

PROJEKAT

Klasifikacija raka dojke pomoću mašinskog učenja

Nemanja Janković

Beograd, avgust 2023. godine

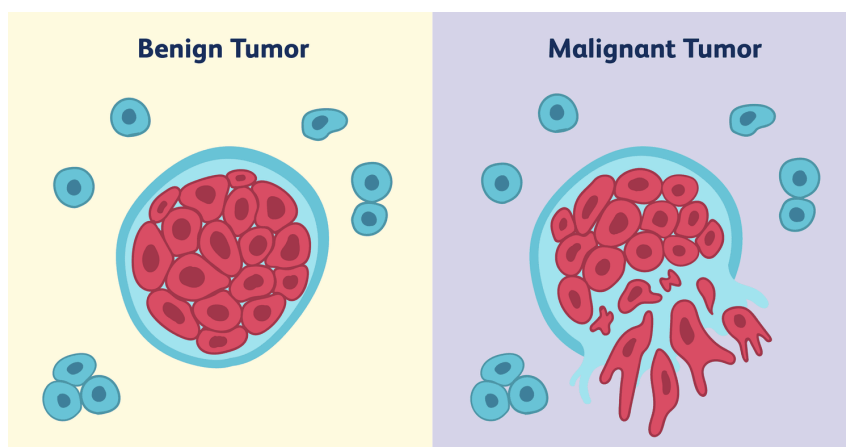
SADRŽAJ

1 UVOD	3
1.1 O raku dojke	3
1.2 O mašinskom učenju	4
2 ALGORITMI MAŠINSKOG UČENJA	5
2.1 Stabla odlučivanja	5
2.2 Slučajne šume	6
2.3 AdaBoost	7
2.4 XGBoost	8
2.5 Logistička regresija	9
2.6 Naivni Bajes	10
3 BAZA PODATAKA	11
3.1 Informacije o bazi	11
3.2 Atributi baze	11
3.3 Tehnike za izbor atributa	12
4 PROGRAMSKI KOD	13
4.1 Biblioteke i main funkcija	13
4.2 Analiza baze	14
4.3 Izbor atributa od značaja	27
4.4 Podela podataka na trening i test skup	28
4.5 Treniranje i testiranje	29
5 POREĐENJE KLASIFIKATORA	30
6 ZAKLJUČAK	33
7 LITERATURA	34

1 UVOD

1.1 O raku dojke

Rak dojke predstavlja najčešću malignu¹ bolest kod žena. Tokom 2020. godine, oko 2.3 miliona žena širom sveta dijagnostikovano je sa ovom bolešću dok 685 000 njih nije uspjelo da preživi. U Srbiji na godišnjem nivou oko 4600 žena oboli od ove bolesti, dok njih 1600 umre. Muškarci takođe mogu oboleti ali je kod žena stotinu puta češći. Kao i kod ostalih karcinoma, rak dojke odlikuje nekontrolisan rast i razmnožavanje ćelija koje u ovom slučaju nastaju u dojci i šire se po celom organizmu. Bolest se pojavljuje najčešće u periodu između 55. i 70. godine. Glavni uzrok tolike smrtnosti je nedovoljna informisanost i neredovni pregledi, kojima bi se bolest mogla identifikovati u ranim fazama i izlečiti sa visokim procentom uspešnosti. Najveći faktori rizika za razvoj raka dojke su: pripadnost ženskom polu, nedovoljna fizička aktivnost, gojaznost i konzumacija alkohola. Sa medicinske strane, benigni i maligni tumor se vizualno **razlikuju**. Maligni tumor je uglavnom nepravilnog oblika i veličine, tamnije boje jezgra, ćelije unutar njega su gusto zbijene dok su ostale rasute.



Slika 1 – Vizualni prikaz benignog i malignog tumora, preuzeto iz [1]

Nacionalni dan borbe protiv raka dojke je 20. mart. Maligne bolesti se leče raznim tehnikama: hirurškim zahvatima, hemoterapijom, zračenjem, kao i kombinovanim lečenjem. Važno je probuditi svest i ne zanemarivati simptome kako bi se bolest preventivno otkrila i uspešno izlečila.

¹ Maligne bolesti – zloćudne bolesti koje napadaju okolna tkiva i metastaziraju

1.2 O mašinskom učenju

Mašinsko učenje predstavlja podskup veštačke inteligencije² koja za cilj ima da iz postojećih podataka omogući računaru da izvuče i nauči bitne informacije i tako se adaptira novim podacima. Isti algoritam može rešiti dva potpuno različita problema.

Mašinsko učenje može biti:

- **Sa nadgledanjem** (*supervised learning*) – na raspolaganju postoje podaci sa tačnom klasom kojoj podatak pripada. Pri testiranju jasno je odrediti da li je algoritam doneo ispravnu odluku zato što je poznata tačna vrednost. Metode koje spadaju u učenje sa nadgledanjem su: klasifikacija (klasifikacija slika, otkrivanje prevare, dijagnostikovanje) i regresija (predviđanje, optimizacija procesa)
- **Bez nadgledanja** (*unsupervised learning*) – na raspolaganju postoje podaci ali nije poznato koji podatak kojoj klasi pripada. Metode koje spadaju u učenje bez nadgledanja su: klasterizacija (segmentacija) i redukcija dimenzija (vizualizacija podataka)
- **Učenje potkrepljivanjem** (*reinforcement learning*) – na raspolaganju postoje podaci pri čemu za svaku donešenu odluku postoji nagrada ukoliko je odluka ispravna ili kazna ukoliko je odluka pogrešna. Metode koje spadaju u učenje sa potkrepljivanjem su: donošenje odluka u realnom vremenu, igranje igara, navigacija robota...

² Veštačka inteligencija – podoblast računarstva koja za cilj ima da razvije algoritme koji će računarima omogućiti da se ponašaju na način koji se može smatrati inteligentnim. Obuhvata mašinsko učenje, prepoznavanje govora, kompjutersku viziju, kao i razne heurističke (optimizacione) metode.

2 ALGORITMI MAŠINSKOG UČENJA

2.1 Stabla odlučivanja

Stablo odlučivanja je neparametarski algoritam koji pripada učenju sa nadgledanjem koja se primenjuje na klasifikaciju i regresiju. Strukture su stabla i čine ga sledeće komponente:

- koreni čvor – čvor koji nema ulaznu granu, od njega kreće grananje stabla pomoću atributa koji najbolje deli podatke u nove što separabilnije skupove
- unutrašnji čvorovi (čvorovi odluke) – čvorovi koji se nalaze između korenog čvora i listova, u njima su obe klase prisutne i nastavljaju sa deljenjem skupa u nove separabilnije skupove
- listovi (terminalni čvorovi) – čvorovi koji nemaju izlazne grane, nalaze se na kraju stabla i u njima se donosi odluka o kojoj klasi je reč, do svakog lista se dolazi jedinstvenim putem
- grane – put koji povezuje čvorove, mogu biti izlazne (ukazuje na uslov koji će se primeniti nad skupom) i ulazne (ukazuju na uslov koji je zadovoljen prethodnim grananjem)

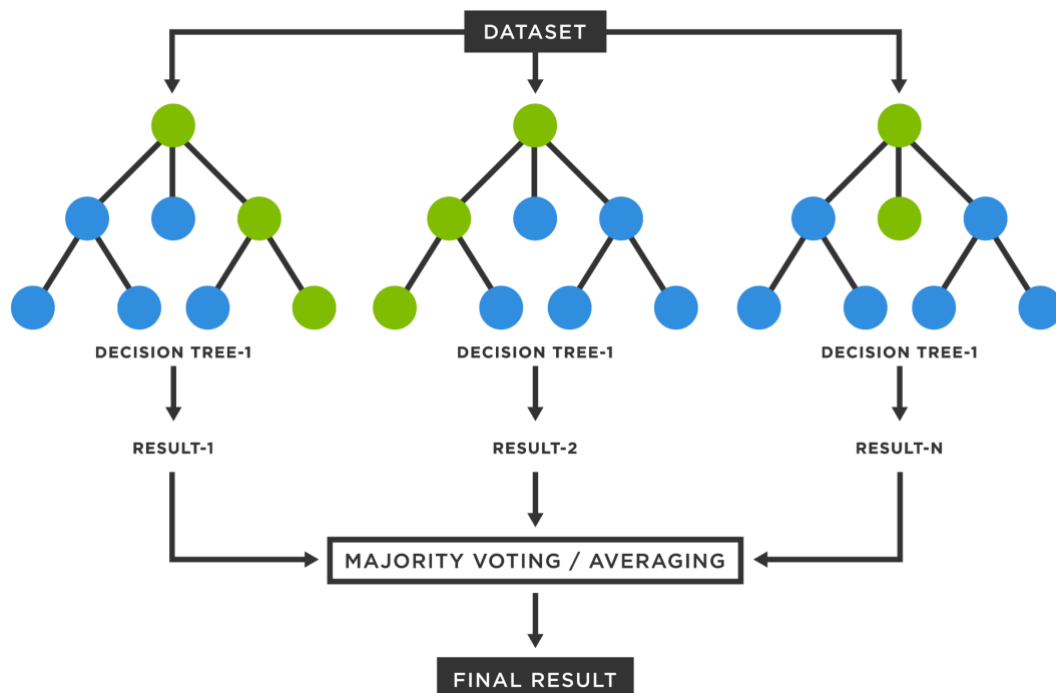
Ono što stablo odlučivanja radi je da svakom odlukom skup deli u homogenije podskupove kako bi na kraju u listovima donelo odluku o kojoj klasi je reč. Oni atributi koji skup deli u homogenije podskupove, nalaze se na višim delovima stabla. Što je stablo razgranatije i ima veću dubinu, to će bolje podeliti trenutne podatke, ali isto tako može doći do težeg prilagođavanja novim podacima jer je stablo previše naviknuto na postojeće podatke. Ovaj problem se može rešiti na više načina:

- *pre-pruning* i rano zaustavljanje – sprečavanje da stablo ne raste previše u širinu i dubinu
- *post-pruning* – stablo raste do maksimalne dubine i nakon toga sledi potkresivanje onih grana koje ispunjavaju zadati kriterijum (srednje-kvadratna greška za regresiona stabla i greška klasifikacije za klasifikaciona stabla)
- *ensembling* metode – usrednjavanje više modela

2.2 Slučajne šume

Slučajne šume spadaju u *ensemble learning* kojeg karakteriše kreiranje više modela i njihovo kombinovanje i agregacija kako bi se dobio jedinstven, unapređen rezultat. Konkretno, kod slučajnih šuma, model predstavlja stablo odluke koje može biti proizvoljne dubine. Koriste se masovno za klasifikaciju i regresiju. Kod klasifikacije, glas većine stabala biće konačan glas odluke, dok u slučaju regresije konačna odluka biće usrednjena vrednost svih stabala. U odnosu na obično stablo odluke, slučajne šume uglavnom daju bolje rezultate, manje su osetljive na trenirajući skup, rešavaju problem *overfitting-a*, dok su sa druge strane skupe i sporije.

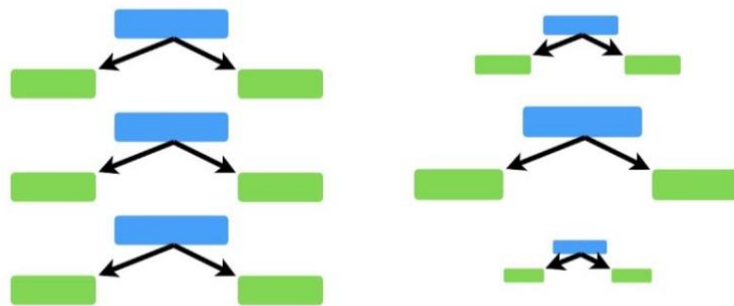
Potrebno je izabrati koliko stabla odluke se kreira, kao i njihova maksimalna dubina. Svako stablo sadrži proizvoljni skup atributa i proizvoljni skup uzoraka. Upravo u tome leži rešenje za *overfitting* i tačniju klasifikaciju, možda neko stablo neće dobro klasifikovati određeni odбирak, ali tu su ostala stabla koja će ga ispraviti.



Slika 2 – Prikaz slučajne šume, preuzeto iz [2]

2.3 AdaBoost

AdaBoost (*Adaptive Boosting*) model spada u učenje sa nadgledanjem i u *ensemble learning*. Sastavljen je od slabih modela koji zajedno agregirani daju jak i tačan model. Njegova najčešća implementacija je pomoću stabla odluke prvog nivoa, što znači da poseduje samo koren i njegova grananja. Kao *boosting* algoritam, bitan je redosled pravljenja stabala. Cilj je da uči iz prethodnih grešaka i napravi novi model koji će prevazići te greške. Na početku svaki podatak ima istu težinu. U odnosu na grešku koju prave stabla odluke, dodeljuju im se različite težine (veća težina se daje podacima koja su pogrešno klasifikovana). Pri kreiranju novog modela, veća je šansa da će izabrati podatke sa većom težinom, odnosno one što su pogrešno klasifikovani. Kada se vrši predikcija, ne vredi glas svakog modela podjednako, tu se takođe dodeljuju težine pri čemu veću težinu dobija onaj model koji je tačniji.



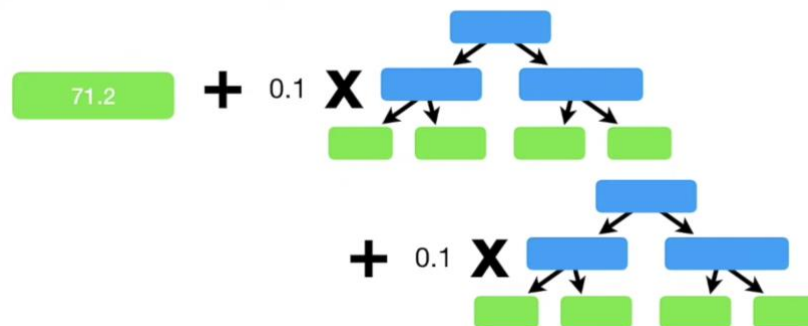
Slika 3 – Prikaz AdaBoost modela, preuzeto iz [3]

Sa **Slike 3** vidi se da je AdaBoost model sačinjen od više jednostavnih modela (stabla odluke prvog nivoa). Veće stablo odluke na slici predstavlja stablo koje ima veću težinu pri donošenju konačne odluke za AdaBoost model.

2.4 XGBoost

XGBoost (*Extreme Gradient Boosting*) model spada u *ensemble learning* i bazira se na *gradient boosting* metodi. Razlika GradientBoost modela i XGBoost modela je ta što XGBoost koristi regularizaciju³ pa je iz tog razloga brži i opštiji za nove podatke pa daje bolje rezultate pri klasifikaciji. XGBoost se često primenjuje kada su na raspolaganju veliki skupovi podataka.

GradientBoost model pri prvoj predikciji uzima srednju vrednost klase nakon čega kreće kreiranje stabala koja se množe stopom učenja. Proces prestaje ili kad je greška jako mala ili kad se dostigne maksimalni broj kreiranih stabala.



Slika 4 – Prikaz GradientBoost modela, preuzeto iz [3]

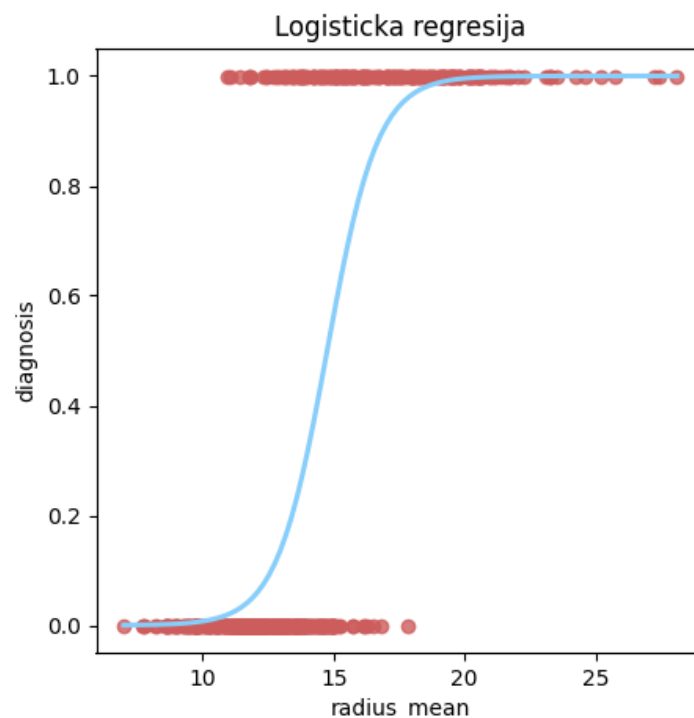
Za razliku od AdaBoost modela gde je nivo stabla odluke uvek 1, kod XGBoost modela obično se uzima da stablo ima od 2 do 5 nivoa. Stabla se takođe skaliraju kao kod AdaBoost modela, pri čemu sva stabla imaju podjednaku težinu pri glasanju. Pri primeni XGBoost modela, koriste se sledeći parametri:

- stopa učenja η (*learning rate*) – određuje koliko stablo utiče na predikciju, samim tim usporava napredak kako bi bio što tačniji
- parametar orezivanja γ (*pruning*) – određuje koliko grananja i nivoa postoji u odnosu na doprinos grananja, što je veći parametar γ to je model skloniji orezivanju
- parametar regularizacije λ – ako je jednak 0 regularizacija ne postoji, ako je veći od 0 onda doprinosi većem potkresivanju odnosno manjem *overfitting*-u

³ Regularizacija – tehnika koja na vreme sprečava pojavu *overfitting*-a, smanjuje varijansu

2.5 Logistička regresija

Logistička regresija je statistički algoritam koji pripada učenju sa nadgledanjem i koja ima čestu primenu u klasifikaciji. Naziv regresija potiče od toga što funkcija uzima izlaz linearne regresije kao ulaz i potom koristi sigmoidnu funkciju za procenu verovatnoće za datu klasu. Za razliku od linearne regresije čiji izlaz je kontinualna promenljiva sa ciljem da što bolje isprati trend ponašanja podataka, logistička regresija predviđa verovatnoću kojoj klasi uzorak pripada.



Slika 5 – Primer logističke regresije za atribut *radius_mean*

Svaki odbirak predstavljen je crvenim kružićem i kada se odredi sigmoidna funkcija (plava funkcija na slici), odbirci se vertikalno preslikavaju na tu funkciju. Y osa je uvek na skali od 0 do 1 jer je sledeći korak upravo preslikavanje kružića sa sigmoidne funkcije na Y osu, a vrednost na kojoj stanu predstavlja verovatnoća da taj odbirak pripada datoj klasi. Oblik i pozicija sigmoidne funkcije podešava se metodom maksimalne verodostojnosti⁴ (ne metodom najmanje srednje-kvadratne greške).

Da bi logistička regresija mogla da se primeni, potrebno je da: odbirci budu nezavisni, da klasa ima 2 moguće vrednosti i da skup podataka nije mali.

⁴ Metoda maksimalne verodostojnosti – metod koji procenjuje parametre raspodele verovatnoće maksimizacijom funkcije verovatnoće, tako da su dobijeni podaci najverovatniji

2.6 Naivni Bajes

Naivni Bajesov klasifikator je statistički algoritam koji pripada učenju sa nadgledanjem i koji ima čestu primenu u klasifikaciji. Bazira se na Bajesovoj teoremi, a reč *naivni* je dobio po tome što podrazumeva nezavisnost atributa (u realnim situacijama to je redak slučaj).

Bajesova teorema:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \quad (2.1)$$

gde je $X = (x_1, \dots, x_n)$ vektor atributa, a y promenljiva klase.

Naivni Bajes kaže da su atributi nezavisni pa važi:

$$P(X|y) = P(x_1, \dots, x_n|y) = \prod_{i=1}^n P(x_i|y) \quad (2.2)$$

Odbirak će biti dodeljen onoj klasi za koju je izraz $P(y) \prod_{i=1}^n P(x_i|y)$ najveći, odnosno drugim rečima, odbirak će upasti u onu klasu gde je najverovatnije, kako bi se minimizovala greška klasifikacije.

3 BAZA PODATAKA

3.1 Informacije o bazi

Baza podataka koja je korišćena u ovom projektu je **Wisconsin Breast Cancer Database** koja se može naći na sajtu **Kaggle**. U bazi se nalazi 569 realnih uzoraka, pri čemu za svaki od njih postoji 32 atributa koja ga detaljnije opisuju.

3.2 Atributi baze

Prvi atribut je **ID** koji služi da jednoznačno opiše svaki uzorak. Drugi atribut je **dijagnoza** koja predstavlja ciljnu promenljivu koja govori o kom tumoru je reč (**benigni** ili **maligni**). Ostali atributi su podeljeni u 3 grupe u zavisnosti od toga da li se odnose na srednju vrednost, srednje-kvadratnu grešku ili na najveće odstupanje. U svakoj grupi nalazi se sledećih 10 atributa:

1. **Radijus** – prosek udaljenosti od centra do graničnih tačaka
2. **Tekstura** – intenzitet nijansi sive u svakom pikselu, odnosno standardna devijacija vrednosti sive skale
3. **Perimetar** – obim
4. **Oblast**
5. **Glatkost** – lokalne varijacije u dužinama radijusa (razlika između dužine radijusa i srednje dužine dve linije poluprečnika koje ga okružuju)
6. **Kompaktnost** – $\frac{\text{perimetar}^2}{\text{oblast}-1}$
7. **Konkavnost** – predstavlja izraženost konkavnih delova konture, mala konkavnost odnosi se na glatkoću dok se velika odnosi na gruboću (postoje udubljenja)
8. **Konkavne tačke** – odnosi se na broj konkavnih delova
9. **Simetrija** – nakon što se odredi najduža linija koja prolazi kroz centar i spaja dve granične tačke, povlače se upravne linije na nju i mere se relativne razlike obe strane
10. **Fraktalna dimenzija**

3.3 Tehnike za izbor atributa

Postoje razne tehnike kojima se biraju atributi koje će biti korišćeni tokom grananja stabla. U ovom radu biće pomenute dve najčešće:

1. entropija i informaciona dobit
2. gini indeks

- 1. Entropija** predstavlja meru razućenosti klasa. Vrednost entropije može biti između 0 (mala razućenost klasa) i 1 (velika razućenost klasa)

$$E = - \sum_{i=1}^n p_i \log_2(p_i) \quad (3.1)$$

gde je p_i verovatnoća pojave klase i , a n broj klasa.

Informaciona dobit meri redukciju entropije nastalu deljenjem skupa pomoću izabranog atributa. Što je redukovanija entropija, to je veća informaciona dobit.

$$IG = E_{roditelja} - E_{deteta} \quad (3.2)$$

- 2. Gini indeks** meri verovatnoću da se proizvoljna instanca pogrešno klasifikuje. Vrednost gini indeksa može biti između 0 (mala razućenost klasa) i 0.5 (velika razućenost klasa).

$$Gini = 1 - \sum_{i=1}^n p_i^2 \quad (3.3)$$

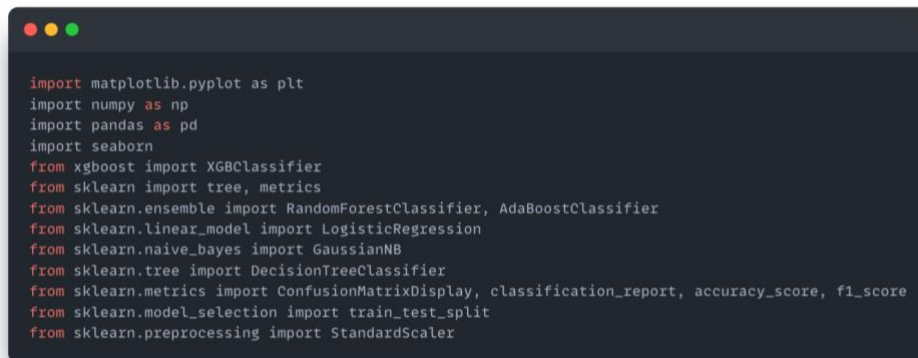
gde je p_i verovatnoća pojave klase i , a n broj klasa.

Poželjni su atributi koji imaju što veću informacionu dobit, a što manji gini indeks kako bi na bolji način podelili trenutni skup u dva homogenija skupa.

4 PROGRAMSKI KOD

4.1 Biblioteke i main funkcija

Kako bi mogle da se koriste već gotove funkcije pogodne za klasifikaciju i obradu podataka, potrebno je učitati odgovarajuće biblioteke u kojima se te funkcije i objekti nalaze:

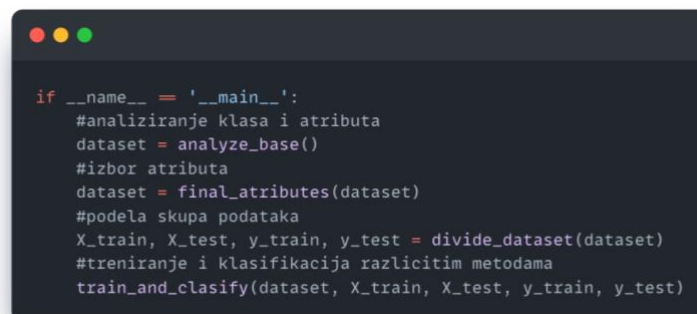


```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn
from xgboost import XGBClassifier
from sklearn import tree, metrics
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import ConfusionMatrixDisplay, classification_report, accuracy_score, f1_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Slika 6 – Biblioteke koje su korišćene u programu

Ceo kod pisan je u *Python* programskom jeziku u okruženju *Visual Studio Code*. Pri pokretanju programa, prvo se poziva funkcija *main()*. U *main()* funkciji se nalaze 4 funkcije koje predstavljaju 4 celine koje moraju biti urađene kako bi se uspešno implementiralo mašinsko učenje na priloženi skup podataka. Funkcije su sledeće:

1. *analyze_base()* – potrebno je analizirati bazu i atribute
2. *final_atributes()* – nakon analize potrebno je odabrati koji atributi su od značaja pri primeni mašinskog učenja
3. *divide_dataset()* – priprema i deli podatke za treniranje i testiranje
4. *train_and_clasify()* – primena mašinskog učenja, prvo ide treniranje nad trening skupom i nakon toga se testira algoritam nad test skupom



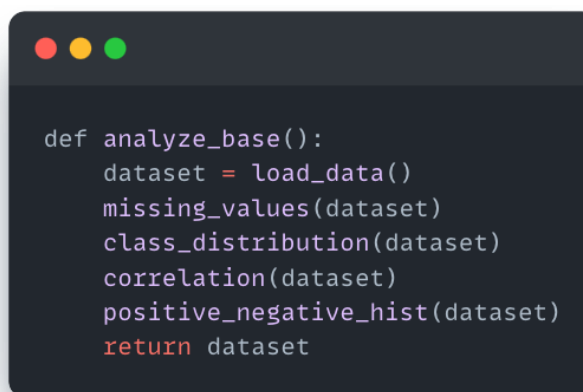
```
if __name__ == '__main__':
    #analiziranje klasa i atributa
    dataset = analyze_base()
    #izbor atributa
    dataset = final_atributes(dataset)
    #podela skupa podataka
    X_train, X_test, y_train, y_test = divide_dataset(dataset)
    #treniranje i klasifikacija razlicitim metodama
    train_and_clasify(dataset, X_train, X_test, y_train, y_test)
```

Slika 7 – Main funkcija

4.2 Analiza baze

Analiza baze sprovodi se kroz sledeće funkcije:

1. *load_data()* – učitava se baza i prilagođava za dalju obradu
2. *missing_values()* – proverava da li postoje nedostajuće vrednosti u bazi
3. *class_distribution()* – prikazuje raspodelu klasa kako bi se stekao uvid u balansiranost klasa
4. *correlation()* – određuju se korelacije
5. *positive_negative_hist()*



```
def analyze_base():
    dataset = load_data()
    missing_values(dataset)
    class_distribution(dataset)
    correlation(dataset)
    positive_negative_hist(dataset)
    return dataset
```

Slika 8 – Analiziranje baze i atributa

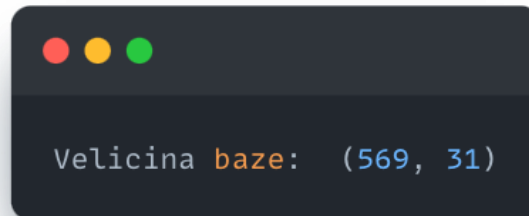
Prvi korak je učitavanje baze. Učitana baza sadrži dve kolone koje nemaju značaja za analizu (kolona *id* i kolona *Unnamed*) i te kolone je poželjno skloniti iz baze. Vrednosti klase B i M potrebno je mapirati u celobrojne vrednosti 0 i 1 radi lakše dalje analize:



```
#ucitavanje baze
def load_data():
    dataset = pd.read_csv('data.csv')
    #izbacivanje kolona koje nisu od znacaja (kolona 'id' i kolona 'Unnamed: 32')
    dataset = dataset.drop(columns=['id', 'Unnamed: 32'])
    print('\nVelicina baze: ', dataset.shape)
    dataset['diagnosis'] = dataset['diagnosis'].replace('B', 0).replace('M', 1)
    print('\n\nPrikaz poslednjih 10 redova baze:\n\n', dataset.tail(10))
    return dataset
```

Slika 9 – Učitavanje baze

Baza sadrži 30 kolona atributa kao i 1 kolonu koja definiše klasu odbirka. Na raspolaganju su 569 odbiraka koji predstavljaju redove baze.



Slika 10 – Dimenzije baze

Prikaz poslednjih 10 redova baze:

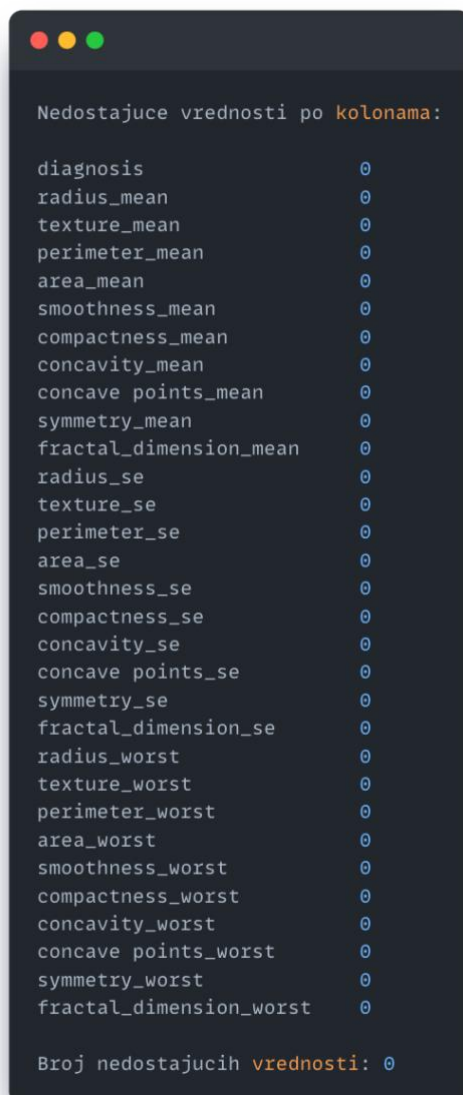
	diagnosis	radius_mean	texture_mean	perimeter_mean	...	concavity_worst	concave	points_worst	symmetry_worst	fractal_dimension_worst
559	0	11.51	23.93	74.52	...	0.3630	0.09653	0.2112		0.08732
560	0	14.05	27.15	91.38	...	0.1326	0.10480	0.2250		0.08321
561	0	11.20	29.37	70.67	...	0.0000	0.00000	0.1566		0.05905
562	1	15.22	30.62	103.40	...	1.1700	0.23560	0.4089		0.14090
563	1	20.92	25.09	143.00	...	0.6599	0.25420	0.2929		0.09873
564	1	21.56	22.39	142.00	...	0.4107	0.22160	0.2060		0.07115
565	1	20.13	28.25	131.20	...	0.3215	0.16280	0.2572		0.06637
566	1	16.60	28.08	108.30	...	0.3403	0.14180	0.2218		0.07820
567	1	20.60	29.33	140.10	...	0.9387	0.26500	0.4087		0.12400
568	0	7.76	24.54	47.92	...	0.0000	0.00000	0.2871		0.07039

Slika 11 – Prikaz poslednjih 10 redova baze

Drugi korak je provera da li postoje nedostajuće vrednosti:

```
#provera da li ima nedostajucih vrednosti
def missing_values(dataset):
    missing_values_per_column = dataset.isna().sum()
    print("\nNedostajuce vrednosti po kolonama:\n\n", missing_values_per_column)
    missing_values = missing_values_per_column.sum()
    print("\nBroj nedostajucih vrednosti:", missing_values)
```

Slika 12 – Provera da li postoje nedostajuće vrednosti



```
Nedostajuće vrednosti po kolonama:

diagnosis                0
radius_mean              0
texture_mean             0
perimeter_mean           0
area_mean                0
smoothness_mean          0
compactness_mean         0
concavity_mean           0
concave points_mean      0
symmetry_mean            0
fractal_dimension_mean   0
radius_se                0
texture_se               0
perimeter_se             0
area_se                  0
smoothness_se            0
compactness_se           0
concavity_se             0
concave points_se        0
symmetry_se              0
fractal_dimension_se     0
radius_worst             0
texture_worst            0
perimeter_worst          0
area_worst               0
smoothness_worst         0
compactness_worst        0
concavity_worst          0
concave points_worst     0
symmetry_worst           0
fractal_dimension_worst  0

Broj nedostajucih vrednosti: 0
```

Slika 13 – Prikaz broja nedostajućih vrednosti

Zaključuje se da nema vrednosti koje nedostaju u bazi.

Treći korak prikazuje raspodelu benignih i malignih tumora u klasi dijagnoza.

```
def class_distribution(dataset):  
    numBenign = sum(dataset['diagnosis'] == 0)  
    numMalignant = sum(dataset['diagnosis'] == 1)  
    print('Velicina baze: ', dataset.shape)  
    print('Broj uzoraka sa benignim oboljenjem: ', numBenign)  
    print('Broj uzoraka sa malignim oboljenjem: ', numMalignant)
```

Slika 14 – Distribucija klasa

```
Broj nedostajucih vrednosti: 0  
Broj uzoraka sa benignim oboljenjem: 357  
Broj uzoraka sa malignim oboljenjem: 212
```

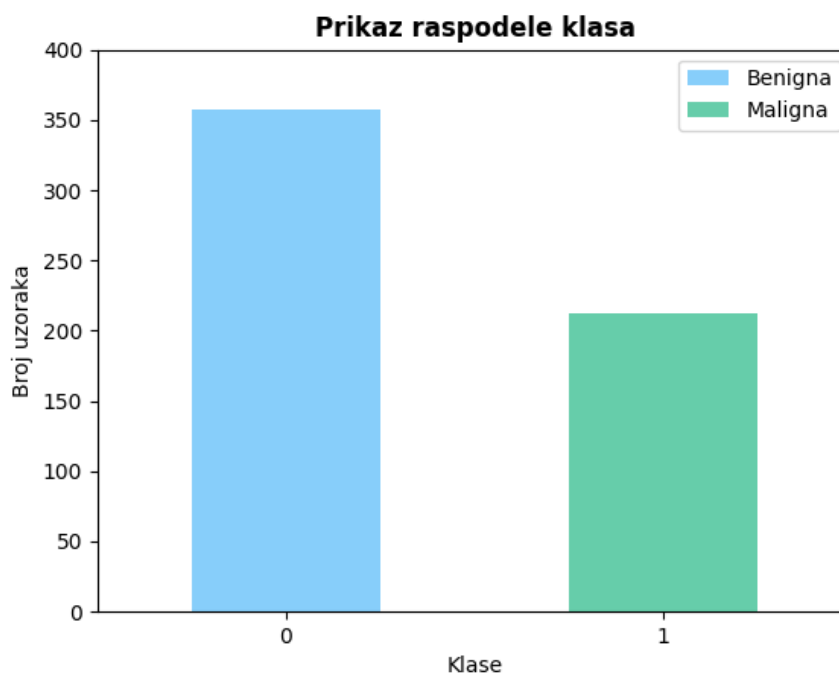
Slika 15 – Prikaz distribucije klasa

Za bolje razumevanje, dat je i vizualni prikaz:

```
#prikaz raspodele klasa  
plt.bar(0, numBenign, color='lightskyblue', width=0.5, label='Benigna')  
plt.bar(1, numMalignant, color='mediumaquamarine', width=0.5, label='Maligna')  
plt.axis([-0.5, 1.5, 0, 400])  
plt.xticks([0, 1])  
plt.xlabel('Klase')  
plt.ylabel('Broj uzoraka')  
plt.title('Prikaz raspodele klasa', weight='bold')  
plt.legend()  
plt.savefig('./images/class_distribution.png')  
plt.show()
```

Slika 16 – Kod za kreiranje vizualnog prikaza distribucije klasa

Raspodela ciljne promenljive prikazana je na sledećoj slici:



Slika 17 – Vizualni prikaz distribucije klasa

Može se zaključiti da je na raspolaganju mnogo veći broj benignih oboljenja nego malignih. Iz tog razloga potrebno je voditi računa o podeli podataka na trening i test skup jer u slučaju nebalansiranih skupova može doći do toga da se određena klasa slabo pojavi u trenirajućem skupu a dosta u testirajućem skupu i obrnuto. Kao posledica pojaviće se velika greška klasifikacije.

Četvrti korak odnosi se na korelisanost atributa:

```
def correlation(dataset):  
    #kreiranje korelacija  
    mean = dataset[dataset.columns[:11]]  
    se = dataset.drop(dataset.columns[1:11], axis=1)  
    se = se.drop(se.columns[11:], axis=1)  
    worst = dataset.drop(dataset.columns[1:21], axis=1)  
  
    titles = ["srednje vrednosti", "srednje-kvadratne greske", "najvece greske"]  
    #prikaz korelacija za dijagnozom  
    correlation_with_diagnosis(mean, se, worst, titles)  
    #prikaz korelacija sa dijagnozom kolor mapom  
    mutual_correlation(dataset, mean, se, worst, titles)
```

Slika 18 – Određivanje korelisanosti atributa

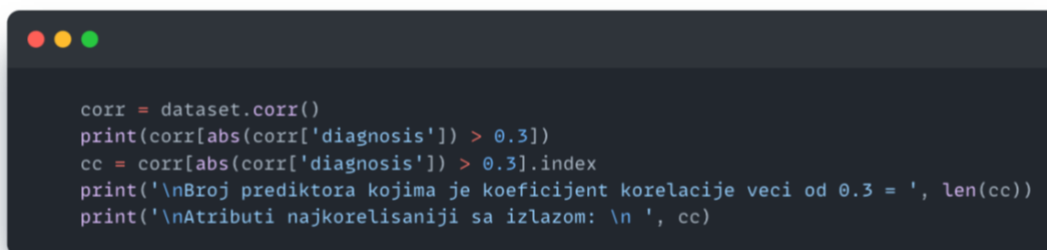
Postoje dva tipa korelisanosti: korelisanost atributa sa dijagnozom i međusobna korelisanost atributa. Poželjni su atributi koji su visoko korelisani sa dijagnozom. Ipak, može da se desi da su dva različita atributa visoko korelisana sa dijagnozom, a da su takođe i međusobno visoko korelisani. U takvom slučaju samo 1 od ta dva atributa je dovoljan jer drugi neće uneti nikakvu dodatnu informaciju pri klasifikaciji, samo će povećati složenost.

U prvom slučaju potrebno je pronaći korelisanosti atributa i klase dijagnoza. To je urađeno podelom atributa u 3 grupe u zavisnosti da li se odnose na srednju vrednost, srednje-kvadratnu grešku ili na najveće odstupanje:



```
def correlation_with_diagnosis(mean, se, worst, titles):
    data = [mean, se, worst]
    colors = ["lightskyblue", "mediumaquamarine", "peachpuff"]
    file_names = ["mean_correlation_map.png", "se_correlation_map.png", "worst_correlation_map.png"]
    for i in range (len(data)):
        correlation = data[i].drop(columns=['diagnosis']).corrwith(data[i].diagnosis)
        correlation.plot(kind='bar', grid=False, color=colors[i])
        plt.figure(figsize = (20, 8))
        plt.title("Korelacija atributa " + titles[i] + " sa dijagnozom", weight='bold')
        plt.ylabel('Nivo korelisanosti')
        plt.savefig("./images/" + file_names[i])
        plt.show()
```

Slika 19 – Određivanje korelisanosti atributa sa dijagnozom



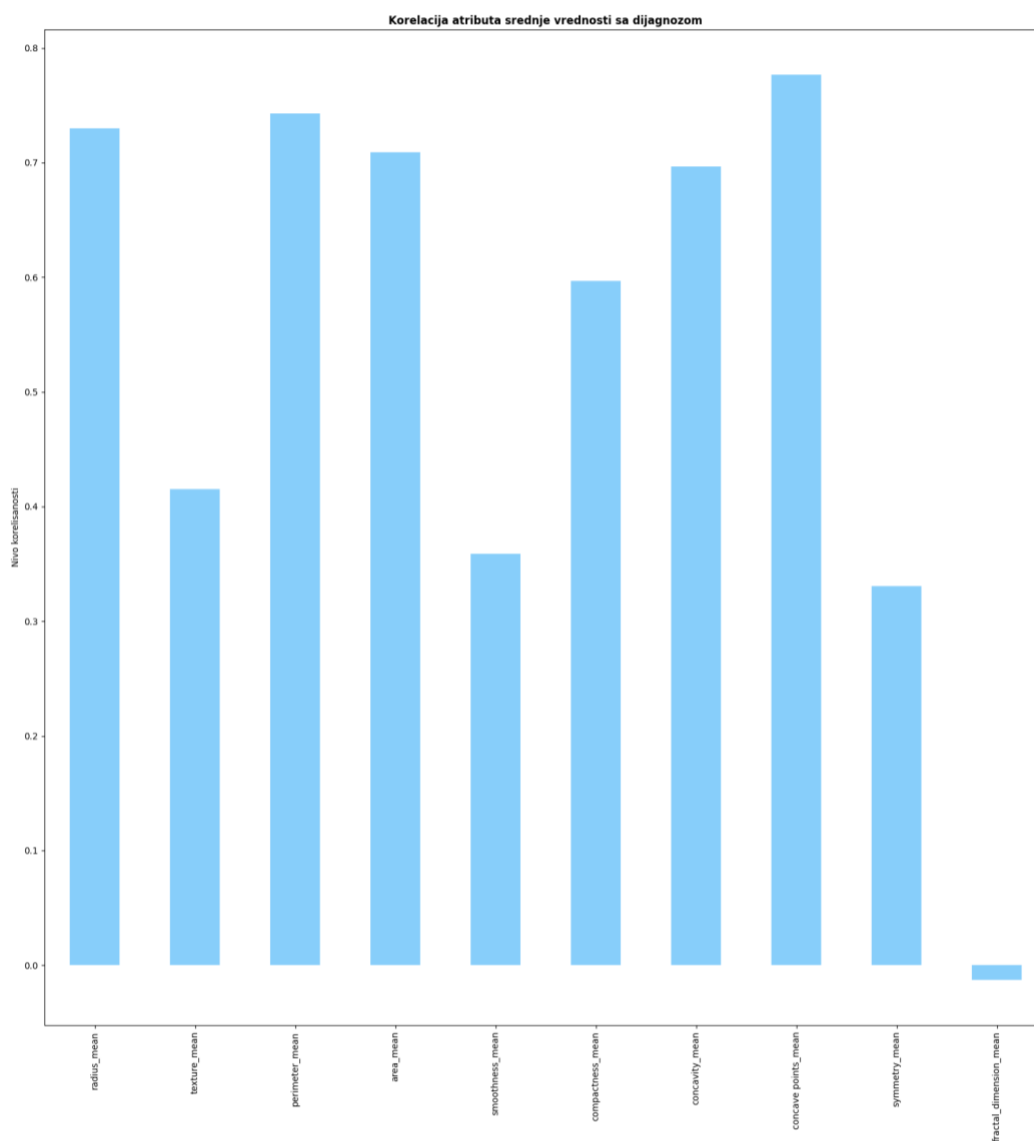
```
corr = dataset.corr()
print(corr[abs(corr['diagnosis']) > 0.3])
cc = corr[abs(corr['diagnosis']) > 0.3].index
print('\nBroj prediktora kojima je koeficijent korelacije veci od 0.3 = ', len(cc))
print('\nAtributi najkorelisaniji sa izlazom: \n ', cc)
```

Slika 20 – Prikaz atributa koji su najkorelisaniji sa dijagnozom

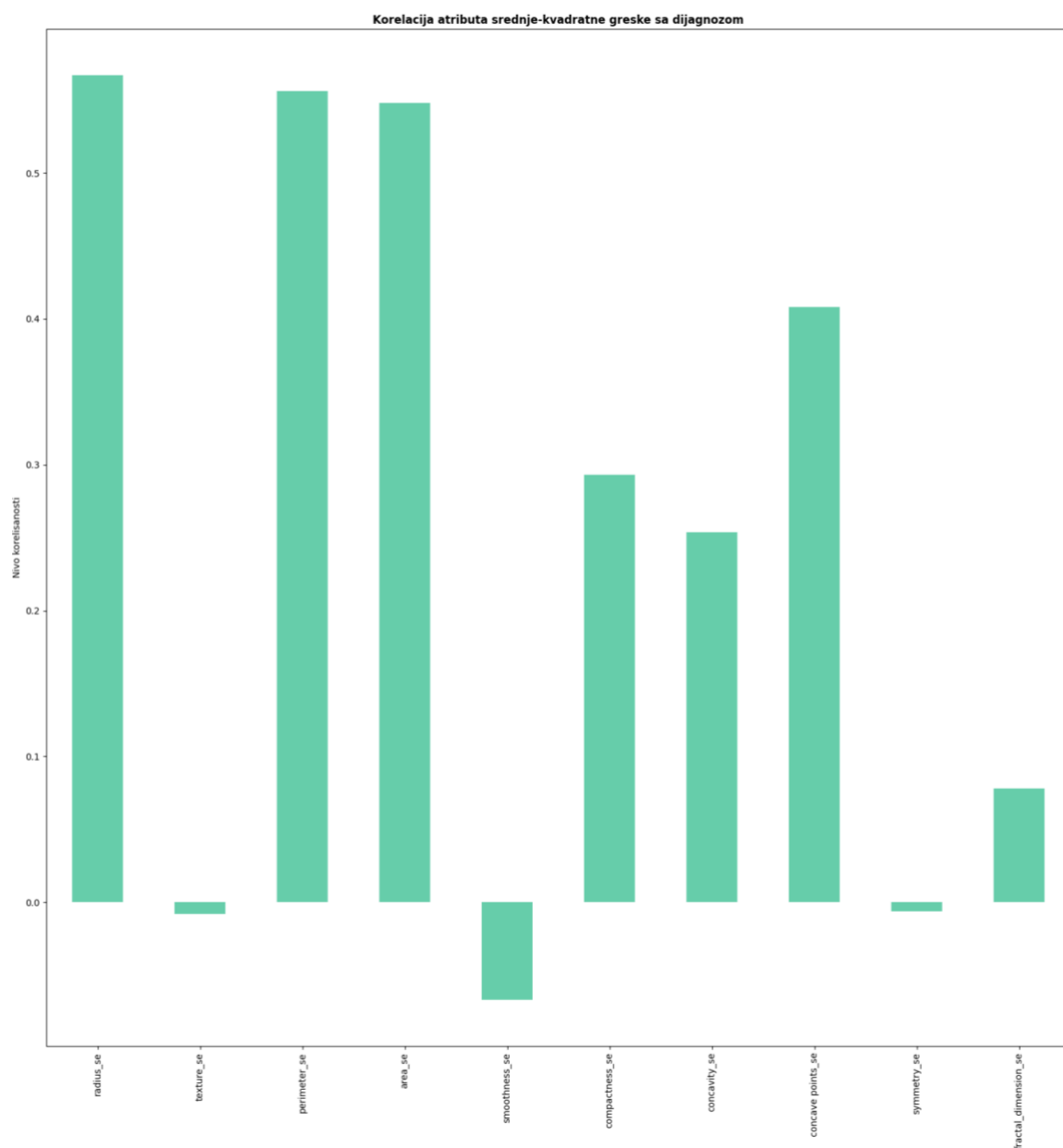
```
Broj prediktora kojima je koeficijent korelacije veci od 0.3 = 24

Atributi najkorelisaniji sa izlazom:
Index(['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'radius_se', 'perimeter_se',
       'area_se', 'concave points_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

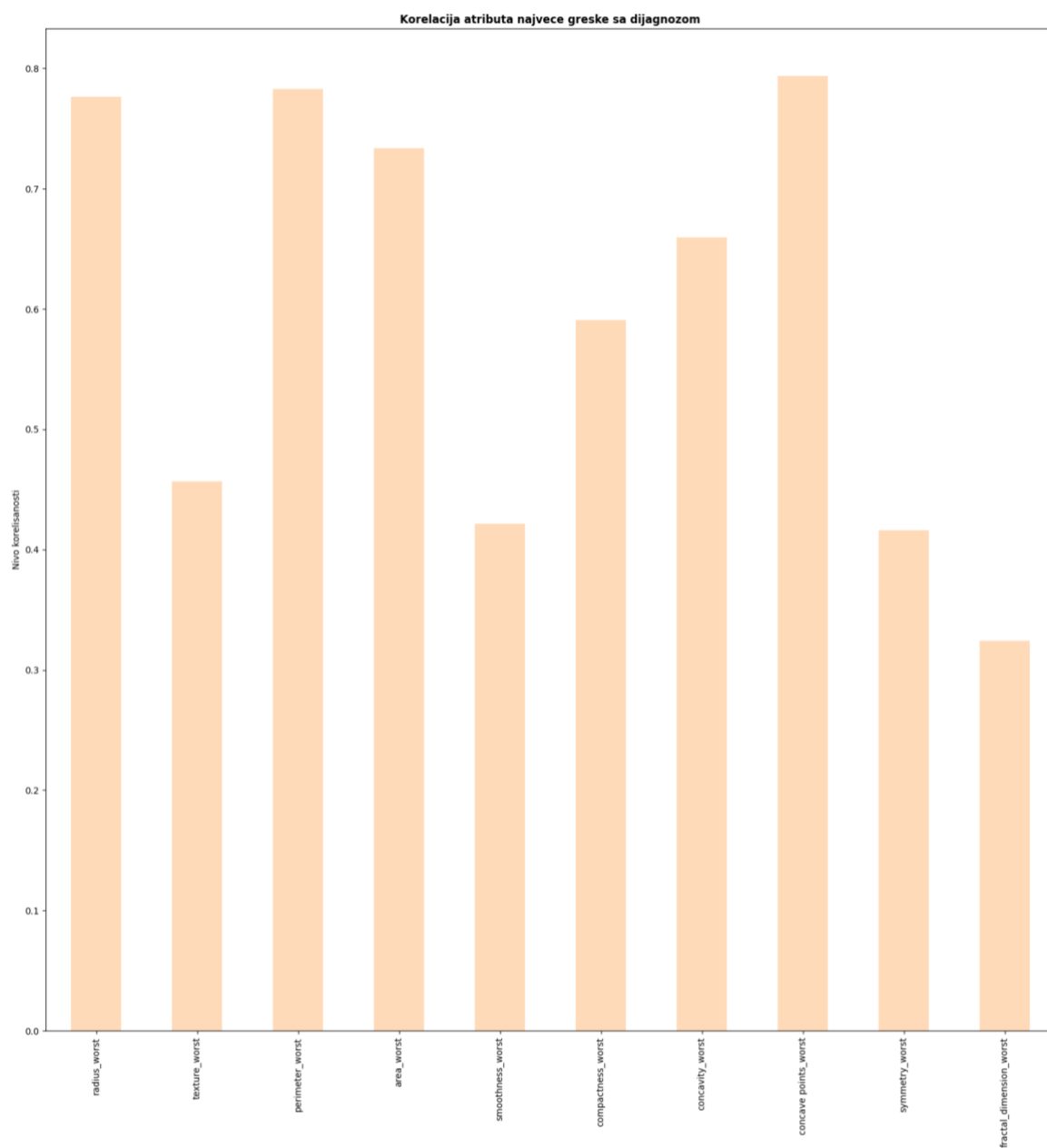
Slika 21 – Atributi najkorelisaniji sa dijagnozom



Slika 22 – Korelacija atributa srednje vrednosti sa dijagnozom

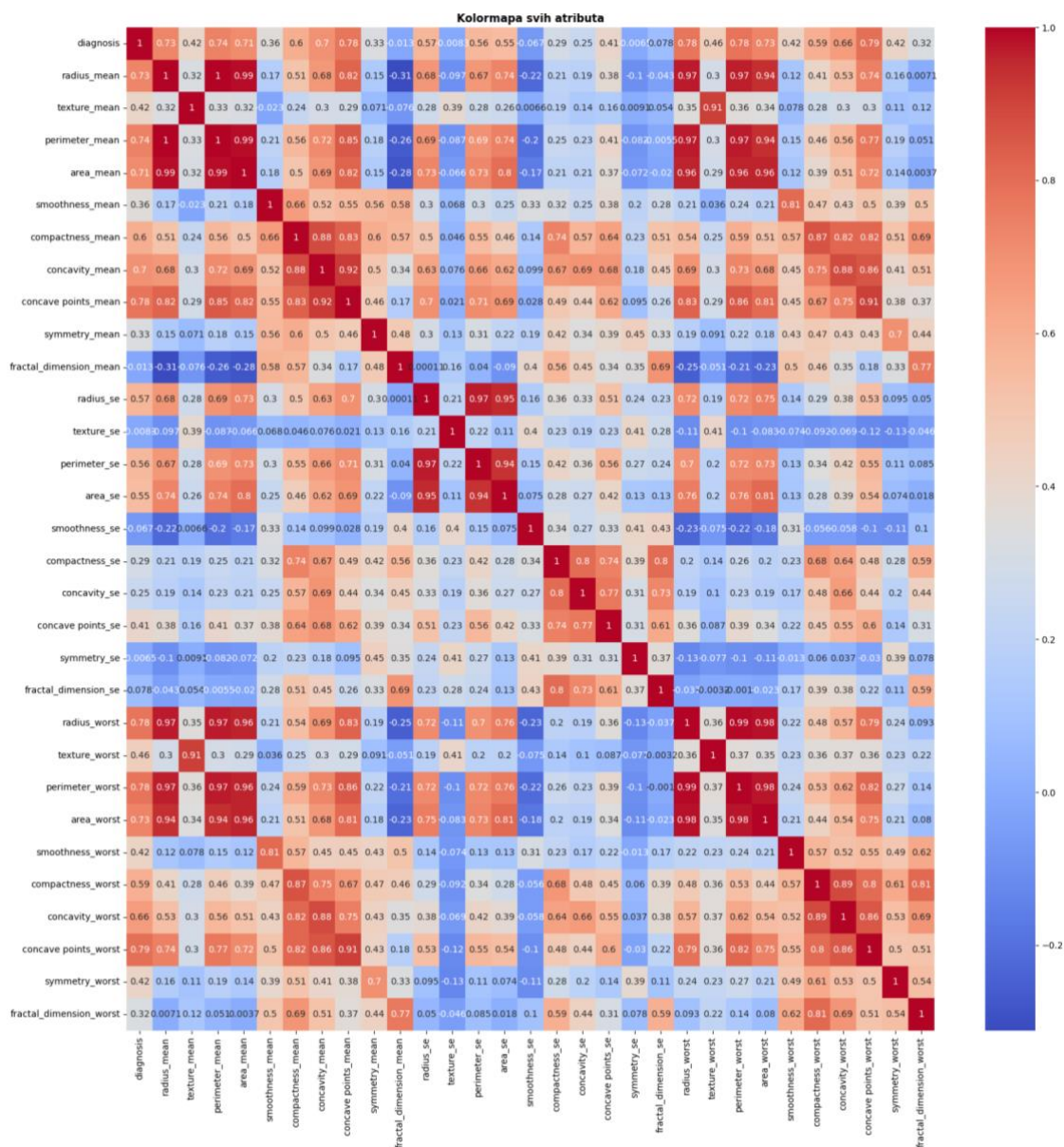


Slika 23 – Korelacija atributa srednje-kvadratne greške sa dijagnozom

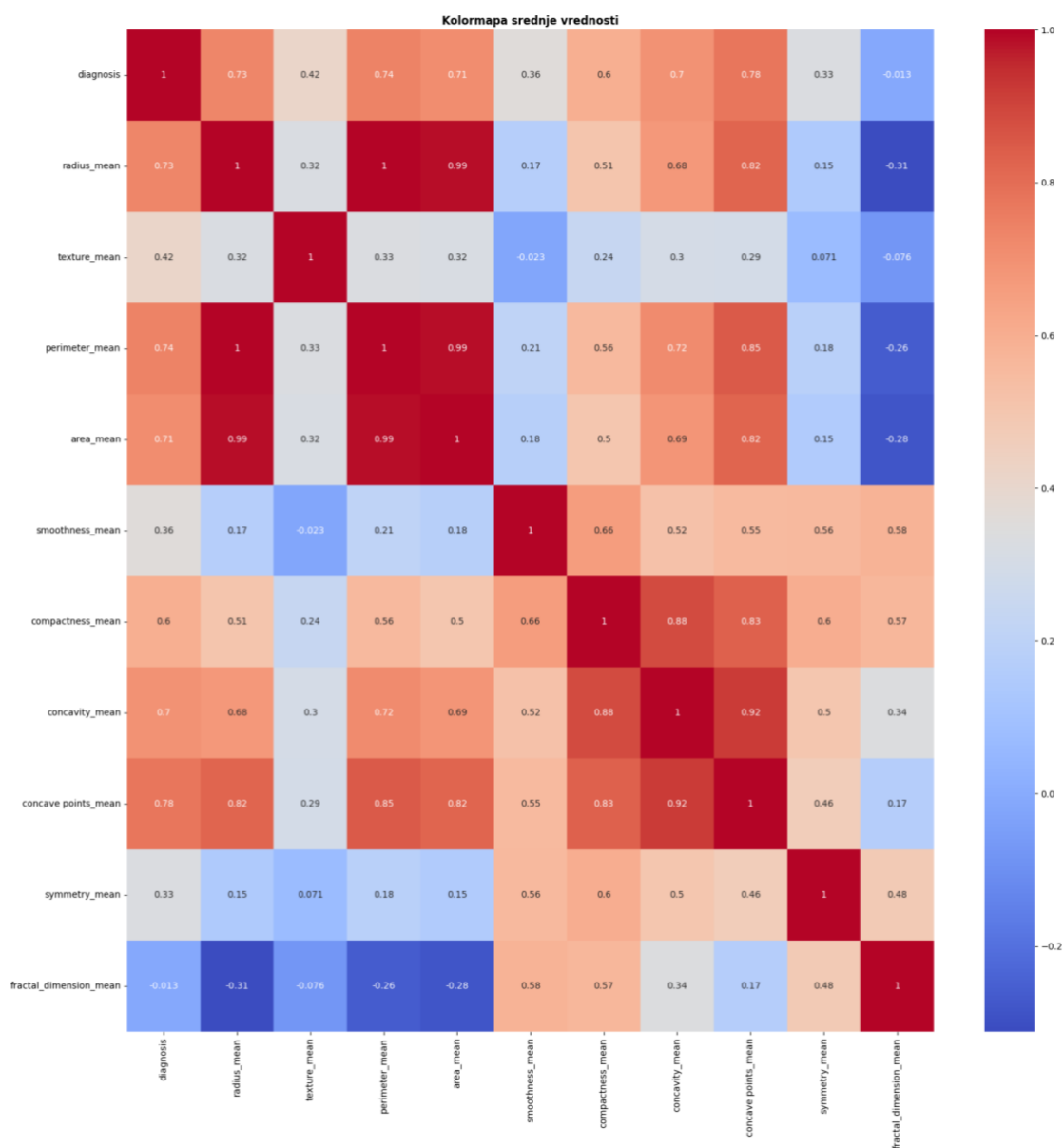


Slika 24 – Korelacija atributa najvećeg odstupanja sa dijagnozom

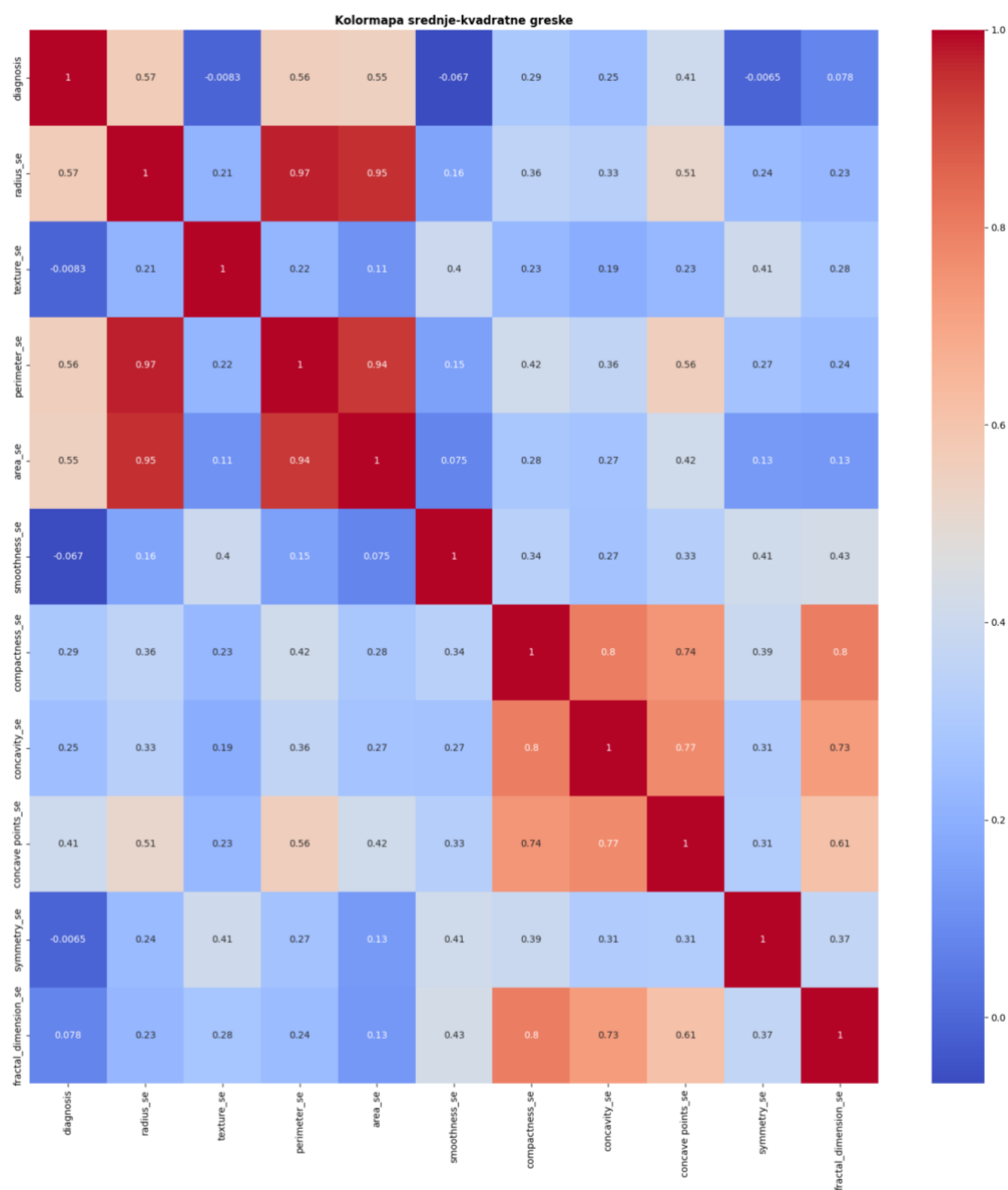
U drugom slučaju potrebno je pronaći međusobnu korelisanost atributa.



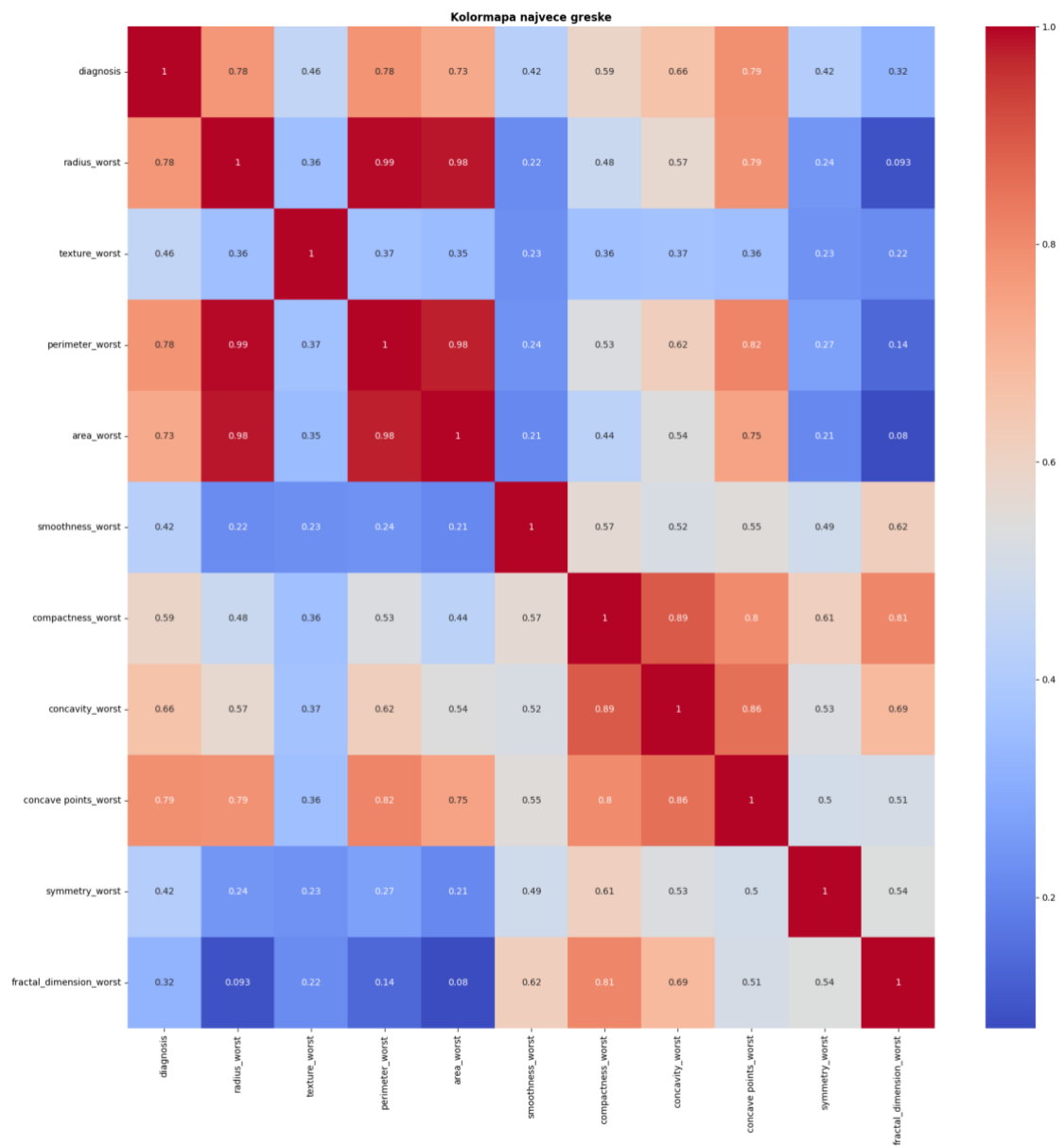
Slika 25 – Međusobna korelisanost svih atributa



Slika 26 – Međusobna korelisanost atributa srednje vrednosti



Slika 27 – Međusobna korelisanost atributa srednje-kvadratne greške



Slika 28 – Međusobna korelisanost najvećeg odstupanja

4.3 Izbor atributa od značaja

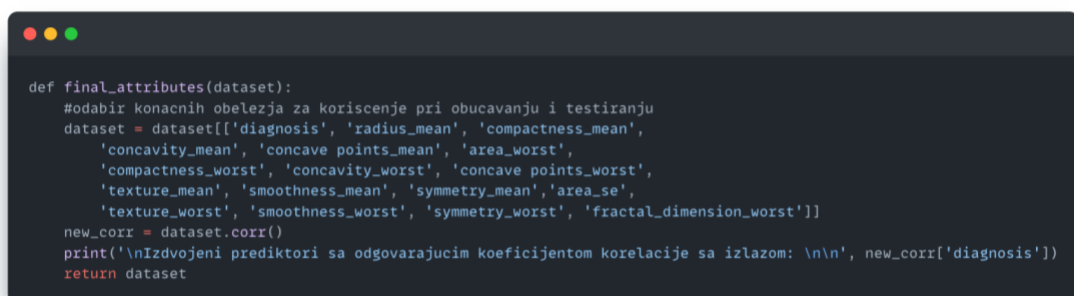
Kako u bazi postoji 30 atributa, potrebno je redukovati taj broj tako da ostanu najinformativniji atributi koji međusobno nemaju veliku korelisanost.

- Analiza korelisanosti atributa i klase

Sa **Slike 22** primetno je da atribut *fractal_dimension_mean* ima malu korelisanost sa klasom te nije od značaja. Takođe, sa **Slike 23** primetno je da atributi: *texture_se*, *smoothness_se*, *symmetry_se*, *fractal_dimension_se*, nisu od značaja jer i oni imaju malu korelisanost sa klasom. Sa **Slike 24** ne može se izvući atribut koji je slabo korelisan sa klasom.

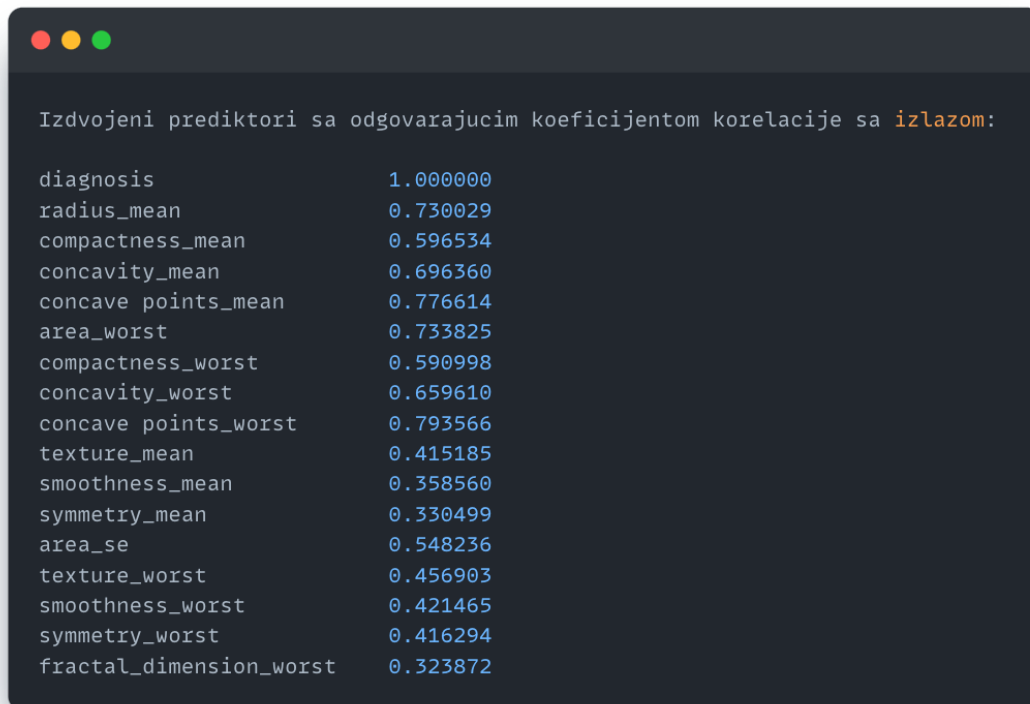
- Analiza međusobne korelisanosti atributa

Sa **Slike 26** primećuje se velika korelisanost (skoro 1) kod atributa: *radius_mean*, *perimeter_mean* i *area_mean*. Dovoljno je uzeti 1 od njih, na primer *radius_mean*. Slično, sa **Slike 28** velika korelisanost se može primetiti kod atributa: *radius_worst*, *perimeter_worst* i *area_worst* te je najbolje uzeti samo 1 od njih, na primer *area_worst*.



```
def final_attributes(dataset):
    #odabir konacnih obelezja za koriscenje pri obucavanju i testiranju
    dataset = dataset[['diagnosis', 'radius_mean', 'compactness_mean',
                        'concavity_mean', 'concave points_mean', 'area_worst',
                        'compactness_worst', 'concavity_worst', 'concave points_worst',
                        'texture_mean', 'smoothness_mean', 'symmetry_mean', 'area_se',
                        'texture_worst', 'smoothness_worst', 'symmetry_worst', 'fractal_dimension_worst']]
    new_corr = dataset.corr()
    print('\nIzdvojeni prediktori sa odgovarajucim koeficijentom korelacije sa izlazom: \n\n', new_corr['diagnosis'])
    return dataset
```

Slika 29 – Izbor konačnih atributa za dalju analizu



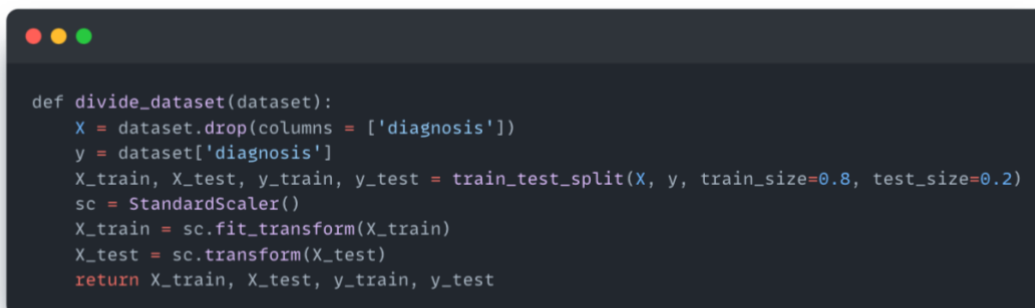
Izdvojeni prediktori sa odgovarajućim koeficijentom korelacije sa **izlazom**:

diagnosis	1.000000
radius_mean	0.730029
compactness_mean	0.596534
concavity_mean	0.696360
concave points_mean	0.776614
area_worst	0.733825
compactness_worst	0.590998
concavity_worst	0.659610
concave points_worst	0.793566
texture_mean	0.415185
smoothness_mean	0.358560
symmetry_mean	0.330499
area_se	0.548236
texture_worst	0.456903
smoothness_worst	0.421465
symmetry_worst	0.416294
fractal_dimension_worst	0.323872

Slika 30 – Korelisanost izabranih atributa sa izlazom

4.4 Podela podataka na trening i test skup

Potrebno je podeliti skup podataka na podatke koji se koriste isključivo za treniranje i na one koji se koriste za testiranje. Pri biranju skupa potrebno je voditi računa o balansiranosti, odnosno da su obe klase zastupljene i u trening skupu i u test skupu. Izabrano je 80% podataka da bude u trening skupu, dok 20% služi za testiranje.

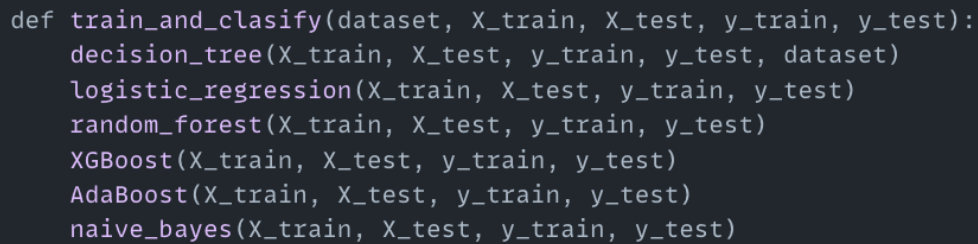


```
def divide_dataset(dataset):  
    X = dataset.drop(columns = ['diagnosis'])  
    y = dataset['diagnosis']  
    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2)  
    sc = StandardScaler()  
    X_train = sc.fit_transform(X_train)  
    X_test = sc.transform(X_test)  
    return X_train, X_test, y_train, y_test
```

Slika 31 – Podela podataka na trening i test skup

4.5 Treniranje i testiranje

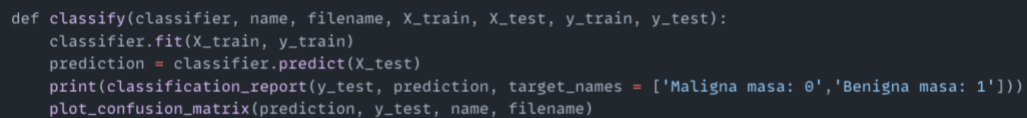
Glavna funkcija *train_and_clasify()* koja se poziva iz *main()* funkcije, služi da redom poziva svaki od kreiranih klasifikatora.



```
def train_and_clasify(dataset, X_train, X_test, y_train, y_test):  
    decision_tree(X_train, X_test, y_train, y_test, dataset)  
    logistic_regression(X_train, X_test, y_train, y_test)  
    random_forest(X_train, X_test, y_train, y_test)  
    XGBoost(X_train, X_test, y_train, y_test)  
    AdaBoost(X_train, X_test, y_train, y_test)  
    naive_bayes(X_train, X_test, y_train, y_test)
```

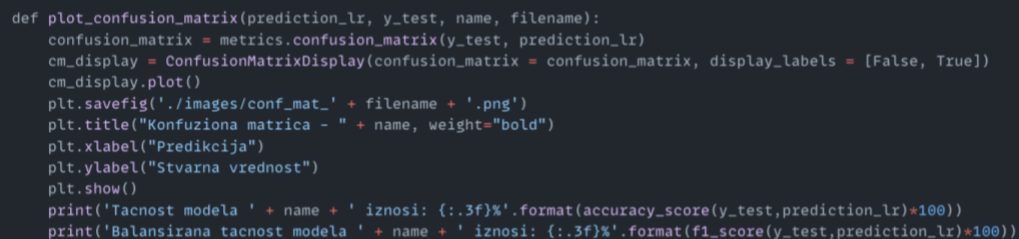
Slika 32 – Funkcija koja redom poziva klasifikatore

Svaki klasifikator nakon kreiranja poziva funkciju *classify()* u kojoj se klasifikator trenira a potom i vrši predikcija čiji se rezultat prikazuje funkcijom *plot_confusion_matrix()*:



```
def classify(classifier, name, filename, X_train, X_test, y_train, y_test):  
    classifier.fit(X_train, y_train)  
    prediction = classifier.predict(X_test)  
    print(classification_report(y_test, prediction, target_names = ['Maligna masa: 0', 'Benigna masa: 1']))  
    plot_confusion_matrix(prediction, y_test, name, filename)
```

Slika 33 – Funkcija za treniranje



```
def plot_confusion_matrix(prediction_lr, y_test, name, filename):  
    confusion_matrix = metrics.confusion_matrix(y_test, prediction_lr)  
    cm_display = ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])  
    cm_display.plot()  
    plt.savefig('./images/conf_mat_' + filename + '.png')  
    plt.title("Konfuziona matrica - " + name, weight="bold")  
    plt.xlabel("Predikcija")  
    plt.ylabel("Stvarna vrednost")  
    plt.show()  
    print('Tacnost modela ' + name + ' iznosi: {:.3f}%'.format(accuracy_score(y_test, prediction_lr)*100))  
    print('Balansirana tacnost modela ' + name + ' iznosi: {:.3f}%'.format(f1_score(y_test, prediction_lr)*100))
```

Slika 34 – Konfuziona matrica za prikaz rezultata

5 POREĐENJE KLASIFIKATORA

Nakon završetka celog procesa moguće je uporediti rezultate klasifikatora:

```
-----  
Stablo odlucivanja:  


|                 | precision | recall | f1-score | support |
|-----------------|-----------|--------|----------|---------|
| Maligna masa: 0 | 0.99      | 0.96   | 0.97     | 70      |
| Benigna masa: 1 | 0.93      | 0.98   | 0.96     | 44      |
| accuracy        |           |        | 0.96     | 114     |
| macro avg       | 0.96      | 0.97   | 0.96     | 114     |
| weighted avg    | 0.97      | 0.96   | 0.97     | 114     |



Tacnost modela stabla odlucivanja iznosi: 96.491%  
Balansirana tacnost modela stabla odlucivanja iznosi: 95.556%

  
-----  
Logisticka regresija:  


|                 | precision | recall | f1-score | support |
|-----------------|-----------|--------|----------|---------|
| Maligna masa: 0 | 0.96      | 1.00   | 0.98     | 70      |
| Benigna masa: 1 | 1.00      | 0.93   | 0.96     | 44      |
| accuracy        |           |        | 0.97     | 114     |
| macro avg       | 0.98      | 0.97   | 0.97     | 114     |
| weighted avg    | 0.97      | 0.97   | 0.97     | 114     |



Tacnost modela logisticke regresije iznosi: 97.368%  
Balansirana tacnost modela logisticke regresije iznosi: 96.471%

  
-----  
Slucajna suma:  


|                 | precision | recall | f1-score | support |
|-----------------|-----------|--------|----------|---------|
| Maligna masa: 0 | 0.96      | 1.00   | 0.98     | 70      |
| Benigna masa: 1 | 1.00      | 0.93   | 0.96     | 44      |
| accuracy        |           |        | 0.97     | 114     |
| macro avg       | 0.98      | 0.97   | 0.97     | 114     |
| weighted avg    | 0.97      | 0.97   | 0.97     | 114     |



Tacnost modela slucajne sume iznosi: 97.368%  
Balansirana tacnost modela slucajne sume iznosi: 96.471%

  
-----  
XGBoost:  


|                 | precision | recall | f1-score | support |
|-----------------|-----------|--------|----------|---------|
| Maligna masa: 0 | 0.99      | 1.00   | 0.99     | 70      |
| Benigna masa: 1 | 1.00      | 0.98   | 0.99     | 44      |
| accuracy        |           |        | 0.99     | 114     |
| macro avg       | 0.99      | 0.99   | 0.99     | 114     |
| weighted avg    | 0.99      | 0.99   | 0.99     | 114     |



Tacnost modela XGBoost iznosi: 99.123%  
Balansirana tacnost modela XGBoost iznosi: 98.851%

  
-----  
AdaBoost:  


|                 | precision | recall | f1-score | support |
|-----------------|-----------|--------|----------|---------|
| Maligna masa: 0 | 0.96      | 0.99   | 0.97     | 70      |
| Benigna masa: 1 | 0.98      | 0.93   | 0.95     | 44      |
| accuracy        |           |        | 0.96     | 114     |
| macro avg       | 0.97      | 0.96   | 0.96     | 114     |
| weighted avg    | 0.97      | 0.96   | 0.96     | 114     |



Tacnost modela AdaBoost iznosi: 96.491%  
Balansirana tacnost modela AdaBoost iznosi: 95.349%

  
-----  
Naivni Bajes:  


|                 | precision | recall | f1-score | support |
|-----------------|-----------|--------|----------|---------|
| Maligna masa: 0 | 0.93      | 0.97   | 0.95     | 70      |
| Benigna masa: 1 | 0.95      | 0.89   | 0.92     | 44      |
| accuracy        |           |        | 0.94     | 114     |
| macro avg       | 0.94      | 0.93   | 0.93     | 114     |
| weighted avg    | 0.94      | 0.94   | 0.94     | 114     |

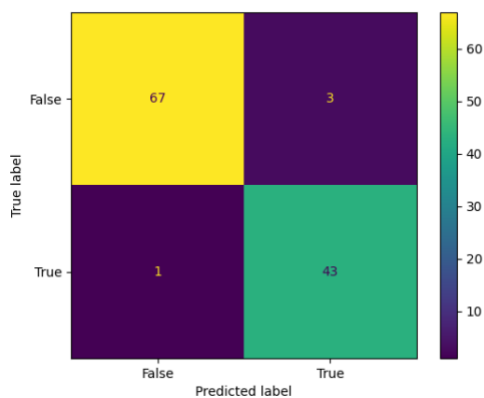


Tacnost modela Naivnog Bajesa iznosi: 93.868%  
Balansirana tacnost modela Naivnog Bajesa iznosi: 91.765%

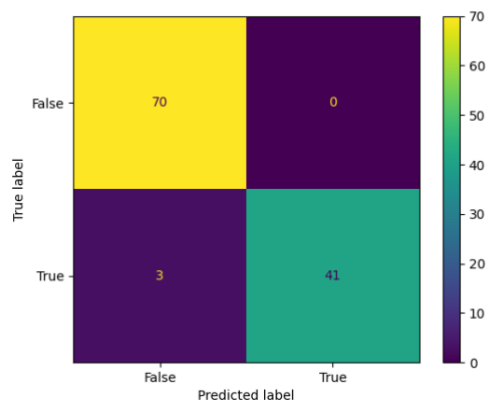
  
-----
```

Slika 35 – Poređenja različitih tehnika klasifikacije

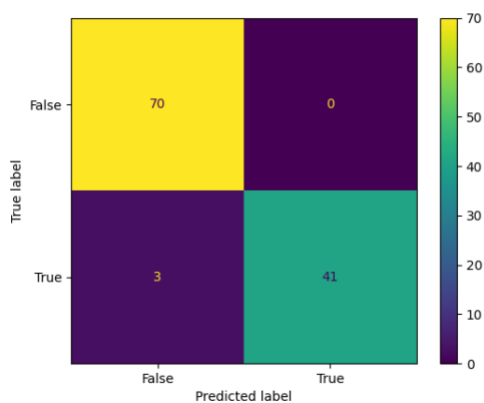
Prikaz konfuzionih matrica:



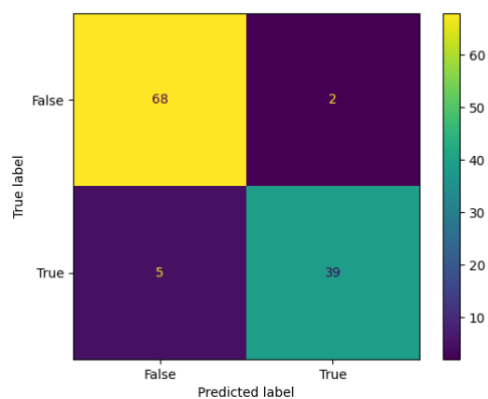
Slika 36 – KM stabla odluke



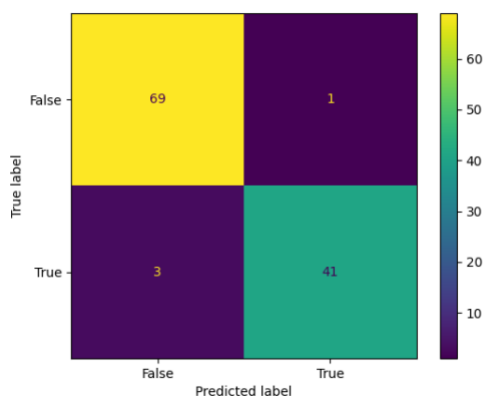
Slika 37 – KM logističke regresije



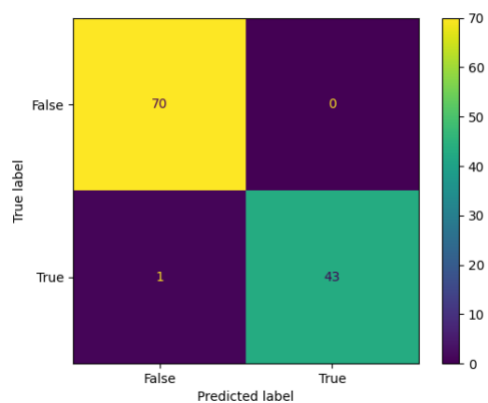
Slika 38 – KM slučajne šume



Slika 39 – KM Naivnog Bajesa



Slika 40 – KM AdaBoost



Slika 41 – KM XGBoost

Ponovnim pokretanjem programa dobiće se neki novi rezultati koji će se razlikovati od trenutnih. Ipak, može se primetiti da metoda Naivnog Bajesa uvek daje lošije rezultate od ostalih, kao i da XGBoost i slučajne šume daju odlične rezultate. Stablo odluke daje zadovoljavajuće rezultate, ali slučajne šume kao njegovo unapređenje svaki put daje bolji rezultat. Svaka tehnika ima svoje prednosti i svoje mane. Tačnost i balansiranost kod svih klasifikatora su preko 90%.

6 ZAKLJUČAK

Mašinsko učenje predstavlja moćan alat u pogledu klasifikacije kada na raspolaganju postoji skup podataka iz kojeg računar može da uči i izvlači korisne informacije kako bi napravio logiku kojom će uspeti da reši neke nove podatke. Kako tehnologija napreduje, tako i mašinsko učenje, a naročito njegova primena koja je u 21. veku sve učestalija. Primene mašinskog učenja u medicini mogle bi drastično da promene kvalitet života i da se pomoću njih na vreme detektuju neželjene bolesti.

7 LITERATURA

- [1] <https://www.verywellhealth.com/what-does-malignant-and-benign-mean-514240>
- [2] <https://www.tibco.com/reference-center/what-is-a-random-forest>
- [3] <https://statquest.org/>
- [4] <https://builtin.com/machine-learning/adaboost>
- [5] <https://blog.quantinsti.com/xgboost-python/>
- [6] <https://towardsdatascience.com/decision-trees-explained-entropy-information-gain-gini-index-ccp-pruning-4d78070db36c>