SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Klasifikacija pacijenata na temelju mogućnosti oboljenja od bolesti srca

Josip Rizner

Osijek, 2022.

SADRŽAJ

1. UVOD 3

2. TEORETSKE OSNOVE KORIŠTENIH MODELA 4

2.1. Logistička regresija 4

2.2. Nasumične šume odlučivanja 5

2.3. Stroj potpornih vektora 6

2.4. Gaussov Naivni Bayes klasifikator 7

2.5. K-najbližih susjeda 7

3. OBRADA PODATAKA 9

3.1. Opis izvornih podataka 9

3.2. Obrada izvornih podataka 10

4. PREGLED PROGRAMSKOG KODA KORIŠTENOG ZA UČENJE I TESTIRANJE MODELA 13

5. ANALIZA REZULTATA 14

5.1. Logistička regresija 14

5.2. Nasumične šume odlučivanja 15

5.3. Stroj potpornih vektora 15

5.4. Gaussov Naivni Bayes klasifikator 16

5.5. K-najbližih susjeda 16

6. IMPLEMENTACIJA APLIKACIJE 18

7. ZAKLJUČAK 19

8. LITERATURA 20

9. POPIS SLIKA 21

1. UVOD

Bolesti srca i krvnih žila najčešći su uzrok smrti u Hrvatskoj, a među njima, na prvom je mjestu ishemijska bolest srca. Prema statističkim podacima svaki drugi građanin Hrvatske umire zbog bolesti srca i krvnih žila. Uz ishemijsku bolest srca, među 10 vodećih uzroka smrti u Hrvatskoj su još cerebrovaskularne bolesti te zatajenje srca.

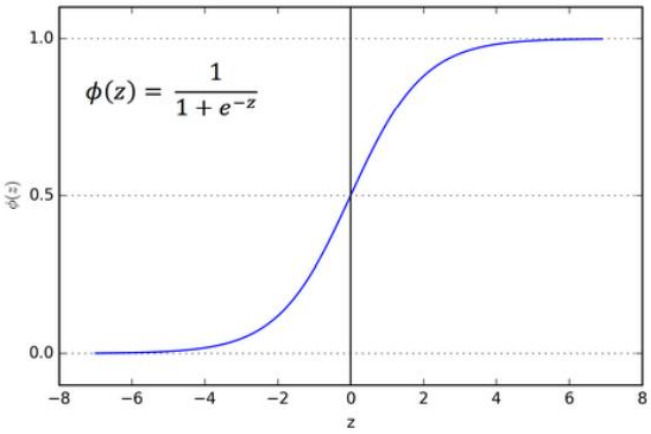
U ovom projektnom zadatku pristupa se problemu klasifikacije pacijenata na temelju mogućnosti oboljenja od bolesti srca pomoću pet modela. To su modeli : stroj potpornih vektora, k najbližih susjeda, logistička regresija, nasumične šume odlučivanja i Gaussov naivni Bayes. Za svaki model provedena je analiza uspješnosti i rezultati modela su uspoređeni. Isto tako, kreirano je korisničko sučelje za pružanje novih podataka modelima u svrhu vršenja predviđanja, a implementirano je u obliku Streamlit aplikacije.

Na početku ovog pisanog rada sažeto su objašnjene teoretske osnove svakog modela i prikazani su koraci koji su napravljeni prije samog rješavanja problema, što uključuje pregled podataka i obradu tih podataka korištenjem programskog jezika Python. Nakon obrade podataka, korištenjem istog programskog jezika i biblioteke scikit-learn, spomenuti modeli primjenjeni su na podatke, točnije, naučeni su na dobivenom skupu za učenje i ispitani na skupu za ispitivanje. Na kraju, rezultati se analiziraju te se testira implementirana aplikacija.

2. TEORETSKE OSNOVE KORIŠTENIH MODELA

2.1. Logistička regresija

Logistička regresija klasifikacijski je algoritam, ali se naziva regresija jer se koriste jako slične tehnike kao kod linearne regresije. Jedan je od najšire korištenih algoritama u strojnom učenju. Pri takvoj klasifikaciji, izlazne su vrijednosti diskretne (y=0 ili y=1), a vrijednost modela $0 \leq h\theta(x) \leq 1$. Logistička može se koristiti i za višeklasnu klasifikaciju. Kako bi definirali model, prvo je potrebno definirati aktivacijsku funkciju koja se naziva sigmoidna ili logistička (logistic) funkcija. Njezina definicija može se vidjeti na slici 2.1.



Slika 2.1. *Logistička funkcija*

Tri karakteristike koje sigmoidu čine dobrim odabirom za aktivacijsku funkciju: funkcija „gnječi“ izlaz na interval od 0 do 1, oblikom je slična funkciji praga i funkcija je derivabilna na cijeloj domeni. Iz navedenog se dolazi do modela logističke regresije koji se može vidjeti na slici 2.2.

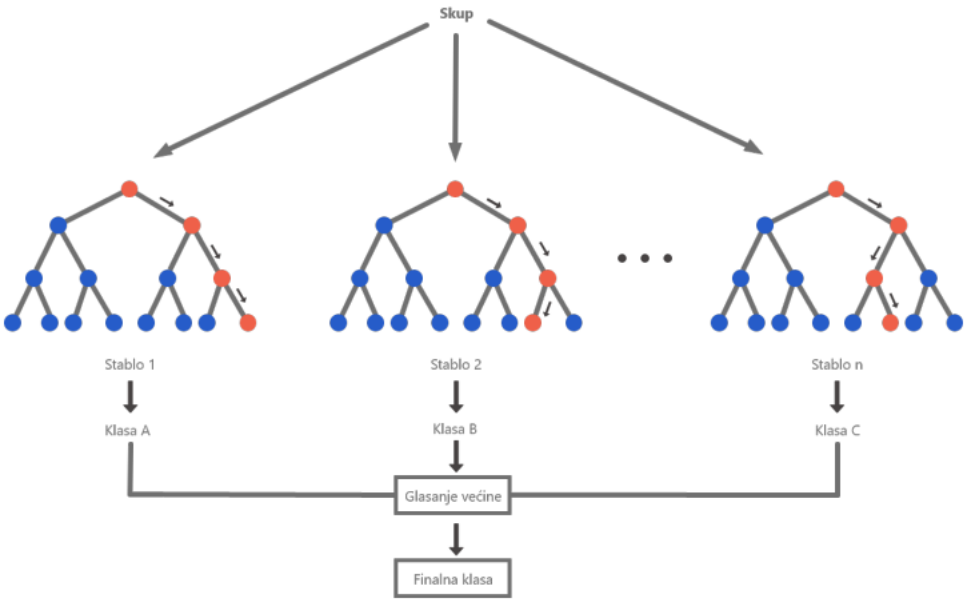
$$h(\mathbf{x};\mathbf{w}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})\right) = \frac{1}{1+\exp(-\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}))}$$

Slika 2.2. *Model logističke regresije*

σ predstavlja aktivacijsku funkciju omotanu oko linearnog modela regresije, odnosno, predstavlja sigmoidu. Vektor w je vektor parametara ili težina, a Φ(x) je funkcija preslikavanja koja kao argument ima primjer x. Klasifikacija se obavlja u prostoru gdje se nalazi granica, to jest, gdje se nalazi linearni model regresije. On predstavlja granicu i ovisno na kojoj strani granice se nalazi primjer spadat će u određenu klasu. Također, udaljenost od same granice, predstavlja sigurnost modela u odabir klase koju je odabrao za odgovarajući primjer.

2.2. Nasumične šume odlučivanja

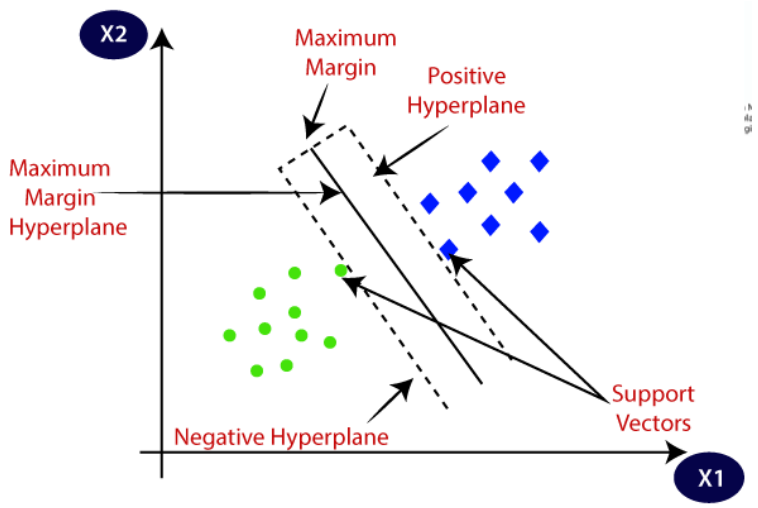
Algoritam nasumičnih šuma gradi veliki broj nekoreliranih stabala. Pri klasifikaciji stabla predstavljaju glasove te algoritam očitava najmnogobrojniji glas, dok pri regresiji algoritam pronalazi prosječnu vrijednost. Algoritam nasumičnih šuma jednostavan je za učenje i podešavanje. Kao posljedica, slučajne šume su jako popularan algoritam. Prednosti ovog algoritma su učinkovitost, a nedostatci su sklonost pretjeranoj prilagodbi, pogotovo ako u podacima ima šuma.



Slika 2.3. *Nasumične šume odlučivanja*

2.3. Stroj potpornih vektora

Support Vector Machines je skup nadziranih metoda učenja koje se koriste za klasifkaciju, regresiju i vanjsko otkrivanje (engl. *outlier detection*). Cilj ove metode je odvajanje pozitivnih primjera od negativnih primjera s maksimalnom granicom, a granica je udaljenost od hiperravnine do najbližeg pozitivnog ili negativnog primjera, tj. cilj je pronaći hiperravninu u n-dimenzionalnom prostoru koji jasno razdvaja podatkovne točke. Ti primjeri nazivaju se potporni vektori (engl. *support vectors*) . Ovi algoritmi koriste teoriju minimalizacije strukturnog rizika. Algoritmi bazirani na SVM upotrebljavaju se za klasifikaciju nekoliko primjera u dvije različite klase. Ovaj algoritam nalazi hiperravninu između te dvije klase tako da odvajanje granica između te dvije klase postaje maksimalno. Klasifikacija testnog primjera ovisi o strani hiperravnine, odnosno strani gdje se testni primjer nalazi. Ulazne značajke mogu se preslikati i u prostor visokih dimenzija, ali u tom slučaju, za smanjenje računskih troškova obuke i postupka ispitivanja u prostoru visokih dimenzija, koriste se neke funkcije kernela.



Slika 2.4. *Stroj potpornih vektora*

2.4. Gaussov Naivni Bayes klasifikator

Naivni Bayes-ovi klasifikatori spadaju u obitelj jednostavnih vjerojatnosnih klasifikatora zasnovanih na Bayesovom teoremu sa strogim pretpostavkama neovisnosti među značajkama. Istaknuta slika (značajka) je izraz – s P(A|B) kao posteriornom vjerojatnošću, P(B|A) s likelihood, P(A) je razred priorne vjerojatnosti, a P(B) je prediktor priorne vjerojatnosti.

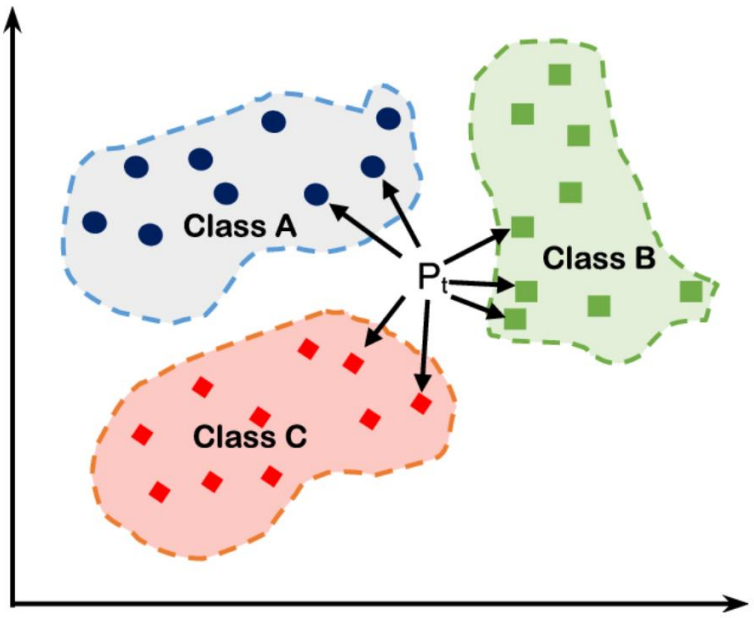
$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

Slika 2.5. *Naive Bayes - formula*

Naive Bayes klasifikatori pokazali su vrlo dobrima u stvarnim situacijama, kao što je klasifikacija dokumenata i filtriranje neželjene pošte. Zahtijevaju malu količinu podataka u skupu za učenje za procjenu potrebnih parametara. Vrlo je jednostavan algoritam za implementaciju i dobri rezultati su postignuti u većini slučajeva. Klasifikatori Naive Bayesa mogu biti iznimno brzi u usporedbi sa sofisticiranijim metodama. Odvajanje raspodjele klasnih uvjetnih značajki znači da se svaka distribucija može samostalno procijeniti kao jednodimenzionalna distribucija.

2.5. K-najbližih susjeda

Algoritam k-najbližih susjeda je jedan od jednostavnijih algoritama za razumijevanje i implementaciju, te se kao takav često koristi. Kada se ispituje klasa nekog novog podatka, tada algoritam provjerava postojeće kategorizirane podatke, te traži k onih koji su mu najsličniji po ulaznim parametrima. Zatim algoritam gleda kojoj klasi pripada najviše susjednih podataka, te na temelju toga dodjeljuje tu kategoriju novom podatku.



Slika 2.6. *K-najbližih susjeda*

3. OBRADA PODATAKA

3.1. Opis izvornih podataka

Podaci su preuzeti sa stranice UCI Machine Learning repository [1]. Podaci o dijagnozi bolesti srca prikupljeni su na četiri mjesta, a to su:

- Klinika u Clevelandu
- Mađarski institut za kardiologiju, Budimpešta
- V.A. Medicinski centar, Long Beach, CA
- Sveučilišna bolnica, Zürich, Švicarska

Podaci originalno imaju 76 atributa, ali samo je 14 zapravo iskoristivo te se svi prethodno objavljeni eksperimenti s ovim skupom podataka baziraju na tih 14 atributa. Svi atributi imaju numeričku vrijednost. Koriste se sljedeći atributi:

- Dob pacijenta
- Spol pacijenta
- Tip boli u prsima
- Krvni tlak prilikom mirovanja [mm Hg]
- Kolesterol [mg/dl]
- Šećer u krvi natašte
- Rezultati mjerenja elektrokardiografije prilikom mirovanja
- Postignut maksimalni broj otkucaja srca
- Angina izazvana vježbanjem
- ST depresija izazvana vježbanjem u odnosu na mirovanje
- ST segment – nagib prilikom vježbe
- Broj velikih žila obojenih fluoroskopijom
- Oblik talasemije
- Dijagnoza bolesti srca (atribut koji se predviđa)

U izvornim podacima ima dosta zapisa u kojem nedostaju stupci odnosno atributi. Izgled izvornih podataka može se vidjeti na slici 3.1.

	Age	Sex	Chest pain type	Resting	Serum	Fasti	Resti	Maximum	Exercise	oldpeak	slope	num	thal	diag
0	67.0000	1.0000	4.0000	160.0	286.0	0.0	2.0	108.0000	1.0000	1.5	2.0	3.0	3.0	2
1	67.0000	1.0000	4.0000	120.0	229.0	0.0	2.0	129.0000	1.0000	2.6	2.0	2.0	7.0	1
2	37.0000	1.0000	3.0000	130.0	250.0	0.0	0.0	187.0000	0.0000	3.5	3.0	0.0	3.0	0
3	41.0000	0.0000	2.0000	130.0	204.0	0.0	2.0	172.0000	0.0000	1.4	1.0	0.0	3.0	0
4	56.0000	1.0000	2.0000	120.0	236.0	0.0	0.0	178.0000	0.0000	0.8	1.0	0.0	3.0	0
5	62.0000	0.0000	4.0000	140.0	268.0	0.0	2.0	160.0000	0.0000	3.6	3.0	2.0	3.0	3
6	57.0000	0.0000	4.0000	120.0	354.0	0.0	0.0	163.0000	1.0000	0.6	1.0	0.0	3.0	0
7	63.0000	1.0000	4.0000	130.0	254.0	0.0	2.0	147.0000	0.0000	1.4	2.0	1.0	7.0	2
8	53.0000	1.0000	4.0000	140.0	203.0	1.0	2.0	155.0000	1.0000	3.1	3.0	0.0	7.0	1
9	57.0000	1.0000	4.0000	140.0	192.0	0.0	0.0	148.0000	0.0000	0.4	2.0	0.0	6.0	0
10	56.0000	0.0000	2.0000	140.0	294.0	0.0	2.0	153.0000	0.0000	1.3	2.0	0.0	3.0	0
11	56.0000	1.0000	3.0000	130.0	256.0	1.0	2.0	142.0000	1.0000	0.6	2.0	1.0	6.0	2

Slika 3.1. *Izvorni podatci*

3.2. Obrada izvornih podataka

Izvorne podatke prvo trebamo obraditi kako bi ih mogli koristiti za učenje modela te kako bi dobili što bolje rezultate. Programski kod koje se koristi za obradu podataka može se vidjeti na slici 3.2.

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import os

projectDirPath = os.path.abspath('../')

#replace every ? with np.nan in dataset
raw_data = pd.read_csv(projectDirPath + "/data/raw/mixed_data.data", sep = ',', header=None)
raw_data = raw_data.replace('?', np.nan)

#splitting classes indicators from rest of the data
X = raw_data.iloc[:, :-1].values
y = raw_data.iloc[:, -1].values

for i in range(0, len(y)):
    if y[i] >= 1:
        y[i] = 1


#Dealing with missing data
imputer = SimpleImputer(missing_values = np.nan, strategy = "mean")
imputer.fit(X)
X = imputer.transform(X)

X_transformed = pd.DataFrame(X)
X_transformed.to_csv(projectDirPath + "/data/cleaned/X_cleaned.csv", index = False)

min_max_scaler = preprocessing.MinMaxScaler()
min_max = min_max_scaler.fit_transform(X)

#Spliting dataset into training and testing data
X_train, X_test, y_train, y_test = train_test_split(min_max, y, test_size = 0.2)

X_train = pd.DataFrame(X_train)
X_test = pd.DataFrame(X_test)
y_train = pd.DataFrame(y_train)
y_test = pd.DataFrame(y_test)


X_train.to_csv(projectDirPath + "/data/cleaned/X_train.csv", index = False)
X_test.to_csv(projectDirPath + "/data/cleaned/X_test.csv", index = False)
y_train.to_csv(projectDirPath + "/data/cleaned/y_train.csv", index = False)
y_test.to_csv(projectDirPath + "/data/cleaned/y_test.csv", index = False)
```

Slika 3.2. *Obrada podataka - programski kod*

Za obradu podataka korištene su *python* biblioteke: *pandas*, *numpy*, *sklearn* i os. Na početku je pročitana *csv* datoteka sa izvornim podatcima korištenjem *pandas* biblioteke. Nakon toga su sve vrijednosti koje nedostaju, a označene su sa znakom *?*, zamijenjene s *np.nan*. Nakon toga se podatci razdvajaju na matricu X i vektor y. Dodatno, u vektoru y nalaze se vrijednosti od 0 do 4,

a predstavljaju razinu prisutnosti bolesti srca kod pacijenta. Vrijednosti od 1 do 4 zamijenit ćemo s 1 i to će predstavljati dijagnozi oboljelosti od srca, a 0 će ostati 0 i to će značiti da je pacijent zdrav. Nakon toga, pomoću biblioteke *sklearn*, kreira se objekt klase *SimpleImputer* koji se koristi za popunjavanje vrijednosti koje nedostaju. Sada kada imamo sve podatke, kreiramo objekt klase *MinMaxScaler* kojim skaliramo sve podatke svakog stupca u interval između 0 i 1. Sada kada su podatci obrađeni, razdvajamo ih u set za učenje te set za testiranje pomoću funkcije *train_test_split*. Petina podataka se koristi za testiranje dok se ostatak koristi za učenje modela. Na kraju se setovi podataka za učenje i testiranje spremaju u obliku *csv* datoteke kako bi im se kasnije moglo pristupi te ih koristiti za učenje, odnosno testiranje modela.

4. PREGLED PROGRAMSKOG KODA KORIŠTENOG ZA UČENJE I TESTIRANJE MODELA

Učenje i testiranje različitih modela vrlo je slično pa će u ovom poglavlju biti prikazan primjer za samo jedan model, odnosno za logičku regresiju.

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import make_scorer, recall_score
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import plot_confusion_matrix
import joblib
import os
import json


projectDirPath = os.path.abspath('../')

#loading training and testing sets of data
X_train = pd.read_csv(projectDirPath + "/data/cleaned/X_train.csv").values
X_test = pd.read_csv(projectDirPath + "/data/cleaned/X_test.csv").values
y_train = pd.read_csv(projectDirPath + "/data/cleaned/y_train.csv").values.reshape(-1,)
y_test = pd.read_csv(projectDirPath + "/data/cleaned/y_test.csv").values.reshape(-1,)

#training anv validating ml model
logisticRegression = LogisticRegression(C=1000)
logisticRegression.fit(X_train, y_train)


y_pred = logisticRegression.predict(X_test)
plot_confusion_matrix(logisticRegression, X_test, y_test)

acc = accuracy_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

print("accuracy on the test set: ", acc)
print("recall on the test set: ", rec)
print("confusion matrix:\n ", cm)


#Saving model and evaluation data for later use in streamlit app
joblib.dump(logisticRegression, open(projectDirPath + "/models/logisticReg.joblib", 'wb'))

logisticRegressionEvaluationData = {"acc": acc, "rec": rec, "tn" : int(cm[0, 0]), "fn" : int(cm[1, 0]), "tp" : int(cm[1, 1]), "fp" : int(cm[0, 1])}

with open(projectDirPath + "\\models\\logisticRegression.json", "w") as file:
    json.dump(logisticRegressionEvaluationData, file)
```

Slika 4.1. *Programski kod korišten za učenje i testiranje modela*

Na slici 4.1 prikazan je programski kod koji se koristi za učenje i testiranje modela. Prvo se učitavaju setovi podataka za učenje i testiranje koji su spremljeni kod obrade podataka. Nakon toga model se trenira i prikazuje se matrica zabune te se provjeravaju točnost i odziv modela pomoću funkcija *plot_confusion_matrix*, *accuracy_score* i *recall_score*. Na kraju se spremaju mjere modela i sam model kako bi se kasnije mogle koristiti unutar *streamlit* aplikacije. Za k najbližih susjeda i stroj potpornih vektora, koraci su slični uz neke specifičnosti kao postavljanje broja susjeda i odabir *kernela*, ali programski kod svodi se na iste korake.

5. ANALIZA REZULTATA

Kod analize rezultata se promatrala matrica zabune, odnosno točnost i odziv modela.

Matrica zabune		Stvarna klasa	
		Klasa +	Klasa -
Predviđeno modelom $h_\theta(\mathbf{x})$	Klasa +	**TP** (true positives)	**FP** (false positive)
	Klasa -	**FN** (false negatives)	**TN** (true negatives)

TP – broj primjera iz klase + koji su točno klasificirani modelom
FP – broj primjera iz klase – koji su pogrešno klasificirani modelom kao klasa +
TN – broj primjera iz klase – koji su točno klasificirani modelom
FN – broj primjera iz klase + koji su pogrešno klasificirani modelom kao klasa -

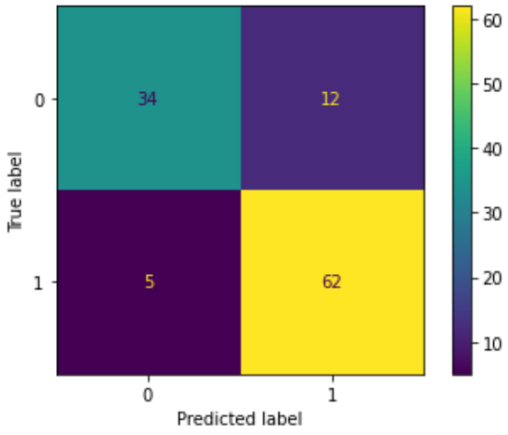
Slika 5.1. *Matrica zabune*

Točnost nam govori koliko je udio točno klasificiranih primjera u cijelom skupu. Odziv nam govori koliki je udio točno klasificiranih primjera u skupu primjera koji pripadaju klasi + ako promatramo sliku 5.1. Odziv modela je iznimno bitan za medicinske svrhe kao što je ova. Pacijentu se ne želi reći da nema nikakvu bolest srca, a da on uistinu ima.

5.1. Logistička regresija

Testiranjem modela logističke regresije na testnom setu podataka dobili smo sljedeće vrijednosti vrijednosti točnosti i odziva:

- Točnost – 0.8496
- Odziv – 0.9254

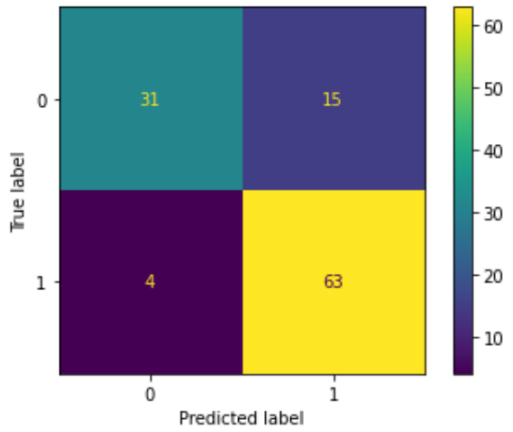


Slika 5.2. *Logistička regresija - matrica zabune*

5.2. Nasumične šume odlučivanja

Testiranjem modela nasumičnih šuma odlučivanja na testnom setu podataka dobili smo sljedeće vrijednosti vrijednosti točnosti i odziva:

- Točnost – 0.8496
- Odziv – 0.9552

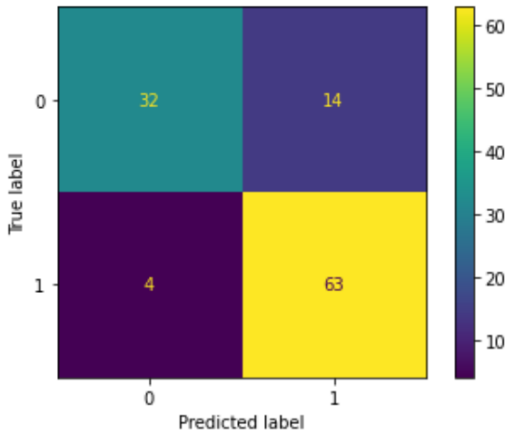


Slika 5.3. *Nasumične šume odlučivanja - matrica zabune*

5.3. Stroj potpornih vektora

Testiranjem modela stroja potpornih vektora na testnom setu podataka dobili smo sljedeće vrijednosti vrijednosti točnosti i odziva:

- Točnost – 0.8407
- Odziv – 0.9403

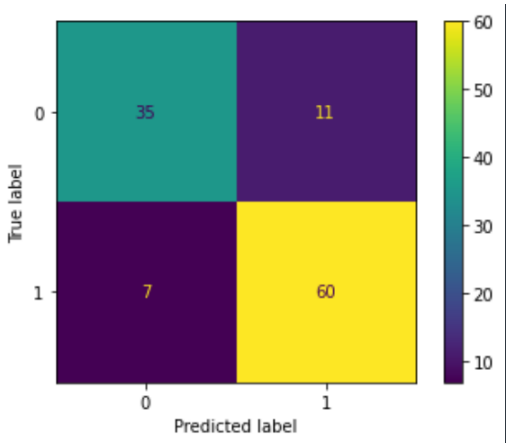


Slika 5.4. *Stroj potpornih vektora - matrica zabune*

5.4. Gaussov Naivni Bayes klasifikator

Testiranjem modela Gaussovog naivnog Bayes-a na testnom setu podataka dobili smo sljedeće vrijednosti vrijednosti točnosti i odziva:

- Točnost – 0.8407
- Odziv – 0.8955

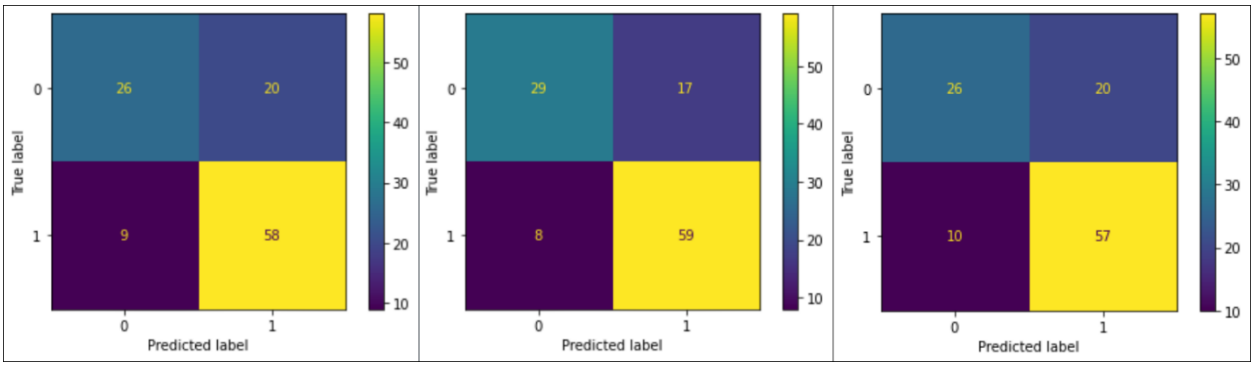


Slika 5.5. *Gaussov Naivni Bayes - matrica zabune*

5.5. K-najbližih susjeda

Trenirala su se tri modela k najbližih susjeda s različitim parametrom k, odnosno „brojem susjeda“. Parametar k je bio tri, četiri i pet. Testiranjem modela dobivene su sljedeće vrijednosti točnosti i odziva:

	Točnost	**Odziv**
k=3	0.7434	0.8657
k=4	0.7788	0.8806
k=5	0.7345	0.8507

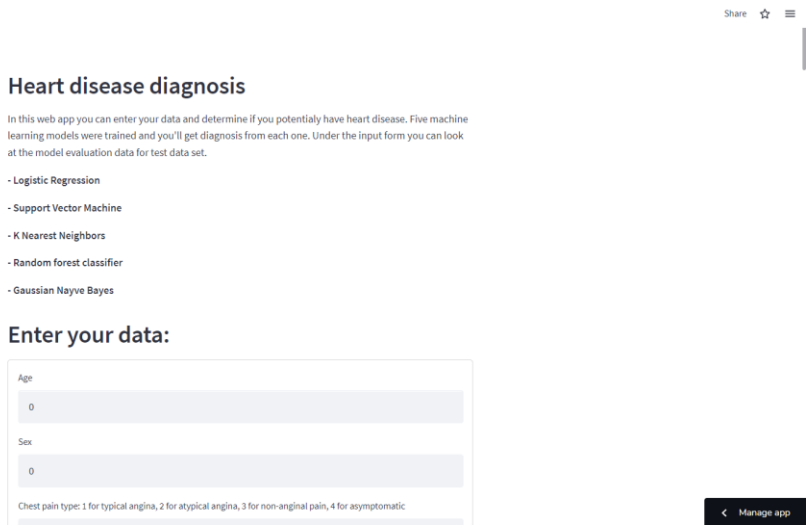


Slika 5.6. *k najbližih susjeda - matrica zabune, k=3 (lijevo), k=4 (sredina), k=5 (desno)*

Najbolji rezultati su postignuti s parametrom k = 4 te se taj se model koristi u *streamlit* aplikaciji.

6. IMPLEMENTACIJA APLIKACIJE

Projekt je implementiran u obliku *streamlit* aplikacije. Aplikacija omogućuje korisnicima unos podataka te procjenu ako su oboljeli od bolesti srca. Za procjenu se koriste svi modeli koji su prethodno opisani u ovom radu. U aplikaciji se također mogu vidjeti rezultati analize, odnosno validacija svih modela i izvorni podaci. Početni zaslon aplikacije može se vidjeti na slici 6.1. *Streamlit* aplikacija dostupna je na linku: https://josip-rizner-heart-disease-classification-streamlitapp-0dgorj.streamlitapp.com/.



Slika 6.1. *Početni zaslon streamlit aplikacije*

7. ZAKLJUČAK

Korištenje strojnog učenja u području medicine može biti vrlo korisno, od ranog otkrivanja pojedinih bolesti do potvrđivanja određenih dijagnoza. U ovom radu uzeti su podatci iz četiri različita izvora, obrađeni su te je pomoću njih istrenirano pet različitih modela. Prikazana je validacija modela i kreirana je korisnička aplikacija gdje korisnici mogu unijeti svoje podatke i dobiti dijagnozu. Najbolji rezultati dobiveni su koristeći algoritam nasumičnih šuma odlučivanja, a najgori koristeći algoritam k najbližih susjeda.

8. LITERATURA

[1] UC Irvine Machine Learning Repository, dataset - https://archive.ics.uci.edu/ml/datasets/Heart+Disease

[2] Scikit-learn, Logistic Regression, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[3] Scikit-learn, Random Forest Classifier, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[4] Scikit-learn, SVC, https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[5] Scikit-learn, GaussianNB, https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[6] Scikit-learn, K-Neighbors Classifier, https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

9. POPIS SLIKA

Slika 2.1. Logistička funkcija 4
Slika 2.2. Model logističke regresije 4
Slika 2.3. Nasumične šume odlučivanja 5
Slika 2.4. Stroj potpornih vektora 6
Slika 2.5. Naive Bayes - formula 7
Slika 2.6. K-najbližih susjeda 8
Slika 3.1. Izvorni podatci 10
Slika 3.2. Obrada podataka - programski kod 11
Slika 4.1. Programski kod korišten za učenje i testiranje modela 13
Slika 5.1. Matrica zabune 14
Slika 5.2. Logistička regresija - matrica zabune 14
Slika 5.3. Nasumične šume odlučivanja - matrica zabune 15
Slika 5.4. Stroj potpornih vektora - matrica zabune 15
Slika 5.5. Gaussov Naivni Bayes - matrica zabune 16
Slika 5.6. k najbližih susjeda - matrica zabune, k=3 (lijevo), k=4 (sredina), k=5 (desno) 17
Slika 6.1. Početni zaslon streamlit aplikacije 18