SVEUČILIŠTE U RIJECI

FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA

Završni projekt

Analiza Srčanih Bolesti

Kolegij: Inteligentni sustavi 2

Nositeljica: Prof. dr.sc. Maja Matetić

Asistent: Dejan Ljubobratović, mag. educ.

Studenti: Sven Kučina, Patrik Goršić

Studij: Sveučilišni diplomski studij informatike

Smjer studija**:** Informacijski i komunikacijski sustavi

Rijeka, lipanj 2022.

Sadržaj

Opis skupa podataka

Odabran je skup podataka za analizu zvan Heart Failure Prediction sa web stranice https://www.kaggle.com/datasets/andrewmvd/heart-failure-clinical-data?resource=download.

Skup može predvidjeti hoće li pacijent preživjeti na temelju varijabli koje predstavljaju razne srčane bolesti. Skup se sastoji od dvanaest varijabli. Opišimo ih u nastavku.

age - starost pacijenta
anaemia - manjak crvenih krvnih stanica (hemoglobin) kao boolean vrijednost
creatinine_phosphokinase - nivo CPK enzima u krvi (mcg/L)
diabetes - boolean vrijednost ako pacijent ima dijabetes ili ne
ejection_fraction - postotak krvi koja "bježi" iz srca svakim otkucajem
high_blood_pressure - boolean vrijednost ima li pacijent hipertenziju (povišen krvni tlak)
platelets - količina trombocita u krvi (kiloplatelets/mL)
serum_creatinine - količina seruma kreatinina u krvi (mEq/L)
serum_sodium - količina serum natrija u krvi (mEq/L) sex - spol pacijenta
smoking - puši li pacijent ili ne (boolean)
time - razdoblje praćenja, vrijeme nakon kojeg je provedena kontrola pacijenta nakon što otpušten iz bolnice
DEATH_EVENT - ako je pacijent preminuo tijekom razdoblja praćenja (boolean)

Istraživačka analiza podataka

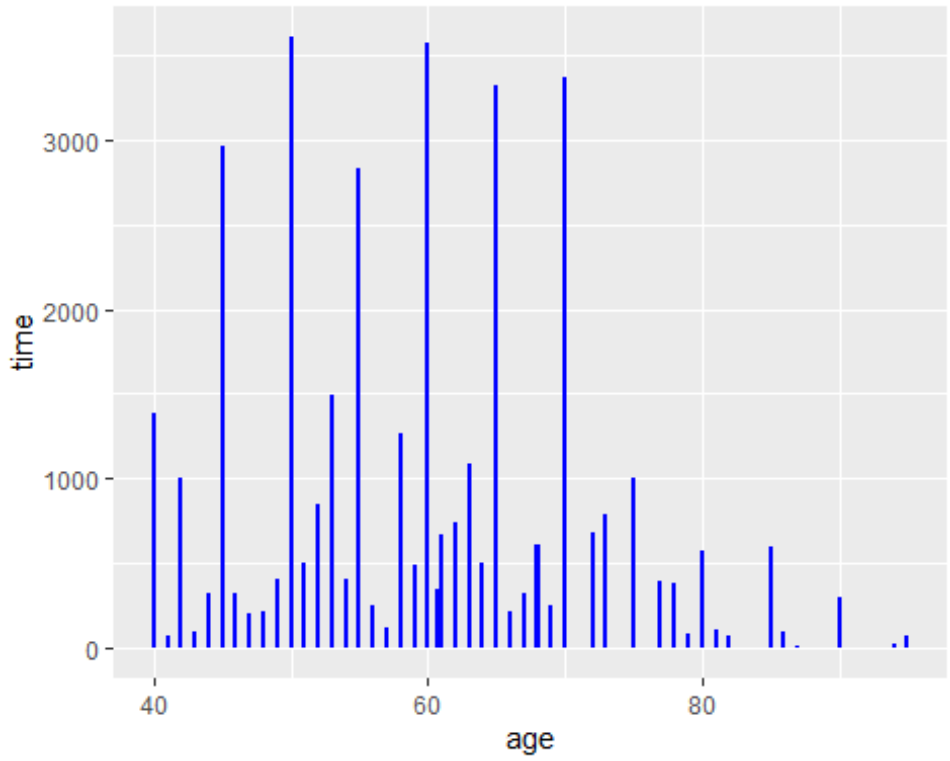
Istraživačka analiza podataka je preliminarno istraživanje podataka da bi se bolje razumjele njihove karakteristike ili analiza na temlju vizalizacije. Probajmo vizualizirati naš dataset. Učitajmo naš dataset.

```
library(ggplot2)
dataset=read.csv("C:/FAKS/IS2/HeartFailureSeminar/heart_failure.csv")
```

Istražimo naš dataset kroz razne grafove.

Najprije ćemo prikazati graf koji na osi x prikazuje godine pacijenta, a na osi y varijablu time koja predstavlja koliko se često prati pacijent. Što je time manji to se pacijent češće prati odnosno pregledava.

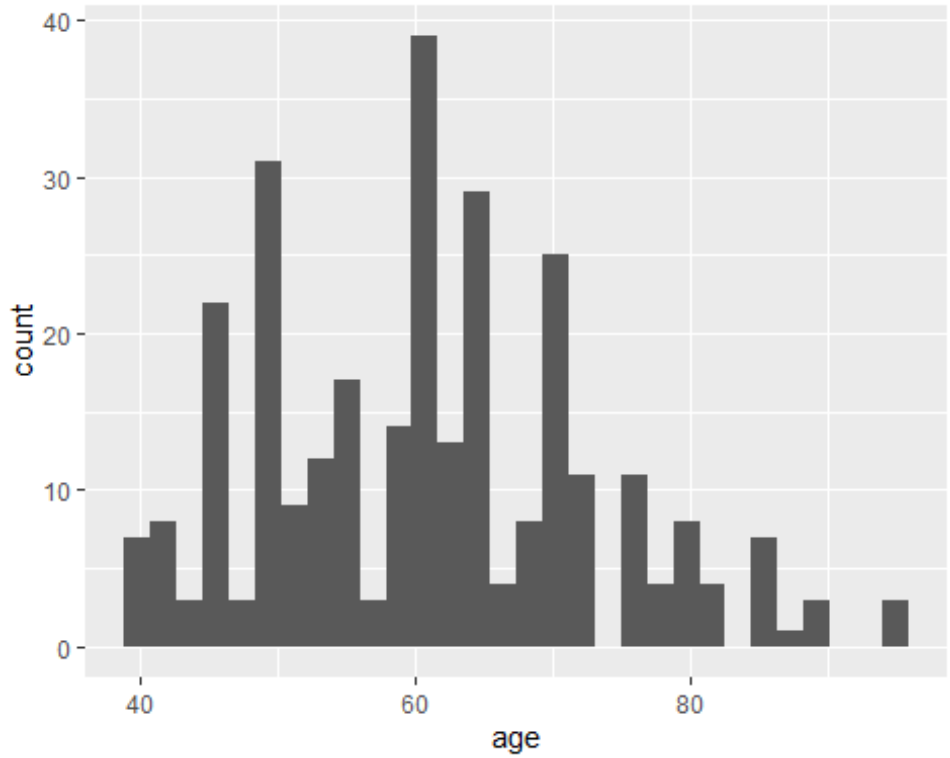
```
ggplot(dataset, aes(x=age,y=time)) + geom_col(fill='blue')
```



Vidimo da uglavnom postoji pravilnost što je pacijent stariji to je time manji što znači da se pacijent češće prati. To ima smisla budući da su stariji pacijenti uglavnom zdravstveno ugroženiji nego mlađi pacijenti.

Pogledajmo zatim histogram koji će nam prebrojati koliko ima pacijenata određenih godina.

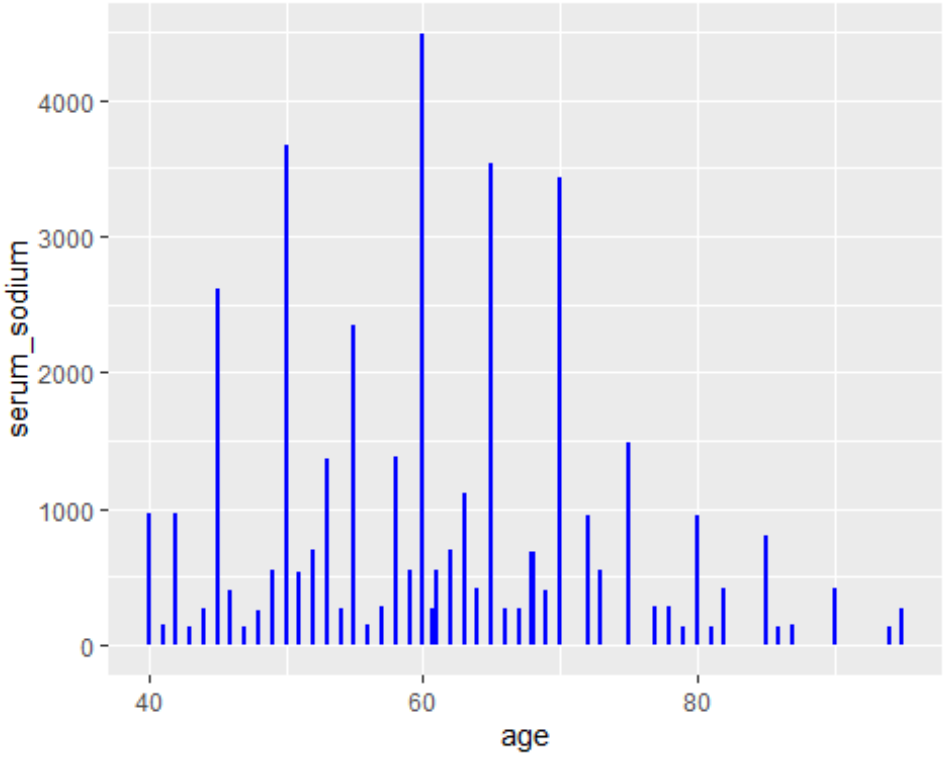
```
ggplot(dataset, aes(x=age)) + geom_histogram()
```



Vidimo da najviše srčanih bolesnika ima oko šezdesetih godina života.

Količina natrija u serumu po godinama

```
ggplot(dataset, aes(x=age,y=serum_sodium)) + geom_col(fill='blue')
```

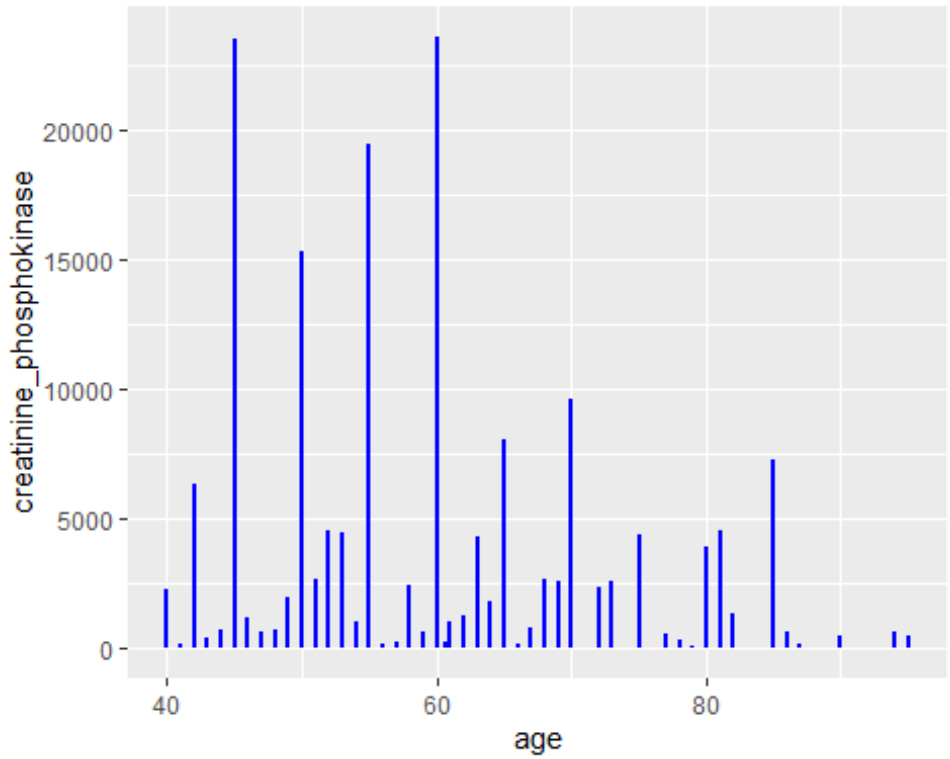


Količina kreatinina u serumu po godinama

```
ggplot(dataset, aes(x=age,y=serum_creatinine)) + geom_col(fill='blue')
```

Količina kreatina kinaze po godinama

```
  ggplot(dataset, aes(x=age,y=creatinine_phosphokinase)) +
geom_col(fill='blue')
```



Priprema podataka za analizu

Prije pripreme podataka potrebno ih je najprije učitati. To radimo putem read.csv() funkcije.

```
  dataset=read.csv("C:/FAKS/IS2/HeartFailureSeminar/heart_failure.csv")
```

Pogledajmo dimenzije našeg dataset-a.

```
  dim(dataset)

## [1] 299  13
```

Vidimo da naš dataset ima 299 slučajeva i 13 atributa.

Prikažimo zatim naš početni dataset.

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
1	75	0	582	0	20	1	265000	1.90	130	1	0	4	1
2	55	0	7861	0	38	0	263358	1.10	136	1	0	6	1
3	65	0	146	0	20	0	162000	1.30	129	1	1	7	1
4	50	1	111	0	20	0	210000	1.90	137	1	0	7	1
5	65	1	160	1	20	0	327000	2.70	116	0	0	8	1
6	90	1	47	0	40	1	204000	2.10	132	1	1	8	1
7	75	1	246	0	15	0	127000	1.20	137	1	0	10	1
8	60	1	315	1	60	0	454000	1.10	131	1	1	10	1
9	65	0	157	0	65	0	263358	1.50	138	0	0	10	1
10	80	1	123	0	35	1	388000	9.40	133	1	1	10	1
11	75	1	81	0	38	1	368000	4.00	131	1	1	10	1
12	62	0	231	0	25	1	253000	0.90	140	1	1	10	1
13	45	1	981	0	30	0	136000	1.10	137	1	0	11	1
14	50	1	168	0	38	1	276000	1.10	137	1	0	11	1
15	49	1	80	0	30	1	427000	1.00	138	0	0	12	0
16	82	1	379	0	50	0	47000	1.30	136	1	0	13	1
17	87	1	149	0	38	0	262000	0.90	140	1	0	14	1
18	45	0	582	0	14	0	166000	0.80	127	1	0	14	1
19	70	1	125	0	25	1	237000	1.00	140	0	0	15	1

Početni dataset

Obratimo pozornost na boolean stupce odnosno varijable. One sadrže vrijednosti 0 i vrijednosti 1. To nam ne paše kada gradimo model stabla odlučivanja jer kao izlaz želimo dobiti “yes” ili “no” vrijednost. Također, kada želimo grafički prikazati stablo odlučivanja, bolje se vidi nakon što varijablu koju predviđamo pretvorimo iz boolean vrijednosti u stupac koji sadrži vrijednosti “yes” i “no”.

Sada želimo predvidjeti varijablu DEATH_EVENT koja nam govori ako je pacijent preminuo ili ne. Najprije je potrebno pretvoriti boolean vrijednosti varijable DEATH_EVENT u vrijednosti “yes” i “no”. To radimo koristeći funkcije ifelse() i mutate().

Učitavamo paket potreban za manipulaciju podataka (filter, arrange, mutate…)

```
library(dplyr)

##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Pomoću ifelse() funkcije, varijablu DEATH_EVENT mijenjamo iz boolean vrijednosti 1 i 0 u vrijednosti "yes" i no. Ako se prepozna vrijednost 1, zapisuje "yes", inače "no". Zatim rezultat spremamo u varijablu DEATH_EVENT_2

```
  DEATH_EVENT_2=ifelse(dataset$DEATH_EVENT==1, "Yes", "No")
```

Pomoću pipe operatora %>% i funkcije mutate() novostvorenu varijablu sa vrijednostima "yes" i "no" spremamo u varijablu DEATH_EVENT te spremamo izmjene u dataset.

```
  dataset=dataset %>% mutate(DEATH_EVENT=DEATH_EVENT_2)
```

Prikažimo novodobiveni dataset.

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
1	75	0	582	0	20	1	265000	1.90	130	1	0	4	Yes
2	55	0	7861	0	38	0	263358	1.10	136	1	0	6	Yes
3	65	0	146	0	20	0	162000	1.30	129	1	1	7	Yes
4	50	1	111	0	20	0	210000	1.90	137	1	0	7	Yes
5	65	1	160	1	20	0	327000	2.70	116	0	0	8	Yes
6	90	1	47	0	40	1	204000	2.10	132	1	1	8	Yes
7	75	1	246	0	15	0	127000	1.20	137	1	0	10	Yes
8	60	1	315	1	60	0	454000	1.10	131	1	1	10	Yes
9	65	0	157	0	65	0	263358	1.50	138	0	0	10	Yes
10	80	1	123	0	35	1	388000	9.40	133	1	1	10	Yes
11	75	1	81	0	38	1	368000	4.00	131	1	1	10	Yes
12	62	0	231	0	25	1	253000	0.90	140	1	1	10	Yes
13	45	1	981	0	30	0	136000	1.10	137	1	0	11	Yes
14	50	1	168	0	38	1	276000	1.10	137	1	0	11	Yes
15	49	1	80	0	30	1	427000	1.00	138	0	0	12	No
16	82	1	379	0	50	0	47000	1.30	136	1	0	13	Yes
17	87	1	149	0	38	0	262000	0.90	140	1	0	14	Yes
18	45	0	582	0	14	0	166000	0.80	127	1	0	14	Yes
19	70	1	125	0	25	1	237000	1.00	140	0	0	15	Yes

Stabla odlučivanja

Podijelimo sada skup na treniranje i testiranje. Postavit ćemo veličinu skupa za treniranje na 80%, a ostalo za skup za testiranje. Koristimo funkciju sample() kako bismo randomizirali podatke. Na kraju, na osnovu indeksa kreairamo skup za treniranje i testiranje.

```
n = nrow(dataset)
n_train=round(0.80*n)
set.seed(123)
train_indeksi <- sample(1:n, n_train)
dataset_train <- dataset[train_indeksi, ]
dataset_test <- dataset[-train_indeksi, ]
```

Kako bismo izradili stablo odlučivanja, koristit ćemo biblioteku rpart te za iscrtavanje stabla koristimo rpart.plot().

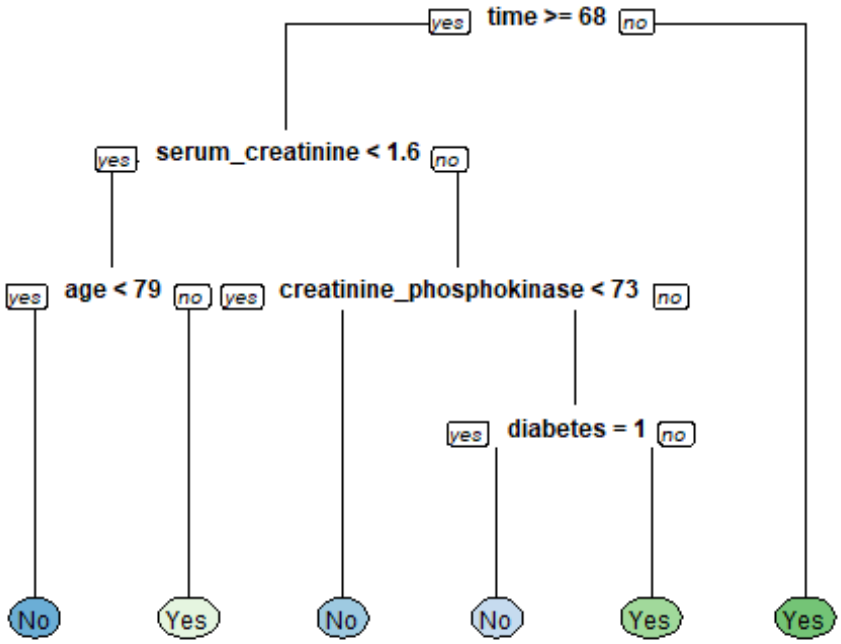
```
library("rpart")
library("rpart.plot")
```

Trenirajmo model.Argument formula određuje prediktore i ciljnu varijablu. U ovom slučaju naša ciljna varijabla je DEATH_EVENT te na temelju svih ostalih varijabli predvidjeti DEATH_EVENT koji označava ako je pacijent preminuo ili ne tijekom razdoblja praćenja.

```
dataset_stablo_model = rpart(formula = DEATH_EVENT ~ .,
                             data=dataset_train,
                             method="class")
```

Naposlijetku, iscrtajmo stablo pomoću rpart.plot().

```
rpart.plot(x = dataset_stablo_model, yesno = 2, type = 0, extra = 0)
```



Iščitajmo značenje stabla. Ako je time >= 68 gleda se razina kreatinina u serumu. Ako je kreatinin u serumu serum_creatinine < 1.6 gledaju se godine. Ako je age < 79 stablo predviđa da pacijent nije preminuo. Također ako time nije veći ili jednak od 68, predviđa se da je pacijent preminuo.

Ridge regresija

Za korištenje Ridge regresije potrebno je uključiti paket glmnet. Glavna funkcija paketa je glmnet() koja se koristi za treniranje modela.

Najprije ponovno učitavamo skup podataka, zatim u varijablu x koja predstavlja prediktore stavljamo cijeli skup podataka. U varijablu y spremamo ciljnu varijablu DEATH_EVENT.

```
  library (ISLR)
  dataset=read.csv("C:/FAKS/IS2/HeartFailureSeminar/heart_failure.csv")
  x<-model.matrix(DEATH_EVENT ~.,dataset )[,-1]
  y<-dataset$DEATH_EVENT
```

Funkcija glmnet() ima argument alpha koji određuje tip modela. Ako posatvimo alpha na 0 izvodit će se Ridge regresija. Funkcija glmnet() izvodi Ridge regresiju za unaprijed odabrani raspon lambda vrijednosti (grid). Zadajemo da se u grid vektoru lambda mijenja od 10^{10}(null) do 10^{-2}(least squares) te time pokrivajući raspon od null modela koji ne sadrži prediktore do modela dobivenog metodom najmanjih kvadrata.

```
  library (glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-4
```

```
  grid =10^seq(10,-2, length=100)
  ridge_model=glmnet(x, y, alpha=0, lambda=grid)
```

Uz svaku lambda vrijednost pridružen je vektor koeficijenata modela kojemu možemo pristupiti sa coeff(). U našem slučaju matrica ima 13 redaka (12 prediktora) i 100 stupaca (100 vrijednosti lambda).

```
  dim(coef(ridge_model))
```

```
## [1]  13 100
```

Ako se koristi velika vrijednost lambda procjene koeficijenata će biti znatno manje. Ako se koristi mala vrijednost lambda procjene koeficijenata će biti znatno veće. Prikažimo vrijednosti koeficijenta kada je lambda 50, dakle na sredini (jer imamo 100 vrijednosti lambda). Također prikažimo L2 norma. L2 norma vektora mjeri udaljenost procjene koeficijenta od nule. Kako lambda raste L2 norma se smanjuje.

```
  cat("lambda:", ridge_model$lambda[50])
```

```
## lambda: 11497.57
```

```
  # Koeficijenti za 50. lambda, tj. lambda=11497
  coef(ridge_model)[,50]
```

```
##              (Intercept)                      age                  anaemia
##             3.211860e-01             4.050715e-07             2.536594e-06
## creatinine_phosphokinase                 diabetes        ejection_fraction
```

```
##             1.227726e-09            -7.461652e-08            -4.309983e-07
##      high_blood_pressure                platelets         serum_creatinine
##             3.151450e-06            -9.540634e-12             5.401856e-06
##             serum_sodium                      sex                  smoking
##            -8.400809e-07            -1.715349e-07            -5.125880e-07
##                     time
##            -1.289327e-07
```

```
  cat("L2 norma:", sqrt(sum(coef(ridge_model)[-1,50]^2)))
```

```
## L2 norma: 6.849521e-06
```

Za puno manji lambda, na primjer lambda=705, koeficijenti i L2 norma će biti sljedeći.

```
  cat("lambda:", ridge_model$lambda[60])
```

```
## lambda: 705.4802
```

```
  cat("koeficijenti:")
```

```
## koeficijenti:
```

```
  coef(ridge_model)[,60]
```

```
##              (Intercept)                      age                  anaemia
##             3.229542e-01             6.595505e-06             4.128589e-05
## creatinine_phosphokinase                 diabetes        ejection_fraction
##             2.000407e-08            -1.200142e-06            -7.019342e-06
##      high_blood_pressure                platelets         serum_creatinine
##             5.129438e-05            -1.552887e-10             8.795704e-05
##             serum_sodium                      sex                  smoking
##            -1.367651e-05            -2.819042e-06            -8.352210e-06
##                     time
##            -2.099594e-06
```

```
  cat("L2 norma:", sqrt(sum(coef(ridge_model)[-1,60]^2)))
```

```
## L2 norma: 0.0001115144
```

Možemo iskoristiti predict() funkciju u kojoj direktno možemo unijeti vrijednost lambde kroz argument s.

```
  predict(ridge_model,s=50,type="coefficients")[1:13,]
```

```
##              (Intercept)                      age                  anaemia
##             3.477702e-01             9.346823e-05             5.813548e-04
## creatinine_phosphokinase                 diabetes        ejection_fraction
##             2.868189e-07            -1.369308e-05            -9.992190e-05
##      high_blood_pressure                platelets         serum_creatinine
##             7.222263e-04            -2.189775e-09             1.247729e-03
##             serum_sodium                      sex                  smoking
##            -1.936068e-04            -4.540312e-05            -1.200419e-04
```

```
##                       time
##             -2.985587e-05
```

Razdijelimo podatke na skup za treniranje i testiranje kako bismo procijenili pogrešku Ridge modela.

```
  set.seed(1)
  train=sample(1:nrow(x), nrow(x)/2)
  test=(-train)
  y.test=y[test]
```

Na skupu za treniranje učimo Ridge regresiju i evaluiramo MSE pogrešku modela na skupu za testiranje uz lambda=4. Koristit ćemo funkciju predict() za predviđanje na podacima za testiranje tako da umjesto type="coefficient" stavimo newx=x[test,].

```
  ridge_model=glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh=1e-12)
  ridge.pred=predict(ridge_model,s=4,newx=x[test,])
  mse_pogreska<-mean((ridge.pred-y.test)^2)
  cat("MSE pogreška:", mse_pogreska)
```

```
## MSE pogreška: 0.1966082
```

Provjerimo sada da li je ridge regresija uz lambda=4 uspješnija od izvođenja metode najmanjih kvadrata (koja je zapravo Ridge regresija uz lambda=0).

```
  ridge.pred=predict.glmnet (ridge_model ,s=0, newx=x[test ,])
  mean((ridge.pred -y.test)^2)
```

```
## [1] 0.1260009
```

```
  lm(y~x, subset =train)
```

```
##
## Call:
## lm(formula = y ~ x, subset = train)
##
## Coefficients:
##               (Intercept)                       xage
##                 2.476e+00                   6.663e-03
##                  xanaemia  xcreatinine_phosphokinase
##                -9.594e-03                   3.372e-05
##                 xdiabetes         xejection_fraction
##                 6.869e-02                  -7.165e-03
##      xhigh_blood_pressure                  xplatelets
##                -4.582e-02                  -7.089e-08
##         xserum_creatinine              xserum_sodium
##                 9.074e-02                  -1.480e-02
##                      xsex                   xsmoking
##                -8.012e-02                   2.922e-02
##                     xtime
##                -2.712e-03
```

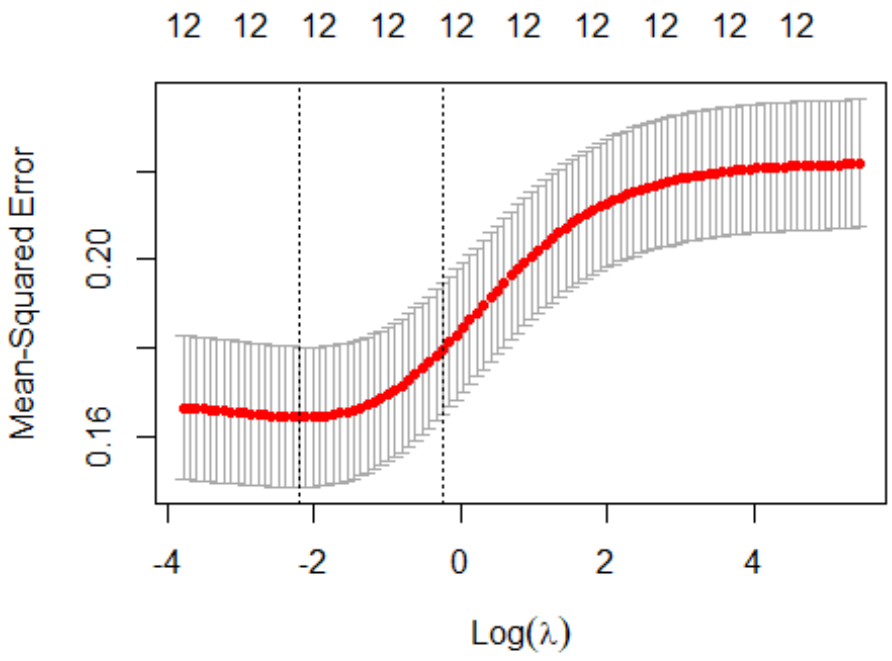
```
  predict(ridge_model,s=0,type="coefficients") [1:13 ,]
```

```
##               (Intercept)                       age                   anaemia
##              2.417828e+00              6.586929e-03             -8.252931e-03
## creatinine_phosphokinase                  diabetes          ejection_fraction
##              3.306544e-05              6.892266e-02             -6.967628e-03
##      high_blood_pressure                 platelets           serum_creatinine
##             -4.302238e-02             -7.686425e-08              8.915870e-02
##              serum_sodium                       sex                   smoking
##             -1.445681e-02             -7.550005e-02              2.661502e-02
##                       time
##             -2.649536e-03
```

Dobili smo manju pogrešku od 0.1260009 tako da je metoda najmanjih kvadrata u ovom slučaju uspješnija.

Odaberimo najbolju lambdu korištenje unakrsne validacije za podeševanje parametra lambda. Koristit ćemo funkciju cv.glmnet().

```
  set.seed(1)
  cv.out=cv.glmnet(x[train,],y[train],alpha=0)
  plot(cv.out)
```




```
  bestlambda=cv.out$lambda.min
  cat("najbolji lambda je:", bestlambda)
```

```
## najbolji lambda je: 0.1110211
```

Vrijednost lambda za koju je pogreška najmanja iznosi 0.1110211.

Pogledajmo koliko iznosi MSE pogreška kada koristimo najbolju lambda vrijednost.

```
  ridge.pred=predict(ridge_model,s=bestlambda,newx=x[test,])
  cat("MSE pogreška kada koristimo bestlambda:", mean((ridge.pred-y.test)^2))
```

```
## MSE pogreška kada koristimo bestlambda: 0.1282796
```

MSE pogreška kada koristimo najbolju lambda vrijednost iznosi 0.1282796. Vidimo da se pogreška smanjila.

Sada ponovno treniramo model na cijelom skupu uzoraka uz najbolju lambda vrijednost koja je dobivena unakrsnom validacijom.

```
out=glmnet(x,y,alpha=0)
predict(out,type="coefficients",s=bestlambda)[1:13,]
```

```
##              (Intercept)                       age                   anaemia 
##             1.613811e+00             5.112440e-03             7.189890e-03 
## creatinine_phosphokinase                 diabetes         ejection_fraction 
##             2.870039e-05             1.379863e-02            -7.951677e-03 
##      high_blood_pressure                platelets          serum_creatinine 
##             2.561445e-03            -7.752909e-08             7.397846e-02 
##             serum_sodium                      sex                   smoking 
##            -7.985241e-03            -4.067871e-02            -8.256322e-03 
##                      time 
##            -2.226752e-03
```

Ni jedan koeficijent nije jednak nuli jer Ridge regresija ne izvodi selekciju prediktora.

Strojno učenje

Želimo napraviti modele strojnog učenja koji predviđaju DEATH_EVENT.

Prije nego što krenemo, moramo napraviti kratku predobradu našeg dataset-a. Potrebno je ciljnu varijablu DEATH_EVENT pretvoriti u faktor sa 2 nivoa. Ako to ne učinimo dobijemo sljedeću grešku.

```
Error: Metric Accuracy not applicable for regression models
In addition: Warning message:
In train.default(x, y, weights = w, ...) :
  You are trying to do regression and your outcome only has two possible values Are you trying to do classification? If
 so, use a 2 level factor as your outcome column.
> library(keras)
```

Greška

To radimo pomoću funkcije mutate() i factor na sljedeći način.

```
  dataset=read.csv("C:/FAKS/IS2/HeartFailureSeminar/heart_failure.csv")

  DEATH_EVENT_faktor=factor(dataset$DEATH_EVENT)

  library(dplyr)
  dataset=dataset %>% mutate(DEATH_EVENT=DEATH_EVENT_faktor)
```

Nakon što je to gotovo, da bismo mogli procijeniti ako nam je model dobar podijelit ćemo dataset na dva skupa. Skup koji sadrži 80% podataka za treniranje te drugi skup od 20% podataka za testiranje. Kako bismo to napravili koristit ćemo funkciju createDataPartition() unutar paketa caret.

```
  library(caret)

## Loading required package: lattice

  validation_index<-createDataPartition(dataset$DEATH_EVENT, p =0.80, list =
FALSE)
  dataset_test<-dataset[-validation_index, ]
  dataset_train<-dataset[validation_index, ]
```

Evaluacija algoritama

Postavljanje 10-fold cross validacije

Ovaj postupak će podijeliti skup podataka na 10 dijelova (9 za treniranje i 1 za testiranje), a zatim će proći sve kombinacije tih skupova. Proces će se ponoviti 3 puta za svaki od 5 algoritama, s različitom podjelom podataka na 10 skupina kako bi se dobile preciznije procjene.

```
  # Run algorithms using 10-fold cross validation
  control<-trainControl(method = "cv", number = 10)
  metric<-"Accuracy"
```

Izgradnja pet modela predviđanja

Naš problem pokušati ćemo riješti korištenjem 5 različitih algoritama:

1 - Linear Discriminant Analysis (LDA)
2 - Classification and Regression Trees (CART)
3 - k-Nearest Neighbors (kNN)
4 - Support Vector Machines (SVM) with a linear kernel
5 - Random Forest (RF)

LDA algoritam:

```
  # Linear Discriminant Analysis (LDA)
  library(e1071)
  set.seed(7)
  fit.lda <- caret::train(DEATH_EVENT~., data=dataset_train, method="lda",
metric=metric, trControl=control)
```

Nelinearni algoritam CART:

```
  # Classification and Regression Trees (CART)
  set.seed(7)
  fit.cart <- caret::train(DEATH_EVENT~., data=dataset_train, method="rpart",
metric=metric, trControl=control)
```

Nelinearni algoritam kNN:

```
  # k-Nearest Neighbors (kNN)
  set.seed(7)
  fit.knn <- caret::train(DEATH_EVENT~., data=dataset_train, method="knn",
metric=metric, trControl=control)
```

SVM algoritam:

```
  library(kernlab)

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha

  # Support Vector Machines (SVM)
  set.seed(7)
  fit.svm <- caret::train(DEATH_EVENT~., data=dataset_train,
method="svmRadial", metric=metric, trControl=control)
```

RF algoritam:

```
  library(randomForest)

## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
  # Random Forest (RF)
  set.seed(7)
  fit.rf <- caret::train(DEATH_EVENT~., data=dataset_train, method="rf",
metric=metric, trControl=control)
```

Odabir najboljeg modela

Nakon što smo kreirali pet modela i procjenili točnost za svaki, sljedeći zadatak je bio usporediti modele i odabrati najtočnije.

Da bi to učinili, napraviti ćemo popis korištenih modela i proslijediti ove rezultate funkciji sažetka "summary()" da bi dobili izlaz koji prikazuje točnost svakog klasifikatora kao i mnoge druge mjerne podatke, poput Kappe.

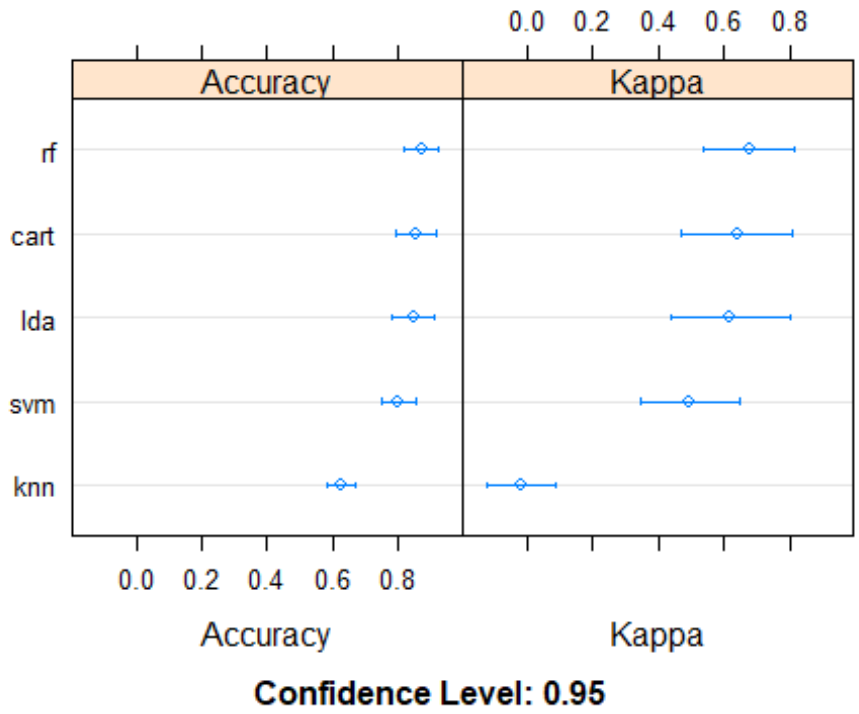
```
  results <- resamples(list(lda=fit.lda,cart=fit.cart, knn=fit.knn,
svm=fit.svm, rf=fit.rf))
  summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: lda, cart, knn, svm, rf
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda  0.7083333 0.7522645 0.8775000 0.8442971 0.9157609 0.9600000    0
## cart 0.6956522 0.7952899 0.8765217 0.8529638 0.9191667 0.9583333    0
## knn  0.5217391 0.5875000 0.6250000 0.6251232 0.6730435 0.7083333    0
## svm  0.6666667 0.7522645 0.8130435 0.8003116 0.8621739 0.8750000    0
## rf   0.7826087 0.8020833 0.8765217 0.8699928 0.9200000 0.9583333    0
##
## Kappa
##            Min.    1st Qu.       Median        Mean    3rd Qu.      Max.
NA's
## lda   0.1882353  0.4243101 7.210508e-01  0.61771214 0.80822542 0.9049430
0
```

```
## cart  0.1827411  0.4988372 7.159357e-01  0.63908155 0.81525735 0.9090909
0
## knn  -0.2842640 -0.1096939 1.480297e-16 -0.01953535 0.07541899 0.2222222
0
## svm   0.1428571  0.3733995 5.347635e-01  0.49596955 0.67196479 0.7272727
0
## rf    0.3575419  0.5120690 7.045260e-01  0.67591384 0.81617647 0.9090909
0
```

Zatim kreirajmo graf rezultata evaluacije modela te usporedimo raspršenost i srednju točnost svakog modela.

```
  dotplot(results)
```



Važno je napomenuti da postoji niz mjerenja točnosti za svaki algoritam, jer je svaki algoritam ocijenjen 10 puta (10 puta unakrsna validacija), zbog čega je trebalo usporediti srednje vrijednosti točnosti modela. Najtočniji model u ovom slučaju bio je RF, s obzirom da je imao najveću srednju točnost s najmanjom raspršenosti.

Kappa statistika nam pokazuje koliko su se instance klasificirane klasifikatorom strojnog učenja podudarale s podacima označenima kao točnim. U osnovi što je kappa bliže 1 to je bolje.

Budući da je RF identificiran kao najbolji model, prikazati ćemo summary samo za njega.

```
  # Sažetak našeg najboljeg modela - RF
  print(fit.rf)
```

```
## Random Forest
##
## 240 samples
##  12 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 216, 215, 215, 216, 215, 216, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8699928  0.6759138
##    7    0.8400870  0.6117095
##   12    0.8277536  0.5839761
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Predviđanje

```
  predictions<-predict(fit.rf, dataset_test)
  confusionMatrix(predictions, dataset_test$DEATH_EVENT)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction  0  1
##          0 34  8
##          1  6 11
##
##                Accuracy : 0.7627
##                  95% CI : (0.6341, 0.8638)
##     No Information Rate : 0.678
##     P-Value [Acc > NIR] : 0.1029
##
##                   Kappa : 0.4411
##
##  Mcnemar's Test P-Value : 0.7893
##
##             Sensitivity : 0.8500
##             Specificity : 0.5789
##          Pos Pred Value : 0.8095
##          Neg Pred Value : 0.6471
##              Prevalence : 0.6780
##          Detection Rate : 0.5763
##    Detection Prevalence : 0.7119
##       Balanced Accuracy : 0.7145
##
##        'Positive' Class : 0
##
```

RF se pokazao kao najprecizniji model na skupu za treniranje, ali moramo odrediti točnost modela na skupu za testiranje da bi dobili neovisnu završnu provjeru ispravnosti najboljeg modela. Zadržali smo skup za testiranje za provjeru valjanosti u slučaju overfitinga.

RF model je pokrenut izravno na skupu za testiranje, a rezultati su sažeti u matrici konfuzije. Točnost je 86%.