvani u datoteku *data_ without_ outliers.csv*, dok su elementi van granica sačuvani u *outliers.csv*. Obe datoteke su u csv formatu. Proces je prikazan u kodu 5. Dimenzije tabele bez elemenata van granica su 16997 redova i 22251 kolona, dok tabela koja sadrži elemente van granica ima isti broj kolona, ali znatno manji broj redova, 82.

```

df = pd.read_csv('combined_data.csv', index_col=False)

lof = LocalOutlierFactor(n_neighbors=5)
lof.fit(df)
lof_factor = lof.negative_outlier_factor_

outlier_factor = 1.8
cluster_df = df[lof_factor >= -outlier_factor]
outlier_df = df[lof_factor < -outlier_factor]

cluster_df.to_csv('data_without_outliers.csv', index=False)
outlier_df.to_csv('outliers.csv', index=False)
```

Kod 5: Uklanjanje elemenata van granice

U tabeli 1 se može videti broj primeraka za svaku od klasa nakon eliminacije elemenata van granica kao i broj uklonjenih primeraka. Vidimo da je klasa kojoj je uklonjeno najviše elemenata klasa četiri, dok je ona sa najmanje uklonjenih elemenata klasa dva.

Klasa	Broj primeraka nakon uklanjanja	Broj uklonjenih primeraka
Klasa jedan	793	7
Klasa dva	66	1
Klasa tri	577	4
Klasa četiri	4133	31
Klasa pet	7853	22
Klasa šest	3060	6
Klasa sedam	515	11

Tabela 1: Broj primeraka za svaku od klasa u tabeli

3 Uvid u podatke

Nakon završenog preprocesiranja možemo izvršiti uvid u podatke. Dobijena tabela sadrži 16997 redova i 22251 kolona. Samo jedan atribut je kategorički, i on predstavlja klasu. Svi ostali podaci su numerički. Kao što je već napomenuto, imamo sedam klasa i predstavićemo ih u tabeli 2.

Klasa	Broj primeraka	Udeo u podacima
Klasa jedan	793	∼0.047
Klasa dva	66	∼0.0039
Klasa tri	577	∼0.034
Klasa četiri	4133	∼0.24
Klasa pet	7853	∼0.46
Klasa šest	3060	∼0.18
Klasa sedam	515	∼0.030

Tabela 2: Broj primeraka za svaku od klasa i njen udeo

Odavde možemo primetiti da najviše primeraka pripada klasi pet, oko 46 procenata od svih podataka. Klasa sa najmanje primeraka je klasa dva, čiji primerci zauzimaju samo oko 0.4 procenta od ukupnog broja svih primeraka. Ovo može da znači da će se predstavnici ove klase, ukoliko se značajno ne razlikuju od predstavnika ostalih klasa, teško precizno klasifikovati. Primetimo da se pomenuti problem može pojaviti i kod klasifikacije za klase jedan, tri i sedam, iz istog razloga. Zbog toga će biti isprobana i klasifikacija na skupu podataka sa balansiranim klasama.

4 Klasifikacija

Nakon završene pripreme podataka može se početi sa klasifikacijom. Prvo što je potrebno uraditi je učitati podatke i podeliti ih na dva skupa, trening i test. Podelu izvršavamo na taj način što uzimamo da nam trening skup 70% podataka, dok nam je skup za test, naravno, ostatak (30%). Pomenuti proces je prikazan u kodu 6. Parametar *random_state* u funkciji koja izvršava podelu na trening i test je izabran nasumično. Bitno je da taj parametar ima istu vrednost pri svakom pozivu funkcije za podelu na trening i test, zato što će nam tada

podela uvek biti ista.

```
1
2  from sklearn.model_selection import train_test_split
3
4  df = pd.read_csv('data_without_outliers.csv')
5
6  y = df['class']
7  X = df.loc[:, df.columns != 'class']
8
9  X_train, X_test, y_train, y_test = train_test_split(X, y,
10                                    test_size=0.3, random_state=27)
```

Kod 6: Priprema za klasifikaciju

Preciznost na test i trening skupu će biti korišćeni kao ocena kvaliteta modela. Kako preciznost može da bude veoma visoka iako model loše predviđa primerke onih klasa koje nemaju mnogo primeraka, biće korišćen i odziv (eng. *recall*). Odziv je procenat broja primeraka koji smo tačno klasifikovali u odnosu na broj koliko primeraka postoji. Takođe će se za svaki model ispisivati i matrica konfuzije.

4.1 K najbližih suseda

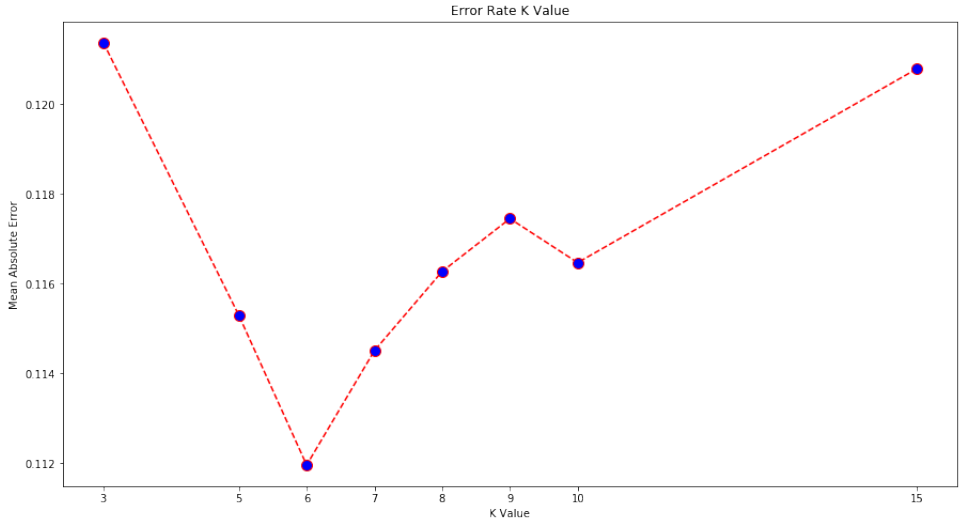
Za prvi model koji pravimo izabran je K najbližih suseda (eng. *K nearest neighbors*) [7] ili drugačije *KNN*. Na početku želimo da napravimo nekoliko modela sa različitim K parametrom, i onda da vidimo koji nam daje najbolje rezultate. Kao meru koliko nam je dobar model koristićemo srednje apsolutnu grešku. Vrednosti za parametar K koje želimo da isprobamo su 3, 5, 6, 7, 8, 9, 10 i 15. Izabrali smo te vrednosti zato što klasa dva ima samo 66 primeraka pa nema razloga isprobavati veće vrednosti za K. Takođe, ne želimo da nam pomenuti parametar ima previše nisku vrednost. Proces je prikazan u kodu 7.

```
1  error = []
2  ks = [3, 5, 6, 7, 8, 9, 10, 15]
3
4  for i in ks:
5      knn_i = KNeighborsClassifier(n_neighbors=i)
6      knn_i.fit(X_train, y_train)
7
8      pred_i = knn_i.predict(X_test)
9      error.append(np.abs(np.mean(pred_i != y_test)))
```

Kod 7: KNN za različitu vrednost K

Iscrtava se grafik koji prikazuje napravljenu grešku na trening skupu za svaku od vrednosti za parametar K. Grafik je prikazan na slici 1. Najbolji rezultat je dobijen za model koji klasifikuje uz pomoć šest suseda.

Sledeće model koji će biti isproban je takođe KNN, ali sada uvodimo težine. Za težine je izabrano rastojanje, odnosno, što je neki primerak neke klase bliži primerku koji klasifikujemo, to će više uticati na klasifikaciju. I ovde će biti napravljeno nekoliko modela za različit parametar K. Izbrane vrednosti za K su iste kao i kada smo testirali najbolje K za model bez težina. Kao ocena greške korišćena je srednje apsolutna greška na test skupu. Proces izvršavanja



Slika 1: Greška za različite vrednosti parametra K sa težinama

je prikazan u kodu 8. Nekon toga je iscrtan grafik koji nam prikazuje dobijenu grešku za svaku vrednost K parametra. Grafik je prikazan na slici 2.

```
error_weighted = []
ks = [3, 5, 6, 7, 8, 9, 10, 15]

for i in ks:
    knn_i = KNeighborsClassifier(n_neighbors=i,
                                 weights='distance')
    knn_i.fit(X_train, y_train)

    pred_i = knn_i.predict(X_test)
    error_weighted.append(np.abs(np.mean(pred_i != y_test)))
```

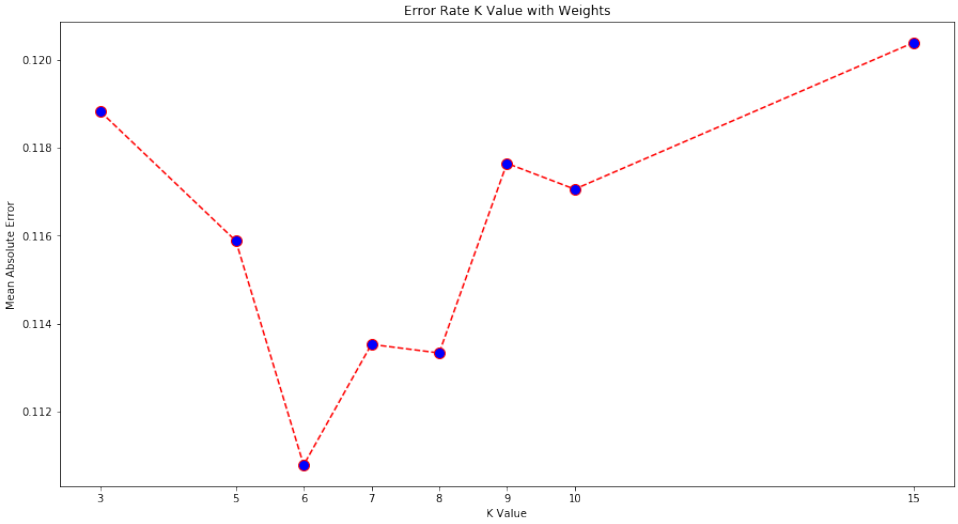
Kod 8: KNN sa težinama za različitu vrednost K

Sa datog grafika možemo zaključiti da nam je model sa najmanjom greškom onaj kome je vrednost parametra K šest. Štaviše, možemo primetiti da nam je model sa težinama dao malo bolji rezultat od onog koji ne koristi težine.

Sada ćemo napraviti naizgled najbolji KNN model, gde je broj suseda jednak šest i koji koristi rastojanje kao težine.

```

knn = KNeighborsClassifier(n_neighbors=6,
                           weights='distance')
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

Kod 9: najbolji KNN



Slika 2: Greška za različite vrednosti parametra K

Dobijeni su sledeći rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8892

Odziv: [0.8636 0.     0.0755 0.8231 0.9658 0.9831 0.7246]

Matrica konfuzije:
[[ 209    0    0    1    1   30    1]
 [   0    0    1    9    6    0    0]
 [   0    0   12  119   28    0    0]
 [   0    0    6 1061  222    0    0]
 [   0    0    1   79 2259    0    0]
 [   2    0    0    5    7  873    1]
 [   5    0    0    0    3   38  121]]
```

Odavde može da se zaključi da ovaj model ima visoku preciznost na test skupu, dok je preciznost na trening skupu očekivano 1,0. Problem je klasifikacija uzoraka klase dva i klase tri, što možemo primetiti iz odziva kao i iz matrice konfuzije. Štaviše, ovaj model nije uspeo da klasifikuje niti jedan primerak neke od klasa kao klasu dva, dok je većinu primeraka klase tri pogrešno klasifikovao kao primerke klase četiri.

4.2 Naivni Bajes

Seda će biti isprobani Naivni Bajes (eng. *Naive Bayes*), klasifikatori koji su zasnovani na verovatnoći [8]. Moguće je da nam ovi klasifikatori ne daju dobre rezultate zato što se primarno koriste za klasifikaciju teksta. Ovi modeli se obično konstruišu veoma brzo, pa će zbog toga biti isprobani. Prvi model koji konstruišemo je *MultinomialNB* [9]. Kod 10 je izbacio prikazane rezultate.

```
mnb = MultinomialNB()
mnb.fit(X_train, y_train)

y_pred = mnb.predict(X_test)
```

Kod 10: MultinomialNB

Rezultati:

```
Preciznost na trening skupu: 0.7131
Preciznost na test skupu: 0.6998

Odziv: [0.9587 0.0625 0.5786 0.6827 0.593  0.9347 0.8802]

Matrica konfuzije:
[[ 232    7    0    0    0    2    1]
 [   0    1    6    9    0    0    0]
 [   0    8   92   53    6    0    0]
 [   0   67  323  880   19    0    0]
 [   0   79  431  442 1387    0    0]
 [  29    1    0    2    0  830   26]
 [   9    0    1    0    0   10  147]]
```

Sledeći model je modifikacija prethodnog, *ComplementNB* [10]. Nakon pokretanja koda 11 dobili smo rezultate.

```
mnb = ComplementNB()
mnb.fit(X_train, y_train)

y_pred = mnb.predict(X_test)
```

Kod 11: ComplementNB

Rezultati:

```
Preciznost na trening skupu: 0.729
Preciznost na test skupu: 0.718

Odziv: [0.0289 0.     0.     0.8937 0.6939 0.991  0.    ]

Matrica konfuzije:
[[   7    0    0    9    0  226    0]
 [   0    0    0   13    3    0    0]
 [   1    0    0   86   72    0    0]
 [   0    0    0 1152  136    1    0]
 [   1    0    0  714 1623    1    0]
 [   5    0    0    3    0  880    0]
 [   1    0    0    0    0  166    0]]
```

Rezultati nisu loši, ali su gori nego za KNN model. Možemo primetiti da *ComplementNB* model ima malo veću preznost, ali ne klasifikuje dobro primerke svih klasa, dok *MultinomialNB* loše klasifikuje klase dva i tri.

4.3 Stabla odlučivanja

U ovom odeljku su prikazani modeli stabla odlučivanja (eng. *Decision trees*) [11]. Isprobana su dve metrike za kvalitet podele, gini i entropija. Nakon toga,

pošto nam klase nisu izbalansirane, konstruisani su modeli koji daju određene težine klasama. Prvo će biti isproban gini kao mera kvaliteta podele.

```
dtc = DecisionTreeClassifier(criterion='gini')
dtc.fit(X_train, y_train)

y_pred = dtc.predict(X_test)
```

Kod 12: Stablo odlučivanja za gini

Dobijeni sledeći rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8294

Odziv: [0.8099 0.     0.1572 0.7618 0.8965 0.9223 0.6647]

Matrica konfuzije:
[[ 196    0    0    1    3   33    9]
 [   0    0    1   12    3    0    0]
 [   0    7   25  101   26    0    0]
 [   1   16   88  982  201    0    1]
 [   1    2   54  179 2097    2    4]
 [  39    0    0    2    3  819   25]
 [  23    0    0    2    3   28  111]]
```

Nakon toga, pokrenut je isti algoritam, ali za entropiju.

```
dtc = DecisionTreeClassifier(criterion='entropy')
dtc.fit(X_train, y_train)

y_pred = dtc.predict(X_test)
```

Kod 13: Stablo odlučivanja za entropiju

Rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8431

Odziv: [0.7975 0.     0.2075 0.7696 0.9149 0.9358 0.6647]

Matrica konfuzije:
[[ 193    0    1    0    0   31   17]
 [   0    0    7    7    2    0    0]
 [   0    4   33   97   25    0    0]
 [   2   23   86  992  186    0    0]
 [   3    1   31  160 2140    3    1]
 [  30    0    0    0    1  831   26]
 [  20    0    0    0    2   34  111]]
```

Možemo primetiti da je algoritam koji je koristio entropiju dao malo veću preciznost na test skupu. Oba modela loše klasifikuju primerke klase dva i klase tri. Preciznosti na trening skupu su 1,0, odakle može da se zaključi da je možda donekle došlo do prepriloagodjavanja.

Sada će svakoj od klasa biti dodeljena neka težina, u pokušaju da se poboljšaju rezultati, posebno na klasama sa malim brojem primeraka. To možemo

uraditi postavljanjem parametra *class_ weight* na vrednost *balanced*. To znači da će se svakoj klasi dodeliti težina izračunata kao $N/(M*K)$, gde je N ukupan broj primeraka svih klasa, M broj klasa, dok je K broj primeraka klase kojoj se dodeljuje težina [12]. Ponovo konstruišemo modele za gini i za entropiju, ali ovaj put sa inicijalizovanim težinama.

```
dtc = DecisionTreeClassifier(criterion='gini',
                             class_weight='balanced')
dtc.fit(X_train, y_train)

y_pred = dtc.predict(X_test)
```

Kod 14: Stablo odlučivanja za gini sa težinama

Rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8047

Odziv: [0.7686 0.125  0.2138 0.7067 0.882  0.911  0.5928]

Matrica konguzije:
[[ 186    0    0    1    1   39   15]
 [   0    2    3   11    0    0    0]
 [   0    4   34   85   36    0    0]
 [   0   18   99  911  258    3    0]
 [   3    3   41  227 2063    2    0]
 [  45    0    0    1    1  809   32]
 [  16    0    0    0    2   50   99]]
```

Model koji koristi entropiju.

```
dtc = DecisionTreeClassifier(criterion='entropy',
                             class_weight='balanced')
dtc.fit(X_train, y_train)

y_pred = dtc.predict(X_test)
```

Kod 15: Stablo odlučivanja za entropiju sa težinama

Rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8233

Odziv: [0.7769 0.0625 0.1698 0.7192 0.8961 0.9414 0.7425]


Matrica konfuzije:
[[ 188    0    0    0    6   39    9]
 [   0    1    2   10    3    0    0]
 [   0    4   27   98   30    0    0]
 [   1   16   93  927  251    1    0]
 [   2    6   32  198 2096    5    0]
 [  27    0    0    0    1  836   24]
 [  11    0    1    0    1   30  124]]
```

Dodavanje težina klasama nam je smanjilo preciznost na oba modela. Možemo primetiti da se odziv za klase dva i tri povećao kod modela koji kao meru kvaliteta podele koristi gini, dok isto važi za klasu dva u modelu sa entropijom, ali se u tom modelu odziv za klasu tri smanjio.

4.4 SVM

Sledeći metod koji će biti isproban je metod potpornih vektora (eng. *Support vector machine*), u daljem tekstu SVM [13]. Konstruisani su modeli za tri različita kernela, *rbf*, *polinomijalni* i *linearni* kernel [14]. Proces konstrukcije i prikaz rezultata je prikazan u nastavku.

4.4.1 Rbf kernel

Na početku je konstruisan rbf kernel. Funkcija rbf se definiše kao $rbf(x) = \exp(-\gamma\|x - x'\|^2)$, gde vrednost parametra gama mora biti veća od nule. U ovom slučaju, gama parametar će biti postavljen na vrednost *'scale'*, tačnije $\gamma = 1/(N * M)$, gde je N broj karakteristika, dok je M varijansa skupa nad kojim gradimo model. Proces je prikazan u kodu 16. Gama definiše uticaj jednog primerka iz trening skupa. Ako je vrednost parametra visoka, onda je uticaj veći, a ako je niska, onda se uticaj smanjuje [13].

```
clf = SVC(C=100, kernel='rbf', gamma='scale')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 16: rbf kernel; C je 100

Dobijeni su sledeći rezultati:

```
Preciznost na trening skupu: 0.9781
Preciznost na test skupu: 0.9555

Odziv: [0.9669 0.25   0.7421 0.9054 0.9914 0.9955 0.8802]

Matrica konfuzije:
[[ 234    0    0    0    0    8    0]
 [   0    4    0    7    5    0    0]
 [   0    1  118   24   16    0    0]
 [   0    1   13 1167  108    0    0]
 [   0    0    4   16 2319    0    0]
 [   2    0    0    0    2  884    0]
 [   2    0    0    1    0   17  147]]
```

Rezultati su veoma dobri, štaviše, najbolji su do sada. Ovaj model dobro klasifikuje klasu tri, dok ne daje dobre rezultate pri klasifikaciji klase dva. Treba napomenuti da je i pored toga, ovaj model tačno pogodio najviše primeraka klase dva od svih modela do sada.

Sada će se posmatrati uticaj paramatra C na klasifikaciju. Parametar C kontroliše koliko razdvajajuća hiper-ravan dopušta pogrešnu klasifikaciju primeraka na trening skupu. Što je vrednost ovog paremetra veća, više primeraka iz trening skupa će se tačno klasifikovati [13]. U kodu 17 je povećana vrednost parametra C.

```
clf = SVC(C=300, kernel='rbf', gamma='scale')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 17: rbf kernel; C je 300

Dobijeni su sledeći rezultati:

```
Preciznost na trening skupu: 0.993
Preciznost na test skupu: 0.9588

Odziv: [0.9835 0.1875 0.7484 0.9255 0.9846 0.9966 0.8922]

Matrica konfuzije:
[[ 238    0    0    0    0    4    0]
 [   0    3    0    9    4    0    0]
 [   0    0  119   26   14    0    0]
 [   0    1   12 1193   83    0    0]
 [   0    0    4   32 2303    0    0]
 [   2    0    0    0    0  885    1]
 [   3    0    0    0    0   15  149]]
```

Rezultati su slični kao i za manju vrednost C. Preciznost i na trening i na test skupu su povećane, ali za malu vrednost. Ono što možemo primetiti je i da je odziv za klasu dva malo niži u modelu sa većim C.

Sada ćemo dodeliti težine svakoj od klasa i ponoviti ceo proces ponovo. Težine se dodeljuju na isti način kao i kod stabla odlučivanja. Prvo će biti prikazan kod i rezultat za model gde je C parametar 100, a nakon toga isto to ali za veću vrednost C.

```
clf = SVC(C=100, kernel='rbf', gamma='scale',
                              class_weight='balanced')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 18: rbf kernel sa balansiranim klasama; C je 100

```
Preciznost na trening skupu: 0.9817
Precinzost na test skupu: 0.9524

Odziv: [0.9876 0.1875 0.7799 0.9279 0.9675 0.9932 0.8982]

Matrica konfuzije:
[[ 239    0    0    0    0    3    0]
 [   0    3    3    8    2    0    0]
 [   0    1  124   24   10    0    0]
 [   0    1   31 1196   61    0    0]
 [   0    2    5   69 2263    0    0]
 [   3    0    0    2    0  882    1]
 [   2    0    0    1    0   14  150]]
```

```
clf = SVC(C=300, kernel='rbf', gamma='scale',
                              class_weight='balanced')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 19: rbf kernel sa balansiranim klasama; C je 300

```
Preciznost na trening skupu: 0.9895
Preciznost na test skupu: 0.9561

Odziv: [0.9793 0.1875 0.761  0.9465 0.9658 0.9955 0.9102]

Matrica konfuzije:
[[ 237    0    0    0    0    5    0]
 [   0    3    2    9    2    0    0]
 [   0    1  121   27   10    0    0]
 [   0    1   17 1220   51    0    0]
 [   1    1    6   72 2259    0    0]
 [   3    0    0    0    0  884    1]
 [   1    0    0    0    0   14  152]]
```

Oba modela su nam dali slične rezultate. Takođe se može primetiti da nema neke velike razlike između modela sa i bez balansiranje klasa.

Za kraj ove sekcije biće napravljen i testiran model sa balansiranim klasama, ali sa nižom vrednošću za C od ostalih.

```
clf = SVC(C=10, kernel='rbf', gamma='scale',
                              class_weight='balanced')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

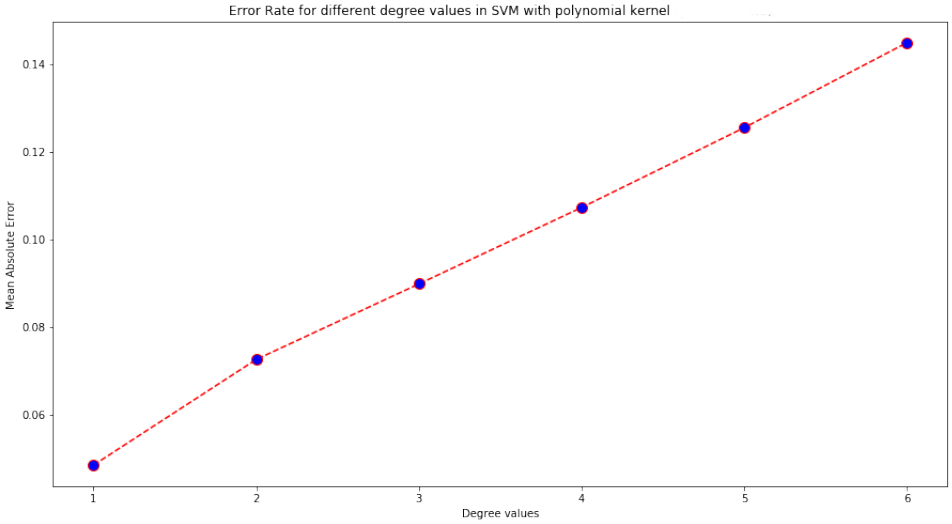
Kod 20: rbf kernel sa balansiranim klasama; C je 10

```
Preciznost na trening skupu: 0.9485
Preciznost na test skupu: 0.9318

Odziv: [0.9959 0.5625 0.8239 0.851  0.9778 0.9459 0.8802]

Matrica konfuzije:
[[ 241    0    0    0    0    1    0]
 [   0    9    2    3    2    0    0]
 [   0    6  131   10   12    0    0]
 [   0   29   46 1097  117    0    0]
 [   0   28   14   10 2287    0    0]
 [  46    0    0    1    1  840    0]
 [  14    0    0    0    0    6  147]]
```

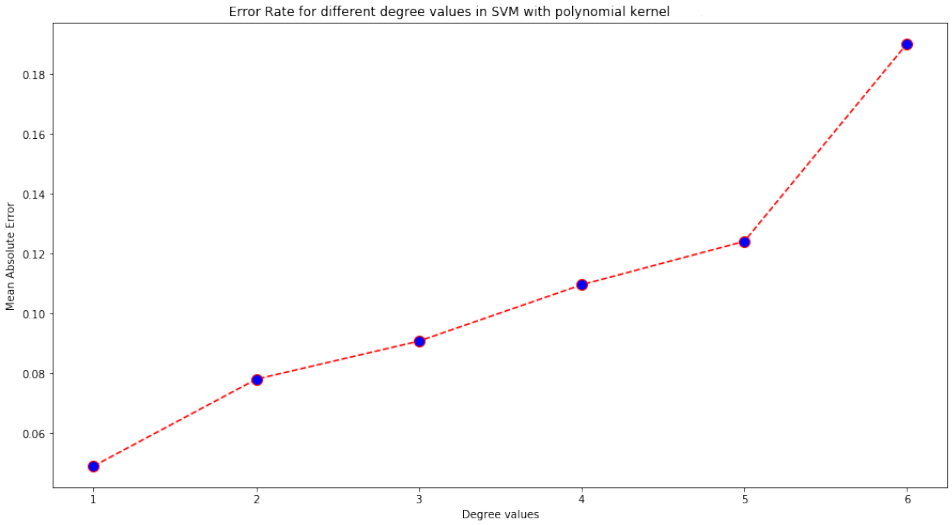
Ovaj model ima manju preciznost i na test i na trening skupu od ostalih. Tačno je pogodio više primeraka klasa dva i tri od ostalih modela, ali je takođe više primeraka ostalih klasa pogrešno klasifikovao kao jednu od te dve klase. Odavde može da se zaključi da je ovaj model gori od ostalih.

4.4.2 Polinomijalni kernel

U ovoj sekciji su konstruisani modeli SVM sa polinomijalin kernelom. Kernel je definisan kao $poly(x) = (\gamma\langle x, x'\rangle + r)^d$. Gama parametar je isti kao i kod rbf kernela. Parametar d određuje stepen polinoma, dok parametar r predstavlja koeficijent. Konstrisano je nekoliko modela za polinom različitog stepena, od prvog do šestog. Rezultati su prikazani na slici 3 za nebalansiran pristup, i na slici 4 za balansirane klase.



Slika 3: Greška za različite stepene polinomijalnog kernela



Slika 4: Greška za različite stepene polinomijalnog kernela za balansirane klase

Kod oba pristupa najmanja greška se dobija za polinom stepena jedan. Takođe izgleda da model bez balansiranja klasa pravi manju grešku na trening skupu. Oba pristupa za polinom stepena jedan će biti testirana. Kod svih modela gama parametar je postavljen na vrednost *'scale'*. Počećemo sa nebalansiranim modelom kome je postavljena vrednost parametra C na 100.

```
clf = SVC(C=100, kernel='poly', gamma='scale',
                        degree=1, class_weight=None)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 21: polinomijalni kernel; C je 100

```
Preciznost na trening skupu: 0.9673
Preciznost na test skupu: 0.9516

Odziv: [0.9504 0.375  0.6981 0.8953 0.9953 0.991  0.8623]

Matrica konfuzije:
[[ 230    0    0    0    0   11    1]
 [   0    6    0    4    6    0    0]
 [   0    1  111   29   18    0    0]
 [   0    1    9 1154  125    0    0]
 [   0    0    3    8 2328    0    0]
 [   1    0    0    3    3  880    1]
 [   1    0    0    1    1   20  144]]
```

Model sa nebalansiranim klasama, ali sa povećanom vrednošću za C.

```
clf = SVC(C=300, kernel='poly', gamma='scale',
                        degree=1, class_weight=None)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 22: polinomijalni kerne; C je 300

```
Preciznost na trening skupu: 0.9787
Preciznost na test skupu: 0.9547

Odziv: [0.9752 0.5    0.6918 0.9085 0.988  0.9944 0.8982]

Matrica konfuzije:
[[ 236    0    0    0    0    6    0]
 [   0    8    0    3    5    0    0]
 [   0    2  110   29   18    0    0]
 [   0    2   11 1171  105    0    0]
 [   0    0    6   22 2311    0    0]
 [   2    0    0    1    1  883    1]
 [   1    0    0    1    0   15  150]]
```

Dobijeni modeli su veoma dobri. Onaj koji ima veću vrednost parametra C je precizniji i na trening i na test skupu. Takođe taj isti je tačno klasifikovao pola primeraka klase dva iz test skupa, a da pri tome nije mnogo primeraka ostalih klasa klasifikovao kao primerke klase dva.

U nastavku su prikazani modeli sa istim parametrima, samo sa balansiranim težinama klasa.

```
clf = SVC(C=100, kernel='poly', gamma='scale',
                        degree=1, class_weight='balanced')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 23: polinomijalni kernel sa balansiranim klasama; C je 100

```
Preciznost na trening skupu: 0.9716
Preciznost na test skupu: 0.951

Odziv: [0.9876 0.5625 0.7547 0.8906 0.9842 0.9921 0.9042]

Matrica konfuzije:
[[ 239    0    0    0    0    3    0]
 [   0    9    1    2    4    0    0]
 [   0    5  120   18   16    0    0]
 [   0   10   36 1148   95    0    0]
 [   0    5   11   21 2302    0    0]
 [   2    0    0    4    0  881    1]
 [   3    0    0    2    0   11  151]]
```

```
clf = SVC(C=300, kernel='poly', gamma='scale',
                        degree=1, class_weight='balanced')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 24: polinomijalni kernel sa balansiranim klasama; C je 300

```
Preciznost na trening skupu: 0.981
Preciznost na test skupu: 0.9473

Odziv: [0.9835 0.375  0.6981 0.8968 0.9778 0.9932 0.9042]

Matrica konfuzije:
[[ 238    0    0    0    0    4    0]
 [   0    6    2    4    4    0    0]
 [   0    4  111   28   16    0    0]
 [   0    9   35 1156   89    0    0]
 [   0    3   12   37 2287    0    0]
 [   3    0    0    2    0  882    1]
 [   5    0    0    1    0   10  151]]
```

Ovi modeli daju dobre rezultate, ali, za razliku od nebalansiranih modela, više primeraka drugih klasa su pogrešno klasifikovali kao primerke klase dva.

4.4.3 Linearni kernel

Linearni kernel je definisan kao $lin(x) = \langle x, x' \rangle$. Prvi model koji je konstruisan u ovoj sekciji je predstavljen u kodu 25.

```
clf = SVC(C=100, kernel='linear', gamma='scale')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 25: linearni kernel; C je 100

Uz pomoć tog modela dobili smo sledeće rezultate.

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.9384

Odziv: [0.9793 0.375  0.6667 0.8991 0.9594 0.9955 0.8982]

Matrica konfuzije:
[[ 237    0    0    0    0    5    0]
 [   0    6    2    3    5    0    0]
 [   0    4  106   30   19    0    0]
 [   0    8   41 1159   81    0    0]
 [   0    3   20   72 2244    0    0]
 [   3    0    0    0    0  884    1]
 [   3    0    0    0    0   14  150]]
```

Dobijeni rezultati su dobri, ali model ne prepoznaje dobro primerke klase dva. Sada će model sa balansiranim težinama klasa biti konstruisan.

```
clf = SVC(C=100, kernel='linear', gamma='scale',
                                  class_weight='balanced')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 26: linearni kernel sa balansiranim klasama; C je 100

Rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.9384

Odziv: [0.9793 0.375  0.6667 0.8991 0.9594 0.9955 0.8982]

Matrica konfuzije:
[[ 237    0    0    0    0    5    0]
 [   0    6    2    3    5    0    0]
 [   0    4  106   30   19    0    0]
 [   0    8   41 1159   81    0    0]
 [   0    3   20   72 2244    0    0]
 [   3    0    0    0    0  884    1]
 [   3    0    0    0    0   14  150]]
```

Primetimo da su dobijeni rezultati apsolutno jednaki. Ništa se ne menja ni pri promeni parametra C, ni za balansiran ni za ne balansiran pristup.

4.5 Neuronske mreže

U ovoj sekciji se za klasifikaciju koriste modeli Neuronskih mreža (eng. *Neural networks*) [15]. Biblioteka *skikit-learn* omogućava tri načina za treniranje neuronskih mreža (rešavača):

- *lbfgs*, tehnika optimizacije zasnovana na Kvazi-njutnovim metodama
- *sgd*, stohastički gradijentni spust
- *adam*, optimizaciona tehnika zasnovana na stohastičkom gradijentnom spustu

Očekivanja su da će se model koji koristi *adam* dobro pokazati, iz tog razloga što taj rešavač obično daje dobre rezultate za velike skupove podataka, koji imaju po nekoliko hiljada trening podataka, kao što je naš [16].

Funkcije aktivacije koje će biti korišćene su:

- *Relu*, $f(x) = max(0, x)$
- *Logistička*, $f(x) = 1/(1 + \exp(-x))$
- *Tangens hiperbolički*, $f(x) = \tanh(x)$

Prvi model u ovoj sekciji će biti neuronska mreža sa podrazumevanim vrednostima. To znači da će kao rešavač koristiti *adam*, dok je funkcija aktivacije *relu*. Ova neuronska mreža ima jedan skriveni sloj sa 100 neurona.

```
clf = MLPClassifier()
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 27: Neuronska mreža sa podrazumevanim parametrima

Rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.9337

Odziv: [0.9545 0.     0.3836 0.8984 0.9765 0.9921 0.8802]

Matrica konfuzije:
[[ 231    0    0    0    0   11    0]
 [   0    0    1   10    5    0    0]
 [   0    0   61   61   37    0    0]
 [   0    0    3 1158  128    0    0]
 [   0    0    1   54 2284    0    0]
 [   3    0    0    2    2  881    0]
 [   2    0    0    0    5   13  147]]
```

Model ima visoku preciznost na trening i test skupu, ali loše klasifikuje primerke klase dva i klase tri.

Sada ćemo isprobati različite rešavače. Model neuronske mreže će imati dva skirvena sloja, u kojima se redom nalaze $N/16$ i $N/64$ neurona, gde je N broj atributa na trening skupu. Funkcija aktivacije je relu. Na žalost, mašina na

kojoj se pokreću modeli nema dovoljno RAM memorije da bismo pokrenuli *lbfgs* rešavač. Prvi model koristi stohastički gradijentni spust (kod 28).

```
clf = MLPClassifier(solver='sgd', activation='relu',
                    hidden_layer_sizes=(n // 16, n // 64))
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 28: Stohasti;ki gradijentni spust; Relu

Rezultati:

```
Preciznost na trening skupu: 0.4664
Preciznost na test skupu: 0.4612

Odziv: [0.0496 0.     0.     0.     0.9996 0.0023 0.    ]

Matrica konfuzije:
[[  12    0    0    0  230    0    0]
 [   0    0    0    0   16    0    0]
 [   0    0    0    0  159    0    0]
 [   0    0    0    0 1289    0    0]
 [   0    0    0    0 2338    1    0]
 [   1    0    0    0  885    2    0]
 [   0    0    0    0  166    1    0]]
```

Ovaj pristup nam nije dao dobra rešenja. Proverićemo da li će *adam* rešavač doneti bolja rešenja.

```
clf = MLPClassifier(solver='adam', activation='relu',
                    hidden_layer_sizes=(n // 16, n // 64))
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 29: Adam; Relu

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.9488

Odziv: [0.9876 0.     0.4717 0.9279 0.9782 0.9955 0.9401]

Matrica konfuzije:
[[ 239    0    0    0    0    3    0]
 [   0    0    0   11    5    0    0]
 [   0    0   75   47   37    0    0]
 [   0    0    9 1196   84    0    0]
 [   0    0    6   45 2288    0    0]
 [   3    0    0    0    0  884    1]
 [   4    0    0    0    1    5  157]]
```

Dobili smo znatno poboljšanje. Ovaj model nije klasifikovao nijedan primerak iz test skupa kao primerak klase dva. Takođe, loše je klasifikovao i primerke klase tri. Sledeći model će koristiti isti rešavač, kao i isti broj skrivenih slojeva sa istim brojem neurona, ali sa drugačijom funkcijom aktivacije. Koristimo tangens hiperbolički.

```
clf = MLPClassifier(solver='adam', activation='tanh',
                    hidden_layer_sizes=(n // 16, n // 64))
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 30: Adam; Tanh

```
Preciznost na trening skupu: 0.706
Preciznost na test skupu: 0.6739

Odziv: [0.0868 0.     0.     0.0853 0.9675 0.9944 0.9581]

Matrica konfuzije:
[[  21    0    0    0    0    2  219]
 [   0    0    0    2   14    0    0]
 [   0    0    0   11  148    0    0]
 [   0    0    0  110 1179    0    0]
 [   0    0    0   76 2263    0    0]
 [   2    0    0    0    0  883    3]
 [   5    0    0    0    0    2  160]]
```

Tangens hiperbolički ima gori rezultat od Relu funkcije. Sledeći model koristi logističku funkciju. Kod i rezultat su prikazani u nastavku.

```
clf = MLPClassifier(solver='adam', activation='logistic',
                    hidden_layer_sizes=(n // 16, n // 64))
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 31: Adam; Logistic

```
Preciznost na trening skupu: 0.9631
Preciznost na test skupu: 0.9251

Odziv: [1.     0.     0.     0.8914 0.9812 0.9921 0.9042]

Matrica konfuzije:
[[ 242    0    0    0    0    0    0]
 [   0    0    0    6   10    0    0]
 [   0    0    0   26  133    0    0]
 [   0    0    4 1149  136    0    0]
 [   0    0    2   42 2295    0    0]
 [   5    0    0    0    1  881    1]
 [   1    0    0    0    0   15  151]]
```

Preciznost je dosta viša u ovom modelu nego u modelu koji kao funkiju aktivacije koristi tangens hiperbolički, ali je niža nego u modelu sa relu funkcijom. Možemo primetiti da je odziv za klasu jedan na trening skupu jednak jedan, što zanči da je sve primerke koji pripadaju klasi jedan klasifikovao kao takve. I ovaj model loše klasifikuje primerke klasa dva i tri.

Odavde možemo zaključiti da nam modeli sa adam rešavačem i relu funkcijom aktivacije daju najbolje rezultate. Sada ćemo testirati nekoliko modela

sa drugačijim brojem skrivenih slojeva, kao i sa drugičijim brojem neurona na svakom od njih. Najbolja dva su prikazana u nastavku.

```
clf = MLPClassifier(solver='adam', activation='relu',
                    hidden_layer_sizes=(n // 32, n // 64))
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 32: Adam; Relu; n/32; n/64

```
Preciznost na trening skupu: 0.9972
Preciznost na test skupu: 0.9488

Odziv: [0.9835 0.     0.522  0.9279 0.9756 0.9955 0.9341]

Matrica konfuzije:
[[ 238    0    0    0    0    4    0]
 [   0    0    1   11    4    0    0]
 [   0    0   83   44   32    0    0]
 [   0    2    7 1196   84    0    0]
 [   0    0    0   57 2282    0    0]
 [   2    0    0    1    1  884    0]
 [   3    0    0    0    0    8  156]]
```

```
clf = MLPClassifier(solver='adam', activation='relu',
                    hidden_layer_sizes=(n // 16, n // 64, n // 128))
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 33: Adam; Relu; n/16; n/64; n/128

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.9543

Odziv: [0.9835 0.1875 0.6415 0.9286 0.9765 0.9955 0.9521]

Matrica konfuzije:
[[ 238    0    1    0    0    3    0]
 [   0    3    0    8    5    0    0]
 [   0    9  102   30   18    0    0]
 [   0    6   23 1197   63    0    0]
 [   0    1   10   44 2284    0    0]
 [   2    0    2    0    0  884    0]
 [   5    0    0    0    2    1  159]]
```

Poslednje napravljen model ima najvišu preciznost na test skupu od svih konstruisanih modela neuronskih mreža. Treba napomenuti da klasifikuje bolje primerke klase tri od ostalih. Ovaj klasifikator nije dao dobre rezultate pri klasifikaciji klase dva.

4.6 Ansambl metode

U ovom odeljku ćemo pokušati da, korišćenjem Ansambl metoda (eng. *Ensemble methods*) [17], poboljšamo već konstruisane modele. Pored toga, biće

uvedeni i neki novi modeli. Krenućemo od Nasumičnih šuma.

4.6.1 Nasumične šume

Prvi konstruisan model koji koristi ansambl metode je model Nasumičnih šuma (eng. *Random forests*) [17]. Kao i kod stabla odlučivanja, konstruisani su modeli koji kao meru kvaliteta podele koriste gini i entropiju. Nakon toga, za iste mere podele, konstruisani su modeli sa balansiranim klasama. Svaki konstruisan model koristi 1000 estimatora u izgradnji modela. Prvo je konstruisan model sa nebalansiranim klasama koji koristi gini kao meru podele.

```
rfc = RandomForestClassifier(n_estimators=1000,
                                    criterion='gini')
rfc.fit(X_train, y_train)

y_pred = rfc.predict(X_test)
```

Kod 34: Nasumične šume; gini

Rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8431

Odziv: [0.6322 0.     0.     0.7432 0.9461 0.9831 0.6168]

Matrica konfuzije:
[[ 153    0    0    0   12   76    1]
 [   0    0    0   10    6    0    0]
 [   0    0    0  121   38    0    0]
 [   0    0    0  958  331    0    0]
 [   0    0    0  126 2213    0    0]
 [   1    0    0    1   12  873    1]
 [   1    0    0    0    1   62  103]]
```

Dobijen rezultat ima veću preciznost nego rezultat koji je dobijen na modelu stabla odlučivanja za gini (kod 12). Razlika je i u tome što ovaj model ne klasifikuje nijedan primerak iz test skupa kao primerak klase dva ili klase tri, što nije slučaj za stablo odlučivanja, sa tim što stablo pravi dosta grešaka prilikom klasifikacije uzoraka te dve klase. Sada čemo videti rezultate nasumičnih šuma kada se kao mera kvaliteta podele koristi entropija.

```
rfc = RandomForestClassifier(n_estimators=1000,
                                    criterion='entropy')
rfc.fit(X_train, y_train)

y_pred = rfc.predict(X_test)
```

Kod 35: Nasumične šume; entropija

Rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8394

Odziv: [0.6198 0.     0.     0.7292 0.9461 0.9831 0.6287]
```

```
Matrica konfuzije:
[[ 150    0    0    1   11   79    1]
 [   0    0    0   10    6    0    0]
 [   0    0    0  122   37    0    0]
 [   0    0    0  940  349    0    0]
 [   0    0    0  126 2213    0    0]
 [   0    0    0    0   14  873    1]
 [   2    0    0    0    2   58  105]]
```

Dobijeni rezultati su dobri, ali ni ovaj model ne klasifikuje primerke klasa dva i tri dobro. Preciznost na test skupu ovog modela je malo niža nego na modelu stabla odlučivanja koji koristi entropiju kao meru podele. Rezultati tog modela se mogu videti ispod koda 13.

Sada će biti prikazani rezultati istih modela, redom za gini pa za entropiju, ali sa dotatim balansiranim težinama klasama.

```
rfc = RandomForestClassifier(n_estimators=1000,
        criterion='gini', class_weight='balanced')
rfc.fit(X_train, y_train)

y_pred = rfc.predict(X_test)
```

Kod 36: Nasumične šume sa balansiranim klasama; gini

Rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8333

Odziv: [0.6198 0.     0.     0.7184 0.9401 0.9786 0.6347]

Matrica konfuzije:
[[ 150    0    0    0   12   79    1]
 [   0    0    0   11    5    0    0]
 [   0    0    0  119   40    0    0]
 [   0    0    0  926  363    0    0]
 [   0    0    0  140 2199    0    0]
 [   1    0    0    0   17  869    1]
 [   0    0    0    0    3   58  106]]
```

```
rfc = RandomForestClassifier(n_estimators=1000,
        criterion='entropy', class_weight='balanced')
rfc.fit(X_train, y_train)

y_pred = rfc.predict(X_test)
```

Kod 37: Nasumične šume sa balansiranim klasama; entropija

Rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8273

Odziv: [0.6157 0.     0.     0.6974 0.9444 0.9662 0.6228]
```

```
Matrica konfuzije:
[[ 149    0    0    0   16   76    1]
 [   0    0    0   10    6    0    0]
 [   0    0    0  119   40    0    0]
 [   0    0    0  899  390    0    0]
 [   0    0    0  130 2209    0    0]
 [   0    0    0    0   29  858    1]
 [   0    0    0    0    7   56  104]]
```

Preciznost na test skupu je opala u ovim modelima u odnosu na modele bez balansiranih klasa, ali je malo veća u odnosu na modele stabla odlučivanja sa istim parametrima. Ni ovi modeli ne klasifikuju primerke klasa dva i tri.

4.6.2 Gradient boosting

Gradijent busting (eng. *Gradient boosting*) [17] je sledeći isprobani pristup. Biblioteka koju koristimo nam dozvoljava da biramo izmedju dve funkcije gubitka, *deviance* i *exponential*. Druga se može koristiti ukoliko imamo dve klase. Iz tog razloga biće korišćena samo prva, koja je definisana kao linearna regresija $y = ax + b + e$, gde je e greška. Kao mera kvaliteta podele korišćena je fridmanova srednje kvadratna greška (eng. *Friedman mse*), što je poboljšana srednje kvadratna greška. To je i podrazumevana vrednost. Pokrenuto je nekoliko modela sa različitim brojem estimatora, i svakim povećanjem dobijena je veća preciznost. U kodu 38 je prikazana izgradnja modela za 700 estimatora.

```
1 gbc = GradientBoostingClassifier(loss='deviance',
2                                  n_estimators=700)
3 gbc.fit(X_train, y_train)
4
5 y_pred = gbc.predict(X_test)
```

Kod 38: Gradijent busting

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.9267

Odziv: [0.9339 0.     0.2956 0.8991 0.9714 0.9955 0.8263]

Matrica konfuzije:
[[ 226    0    0    0    1   14    1]
 [   0    0    2    8    6    0    0]
 [   0    1   47   96   15    0    0]
 [   1    3   14 1159  112    0    0]
 [   0    4    6   57 2272    0    0]
 [   3    0    0    0    1  884    0]
 [   1    0    0    0    0   28  138]]
```

Precniznost na test i trening skupu je visoka. Kao i većina modela do sada, ni ovaj model ne klasifikuje dobro primerke klasa dva i tri iz test skupa.

4.6.3 Bagging

U ovoj sekciji, korišćenjem beginga (eng. *Bagging*) [17] pokušavamo da poboljšamo rezultate prethodnih modela. Bitni parametri klasifikatora u biblioteci

koju koristimo su broj estimatora i maksimalan broj uzoraka. Broj estimatora nam govori od koliko zasebnih modela pravimo novi. Drugi parametar pokazuje koliko će primeraka svaki od estimatora koristiti. Ako je vrednost ceo broj onda će koristiti baš taj broj primeraka, a ako je između nula i jedan, onda će svaki od estimatora konstruisati model nad $K * N$ primeraka, gde je K taj broj između nula i jedan, a N broj primeraka u trening skupu.

Na početku primenjujemo beging na najbolji konstruisani KNN model. Vrednosti parametara nisu mogle biti postavljene na veće vrednost usled nedostatka RAM memorije na mašini na kojoj je model konstruisan.

```
knn = KNeighborsClassifier(n_neighbors=6,
                           weights='distance')

bclf = BaggingClassifier(base_estimator=knn,
                     n_estimators=4, max_samples=0.25)
bclf.fit(X_train, y_train)

y_pred = bclf.predict(X_test)
```

Kod 39: Beging K najbližih suseda

Dobijeni rezultati:

```
Preciznost na trening skupu: 0.9103
Preciznost na test skupu: 0.8637

Odziv: [0.6983 0.     0.0314 0.7983 0.953  0.9673 0.6826]

Matrica konfuzije:
[[ 169    0    0    1    8   64    0]
 [   0    0    0   14    2    0    0]
 [   0    0    5  122   31    0    1]
 [   0    0    1 1029  259    0    0]
 [   0    0    2  108 2229    0    0]
 [   6    0    0    9   13  859    1]
 [   6    0    0    3    5   39  114]]
```

Dobijena je manja preciznost i na trening i na test skupu nego kod KNN modela bez beginga. Preciznost bi se verovatno povećala ako bismo konstruisali model sa višim vrednostima za parametre.

Sledeći model koji poboljšavamo je SVM. Izabran je SVM sa polinomijalnim kernelom stepena jedan. Pri konstrukciji ovog modela takođe postoji memorijsko ograničenje zbog koga ne možemo da postavimo parametre na više vrednosti od postavljenjih.

```
svm = SVC(C=300, kernel='poly', gamma='scale',
                     degree=1, class_weight=None)

bclf = BaggingClassifier(base_estimator=svm,
                     n_estimators=10, max_samples=1.0)
bclf.fit(X_train, y_train)

y_pred = bclf.predict(X_test)
```

Kod 40: Beging SVM

Dobijeni rezultati:

```
Preciznost na trening skupu: 0.9776
Preciznost na test skupu: 0.9539

Odziv: [0.9711 0.3125 0.717  0.9077 0.9876 0.9955 0.8802]

Matrica konfuzije:
[[ 235    0    0    0    0    6    1]
 [   0    5    0    6    5    0    0]
 [   0    3  114   25   17    0    0]
 [   0    2   16 1170  101    0    0]
 [   1    0    6   22 2310    0    0]
 [   2    0    0    0    1  884    1]
 [   1    0    0    1    0   18  147]]
```

Model koji koristi beging i model koji ga ne koriste imaju veoma slične rezultate. Razlika u precinzosti na trening i test skupu je veoma mala. Model sa begingom lošije klasifikuje klasu dva, ali malo bolje klasifikuje klasu tri.

Na kraju ove sekcije primenjujemo beging na neuronsku mrežu. Izabran je jedan od boljih modela, funkcija aktivacije je relu, dok je rešavač adam. Ova neuronska mreža ima tri skrivena sloja. Broj neurona na svakim sloju zavisi od N, broja atributa na trening skupu. I ovde važi ograničenje nad parametrima iz prethodno pomenutih razloga.

```
clf = MLPClassifier(solver='adam', activation='relu',
        hidden_layer_sizes=(n // 16, n // 64, n // 128))

bclf = BaggingClassifier(base_estimator=clf,
                    n_estimators=3, max_samples=1.0)
bclf.fit(X_train, y_train)
```

Kod 41: Beging neuronske mreže

```
Preciznost na trening skupu: 0.9867
Preciznost na test skupu: 0.9508

Odziv: [0.9835 0.     0.522  0.9457 0.9752 0.9955 0.8623]

Matrica konfuzije:
[[ 238    0    0    0    0    4    0]
 [   0    0    2   12    2    0    0]
 [   0    0   83   59   17    0    0]
 [   0    0   20 1219   50    0    0]
 [   0    0    2   56 2281    0    0]
 [   3    0    0    0    1  884    0]
 [   2    0    0    0    1   20  144]]
```

Model koji koristi beging ima lošije rezultate. Najveća razlika se može primetiti u klasifikaciji klasa dva i tri, gde je model koji ne koristi beging tačno klasifikovao više primeraka, ali je više i grešio.

4.6.4 Boosting

Sada poboljšavamo prethodno konstruisane modele busting (eng. *Boosting*) metodom [17]. Korišćen je *AdaBoost* (*Adaptive boosting*) algoritam [17]. Kao

i beging i ovaj metod koristi nekoliko estimatora da bi konstruisao bolji model. AdaBoost ne možemo da primenimo na KNN modele iz tog razloga što ne podržavaju dodeljivanje težina uzorcima, što je potrebno da bi pomenuti algoritam radio. Isto važi i za neuronske mreže.

Počećemo sa stablima odlučivanja. Najbolji model je koristio entropiju, pa ćemo njega pokušati da poboljšamo.

```
dtc = DecisionTreeClassifier(criterion='entropy')

clf = AdaBoostClassifier(base_estimator=dtc,
                                  n_estimators=10)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 42: Busting stabla odlučivanja

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8441

Odziv: [0.8017 0.     0.2201 0.7719 0.9124 0.9358 0.6946]

Matrica konfuzije:
[[ 194    0    0    0    1   29   18]
 [   0    0    6    8    2    0    0]
 [   0    6   35   91   27    0    0]
 [   2   16   82  995  194    0    0]
 [   3    2   37  160 2134    3    0]
 [  26    0    0    0    0  831   31]
 [  14    0    0    0    3   34  116]]
```

Dobili smo malo poboljšanje u preciznosti. Odziv je malo viši, ili isti svaku od klasa. Pokrenuto je nekoliko modela za različit broj estimatora, i ovaj je dao najbolji rezultat, bolji čak i od modela sa više estimatora.

Primenjujemo busting na nasumične šume, modele koji su nastali begingom. Izabran je model sa najvećom preciznošću od svih modela nasumičnih šuma. Parametri i proces su prikazani u sledećem kodu.

```
rfc = RandomForestClassifier(n_estimators=1000,
                                    criterion='gini')

clf = AdaBoostClassifier(base_estimator=rfc,
                                    n_estimators=500)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 43: Busting stabla odlučivanja

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8422

Odziv: [0.6322 0.     0.     0.7448 0.9436 0.9809 0.6228]
```

```
Matrica konfuzije:
[[ 153    0    0    1   10   77    1]
 [   0    0    0   10    6    0    0]
 [   0    0    0  120   39    0    0]
 [   0    0    0  960  329    0    0]
 [   0    0    0  132 2207    0    0]
 [   1    0    0    1   14  871    1]
 [   2    0    0    0    3   58  104]]
```

Model na koji je primenjen busting i model na koji nije imaju gotovo identičan rezultat, i sve klase klasifikuju sa skoro istim odzivom.

4.6.5 Klasifikatori zasnovani na glasanju

Klasifikatori zasnovani na glasanju (eng. *Voting classifiers*) [17], koriste nekoliko različitih modela da bi izgradili novi. Postoje dve vrste glasanja, tvrdo (eng. *hard voting*) i meko (eng. *soft voting*). U prvom pristupu kada se klasifikuje neki primerak dobija se kojoj klasi pripada, dok u drugom, mekom, se dobijaju verovatnoće pripadnosti uzorka svakoj od klasa. Oba pristupa će biti korišćena u nastavku. [18]

Prvi model koji je konstruisan koristi četiri modela iz prethodnih sekcija. Ti modeli, kao i njihovi parametri su prikazani u kodu 44.

```
rfc = RandomForestClassifier(n_estimators=1000,
                             criterion='gini')
knn = KNeighborsClassifier(n_neighbors=6,
                             weights='distance')
mnb = MultinomialNB()
svc = SVC(C=300, kernel='poly', gamma='scale', degree=1,
                             class_weight=None)

vclf = VotingClassifier(estimators=[('RFC', rfc),
                    ('KNN', knn), ('MNB', mnb),
                    ('SVM', svc)], voting='hard')
vclf.fit(X_train, y_train)

y_pred = vclf.predict(X_test)
```

Kod 44: Glasanje

```
Preciznost na trening skupu: 0.9994
Preciznost na test skupu: 0.9249

Odziv: [0.9752 0.     0.4403 0.8937 0.9649 0.9899 0.7365]

Matrica konfuzije:
[[ 236    0    0    0    0    6    0]
 [   0    0    0   12    4    0    0]
 [   0    0   70   70   19    0    0]
 [   0    3    3 1152  131    0    0]
 [   0    0    4   78 2257    0    0]
 [   2    0    0    1    6  879    0]
 [   4    0    0    1    0   39  123]]
```

Precniznost je visoka, ali su klasa dva i klasa tri na test skupu loše klasifikovane. Sada ćemo dodati težine klasifikatorima od kojih pravimo model. Što model ima veću težinu, to je njegov glas bitniji u pravljenju zajednučkog modela. SVM model je dao najbolje rezultate od svih, tako da se njemu dodeljuje najveća težina, dok obrnuto važi za Bajesov klasifikator. Ostala dva bi trebalo da imaju veću težinu od Bajesa, ali nižu od SVM. Kako KNN model ima malo veću preciznost od nasumičnih šuma, njegov glas nam je bitniji. Modeli i težine se mogu videti u kodu 45.

```
rfc = RandomForestClassifier(n_estimators=1000,
                             criterion='gini')
knn = KNeighborsClassifier(n_neighbors=6,
                             weights='distance')
mnb = MultinomialNB()
svm = SVC(C=300, kernel='poly', gamma='scale', degree=1,
                             class_weight=None)

vclf = VotingClassifier(estimators=[('RFC', rfc),
                    ('KNN', knn), ('MNB', mnb),
                    ('SVM', svm)], voting='hard',
                    weights=[1.5, 1.7, 1, 2.5])
vclf.fit(X_train, y_train)

y_pred = vclf.predict(X_test)
```

Kod 45: Glasanje sa težinama

```
Preciznost na trening skupu: 0.9973
Preciznost na test skupu: 0.9371

Odziv: [0.9669 0.125  0.4465 0.8852 0.9846 0.9955 0.8623]

Matrica konfuzije:
[[ 234    0    0    0    0    7    1]
 [   0    2    0    9    5    0    0]
 [   0    0   71   68   20    0    0]
 [   0    0    3 1141  145    0    0]
 [   0    0    4   32 2303    0    0]
 [   0    0    0    1    3  884    0]
 [   2    0    0    1    0   20  144]]
```

Dodavanjem težina se povećala preciznost na test skupu, dok se na trening skupu smanjila, ali skoro neprimetno. Primetimo da ovaj model malo bolje klasifikuje primerke klase dva i tri nego model bez težina.

Sada će biti demonstrano meko glasanje. Kada imamo takav pristup, modeli koji učestvuju u glasanju moraju da podržavaju računanje verovatnoća pripadnosti primerka klasama. KNN model nije takav, pa ga ne možemo koristiti. Taj model je zamenjen stablom odlučivanja, koji kao meru kvaliteta podele koristi entropiju. Takođe, da bi SVM model mogao da računa verovatnoće, mora mu se postaviti parametar *probability* na istinitosnu vrednost.

```
rfc = RandomForestClassifier(n_estimators=1000,
                    criterion='gini')
dt = DecisionTreeClassifier(criterion='entropy')
```

```
mnb = MultinomialNB()
svc = SVC(C=100, kernel='poly', degree=1,
                    gamma='scale', probability=True)

vclf = VotingClassifier(estimators=[('RFC', rfc),
                    ('DT', dt), ('MNB', mnb),
                    ('SVM', svc)], voting='soft')
vclf.fit(X_train, y_train)
```

Kod 46: Meko glasanje

```
Preciznost na trening skupu: 0.9986
Preciznost na test skupu: 0.9284

Odziv: [0.9504 0.     0.522  0.8821 0.9679 0.9865 0.8683]

Matrica konfuzije:
[[ 230    0    0    0    0   11    1]
 [   0    0    2   12    2    0    0]
 [   0    0   83   63   13    0    0]
 [   2    1   37 1137  112    0    0]
 [   0    0    7   68 2264    0    0]
 [  10    0    0    1    0  876    1]
 [   5    0    0    0    0   17  145]]
```

Rezultati su dobri. Model ne klasifikuje dobro primerke klase dva. Da bismo demonstrirali kako funkcionišu verovatnoće, izabraćemo nekoliko nasumičnih primeraka iz test skupa i predvideti kojoj klasi pripadaju. Pokrećemo kod 47.

```
print('pattern1')
print(np.round(vclf.predict_proba([X_test.iloc[4444]]), 3))

print('primerak2')
print(np.round(vclf.predict_proba([X_test.iloc[800]]), 3))

print('primerak4')
print(np.round(vclf.predict_proba([X_test.iloc[2222]]), 3))

print('primerak5')
print(np.round(vclf.predict_proba([X_test.iloc[0]]), 3))
```

Kod 47: Predviđanje uz pomoć modela mekog glasanja

Dobijen je sledeći ispis:

```
primerak1
[[0.008 0.006 0.045 0.44  0.482 0.013 0.005]]
primerak2
[[0.006 0.    0.003 0.276 0.687 0.026 0.002]]
primerak3
[[0.007 0.    0.001 0.002 0.009 0.971 0.011]]
primerak4
[[0.42  0.002 0.157 0.042 0.151 0.193 0.035]]
```

Za svako predviđanje je dobijena lista, takva da je na i-tom mestu verovatnoća pripadnosti i-toj klasi datog primerka. Posmatrajmo verovatnoće primerka tri. Možemo videti da on pripada klasi šest sa verovatnoćom 0.971, što je

dosta visoko. Kada pogledamo kojoj klasi taj primerak pripada, stvarno dobijamo klasu šest. Greške u klasifikaciji mogu da nastanu kod primeraka kao što je primerak jedan. Vidimo da su njegove verovatnoće pripadnosti klasi četiri i klasi pet gotovo jednake, ali je verovatnoća za klasu pet veća pa će biti klasifikovan kao takav. Taj primerak je tačno klasifikovan. Interesantan je i rezultat predviđanja za primerak četiri, koji ne pripada nijednoj klasi sa visokom verovatnoćom. On je klasifikovan kao primerak klase jedan, što je netačno, zbog toga što je to stvarno primerak klase sedam, iako ima veoma nisku verovatnoću pripadnosti za tu klasu. Na kraju, primetimo i da je primerak dva tačno klasifikovan.

Na samom kraju ove sekcije, želimo da konstruišemo model glasanja od modela koji su nam dali baš dobre rezultate. Izabrana su četiri SVM modela, dva koriste polinomijalni kernel, dok druga dva koriste rbf. Modeli, parametri kao i težine modela se mogu videti u kodu 48. Izabrano je tvrdo glasanje.

```
svc1 = SVC(C=100, kernel='poly', degree=1, gamma='scale')
svc2 = SVC(C=300, kernel='poly', degree=1, gamma='scale')
svc3 = SVC(C=100, kernel='rbf', gamma='scale')
svc4 = SVC(C=300, kernel='rbf', gamma='scale')

vclf = VotingClassifier(estimators=[('SVC1', svc1),
        ('SVC2', svc2), ('SVC3', svc3), ('SVC4', svc4)],
        voting='hard', weights=[1, 1, 1, 1])
vclf.fit(X_train, y_train)

y_pred = vclf.predict(X_test)
```

Kod 48: Glasanje svm modela

Dobijeni su sledeći rezultati:

```
Preciznost na trening skupu: 0.9801
Preciznost na test skupu: 0.9576

Odziv: [0.9752 0.375  0.7673 0.9092 0.991  0.9944 0.8802]

Matrica konfuzije:
[[ 236    0    0    0    6    0]
 [   0    6    0    5    5    0    0]
 [   0    1  122   22   14    0    0]
 [   0    2   12 1172  103    0    0]
 [   0    0    4   17 2318    0    0]
 [   2    0    0    0    2  883    1]
 [   2    0    0    1    0   17  147]]
```

Dobijeni rezultati su dobri, i preciznost na trening i test skupovima je visoka. Ovaj model, iako se po tom pitanju ponaša bolje od većine do sada konstruisanih modela, nema dobar odziv pri klasifikaciji klase dva.

5 Balansiranje klasa

U prethodnoj sekciji je isproban veliki broj klasifikatora, i velika većina je loše klasifikovala primerke klase dva, iako je preciznost na test skupu bila visoka.

Ovo donekle važi i za klasu tri, ali su neki modeli, kao na primer SVM, za tu klasu dali zadovoljavajuće rezultate. Uzrok ovoga može da bude ili mali broj primeraka klasa koje nam predstavljaju problem, ili sličnost tih primeraka sa nekom drugom klasom, a može biti i jedno i drugo.

5.1 Naivan pristup

Kako nam problematična klasa dva zauzima približno 0.4% od ukupnog broja primeraka, smatraćemo je nebitnom, pa ćemo je zbog toga izbrisati. Dobijena je nova tabela sa 16931 redova i 22251 kolona. Nakon brisanja, preostale podatke delimo na trening i test skup, i primenjujemo neke od modela koji su nam dali najbolji rezultat. Kao i u prethodnoj glavi, trening skup će činiti 70% podataka, a test skup ostatak. Dobijeni rezultati će biti upoređeni sa onima koji su napravljeni nad podacima koji sadrže klasu dva [1].

5.1.1 K najbližih suseda

Konstruisaćemo najbolji KNN model iz prethodnog dela. Model za klasifikaciju koristi šest suseda, i koristi rastojanje kao težine. Dobijeni su sledeći rezultati na test skupu:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8902

Odziv: [0.8436 0.0755 0.8143 0.9668 0.9878 0.7401]

Matrica konfuzije:
[[ 205    0    1    0   35    2]
 [   0   12  122   25    0    0]
 [   0    9 1048  229    0    1]
 [   0    2   72 2239    2    1]
 [   2    0    5    3  887    1]
 [  10    0    0    1   35  131]]
```

Preciznost na test skupu je visoka, i malo je viša nego na modelu koji je napravljen sa klasom dva. Nema velike razlike u rezultatima između dva pomenuta modela. Oba loše klasifikuju klasu tri. Pokušaćemo da dobijemo bolje rezultate uz pomoć nekih drugih modela.

5.1.2 Stabla odlučivanja

Od modela stabla odlučivanja najbolji rezultat je dobijen kada je korišćena entropija kao mera kvaliteta podele, pa će taj model biti pokrenut i nad podacima bez klase dva. Prikazujemo rezultate:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8346

Odziv: [0.7737 0.2264 0.7498 0.8998 0.9321 0.7345]
```

[1] Kada se bude prikazivala matrica konfuzije u ovoj sekciji, druga kolona i druga vrsta će predstavljati klasu tri, treća kolona i treća vrsta klasu četiri i tako analogno za ostale klase. Prva kolona i prva vrsta i dalje predstavljaju klasu jedan

```
Matrica konfuzije:
[[ 188    1    0    2   33   19]
 [   0   36  100   23    0    0]
 [   0  117  965  205    0    0]
 [   0   38  190 2084    1    3]
 [  35    0    1    5  837   20]
 [  19    0    1    0   27  130]]
```

Preciznost se jeste povećala sa ovim pristupom ali je razlika nebitna (iznosi 0,005). Ovaj model je klasifikovao više primeraka u klasu tri tačno, ali ih je klasifikovao više i pogrešno. Konstruisaćemo prethodni model, ali sa dodatim težinama klasama, da bismo videli da li je došlo do nekih poboljšanja. Rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8285

Odziv: [0.8354 0.2893 0.7405 0.8873 0.9232 0.6949]

Matrica konfuzije:
[[ 203    0    0    2   30    8]
 [   0   46   86   27    0    0]
 [   0   97  953  236    0    1]
 [   0   38  218 2055    3    2]
 [  34    0    0    5  829   30]
 [  16    0    0    1   37  123]]
```

Ovaj model daje malo nižu preciznost na test skupu u odnosu na model bez balansiranih klasa, ali daje bolje rezultate od modela koji je napravljen nad podacima sa klasom dva. Ovaj model daje bolji odziv za klasu tri u odnosu na ostala dva modela.

5.1.3 SVM

Sada ćemo testirati kakve razultate dobijamo uz pomoć SVM modela. Izbrana su četiri modela, dva koriste rbf kernel s tim što jedan ima dodeljene težine klasama dok drugi nema. Za druga dva važi sve isto, samo je kernel drugačiji, polinomijalni. Vrednost parametra C je postavljena na 300. Prvo je konstruisan model sa rbf kernelom bez težina. Dobijeni rezultati su prikazani u nastavku.

```
Preciznost na trening skupu: 0.9905
Preciznost na test skupu: 0.963

Odziv: [0.9794 0.7736 0.9176 0.9883 0.9955 0.9435]

Matrica konfuzije:
[[ 238    0    0    0    5    0]
 [   0  123   30    6    0    0]
 [   0   11 1181   95    0    0]
 [   0    3   23 2289    0    1]
 [   2    0    1    0  894    1]
 [   4    0    0    0    6  167]]
```

Dobijeni su dobri rezultati za svaku od klasa, i slični su modelu konstruisanom nad podacima sa klasom dva. Nakon toga je pokrenut model sa istim

parametrima, ali sa dodeljenim balansiranum težinama svakoj od klasa. Rezultati:

```
Preciznost na trening skupu: 0.9921
Preciznost na test skupu: 0.9665

Odziv: [0.9794 0.7987 0.9549 0.9737 0.9944 0.9492]

Matrica konfuzije:
[[ 238    0    0    0    5    0]
 [   0  127   28    4    0    0]
 [   1   19 1229   38    0    0]
 [   0    9   51 2255    0    1]
 [   3    0    1    0  893    1]
 [   4    0    0    0    5  168]]
```

Preciznost i na trening i na test skupu nam se povećala uvođenjem težina. Ovaj model ima veći odziv za klasu tri od prethodnog. Ponovimo prethodni proces za polinomijalni kernel. Rezultati bez dodeljenih težina su sledeći:

```
Preciznost na trening skupu: 0.9805
Preciznost na test skupu: 0.9579

Odziv: [0.9712 0.6855 0.9029 0.9918 0.9955 0.9492]

Matrica konfuzije:
[[ 236    0    0    0    7    0]
 [   0  109   40   10    0    0]
 [   0   11 1162  113    0    1]
 [   0    1   18 2297    0    0]
 [   2    0    1    0  894    1]
 [   3    0    0    0    6  168]]
```

Dobri rezultati i visoka preciznost, ali je model sa rbf kernelom precizniji. Može se primetiti i značajnija razlika između pomenuta dva modela u odzivu za klasu tri. Ni ovaj model se ne razlikuje mnogo od modela konstruisanog nad podacima sa klasom dva. Kada dodelimo težine klasama dobijamo:

```
Preciznost na trening skupu: 0.9823
Preciznost na test skupu: 0.9531

Odziv: [0.9835 0.6918 0.8982 0.9814 0.9955 0.9605]

Matrica konfuzije:
[[ 239    0    0    0    4    0]
 [   0  110   38   11    0    0]
 [   0   34 1156   96    0    1]
 [   0   14   28 2273    0    1]
 [   3    0    1    0  894    0]
 [   5    0    0    0    2  170]]
```

Uvođenjem težina nam ja opala preciznost na test skupu u odnosu na model bez težina, ali za malu vrednost. Odziv na ovom modelu za klase jedan, tri i sedam je povećan, dok je odziv za klase četiri i pet opala. Ovaj model ima više pogrešno klasifikovanih primeraka drugih klasa kao primeraka klase tri od prethodnog modela.

5.1.4 Neuronske mreže

Izabran je model neuronskih mreža koji je imao najbolje rezultate, i konstruisan je ponovo, za nove podatke. Taj model kao funkciju aktivacije koristi relu funkciju i koristi adam rešavač. Ima tri skrivena sloja sa po $N/16$, $N/64$ i $N/128$ neurona, gde je N broj atributa na trening skupu. Taj model nam je dao sledeće rezultate:

```
Preciznost na trening skupu: 9964
Preciznost na test skupu: 0.9504

Odziv: [1.     0.4591 0.9332 0.9732 0.9944 0.9266]

Matrica konfuzije:
[[ 243    0    0    0    0    0]
 [   0   73   50   36    0    0]
 [   0   14 1201   71    0    1]
 [   0    0   61 2254    0    1]
 [   5    0    0    0  893    0]
 [   8    0    0    0    5  164]]
```

Model ima visoku preciznost i na trening i na test skupu, ali daje gore rezultate prilikom klasifikacije klase tri. Napomenimo da je tačno predvideo sve primerke klase jedan iz test skupa.

5.1.5 Klasifikatori zasnovani na glasanju

Ovde će biti isproban samo jedan model koji kombinuje različite SVM klasifikatore u jedan. Izabrani su neki od modela koji su nam do sada dali dobre rezultate. Modeli i parametri se mogu videti u kodu 49. Primenjeno je tvrdo glasanje bez dodatih težina klasifikatorima.

```
svc1 = SVC(C=100, kernel='poly', degree=1, gamma='scale')
svc2 = SVC(C=100, kernel='rbf', degree=1, gamma='scale',
                class_weight='balanced')
svc3 = SVC(C=300, kernel='rbf', gamma='scale',
                class_weight='balanced')
svc4 = SVC(C=300, kernel='rbf', gamma='scale')

vclf = VotingClassifier(estimators=[('SVC1', svc1),
                ('SVC2', svc2), ('SVC3', svc3),
                ('SVC4', svc4)], voting='hard')
vclf.fit(X_train, y_train)

y_pred = vclf.predict(X_test)
```

Kod 49: Glasanje svm modela bez klase dva

Rezultat:

```
Preciznost na trening skupu: 0.9909
Preciznost na test skupu: 0.9657

Odziv: [0.9835 0.805  0.9308 0.9853 0.9944 0.9379]
```

```
Matrica konfuzije:
[[ 239    0    0    0    4    0]
 [   0  128   25    6    0    0]
 [   0   17 1198   72    0    0]
 [   0    7   26 2282    0    1]
 [   3    0    1    0  893    1]
 [   4    0    0    0    7  166]]
```

Ovaj model je veoma dobar. Preciznost mu je visoka i na test i na trening skupu, kao i odziv za svaku od klasa. Na matrici konfuzije se jasno može primetiti da je dijagonala dominantna.

5.2 Dodavanje veštačkih primeraka

U ovoj sekciji ćemo pokušati da veštački generišemo dodatne primerke klasa koje su dosadašnji modeli loše klasifikovali, pošto nemaju mnogo primeraka. To je urađeno korišćenjem funkcije *resample* iz već pomenute biblioteke *scikit-learn*. Napomenimo to da se pomenuta funkcija može koristiti i za smanjivanje broja primeraka neke klase.

Prvo ćemo podeliti podatke na trening (70% podataka) i test (30% podataka), i onda ćemo izvršiti dodavanje. Ukoliko uradimo obrnuto može doći do toga da postoje isti primerci u test i trening skupu. To nije dobro iz tog razloga što klasifikator onda može da zapamti neke primerke pri treniranju, što može dovesti do preprilagođavanja podataka. [19]

Problem je zaključiti koliko primeraka treba da se doda da bi se dobilo poboljšanje. Za početak će biti postavljen novi broj primeraka klase dva na 400, zbog toga što je to slična vrednost broju primeraka klase tri. Dodavanje elemenata je prikazano u kodu 50. Promenljiva *class2* predstavlja izdvojene primerke klase dva.

```
class2_upsampled = resample(class2, random_state=27,
                            n_samples=400, replace=True)

upsampled = pd.concat([rest_of_the_data, class2_upsampled])
```

Kod 50: Dodavanje primeraka klase dva

5.2.1 SVM

Konstruisaćemo najbolje SVM modele nad novim podacima. Prvi SVM model će koristiti rbf kernel i vrednost parametra C će biti 300. Parametri se mogu videti u kodu 51.

```
clf = SVC(C=300, kernel='rbf', gamma='scale')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

Kod 51: SVM sa rbf kernelom nakon dodavanja primeraka klase dva

Dobijeni rezultati:

```
Preciznost na trening skupu: 0.9935
Preciznost na test skupu: 0.9592
```

```
Odziv: [0.9835 0.25   0.7421 0.9263 0.985  0.9966 0.8922]

Matrica konfuzije:
[[ 238    0    0    0    0    4    0]
 [   0    4    0    8    4    0    0]
 [   0    1  118   25   15    0    0]
 [   0    1   12 1194   82    0    0]
 [   0    1    5   29 2304    0    0]
 [   2    0    0    0    0  885    1]
 [   3    0    0    0    0   15  149]]
```

Ovaj model ima malo veću preciznost i na trening i na test skupu od SVM modela sa istim parametrima konstruisanog nad podacima kojima nisu dodati veštački generisani primerci klase dva. Takođe, ovaj model ima veću vrednost odziva za klasu dva. Primerke ostalih klasa klasifikuje skoro isto kao i malopre pomenuti model. Sada konstruišemo SVM sa polinomijalnim kernelon stepena jedan. Vrednost parametra C ovog modela je takođe 300. Dobijeni su rezultati:

```
Preciznost na trening skupu: 0.9801
Preciznost na test skupu: 0.9529

Odziv: [0.9711 0.4375 0.6792 0.9061 0.9876 0.9944 0.8922]

Matrica konfuzije:
[[ 235    0    0    0    0    7    0]
 [   0    7    0    4    5    0    0]
 [   0    4  108   29   18    0    0]
 [   0    6   10 1168  105    0    0]
 [   0    2    6   21 2310    0    0]
 [   2    0    0    1    1  883    1]
 [   1    0    0    1    0   16  149]]
```

Ovaj model ima malu nižu preciznost i na test i na trening skupu od prethodnog. Preciznost u odnosu na model koji je konstruisan nad podacima sa manjim brojem primeraka klase dva je približno ista, što ne važi za vrednost odziva za klasu dva, koji ima nižu vrednost. Možemo primetiti i da je dodavanjem primeraka klase dva porastao broj pogrešno klasifikovanih primeraka drugih klasa u klasu dva.

5.2.2 Klasifikatori zasnovani na glasanju

Pokušaćemo glasanjem između različith SVM modela da dobijemo bolji rezultat. Izabrana su dva modela sa polinomijalnim kernelom, i dva sa rbf kernelom. Parametri modela se mogu videti u kodu 52.

```
svc1 = SVC(C=100, kernel='poly', degree=1, gamma='scale')
svc2 = SVC(C=300, kernel='poly', degree=1, gamma='scale')
svc3 = SVC(C=100, kernel='rbf', gamma='scale')
svc4 = SVC(C=300, kernel='rbf', gamma='scale')

vclf = VotingClassifier(estimators=[('SVC1', svc1),
                        ('SVC2', svc2), ('SVC3', svc3),
                        ('SVC4', svc4)], voting='hard')
vclf.fit(X_train, y_train)
```

```
10
11 y_pred = vclf.predict(X_test)
```

Kod 52: Glasanje svm modela sa dodatnim primercima klase dva

Prethodni kod je ispisao sledeće:

```
Preciznost na trening skupu: 0.9814
Preciznost na test skupu: 0.9569

Odziv: [0.9711 0.4375 0.7547 0.9092 0.9906 0.9944 0.8743]

Matrica konfuzije:
[[ 235    0    0    0    0    7    0]
 [   0    7    0    4    5    0    0]
 [   0    3  120   22   14    0    0]
 [   0    3   12 1172  102    0    0]
 [   0    1    4   17 2317    0    0]
 [   2    0    0    0    2  883    1]
 [   2    0    0    1    0   18  146]]
```

Rezultati su dobri. Ako uporedimo klasifikaciju klase dva ovog modela sa klasifikacijom klase dva prethodna dva SVM modela možemo primetiti da je ovaj model tačno klasifikovao isto onoliko primeraka klase dva koliko je i model sa polinomijalnim kernelom (što je više od modela sa rbf kernelom), ali uz manji broj grešaka. Ovaj model takođe bolje klasifikuje klasu tri od prethodna dva. Odziv za svaku od klasa je visok, sem za klasu dva.

5.3 Jednak broj primeraka svih klasa

U ovoj sekciji testiramo model koji su konstruisani nad podacima koji imaju jednak broj primeraka svake od klasa. Napomenimo da postoji biblioteka koju možemo da koristimo da bismo balansirali podatke, ali sve funkcije koje balansiraju numeričke podatke rade samo u slučaju kada imamo dve klase, što kod naših podataka nije slučaj. Ta biblioteka se zove *imbalanced-learn* [20]. Naravno. veliko je pitanje koji broj primeraka treba izabrati. Taj broj je postavljen na 2500. Razlog zašto je, pre ikakvog testiranja, izabran baš taj broj je taj da će tabela koja se dobije na ovakav način moći da stane u RAM memoriju naše mašine, a i želimo da dodamo što manje veštačkih primeraka klasa, ali da taj broj ostane dovoljno veliki. Takođe, dobiće se približno isti broj podataka kao kod podataka kojima klase nisu balansirane.

5.3.1 Stabla odlučivanja

Prvi modeli koji konstruišemo nad novodobijenim podacima su modeli stabla odlučivanja. Prvo je konstruisan model koji kao meru kvaliteta podele koristi gini. Dobijeni su sledeći rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.8271

Odziv: [0.7851 0.     0.1698 0.7401 0.8961 0.9381 0.7066]

Matrica konfuzije:
```

```
[[ 190    0    0    0    3   37   12]
 [   0    0    2   11    3    0    0]
 [   0    0   27  104   28    0    0]
 [   0   14   90  954  230    0    1]
 [   0    3   33  194 2096    4    9]
 [  31    0    0    2    4  833   18]
 [  18    0    1    0    1   29  118]]
```

Dobijeni rezultati su dobri. Ovaj model ne klasifikuje klase dva i tri dobro, uprkos balansiranju klasa. Uporedimo ga sa modelom koji je konstruisan nad podacima koji nisu balansirani. Preciznost na test skupu ovog modela je zanemarljivo niža (razlika je 0,0023), ali ovaj model manje greši prilikom klasifikacije klasa dva i tri. Sada pokrećemo model koji kao meru kvaliteta podele koristi entropiju. Rezultati:

```
Preciznost na trening skupu: 1.0
Preciznost na test skupu: 0.838

Odziv: [0.7851 0.     0.1698 0.7541 0.9162 0.9347 0.6707]

Matrica konfuzije:
[[ 190    0    0    0   32   20]
 [   0    0    5    8    3    0    0]
 [   0    7   27   93   32    0    0]
 [   2   23   90  972  202    0    0]
 [   3    1   33  155 2143    3    1]
 [  32    0    1    0    0  830   25]
 [  21    0    0    0    2   32  112]]
```

Konstruisani model ima veću preciznost na test skupu od prethodnog. Kada se uporedi sa modelom sa istim parametrima koji je kosntruisan nad podacima bez balansiranja dobija se manja preciznost na test skupu, kao i malo niže vrednsoti odziva za svaku od klasa sem klase sedam.

5.3.2 SVM

Sada konstruišemo SVM modele nad novonastalim podacima. Prvi model koristi rbf kernel. Njegovi parametri se mogu videti u kodu 53.

```
1 clf = SVC(C=300, kernel='rbf', gamma='scale')
```

Kod 53: SVM sa rbf kernelom nad balansiranim podacima

Nakon izvršenog predviđanja dobijeni su sledeći rezultati:

```
Preciznost na trening skupu: 0.993
Preciznost na test skupu: 0.9588

Odziv: [0.9835 0.1875 0.7484 0.9255 0.9846 0.9966 0.8922]

Matrica konfuzije:
[[ 238    0    0    0    0    4    0]
 [   0    3    0    9    4    0    0]
 [   0    0  119   26   14    0    0]
 [   0    1   12 1193   83    0    0]
 [   0    0    4   32 2303    0    0]
```

```
 [   2    0    0    0    0  885    1]
 [   3    0    0    0    0   15  149]]
```

Rezultati su dobri. Model ne klasifikuje dobro klasu dva. Ako uporedimo ovaj model sa model sa istim parametrima konstruisanom nad ne balansiranim podacima, primećujemo da su rezultati identični. Sada konstruišemo SVM sa polinomijalnim kernelom. Parametri modela su prikazani u kodu 54

```
clf = SVC(C=300, kernel='poly', gamma='scale', degree=1)
```

Kod 54: SVM sa polinomijalnim kernelom nad balansiranim podacima

Uz pomoć ovog modela su dobijeni rezultati:

```
Preciznost na trening skupu: 0.9787
Preciznost na test skupu: 0.9547

Odziv: [0.9752 0.5    0.6918 0.9085 0.988  0.9944 0.8982]

Matrica konfuzije:
[[ 236    0    0    0    0    6    0]
 [   0    8    0    3    5    0    0]
 [   0    2  110   29   18    0    0]
 [   0    2   11 1171  105    0    0]
 [   0    0    6   22 2311    0    0]
 [   2    0    0    1    1  883    1]
 [   1    0    0    1    0   15  150]]
```

Ovaj model je takođe dobar. Klasifikuje klasu dva dosta bolje od prethodnog, ali isto tako ima niži odziv za klasu tri. I ovaj model ima identične rezultate kao model sa istim parametrima konstruisan nad ne balansiranim podacima. Napomenimo i to da ta dva modela bolje klasifikuju klasu dva od svih ostalih.

5.3.3 Gradient boosting

Na samom kraju nad balansiranim podacima konstruišemo *Gradient boosting* model. Korišćeno je 500 estimatora i funkcija gubitka je linearna regresija. Nakon konstrukcije modela predstavljenog u kodu 55 i izvršenog predviđanja, prikazani su sledeći rezultati.

```
gbc = GradientBoostingClassifier(loss='deviance',
                                 n_estimators=500)
```

Kod 55: Gradient boosting nad balansiranim podacima

```
Preciznost na trening skupu: 0.9998
Preciznost na test skupu: 0.9237

Odziv: [0.9298 0.     0.283  0.8914 0.9714 0.9955 0.8144]

Matrica konfuzije:
[[ 225    2    0    0    1   14    0]
 [   0    0    2    8    6    0    0]
 [   0    2   45   95   17    0    0]
 [   2    9   12 1149  117    0    0]
 [   0    5    4   58 2272    0    0]
```

```
[   2    0    1    0    1  884    0]
[   0    1    0    0    0   30  136]]
```

Model ima visoku preciznost i na trening i na test skupu ali loše klasifikuje klase dva i tri. Primerimo i to da je klasifikovao nekoliko primeraka kao primerke klase dva, ali svaki od njih pogrešno.

6 Klasifikacija elemenata van granica

Sada ćemo primeniti neke od modela koji su dali najbolje rezultate do sada na elemente van granica. Podaci koji predstavljaju elemente van granica se čuvaju u posebnoj datoteci u csv formatu sa 82 reda i 22251 kolona. Kako su modeli sa najboljim rezultatima SVM modeli primenićemo njih. Izabrani su modeli sa rbf i polinomijalnim kernelom konstruisani nad balansiranim podacima. Napomenimo da su nam ti modeli dali identične rezultate kao modeli sa istim parametrima konstruisani nad nebalansiranim podacima. Pored toga, na elemente van granica ćemo primeniti već pomenute modele, ali konstruisane nad podacima gde su veštački dodati primerci klase dva. Uporedićemo rezultate. Za svaki od modela če biti izračunati preciznost i matrica konfuzije. Počećemo sa SVM modelom sa rbf kernelom, parametar C je 300 dok je vrednost parametra gama postavljena na *scale* (vrednost *scale* je već objašnjena u sekciji *SVM* u poglavlju *Klasifikacija*). Dobijeni su sledeći rezultati:

```
Preciznost: 0.7683

Matrica konfuzije:
[[ 5  0  0  0  0  1  1]
 [ 0  0  0  1  0  0  0]
 [ 0  0  3  1  0  0  0]
 [ 0  3  1 25  1  0  1]
 [ 0  0  0  3 19  0  0]
 [ 0  0  0  0  0  5  1]
 [ 2  0  0  0  0  3  6]]
```

Ovaj model se dobro klasifikovao elemente van granica. Možemo primetiti da su klasa dva (napomenimo da ona ima samo jednog predstavnika) i klasa sedam loše klasifikovane. Sada se predviđaju elementi van granica uz pomoć SVM modela sa polinomijalnim kernelom stepena jedan. Vrednosti parametara gama i C su redom *scale* i 300. Rezultati:

```
Preciznost: 0.7317

Matrica konfuzije:
[[ 5  0  0  0  0  1  1]
 [ 1  0  0  0  0  0  0]
 [ 1  0  1  2  0  0  0]
 [ 1  1  2 25  2  0  0]
 [ 0  0  1  2 19  0  0]
 [ 2  0  0  0  0  4  0]
 [ 2  0  0  1  0  2  6]]
```

Rezultati koje nam je dao ovaj model su slični rezultatima prethodnog. Najveća rezlika je u tome što je ovaj model dosta primeraka pogrešno klasifikovao kao u klasu jedan.

Sada ćemo klasifikovati elemente van granica uz pomoć modela sa istim parametrima, ali konstruisanim nad nebalansiranim podacima sa veštački dodatim primercima klase dva. Rezultati dobijeni preko modela sa rbf kernelom:

```
Preciznost: 0.7073

Matrica konfuzije:
[[ 5  0  0  0  0  1  1]
 [ 0  0  0  1  0  0  0]
 [ 0  0  1  3  0  0  0]
 [ 1  3  2 24  1  0  0]
 [ 0  1  1  2 18  0  0]
 [ 2  0  0  0  0  4  0]
 [ 2  0  0  1  0  2  6]]
```

Rezultati dobijeni modelom sa polinomijalnim kernelom:

```
Preciznost: 0.7195

Matrica konfuzije:
[[ 5  0  0  0  0  1  1]
 [ 0  0  0  1  0  0  0]
 [ 0  1  2  1  0  0  0]
 [ 0  5  1 24  1  0  0]
 [ 0  2  0  3 17  0  0]
 [ 0  0  0  0  0  5  1]
 [ 2  0  0  0  0  3  6]]
```

Ovi modeli imaju nižu preciznost od prethodnih. Napomenimo to da nema mnogo elemenata van granica pa pogrešna klasifikacija jednog može lako da se primeti. Stoga, nije velika razlika između rezultata ovih modela i prethodnih.

7 Zaključak

U cilju klasifikacije podataka isproban je veliki broj različitih algoritama sa različitim parametrima. Na neke od modela su primenjene i ansambl metode, ali većinom bez poboljšanja. Razlog tome je nedovoljno memorije na mašini na kojoj su modeli konstruisani. Jedina ansambl metoda koja ima veoma dobre rezultate je glasanje.

Nakon toga, urađeno je balansiranje klasa, zbog loše klasifikacije primarno klase dva, ali i klase tri nekih modela. Dosta modela konstruisano nad takvim podacima imaju dobre rezultate, ali nije postignuto poboljšanje u klasifikaciji klase dva.

Najbolji rezultati su dobijeni uz pomoć SVM modela sa polinomijalnim i rbf kernelom. Takvi modeli su imali visoku preciznost i na test i na trening skupu i klasifikovali su klase dva i tri bolje od ostalih. SVM modeli imaju slične rezultate nezavisno od toga da li konstruisani nad podacima sa balansiranim klasama ili ne. Takođe, veoma dobri rezultati su dobijeni glasanjem između različitih SVM modela. Na samom kraju još jednom prikazujemo preciznost i matrice konfuzije tri najbolja modela i upoređujemo ih. To su redom SVM sa rbf kernelom (kod 56), SVM sa polinomijalnim kernelom stepena jedan (kod 57), i glasanje različitih SVM modela (kod 58).

```
clf = SVC(C=300, kernel='rbf', gamma='scale')
```

Kod 56: SVM; C je 300; rbf kernel

```
Preciznost na trening skupu: 0.9930
Preciznost na test skupu: 0.9588

Matrica konfuzije:
[[ 238    0    0    0    0    4    0]
 [   0    3    0    9    4    0    0]
 [   0    0  119   26   14    0    0]
 [   0    1   12 1193   83    0    0]
 [   0    0    4   32 2303    0    0]
 [   2    0    0    0    0  885    1]
 [   3    0    0    0    0   15  149]]
```

```
clf = SVC(C=300, kernel='poly', gamma='scale', degree=1)
```

Kod 57: SVM; C je 300; polinomijalni kernel

```
Preciznost na trening skupu: 0.9787
Preciznost na test skupu: 0.9547

Matrica konfuzije:
[[ 236    0    0    0    0    6    0]
 [   0    8    0    3    5    0    0]
 [   0    2  110   29   18    0    0]
 [   0    2   11 1171  105    0    0]
 [   0    0    6   22 2311    0    0]
 [   2    0    0    1    1  883    1]
 [   1    0    0    1    0   15  150]]
```

```
svc1 = SVC(C=100, kernel='poly', degree=1, gamma='scale')
svc2 = SVC(C=300, kernel='poly', degree=1, gamma='scale')
svc3 = SVC(C=100, kernel='rbf', gamma='scale')
svc4 = SVC(C=300, kernel='rbf', gamma='scale')

vclf = VotingClassifier(estimators=[('SVC1', svc1),
                            ('SVC2', svc2), ('SVC3', svc3),
                            ('SVC4', svc4)], voting='hard',
                            weights=[1, 1, 1, 1])
```

Kod 58: Glasanje SVM modela

```
Preciznost na trening skupu: 0.9801
Preciznost na test skupu: 0.9576

Matrica konfuzije:
[[ 236    0    0    0    0    6    0]
 [   0    6    0    5    5    0    0]
 [   0    1  122   22   14    0    0]
 [   0    2   12 1172  103    0    0]
 [   0    0    4   17 2318    0    0]
 [   2    0    0    0    2  883    1]
 [   2    0    0    1    0   17  147]]
```

Od ova tri modela prvi ima najvišu precinzost i na test i na trening skupu, ali najgore klasifikuje primerke klase dva. Drugi model ima najviše tačno pogođenih primeraka klase dva, ali najlošnije klasifikuje klasu tri. Model zasnovan na glasanju različitih SVM modela se najbolje pokazao, zato što ima visoku preciznost i na test i na trening skupu, a i klasifikacija klase dva mu je bolja od klasifikacije klase dva SVM modela sa rbf kernelom. Bolje klasifikuje klasu tri od ostalih modela.

Za sam kraj napomenimo i to da su modeli poput neuronskih mreža i gradient busting imali visoku preciznost, ali su imali ili nižu preciznost na test ili trening skupu, ili su dosta lošije klasifikovali primerke klasa dva i tri.

Literatura

[1] Pandas documentation. on-line at: http://pandas.pydata.org/pandas-docs/stable/.

[2] Numpy documentation. on-line at: https://www.numpy.org/devdocs/.

[3] Scikit-learn. on-line at: https://scikit-learn.org/stable/.

[4] Matplotlib. on-line at: https://matplotlib.org/.

[5] Google colaboratory. on-line at: https://research.google.com/colaboratory/faq.html and https://colab.research.google.com/notebooks/welcome.ipynb#recent=true.

[6] Phillip Wenig. Local outlier factor for anomaly detection. 2018. on-line at: https://towardsdatascience.com/local-outlier-factor-for-anomaly-detection-cc0c770d2ebe.

[7] Devin Soni. Introduction to k-nearest-neighbors. 2018. on-line at: https://towardsdatascience.com/introduction-to-k-nearest-neighbors-3b534bb11d26.

[8] Victor Roman. Naive bayes algorithm: Intuition and implementation in a spam detector. 2019. on-line at: https://towardsdatascience.com/naive-bayes-intuition-and-implementation-ac328f9c9718.

[9] Scikit-learn. Multinomial-naive-bayes. on-line at: https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes.

[10] Scikit-learn. Complement-naive-bayes. on-line at: https://scikit-learn.org/stable/modules/naive_bayes.html#complement-naive-bayes.

[11] Thushan Ganegedara. Intuitive guide to understanding decision trees. 2018. on-line at: https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb.

[12] Scikit-learn. Decision-tree-classifier. on-line at: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.

[13] Rushikesh Pupale. Support vector machines(svm) - an overview. 2018. on-line at: https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989.

[14] Scikit-learn. Kernel functions. on-line at: https://scikit-learn.org/stable/modules/svm.html#kernel-functions.

[15] Suryansh S. Neural networks: All you need to know. 2018. on-line at: https://towardsdatascience.com/nns-aynk-c34efe37f15a.

[16] Scikit-learn. Mlpclassifier. on-line at: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier.

[17] Joseph Rocca and Baptiste Rocca. Ensemble methods: bagging, boosting and stacking. 2019. on-line at: https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205.

[18] Scikit-learn. Votingclassifier. on-line at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html.

[19] Tara Boyle. Dealing with imbalanced data. 2019. on-line at: https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18.

[20] Imbalanced-learn documentation. on-line at: https://imbalanced-learn.readthedocs.io/en/stable/.