Usporedba metoda za klasifikaciju podataka temeljem skupa podataka koji opisuje simptome i faktore za srčana oboljenja

Gašparić, Matej

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:578793

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Matej Gašparić

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

Usporedba metoda za klasifikaciju podataka temeljem skupa podataka koji opisuje simptome i faktore za srčana oboljenja

Mentor: Student:

Izv. prof. dr. sc. Tomislav Stipančić, dipl. ing. Matej Gašparić

Zagreb, 2024.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se svom mentoru izv. prof. dr. sc. Tomislavu Stipančiću koji mi je pomogao osmisliti ideju i pružio pomoć te savjete za izradu ovog rada.

Također, zahvaljujem se svojoj obitelji, kolegama i prijateljima na svakom obliku podrške tijekom ovog studija.

Matej Gašparić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 24 – 06 / 1**	
Ur.broj: **15 – 24 –**	

ZAVRŠNI ZADATAK

Student: **Matej Gašparić** JMBAG: **0035226757**

Naslov rada na hrvatskom jeziku: **Usporedba metoda za klasifikaciju podataka temeljem skupa podataka koji opisuje simptome i faktore za srčana oboljenja**

Naslov rada na engleskom jeziku: **A comparison of data classification methods based on a dataset describing symptoms and factors for heart disease**

Opis zadatka:

U svijetu podataka i velike količine informacija algoritmi umjetne inteligencije postaju sve značajniji. U ovisnosti o vrsti i strukturi podataka, metode strojnog učenja koriste se prilikom predviđanja, odlučivanja, klasifikacije, prepoznavanja i slično.
Koristeći nekoliko metoda za klasifikaciju primijenjenih na skupu podataka koji opisuje simptome i faktore značajne kod pojave srčanih oboljenja potrebno je analizirati mogućnosti predviđanja odabranih metoda te usporediti rezultate.
U sklopu rada potrebno je:

- pripremiti te analizirati podatke o simptomima i faktorima srčanih oboljenja da budu prikladni za korištenje od strane korištenih metoda
- primijeniti barem pet metode za klasifikaciju na zadanom skupu podataka (npr. logistic regression, random forest, SVM, decision trees, te random forest)
- načiniti analizu te usporediti rezultate primjene pojedinog klasifikatora.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:

30. 11. 2023.

Zadatak zadao:

Izv. prof. dr. sc. Tomislav Stipančić

Datum predaje rada:

1. rok: 22. i 23. 2. 2024.
2. rok (izvanredni): 11. 7. 2024.
3. rok: 19. i 20. 9. 2024.

Predviđeni datumi obrane:

1. rok: 26. 2. – 1. 3. 2024.
2. rok (izvanredni): 15. 7. 2024.
3. rok: 23. 9. – 27. 9. 2024.

Predsjednik Povjerenstva:

Prof. dr. sc. Damir Godec

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Opis
w_i	Težine
x_i	Ulazna karakteristika
b	Vrijednost biasa
$\sigma(z)$	Sigmoidna funkcija
P	Vjerojatnost
H	Entropija
p_i	Set vjerojatnosti
S	Skup karakteristika
F	Moguća karakteristika iz skupa
S_f	Broj članova skupa *S* koji imaju vrijednost *f* za karakteristiku *F*
$\mathbf{\Phi}$	Vektor ulaznih elemenata
$\Phi(x_i)$	Funkcija ulaza x_i
O	Komputacijska složenost
$\boldsymbol{K}$	Gramova matrica
r	Pearsonov koeficijent korelacije
X	Originalna vrijednost karakteristike
X_{max}	Maksimalna vrijednost karakteristike
X_{min}	Minimalna vrijednost karakteristike
TP	Stvarno pozitivni primjeri
FP	Lažno pozitivni primjeri
FN	Lažno negativni primjeri
TN	Stvarno negativni primjeri
TPR	Stopa stvarnih pozitiva
FPR	Stopa lažnog alarma
X_test	Primjeri za testiranje
y_test	Klase primjera za testiranje

SAŽETAK

U svijetu podataka i velike količine informacija, algoritmi umjetne inteligencije postaju sve značajniji. U ovisnosti o vrsti i strukturi podataka, metode strojnog učenja koriste se prilikom predviđanja, odlučivanja, klasifikacije, prepoznavanja i slično. U ovom radu koristit će se nekoliko metoda za klasifikaciju koje će biti primijenjene na skupu podataka koji opisuje simptome i faktore značajne kod pojave srčanih oboljenja. Korištene metode su: logistička regresija, stablo odlučivanja, nasumična šuma, metoda potpornih vektora, k-najbližih susjeda od kojih je svaka ukratko objašnjena prije obrade podataka i usporedbe rezultata. Prikazana je priprema podataka i njihova vizualizacija te na kraju implementacija algoritama i usporedba njihovih rezultata prema mjerama vrednovanja modela strojnog učenja.

Ključne riječi: algoritam, strojno učenje, logistička regresija, stablo odlučivanja, nasumična šuma, metoda potpornih vektora, k-najbližih susjeda, umjetna inteligencija, srčane bolesti

SUMMARY

In the world of data and a vast amount of information, artificial intelligence algorithms are becoming increasingly significant. Depending on the type and structure of data, machine learning methods are used for prediction, decision-making, classification, recognition, and similar tasks. In this paper, several classification methods will be utilized and applied to a dataset describing symptoms and factors relevant to the occurrence of heart diseases. The methods employed include logistic regression, decision tree, random forest, support vector machine, and k-nearest neighbors, each briefly explained before data processing and result comparison. Data preparation and visualization are presented, followed by the implementation of algorithms and a comparison of their results based on machine learning model evaluation metrics.

Key words: algorithm, machine learning, logistic regression, decision tree, random forest, support vector machine, k-nearest neighbors, artificial intelligence, heart diseases

1. UVOD

Kardiovaskularne bolesti vodeći su uzrok smrti globalno koje svake godine odnose oko 17.9 milijuna života. Više od 80 % slučajeva kardiovaskularnih bolesti uzrokovano je srčanim i moždanim udarom, a trećina smrti dogodi se osobama mlađim od 70 godina [1]. Predviđanje pojave srčanih bolesti ključni je aspekt preventivne zaštite, koji omogućuje pravovremenu intervenciju i personalizirane strategije liječenja. Posljednjih godina integracija tehnika strojnog učenja (engl. *machine learning*) u zdravstvu pokazala je obećavajuće rezultate, a posebno u domeni predviđanja bolesti srca. Kao jedan dio umjetne inteligencije, strojno učenje bavi se razvojem i istraživanjem algoritama koji uče iz podataka da bi dalje mogli obavljati zadatke bez eksplicitnih uputa. U okviru ovog rada, prikazane su različite metode klasifikacije na temelju podataka koji ukazuju na simptome i faktore kod srčanih oboljenja. Uspoređeno je petero metoda, odnosno algoritama: logistička regresija, stablo odlučivanja, nasumična šuma, metoda potpornih vektora i k-najbližih susjeda. Dalje u nastavku bit će objašnjene osnovne podjele strojnog učenja, nadzirano i nenadzirano učenje te principi rada iza svake od metoda. Nakon toga, nekoliko poglavlja posvećeno je pripremi i vizualizaciji podataka gdje su provedene univarijatna i bivarijatna analiza podataka s pripadajućim metodama vizualizacije. Neke od korištenih vizualizacija su histogrami, kutijasti dijagrami, dijagrami raspršenja, matrica korelacija od kojih je za svaku prikazan postupak izrade u korištenom programskom jeziku, *Pythonu*. Na kraju, provedeno je treniranje modela da bi se njihov rad i točnost mogli usporediti, a iz te usporedbe dalje su odabrana tri najbolja algoritma koji će se detaljnije usporediti. Za tu daljnju usporedbu korištene su matrice zabune i ROC krivulje kao mjere vrednovanja modela.

2. STROJNO UČENJE

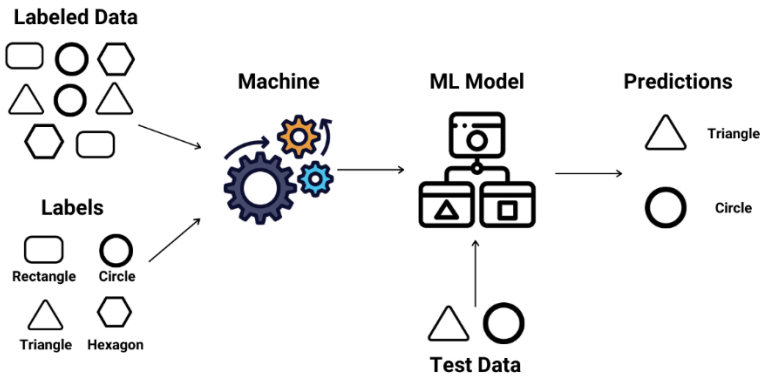
Algoritmi strojnog učenja pokazuju značajan potencijal u rješavanju složenih problema. Svakodnevne primjene poput pretraživanja interneta i prepoznavanja govora primjeri su značajnih napretka ostvarenih u ispunjavanja potencijala strojnog učenja. Takvi algoritmi mogu rješavati mnoge od takvih zahtjevnih problema na općenit način, odnosno ne zahtijevaju eksplicitni detaljni dizajn. Umjesto toga, usvajaju detaljni dizajn iz skupa označenih podataka, tj. skupa primjera koji ilustriraju ponašanje programa. Drugim riječima, algoritmi strojnog učenja uče iz podataka. Čim je skup podataka veći, algoritmi postaju točniji. Cilj strojnog učenja je naučiti model pravila iz označenog skupa podataka kako bi ispravno predvidjeli oznake podataka (npr. slike) koje nisu dio tog skupa. Sporiji napredak u istraživanju strojnog učenja trajao je do sredine 2000-tih godina, nakon čega se napredak u području naglo ubrzao. Veća računalna snaga i memorija, dostupnost velike količine podataka pohranjenih na internetu i unaprijeđeni algoritmi koji su optimizirani za velike količine podataka su glavni čimbenici koji su omogućili takav ubrzani razvoj ovog područja. Samim time, područje je postalo dostupnije općoj populaciji za razliku od ranijih faza razvoja gdje su pristup imali samo najbolji znanstvenici. Tipovi problema koje strojno učenje rješava su prepoznavanje i pretvaranje govora u tekst, prevođenje jezika te autonomna vožnja vozila kao jedna od aktualnih primjena u razvoju [2]. Iako je bitno koji tip algoritma primijeniti na određeni problem, najvažnije je odabrati dobre podatke iz kojih će algoritam učiti. Postoje razni problemi koji se mogu pojaviti s podacima poput nedostataka, loše kvalitete podataka, netočnosti, nepovezanosti, ponavljajućih vrijednosti i slično. Što se tiče daljnje podjele, na temelju korištenih metoda i načinu učenja, strojno učenje podijeljeno je na nadzirano učenje, nenadzirano učenje, polunadzirano učenje i pojačano učenje[3].

2.1. Podjela strojnog učenja

2.1.1. Nadzirano učenje

Osnovni proces učenja u jednostavnom modelu strojnog učenja podijeljen je u dva koraka: treniranje i testiranje. Tijekom treniranja, model uči uzorke, tj. značajke iz ulaznih podataka unutar podataka za treniranje i time se stvara model učenja. Tijekom testiranja, model radi predviđanja za testne podatke ili za stvarne podatke. Izlaz ovog procesa su označeni podaci, koji pružaju završno predviđanje ili klasificirane podatke. Označeni podaci podrazumijevaju skup podataka u kojima su podaci podijeljeni, tj. kojima je dodijeljena oznaka na temelju nekih karakteristika [4]. Npr. ako treniramo model da prepoznaje je li na slikama pas ili mačka,

potrebno je imati set podataka u kojima će kod svake pojedine slike biti označeno je li na slici pas ili mačka. Tok procesa nadziranog učenja prikazan je na Slici 1:



Slika 1. Shema nadziranog učenja [5]

Nadzirano učenje najčešća je tehnika u rješavanju klasifikacijskih problema, budući da je glavni cilj naučiti stroj da dalje primjenjuje klasifikacijski sustav koji smo stvorili. U ovom pristupu, stroj dobiva set podataka, npr. tisuće različitih slika. Za svaku sliku dobiva točan odgovor (što je prikazano na slici), tj. dodaje se oznaka slici. Stroj tada „uči“ karakteristike između tih slika da može na kraju dati pravilan odgovor. Nadzirano učenje dijeli se u dvije kategorije, klasifikaciju i regresiju [2].

2.1.1.1. Klasifikacija

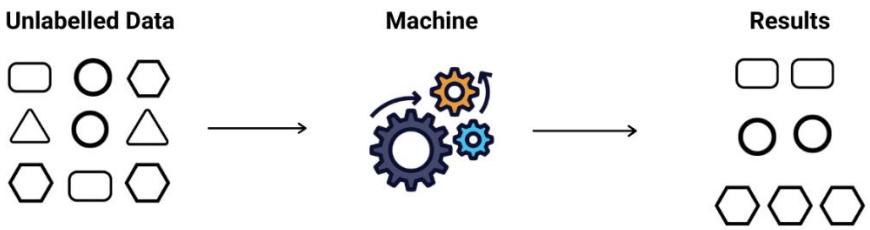
Pojam klasifikacije pojavljuje se u raznim tipovima ljudskog rada. U općenitom smislu taj pojam označava svaku situaciju gdje se na temelju postojećih informacija donosi odluka ili predviđanje, a proces klasifikacije je tada metoda kojom se do takvih istih rezultata dolazi u različitim problemima [6] Procesi u kojima se organizira pošta pomoću skeniranih brojeva, procjena rizika za izdavanje kredita na temelju osobnih i bankovnih podataka te dijagnoza pacijentove bolesti i planiranje liječenja samo su neki od primjera u kojima je klasifikacija kao metoda vrlo zastupljena [6]. Najbolji primjer klasifikacije je prepoznavanje *spam* elektroničke pošte. To je slučaj binarne klasifikacije, budući da se određuje je li dobivena poruka *spam* ili nije, tj. postoje samo dvije klase poruke. Algoritam učenja, u ovom slučaju klasifikator, uči iz podataka za treniranje kako bi utvrdio veze između ulaznih varijabli i ostalih klasa. Klasifikator se oslanja na poznate uzorke *spam* poruke i poruke koja nije *spam.* Nakon dovoljno treniranja, klasifikator može uspješno prepoznati nepoznate, tj. *spam* poruke [7].

2.1.1.2. Regresija

Regresijski algoritmi pokušavaju utvrditi vezu između ulaznih i izlaznih varijabli tako da stvaraju matematički model koji se podudara s podacima koje dajemo algoritmu. Cilj ovog postupka je otkriti matematičku vezu između ulaznih karakteristika i ciljane varijable, a zatim time omogućiti precizno predviđanje algoritma na novim i nepoznatim podacima. Budući da je regresija statistička metoda, koristi se za razne primjene poput predviđanja, testiranja hipoteza, kontrole varijabli i prognoziranja na temelju ranijih podataka [8].

2.1.2. Nenadzirano učenje

Za razliku od nadziranih algoritama, nenadzirani algoritmi strojnog učenja ne zahtijevaju označene podatke za uspješno treniranje što ih čini pogodnima za otkrivanje uzoraka, skupova i razlika u nepoznatom skupu podataka. Drugim riječima, takvi algoritmi samostalno pronalaze uzorke u podacima. Uspješno obavljaju zadatke kao što su istraživačka analiza podataka, segmentacija kupaca i prepoznavanje slika. *Chatbotovi*, autonomna vozila, sustavi prepoznavanja lica neki su od najraširenijih sustava koji primjenjuju ove algoritme. Prednosti ovakvih algoritama su manjak potrebe za prethodnim označivanjem podataka, mogućnost rada sa zahtjevnijim setovima podataka i financijska pristupačnost, budući da nema potrebe za dodatnim radom u obliku označavanja i pripreme podataka za treniranje algoritma. S druge strane, vrijeme treniranja je duže nego kod nadziranog učenja budući da algoritmi nenadziranog učenja traže znatno veće setove podataka za učenje. Također, teže im je odrediti točnost jer ne postoje označeni podaci s kojima bi se potvrdili rezultati [9]. Pojednostavljeni tok procesa nenadziranog učenja vidljiv je na Slici 2:

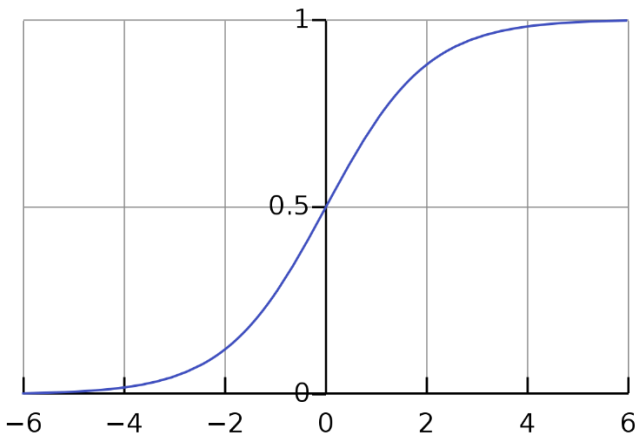


Slika 2. Shema nenadziranog učenja[5]

2.2. Klasifikacijski algoritmi

2.2.1. Logistička regresija

Algoritmi logističke regresije procjenjuju kako su različite varijable u međusobnom odnosu i razvrstavaju podatke u određene klase. Često primjenjivane u predvidljivom modeliranju, takve metode računaju vjerojatnost da neki podatak pripada određenoj kategoriji tako da procjenjuju matematičku vjerojatnost. Za primjer, u slučaju ovog rada gdje se procjenjuje na temelju dostupnih varijabli hoće li osoba imati srčano oboljenje postoje dva odgovora, da ili ne. Za upravo takve binarne klasifikacije se najčešće rabi logistička regresija, gdje je izlazna varijabla jedna od dvije kategorije. Mapiranje predviđanja i njihovih vjerojatnosti ostvaruje se pomoću sigmoidne funkcije, logističke funkcije u obliku slova S koja transformira bilo koju realnu vrijednost u raspon između 0 i 1[10].



Slika 3. Graf sigmoidne funkcije[11]

Opisana formulom (1) , sigmoidna funkcija je aktivacijska funkcija za logističku regresiju i definirana je kao:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

Gdje su:

e – baza prirodnog logaritma,

x – brojčana vrijednost koja se želi tranfsormirati

U slučaju jednog ulaza, x je prikazan kao vektor karakteristika $[x_1, x_2, \dots, x_n]$. Izlaz y ovakve vrste klasifikatora može biti 1 što znači da trenutno opažanje pripada klasi ili da ne pripada klasi odnosno izlaz je jednak 0. Logistička regresija to rješava učeći vektor težina i *biasa* iz seta podataka za treniranje. *Bias* označava grešku koja je posljedica aproksimiranja stvarnog problema koji je često previše složen s jednostavnijim modelom. Svaka težina w_i realan je broj i povezana je s jednom od ulaznih karakteristika x_i. Težina w_i klasifikacijskoj odluci predstavlja koliko je bitna određena ulazna karakteristika. Ako je karakteristika dio pozitivne klase, odnosno izlaz y je 1, tada je i težina pozitivna. Suprotno, ako je karakteristika dio negativne klase i težina je negativna. Za primjer bi se moglo reći da riječ „izvanredno“ ima visoko pozitivnu težinu dok riječ „užasno“ ima vrlo negativnu težinu. Da bi došao do odluke na podacima za testiranje, klasifikator množi svaku karakteristiku x_i s težinom w_i, zatim zbroji sve te pomnožene karakteristike i na kraju dodaje tome b vrijednost *biasa* [12]. Rezultat je broj z koji prikazuje težinski zbroj dokaza za klasu:

$$z = \left(\sum_{i=1}^{n} w_i\, x_i\right) + b \tag{2}$$

Budući da se izrazom (2) dobiva realan broj koji nije unutar raspona od 0 do 1, broj z je potrebno uvrstiti u sigmoidnu funkciju (1). Time ćemo dobiti jednadžbu iz koje se dobiva vjerojatnost:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3}$$

Iz jednadžbe (3) dobivamo vrijednost između 0 i 1. Da bi to bila vjerojatnost, potrebno je zapisati da suma slučajeva $P(y = 1)$ i $P(y = 0)$ iznosi 1. Također, jednadžbu (2) možemo drukčije zapisati tako da umjesto sume zapišemo skalarni produkt vektora w_i i x_i[12]:

$$z = \boldsymbol{w} \cdot \boldsymbol{x} + b \tag{4}$$

Dalje se može zapisati:

$$P(y = 1) = \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

$$= \frac{1}{1 + e^{-(\boldsymbol{w} \cdot \boldsymbol{x} + b)}}$$

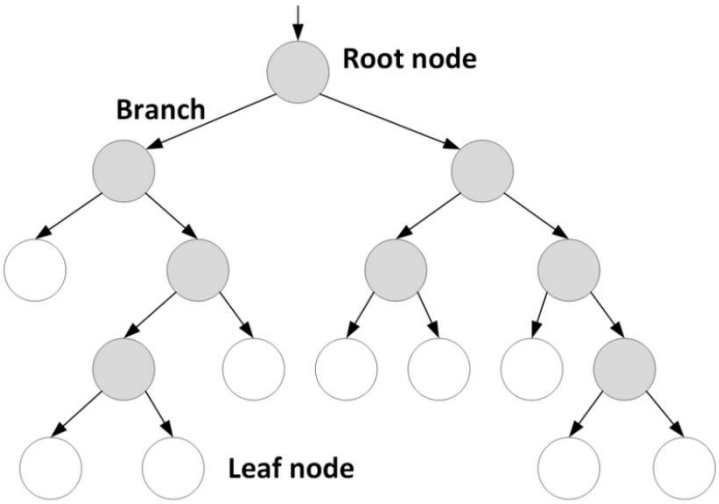
$$P(y = 0) = 1 - \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

$$= 1 - \frac{1}{1 + e^{-(\boldsymbol{w} \cdot \boldsymbol{x} + b)}}$$

$$= \frac{e^{-(\boldsymbol{w} \cdot \boldsymbol{x} + b)}}{1 + e^{-(\boldsymbol{w} \cdot \boldsymbol{x} + b)}} \tag{5}$$

2.2.2. Stabla odlučivanja

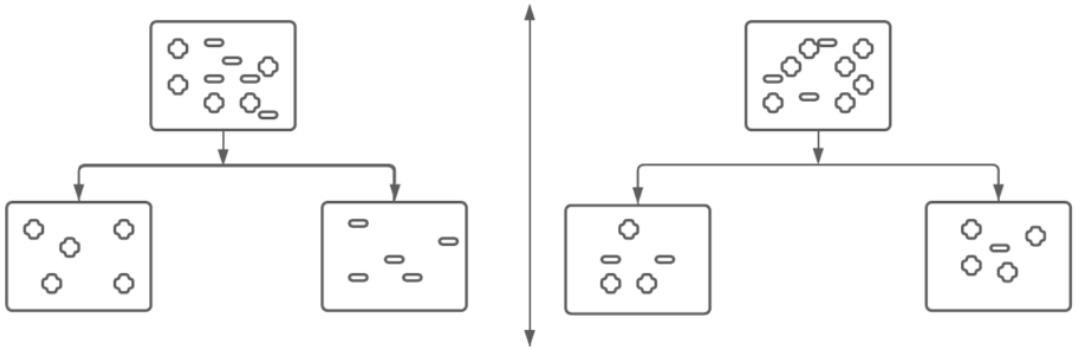
Mapa ili shema mogućih ishoda za seriju nekih povezanih izbora zove se stablo odlučivanja (eng. *decision tree*). Omogućuje pojedincima ili organizacijama da odvagnu moguće odluke na temelju cijena, vjerojatnosti i pogodnosti. Stablo odlučivanja počinje iz samo jednog čvora koji se dalje grana na više čvorova i naposljetku do kraja stabla, odnosno lišća. Da bi se lakše razvrstali podaci, čvor postavlja pitanje, a različite grane predstavljaju mogućnosti do kojih taj čvor može voditi[13].



Slika 4. Stablo odlučivanja [14]

Svaki čvor stabla predstavlja testni slučaj za neku karakteristiku, a svako grananje čvoru pruža najbolje odgovore za testni slučaj. Taj proces se ponavlja za svako podstablo koje izlazi iz sljedećeg čvora. Glavni izazov u stvaranju stabla odlučivanja je prepoznati koje karakteristike mogu biti izabrane kao početni čvorovi na različitim razinama zbog čega postoje različite mjere odabira karakteristika koje biraju početne čvorove stabla. Stvaranje podčvorova upotrebljava više algoritama, a svaki s ciljem da stablo prvo podijeli čvorove po svim varijablama i nakon toga bira grananje koje omogućuje nastanak homogenih podčvorova. Homogeni podčvorovi sadrže samo podatke koji pripadaju istoj klasi, tj. atributu.

Na Slici 5 lijevo prikazani su homogeni podčvorovi koji sadrže podatke iste klase, a desno heterogeni podčvorovi [15]



Slika 5. Prikaz homogenih i heterogenih podčvorova [16]

Budući da se početni čvor ne može samo nasumično odabrati, uvedeni su kriteriji entropije i informacijske dobiti. Ti kriteriji računaju vrijednosti za svaki atribut, a zatim su atributi po tim vrijednostima poredani u stablu. Atribut s najvišom vrijednošću stavljen je kao početni čvor [15].

2.2.2.1. Entropija

U ovom kontekstu, entropija je matematički pojam koji opisuje mjeru nasumičnosti različitih klasa u podacima, odnosno izražava razinu „nečistoća“ u setu karakteristika. Entropija H za set vjerojatnosti p_i iznosi:

$$Entropija(p) = -\sum_i p_i log_2 p_i \tag{6}$$

Koristi se baza logaritma 2 zato što je zamišljeno da se sve enkodira binarnim znamenkama (bitovima). Ako neka karakteristika dijeli uzorak podataka na 50 % pozitivnih i 50 % negativnih tada je iznos entropije najveći. Suprotno, ako bi iznos entropije bio jednak nuli to bi značilo da je taj podčvor potpuno homogen, odnosno sadrži atribute samo jednog tipa [17]Za primjer se može uzeti odluka hoćemo li gledati film.

Tablica 1. Mogući izbori kod grananja

Gledati film?	
Da	**Ne**
9	6

Koristeći formulu, možemo zapisati:

$$Entropija(Gledati\ film) = Entropija(9{,}6)$$

$$= Entropija\left(\frac{9}{15}, \frac{6}{15}\right)$$

$$= -\left(0.6log_2(0.6)\right) - \left(0.4log_2(0.4)\right)$$

$$= 0.97$$

2.2.2.2. Informacijska dobit

Sljedeći korak je dokučiti koliko bi se entropija cijelog seta za treniranje, odnosno početnog čvora u stablu smanjila ako odaberemo svaku od karakteristika seta za sljedeće čvorove. Ako od entropije cijelog seta oduzmemo entropiju koja nastaje kad odaberemo neku od karakteristika dobivena je informacijska dobit tog odabira:

$$Gain(S, F) = Entropy(S) - \sum_{f \in values(F)} \frac{|S_f|}{|S|} Entropy(S_f) \qquad (7)$$

Gdje su:

S – set karakteristika

F – moguća karakteristika iz seta

$|S_f|$ – broj članova seta S koji imaju vrijednost f za karakteristiku F

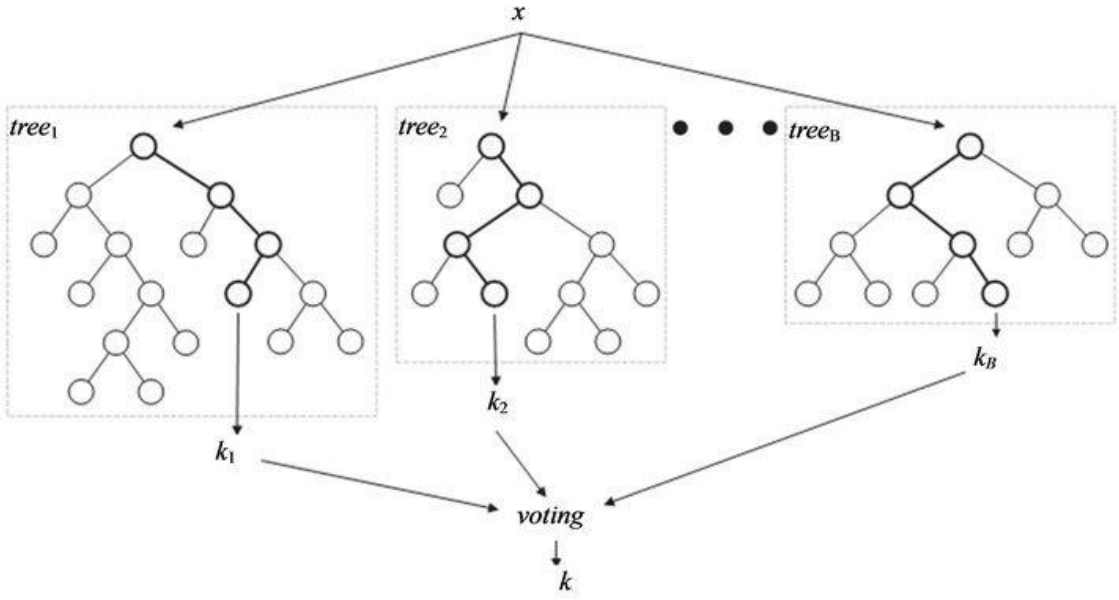
Na tom principu algoritam bira koje će grananje biti sljedeće, odnosno koje će karakteristike postati sljedeći čvorovi. Na svakoj razini odabire onu kombinaciju koja ima najviši iznos informacijske dobiti. Nakon što je odabrana uklanja se iz opcija za sljedeću razinu i isti proces se opet ponavlja. Na tom principu rade ID3 i CART (*Classification And Regression Tree*) algoritmi [17].

2.2.3. Slučajna šuma

Slučajna šuma (eng. *random forest*) klasifikator je koji se najčešće koristi u problemima klasifikacije iako se može koristiti i kod regresije gdje pronalazi srednju vrijednost. Algoritam koristi veliki broj stabala odlučivanja koja u slučajevima klasifikacije glasanjem dolaze do rješenja. Osnovna početna metoda koja se ovdje koristi je takozvani *bagging*, najjednostavnija metoda za kombiniranje klasifikatora. Time se dobiva više setova podataka koji su iste veličine kao početni set, ali s određenim zamjenama (*bootstrap* setovi). Neki od tih novih setova mogu sadržavati podatke iz originalnog seta koji se više puta ponavljaju, a u drugima da ih ni nema. Takav pristup zvuči pomalo apsurdno budući da je broj setova s kojima se radi mnogo veći, ali time se dobiva više modela koji uče na različite načine. Takvim kombiniranjem više klasifikatora dobiva se pogodniji model za predviđanje, tj. ansambl metoda. Za stvaranje šume prvo je potrebno izraditi stabla koja će biti različita trenirajući ih na različitim podacima primjenom *bagging* metode. U svrhu dodatnog povećanja nasumičnosti potrebno je ograničiti izbore kod stabala. Zbog toga se na svakom čvoru stablu dodjeljuje nasumični podskup značajki, a ne cijeli set iz kojega stablo može dalje birati. Time je ubrzano vrijeme treniranja budući da ima manje značajki koje se traže na svakoj razini. Dalje su navedeni osnovni koraci u algoritmu slučajne šume [17].

Za svako od $\boldsymbol{N}$ stabala:

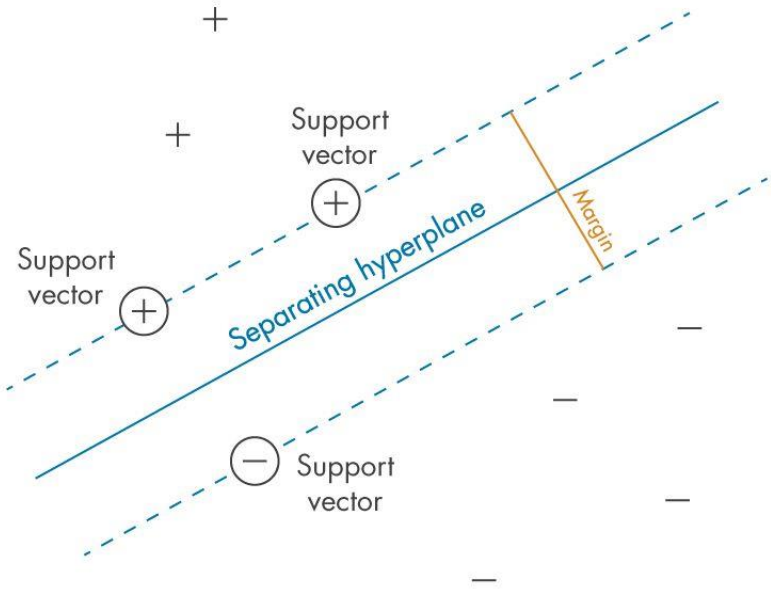
- izraditi novi bootstrap uzorak iz seta za treniranje
- pomoću tog uzorka trenirati stablo odlučivanja
- na svakom čvoru stabla odlučivanja, nasumično odabrati $\boldsymbol{m}$ značajki i izračunati informacijsku dobit samo na tom setu značajki, birajući optimalnu
- ponavljati proces dok stablo nije dovršeno



Slika 6. Shema slučajne šume [18]

2.2.4. Metoda potpornih vektora

Metoda potpornih vektora (eng. *support vector machine, SVM*) je algoritam koji se primjenjuje najčešće za klasifikaciju, ali često i kod regresije. Osmišljena je početkom 1990-ih godina, a danas je sve popularnija i koristi se u raznim područjima. Može obavljati prepoznavanje lica i teksta, medicinske dijagnoze, predviđanje vremenskih nizova itd. Od ostalih metoda mnogo je naprednija u pronalaženju suptilnih uzoraka u podacima. Podatke za klasifikaciju moguće je podijeliti u dvije cjeline pomoću hiperravnine. Cilj je da hiperravnina bude smještena tako da margina, tj. razmak između najbližih točaka različitih klasa bude najveći [19].



Slika 7. Skica hiperravnine [20]

Iz skice je vidljivo da ja margina najveća širina plohe paralelne hiperravnini koja ne sadrži točke podataka u sebi (u ovom slučaju + ili -). Na tim rubovima su definirani potporni vektori koji leže na točkama koje su najbliže ravnini i predstavljaju ekstremne vrijednosti klasa. Da bi se postizao bolji rezultat metoda mora tražiti maksimalni iznos margine. Ako se pokaže da problem nije linearno separabilan, algoritam propušta mali dio podataka da bude krivo kategoriziran što se penalizira zbog netočnosti [19]. U takvim slučajevima potrebno je provesti određene transformacije podataka. Ako je moguće koristiti više dimenzija može se pronaći linearna odluka koja razdvaja klase. Naglasak je na tome da ne postoje novi podaci nego se već iz postojećih stvara funkcija $\boldsymbol{\phi}(\boldsymbol{x_i})$ iz ulaza $\boldsymbol{x_i}$ [**17**].

2.2.2.4. Kernel

Radi boljeg objašnjenja jezgrenih (*kernel*) metoda može se započeti od primjera s polinomom drugog stupnja. On sadrži konstantu 1, svaki od ulaznih elemenata $x_1, x_2, \dots, x_d$, kvadrate tih ulaznih elemenata $x_1^2, x_2^2, \dots, x_d^2$ i produkt svakog para tih elemenata $x_1x_2, x_1x_3, \dots, x_{d-1}x_d$. Vektor koji sve to sadrži zapisuje se kao $\mathbf{\Phi}$(x). Nedostatak je što funkcija $\Phi(x_i)$ ima $d^2/2$ elemenata koji se množe s još jednom iste veličine i to više puta zbog čega je komputacijski zahtjevno. Rješenje toga je da se skalarni umnožak vektora $\mathbf{\Phi}(x)^T\mathbf{\Phi}$(y) može zapisati kao:

$$\mathbf{\Phi}(x)^T\mathbf{\Phi}(\mathrm{y}) = \sum_{i,j=1;i<j}^{d} x_i\, x_j y_i y_j \tag{8}$$

Daljnjom faktorizacijom dobiven je sljedeći zapis:

$$\mathbf{\Phi}(x)^T\mathbf{\Phi}(\mathrm{y}) = (1 + \boldsymbol{x}^T\boldsymbol{y})^2 \tag{9}$$

Time je broj potrebnih množenja smanjen na d, što je znatno bolje jer je komputacijska složenost pala sa $O(d^2)$ na $O(d)$. Važno je da se uklonio problem računanja skalarnih umnožaka svih proširenih vektora. To je zamijenjeno korištenjem Gramove matrice (kernel matrice) $\boldsymbol{K}$ koja se sastoji od skalarnih umnožaka originalnih vektora. To je poznatije kao *kernel* trik i omogućuje da se podaci znatno efikasnije transformiraju u više dimenzije. Taj postupak primjenjiv je i kod drugih komputacijskih problema gdje se upotrebljavaju skalarni umnošci [17].

2.2.5. K-najbližih susjeda

K-najbližih susjeda (engl. *k-nearest neighbors*) ili skraćeno KNN algoritam jedan je od najjednostavnijih, ali opet i ključnih klasifikacijskih algoritama. Široku upotrebu ima u stvarnim slučajevima budući da je neparametarski model što znači da ne donosi neke temeljne pretpostavke o distribuciji podataka. Također, može raditi s numeričkim i kategoričkim varijablama što ga čini pogodnim alatom za probleme klasifikacije i regresije. Osnova rada KNN algoritma je pronalazak *k* najbližih susjeda, odnosno klasa određenoj podatkovnoj točki na temelju metrike udaljenosti kao što je npr. euklidska udaljenost. Klasa te podatkovne točke tada se određuje većinom glasova ili prosjekom *k* susjeda. Ključan korak u izradi dobrog modela korištenjem ovog algoritma je odabir vrijednosti *k* koja se bira na temelju ulaznih podataka. Ako je prisutno puno ekstremnih vrijednosti ili šumova poželjno je birati veću vrijednost. Također, preporučeno je uzimati vrijednost koja je neparan broj kako bi se izbjegle veze u klasifikaciji [21]. Postoji više načina na koji se može računati udaljenost između klasa i točaka s obzirom na podatke koji se koriste od kojih su neke navedene u nastavku.

- **Euklidska udaljenost**: kartezijska udaljenost između dvije točke koje se nalaze u istoj ravnini/hiperravnini, odgovara duljini ravne linije koja povezuje dvije točke. Računa se prema sljedećoj formuli:

$$\boldsymbol{d(x,y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}} \tag{10}$$

Gdje su:

d – udaljenost između dvije točke

$\boldsymbol{x_i}$ – vrijednosti vertikalne osi u koordinatnoj ravnini

$\boldsymbol{y_i}$ – vrijednosti horizontalne osi u koordinatnoj ravnini

- **Manhattanova udaljenost**: koristi se za računanje ukupne duljine koju je objekt prošao umjesto pomicanja, jednaka je sumi apsolutne razlike između koordinata točaka:

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i| \tag{11}$$

- **Minkowskijeva udaljenost**: prethodne dvije formule zapravo su posebni slučajevi ove metode računanja udaljenosti, prema navedenoj formuli kad je $p = 2$ dobiva se euklidska udaljenost, a kad je $p = 0$ dobivena je Manhattanova udaljenost:

$$d(x,y) = (\sum_{i=1}^{n}(x_i - y_i)^p)^{\frac{1}{p}} \tag{12}$$

Rad algoritma može se objasniti u nekoliko koraka. Prvo je potrebno odabrati optimalnu vrijednost *k* susjeda, odnosno klasa koje dolaze u obzir kod predviđanja. Da bi se odredila sličnost između podatkovne točke i ostalih klasa, potrebno je izračunati udaljenosti između njih. Nakon toga, *k* točaka iz skupa podataka koji su najbliže podatkovnoj točki koju želimo klasificirati su prepoznati kao najbliži susjedi. Time se dolazi do glasanja, odnosno klasa koja se najviše puta pojavljuje među susjedima postaje predviđena klasa te podatkovne točke.

Iako algoritam nije kompleksan i ne zahtijeva mnogo parametara za treniranje modela, pri velikoj količini podataka dolazi do poteškoća s pravilnim radom. Također, ovaj algoritam pripada grupi koja se zovu lijeni algoritmi (engl. *lazy algorithm*) čija je karakteristika da ne izrađuju eksplicitno model tijekom faze treniranja. Umjesto toga, lijeni algoritmi vrše predviđanja ili klasifikaciju na temelju sličnosti novih, neviđenih podataka s onima u skupu za treniranje. Zbog toga su dosta zahtjevni po pitanju potrebne računalne snage, ali i pohrane podataka [21].

3. PRIPEMA I ANALIZA PODATAKA

Za pripremu i analizu podataka, primjenu klasifikacijskih metoda i njihove usporedbe korišten je programski jezik opće namjene *Python*. Zbog automatske memorijske alokacije sličan je jezicima kao što su *Ruby, Smalltalk, Perl*. Pogodan je za objektno orijentirano, strukturno i aspektno orijentirano programiranje što ga trenutno čini jednim od najpopularnijih programskih jezika. U ovom radu korištena je verzija *Python* 3.12.0 [22].

3.1. Biblioteke

Podatke s kojima se radi potrebno je čim bolje analizirati i vizualizirati. Iz tog razloga na početku pisanja programa uvode se potrebne biblioteke, paketi i moduli.

3.1.1. NumPy

NumPy je osnovni paket koji se koristi za znanstveno računanje u *Pythonu*. Omogućuje operacije s objektima višedimenzijskih nizova, niz rutina za brze operacije nad nizovima koje obuhvaćaju matematičke i logičke operacije, manipulacije oblicima, sortiranje, selektiranje, diskretnu Fourierovu transformaciju, osnove linearne algebre, osnovne statističke operacije i razne druge mogućnosti [23].

3.1.2. Pandas

Za lakši i efikasniji rad s podacima korišten je Python paket *pandas* koji pruža fleksibilne i brze strukture podataka. Služi kao alat za praktičnu analizu podataka iz stvarnog svijeta i primjeren je za više različitih vrsta podataka poput: tabličnih podataka s heterogenim stupcima (SQL, Excel tablice), uređene i neuređene vremenske serije podataka, proizvoljni matrični podaci s označenim redcima i stupcima i bilo koji oblik opažajućeg/statističkog seta podataka. Dvije osnovne strukture podataka ovog paketa su *Series* i *DataFrame* koje se koriste u mnogim slučajevima u statistici, financijama i raznim područjima inženjerstva [24].

3.1.3. Matplotlib.pyplot

Matplotlib.pyplot je modul unutar *Matplotlib* biblioteke koji se koristi za izradu 2D grafova i sličnu vizualizaciju podataka. *Pyplot* modul služi kao praktično sučelje za stvaranje niza grafova poput linijskih grafova, dijagrama raspršenja, histograma, stupčastih dijagrama itd. Sučelje je vrlo slično onom u MATLAB-u i s tim ciljem je dizajnirano.

3.1.4. Seaborn

Seaborn je biblioteka koja se koristi za izradu statističkih grafova. Usko se povezuje s podatkovnim strukturama *pandas* paketa i nadovezuje se na *matplotlib*. Njegove funkcije crtanja rade na podatkovnim okvirima i nizovima koji sadrže cijele skupove podataka i izvode potrebno semantičko mapiranje i statističko združivanje za izradu dijagrama [25].

3.2. Učitavanje podataka

Kako bi bilo moguće analizirati i vizualizirati podatke potrebno je prvo učitati sve potrebne biblioteke, module i pakete pomoću naredbe **import as.**

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline

        import warnings
        warnings.filterwarnings('ignore')
```

Slika 8. Unos biblioteka

Warnings modul dodan je da bi se sigurno mogla izbjeći upozorenja koja bi ometala izvođenje programa. Kada je sigurno da upozorenja nisu toliko ozbiljna da bi se radi njih narušavao rad programa onda je pogodno koristiti **warnings.filterwarnings ('ignore')** čime se zanemaruju sva upozorenja pri izvođenju. Skup podataka koji je korišten zapisan je u tabularnom obliku u **csv** datoteci gdje je svaki podatak zarezom odvojen od drugih. Učitavanje skupa podataka vrši se pomoću naredbe **pd.read_csv**. Nakon što su podaci učitani, korištenjem **df.head()** moguće je vidjeti prvih 5 redova u setu podataka, a posljednjih 5 pomoću **df.tail()**.

```
In [2]: df = pd.read_csv('heart.csv')

        df.head()
```

Out[2]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

Out[3]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
913	45	M	TA	110	264	0	Normal	132	N	1.2	Flat	1
914	68	M	ASY	144	193	1	Normal	141	N	3.4	Flat	1
915	57	M	ASY	130	131	0	Normal	115	Y	1.2	Flat	1
916	57	F	ATA	130	236	0	LVH	174	N	0.0	Flat	1
917	38	M	NAP	138	175	0	Normal	173	N	0.0	Up	0

Slika 9. Učitavanje podataka

3.3. Eksploracijska analiza podataka

Prvi korak u izradi modela strojnog učenja je razumijevanje i analiza seta podataka s kojim se radi. Eksploracijska analiza podataka podrazumijeva pronalaženje glavnih karakteristika podataka, otkrivanje uzoraka i anomalija. Time se lakše donose informirane odluke kako će se podaci pretprocesirati i transformirati da budu pogodni za izradu modela. Provodi se u više faza s obzirom koliko varijabla se odjednom analizira i time se dijeli na univarijatnu ili multivarijatnu analizu.

Za dobivanje sažetog sadržaja okvira podataka (engl. *data frame*) korištena je naredba **df.info()**.

```
In [5]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

Slika 10. Osnovne informacije o setu podataka

Iz slike 10 vidljiv je sadržaj seta podataka i tipovi podataka koji su u njemu zapisani. *RangeIndex* pokazuje ukupni broj unosa, *Data columns* broj redova koji predstavljaju atribute (dob, spol, kolesterol...). *Non-null count* predstavlja broj unosa u svakom stupcu koji nisu prazne vrijednosti, odnosno nula. Takve vrijednosti je potrebno uklanjati jer mogu predstavljati više problema za model, a u ovom slučaju nijedan stupac ne sadrži takve vrijednosti. *Dtype* daje vrstu podataka pojedinog atributa ovisno je li cijeli broj (oznaka *int64*) ili podatak bilo koje druge vrste (oznaka *object*).

Prikaz detaljnije opisne statistike dobiven je korištenjem naredbe **df.describe()** koja vraća sažetak centralne tendencije, statističke disperzije i oblik distribucije numeričkih stupaca seta podataka.

```
In [6]: df.describe()
```

Out[6]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

Slika 11. Opisna statistika seta podataka

Statistika koju naredba vraća je sljedeća:

- Count – broj ne praznih (non-null) vrijednosti
- Mean – prosječna vrijednost svakog stupca
- std – standardna devijacija
- min – minimalna vrijednost svakog stupca
- 25 % - prvi kvartal, vrijednost od koje je manje 25 % vrijednosti iz cijelog seta podataka
- 50 % - medijan, određuje sredinu distribucije svakog stupca
- 75 % - treći kvartal, vrijednost od koje je manje 75 % vrijednosti iz cijelog seta podataka
- max – maksimalna vrijednost svakog stupca

Budući da je pomoću prethodne naredbe dobivena opisna statistika za brojčane vrijednosti isto se može primijeniti i za ostale podatke. U ovom slučaju dobivene su drugačije opisne karakteristike:

- unique – broj unikatnih vrijednosti u stupcu
- top – vrijednost koja se najviše ponavlja u stupcu
- freq – frekvencija pojavljivanja top vrijednosti u stupcu

```
In [7]: df.describe(include='object')
```

Out[7]:

	Sex	ChestPainType	RestingECG	ExerciseAngina	ST_Slope
count	918	918	918	918	918
unique	2	4	3	2	3
top	M	ASY	Normal	N	Flat
freq	725	496	552	547	460

Slika 12. Opisna statistika ne brojčanih podataka

Radi boljeg pregleda jedinstvenih vrijednosti dalje je korištena naredba **df.nunique()**. Time se prebrojavaju unikatne vrijednosti u svakom stupcu i daje se bolji uvid u razinu različitosti ili kardinalnosti svakog stupca.

```
In [8]: df.nunique()

Out[8]: Age               50
        Sex                2
        ChestPainType      4
        RestingBP         67
        Cholesterol      222
        FastingBS          2
        RestingECG         3
        MaxHR            119
        ExerciseAngina     2
        Oldpeak           53
        ST_Slope           3
        HeartDisease       2
        dtype: int64
```

Slika 13. Unikatne vrijednosti stupaca

3.3.1. Korelacija između varijabli

U statistici, korelacija se odnosi na opis povezanosti između dvije ili više statističkih varijabli. Takav opis često nije potpuno precizan i može uključivati iznimke, no omogućuje predviđanje vrijednosti jedne varijable na temelju poznatih vrijednosti druge varijable. Za primjer, može se očekivati da će tjelesna masa viših ljudi biti veća, ali postoje iznimke. Gledajući prema vrsti veze između varijabli, korelacija može biti linearna ili nelinearna. Linearna korelacija prikazuje linearni porast (pozitivna korelacija) ili linearno smanjenje (negativnu korelaciju) jedne statističke varijable pri linearnom porastu druge. Mjera za jakost linearne korelacije izražava se Pearsonovim koeficijentom korelacije $\boldsymbol{r}$ koji može imati vrijednosti između +1 i -1. Ovisno o vrijednosti $\boldsymbol{r}$ moguće je više slučajeva [26]:

- Potpuna korelacija – postiže se kada je koeficijent korelacije $\boldsymbol{r = +1}$ (pozitivna korelacija) ili $\boldsymbol{r = -1}$ (negativna korelacija), odnosno vjerojatnost da određenoj vrijednosti jedne varijable odgovara određena vrijednost druge varijable jednaka je 1

- Nepotpuna korelacija – postiže se kada je koeficijent korelacije između vrijednosti -1 i 1, ali različit od nule (-1< $\boldsymbol{r}$<0 , 0< $\boldsymbol{r}$<1), odnosno kada je vjerojatnost da će određenoj vrijednosti jedne varijable odgovarati određena vrijednost druge varijable između ovog raspona

- Ne postoji korelacija – postiže se kada je $\boldsymbol{r = 0}$

Ako se radi s većim brojem varijabli i potrebno je vidjeti na koji način više njih utječe jedna na drugu izrađuje se matrica korelacije. U takvoj matrici reci i stupci predstavljaju varijable, a presjek na određenom mjestu je koeficijent korelacije tih varijabli. Matrica je dijagonalno popunjena s jedinicama budući da su na dijagonali presjeci gdje su i jedna i druga varijabla iste. Također, matrica je simetrična što znači da su iznad ili ispod dijagonale koeficijenti korelacije za isti par varijabli identični. Matrica korelacije izrađena je korištenjem naredbe **df.corr()** .

```
In [11]: df.corr()
```

Out[11]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
Age	1.000000	0.254399	-0.095282	0.198039	-0.382045	0.258612	0.282039
RestingBP	0.254399	1.000000	0.100893	0.070193	-0.112135	0.164803	0.107589
Cholesterol	-0.095282	0.100893	1.000000	-0.260974	0.235792	0.050148	-0.232741
FastingBS	0.198039	0.070193	-0.260974	1.000000	-0.131438	0.052698	0.267291
MaxHR	-0.382045	-0.112135	0.235792	-0.131438	1.000000	-0.160691	-0.400421
Oldpeak	0.258612	0.164803	0.050148	0.052698	-0.160691	1.000000	0.403951
HeartDisease	0.282039	0.107589	-0.232741	0.267291	-0.400421	0.403951	1.000000

Slika 14. Matrica korelacije

3.3.2. Grupiranje objekata

Kako bi se bolje prikazali neki kategorični podaci u stupcima korištena je naredba **df.groupby()**. Time je podatkovni okvir podijeljen u grupe po određenim kriterijima kao što su npr. ime stupca, lista imena stupaca. Nakon toga moguće je primijeniti funkciju **mean()** i stvoren je novi podatkovni okvir. U ovom slučaju kriteriji po kojima je dijeljeno bili su stupci ChestPainType RestingECG. Nakon toga dobiven je bolji pregled rasporeda vrste boli u prsima i vrijednosti EKG-a po prosječnim vrijednostima drugih kriterija.

```
In [4]: grupno_cpt=df.groupby('ChestPainType')
        grupno_cpt.mean()
```

Out[4]:

ChestPainType	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
ASY	54.959677	133.229839	186.645161	0.284274	128.477823	1.162702	0.790323
ATA	49.242775	130.624277	233.046243	0.109827	150.208092	0.307514	0.138728
NAP	53.310345	130.960591	197.438424	0.201970	143.236453	0.674877	0.354680
TA	54.826087	136.413043	207.065217	0.282609	147.891304	1.036957	0.434783

Slika 15. Grupiranje objekata za stupac ChestPainType

```
In [6]: grupno_ECG=df.groupby('RestingECG')
        grupno_ECG.mean()
```

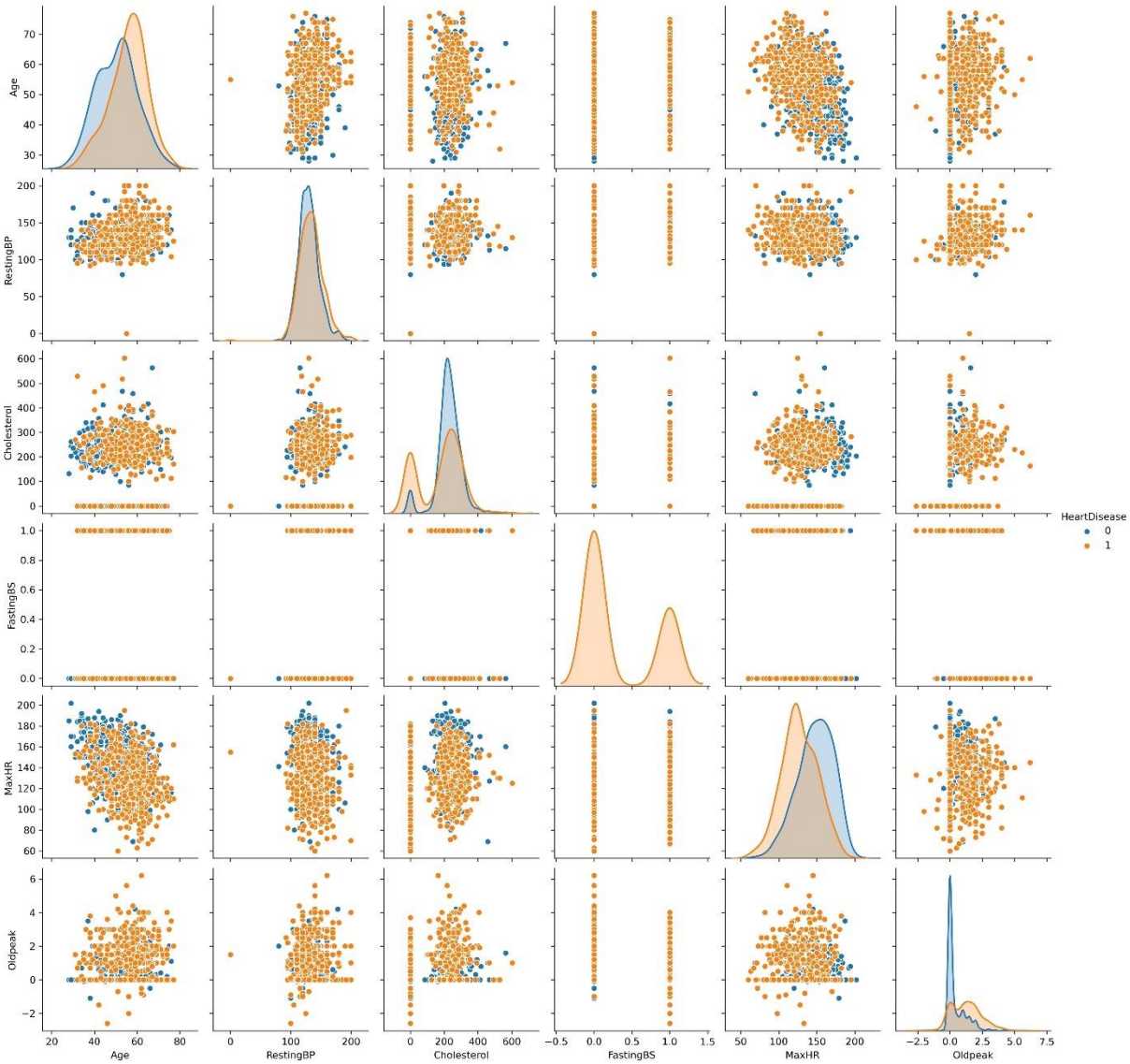
Out[6]:

RestingECG	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
LVH	56.218085	134.335106	236.946809	0.223404	143.117021	1.069681	0.563830
Normal	51.740942	130.635870	195.375000	0.201087	137.302536	0.786051	0.516304
ST	56.140449	135.808989	169.129213	0.342697	128.617978	1.008989	0.657303

Slika 16. Grupiranje objekata za stupac RestingECG

Za prikaz svakog para stupaca koji sadrže brojčane vrijednosti uporabljena je naredba **sns.pairplot()** iz Seaborn biblioteke. Time je stvoren matrični prikaz dijagrama raspršenja gdje su na dijagonali grafovi gustoće jezgre. Točke raspršenja su vrijednosti iz stupca HeartDisease, odnosno je li prisutna srčana bolest ili nije. Naredbom **plt.savefig()** grafički prikaz spremljen je u formatu slike.

```
In [8]: sns.pairplot(df,hue='HeartDisease')
        plt.savefig('seaborn_plot.jpg',dpi=300)
```



Slika 17. Stvaranje matričnog dijagrama raspršenja

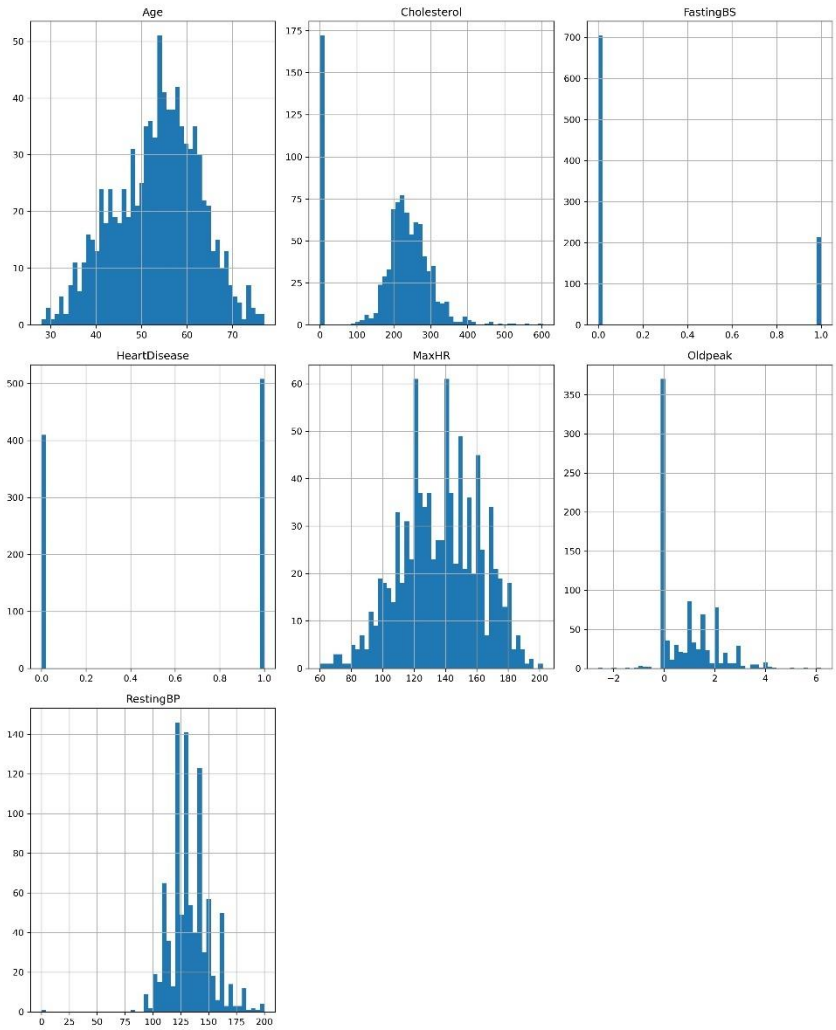
3.3.3. Univarijatna analiza

Univarijatnom analizom promatra se, odnosno prikazuje samo jedna varijabla odjednom kako bi se lakše utvrdila distribucija određene varijable u podacima. Najčešći prikazi za ovakvu vrstu analize su stupčasti i kutijasti dijagrami, histogrami i strukturni krug.

Frekvencija pojavljivanja brojčanih vrijednosti po svakom pojedinom stupcu može se prikazati histogramom. Vertikalna os histograma pokazuje broj koliko se puta vrijednosti ponavljaju dok horizontalna os predstavlja same vrijednosti. Izrada histograma vrši se pomoću naredbe **df.hist()**, a pomoću naredbu **plt.suptitle()** dodan je naziv i prilagođena veličina prikaza. Za podešavanje razmaka na prikazu korištena je naredba **plt.tight_layout()**.

```
In [12]: df.hist(figsize=(13,16),bins=50)
         plt.suptitle('Distribucija značajki',x=0.5,y=1.02,ha='center',fontsize='large')
         plt.tight_layout()
         plt.savefig('histogram_plot.jpg',dpi=300)
```

Slika 18. Stvaranje histograma

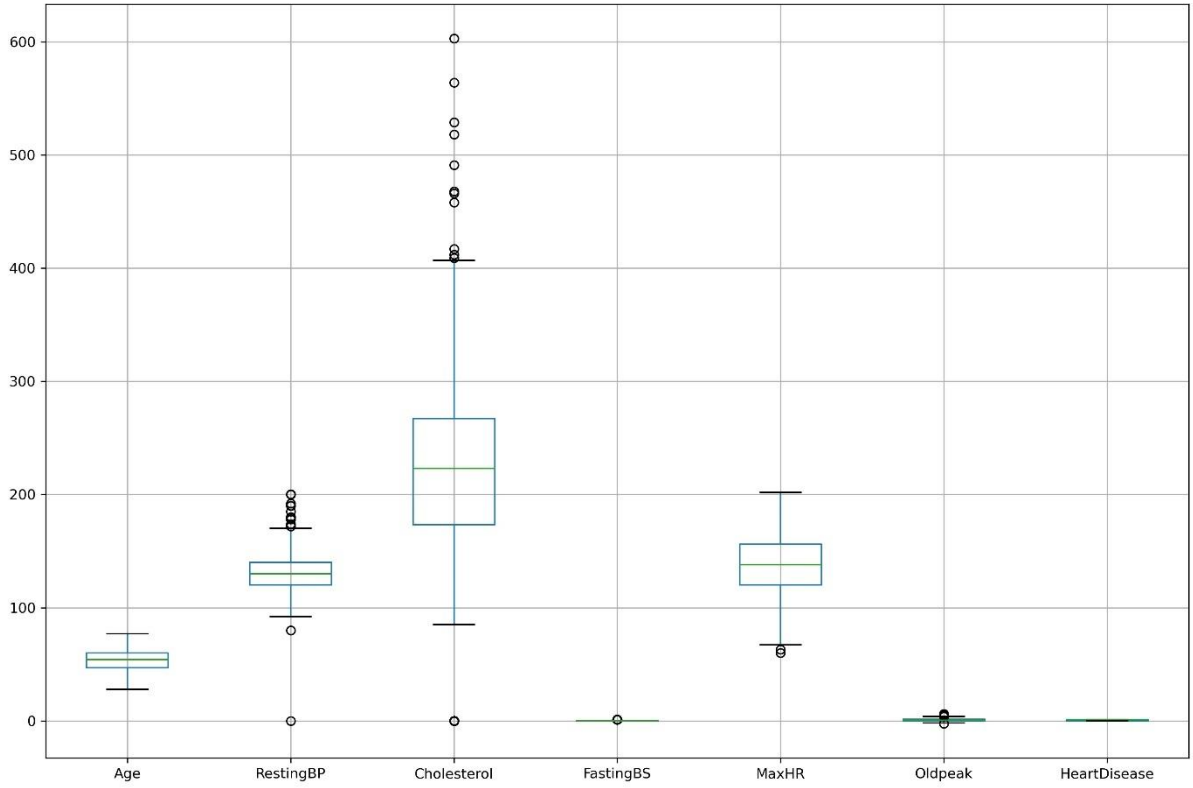


Slika 19. Grupni histogram svih brojčanih značajka

Još jedna pogodna vrsta prikaza je kutijasti dijagram. Njime se praktično prikazuje odnos minimuma, maksimuma, donjega i gornjega kvartala te medijana podataka. Krajnje linije kutije na dijagramu su iznosi donjega i gornjega kvartala, crta u sredini kutije označava iznos medijana dok crtice udaljene od kutije označuju minimalne i maksimalne vrijednosti. Također, točkama se prikazuju vrijednosti koje odstupaju. Kutijasti dijagram izrađen je pomoću **df.boxplot()** naredbe, a veličina i razmak prikaza postavljeni su kao i u prethodnom primjeru za izradu histograma.

```
In [14]: df.boxplot(figsize=(12,8))
         plt.suptitle('Kutijasti dijagram značajki',x=0.5,y=1.02,ha='center',fontsize='large')
         plt.tight_layout()
         plt.savefig('box_plot.jpg',dpi=300)
```

Slika 20. Stvaranje kutijastog dijagrama



Slika 21. Kutijasti dijagram svih brojčanih značajki

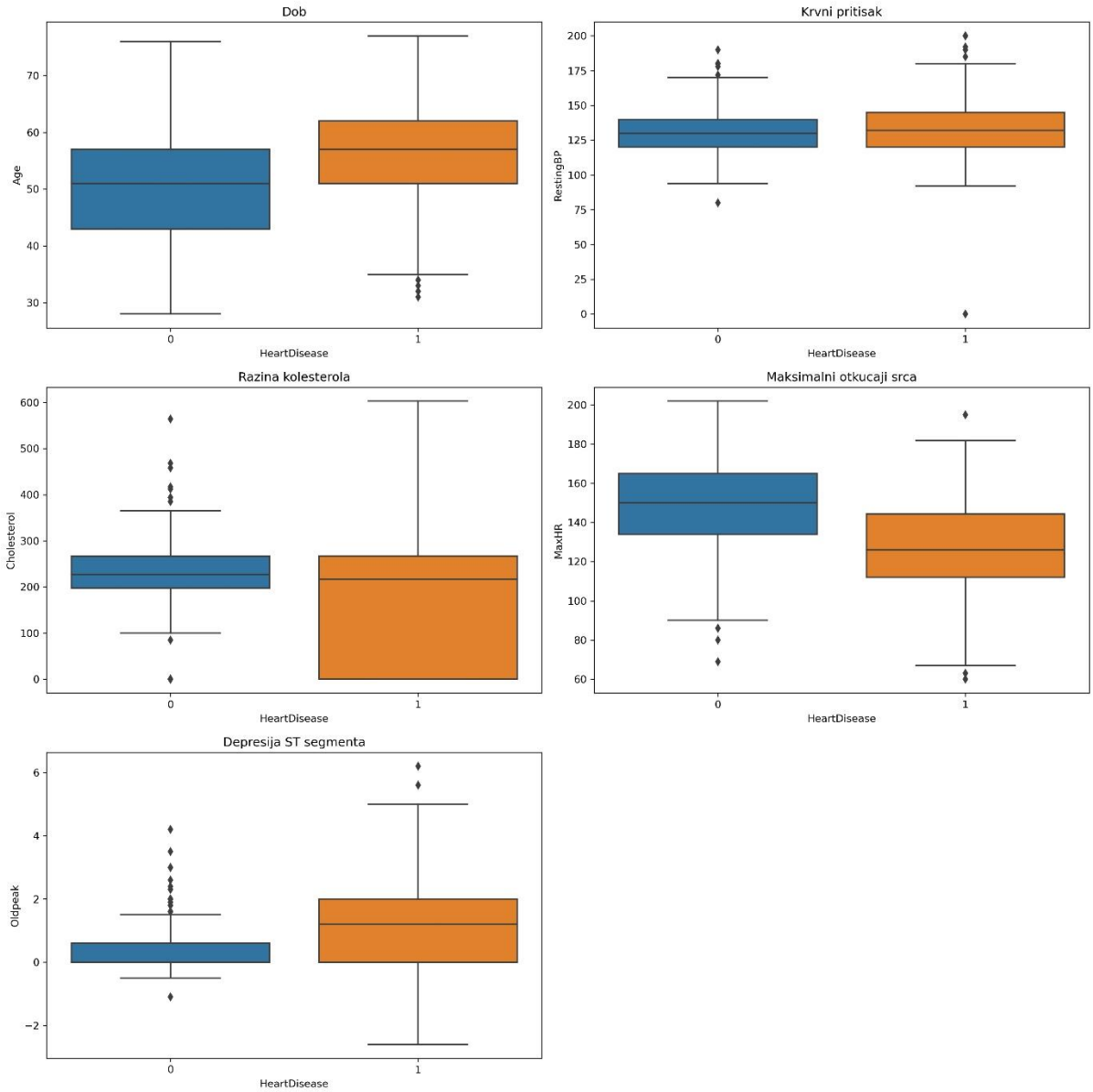
3.3.4. Bivarijatna analiza

Istovremena analiza dvije varijabla naziva se bivarijatna analiza. Svrha takvog pristupa je pronalaženje veza ili asocijacija između varijabli i koliko je to izraženo. Analogno tome služi da se vide razlike između varijabli i kakav je utjecaj tih razlika.

Kutijasti dijagrami također su pogodni za ovakvu analizu jer sve brojčane vrijednosti možemo promatrati s obzirom na stupac HeartDisease, odnosno je li prisutna (označeno plavom kutijom) srčana bolest ili nije (označeno narančastom kutijom). Svaki zasebni dijagram izrađen je korištenjem naredbe **sns.boxplot()** gdje je svugdje za x os odabran stupac HeartDisease, a za y os drugi stupci brojčanih vrijednosti. Naredba **plt.subplot()** korištena je kako bi svi dijagrami bili na jednom prikazu.

```
In [20]: plt.figure(figsize=(15,15))

         plt.subplot(3,2,1)
         sns.boxplot(x=df['HeartDisease'],y=df['Age'])
         plt.title('Dob')

         plt.subplot(3,2,2)
         sns.boxplot(x=df['HeartDisease'],y=df['RestingBP'])
         plt.title('Krvni pritisak')

         plt.subplot(3,2,3)
         sns.boxplot(x=df['HeartDisease'],y=df['Cholesterol'])
         plt.title('Razina kolesterola')

         plt.subplot(3,2,4)
         sns.boxplot(x=df['HeartDisease'],y=df['MaxHR'])
         plt.title('Maksimalni otkucaji srca')

         plt.subplot(3,2,5)
         sns.boxplot(x=df['HeartDisease'],y=df['Oldpeak'])
         plt.title('Depresija ST segmenta')

         plt.tight_layout()
         plt.savefig('box_plot_bi.jpg',dpi=300)
```

Slika 22. Bivarijatna analiza - stvaranje kutijastog dijagrama



Slika 23. Bivarijatna analiza - kutijasti dijagrami svih brojčanih stupaca

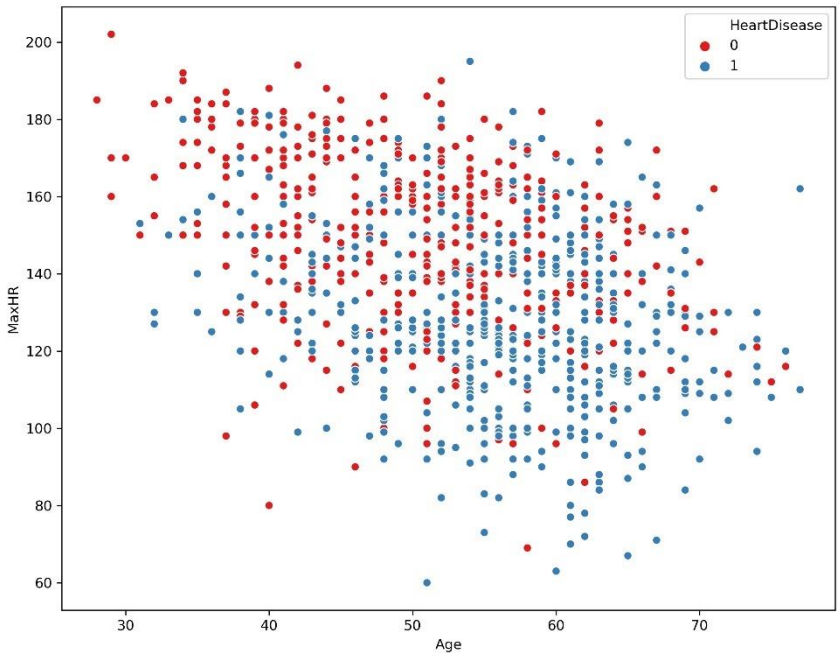
3.3.4.1. Dijagrami raspršenja

Dijagramom raspršenja moguće je istodobno prikazati dvije varijable gdje je svaka varijabla na zasebnoj osi. Ako postoji korelacija između varijabli točke na dijagramu pratit će neku krivulju, odnosno bit će gušće raspoređene ako je veća korelacija. Dvije vrste točaka na dijagramu će biti semantični podaci iz stupca HeartDisease. Dijagrami raspršenja stvoreni su korištenjem naredbe **sns.scatterplot()**. Parametri funkcije x i y predstavljaju osi dijagrama za što su odabrani stupci Age i MaxHR. Parametar **hue** odnosi se na kategoričnu varijablu pomoću

koje će se obojati točke raspršenja i za to je postavljen stupac HeartDisease. Pomoću zadnjega parametra **palette** izabrana je boja točaka.

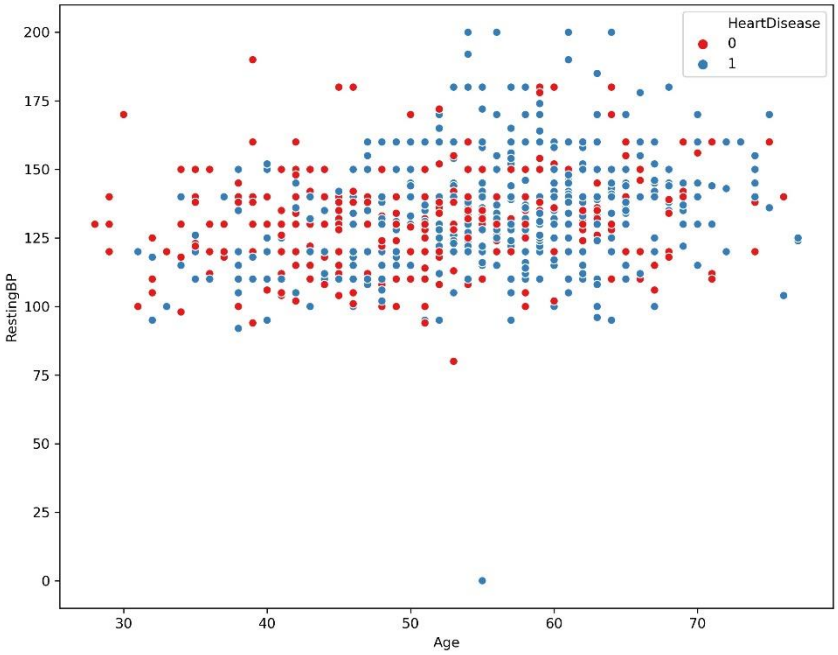
```
In [14]: plt.figure(figsize=(10,8))
         sns.scatterplot(data=df,x='Age',y='MaxHR',hue='HeartDisease',palette='Set1')
         plt.savefig('scatter_bi.jpg',dpi=300)
```

Slika 24. Stvaranje dijagrama raspšenja za varijable dob i maksimalni otkucaji srca

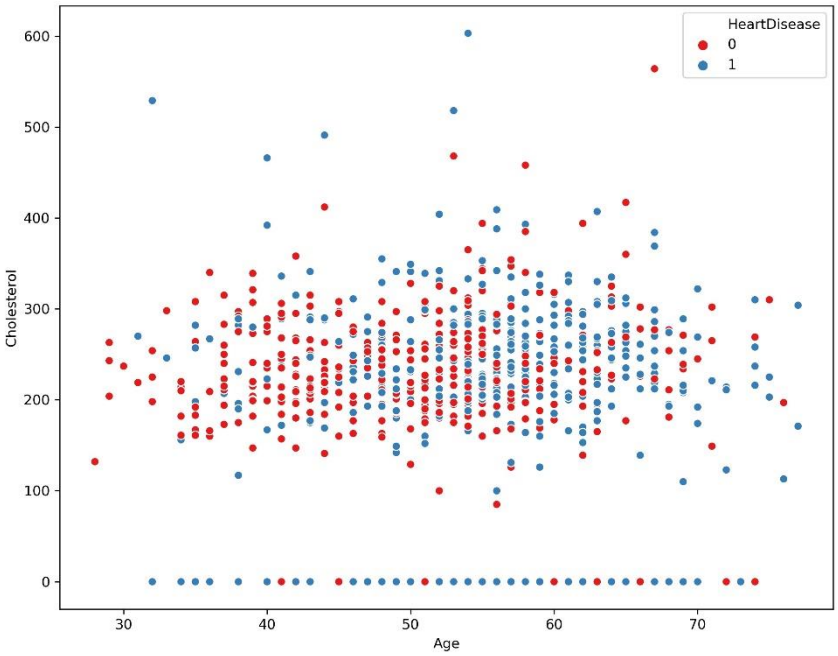


Slika 25. Dijagram raspšenja za varijable dob i maksimalni otkucaji srca

Istim takvim postupkom izrađena su daljnja dva dijagrama koji kao horizontalnu os zadržavaju dobnu varijablu. Izmijenjena je jedino vertikalna os koja u jednom slučaju prikazuje iznos krvnog tlaka (stupac RestingBP), a u drugom iznos kolesterola (stupac Cholesterol).



Slika 26. Dijagram raspršenja za varijable dob i krvni pritisak



Slika 27. Dijagram raspšenja za varijable dob i razina kolesterola

3.3.4.2. Jednokratno enkodiranje

Kategorične vrijednosti predstavljaju vrstu podataka koja može biti podijeljena u grupe. U zadanom setu podataka to su varijable spol (stupac Sex), tip boli u prsima (stupac Chest Pain Type), EKG očitanje (stupac RestingECG), angina pri vježbanju (stupac Exercise Angina) i depresija ST segmenta (stupac ST Slope). Određeni algoritmi, poput stabala odlučivanja, mogu obraditi kategoričke podatke izravno u nekim primjenama. Međutim, većina algoritama zahtijeva numeričke ulaze ili izlaze zbog čega je potrebno transformirati kategoričke podatke u cijele brojeve. Rješenje se može postići postupkom jednokratnog enkodiranja (*engl. one-hot encoding*) pomoću kojeg se svaka kategorička vrijednost transformira u poseban kategorički stupac kojima se dodjeljuju vrijednosti 1 ili 0. Svaka vrijednost cijeloga broja prikazana je kao binarni vektor gdje su sve vrijednosti nula osim indeksa označenog s 1 [27]. Kao primjer može se uzeti varijabla spola i njezine vrijednosti u tablici ispod:

Tablica 2. Enkodirane vrijednosti za varijablu spol

Spol	**Muško (enkodirano)**	**Žensko (enkodirano)**
Muško	1	0
Žensko	0	1

Nakon što su odabrane numeričke vrijednosti za svaki spol stvoren je vektor koji predstavlja te vrijednosti. Time vrijednost muško može biti zapisana kao vektor **(**1,0**)** dok se vrijednost žensko zapisuje kao **(**0,1**)**.

Za provedbu jednokratnog enkodiranja korištena je funkcija **pd.get_dummies()** iz paketa *pandas*. Za parametre funkcije prvo je uveden skup podataka koji se enkodira. Nakon toga odabran je stupac za enkodiranje i pomoću postavljanja parametra **drop_first = True** izostavljena je prva kategorija svakog stupca, odnosno ona neće biti predstavljena sa zasebnim enkodiranim stupcem. Ako su sve ostale kategorije enkodirane kao 0 to onda znači da je izostavljena kategorija označena kao 1. Stanje nakon svakog enkodiranja pregledano je pomoću funkcije **head()** koja vraća prikaz prvih 5 redova skupa podataka.

```
In [5]: enkodirano_spol = pd.get_dummies(df,columns=['Sex'],drop_first=True)
        enkodirano_spol.head()
```

Out[5]:

	Age	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease	Sex_M
0	40	ATA	140	289	0	Normal	172	N	0.0	Up	0	1
1	49	NAP	160	180	0	Normal	156	N	1.0	Flat	1	0
2	37	ATA	130	283	0	ST	98	N	0.0	Up	0	1
3	48	ASY	138	214	0	Normal	108	Y	1.5	Flat	1	0
4	54	NAP	150	195	0	Normal	122	N	0.0	Up	0	1

Slika 28. Jednokratno enkodiranje kategorije spol

```
In [6]: enkodirano_bol = pd.get_dummies(enkodirano_spol,columns=['ChestPainType'],drop_first=True)
        enkodirano_bol.head()
```

Out[6]:

esterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease	Sex_M	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA
289	0	Normal	172	N	0.0	Up	0	1	1	0	0
180	0	Normal	156	N	1.0	Flat	1	0	0	1	0
283	0	ST	98	N	0.0	Up	0	1	1	0	0
214	0	Normal	108	Y	1.5	Flat	1	0	0	0	0
195	0	Normal	122	N	0.0	Up	0	1	0	1	0

Slika 29. Jednokratno enkodiranje kategorije tip boli u prsima

```
In [7]: enkodirano_ekg = pd.get_dummies(enkodirano_bol,columns=['RestingECG'],drop_first=True)
        enkodirano_ekg.head()
```

Out[7]:

HR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease	Sex_M	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA	RestingECG_Normal	RestingECG_ST
72	N	0.0	Up	0	1	1	0	0	1	0
56	N	1.0	Flat	1	0	0	1	0	1	0
98	N	0.0	Up	0	1	1	0	0	0	1
08	Y	1.5	Flat	1	0	0	0	0	1	0
22	N	0.0	Up	0	1	0	1	0	1	0

Slika 30. Jednokratno enkodiranje kategorije EKG očitanje

```
In [8]: enkodirano_ang = pd.get_dummies(enkodirano_ekg,columns=['ExerciseAngina'],drop_first=True)
        enkodirano_ang.head()
```

Out[8]:

Oldpeak	ST_Slope	HeartDisease	Sex_M	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA	RestingECG_Normal	RestingECG_ST	ExerciseAngina_Y
0.0	Up	0	1	1	0	0	1	0	0
1.0	Flat	1	0	0	1	0	1	0	0
0.0	Up	0	1	1	0	0	0	1	0
1.5	Flat	1	0	0	0	0	1	0	1
0.0	Up	0	1	0	1	0	1	0	0

Slika 31. Jednokratno enkodiranje kategorije angina pri vježbanju

```
In [10]: final_df = pd.get_dummies(enkodirano_ang,columns=['ST_Slope'],drop_first=True)

         final_df.head()
```

Out[10]:

ease	Sex_M	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA	RestingECG_Normal	RestingECG_ST	ExerciseAngina_Y	ST_Slope_Flat	ST_Slope_Up
0	1	1	0	0	1	0	0	0	1
1	0	0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	0	0	1
1	0	0	0	0	1	0	1	1	0
0	1	0	1	0	1	0	0	0	1

Slika 32. Jednokratno enkodiranje kategorije depresija ST segmenta

Nakon enkodiranja, naredbom **final_df.info()** dobiven je prikaz iz kojeg je vidljivo da je enkodiranje uspješno provedeno budući da su svi podaci sada označeni kao numeričke vrijednosti.

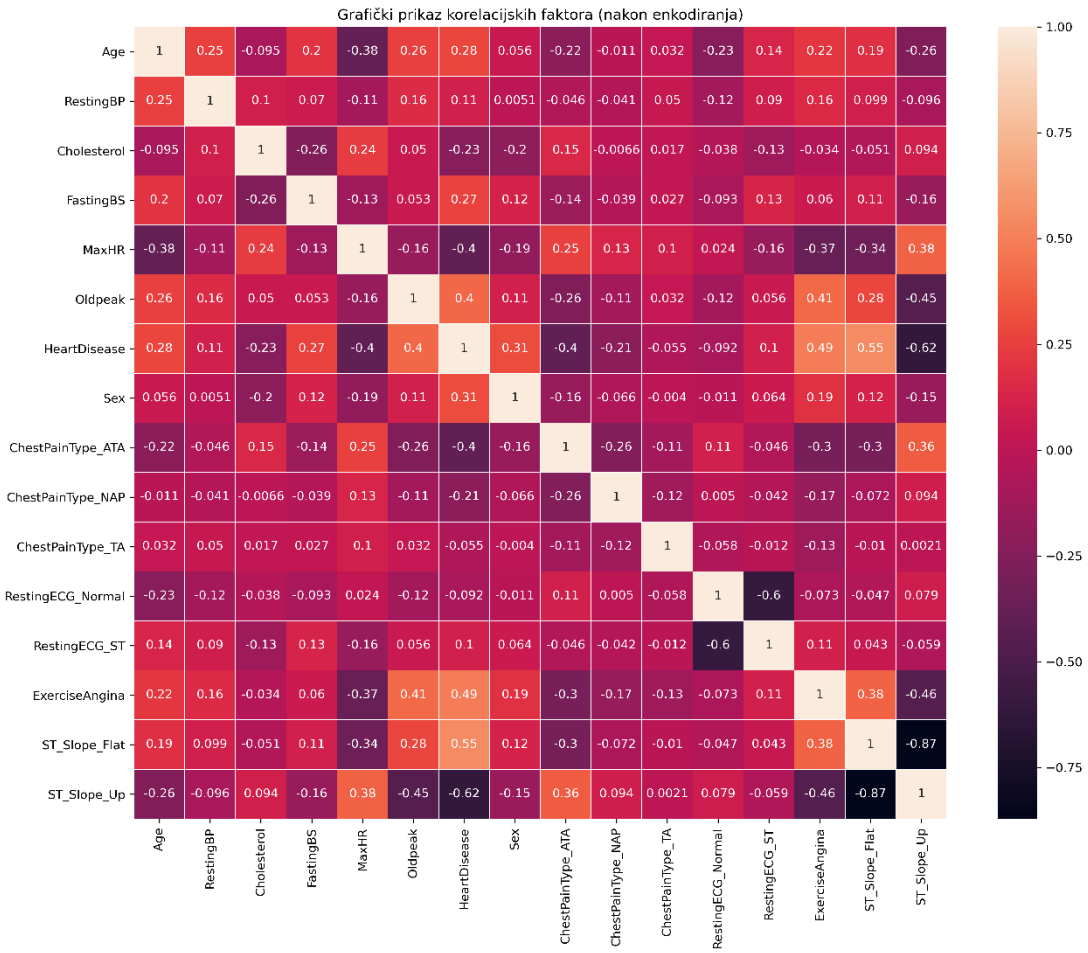
```
In [12]: final_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 16 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Age                918 non-null    int64
 1   RestingBP          918 non-null    int64
 2   Cholesterol        918 non-null    int64
 3   FastingBS          918 non-null    int64
 4   MaxHR              918 non-null    int64
 5   Oldpeak            918 non-null    float64
 6   HeartDisease       918 non-null    int64
 7   Sex                918 non-null    uint8
 8   ChestPainType_ATA  918 non-null    uint8
 9   ChestPainType_NAP  918 non-null    uint8
 10  ChestPainType_TA   918 non-null    uint8
 11  RestingECG_Normal  918 non-null    uint8
 12  RestingECG_ST      918 non-null    uint8
 13  ExerciseAngina     918 non-null    uint8
 14  ST_Slope_Flat      918 non-null    uint8
 15  ST_Slope_Up        918 non-null    uint8
dtypes: float64(1), int64(6), uint8(9)
memory usage: 58.4 KB
```

Slika 33. Sažeti opis skupa podataka nakon enkodiranja

Potpuni prikaz skupa podataka nakon enkodiranja izveden je kao matrični prikaz korelacijskih faktora. Iz njega je moguće vidjeti faktore korelacije između svih varijabli, a boja određenog polja označuje koliki je intenzitet korelacije između tih pojedinih varijabli. Grafički prikaz stvoren je pomoću naredbe **sns.heatmap()** gdje je parametar **annot=True** postavljen kako bi se vidjele numeričke vrijednosti unutar svakog polja dijagrama.

```
In [13]: final_df_corr = final_df.corr()
         plt.figure(figsize=(15,12))
         plt.title('Grafički prikaz korelacijskih faktora (nakon enkodiranja)')
         sns.heatmap(final_df_corr,annot=True,linecolor='white',linewidths=0.2)
         plt.savefig('heatmap',dpi=300)
```

Slika 34. Stvaranje matričnog prikaza korelacijskih faktora



Slika 35. Grafički prikaz korelacijskih faktora

4. IMPLEMENTACIJA I USPOREDBA ALGORITAMA

4.1. Učitavanje biblioteka i modula

Biblioteka iz koje su korišteni svi algoritmi i koja je koristila za njihovu usporedbu je *sklearn*. Punim imenom *Scikit-learn* je biblioteka otvorenog koda za strojno učenje i modeliranje podataka u *Pythonu* koja sadrži različite algoritme za klasifikaciju i regresiju. Dizajnirana je tako da ima potpunu povezanost s *NumPy* i *SciPy* bibliotekama.

```
In [9]: from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import MinMaxScaler
        from sklearn.metrics import accuracy_score,log_loss
        from sklearn.metrics import classification_report,confusion_matrix
        from sklearn.metrics import roc_curve,roc_auc_score
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.svm import SVC
        from sklearn.linear_model import LogisticRegression
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.naive_bayes import CategoricalNB
        import shap
```

Slika 36. Unos biblioteka i funkcija

4.1.1. Funkcije za pretprocesiranje podataka

Iz biblioteke *sklearn* uvezene su funkcije koje su korištene za pretprocesiranje podataka i za provjeru točnosti modela. Funkcija **train_test_split** korištena je za podjelu skupa podataka u dva skupa za treniranje i testiranje. Time se olakšava provjera točnosti modela jer se testiranje vrši na novom, različitom skupu podataka. **MinMaxScaler** funkcija koristi se kako bi se sve brojčane vrijednosti mogle prikazati u određenom rasponu, tipično između 0 i 1. To se postiže korištenjem sljedeće formule za svaku karakteristiku:

$$\boldsymbol{X_{skalirano}} = \frac{\boldsymbol{X} - \boldsymbol{X_{min}}}{\boldsymbol{X_{max}} - \boldsymbol{X_{min}}} \tag{13}$$

Gdje su:

$\boldsymbol{X_{min}}$ – minimalna vrijednost karakteristike

$\boldsymbol{X_{max}}$ – maksimalna vrijednost karakteristike

$\boldsymbol{X}$ – originalna vrijednost karakteristike

Time je omogućeno da su različite brojčane vrijednosti svedene na istu skalu što je bitno određenim algoritmima koji su osjetljivi na skalu ulaznih brojčanih vrijednosti poput logističke regresije i KNN algoritma.

4.1.2. Funkcije za mjere točnosti modela

Kako bi se drukčiji modeli, odnosno algoritmi mogli usporediti iz biblioteke *sklearn* su uvezene i funkcije za mjere točnosti modela. Funkcija **accuracy_score** dio je *metrics* modula i koristi se za evaluaciju točnosti modela gdje točnost predstavlja omjer broja točno predviđenih vrijednosti i ukupnog broja vrijednosti. Za računanje unakrsne entropije, odnosno logaritamskog gubitka korištena je funkcija **log_loss**. Logaritamski gubitak pokazatelj je koliko je vjerojatnost predviđanja blizu odgovarajuće stvarne vrijednosti (0 ili 1 u slučajevima binarne klasifikacije). Što više predviđena vrijednost odstupa od stvarne to je veća vrijednost logaritamskog gubitka. Funkcija **classification_report** pruža sažeto izvješće koje sadrži više različitih metoda mjere točnosti klasifikacijskih modela. Neke od metrika koje su izračunate i prikazane ovom funkcijom su preciznost, prisjećanje, mjera F1 i *support*, odnosno broj ponavljanja svake klase unutar skupa podataka.

4.1.2.1. Matrica zabune

Korištenjem funkcije **confusion_matrix** moguće je kao mjeru točnosti uvesti matricu zabune, kvadratnu matricu koja na sažet način prikazuje broj podudaranja i nepodudaranja predikcija i stvarnih oznaka. Matrica zabune za binarni klasifikator je dimenzija $2\ x\ 2$ i izgleda ovako:

$$\begin{array}{c} \\ y_{pred} = 1 \\ y_{pred} = 0 \end{array} \begin{array}{c} \begin{array}{cc} y_{true} = 1 & y_{true} = 0 \end{array} \\ \begin{pmatrix} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{pmatrix} \end{array} \tag{14}$$

Ovdje reci odgovaraju predviđenim oznakama (*pred*), a stupci stvarnim oznakama (*true*). Razlikuju se četiri vrste slučajeva koji odgovaraju elementima matrice:

- **stvarno pozitivni** (engl. *true positive, TP*) - predikcija je 1 i stvarna oznaka je 1
- **lažno pozitivni** (engl. *false positive, FP*) – predikcija je 1, a stvarna oznaka je 0
- **lažno negativni** (engl. *false negative, FN*) – predikcija je 0, a stvarna oznaka je 1

- **stvarno negativni** (engl. *true negative, TN*) – predikcija je 0 i stvarna oznaka je 0

Zbroj svih elemenata matrice daje nam broj ukupnih primjera. Također, broj pozitivnih primjera dobiva se zbrajanjem *TP* i *FN* elemenata dok se broj negativnih primjera dobiva zbrajanjem *FP* i *TN* elemenata. Elementi na dijagonali matrice označuju točno klasificirane primjere. Dalje se nad takvom matricom može definirati niz mjera vrednovanja kao što su točnost (engl. *accuracy*), preciznost (engl. *precision*), odziv (engl. *recall*) itd. [29]

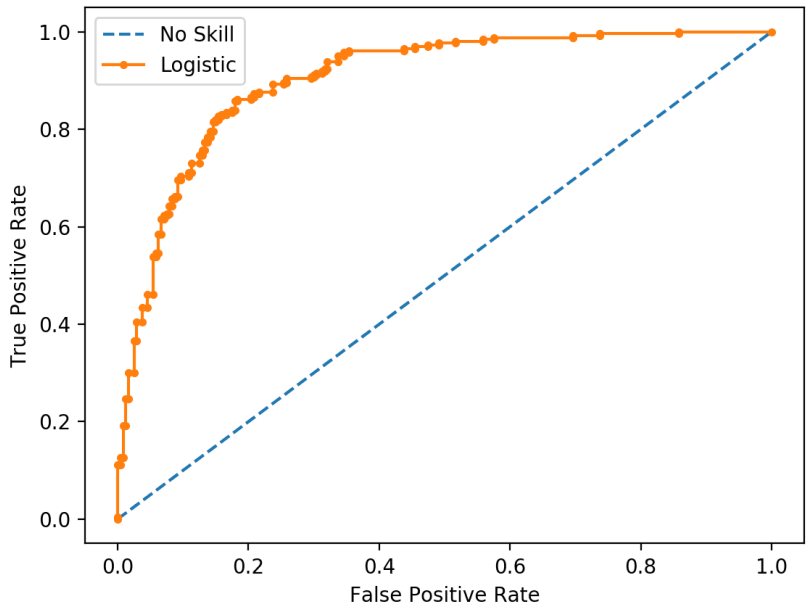
4.1.2.2. ROC krivulja

Navedene mjere točnosti modela su definirane za određeni prag klasifikacije. Većina klasifikatora procjenjuje vjerojatnost da neka vrijednost pripada određenoj klasi pri čemu je prag konvencionalno određen kao 0.5, odnosno ako je vjerojatnost veća ili jednaka 50% vrijednost će biti dodijeljena klasi. Prag se može proizvoljno mijenjati na druge vrijednosti, ali to opet daje evaluaciju modela samo na toj razini praga. Robusnija procjena je pregled klasifikacijskih metrika preko više pragova što se postiže korištenjem ROC krivulje. Skraćeno od engl. *receiver operating curve,* ROC je u osnovi separator dviju distribucija vjerojatnosti koje se preklapaju za pozitivne i negativne slučajeve. Krivulja predstavlja vrijednost stope stvarnih pozitiva (engl. *true positive rate, TPR*) kao funkciju stope lažnog alarma (engl. *false positive rate, FPR*) pri različitim vrijednostima praga. Navedene metrike se računaju prema sljedećim formulama :

$$TPR = \frac{TP}{TP + FN} \tag{15}$$

$$\boldsymbol{FPR = \frac{FP}{FP + TN}} \tag{16}$$

Frekvencije pojavljivanja stvarno pozitivnih (*TP*) i lažno pozitivnih (*FP*) slučajeva mijenjaju se kako se mijenja vrijednost praga. Smanjivanjem praga dobiva se više lažno pozitivnih slučajeva čime će i *FPR* rasti. Idealno je da krivulja raste strmo što znači da bi se vrijednost *FPR-a* trebala držati na niskoj razini [29].



Slika 37. ROC krivulja [30]

Plava, isprekidana linija na Slici 37. prikazuje ponašanje nasumičnog klasifikatora bez mogućnosti raspoznavanja. Drugim riječima, u takvom stanju klasifikator nije ništa bolji od nasumične vjerojatnosti. Ako se krivulja nalazi iznad te linije to upućuje da model radi bolje od nasumičnog dodjeljivanja klasa. Površina pod ROC krivuljom (engl. *area under ROC curve*), skraćeno AUC koja služi kao mjera vrednovanja modela gdje ukupna površina iznosi 1. Čim je površina ispod krivulje bliža tom iznosu to su performanse modela bolje [28]. Iz biblioteke *sklearn* uvedene su funkcije **roc_curve** i **roc_auc_score** pomoću kojih su u nastavku izrađene krivulje za različite algoritme.

4.2. Pretprocesiranje podataka

Prvi korak u pretprocesiranju je podjela podataka na ulazne i izlazne značajke. Ulazne značajke *X* su svi stupci u završnom skupu podataka osim stupca HeartDisease. Taj stupac uklonjen je korištenjem funkcije **drop()** dok za izlazne značajke *y* ostaje samo posljednji stupac skupa podataka.

```
In [10]: X = final_df.drop('HeartDisease',axis=1)
         y = final_df['HeartDisease']
```

Slika 38. Podjela ulaznih i izlaznih značajki

Za podjelu ulaznih i izlaznih značajki na skupove za treniranje i testiranje uporabljena je funkcija **train_test_split()**. Kao ulazni parametri funkcije upotrijebljene su ulazne značajke *X* i izlazne značajke *y*. Veličina skupa za testiranje postavljena je na vrijednost 0.3, odnosno 30 % podataka će biti iskorišteno isključivo za testiranje. Pomoću parametra **random_state** može se postići da podjela podataka ne bude nasumična nego da se za određenu brojčanu vrijednost podjela može vršiti na isti način svakim ponovnim pokretanjem koda. Nakon podjele, funkcijom **shape()** dobiven je oblik dvodimenzionalnih nizova skupova za treniranje i testiranje.

```
In [11]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

In [12]: X_train.shape, X_test.shape, y_train.shape, y_test.shape

Out[12]: ((642, 15), (276, 15), (642,), (276,))
```

Slika 39. Podjela na skupove za treniranje i testiranje

Skup značajki *X* potrebno je skalirati kako bi vrijednosti bile pogodne za sve algoritme. Izrađen je posebni objekt *scaler* pomoću funkcije **MinMaxScaler()** na kojem se kasnije dodaju funkcije **fit_transform()** i **transform()**. Korištenjem **fit_transform()** prvo se određuju potrebni parametri skupa podataka poput minimalne i maksimalne vrijednosti svake značajke. Nakon toga, podaci se transformiraju u raspon između 0 i 1.

```
In [13]: scaler = MinMaxScaler()

         X_train = scaler.fit_transform(X_train)

         X_test = scaler.transform(X_test)
```

Slika 40. Skaliranje X skupa značajki

4.3. Rad algoritama

Nakon završenog pretprocesiranja podataka, izrađeni su objekti za svakog pojedinog klasifikatora. Kod klasifikatora nasumična šuma parametar **n_estimators** određuje koliko će stabala biti stvoreno na kojima će se trenirati model. Povećanje broja stabala dobro djeluje sve do neke točke nakon koje se vrijeme treniranja modela znatno produžuje i dolazi do prenapučivanja modela (engl. o*verfitting*). Slično je i kod klasifikatora KNN gdje parametar **n_neighbors** određuje koliko će najbližih susjeda biti pri novom predviđanju.

```
In [15]: lr = LogisticRegression()
         dtree = DecisionTreeClassifier()
         rfc = RandomForestClassifier(n_estimators=200)
         svm = SVC(probability=True)
         knn = KNeighborsClassifier(n_neighbors=5)
```

Slika 41. Stvaranje pojedinog klasifikatora

Dalje je potrebno trenirati i ostvariti predviđanja na svim klasifikatorima. Za taj dio koda korištena je **for petlja** koja prolazi kroz listu **algoritmi**. Svaki klasifikator prvo se trenira pomoću naredbe **i.fit(X_train, y_train)**, a nakon toga naredba **i.predict(X_test)** ostvaruje predviđanja na testnom skupu podataka. Mjere vrednovanja, rezultat točnosti i logistički gubitak izračunati su i ispisani za svaki algoritam.

```
In [17]: algoritmi = [lr,dtree,rfc,svm,knn]
         ml_algoritmi = ['Logistička regresija','Stablo odlučivanja','Nasumična šuma','SVM','KNN']

         for i,j in zip(algoritmi,ml_algoritmi):
             i.fit(X_train,y_train)
             pred = i.predict(X_test)
             print(j,':\n')
             print('Rezultat točnosti:{:.2f}%'.format(accuracy_score(y_test,pred)*100))
             print('Log Loss:{:.2f}'.format(log_loss(y_test,pred)))
             print('='*40)
```

Slika 42. For petlja za treniranje i vrednovanje algoritama

Nakon što je petlja završena, dobiven je sljedeći ispis rezultata za svaki algoritam:

```
Logistička regresija :

Rezultat točnosti:87.68%
Log Loss:4.25
========================================
Stablo odlučivanja :

Rezultat točnosti:76.09%
Log Loss:8.26
========================================
Nasumična šuma :

Rezultat točnosti:88.04%
Log Loss:4.13
========================================
SVM :

Rezultat točnosti:85.87%
Log Loss:4.88
========================================
KNN :

Rezultat točnosti:85.51%
Log Loss:5.01
========================================
```

Slika 43. Rezultati točnosti i logistički gubitak algoritama

Vidljivo je iz ispisa da stablo odlučivanja naspram ostalih klasifikatora ima najmanju točnost i najveći iznos logističkog gubitka koji bi za dobar pokazatelj trebao biti čim nižeg iznosa. Ostali algoritmi su približni u rezultatima, stoga su u nastavku detaljno uspoređeni rezultati tri najbolja algoritma, a to su logistička regresija, nasumična šuma i SVM algoritam.

4.4. Usporedba performansi algoritama

4.4.1. Logistička regresija

Kao prva mjera vrednovanja modela odabrano je klasifikacijsko izvješće jer daje sažeti prikaz više različitih metrika poput točnosti, preciznosti, mjere F1 itd. Klasifikator se ponovno trenira pomoću funkcije **fit(),** a predviđanje se ostvaruje korištenjem funkcije **pred()**. Za stvaranje izvješća služi funkcija **classification_report()**.

```
In [19]: lr.fit(X_train,y_train)

         lr_pred = lr.predict(X_test)

         print('Klasifikacijsko izvješće za logističku regresiju :\n')
         print(classification_report(y_test,lr_pred))

         Klasifikacijsko izvješće za logističku regresiju :

                       precision    recall  f1-score   support

                    0       0.82      0.88      0.85       112
                    1       0.92      0.87      0.89       164

             accuracy                           0.88       276
            macro avg       0.87      0.88      0.87       276
         weighted avg       0.88      0.88      0.88       276
```

Slika 44. Izvješće metrika za logističku regresiju

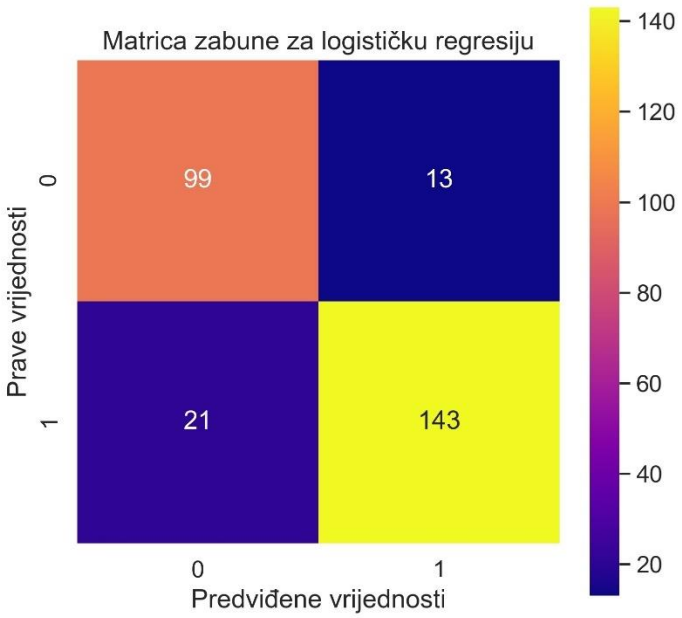
Metrike iz Slike 44. ukratko su opisane u nastavku.

- **Preciznost** (engl. *precision*): udio stvarno pozitivnih primjera (*TP*) u skupu svih primjera koje je klasifikator označio pozitivno (*TP*+*FP*), idealno jednaka je 1, odnosno svi primjeri koje je klasifikator označio pozitivno doista i jesu pozitivni.
- **Odziv** (engl. *recall*): udio stvarno pozitivnih primjera (*TP*) u skupu svih pozitivnih primjera (*TP*+*FN*), idealno jednak je 1, odnosno sve pozitivne primjere klasifikator će označiti kao takve. Drugi nazivi su **stopa stvarnih pozitiva** (engl. *true positive rate, TPR*) i **osjetljivost** (engl. *sensitivity*).
- **Mjera F1** (engl. *f1-score*): definirana je kao harmonijska sredina preciznosti i odziva, vrijednost mjere se nalazi u intervalu od 0 do 1. Rezultat se smatra boljim čim je vrijednost bliža iznosu 1.
- **Support** : broj stvarnih pojava klase u skupu podataka, jednako je zbroju stvarno pozitivnih primjera (*TP*) i lažno negativnih primjera (*FN*).

Kao daljnje mjere vrednovanja korištene su metrike koje se mogu i grafički prikazati poput matrice zabune i ROC krivulje. Matrica zabune izrađena je korištenjem funkcije **confusion_matrix()** u kojoj su za parametre dani testni podaci skupa *y* i njegove predviđene vrijednosti. Pomoću funkcija **figure()** i **set()** podešeni su veličina prikaza i font, a **heatmap()** funkcija služi da se pomoću intenziteta boja prikažu vrijednosti.

```
In [17]: lr_cm = confusion_matrix(y_test,lr_pred)


         plt.figure(figsize=(6, 6))
         sns.set(font_scale=1.2)

         sns.heatmap(lr_cm, annot=True, fmt='d', cmap='plasma', square=True,xticklabels=['0', '1'],yticklabels=['0', '1'])

         plt.xlabel('Predviđene vrijednosti')
         plt.ylabel('Prave vrijednosti')
         plt.title('Matrica zabune za logističku regresiju')
         plt.savefig('matrica_zabune_lr.jpg',dpi=300)
```

Slika 45. Stvaranje matrice zabune za logističku regresiju



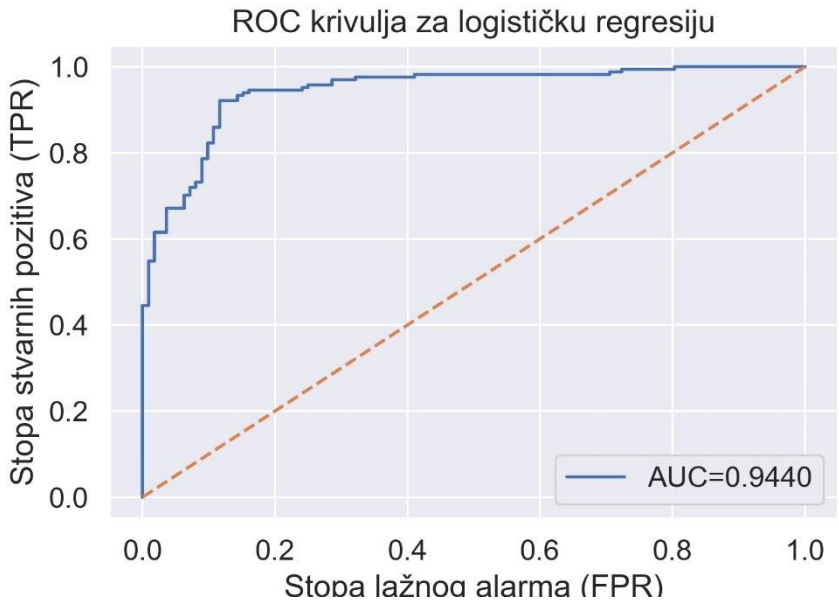
Slika 46. Matrica zabune za logističku regresiju

Vrijednosti na dijagonali matrice zabune pokazuju koliko je vrijednosti algoritam točno predvidio. Za 99 pacijenata koji nemaju srčano oboljenje točno je predvidio da ga nemaju (klasa 0), a za 143 koji ga stvarno imaju je točno predvidio da ga doista imaju (klasa 1). Suprotna dijagonala pokazuje koliko je algoritam pogriješio. Za 13 pacijenata predvidio je da imaju oboljenje dok ga u stvarnosti nemaju, a za 21 pacijenta previdio je da nemaju oboljenje dok ga u stvarnosti imaju. Matrice zabune različitih modela mogu se uspoređivati prema točno predviđenim vrijednostima. Model koji ima više točno predviđenih vrijednosti generalno ima prednost nad drugima.

Zadnja mjera vrednovanja koja je korištena i također se može grafički prikazati je ROC krivulja, odnosno površina pod ROC krivuljom, AUC. Prvi korak u izradi krivulje je predviđanje vjerojatnosti pozitivne klase (klasa 1) u skupu podataka ***X*_test**. Za to je korištena metoda **predict_proba()** koja vraća niz vjerojatnosti za svaku klasu, a dodavanjem **[:,1]** odvojene su vjerojatnosti za klasu 1. Potrebno je izraditi novi *dataframe* koji će predstavljati stvarne ciljane vrijednosti iz skupa ***y*_test**. Naredbom **np.array(y_test)** ***y*_test** skup se pretvara u NumPy niz kako bi tip podataka bio kompatibilan za izradu. Zatim je pomoću naredbe **pd.DataFrame()** izrađen *dataframe* sa jednim stupcem naziva **y actual** koji predstavlja stvarne ciljane vrijednosti iz testnog skupa. Varijabla **lr_pred_prob** sadrži predviđene vjerojatnosti za pozitivnu klasu 1 dobivene iz modela logističke regresije i korištenjem **pd.DataFrame()** dobiven je drugi *dataframe* koji sadrži stupac **y pred prob**. U njemu su sadržane predviđene vjerojatnosti za pozitivnu klasu. Ta dva stupca spojena su u jedan *dataframe* pomoću funkcije **pd.concat()** i parametra **axis=1**. Na kraju, kako bi bilo lakše usporediti stvarne i predviđene vrijednosti korištena je funkcija **index()** koja indekse novog *dataframe-a* postavlja da budu jednaki kao indeksi originalnog skupa **y_test**. Računanje parametara krivulje odrađeno je pomoću funkcije **roc_curve()** koja daje vrijednosti stope lažnog alarma (*FPR*), stope stvarnih pozitiva (*TPR*) i pragova (engl. *thresholds*). Površinu ispod ROC krivulje (AUC) izračunava funkcija **roc_auc_score**.

```
In [20]: lr_pred_prob = lr.predict_proba(X_test)[:][:,1]

         lr_actual_predict = pd.concat([pd.DataFrame(np.array(y_test),columns=['y actual']),
                                        pd.DataFrame(lr_pred_prob,columns=['y pred prob'])],axis=1)
         lr_actual_predict.index = y_test.index

         fpr, tpr, tr = roc_curve(lr_actual_predict['y actual'],lr_actual_predict['y pred prob'])
         auc = roc_auc_score(lr_actual_predict['y actual'],lr_actual_predict['y pred prob'])

         plt.plot(fpr,tpr, label='AUC=%.4f'%auc)
         plt.plot(fpr,fpr,linestyle='--')
         plt.xlabel('Stopa lažnog alarma (FPR)')
         plt.ylabel('Stopa stvarnih pozitiva (TPR)')
         plt.title('ROC krivulja za logističku regresiju')
         plt.legend()
         plt.savefig('roc1_lr.jpg',dpi=300)
```

Slika 47. Stvaranje ROC krivulje za logističku regresiju



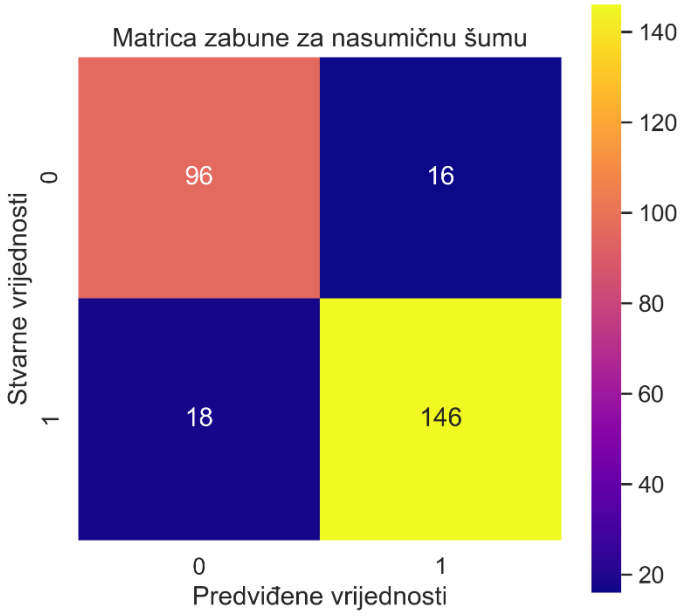
Slika 48. ROC krivulja za logističku regresiju

4.4.2. Nasumična šuma

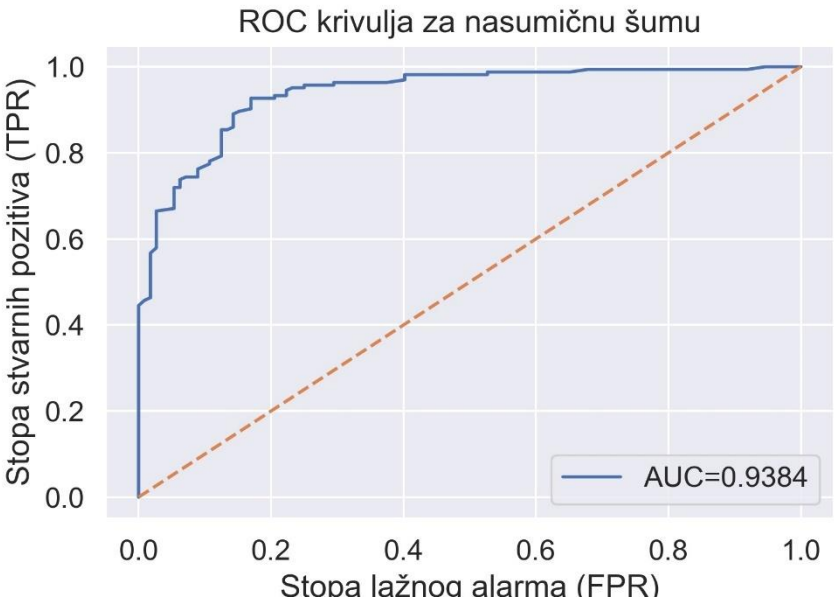
Iste mjere vrednovanja primijenjene su i na ostale modele. Dalje su priložene metrike, matrica zabune i ROC krivulja za algoritam nasumična šuma.

```
In [24]: rfc.fit(X_train,y_train)

         rfc_pred = rfc.predict(X_test)

         print('Klasifikacijsko izvješće za nasumičnu šumu :\n')
         print(classification_report(y_test,rfc_pred))

         Klasifikacijsko izvješće za nasumičnu šumu :

                       precision    recall  f1-score   support

                    0       0.84      0.86      0.85       112
                    1       0.90      0.89      0.90       164

             accuracy                           0.88       276
            macro avg       0.87      0.87      0.87       276
         weighted avg       0.88      0.88      0.88       276
```

Slika 49. Izvješće metrika za nasumičnu šumu



Slika 50. Matrica zabune za nasumičnu šumu



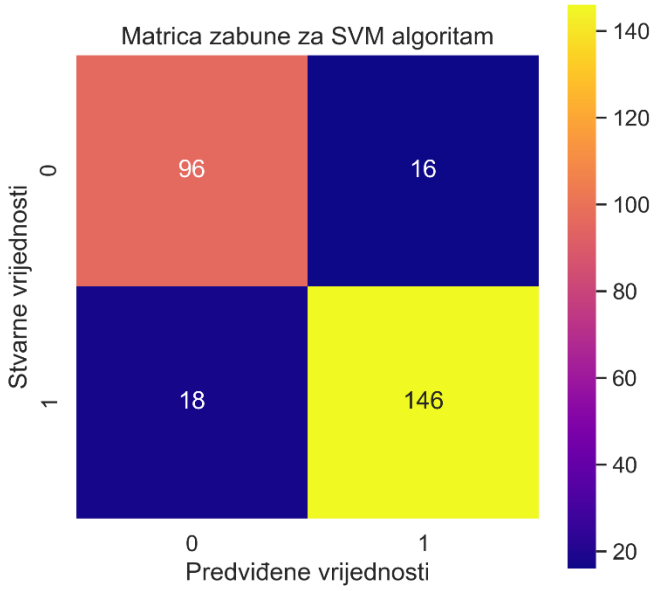
Slika 51. ROC krivulja za nasumičnu šumu

4.4.3. Metoda potpornih vektora (SVM)

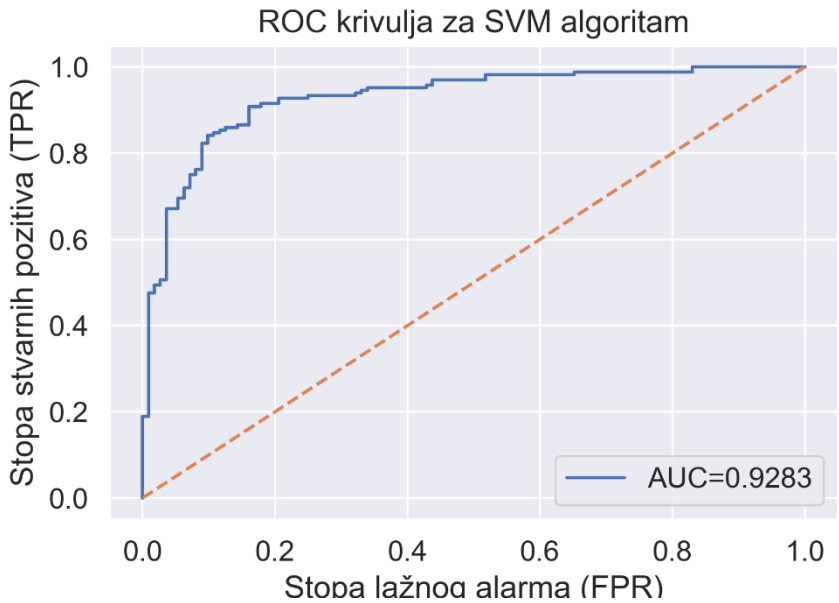
Zadnja metoda koja sudjeluje u usporedbi je metoda potpornih vektora, skraćeno SVM. Svi prikazi izrađeni su identično na način kao za dvije prethodne metode uz izmjenu varijabli u kodu za trenutnu metodu.

```
In [27]: svm.fit(X_train,y_train)

         svm_pred = svm.predict(X_test)

         print('Klasifikacijsko izvješće za SVM :\n')
         print(classification_report(y_test,svm_pred))

         Klasifikacijsko izvješće za SVM :

                       precision    recall  f1-score   support

                    0       0.81      0.85      0.83       112
                    1       0.89      0.87      0.88       164

             accuracy                           0.86       276
            macro avg       0.85      0.86      0.85       276
         weighted avg       0.86      0.86      0.86       276
```

Slika 52. Izvješće metrika za SVM algoritam



Slika 53. Matrica zabune za SVM algoritam



Slika 54. ROC krivulja za SVM algoritam

4.4.4. Rezultat usporedbe

Na temelju svih dobivenih vrijednosti mjera vrednovanja, najbolja od obrađenih metoda je nasumična šuma (engl. *random forest*). Od svih metoda ima najveći rezultat točnosti (88.04 %) i najniži iznos logističkog gubitka (4.13). Vrijednosti elemenata u matrici zabune dobivene su identično kao i kod SVM algoritma i dobiven je zadovoljavajući iznos AUC vrijednosti (0.9384). Točnost modela bi se dalje mogla povisiti korištenjem tehnika poput hiper-parametarskog podešavanja gdje se izvlače najbolji parametri modela. Također, u ovom primjeru radilo se s 15 značajki što je dosta veliki broj za treniranje modela i time direktno utječe na rezultate metrika.

5. ZAKLJUČAK

Porastom stope kardiovaskularnih bolesti, pravovremeno dijagnosticiranje ključno je za daljnje i uspješno liječenje. Naglim razvojem umjetne inteligencije, a time i strojnog učenja omogućeno je razvijanje modela koji će razvoj ovakvih i sličnih bolesti moći uspješno predviđati. U ovom radu dan je pregled i dio teorije iza algoritama koji služe za probleme klasifikacije na temelju skupa podataka koji opisuju simptome i faktore kod srčanih oboljenja. Objašnjeni su algoritmi logistička regresija, stablo odlučivanja, nasumična šuma, metoda potpornih vektora i k-najbližih susjeda. Prikazan je proces vizualizacije skupa podataka kako bi se jasnije vidjelo kakav je međusobni utjecaj različitih varijabli. Kod vizualizacije korišteni su razni dijagrami i prikazi poput histograma, dijagrama raspršenja, kutijastih dijagrama itd. Također, odrađeno je pretprocesiranje podataka pomoću čega su kategoričke varijable enkodirane kako bi bile u obliku koji je potreban za implementaciju algoritama. Nakon faze učenja, rezultati algoritama uspoređeni su metrikama koje se koriste za vrednovanje modela strojnog učenja poput točnosti, preciznosti, logističkog gubitka, matrica zabune i ROC krivulja. Rezultati su pokazali da je najuspješniji algoritam u ovom zadatku bio nasumična šuma (engl. *random forest*).

6. LITERATURA

[1] „Cardiovascular diseases“, https://www.who.int/health-topics/cardiovascular-diseases#tab=tab_1, pristupljeno: 02. veljače 2024.

[2] G. Rebala, A. Ravi, i S. Churiwala, *An Introduction to Machine Learning*. Springer International Publishing, 2019.

[3] O. Campesato, *Artificial intelligence, machine learning, and deep learning*. Mercury Learning and Information, 2020.

[4] V. Nasteski, „An overview of the supervised machine learning methods“, *Horizons. b*, sv. 4, str. 51–62, 2017.

[5] „Supervised and Unsupervised Learning (an Intuitive Approach) | Medium“, https://medium.com/@metehankozan/supervised-and-unsupervised-learning-an-intuitive-approach-cd8f8f64b644, pristupljeno: 02. siječnja 2024.

[6] D. Michie, D. J. Spiegelhalter, i C. C. Taylor, „Machine learning, neural and statistical classification“, 1994.

[7] „Classification in Machine Learning: An Introduction | Built In“, https://builtin.com/machine-learning/classification-machine-learning, pristupljeno: pristupljeno: 03. siječnja 2024.

[8] „Regression algorithms. Regression algorithms are a type of… | by Arun Kumar Pandey (Ph.D.) | Medium“, https://medium.com/@arunp77/regression-algorithms-29f112797724, pristupljeno: 03. siječnja 2024.

[9] „What is Unsupervised Learning? | Definition from TechTarget“, https://www.techtarget.com/searchenterpriseai/definition/unsupervised-learning, pristupljeno: 05. siječnja 2024.

[10] „Logistic Regression: Equation, Assumptions, Types, and Best Practices“, https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/, pristupljeno: 06. siječnja 2024.

[11] „Sigmoid function - Wikipedia“, https://en.wikipedia.org/wiki/Sigmoid_function, pristupljeno: 06. siječnja 2024.

[12] J. Daniel i J. H. Martin, „Speech and Language Processing“, 2023.

[13] „Random Forest Algorithm for Machine Learning | by Madison Schott | Capital One Tech | Medium“, https://medium.com/capital-one-tech/random-forest-algorithm-for-machine-learning-c4b2c8cc9feb, pristupljeno: 06. siječnja 2024.

[14] „The decision tree scheme. | Download Scientific Diagram“., https://www.researchgate.net/figure/The-decision-tree-scheme_fig4_371645234, pristupljeno: 06. siječnja 2024.

[15] „Decision Tree Algorithm, Explained - KDnuggets“, https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html, pristupljeno: 07. siječnja 2024.

[16] „A Complete Guide to Decision Tree Split using Information Gain“, https://analyticsindiamag.com/a-complete-guide-to-decision-tree-split-using-information-gain/, pristupljeno: 07. siječnja 2024.

[17] A. A. Perspective, „Chapman & Hall/CRC Machine Learning & Pattern Recognition Series Chapman & Hall/CRC Machine Learning & Pattern Recognition Series Machine Learning“

[18] „(PDF) Comparing throat and acoustic microphones for laryngeal pathology detection from human voice“, https://www.researchgate.net/publication/289191953_Comparing_throat_and_acoustic_microphones_for_laryngeal_pathology_detection_from_human_voice/figures?lo=1, pristupljeno: 09. siječnja 2024.

[19] M. Hossain, *Support Vector Machine**. 2022.
[20] „Support Vector Machine (SVM) Explained - MATLAB & Simulink“, https://www.mathworks.com/discovery/support-vector-machine.html, pristupljeno: 10. siječnja 2024.
[21] „K-Nearest Neighbor(KNN) Algorithm - GeeksforGeeks“, https://www.geeksforgeeks.org/k-nearest-neighbours/, pristupljeno: 04. veljače 2024.
[22] „What is Python? Executive Summary | Python.org“, https://www.python.org/doc/essays/blurb/, pristupljeno 17. siječnja 2024.
[23] „What is NumPy? — NumPy v1.26 Manual“, https://numpy.org/doc/stable/user/whatisnumpy.html, pristupljeno: 17. siječnja 2024.
[24] „Package overview — pandas 2.1.4 documentation“, https://pandas.pydata.org/docs/getting_started/overview.html, pristupljeno: 17. siječnja 2024.
[25] „An introduction to seaborn — seaborn 0.13.1 documentation“, https://seaborn.pydata.org/tutorial/introduction.html, pristupljeno: 17. siječnja 2024
[26] „korelacija - Hrvatska enciklopedija“, https://www.enciklopedija.hr/clanak/korelacija, pristupljeno: 18. siječnja 2024.
[27] „Data Science in 5 Minutes: What is One Hot Encoding?“, https://www.educative.io/blog/one-hot-encoding, pristupljeno: 21. siječnja 2024.
[28] D. L. Streiner i J. Cairney, „What’s under the ROC? An introduction to receiver operating characteristics curves“, *Canadian Journal of Psychiatry*, sv. 52, izd. 2, str. 121–128, 2007.
[29] Jan Šnajder, predavanja iz kolegija Strojno učenje 1, Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu, 2022./2023.
[30] „How to Use ROC Curves and Precision-Recall Curves for Classification in Python“, https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/, pristupljeno: 31. siječnja 2024.

7. PRILOZI

Programski kod u Python programskom jeziku

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

# učitavanje skupa podataka
df = pd.read_csv('heart.csv')

# očitavanje prvih 5 redaka skupa podataka
df.head()

# očitavanje zadnjih 5 redaka skupa podataka
df.tail()

# informacije o dataframe-u
df.info()

# opis skupa podataka
df.describe()

# opis skupa podataka uključujući objekt tip podataka
df.describe(include='object')

# jedinstvene vrijednosti svakog stupca
df.nunique()

# provjera null vrijednosti
df.isna().sum()

# korelacija između značajki
df.corr()


grupno_cpt = df.groupby('ChestPainType')
grupno_cpt.mean()


grupno_ECG = df.groupby('RestingECG')
grupno_ECG.mean()
```

```
# histogram skupa podataka
sns.pairplot(df,hue='HeartDisease')

# histogram distribucije značajki
df.hist(figsize=(13,16),bins=60)
plt.suptitle('Distribucija
značajki',x=0.5,y=1.02,ha='center',fontsize='large')
plt.tight_layout()
plt.savefig('histogram_plot.jpg',dpi=300)

# kutijasti dijagram svih značajki
df.boxplot(figsize=(12,8))
plt.suptitle('Kutijasti dijagram
značajki',x=0.5,y=1.02,ha='center',fontsize='large')
plt.tight_layout()
plt.savefig('box_plot.jpg',dpi=300)

# kutijasti dijagrami distribucije značajki
plt.figure(figsize=(15,15))

plt.subplot(3,2,1)
sns.boxplot(x=df['HeartDisease'],y=df['Age'])
plt.title('Dob')

plt.subplot(3,2,2)
sns.boxplot(x=df['HeartDisease'],y=df['RestingBP'])
plt.title('Krvni pritisak')

plt.subplot(3,2,3)
sns.boxplot(x=df['HeartDisease'],y=df['Cholesterol'])
plt.title('Razina kolesterola')

plt.subplot(3,2,4)
sns.boxplot(x=df['HeartDisease'],y=df['MaxHR'])
plt.title('Maksimalni otkucaji srca')

plt.subplot(3,2,5)
sns.boxplot(x=df['HeartDisease'],y=df['Oldpeak'])
plt.title('Depresija ST segmenta')

plt.tight_layout()
plt.savefig('box_plot_bi.jpg',dpi=300)

# dijagram rasprešenja dob/max.otkucaji
plt.figure(figsize=(10,8))
sns.scatterplot(data=df,x='Age',y='MaxHR',hue='HeartDisease',palette='S
et1')
plt.savefig('scatter_bi.jpg',dpi=300)
```

```
# dijagram raspršenja dob/krvni tlak
plt.figure(figsize=(10,8))
sns.scatterplot(data=df,x='Age',y='RestingBP',hue='HeartDisease',palett
e='Set1')
plt.savefig('scatter2_bi.jpg',dpi=300)

# dijagram raspršenja dob/kolesterol
plt.figure(figsize=(10,8))
sns.scatterplot(data=df,x='Age',y='Cholesterol',hue='HeartDisease',pale
tte='Set1')
plt.savefig('scatter3_bi.jpg',dpi=300)

# one-hot encoding
enkodirano_spol = pd.get_dummies(df,columns=['Sex'],drop_first=True)
enkodirano_spol.head()

enkodirano_bol =
pd.get_dummies(enkodirano_spol,columns=['ChestPainType'],drop_first=Tru
e)
enkodirano_bol.head()

enkodirano_ekg =
pd.get_dummies(enkodirano_bol,columns=['RestingECG'],drop_first=True)
enkodirano_ekg.head()

enkodirano_ang =
pd.get_dummies(enkodirano_ekg,columns=['ExerciseAngina'],drop_first=Tru
e)
enkodirano_ang.head()

final_df =
pd.get_dummies(enkodirano_ang,columns=['ST_Slope'],drop_first=True)
final_df.head()

final_df.rename(columns={'Sex_M':'Sex',
'ExerciseAngina_Y':'ExerciseAngina'},inplace=True)
final_df.columns


final_df.info()

# grafički prikaz korelacijskih faktora(nakon enkodiranja)
final_df_corr = final_df.corr()

plt.figure(figsize=(15,12))

plt.title('Grafički prikaz korelacijskih faktora (nakon enkodiranja)')

sns.heatmap(final_df_corr,annot=True,linecolor='white',linewidths=0.2)
```

```
plt.savefig('heatmap',dpi=300)


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score,log_loss
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import roc_curve,roc_auc_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import CategoricalNB
import shap

X = final_df.drop('HeartDisease',axis=1)
y = final_df['HeartDisease']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

X_train.shape, X_test.shape, y_train.shape, y_test.shape

scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

lr = LogisticRegression()
dtree = DecisionTreeClassifier()
rfc = RandomForestClassifier(n_estimators=200)
svm = SVC(probability=True)
knn = KNeighborsClassifier(n_neighbors=5)

algoritmi = [lr,dtree,rfc,svm,knn]
ml_algoritmi = ['Logistička regresija','Stablo odlučivanja','Nasumična
šuma','SVM','KNN']

for i,j in zip(algoritmi,ml_algoritmi):
    i.fit(X_train,y_train)
    pred = i.predict(X_test)
    print(j,':\n')
    print('Rezultat
točnosti:{:.2f}%'.format(accuracy_score(y_test,pred)*100))
    print('Log Loss:{:.2f}'.format(log_loss(y_test,pred)))
    print('='*40)
```

```
# klasifikacijsko izvješće-log.reg
lr.fit(X_train,y_train)
lr_pred = lr.predict(X_test)
print('Klasifikacijsko izvješće za logističku regresiju :\n')
print(classification_report(y_test,lr_pred))

# matrica zabune-log.reg
lr_cm = confusion_matrix(y_test,lr_pred)
plt.figure(figsize=(6, 6))
sns.set(font_scale=1.2)

sns.heatmap(lr_cm, annot=True, fmt='d', cmap='plasma',
square=True,xticklabels=['0', '1'],yticklabels=['0', '1'])

plt.xlabel('Predviđene vrijednosti')
plt.ylabel('Prave vrijednosti')
plt.title('Matrica zabune za logističku regresiju')
plt.savefig('matrica_zabune_lr.jpg',dpi=300)

# ROC krivulja-log.reg
lr_pred_prob = lr.predict_proba(X_test)[:][:,1]

lr_actual_predict =
pd.concat([pd.DataFrame(np.array(y_test),columns=['y actual']),
                                pd.DataFrame(lr_pred_prob,columns=['y
pred prob'])],axis=1)
lr_actual_predict.index = y_test.index

fpr, tpr, tr = roc_curve(lr_actual_predict['y
actual'],lr_actual_predict['y pred prob'])
auc = roc_auc_score(lr_actual_predict['y actual'],lr_actual_predict['y
pred prob'])

plt.plot(fpr,tpr, label='AUC=%.4f'%auc)
plt.plot(fpr,fpr,linestyle='--')
plt.xlabel('Stopa lažnog alarma (FPR)')
plt.ylabel('Stopa stvarnih pozitiva (TPR)')
plt.title('ROC krivulja za logističku regresiju')
plt.legend()
plt.savefig('roc_lr.jpg',dpi=300)

# klasifikacijsko izvješće-nasumična šuma
rfc.fit(X_train,y_train)
rfc_pred = rfc.predict(X_test)
print('Klasifikacijsko izvješće za nasumičnu šumu :\n')
print(classification_report(y_test,rfc_pred))

# matrica zabune-nasumična šuma
```

```
rfc_cm = confusion_matrix(y_test,rfc_pred)
plt.figure(figsize=(6, 6))
sns.set(font_scale=1.2)

sns.heatmap(rfc_cm, annot=True, fmt='d', cmap='plasma',
square=True,xticklabels=['0', '1'],yticklabels=['0', '1'])

plt.xlabel('Predviđene vrijednosti')
plt.ylabel('Stvarne vrijednosti')
plt.title('Matrica zabune za nasumičnu šumu')
plt.savefig('matrica_zabune_rfc',dpi=300)

# ROC krivulja - nasumična šuma
rfc_pred_prob = rfc.predict_proba(X_test)[:][:,1]

rfc_actual_predict =
pd.concat([pd.DataFrame(np.array(y_test),columns=['y actual']),
                                pd.DataFrame(rfc_pred_prob,columns=['y
pred prob'])],axis=1)
rfc_actual_predict.index = y_test.index

fpr, tpr, tr = roc_curve(rfc_actual_predict['y
actual'],rfc_actual_predict['y pred prob'])
auc = roc_auc_score(rfc_actual_predict['y
actual'],rfc_actual_predict['y pred prob'])

plt.plot(fpr,tpr, label='AUC=%.4f'%auc)
plt.plot(fpr,fpr,linestyle='--')
plt.xlabel('Stopa lažnog alarma (FPR)')
plt.ylabel('Stopa stvarnih pozitiva (TPR)')
plt.title('ROC krivulja za nasumičnu šumu')
plt.legend()
plt.savefig('roc_rfc.jpg',dpi=300)

# klasifikacijsko izvješće-SVM
svm.fit(X_train,y_train)
svm_pred = svm.predict(X_test)
print('Klasifikacijsko izvješće za SVM :\n')
print(classification_report(y_test,svm_pred))

# matrica zabune-SVM
svm_cm = confusion_matrix(y_test,rfc_pred)
plt.figure(figsize=(6, 6))
sns.set(font_scale=1.2)

sns.heatmap(svm_cm, annot=True, fmt='d', cmap='plasma',
square=True,xticklabels=['0', '1'],yticklabels=['0', '1'])

plt.xlabel('Predviđene vrijednosti')
```

```
plt.ylabel('Stvarne vrijednosti')
plt.title('Matrica zabune za SVM algoritam')
plt.savefig('matrica_zabune_svm',dpi=300)

# ROC krivulja-SVM
svm_pred_prob = svm.predict_proba(X_test)[:][:,1]

svm_actual_predict =
pd.concat([pd.DataFrame(np.array(y_test),columns=['y actual']),
                                pd.DataFrame(svm_pred_prob,columns=['y
pred prob'])],axis=1)
svm_actual_predict.index = y_test.index

fpr, tpr, tr = roc_curve(svm_actual_predict['y
actual'],svm_actual_predict['y pred prob'])
auc = roc_auc_score(svm_actual_predict['y
actual'],svm_actual_predict['y pred prob'])

plt.plot(fpr,tpr, label='AUC=%.4f'%auc)
plt.plot(fpr,fpr,linestyle='--')
plt.xlabel('Stopa lažnog alarma (FPR)')
plt.ylabel('Stopa stvarnih pozitiva (TPR)')
plt.title('ROC krivulja za SVM algoritam')
plt.legend()
plt.savefig('roc_svm',dpi=300)
```