

PROJEKAT

Klasifikacija raka dojke

pomoću mašinskog učenja

Nemanja Janković

Beograd, septembar 2023. godine

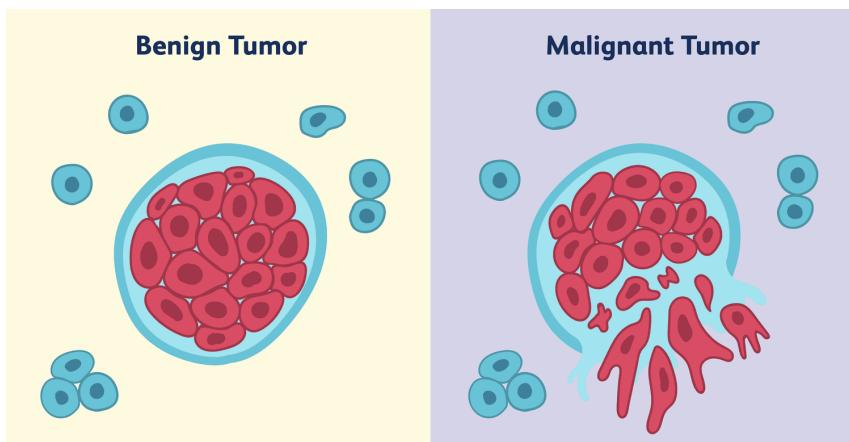
SADRŽAJ

1 UVOD	<hr/> 3
1.1 O raku dojke	<hr/> 3
1.2 O mašinskom učenju	<hr/> 4
2 ALGORITMI MAŠINSKOG UČENJA	<hr/> 5
2.1 Stabla odlučivanja	<hr/> 5
2.2 Slučajne šume	<hr/> 6
2.3 AdaBoost	<hr/> 7
2.4 XGBoost	<hr/> 8
2.5 Logistička regresija	<hr/> 9
2.6 Naivni Bajes	<hr/> 10
2 BAZA PODATAKA	<hr/> 11
2.1 Informacije o bazi	<hr/> 11
2.2 Atributi baze	<hr/> 11
2.3 Tehnike izbora atributa	<hr/> 12
3 PROGRAMSKI KOD	<hr/> 14
3.1 Biblioteke i main funkcija	<hr/> 14
3.2 Analiza baze	<hr/> 15
3.3 Izbor atributa od značaja	<hr/> 29
3.4 Kreiranje klasifikatora	<hr/> 33
3.5 Podela podataka na trening i test skup	<hr/> 34
3.6 Treniranje i testiranje	<hr/> 35
3.7 Pravljenje izveštaja	<hr/> 35
4 REZULTATI	<hr/> 38
4.1 Stablo odluke	<hr/> 38
4.2 Slučajne šume	<hr/> 42
4.3 AdaBoost	<hr/> 43
4.4 XGBoost	<hr/> 44
4.5 Logistička regresija	<hr/> 45
4.6 Naivni Bajes	<hr/> 46
4.7 Poređenje svih modela	<hr/> 47
5 ZAKLJUČAK	<hr/> 50
6 LITERATURA	<hr/> 51

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 uspelo da prezivi. 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 slike, 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 primjenjuje 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)

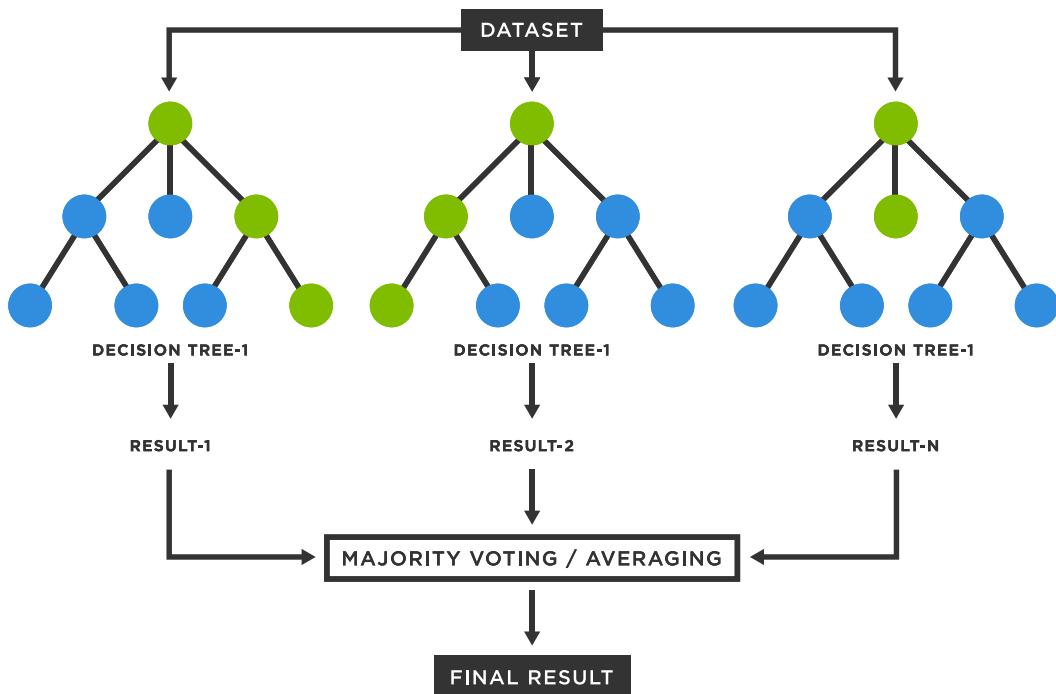
Ono što stablo odlučivanja radi je da svakom odlukom skup deli u homogenije podskupove kako bi na kraju u listovima donelo odluku o kojoj klasi je reč. Oni atributi koji skup deli u homogenije podskupove, nalaze se na višim delovima stabla. Što je stablo razgranati i ima veću dubinu, to će bolje podeliti 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.

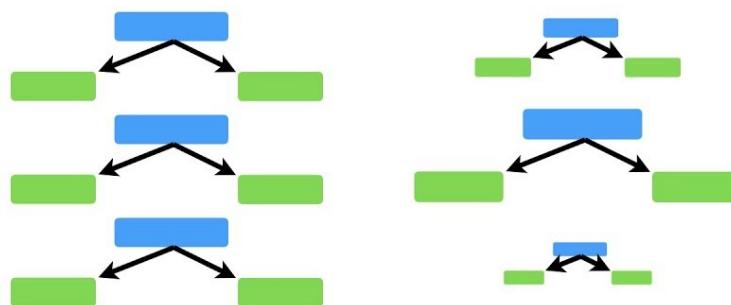
Potrebno je izabrati koliko stabla odluke se kreira, kao i njihova maksimalna dubina. Svako stablo sadrži proizvoljni skup atributa i proizvoljni skup uzoraka. Upravo u tome leži rešenje za *overfitting* i tačniju klasifikaciju, možda neko stablo neće dobro klasifikovati određeni odbirak, ali tu su ostala 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, a takođe i u *ensemble learning*. Sastavljen je od slabih modela koji zajedno agregirani daju tačan i jak 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.



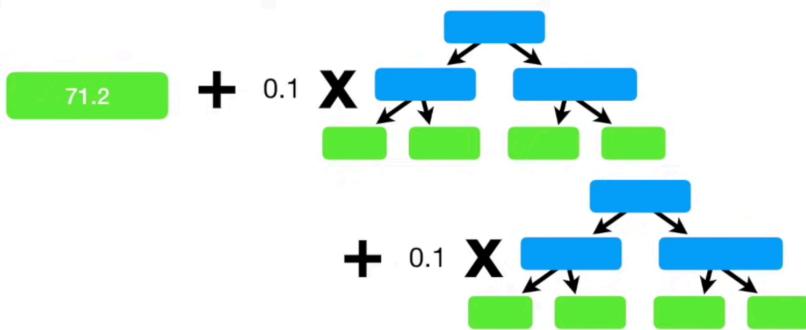
Slika 3 – Prikaz AdaBoost modela, preuzeto iz [3]

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 učenje sa nadgledanjem, a takođe i u *ensemble learning*. Bazira se na *gradient boosting* metodi koju koristi GradientBoost model. Razlika GradientBoost modela i XGBoost modela je ta što XGBoost koristi regularizaciju³ pa je iz tog razloga brži i opštiji za nove podatke, pa iz tog razloga 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]

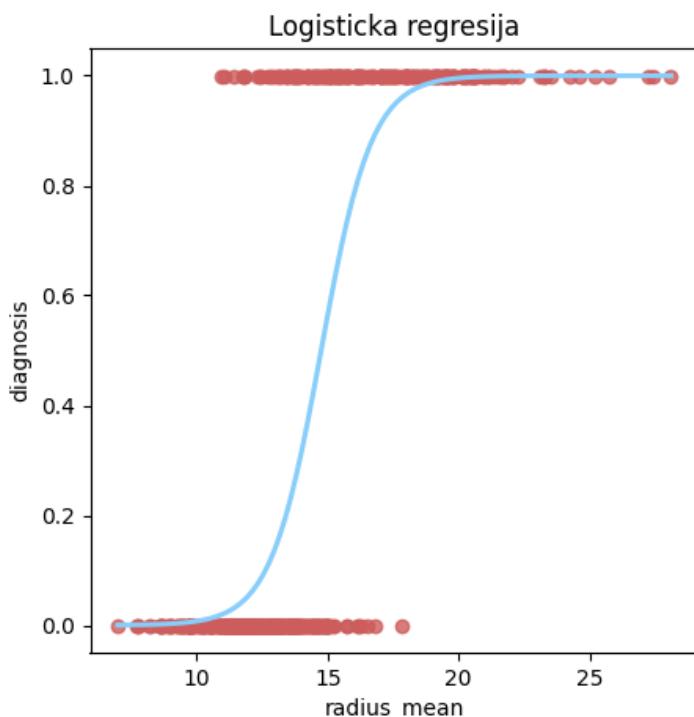
Za razliku od AdaBoost modela gde je nivo stabla odluke uvek 1, kod XGBoost modela obično se uzima da stablo ima od 2 do 5 nivoa. Stabla se takođe skaliraju kao kod AdaBoost modela, pri čemu sva stabla imaju podjednaku težinu pri glasanju. Pri primeni XGBoost modela, koriste se sledeći parametri:

- 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 klasu 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. Vrednost na kojoj odbirci stanu, predstavlja verovatnoću da taj odbirak pripada dатој 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.

2 BAZA PODATAKA

2.1 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 33 atributa koja ga detaljnije opisuju. Od tih 33 atributa, jedan predstavlja klasu od interesa, jedan predstavlja id uzorka i jedan predstavlja kolonu koja nije od znacaja jer ni za jedan atribut nema vrednost. Zaključuje se da baza ima 30 atributa od interesa.

2.2 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**). Poslednji atribut je **unnamed** i ne daje nikakvu korisnu informaciju. 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 duzinama 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**

Svi atributi su zaokruženi na 4 značajne cifre. Nema nedostajućih vrednosti u bazi ni za jedan atribut.

2.3 Tehnike izbora atributa

Postoje razne tehnike kojima se redukuje broj atributa pri čemu se biraju oni koji su najkorisniji pri daljoj analizi. Tehnike koje su uzete u obzir su sledeće:

1. entropija i informaciona dobit
2. fišerov skor
3. rekurzivna eliminacija atributa
4. gini indeks

1. **Informaciona dobit** tehnika je za određivanje atributa. Pre nego što se uvede, potrebno je uvesti pojam entropije. **Entropija** predstavlja meru razuđenosti klasa. Vrednost entropije može biti između 0 (mala razuđenost klasa) i 1 (velika razuđenost klasa)

$$E = - \sum_{i=1}^n p_i \log_2(p_i) \quad (2.1)$$

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 redukovana entropija, to je veća informaciona dobit.

$$IG = E_{roditelja} - E_{deteta} \quad (2.2)$$

2. **Fišerov skor** je tehnika koja koristi procenu maksimalne verovatnoće. Procena maksimalne verovatnoće procenjuje parametre raspodele verovatnoće maksimizovanjem funkcije verovatnoće, tako da su po pretpostavljenom statističkom modelu uočeni podaci najverovatniji. Fišerov skor predstavlja gradijent ili izvod procene maksimalne verovatnoće. Podešavanjem parametra θ maksimizuje se funkcija verovatnoće uzorka x . Fišerov skor je $u(\theta)$.

$$\begin{aligned} p(x | \theta) &= \prod_{i=1}^n f_\theta(x_i) \\ \theta_{MLE} &= \arg \max_{\theta} \log p(x | \theta) = \arg \max_{\theta} \sum_{i=1}^n \log f_\theta(x_i) \\ u(\theta) &= \nabla_{\theta} \log p(x | \theta) \end{aligned} \quad (2.3)$$

3. **Rekurzivna eliminacija atributa** je za razliku od ostalih metoda koje spadaju u metode filtriranja, ova metoda spada u metodu omotača. Razlika metode filtriranja i metode omotača je ta što se kod metode filtriranja meri relevantnost karakteristika njihovom korelacijom sa klasom, dok se kod metode omotača meri korisnost podskupa karakteristika tako što se model obučava nad tom podskupu. Algoritam radi po sledećem principu: kreće se od celog skupa atributa i model se trenira a onda i testira. Potom se skup atributa smanji za 1 atribut i model se opet trenira i testiraju se njegove performanse. Proces se nastavlja i kao rezultat izdvajaju se traženi atributi. Potrebno je navesti koliki je željeni broj atributa pre samog procesa.
4. **Gini indeks** meri verovatnoću da se proizvoljna instanca pogrešno klasificuje. Vrednost gini indeksa može biti između 0 (mala razuđenost klasa) i 0.5 (velika razuđenost klasa). Formula glasi:

$$Gini = 1 - \sum_{i=1}^j p_i^2 \quad (2.4)$$

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

Traže se oni atributi koji imaju što veću informacionu dobit, što veći fišerov skor, a što manji gini indeks kako bi na bolji način podelili trenutni skup u dva homogenija skupa.

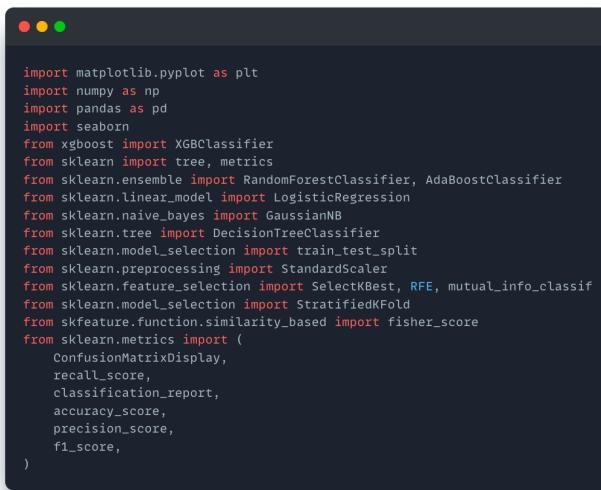
Ono što stablo odlučivanja radi je da svakom odlukom skup deli u homogenije podskupe kako bi na kraju u listovima donelo odluku o kojoj klasi je reč. Što je stablo razgranatije i ima veću debinu, to će bolje podeliti trenutne podatke, ali isto tako može doći do težeg prilagođavanja novim podacima jer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

- ranim zaustavljanjem i *pre-prunning-om* – sprečavanje da stablo ne raste previše u širinu i dubinu
- *post-prunning-om* – 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 metodom* i usrednjavanjem više modela poput *random forest-a*

3 PROGRAMSKI KOD

3.1 Biblioteke i main funkcija

Kako bi mogle da se koriste već gotove funkcije pogodne za treniranje, klasifikaciju i celokupnu 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.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, RFE, mutual_info_classif
from sklearn.model_selection import StratifiedKFold
from skfeature.function.similarity_based import fisher_score
from sklearn.metrics import (
    ConfusionMatrixDisplay,
    recall_score,
    classification_report,
    accuracy_score,
    precision_score,
    f1_score,
)
```

Slika 3 – 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 funkciji se nalaze 3 funkcije koje predstavljaju 3 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. *choose_attributes()* – nakon analize potrebno je odabrati koji atributi su od značaja pri primeni mašinskog učenja
3. *classify()* – prvo se model trenira nad trening skupom i nakon toga se testira algoritam nad test skupom za različite tehnike mašinskog učenja



```
if __name__ == "__main__":
    # analiziranje Klasa i atributa
    dataset = analyze_base()
    # izbor atributa
    dataset = choose_attributes(dataset)
    # podela skupa podataka, treniranje i klasifikacija razlicitim modelima
    classify(dataset)
```

Slika 4 – Main funkcija

3.2 Analiza baze

Analiza baze sprovodi se kroz sledeće funkcije:

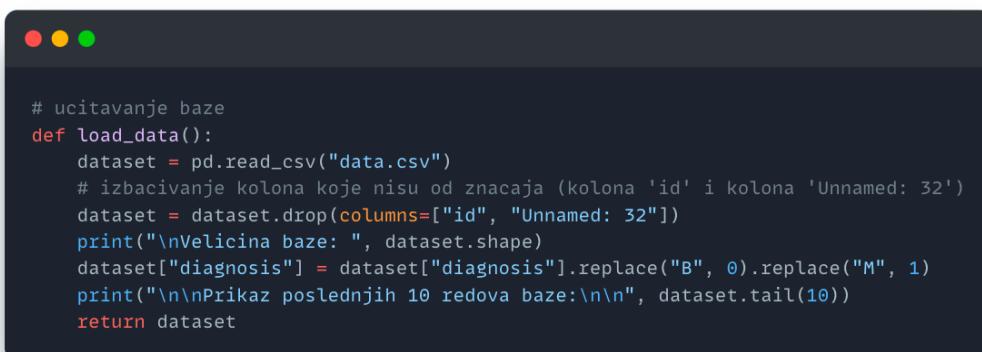
1. *load_data()* – učitava bazu i prilagođava je za dalju obradu
2. *missing_values()* – proverava da li postoje nedostajuće vrednosti u bazi
3. *class_distribution()* – prikazuje raspodelu klasa kako bi se stekao uvid u balansiranost klasa
4. *correlation()* – određuju se međusobne korelisanosti atributa kao i sa klasom
5. *plot_class_histogram_for_every_attribute()* – prikazuje histograme kako svaki atribut razdvaja klase



```
# analiza baze
def analyze_base():
    dataset = load_data()
    missing_values(dataset)
    class_distribution(dataset)
    correlation(dataset)
    plot_class_histogram_for_every_attribute(dataset)
    return dataset
```

Slika 5 – 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 6 – Učitavanje baze

Baza sadrži 33 kolona od kojih su 30 kolona atributa. Na raspolaganju su 569 odbiraka koji predstavljaju redove baze.



Slika 7 – Dimenzije baze

A screenshot of a dark-themed terminal window. The title bar says "Prikaz poslednjih 10 redova baze:". Below it is a table with 10 rows of data. The columns are labeled: diagnosis, radius_mean, texture_mean, ..., concave points_worst, symmetry_worst, fractal_dimension_worst. The data values are as follows:

559	0	11.51	23.93	...	0.09653	0.2112	0.08732
560	0	14.05	27.15	...	0.10480	0.2250	0.08321
561	0	11.20	29.37	...	0.00000	0.1566	0.05905
562	1	15.22	30.62	...	0.23560	0.4089	0.14090
563	1	20.92	25.09	...	0.25420	0.2929	0.09873
564	1	21.56	22.39	...	0.22160	0.2060	0.07115
565	1	20.13	28.25	...	0.16280	0.2572	0.06637
566	1	16.60	28.08	...	0.14180	0.2218	0.07820
567	1	20.60	29.33	...	0.26500	0.4987	0.12400
568	0	7.76	24.54	...	0.00000	0.2871	0.07039

Slika 8 – Prikaz poslednjih 10 redova baze

Sledeći korak je provera da li postoje nedostajuće vrednosti:

A screenshot of a dark-themed terminal window. The code is as follows:

```
#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 9 – Provera da li postoje nedostajuće vrednosti

Sa sledeće slike zaključuje se da nema vrednosti koje nedostaju u bazi.

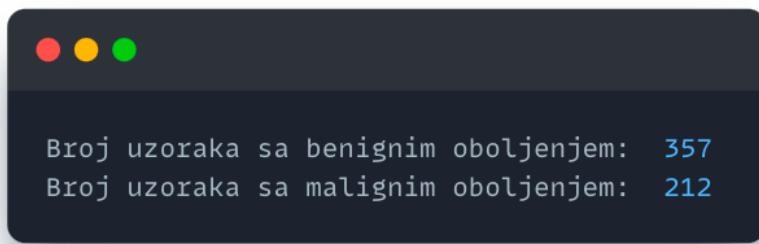
```
Nedostajuce vrednosti po kolonama:  
diagnosis          0  
radius_mean         0  
texture_mean        0  
perimeter_mean     0  
area_mean           0  
smoothness_mean    0  
compactness_mean   0  
concavity_mean     0  
concave_points_mean 0  
symmetry_mean      0  
fractal_dimension_mean 0  
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  
  
Ukupan broj nedostajucih vrednosti: 0
```

Slika 10 – Prikaz broja nedostajućih vrednosti

Sledeći korak je određivanje raspodele benignih i malignih tumora u klasi dijagnoza.

```
# distribucija klase  
def class_distribution(dataset):  
    numBenign = sum(dataset["diagnosis"] == 0)  
    numMalignant = sum(dataset["diagnosis"] == 1)  
    print("Broj uzoraka sa benignim oboljenjem: ", numBenign)  
    print("Broj uzoraka sa malignim oboljenjem: ", numMalignant, "\n")
```

Slika 11 – Distribucija klasa

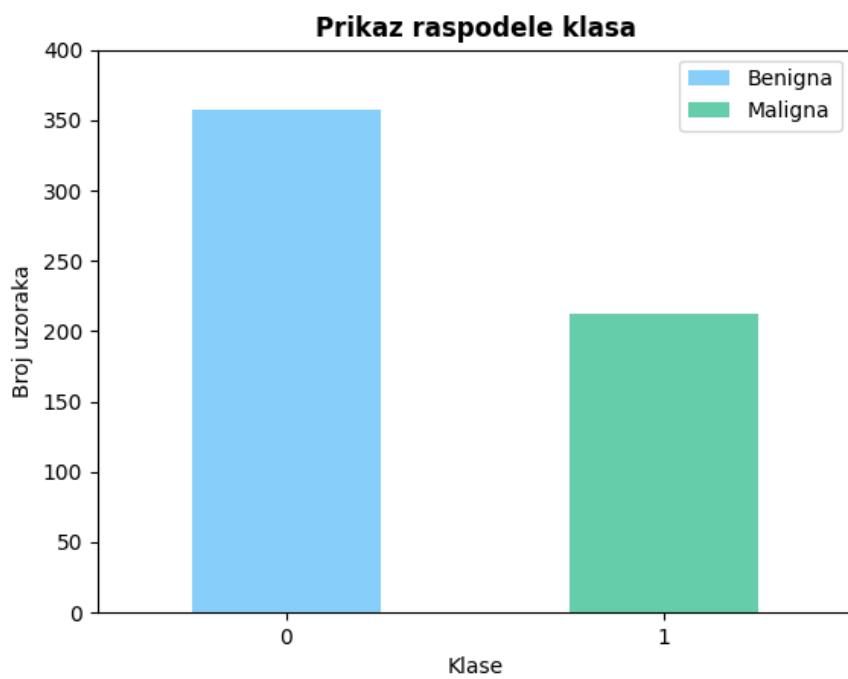


Slika 12 – Prikaz distribucije klasa

Raspodela je prikazana i vizualno:



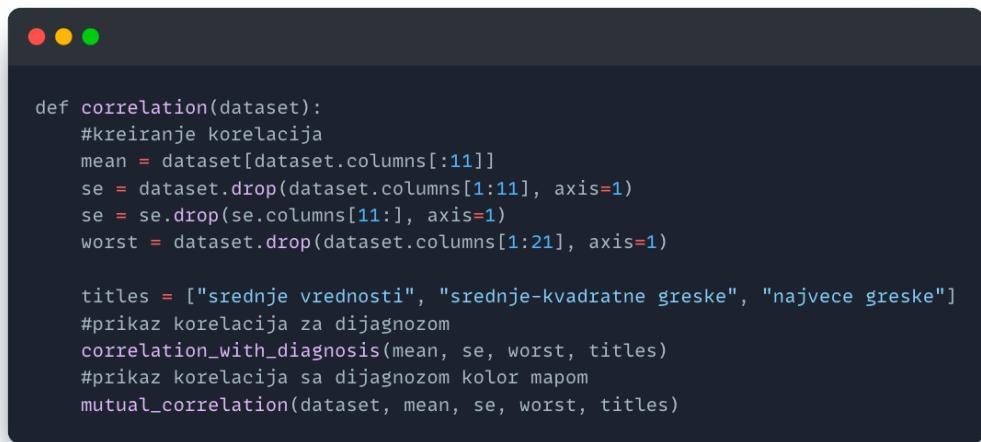
Slika 13 – Distribucija klasa, vizualni prikaz



Slika 14 – 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 nedovoljno pojavljuje u trenirajućem skupu a dosta u testirajućem skupu i obrnuto. Kao posledica pojaviće se velika greška klasifikacije.

Sledeći korak odnosi se na korelisanost atributa (međusobna korelisanost kao i korelisanost sa klasom):



```
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 15 – Određivanje korelisanosti atributa

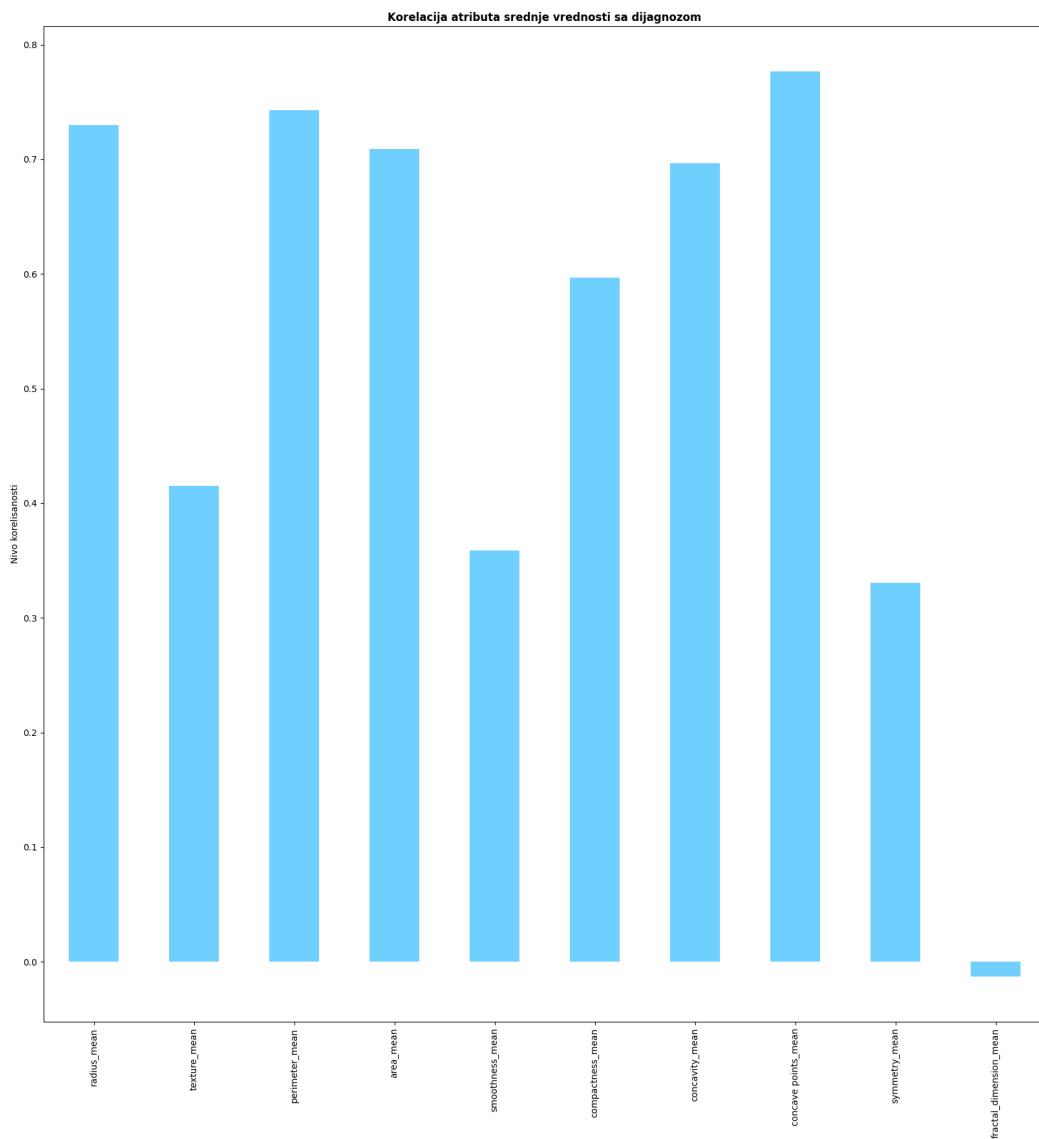
Postoje dva tipa korelisanosti: korelisanost atributa sa dijagnozom i međusobna korelisanost atributa. Poželjni su atributi koji su visoko korelisani sa dijagnozom. Ipak, može da se desi da su dva različita atributa jako korelisana sa dijagnozom, a da su takođe i međusobno jako korelisani. U takvom slučaju 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:



```
# korelacija atributa iz svake grupe sa klasom
def correlation_with_diagnosis(mean, se, worst, titles):
    data = [mean, se, worst]
    colors = ["lightskyblue", "mediumaquamarine", "peachpuff"]
    file_names = ["mean_corr_class.png", "se_corr_class.png", "worst_corr_class.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], figsize=(20, 20))
        plt.title("Korelacija atributa " + titles[i] + " sa dijagnozom", weight="bold")
        plt.ylabel("Nivo korelisanosti")
        plt.savefig("./images/" + file_names[i])
        plt.show()
```

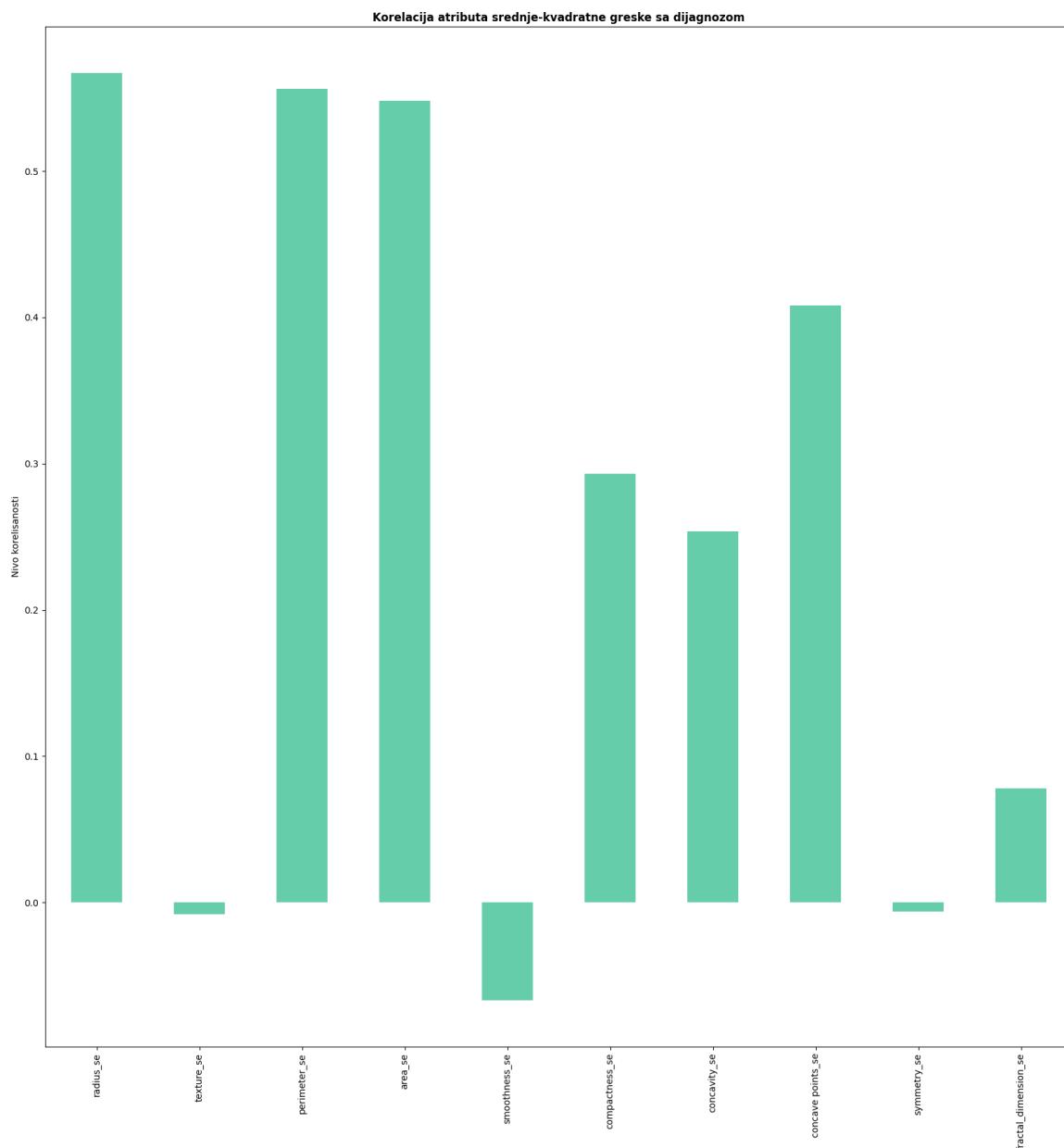
Slika 16 – Određivanje korelisanosti atributa sa klasom

Klasifikacija raka dojke pomoću mašinskog učenja



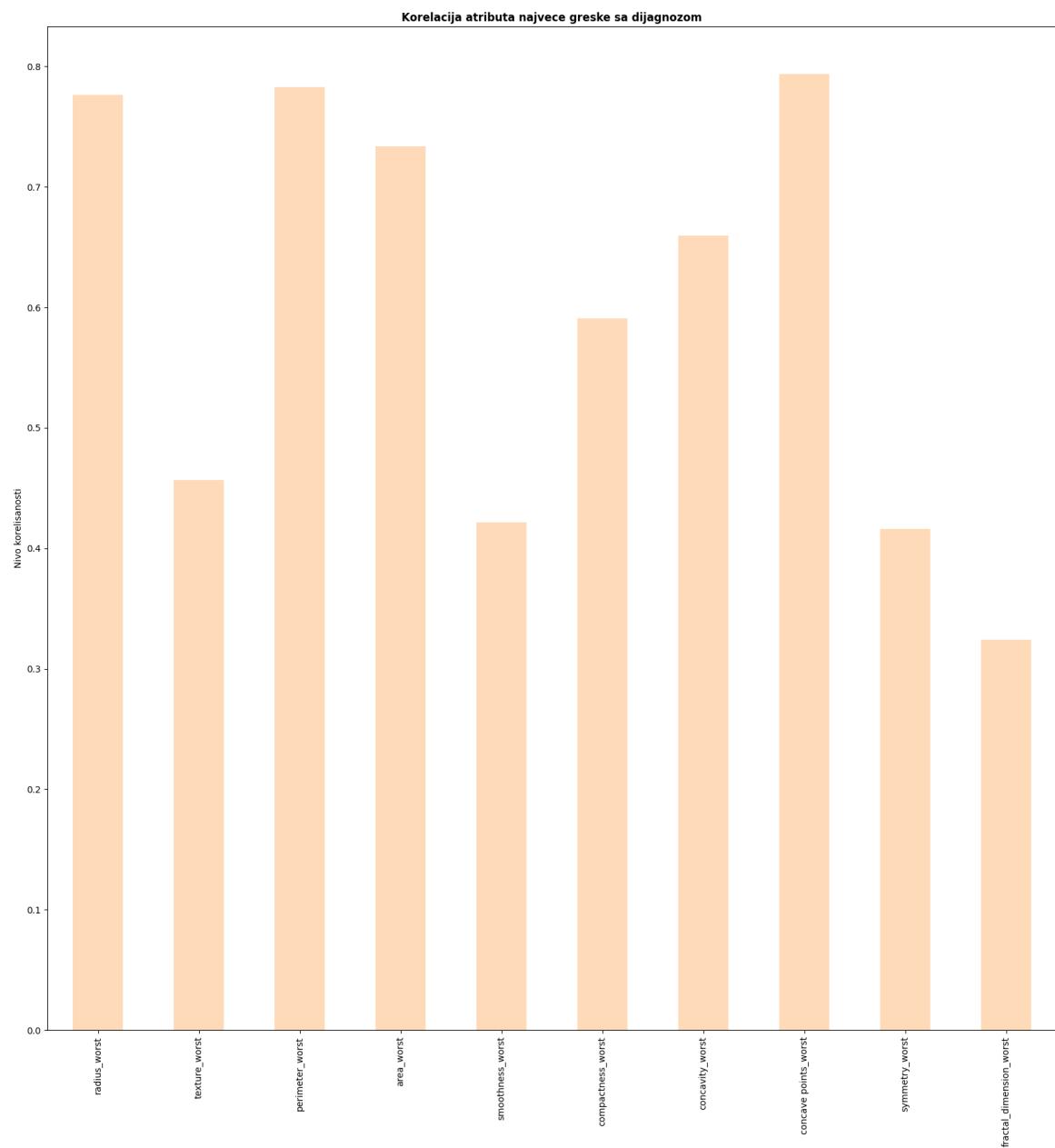
Slika 17 – Korelisanost atributa srednje vrednosti sa klasom

Klasifikacija raka dojke pomoću mašinskog učenja



Slika 18 – Korelisanost atributa srednje-kvadratne greške sa klasom

Klasifikacija raka dojke pomoću mašinskog učenja

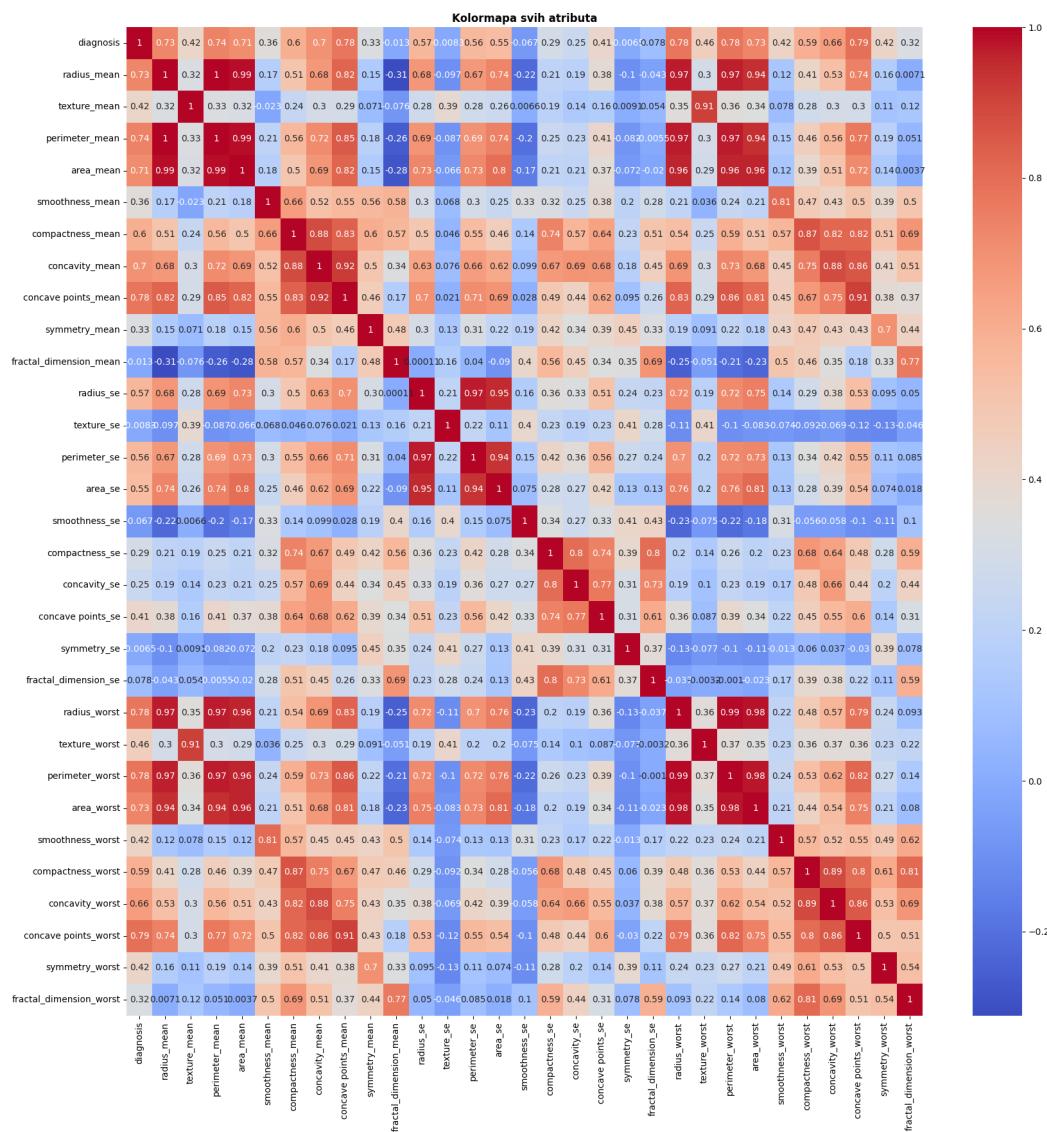


Slika 19 – Korelisanost atributa najvećeg odstupanja sa klasom

U drugom slučaju potrebno je pronaći međusobnu korelisanost atributa kako bi se pronašli visokokorelisani atributi koji se mogu izbaciti za dalju analizu.

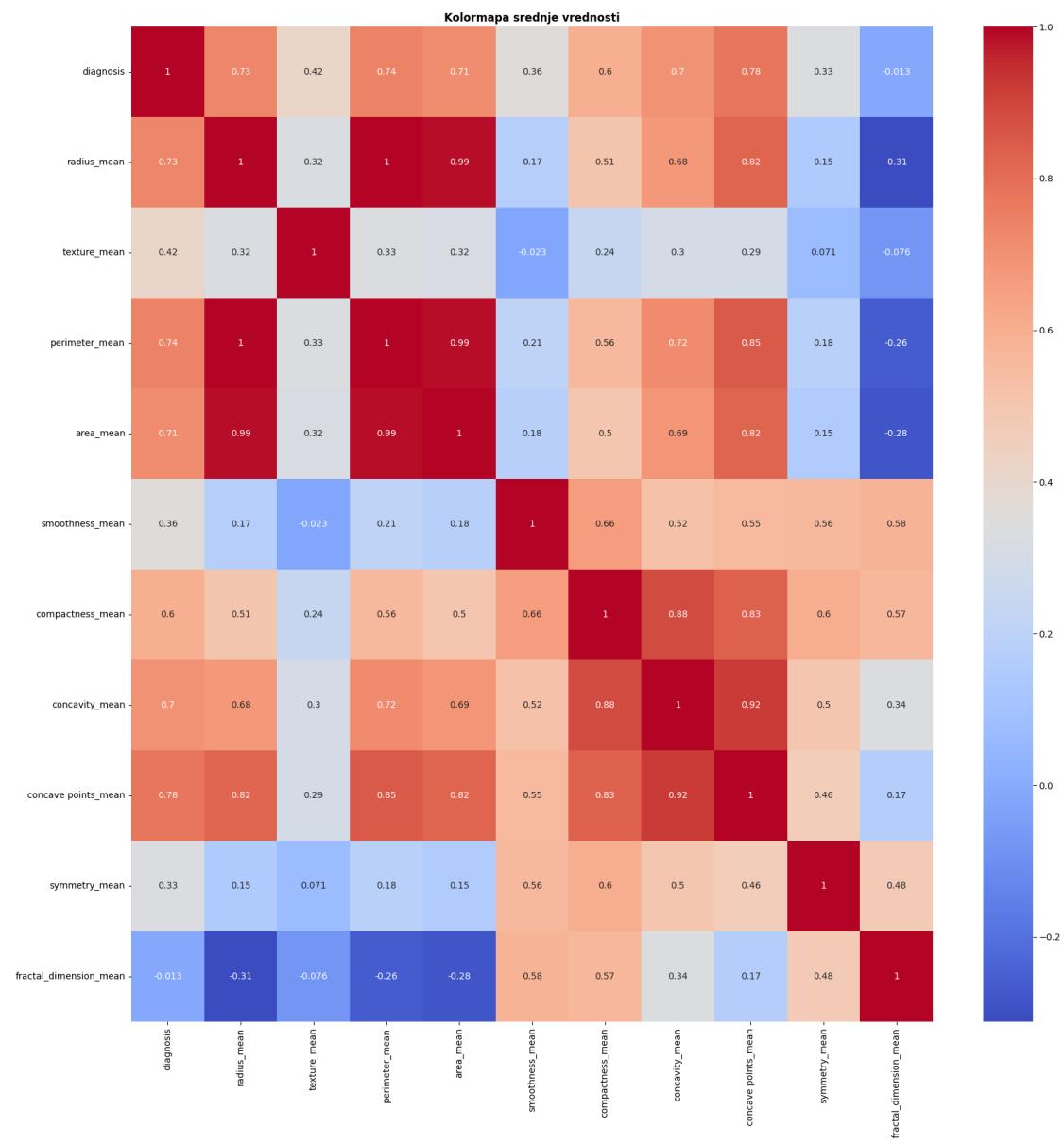


Slika 20 – Međusobna korelisanost atributa



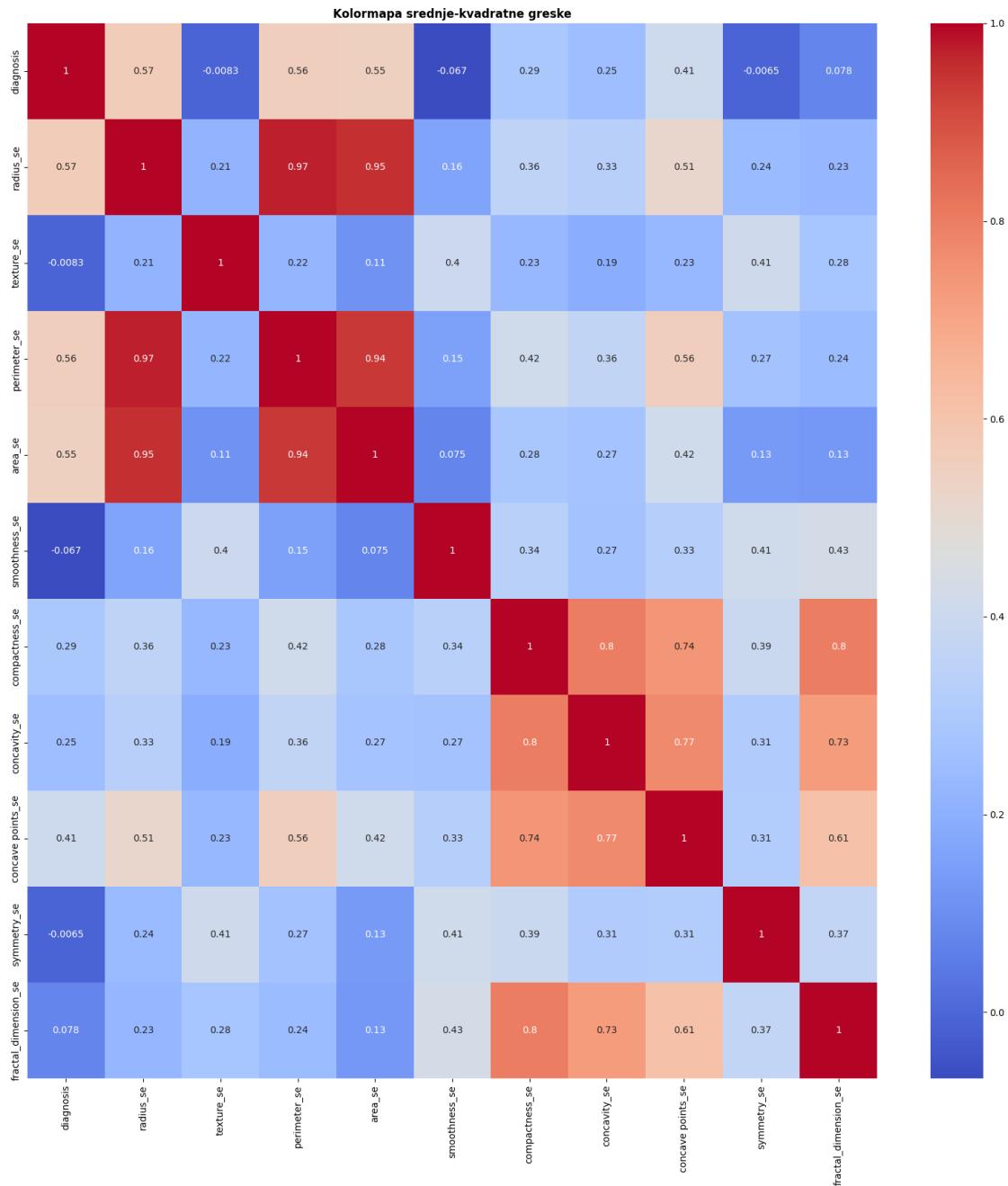
Slika 21 – Međusobna korelisanost svih atributa

Klasifikacija raka dojke pomoću mašinskog učenja



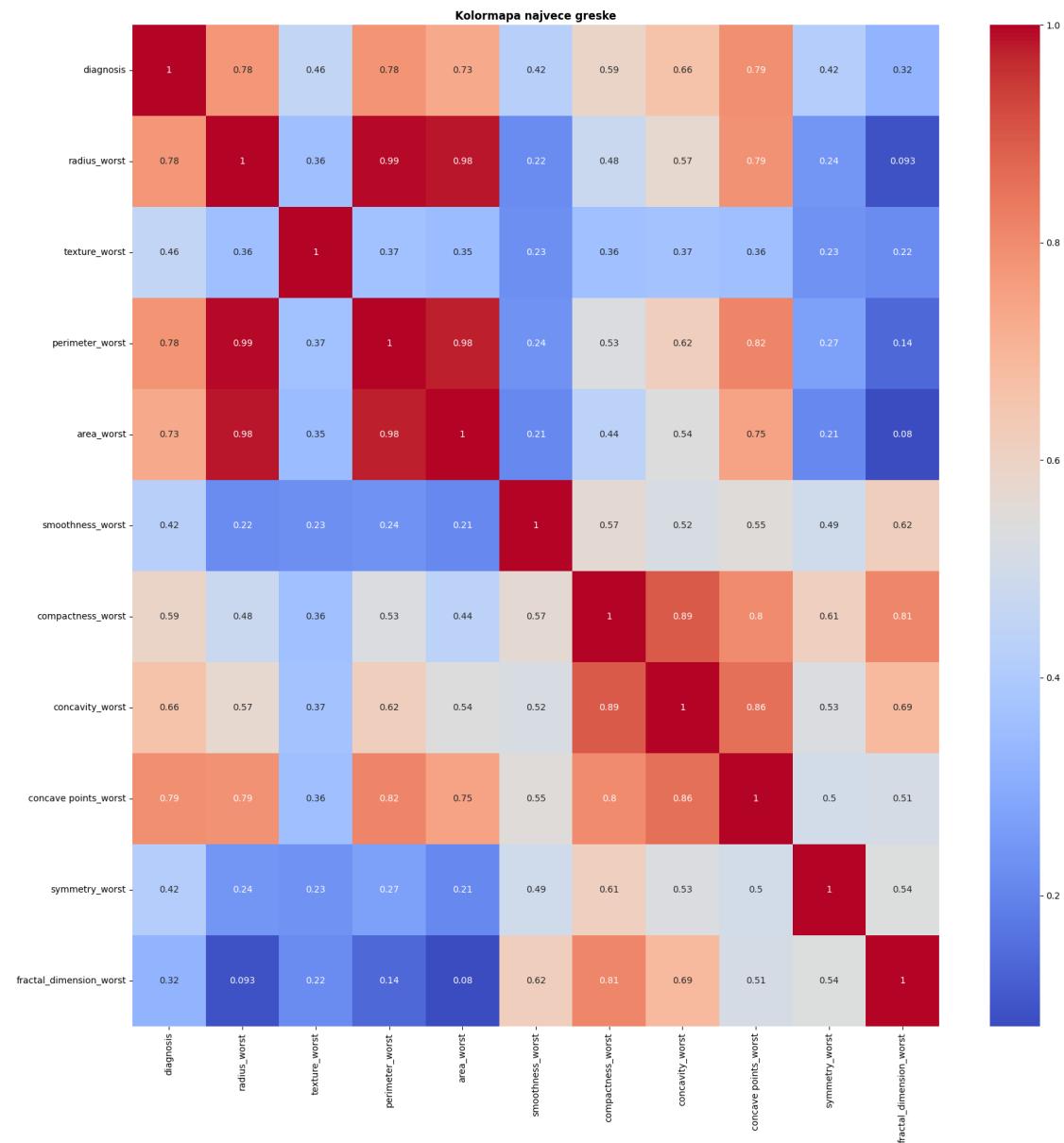
Slika 22 – Međusobna korelisanost atributa srednje vrednosti

Klasifikacija raka dojke pomoću mašinskog učenja



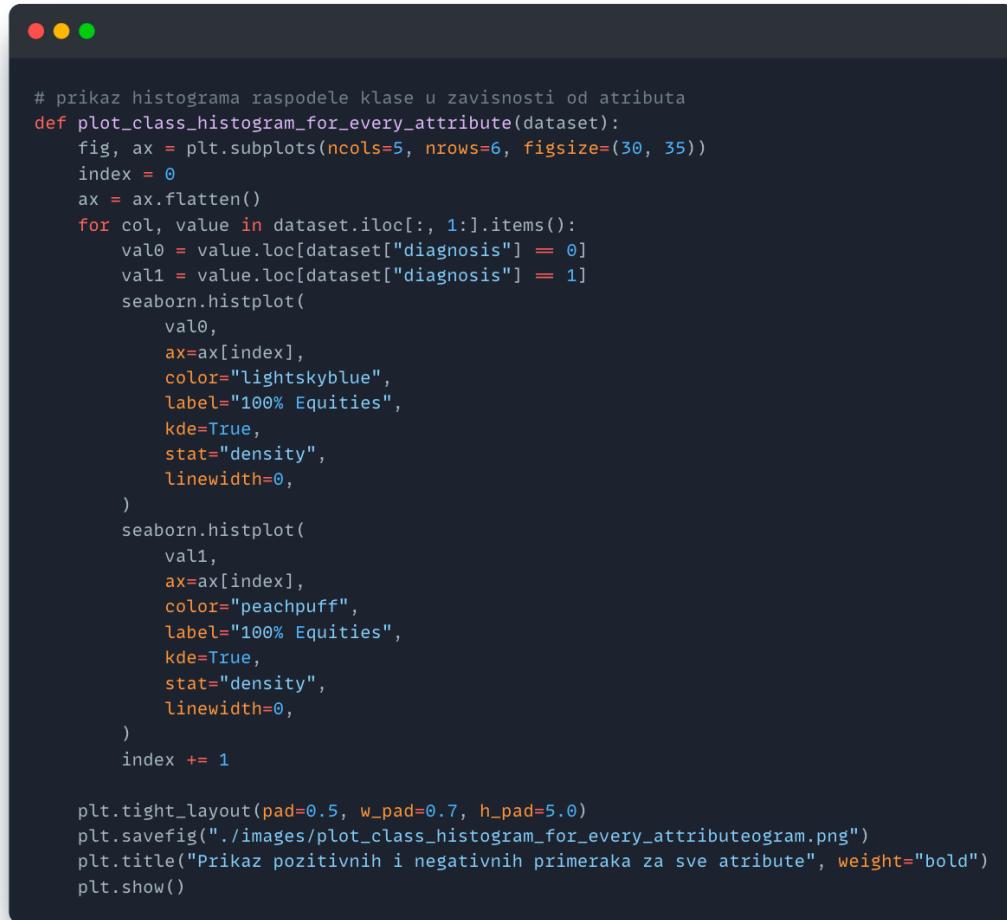
Slika 23 – Međusobna korelisanost atributa srednje-kvadratne greške

Klasifikacija raka dojke pomoću mašinskog učenja



Slika 24 – Međusobna korelisanost najvećeg odstupanja

Za svaki atribut može se prikazati histogram raspodele klase. Takav prikaz bi mogao da pomogne u razumevanju koliko svaki atribut razdvaja klase.



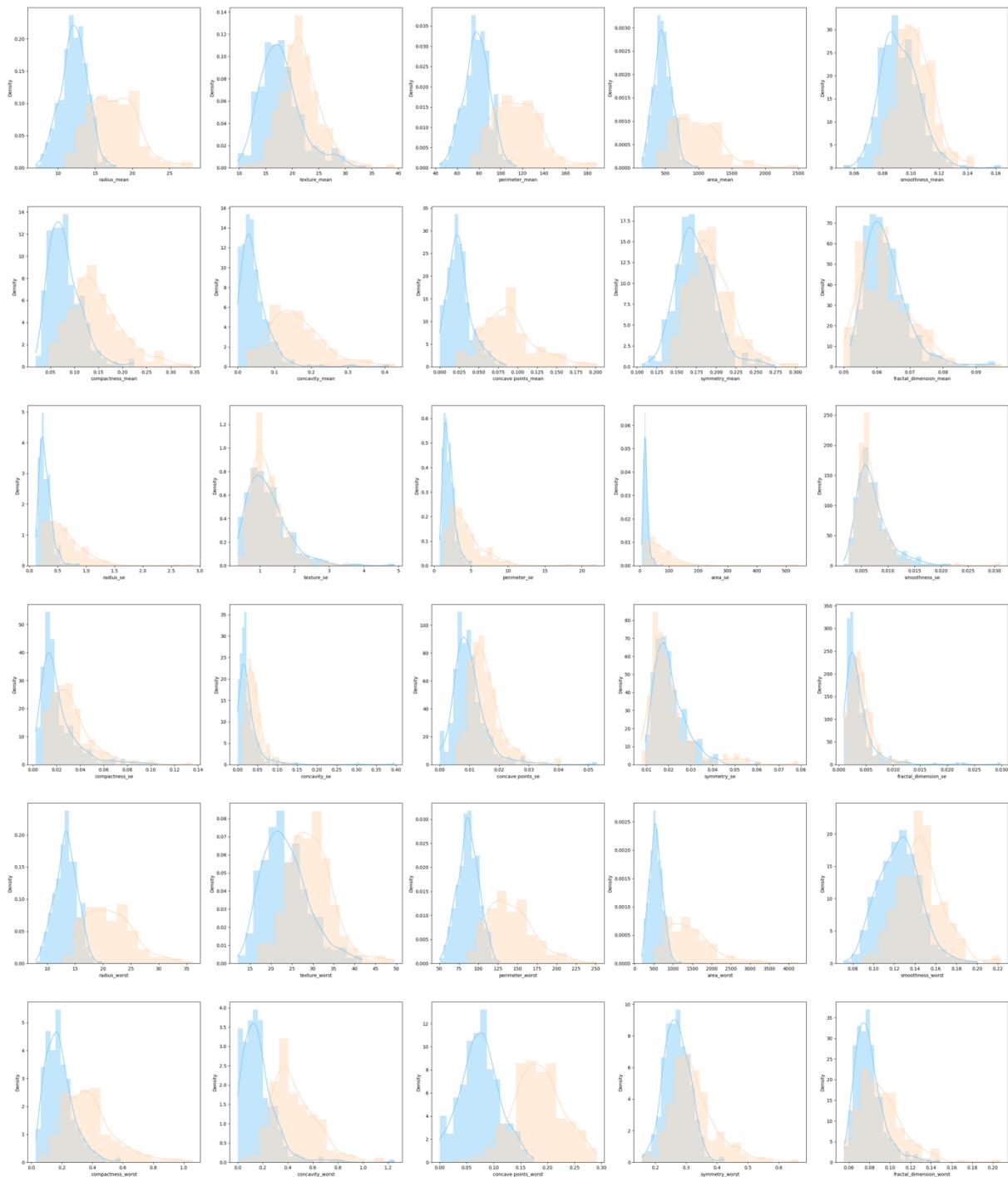
```
# prikaz histograma raspodele klase u zavisnosti od atributa
def plot_class_histogram_for_every_attribute(dataset):
    fig, ax = plt.subplots(ncols=5, nrows=6, figsize=(30, 35))
    index = 0
    ax = ax.flatten()
    for col, value in dataset.iloc[:, 1:].items():
        val0 = value.loc[dataset["diagnosis"] == 0]
        val1 = value.loc[dataset["diagnosis"] == 1]
        seaborn.histplot(
            val0,
            ax=ax[index],
            color="lightskyblue",
            label="100% Equities",
            kde=True,
            stat="density",
            linewidth=0,
        )
        seaborn.histplot(
            val1,
            ax=ax[index],
            color="peachpuff",
            label="100% Equities",
            kde=True,
            stat="density",
            linewidth=0,
        )
        index += 1

    plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
    plt.savefig("./images/plot_class_histogram_for_every_attributeogram.png")
    plt.title("Prikaz pozitivnih i negativnih primeraka za sve atribute", weight="bold")
    plt.show()
```

Slika 25 – Kod za prikaz histograma raspodele klase za svaki atribut

Klasifikacija raka dojke pomoću mašinskog učenja

Sa sledeće slike može se primetiti kako svaki atribut razdvaja klase. Što su klase razdvojenije to će klasifikator biti tačniji.



Slika 26 – Prikaz histograma raspodele klase za svaki atribut

3.3 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 17** primetno je da atribut *fractal_dimension_mean* ima malu korelisanost sa klasom te on nije od značaja. Takođe, sa **Slike 18** 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 19** se primećuje da su svi atributi dobro korelirani sa klasom.

- Analiza međusobne korelisanosti atributa

Sa **Slike 21** primećuje se velika korelisanost kod sledećih grupa atributa:

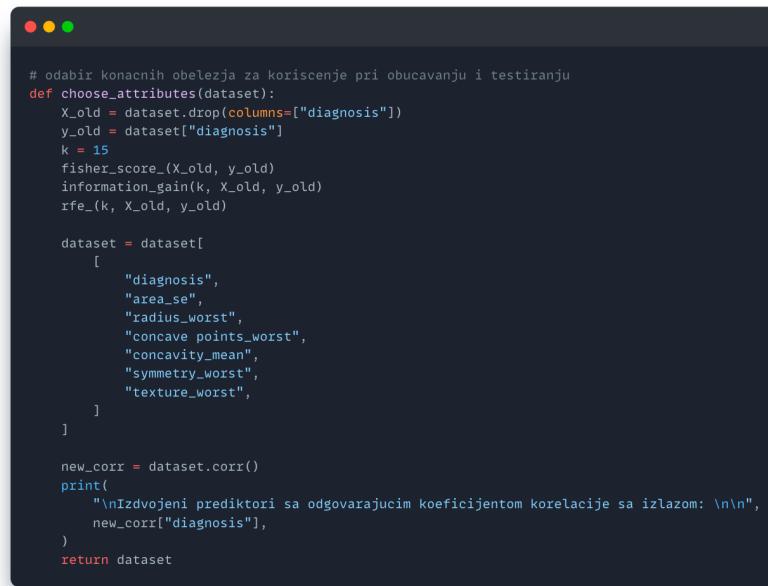
1. *radius_mean*, *perimeter_mean* i *area_mean*
2. *radius_se*, *perimeter_se* i *area_se*
3. *radius_worst*, *perimeter_worst* i *area_worst*
4. *prva i treća grupa međusobno*
5. *area_mean* i *area_se*
6. *compactness_mean*, *concavity_mean* i *concave points_mean*
7. *compactness_worst*, *concavity_worst* i *concave points_worst*
8. *šesta i sedma grupa*
9. *compactness_mean* sa prvom grupom

- Analiza pomoću tehnika za selekciju atributa

Koriste se već pomenute tehnike:

1. *fišerov skor*
2. *information gain*
3. rekurzivna eliminacija atributa

Klasifikacija raka dojke pomoću mašinskog učenja

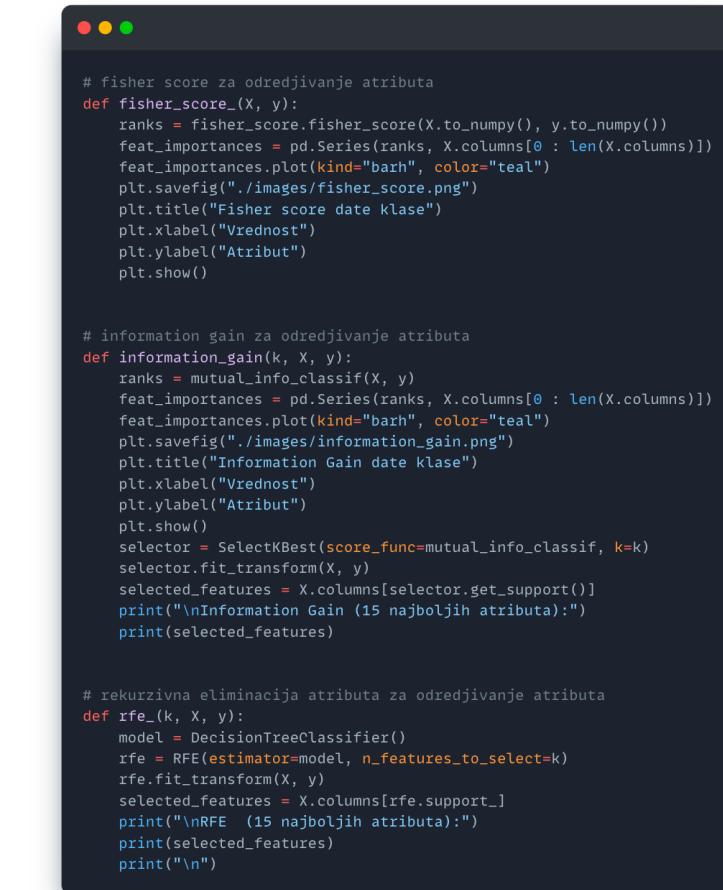


```
# odabir konačnih objeležja za koriscenje pri obucavanju i testiranju
def choose_attributes(dataset):
    X_old = dataset.drop(columns=["diagnosis"])
    y_old = dataset["diagnosis"]
    k = 15
    fisher_score_(X_old, y_old)
    information_gain(k, X_old, y_old)
    rfe_(k, X_old, y_old)

    dataset = dataset[
        [
            "diagnosis",
            "area_se",
            "radius_worst",
            "concave_points_worst",
            "concavity_mean",
            "symmetry_worst",
            "texture_worst",
        ]
    ]

    new_corr = dataset.corr()
    print(
        "\nIzdvojeni prediktori sa odgovarajućim koeficijentom korelacije sa izlazom: \n\n",
        new_corr["diagnosis"],
    )
)
return dataset
```

Slika 27 – Biranje konačnih atributa za dalju analizu



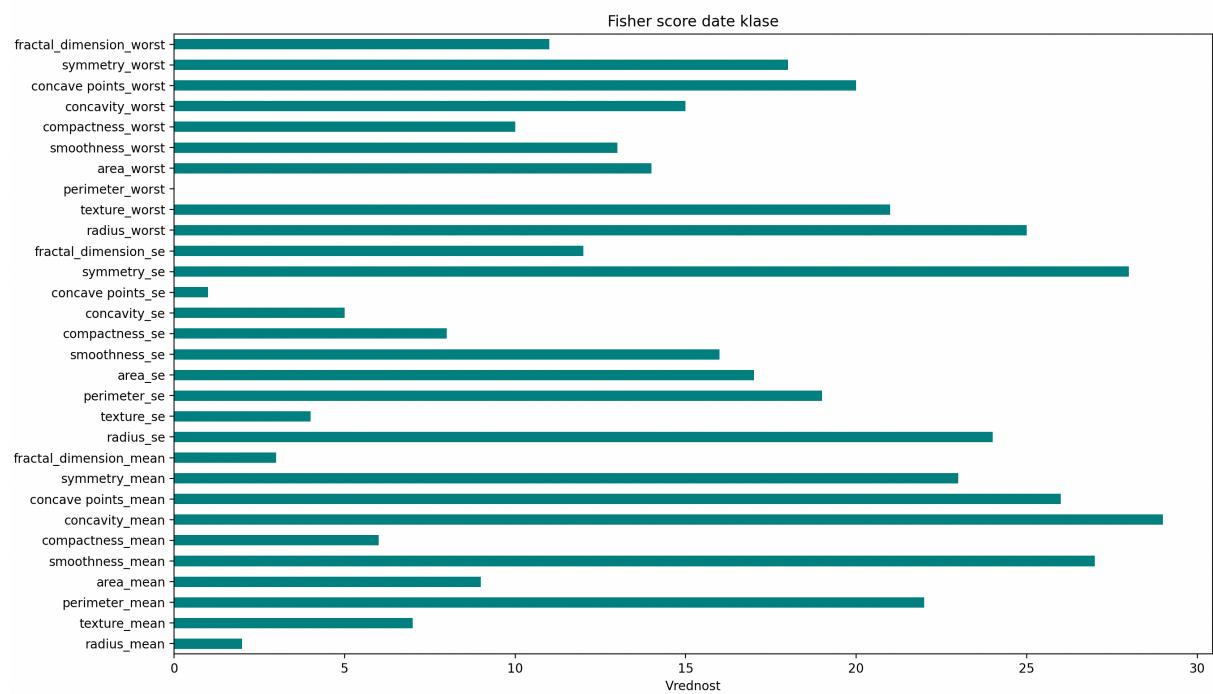
```
# fisher score za određivanje atributa
def fisher_score_(X, y):
    ranks = fisher_score.fisher_score(X.to_numpy(), y.to_numpy())
    feat_importances = pd.Series(ranks, X.columns[0 : len(X.columns)])
    feat_importances.plot(kind="barh", color="teal")
    plt.savefig("./images/fisher_score.png")
    plt.title("Fisher score date klase")
    plt.xlabel("Vrednost")
    plt.ylabel("Atribut")
    plt.show()

# information gain za određivanje atributa
def information_gain(k, X, y):
    ranks = mutual_info_classif(X, y)
    feat_importances = pd.Series(ranks, X.columns[0 : len(X.columns)])
    feat_importances.plot(kind="barh", color="teal")
    plt.savefig("./images/information_gain.png")
    plt.title("Information Gain date klase")
    plt.xlabel("Vrednost")
    plt.ylabel("Atribut")
    plt.show()
    selector = SelectKBest(score_func=mutual_info_classif, k=k)
    selector.fit_transform(X, y)
    selected_features = X.columns[selector.get_support()]
    print("\nInformation Gain (15 najboljih atributa):")
    print(selected_features)

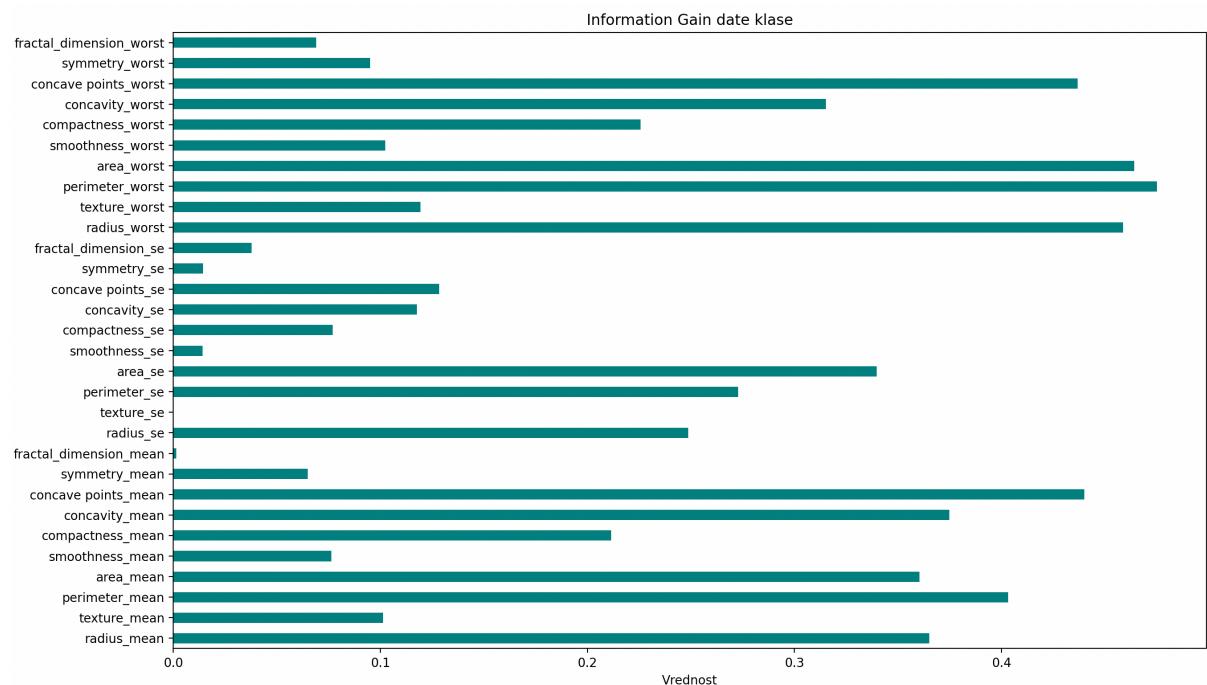
# rekursivna eliminacija atributa za određivanje atributa
def rfe_(k, X, y):
    model = DecisionTreeClassifier()
    rfe = RFE(estimator=model, n_features_to_select=k)
    rfe.fit_transform(X, y)
    selected_features = X.columns[rfe.support_]
    print("\nRFE (15 najboljih atributa):")
    print(selected_features)
    print("\n")
```

Slika 28 – Tehnike za selekciju atributa

Klasifikacija raka dojke pomoću mašinskog učenja



Slika 29 – Fišerov skor za sve atribute



Slika 30 – Information gain za sve atribute

```
Information Gain izabrani atributi:  
'radius_mean', 'perimeter_mean', 'area_mean', 'compactness_mean',  
'concavity_mean', 'concave points_mean', 'radius_se', 'perimeter_se',  
'area_se', 'radius_worst', 'perimeter_worst', 'area_worst',  
'compactness_worst', 'concavity_worst', 'concave points_worst'  
  
RFE izabrani atributi:  
'concave points_mean', 'fractal_dimension_mean', 'radius_se',  
'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',  
'radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst',  
'smoothness_worst', 'compactness_worst', 'concavity_worst',  
'concave points_worst'
```

Slika 31 – Najboljih 15 atributita prema IG i RFE

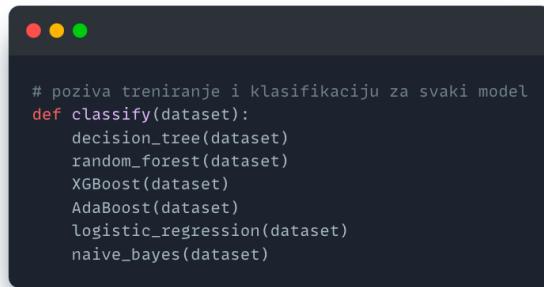
Nakon svih analiza odlučeno je da se izaberu sledeći atributi:

```
Izdvojeni prediktori sa odgovarajućim koeficijentom korelacije sa izlazom:  
area_se           0.548236  
radius_worst      0.776454  
concave points_worst 0.793566  
concavity_mean    0.696360  
symmetry_worst    0.416294  
texture_worst      0.456903
```

Slika 32 – Prikaz izabranih atributa

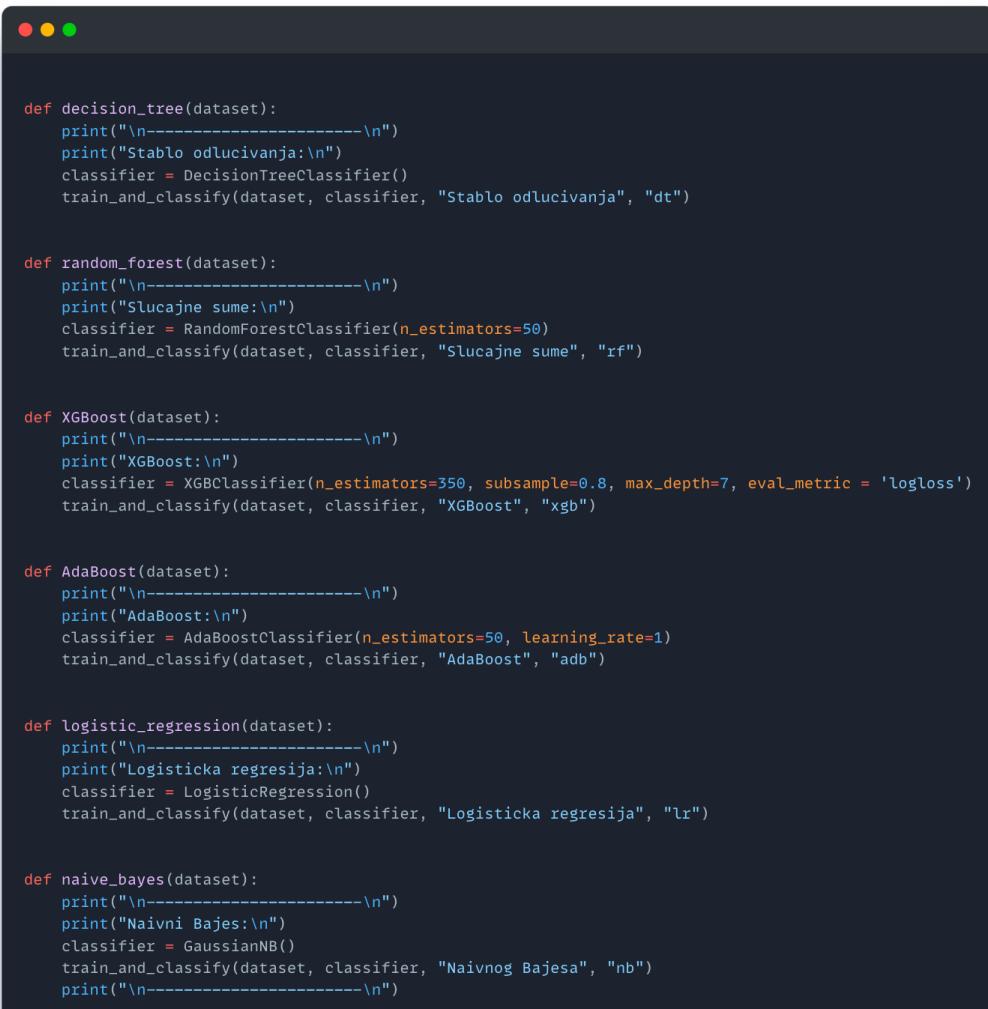
3.4 Kreiranje klasifikatora

Za svaki model se kreira klasifikator i poziva se funkcija `train_and_classify()` koja vrši dalje obradu.



```
# poziva treniranje i klasifikaciju za svaki model
def classify(dataset):
    decision_tree(dataset)
    random_forest(dataset)
    XGBoost(dataset)
    AdaBoost(dataset)
    logistic_regression(dataset)
    naive_bayes(dataset)
```

Slika 33 – Funkcija koja poziva klasifikaciju za svaki model



```
def decision_tree(dataset):
    print("\n-----\n")
    print("Stablo odlucivanja:\n")
    classifier = DecisionTreeClassifier()
    train_and_classify(dataset, classifier, "Stablo odlucivanja", "dt")

def random_forest(dataset):
    print("\n-----\n")
    print("Slucajne sume:\n")
    classifier = RandomForestClassifier(n_estimators=50)
    train_and_classify(dataset, classifier, "Slucajne sume", "rf")

def XGBoost(dataset):
    print("\n-----\n")
    print("XGBoost:\n")
    classifier = XGBClassifier(n_estimators=350, subsample=0.8, max_depth=7, eval_metric = 'logloss')
    train_and_classify(dataset, classifier, "XGBoost", "xgb")

def AdaBoost(dataset):
    print("\n-----\n")
    print("AdaBoost:\n")
    classifier = AdaBoostClassifier(n_estimators=50, learning_rate=1)
    train_and_classify(dataset, classifier, "AdaBoost", "adb")

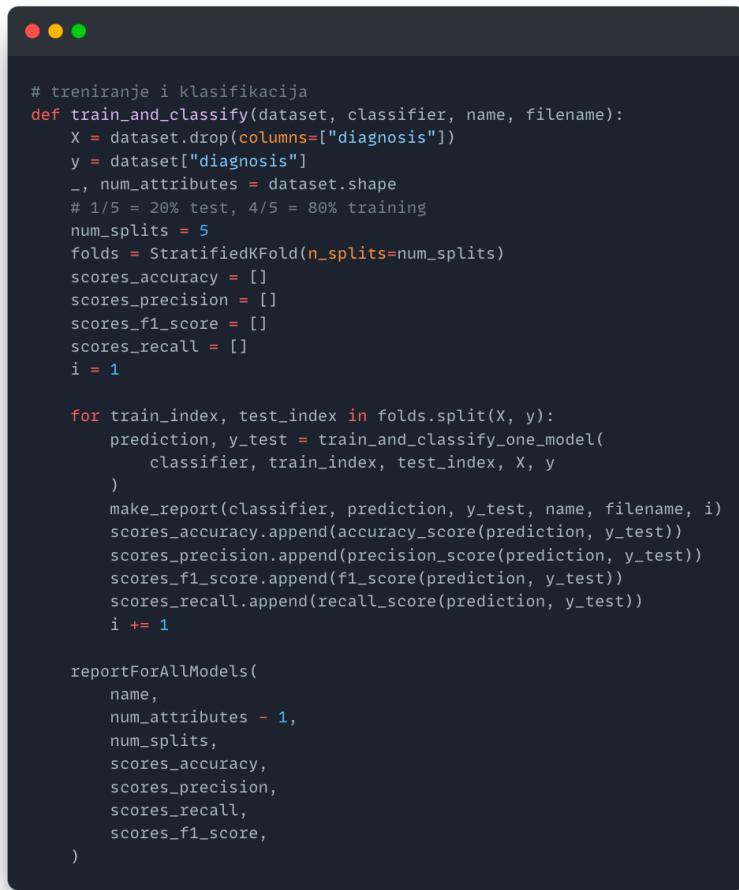
def logistic_regression(dataset):
    print("\n-----\n")
    print("Logisticka regresija:\n")
    classifier = LogisticRegression()
    train_and_classify(dataset, classifier, "Logisticka regresija", "lr")

def naive_bayes(dataset):
    print("\n-----\n")
    print("Naivni Bajes:\n")
    classifier = GaussianNB()
    train_and_classify(dataset, classifier, "Naivnog Bajesa", "nb")
    print("\n-----\n")
```

Slika 34 – Kreiranje svakog modela

3.5 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. Moguće je da pri odabiru skupa za treniranje i testiranje u velikoj meri variraju performanse klasifikatora. Kako bi dobili realan rezultat, potrebno je izvršiti kros-validaciju. Izabrana je K-Fold kros-validaciju gde je za K izabran broj 5 kako bi time bilo obezbeđeno da pri svakoj iteraciji postoji 80% podataka u trening skupu i 20% u test skupu. Za svaku iteraciju vrši se klasifikacija i pravi se izveštaj performanse tog klasifikatora, a na samom kraju prvi se izveštaj usrednjeni za sve modele kako bi konačan rezultat bio realan.



```
# treniranje i klasifikacija
def train_and_classify(dataset, classifier, name, filename):
    X = dataset.drop(columns=["diagnosis"])
    y = dataset["diagnosis"]
    _, num_attributes = dataset.shape
    # 1/5 = 20% test, 4/5 = 80% training
    num_splits = 5
    folds = StratifiedKFold(n_splits=num_splits)
    scores_accuracy = []
    scores_precision = []
    scores_f1_score = []
    scores_recall = []
    i = 1

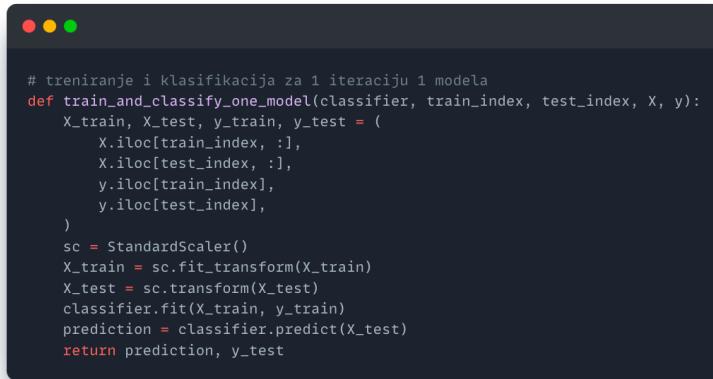
    for train_index, test_index in folds.split(X, y):
        prediction, y_test = train_and_classify_one_model(
            classifier, train_index, test_index, X, y
        )
        make_report(classifier, prediction, y_test, name, filename, i)
        scores_accuracy.append(accuracy_score(prediction, y_test))
        scores_precision.append(precision_score(prediction, y_test))
        scores_f1_score.append(f1_score(prediction, y_test))
        scores_recall.append(recall_score(prediction, y_test))
        i += 1

    reportForAllModels(
        name,
        num_attributes - 1,
        num_splits,
        scores_accuracy,
        scores_precision,
        scores_recall,
        scores_f1_score,
    )
```

Slika 33 – Podela podataka, pozivanje klasifikacije i formiranje rezultata

3.6 Treniranje i testiranje

Funkcija `train_and_classify_one_model()` služi da kreira klasifikator, trenira ga, i izvrši predikciju. Podaci se prvo skaliraju na nultu varijansu i srednju vrednost.



```
# treniranje i klasifikacija za 1 iteraciju 1 modela
def train_and_classify_one_model(classifier, train_index, test_index, X, y):
    X_train, X_test, y_train, y_test = (
        X.iloc[train_index, :],
        X.iloc[test_index, :],
        y.iloc[train_index],
        y.iloc[test_index],
    )
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    classifier.fit(X_train, y_train)
    prediction = classifier.predict(X_test)
    return prediction, y_test
```

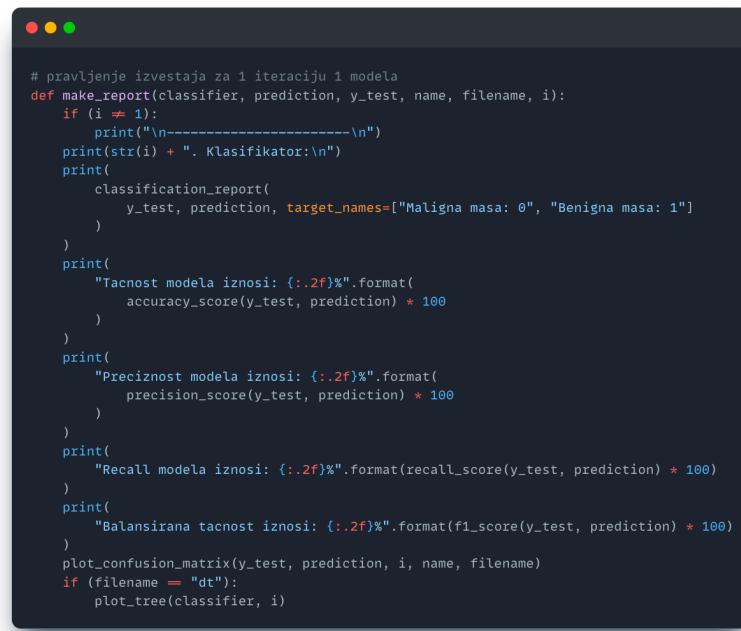
Slika 34 – Funkcija za treniranje i testiranje jednog modela

Pri izboru parametra za kreiranje stabla odluke, podešavani su prametri za dubinu stabla i za kriterijum pomoću kojeg se vrši izbor atributa za grananje stabla. Isprobavane su različite vrednosti za dubinu stabla (1 - 6) kao kriterijum (gini indeks, entropija), a najbolji rezultati dobijeni su za dubinu stabla 5 i kriterijum entropije.

3.7 Pravljenje izveštaja

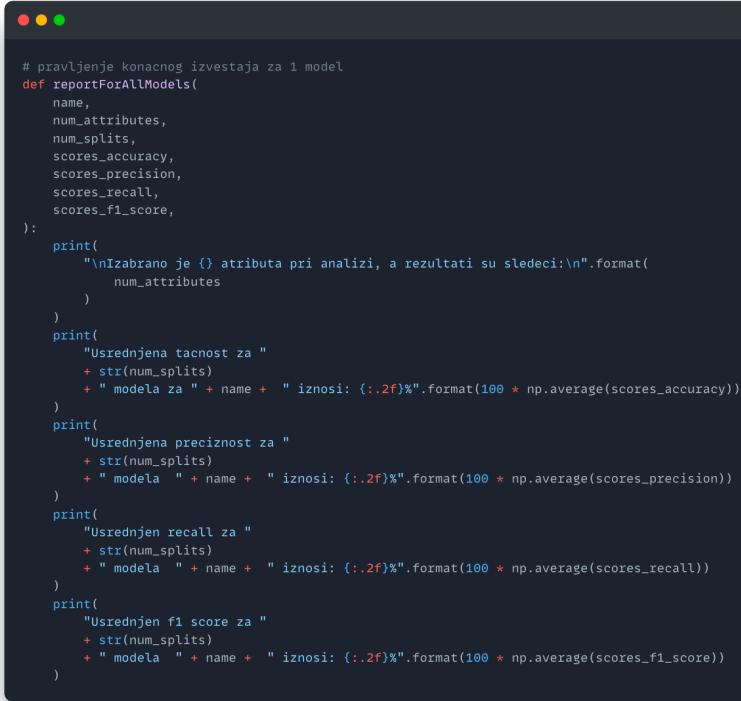
Nakon celokupne obrade potrebno je analizirati performanse klasifikacije stabla odluke. Rezultati su prikazani u vidu ispisa konzole, prikaza konfuzione matrice, kao i prikaza samog izgleda stabla odluke. Izveštaji daju uvid u to kako su izabrani atributi, parametri stabla odluke kao i sama podela skupa podataka na trenirajući i testirajući skup, rezultovali u klasifikaciji. Upravo iz tih rezultata dobijaju se nove ideje za menjanje pomenutih stvari.

Klasifikacija raka dojke pomoću mašinskog učenja



```
# pravljenje izvestaja za 1 iteraciju 1 modela
def make_report(classifier, prediction, y_test, name, filename, i):
    if (i != 1):
        print("\n-----\n")
    print(str(i) + ". Klasifikator:\n")
    print(
        classification_report(
            y_test, prediction, target_names=["Maligna masa: 0", "Benigna masa: 1"]
        )
    )
    print(
        "Tacnost modela iznosi: {:.2f}%".format(
            accuracy_score(y_test, prediction) * 100
        )
    )
    print(
        "Preciznost modela iznosi: {:.2f}%".format(
            precision_score(y_test, prediction) * 100
        )
    )
    print(
        "Recall modela iznosi: {:.2f}%".format(recall_score(y_test, prediction) * 100)
    )
    print(
        "Balansirana tacnost iznosi: {:.2f}%".format(f1_score(y_test, prediction) * 100)
    )
    plot_confusion_matrix(y_test, prediction, i, name, filename)
    if (filename == "dt"):
        plot_tree(classifier, i)
```

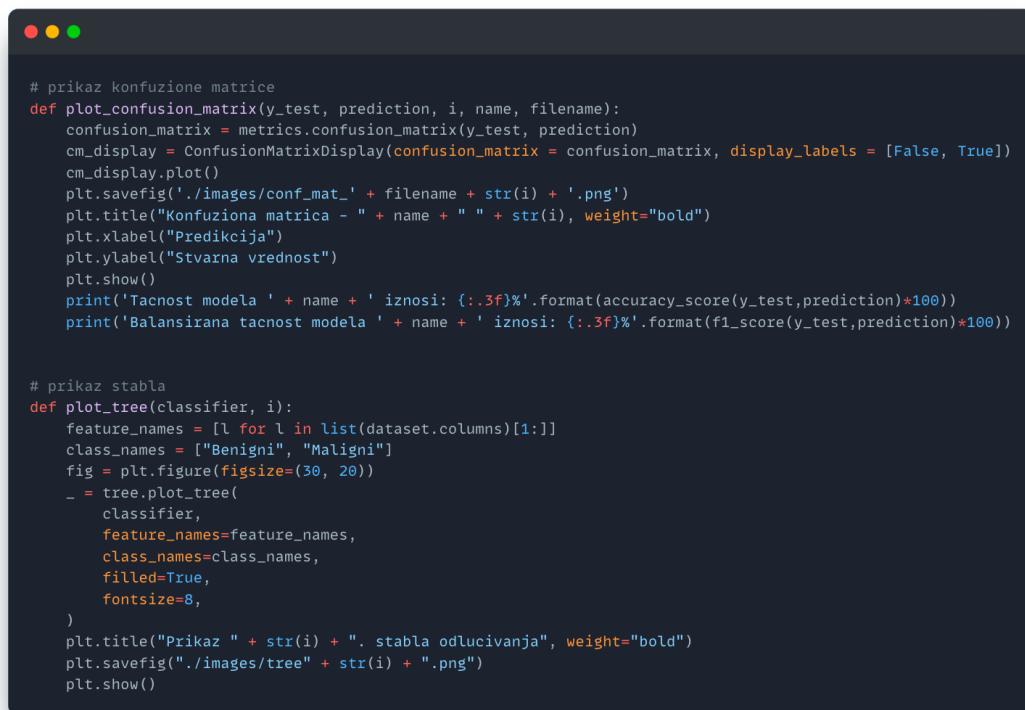
Slika 35 – Pravljenje izveštaja za 1 iteraciju 1 modela



```
# pravljenje konacnog izvestaja za 1 model
def reportForAllModels(
    name,
    num_attributes,
    num_splits,
    scores_accuracy,
    scores_precision,
    scores_recall,
    scores_f1_score,
):
    print(
        "\nIzabрано je {} atributa pri analizi, a rezultati su sledeći:\n".format(
            num_attributes
        )
    )
    print(
        "Usrednjena tacnost za "
        + str(num_splits)
        + " modela " + name + " iznosi: {:.2f}%".format(100 * np.average(scores_accuracy))
    )
    print(
        "Usrednjena preciznost za "
        + str(num_splits)
        + " modela " + name + " iznosi: {:.2f}%".format(100 * np.average(scores_precision))
    )
    print(
        "Usrednjeni recall za "
        + str(num_splits)
        + " modela " + name + " iznosi: {:.2f}%".format(100 * np.average(scores_recall))
    )
    print(
        "Usrednjeni f1 score za "
        + str(num_splits)
        + " modela " + name + " iznosi: {:.2f}%".format(100 * np.average(scores_f1_score))
    )
```

Slika 36 – Krajnji izveštaj koji daje usrednjenu vrednost za 1 model

Prikazom konfuzione matrice dobija se jasan uvid u to koje podatke je klasifikator dobro klasifikovao a koje nije. Posebno treba obratiti pažnju na to kako klasifikator klasificuje manje zastupljenu klasu.



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

# prikaz stabla
def plot_tree(classifier, i):
    feature_names = [l for l in list(dataset.columns)[1:]]
    class_names = ["Benigni", "Maligni"]
    fig = plt.figure(figsize=(30, 20))
    _ = tree.plot_tree(
        classifier,
        feature_names=feature_names,
        class_names=class_names,
        filled=True,
        fontsize=8,
    )
    plt.title("Prikaz " + str(i) + ". stabla odlucivanja", weight="bold")
    plt.savefig("./images/tree" + str(i) + ".png")
    plt.show()
```

Slika 37 – Funkcije za prikaz konfuzione matrice i stabla odluke

Prikazom stabla odluke dobija se vizualni prikaz kako je stablo odluke napravljeno. Listovi stabla predstavljaju konačnu odluku dok izabrani atributi vrše grananje pri čemu se najinformativniji atributi nalaze na vrhu stabla.

4 REZULTATI

4.1 Stablo odluke

Pri uključivanju svih atributa u treniranje i klasifikaciju i pri default-noj vrednosti za kriterijum grananja stabla (gini indeks), dobijaju se sledeći rezultati:

The image shows two side-by-side terminal windows. Both windows have a dark background with light-colored text. They display classification reports for decision trees with varying numbers of attributes (1, 4, and 5).

Left Terminal Window (1 attribute):

- 1. Stablo odlucivanja:**
- Confusion matrix:

	precision	recall	f1-score	support
Maligna masa: 0	0.96	0.90	0.93	71
Benigna masa: 1	0.85	0.93	0.89	43
- Metrics:

Tacnost modela iznosi: 91.23%
Preciznost modela iznosi: 85.11%
Recall modela iznosi: 93.02%
Balansirana tacnost iznosi: 88.89%
- 2. Stablo odlucivanja:**
- Confusion matrix:

	precision	recall	f1-score	support
Maligna masa: 0	0.93	0.93	0.93	71
Benigna masa: 1	0.88	0.88	0.88	43
- Metrics:

Tacnost modela iznosi: 91.23%
Preciznost modela iznosi: 88.37%
Recall modela iznosi: 88.37%
Balansirana tacnost iznosi: 88.37%
- 3. Stablo odlucivanja:**
- Confusion matrix:

	precision	recall	f1-score	support
Maligna masa: 0	0.93	0.96	0.95	72
Benigna masa: 1	0.93	0.88	0.90	42
- Metrics:

Tacnost modela iznosi: 92.98%
Preciznost modela iznosi: 92.50%
Recall modela iznosi: 88.10%
Balansirana tacnost iznosi: 90.24%

Right Terminal Window (4 attributes):

- 4. Stablo odlucivanja:**
- Confusion matrix:

	precision	recall	f1-score	support
Maligna masa: 0	0.93	0.96	0.95	72
Benigna masa: 1	0.93	0.88	0.90	42
- Metrics:

Tacnost modela iznosi: 92.98%
Preciznost modela iznosi: 92.50%
Recall modela iznosi: 88.10%
Balansirana tacnost iznosi: 90.24%
- 5. Stablo odlucivanja:**
- Confusion matrix:

	precision	recall	f1-score	support
Maligna masa: 0	0.98	0.85	0.91	71
Benigna masa: 1	0.79	0.98	0.87	42
- Metrics:

Tacnost modela iznosi: 89.38%
Preciznost modela iznosi: 78.85%
Recall modela iznosi: 97.62%
Balansirana tacnost iznosi: 87.23%
- Izabrano je 30 atributa pri analizi, a rezultati su sledeći:
- Summary metrics:

Usrednjena tacnost za 5 modela iznosi: 91.56%
Usrednjena preciznost za 5 modela iznosi: 91.04%
Usrednjeni recall za 5 modela iznosi: 87.46%
Usrednjeni f1 score za 5 modela iznosi: 89.00%

Slika 38 – Rezultati klasifikatora bez bilo kojih podešavanja

Kako su klase nebalansirane, sama tačnost klasifikacije nije toliko bitna koliko su preciznost, recall i f1 skor.

Najbolji rezultati dobijeni su sledećim izborom:

- Izabrano je 6 atributa – radius_worst, concave points_worst, concavity_mean, symmetry_worst, texture_worst
- Izabrani su sledeći parametri stabla odluke – maksimalna dubina stabla (5), kriterijum grananja (entropija)

1. Stablo odlucivanja:

	precision	recall	f1-score	support
Maligna masa: 0	0.96	0.97	0.97	71
Benigna masa: 1	0.95	0.93	0.94	43
accuracy			0.96	114
macro avg	0.96	0.95	0.95	114
weighted avg	0.96	0.96	0.96	114

Tacnost modela iznosi: 95.61%
Preciznost modela iznosi: 95.24%
Recall modela iznosi: 93.02%
Balansirana tacnost iznosi: 94.12%

2. Stablo odlucivanja:

	precision	recall	f1-score	support
Maligna masa: 0	0.93	0.99	0.96	71
Benigna masa: 1	0.97	0.88	0.93	43
accuracy			0.95	114
macro avg	0.95	0.93	0.94	114
weighted avg	0.95	0.95	0.95	114

Tacnost modela iznosi: 94.74%
Preciznost modela iznosi: 97.44%
Recall modela iznosi: 88.37%
Balansirana tacnost iznosi: 92.68%

3. Stablo odlucivanja:

	precision	recall	f1-score	support
Maligna masa: 0	0.97	0.97	0.97	72
Benigna masa: 1	0.95	0.95	0.95	42
accuracy			0.96	114
macro avg	0.96	0.96	0.96	114
weighted avg	0.96	0.96	0.96	114

Tacnost modela iznosi: 96.49%
Preciznost modela iznosi: 95.24%
Recall modela iznosi: 95.24%
Balansirana tacnost iznosi: 95.24%

4. Stablo odlucivanja:

	precision	recall	f1-score	support
Maligna masa: 0	0.93	0.97	0.95	72
Benigna masa: 1	0.95	0.88	0.91	42
accuracy			0.94	114
macro avg	0.94	0.93	0.93	114
weighted avg	0.94	0.94	0.94	114

Tacnost modela iznosi: 93.86%
Preciznost modela iznosi: 94.87%
Recall modela iznosi: 88.10%
Balansirana tacnost iznosi: 91.36%

5. Stablo odlucivanja:

	precision	recall	f1-score	support
Maligna masa: 0	1.00	0.99	0.99	71
Benigna masa: 1	0.98	1.00	0.99	42
accuracy			0.99	113
macro avg	0.99	0.99	0.99	113
weighted avg	0.99	0.99	0.99	113

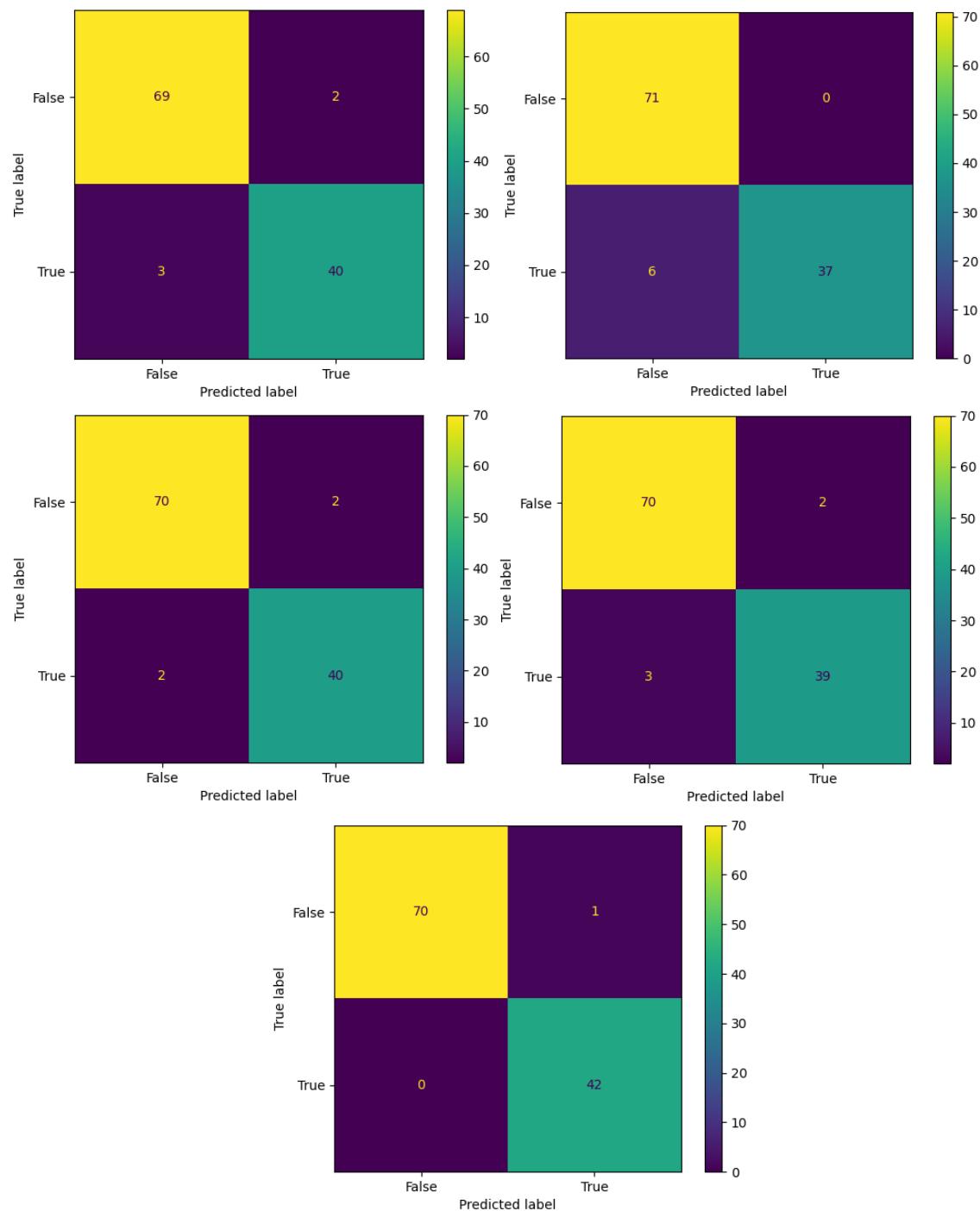
Tacnost modela iznosi: 99.12%
Preciznost modela iznosi: 97.67%
Recall modela iznosi: 100.00%
Balansirana tacnost iznosi: 98.82%

Izabrano je 6 atributa pri analizi, a rezultati su sledeći:

Usrednjena tacnost za 5 modela iznosi: 95.96%
Usrednjena preciznost za 5 modela iznosi: 92.95%
Usrednjeni recall za 5 modela iznosi: 96.09%
Usrednjeni f1 score za 5 modela iznosi: 94.44%

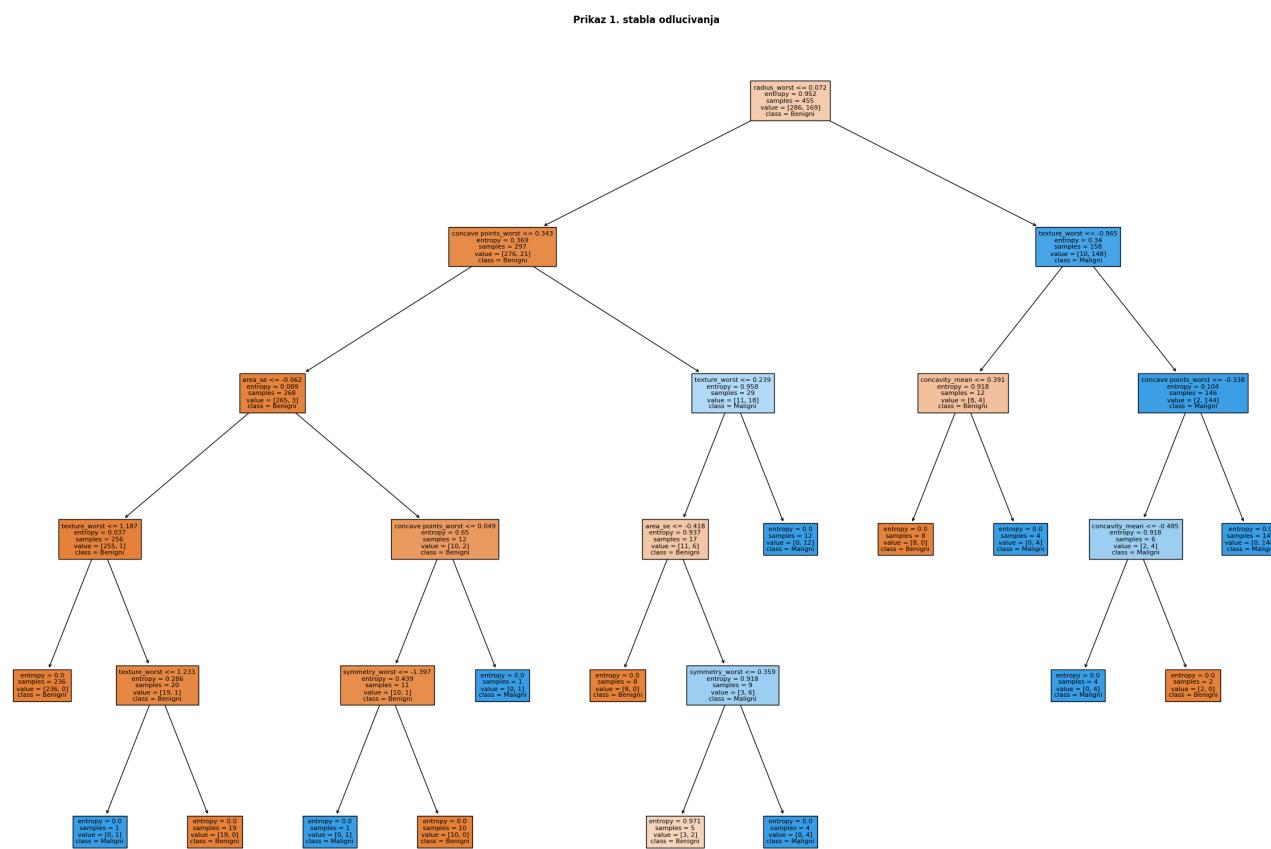
Slika 39 – Rezultati za najbolje izabране parametre

Na sledećim slikama date su konfuzione matrice za prikazana stabla:



Slika 40 – Rezultati za najbolje izabrane parametre

Pri svakom drugom izboru atributa i parametara stabla odlučivanja dobijeni su lošiji rezultati. Eksperimentalnim putem i zaključcima iz **sekcije 3.3**, primećeno je kako postoji veliki broj koreliranih grupa atributa, a najbolji rezultati su dobijeni kad je izabran samo po 1 najbolji predstavnik iz svake grupe čime je značajno smanjen broj atributa koji učestvuje u klasifikaciji. Dosta pažnje je posvećeno na to koliko je svaki atribut informativan i dobar u razdvajaju klasa primenom informacione dobiti, fišerovog skora i rekurzivne eliminacije atributa. Iako se u početnoj bazi nalazilo 30 atributa, najbolji rezultati dobijeni su sa 6 atributa. Pri izboru većeg broja atributa (na primer 15), postoji dosta sličnih atributa koji su jako korelirani i time sem što je uvedena nepotrebna kompleksnost i dimenzionalnost stabla odluke, nego su i smanjene same performanse.



Slika 41 – Primer izgleda stabla

4.2 Slučajne šume

4.3 AdaBoost

4.4 XGBoost

4.5 Logistička regresija

4.6 Naivni Bajes

4.7 Poređenje svih modela

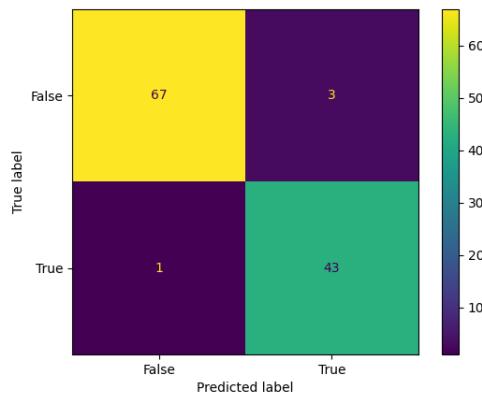
Nakon završetka celog procesa moguće je uporediti rezultate klasifikatora:

```
-----  
Stablo odlucivanja:  
precision recall f1-score support  
Maligna masa: 0 0.99 0.96 0.97 78  
Benigna masa: 1 0.93 0.98 0.96 44  
  
accuracy 0.96  
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 78  
Benigna masa: 1 1.00 0.93 0.96 44  
  
accuracy 0.97  
macro avg 0.98 0.97 0.97 114  
weighted avg 0.97 0.97 0.97 114  
  
Tacnost modela logistickie regresije iznosi: 97.368%  
Balansirana tacnost modela logistickie regresije iznosi: 96.471%  
  
-----  
Slucajna suma:  
precision recall f1-score support  
Maligna masa: 0 0.96 1.00 0.98 78  
Benigna masa: 1 1.00 0.93 0.96 44  
  
accuracy 0.97  
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 78  
Benigna masa: 1 1.00 0.98 0.99 44  
  
accuracy 0.99  
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 78  
Benigna masa: 1 0.98 0.93 0.95 44  
  
accuracy 0.96  
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 78  
Benigna masa: 1 0.95 0.89 0.92 44  
  
accuracy 0.94  
macro avg 0.94 0.93 0.93 114  
weighted avg 0.94 0.94 0.94 114  
  
Tacnost modela Naivnog Bajesa iznosi: 93.860%  
Balansirana tacnost modela Naivnog Bajesa iznosi: 91.765%  
-----
```

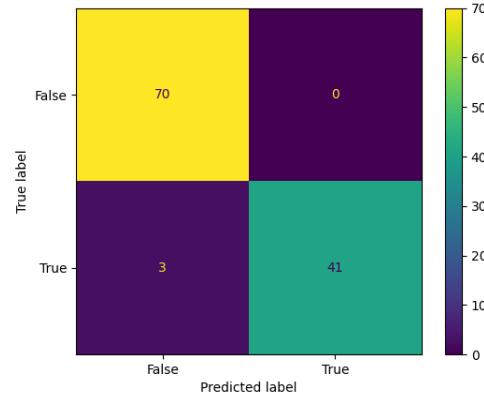
Slika 35 – Poređenja različitih tehnika klasifikacije

Klasifikacija raka dojke pomoću mašinskog učenja

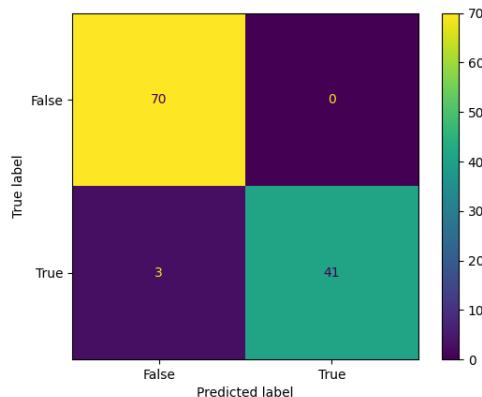
Prikaz konfuzionih matrica:



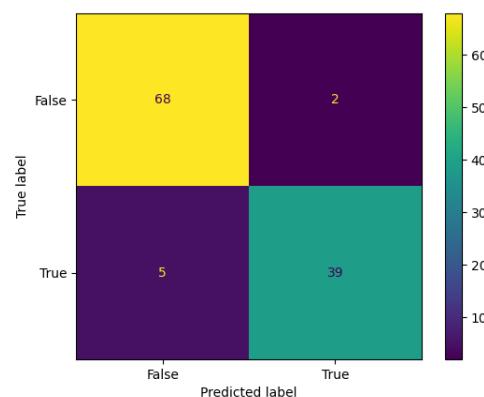
Slika 36 – KM stabla odluke



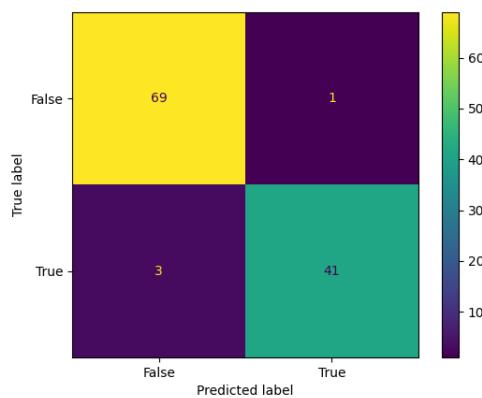
Slika 37 – KM logističke regresije



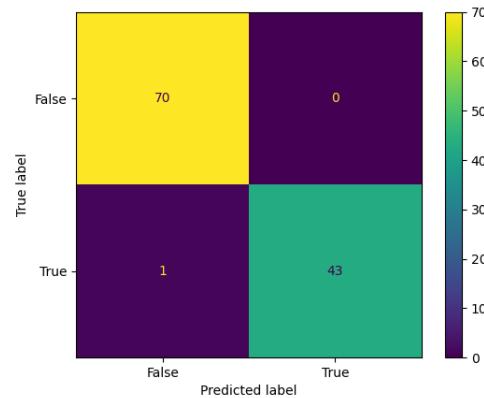
Slika 38 – KM slučajne šume



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%.

5 ZAKLJUČAK

Izbor atributa za klasifikaciju je od ključnog značaja, kao i podešavanje parametara samog stabla odluke. Veći broj atributa može dati bolje rezultate ali i podsticati klasifikator da preobuči podatke i tako loše generalizuje neke nove. Manjim brojem atributa, osim same generalizacije, postiže se i izbegavanje kompleksnosti klasifikatora koja može biti veoma zahtevna. Treba naći kompromis između toga. Podešavanja parametara klasifikatora takođe ima krucijalan značaj u konačnim performansama. Dubina stabla takođe poboljšava klasifikaciju, ali i dozvoljava preobučavanje. Potrebno je odrediti parametar za dubinu stabla kako bi potkresivanjem stabla naterati model da ne preobučava podatke. Baza podataka je nebalansirana i jako je bitno voditi računa o podeli podataka na trening i test skup, kako ne bi bili zapostavljeni uzorci manje zastupnjene klase. Takođe, sam odnos obe klase treba biti jednak i u trening i u test skupu. Velikim brojem pokretanja programa zaključuje se da rezultati za iste parametre u velikoj meri variraju, te je bila potreba da se implementira kros-validacija koja će dati dosta realniji rezultat performansi stabla odluke.

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 mogле bi drastično da promene kvalitet života i da se pomoću njih na vreme detektuju neželjene bolesti.

6 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>