UNIVERSITÀ DEGLI STUDI DI MILANO

Module Statistical Learning, Deep Learning and Artificial Intelligence

Final Exam ~ Supervised Project

June, 2020 – 2021

Nazli Begum CIRPANLI

942345

Table of Contents

Aim of the Analysis 3

Data Description 3

Categorical variables 4

Numerical variables 5

Standardization of Numerical Variables 7

Methodology 7

Logistic Regression 7

Logistic Regression Assumptions: 8

Linear Discriminant Analysis 9

Penalized Logistic Regression - LASSO 10

Decision Tree Classification 11

Pruning the trees 11

ROC CURVES 13

Bagging 14

Random Forest 14

Conclusion 16

Appendix: R Code 17

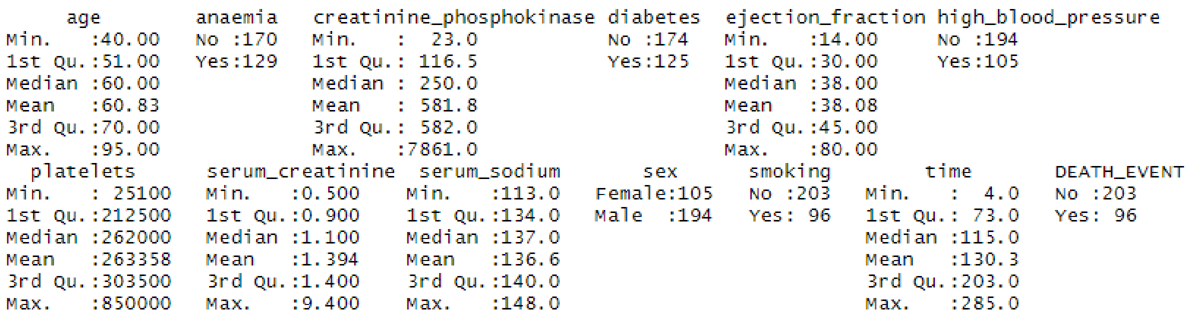
Aim of the Analysis

Purpose of this analysis is to find a good prediction model that can output if a patient with a heart disease survive or die. Several different classification techniques were studied and their results were compared. According to the results, all of the models gave similar results and achieved more than 80% accuracy rate.

Data Description

Dataset used in this analysis consists of medical records of 299 heart failure patients of Faisalabad Institute of Cardiology and the Allied Hospital in Faisalabad (Pakistan), during April–December 2015.

Dataset was taken from https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records



```
      age          anaemia   creatinine_phosphokinase diabetes  ejection_fraction high_blood_pressure
 Min.   :40.00   No :170   Min.   :  23.0           No :174   Min.   :14.00     No :194
 1st Qu.:51.00   Yes:129   1st Qu.: 116.5           Yes:125   1st Qu.:30.00     Yes:105
 Median :60.00             Median : 250.0                     Median :38.00
 Mean   :60.83             Mean   : 581.8                     Mean   :38.08
 3rd Qu.:70.00             3rd Qu.: 582.0                     3rd Qu.:45.00
 Max.   :95.00             Max.   :7861.0                     Max.   :80.00
   platelets      serum_creatinine  serum_sodium      sex        smoking       time        DEATH_EVENT
 Min.   : 25100   Min.   :0.500   Min.   :113.0   Female:105   No :203   Min.   :  4.0   No :203
 1st Qu.:212500   1st Qu.:0.900   1st Qu.:134.0   Male  :194   Yes: 96   1st Qu.: 73.0   Yes: 96
 Median :262000   Median :1.100   Median :137.0                          Median :115.0
 Mean   :263358   Mean   :1.394   Mean   :136.6                          Mean   :130.3
 3rd Qu.:303500   3rd Qu.:1.400   3rd Qu.:140.0                          3rd Qu.:203.0
 Max.   :850000   Max.   :9.400   Max.   :148.0                          Max.   :285.0
```

There are 6 binary and 7 continous variables with no missing values.Brief descriptions for each variable are as follows:

Age: Age of the patient (years)

Anaemia: Decrease of red blood cells or hemoglobin (boolean)

High blood pressure: If the patient has hypertension (boolean)

Creatinine phosphokinase (CPK): Level of the CPK enzyme in the blood (mcg/L)

Diabetes: If the patient has diabetes (boolean)

Ejection fraction: Percentage of blood leaving the heart at each contraction (percentage)

Platelets: Platelets in the blood (kiloplatelets/mL)

Sex: woman or man (binary)

Serum creatinine: level of serum creatinine in the blood (mg/dL)

Serum sodium: Level of serum sodium in the blood (mEq/L)

Smoking: If the patient smokes or not (boolean)

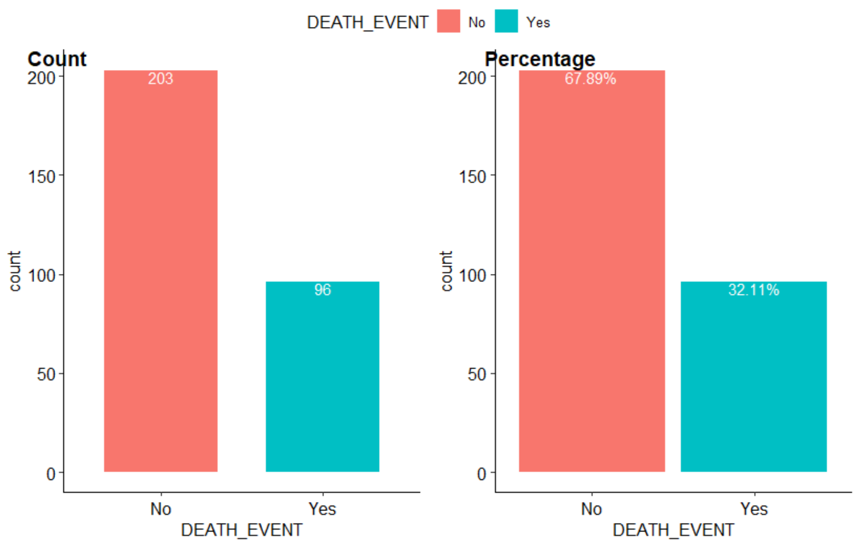
Time: Follow-up period (days)

[target] Death event: If the patient deceased during the follow-up period (boolean)

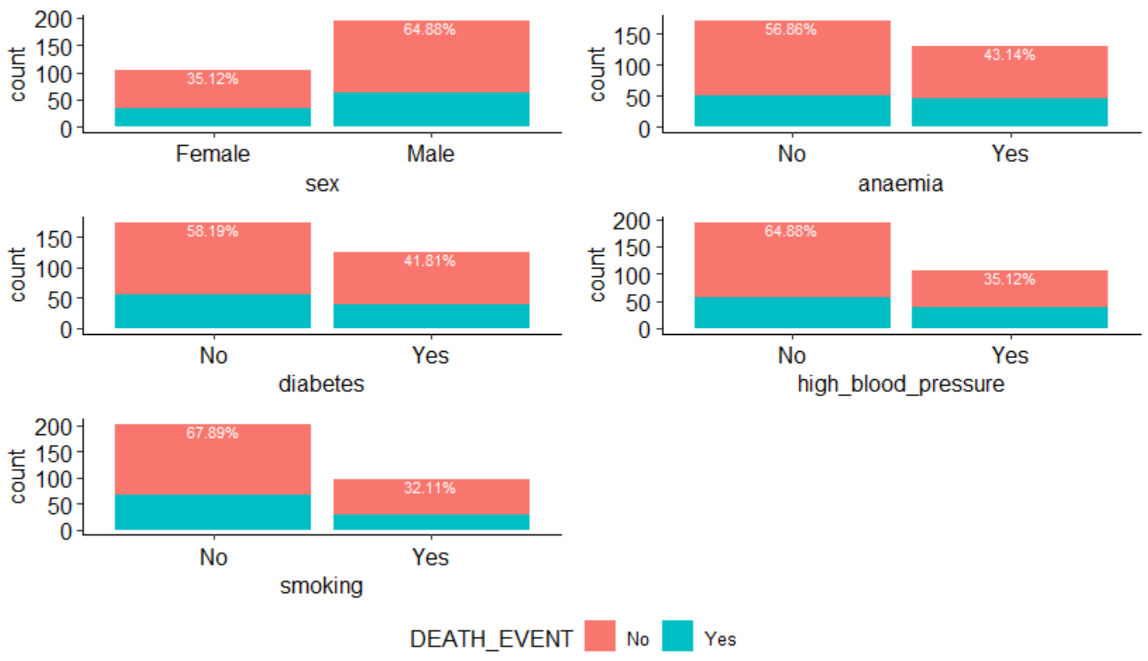
Categorical and numerical variables were analyzed separately.

Categorical variables

Death event is the binary response variable which and has 2 two different labels ‘Yes’ and ‘No’. Class distribution can be observed from the figures below. 67.89% of observations are labeled as ‘Yes’.

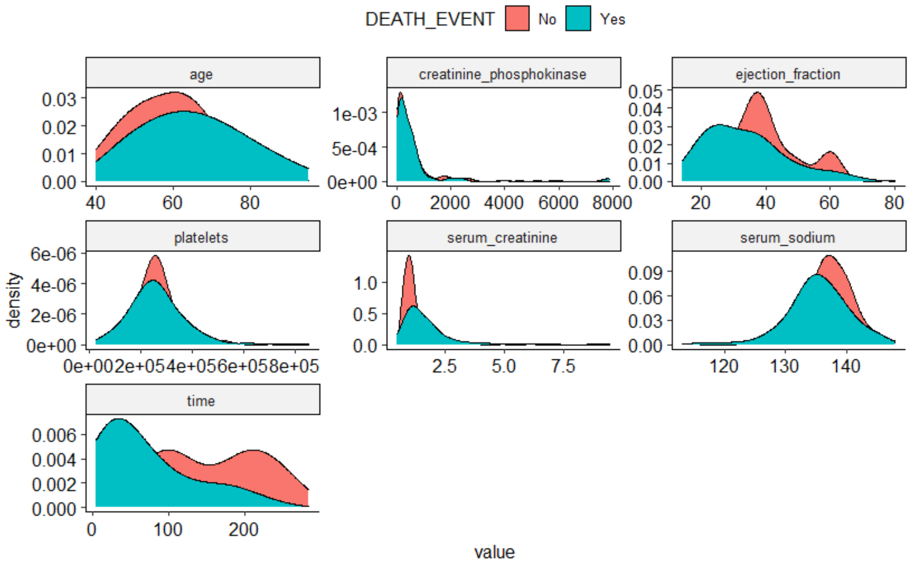


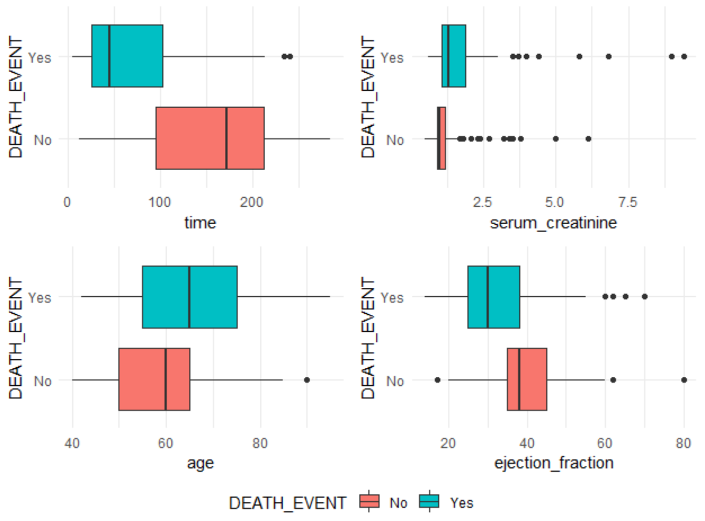
Other categorical variables are all binary as well and their distributions over the response variable as percentages are also shown below.



Numerical variables

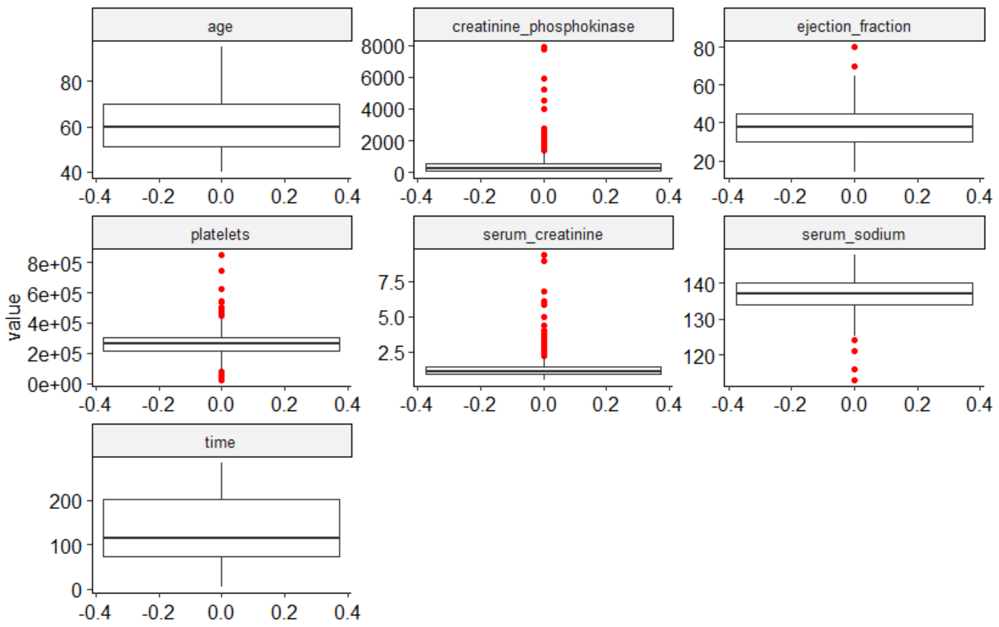
Density plots of the numerical variables were examined. These plots reveal that time, ejection_function and serum_creatinin differ by death event. Hence, these features were expected to be found significant by the models trained. By only looking at the density plots, other variables do not show any significant difference according to the death event.



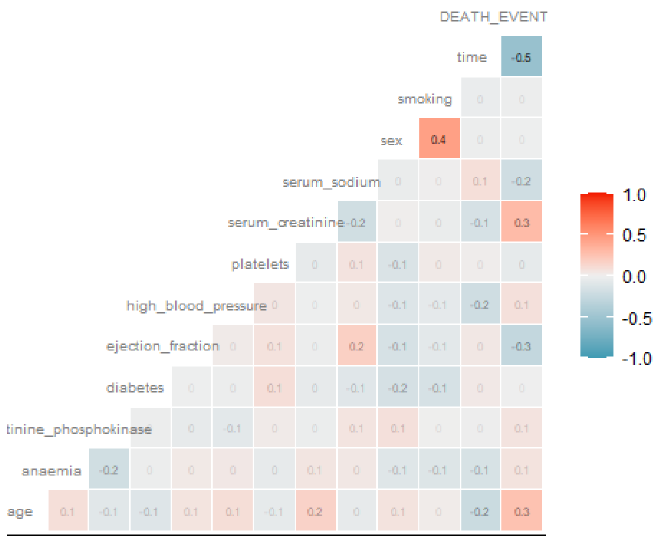


Boxplots also show that death event changes mostly by the follow up day. Serum_creatinine, age and ejection function has slight effects on the dependent variable.

To detect the outliers, boxplots were created. Red points mark the outliers which lie more than 1.5 times the interquartile range above the third quartile or below the first quartile.



It is also important to inspect the correlation between variables. Having or not having correlation affects the choice of models performed on the dataset. Hence, by looking at the correlation plot below, it can be said that predictors are uncorrelated. Only predictor pairs that seem to be slightly correlated are sex-smoking and time- death event.



Since there seems to be a little correlation between sex and smoking variables, it can be tested to check if it is valid or not. By producing a p-value less than 0.05, Chi-squared test result below proves that there is a valid correlation between these two variables.

```
                Pearson's Chi-squared test

data:  smoking and sex
X-squared = 59.447, df = 1, p-value = 1.256e-14
```

Standardization of Numerical Variables

Numerical variables were first standardized, since the values are in different scales, before applying Logistic Regression. Numerical variables were scaled and made to have 0 mean and 1 standard deviation. Furthermore, binary features were factorized and made suitable to be used in model training.

Methodology

In this study, methods used to build a classifier are Logistic Regression, Linear Discriminant Analysis, Decision Trees, Bagging and Random Forest.

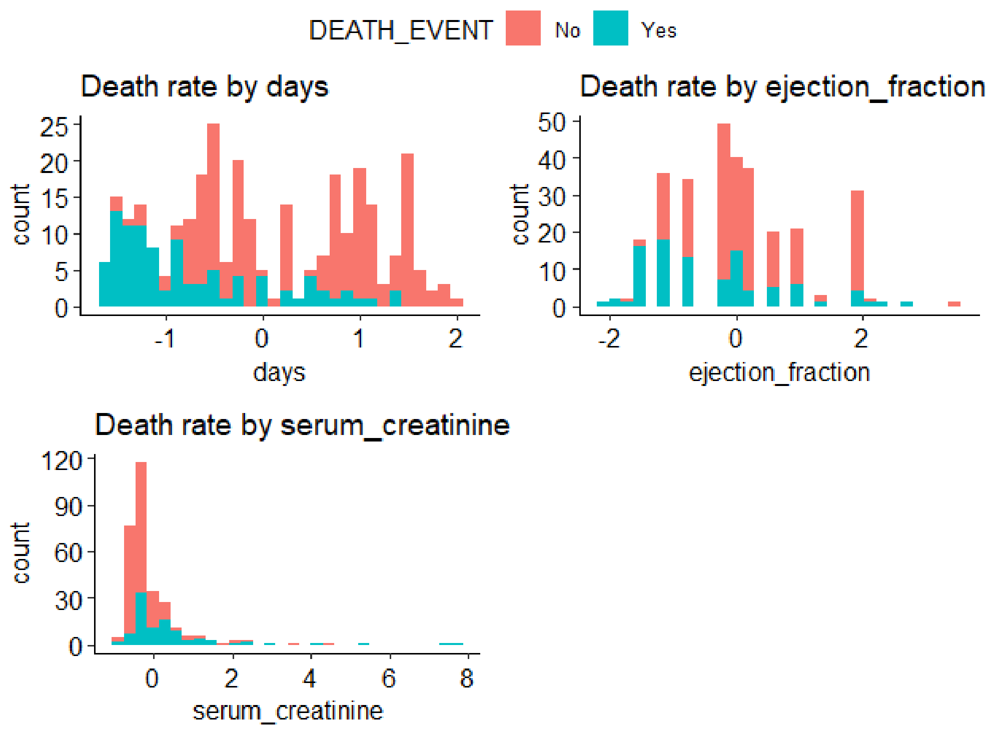
Before the application of methods mentioned above, data was split into train and test sets by the ratio of 0.75. Class balance in both sets was checked and ensured that it preserves the same ratio as the original dataset. (67.89% Yes, 32.11% No)

In order to validate the models trained on the train set, 10-fold cross validation was employed for each model. Whenever the model had any hyperparameters, appropriate tuning methods were performed in order to reach the best possible accuracy level.

Logistic Regression

Logistic regression model was fitted to the training set with 10-fold cross validation. The overall accuracy of the model is 84%. And the variables found significant by the model are ejection function, serum creatinine, time and serum sodium. This result is consistent with the expectation made while examining the density plots.

```
> summary(lr1)

Call:
NULL

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-2.1439  -0.5650  -0.2193   0.4678   2.4782  

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)    
(Intercept)              -0.85179    0.47397  -1.797 0.072313 .  
age                       0.37171    0.22108   1.681 0.092700 .  
creatinine_phosphokinase  0.22175    0.26002   0.853 0.393760    
ejection_fraction        -0.94223    0.23297  -4.044 5.25e-05 ***
platelets                -0.22401    0.25486  -0.879 0.379436    
serum_creatinine          0.64467    0.19422   3.319 0.000903 ***
serum_sodium             -0.40384    0.19832  -2.036 0.041722 *  
time                     -1.64096    0.27409  -5.987 2.14e-09 ***
anaemiaYes               -0.20233    0.41927  -0.483 0.629390    
diabetesYes              -0.05689    0.40283  -0.141 0.887695    
high_blood_pressureYes   -0.10884    0.41718  -0.261 0.794180    
sexMale                  -0.54533    0.48523  -1.124 0.261070    
smokingYes               -0.16818    0.47968  -0.351 0.725883    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```



Logistic regression diagnostics

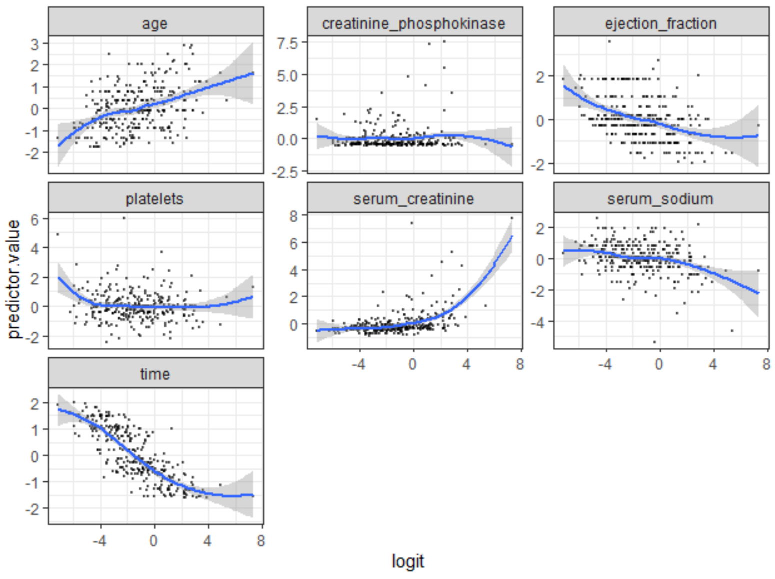
Assumption which are supposed to hold for the logistic regression model should be checked in order to have a good and reliable model.

Logistic Regression Assumptions:

- There is a linear relationship between the logit of the outcome and each predictor variables. Logit function is logit(p) = log(p/(1-p)), where p is the probabilities of the outcome
- Response variable is binary
- There is no influential values in the data
- There is no multicollinearity among predictors

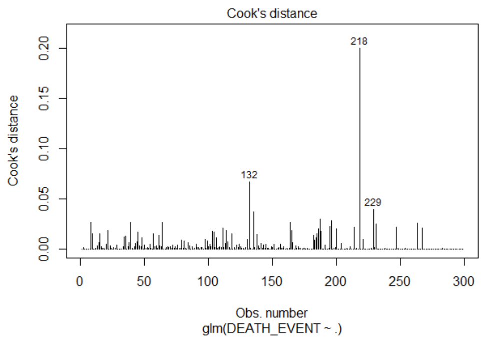
Linearity assumption

Scatter plots can be used to check whether linearity assumption holds for this dataset or not. Below, relationship between logit of the outcome and the predictors can be inspected. All of the contionus predictors except for serum creatinine have linear relationship with the logit of the outcome. Serum_creatinine needs to be transformed to obtain a linear relationship, and this transformation can also improve the model.



High Leverage Points:

In order to detect the high leverage points, the standardized residual error can be inspected. Data points with absolute standardized residuals above 3 are seen as possible outliers and should be checked. The data for the top 3 largest values, according to the Cook's distance can be seen below. When those data points were checked, it was seen that there is no influential observations in this data.

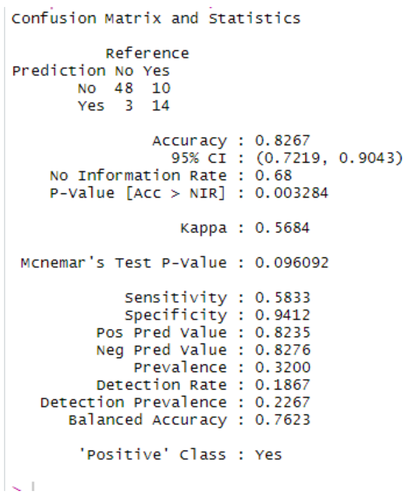


Multicollinearity

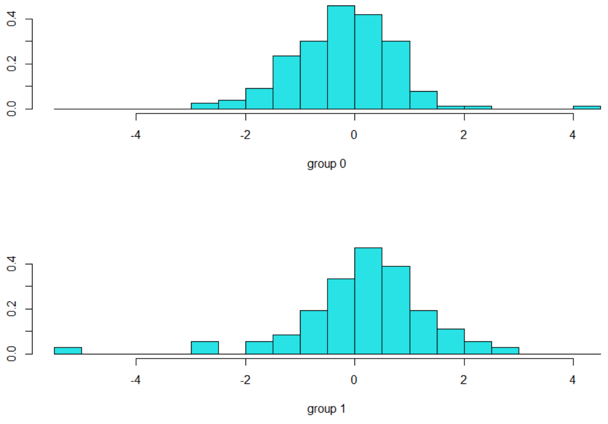
Multicollinearity occurs when the predictors are highly correlated with each other. Variance inflation factors were calculated for each predictor. None of the predictors have a VIF value greater than 5.Thus, it is clear that there is no multicollinearity in this data.

Linear Discriminant Analysis

Another method applied is Linear Discriminant Analysis. Again, the dataset is split into two and model was fitted to the train set and validated with 10-fold cross validation. Accuracy score reached with this model is 82.67%. Figures below depict the groups which were separated by the classifier.



```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  48  10
       Yes  3  14

               Accuracy : 0.8267
                 95% CI : (0.7219, 0.9043)
    No Information Rate : 0.68
    P-Value [Acc > NIR] : 0.003284

                  Kappa : 0.5684

 Mcnemar's Test P-Value : 0.096092

            Sensitivity : 0.5833
            Specificity : 0.9412
         Pos Pred Value : 0.8235
         Neg Pred Value : 0.8276
             Prevalence : 0.3200
         Detection Rate : 0.1867
   Detection Prevalence : 0.2267
      Balanced Accuracy : 0.7623

       'Positive' Class : Yes
```



Penalized Logistic Regression - LASSO

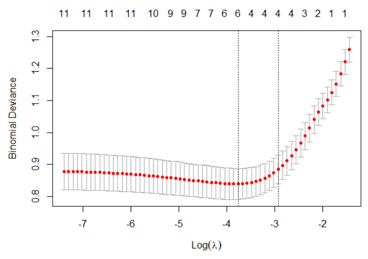
The lasso shrinks the coefficient estimates towards zero. Therefore, it also serves as a variable selection method. When λ = 0, then the lasso simply gives the least squares fit, and when λ becomes sufficiently large, the lasso gives the null model in which all coefficient estimates equal zero.

Model fitted to the training data and validated by 10 fold cross validation. Different lambda values were searched by the algorithm the one which maximizes the accuracy was selected.

Best lambda given by the algorithm is 0.02154 and the variables included in the final model are age, creatinine phosphokinase, ejection fraction, platelets, serum creatinine,serum sodium and time. When compared with the variables found weakly or strongly significant by the logistic regression,the only different variable is platelets.

```
                                   1
(Intercept)              -1.13667066
age                       0.17799856
creatinine_phosphokinase  0.00865856
ejection_fraction        -0.59574291
platelets                -0.01533172
serum_creatinine          0.47216810
serum_sodium             -0.26679469
time                     -1.24299816
anaemiaYes                .
diabetesYes               .
high_blood_pressureYes    .
sexMale                   .
smokingYes                .
```

The overall accuracy achieved with lasso model is 85.33% which is slightly better than the one given by the logistic regression model.



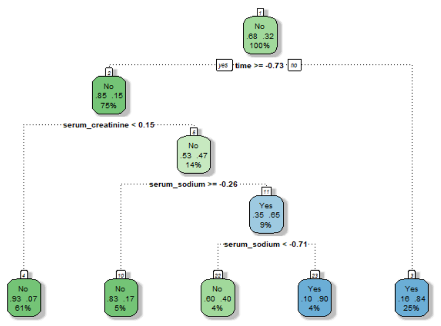
Binomial Deviance vs. Log(λ)

Decision Tree Classification

Decision trees are useful and easy to interpret. In this analysis, for the sake of interpretation and simplicity, decision trees were also employed. However, they might not always lead to the best accuracy. Therefore , in addition to decision trees, some ensemble methods such as bagging and random forests were also performed in order to improve the prediction accuracy.

Two different split criteria were taken into consideration while building decision tree models: GINI index and entropy. These two indices are used to calculate the information gain. **Gini Index** has values inside the interval [0, 0.5] *where 0 expresses the purity of classification* While designing the decision tree, the features possessing the least value of the Gini Index would get preferred whereas the interval of the **Entropy** is [0, 1]. Information Gain is applied to quantify which feature provides maximal information about the classification based on the notion of entropy

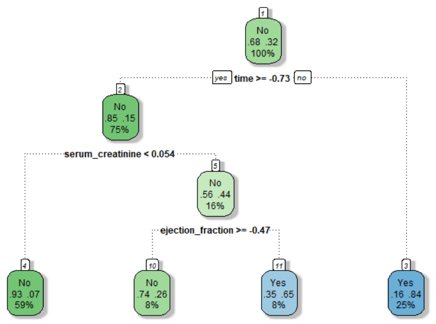
Tree model using GINI index
Accuracy : 82.67%



Tree based on GINI Index

Sensitivity :70.83 % Specificity : 88.24 %

Tree model using Entropy index
Accuracy : 81.33%



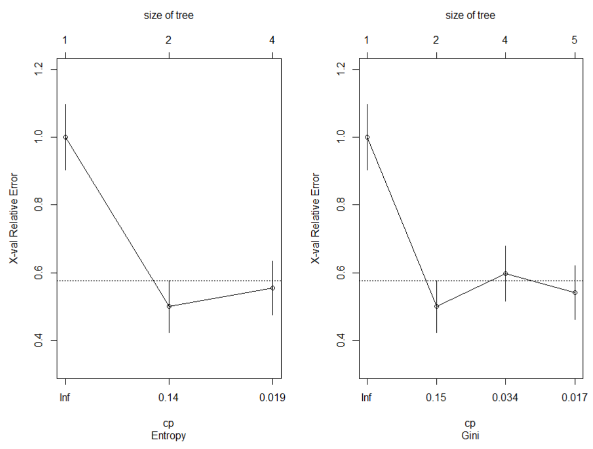
Tree based on Entropy Index

Sensitivity : 87.5% Specificity : 78.43 %

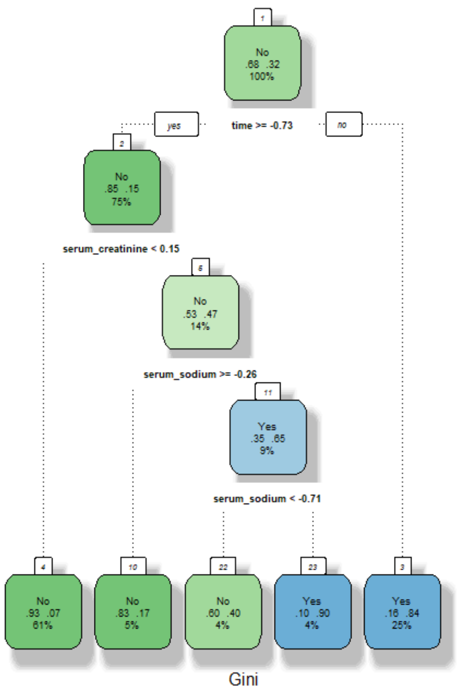
Both models have similar overall accuracy rates. However, sensitivity and specifity metrics differ in opposite directions.

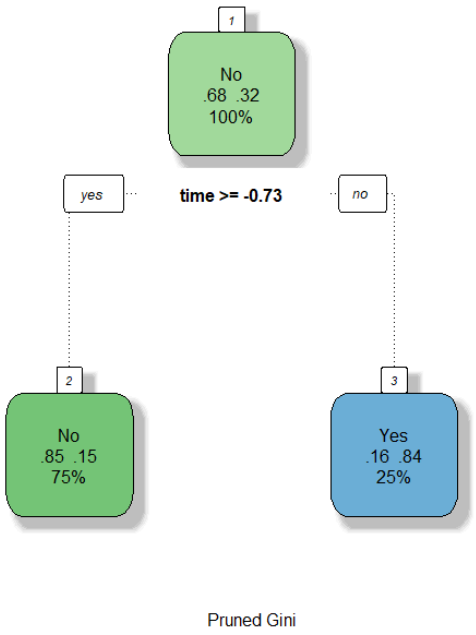
Pruning the trees

With the aim of avoiding overfitting, trees were pruned based on the complexity parameter with the lowest cross validated error.

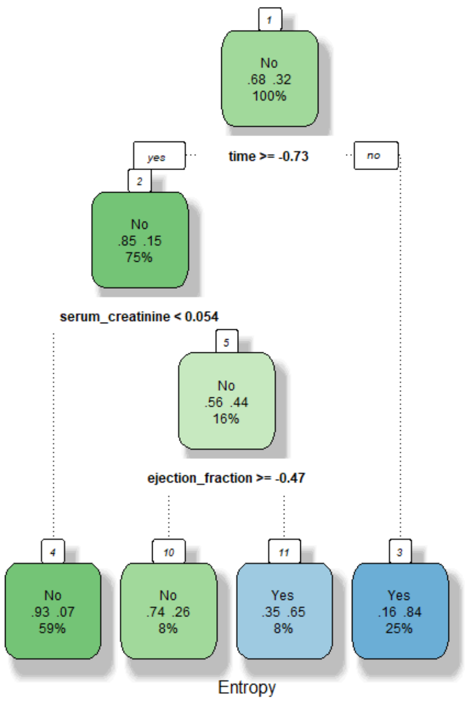


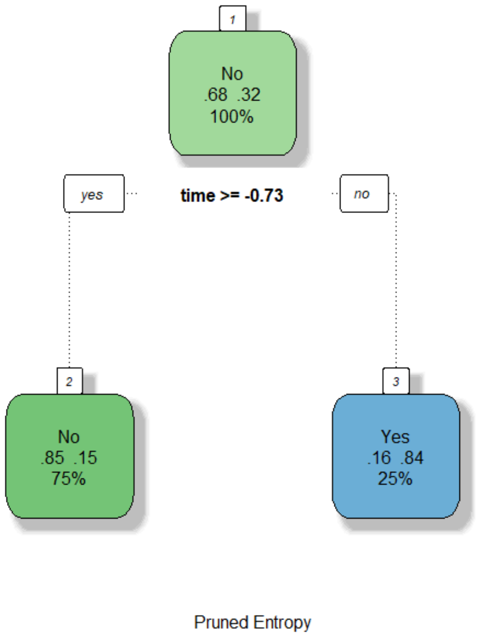
Pruning trees actually gave slightly better accuracy rates and simpler trees. Accuracy rates of both pruned trees are 84%





Pruned Tree Model Using GINI Index

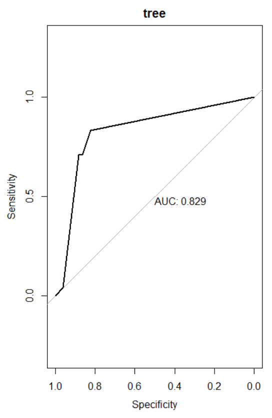


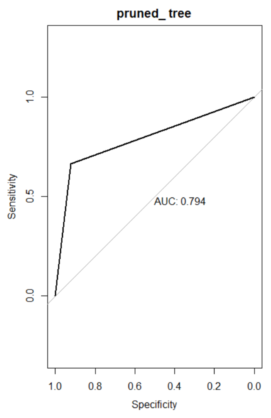


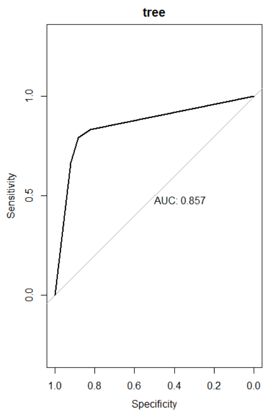
Pruned Tree Model Using Entropy

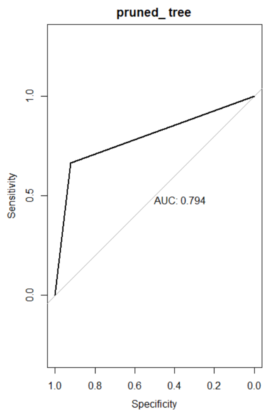
ROC Curves

GINI Index — Entropy









AUC for

- Tree model using GINI Index: 0.83
- Pruned :0.794

AUC for

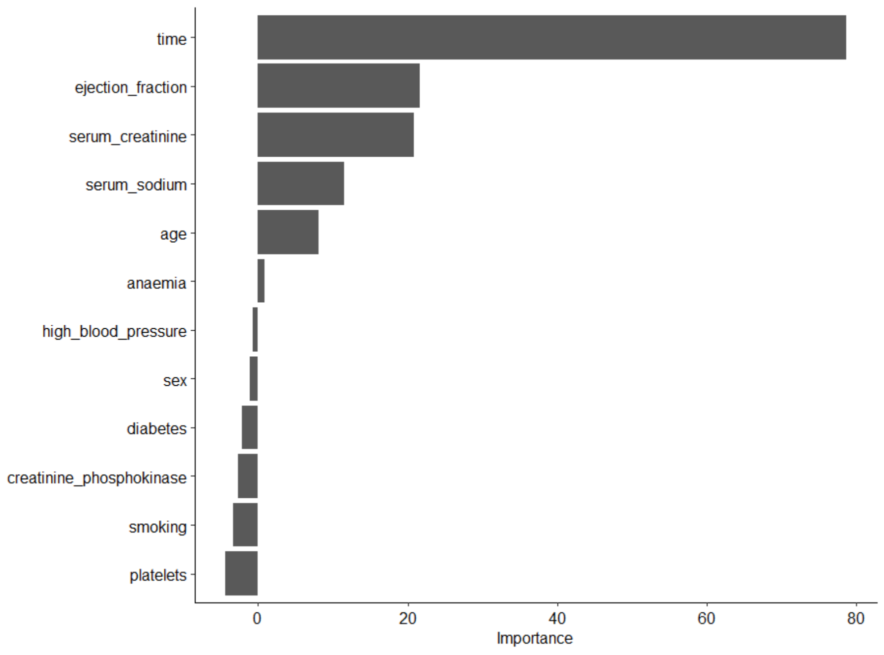
- Tree model using Entropy: 0.857
- Pruned: 0.794

AUC values for both tree models, either pruned or not, indicate that these classifiers are both good.

Bagging

Bagging classifier was built with 1000 trees. OOB estimate of error rate: 18.75% and the over all accuracy is 85.33%. Number of trees can be increased without the threat of overfitting.

Bagging improved the accuracy rates of the single trees built based on two different split criteria. Variables ranked according to their importance are below:



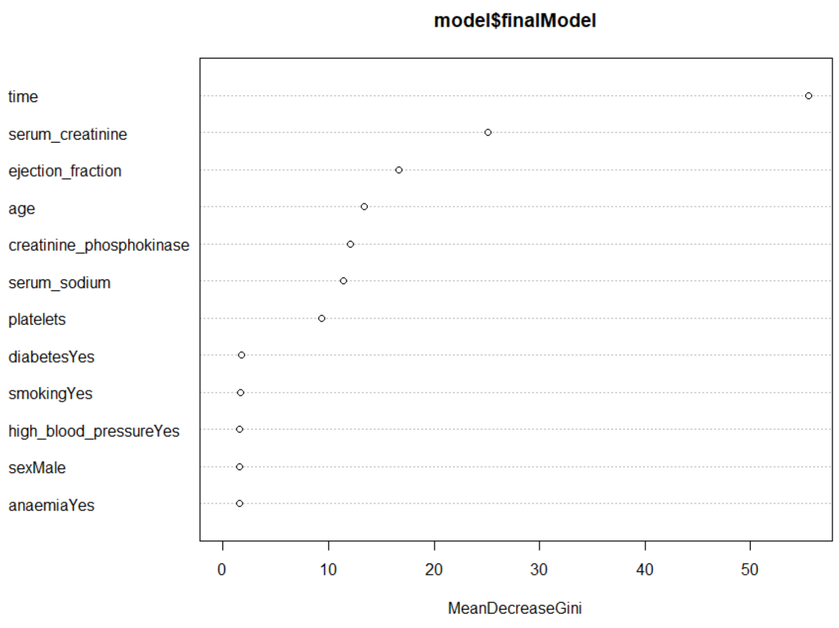
Random Forest

Random forest allows to decorrelate the bagged tree by limiting the selection of parameter at each step. Basically, bagging is random forest built with the same amount of the parameters as the predictors. As in the case of bagging, number of trees does not lead to overfitting with random forest. Therefore, in this analysis, random forest classifier was built with 1000 trees and validated using 10 fold cross validation. Model was fitted to the oversampled training set. Mtry is the number of predictors picked at each split. In order to find the optimal number of mtry , additional hypermeter tuning was performed.
Final model has the accuracy rate of 81.33% , OOB estimate of error rate 6.91%. Highest accuracy was reached when mtry was 4.
For a given tree, the out-of-bag (OOB) error is the model error in predicting the data left out of the training set for that tree (P. Bruce and Bruce 2017). OOB is a very straightforward way to estimate the test error of a bagged model, without the need to perform cross-validation or the validation set approach.

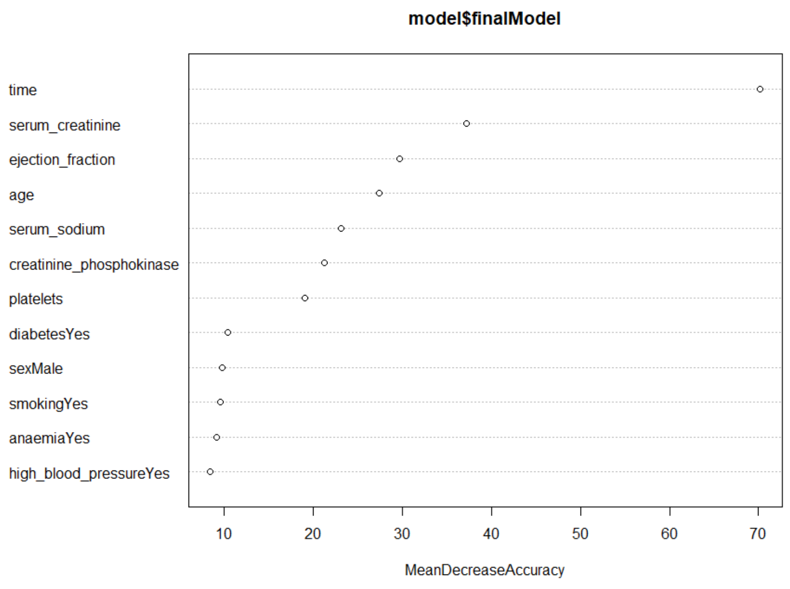
Variable importance

```
                          Importance
time                        100.0000
serum_creatinine             41.6229
ejection_fraction            33.2527
age                          25.1893
serum_sodium                 16.5868
creatinine_phosphokinase     14.8547
platelets                    12.5213
diabetesYes                   2.5170
sexMale                       1.0069
anaemiaYes                    0.8154
smokingYes                    0.2896
high_blood_pressureYes        0.0000
```

Importantant variables based on mean decreases in accuracy and Gini are as follows:



Mean Decrease Gini



Mean Decrease Accuracy

Conclusion

- ✓ Overall accuracy rates of the models built in this study are listed below. Since the purpose of this analysis was to find the classifier having the best predictive power, overall accuracy rates were considered as a relevant metric to examine.

- ✓ LASSO and Bagging were found to be the models giving the largest accuracy rates among others. However, as seen below, all the models included in this study achieved a good performance with having overall accuracy rates above 80%.

	Accuracy (%)
RANDOM FOREST	81.33
LINEAR DISCRIMINANT ANALYSIS	82.67
DECISION TREES	82.67
LOGISTIC REGRESSION	84
LASSO	85.33
BAGGING	85.33

- ✓ Time, serum_creatinine,ejection_function, serum_sodium and age were found significant and to have great influence by almost all models trained.

Appendix: R Code

```
# loading the libraries
library(car)
library(ggplot2)
library(ggpubr)
theme_set(theme_pubr())
library(tidyverse)
library(hrbrthemes)
library(dplyr)
library(tidyr)
library(viridis)
library(gplots)
library(plyr)

#loading the data

data=read.csv("heart_failure_clinical_records_dataset.csv")
View(data)


dim(data)
sum(is.na(data)) # no N/A values

#data preprocessing
str(data_clean)

data_clean <- data[complete.cases(data), ]

data_clean$anaemia <- as.factor(mapvalues(data_clean$anaemia,
                              from=c("0","1"),
                              to=c("No", "Yes")))
data_clean$diabetes <- as.factor(mapvalues(data_clean$diabetes,
                               from=c("0","1"),
                               to=c("No", "Yes")))

data_clean$high_blood_pressure <- as.factor(mapvalues(data_clean$high_blood_pressure,
                                          from=c("0","1"),
                                          to=c("No", "Yes")))
data_clean$smoking <- as.factor(mapvalues(data_clean$smoking,
                              from=c("0","1"),
                              to=c("No", "Yes")))
data_clean$sex <- as.factor(mapvalues(data_clean$sex,
                          from=c("0","1"),
                          to=c("Female", "Male")))
data_clean$DEATH_EVENT <- as.factor(mapvalues(data_clean$DEATH_EVENT,
                                  from=c("0","1"),
                                  to=c("No", "Yes")))
```

```
attach(data_clean)

############ Visualising distributions ##################
##Categorical Variables
## DEATH_EVENT##

p1<-ggplot(data=data_clean,aes(x = DEATH_EVENT)) +
  geom_bar(stat="count",width=0.7,aes(fill = DEATH_EVENT))+
  geom_text(stat='count', aes(label=..count..), vjust=1,colour="white")

p1.1<-ggplot(data_clean, aes(x = DEATH_EVENT)) +
  geom_bar(aes(fill = DEATH_EVENT)) +
  geom_text(aes(y = ..count.. ,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            vjust=1,colour="white")


figure <- ggarrange(p1,p1.1,
                    labels = c("Count", "Percentage"),
                    ncol = 2, nrow = 1,
                    common.legend = TRUE)
figure

##others
require(reshape2)

melt.data = melt(select(data_clean,c(2,4,6,10,11,13)),id.vars = "DEATH_EVENT")
head(melt.data)

data %>%
ggplot(data = melt.data, mapping= aes(x = value)) +
  geom_bar(stat="count",width=0.7,fill="#69b3a2") +
  geom_text(stat='count', aes(label=..count..), vjust=1,colour="white")+
  facet_wrap(~ variable, scales = "free")

##########
### Categorical Variables ###
b1<-  ggplot(data_clean, aes(x =sex)) +
  geom_bar(aes(fill = DEATH_EVENT)) +
  geom_text(aes(y = ..count..,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            vjust=1,colour="white",
            position = position_dodge(.1),
            size = 3)


b2<-  ggplot(data_clean, aes(x =anaemia)) +
  geom_bar(aes(fill = DEATH_EVENT)) +
  geom_text(aes(y = ..count..,
            label = paste0(round(prop.table(..count..),4) * 100, '%')),
         stat = 'count',
         vjust=1,colour="white",
         position = position_dodge(.1),
         size = 3)


b3<-  ggplot(data_clean, aes(x =diabetes)) +
  geom_bar(aes(fill = DEATH_EVENT)) +
  geom_text(aes(y = ..count..,
            label = paste0(round(prop.table(..count..),4) * 100, '%')),
         stat = 'count',
         vjust=1,colour="white",
         position = position_dodge(.1),
         size = 3)

b4<-  ggplot(data_clean, aes(x =high_blood_pressure)) +
  geom_bar(aes(fill = DEATH_EVENT)) +
  geom_text(aes(y = ..count..,
            label = paste0(round(prop.table(..count..),4) * 100, '%')),
         stat = 'count',
         vjust=1,colour="white",
         position = position_dodge(.1),
         size = 3)

b5<- ggplot(data_clean, aes(x =smoking)) +
  geom_bar(aes(fill = DEATH_EVENT)) +
  geom_text(aes(y = ..count..,
            label = paste0(round(prop.table(..count..),4) * 100, '%')),
         stat = 'count',
         vjust=1,colour="white",
         position = position_dodge(.1),
         size = 3)

figure2 <- ggarrange(b1,b2,b3,b4,b5,
                 #labels = c("Sex", "Anaemia","Diabetes",
                 #          "High Blood Pressure","Smoking"),
                 ncol = 2, nrow = 3,
                 common.legend = TRUE, legend = "bottom")
figure2
########################################################################
##Numeric variables
#density plots #

require(reshape2)

melt.data = melt(select(data_std,-c(2,4,6,10,11)))
ggplot(data = melt.data, aes(x = value, fill=DEATH_EVENT)) +
  geom_density(adjust=1.5) +
  facet_wrap(~variable, scales = "free")
```

```
#boxplots # --- Identifying outliers ---
melt.data = melt(data_clean)
head(melt.data)

ggplot(data = melt.data, aes(y=value)) +
  geom_boxplot(outlier.color="red") +
  facet_wrap(~variable, scales = "free")

####


##Covariation###

c<-ggplot(data = data_clean, mapping = aes(x = DEATH_EVENT, y = time,fill=DEATH_EVENT)) +
  geom_boxplot()+ coord_flip()+theme_minimal()

c1<-ggplot(data = data_clean, mapping = aes(x = DEATH_EVENT, y =
serum_creatinine,fill=DEATH_EVENT)) +
  geom_boxplot()+ coord_flip()+theme_minimal()
c2<-ggplot(data = data_clean, mapping = aes(x = DEATH_EVENT, y = age,fill=DEATH_EVENT)) +
  geom_boxplot()+ coord_flip()+theme_minimal()
c3<-ggplot(data = data_clean, mapping = aes(x = DEATH_EVENT, y =
ejection_fraction,fill=DEATH_EVENT)) +
  geom_boxplot()+ coord_flip()+theme_minimal()

figure3 <- ggarrange(c,c1,c2,c3,
                    #labels = c("Sex", "Anaemia","Diabetes",
                    #          "High Blood Pressure","Smoking"),
                    ncol = 2, nrow = 2,
                    common.legend = TRUE, legend = "bottom")
figure3


## correlations ##
###

library(GGally)
ggcorr(data, method = c("everything", "pearson"),
       hjust = 0.75, size = 2.75, color = "grey50",
       label_alpha= TRUE,
       label = TRUE, label_size = 2, layout.exp= 0,)

a=chisq.test(smoking, sex, correct=FALSE)

##########################################

## Logistic Regression ##

data_clean <- as.data.frame(
  cbind(scale(data_clean[,c(1,3,5,7,8,9,12)]),data_clean[,c(2,4,6,10,11,13)]))
str(data_clean)
summary(data_clean)
```

```
###Divide the dataset
set.seed(123)
split = sample.split(data_clean$DEATH_EVENT, SplitRatio = 0.75)
data.train = subset(data_clean, split == TRUE)
data.test = subset(data_clean, split == FALSE)
performance_metric="Accuracy"

# Check class balance
table(data_clean$DEATH_EVENT, dnn="Overall")

table(data.train$DEATH_EVENT, dnn="training")

table(data.test$DEATH_EVENT, dnn="testing")



## Logistic Regression

# train the model on training set
lr1 <- train(DEATH_EVENT ~ .,
             data = data.train,
             trControl = trainControl(method = "cv", number = 10),metric = performance_metric,
             method = "glm",
             family=binomial())

lr_pred1 =lr1 %>% predict(data.test)

# Model accuracy
observed.classes <- data.test$DEATH_EVENT
table(Predicted = lr_pred1, Actual = data.test$DEATH_EVENT)
mean(predicted.classes == observed.classes)

summary(lr1)

# Test
confusionMatrix(
  as.factor(lr_pred1),
  as.factor(data.test$DEATH_EVENT),
  positive = "Yes" )

# ROC
test_roc = roc(data.test$DEATH_EVENT ~ probabilities, plot = TRUE, print.auc = TRUE)
as.numeric(test_roc$auc)


######################
# relation with days
p23 <- ggplot(data_clean, aes(x = time , fill = DEATH_EVENT)) +
  geom_histogram() +
  labs(x = "days",
       title = "Death rate by days")
p23
```

```
# relation with days ejection_fraction"
p24 <- ggplot(data_clean, aes(x = ejection_fraction , fill = DEATH_EVENT)) +
  geom_histogram() +
  labs(x = "ejection_fraction",
       title = "Death rate by ejection_fraction")
p24
#relation with days serum_creatinine
p25 <- ggplot(data_clean, aes(x = serum_creatinine , fill = DEATH_EVENT)) +
  geom_histogram() +
  labs(x = "serum_creatinine",
       title = "Death rate by serum_creatinine")
p25
figure <- ggarrange(p23,p24,p25,
                    ncol = 2, nrow = 2,
                    common.legend = TRUE)
figure

# Penalized Logistic Regression - LASSO

# Dumy code categorical predictor variables
x <- model.matrix(DEATH_EVENT~., data.train)[,-13]
# Convert the outcome (class) to a numerical variable
y <- ifelse(data.train$DEATH_EVENT == "Yes", 1, 0)

##LASSO

# Find the best lambda using cross-validation
set.seed(123)
cv.lasso = cv.glmnet(x, y, alpha = 1, family = "binomial")
plot(cv.lasso)
cv.lasso$lambda.min

###
lambda <- 10^seq(-5, 5, length = 100)
set.seed(123)
lasso.fit= train(
  DEATH_EVENT ~., data = data.train, method = "glmnet",
  trControl = trainControl("cv", number = 10),metric = performance_metric,
  tuneGrid = expand.grid(alpha = 1, lambda = lambda)
)
# Model coefficients
coef(lasso.fit$finalModel, lasso.fit$bestTune$lambda)
lasso.fit$bestTune$lambda
# Make predictions on the test data
lasso.fit
predicted.classes =lasso.fit%>% predict(data.test)

# Model accuracy
observed.classes <- data.test$DEATH_EVENT
table(Predicted = predicted.classes, Actual = data.test$DEATH_EVENT)
# Test
confusionMatrix(
  as.factor(predicted.classes),
  as.factor(observed.classes),
  positive = "Yes" )

#####################################################
# Logistic regression diagnostics
##Linearity assumption
model <- glm(DEATH_EVENT ~., data = data_clean,
          family = binomial)
# Predict the probability (p) of death event
probabilities <- predict(model, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, "Yes", "No")

# Select only numeric predictors
mydata <- data_clean %>%
  dplyr::select_if(is.numeric)
predictors <- colnames(mydata)
# Bind the logit and tidying the data for plot
mydata <- mydata %>%
  mutate(logit = log(probabilities/(1-probabilities))) %>%
  gather(key = "predictors", value = "predictor.value", -logit)

#2. Create the scatter plots:

ggplot(mydata, aes(logit, predictor.value))+
  geom_point(size = 0.5, alpha = 0.5) +
  geom_smooth(method = "loess") +
  theme_bw() +
  facet_wrap(~predictors, scales = "free_y")

# Influential values

plot(model, which = 4, id.n = 3)

library(broom)
# Extract model results
model.data <- augment(model) %>%
  mutate(index = 1:n())
model.data %>% top_n(3, .cooksd)
#Plot the standardized residuals:

ggplot(model.data, aes(index, .std.resid)) +
  geom_point(aes(color = diabetes), alpha = .5) +
  theme_bw()
#Filter potential influential data points
model.data %>%
  filter(abs(.std.resid) > 3)
#There is no influential observations in our data.
# Multicollinearity
car::vif(model)

#In our example,
#there is no collinearity: all variables have a value of VIF well below 5.

##Linear Discriminat Analysis

set.seed(123)
lda= train(
  DEATH_EVENT ~., data = data.train, method = "lda",metric = performance_metric,
  trControl = trainControl("cv", number = 10)
)
# Make predictions on the test data

predicted.classes =lda %>% predict(data.test)

# Model accuracy
observed.classes <- data.test$DEATH_EVENT
table(Predicted = predicted.classes, Actual = data.test$DEATH_EVENT)

# Test
confusionMatrix(
  as.factor(predicted.classes),
  as.factor(observed.classes),
  positive = "Yes" )
plot(lda$finalModel)

################################################################################
# Decision Trees
library(rpart)

# Build a tree model using GINI index
tree_gini = rpart(DEATH_EVENT ~ .,
              data = data.train,
              method = "class",
              control = rpart.control(cp = 0.01),
              parms = list(split="gini"))

# Build a tree model using Entropy index
tree_entropy = rpart(DEATH_EVENT ~ .,
              data = data_oversample,
              method = "class",
              control = rpart.control(cp=0.01),
              parms = list(split="information"))

# Ploting tree
library(rattle)
fancyRpartPlot(tree_gini,
            sub = "Tree based on GINI Index")
fancyRpartPlot(tree_entropy,
            sub = "Tree based on Entropy Index")


#predictions:
```

```
pred_gini = predict(tree_gini,
                data.test,
                type = "class")


pred_ent = predict(tree_entropy,
                data.test,
                type = "class")
# assesing the trees

cm_gini = table(data.test$DEATH_EVENT, pred_gini)
cm_ent = table(data.test$DEATH_EVENT, pred_ent)

cf_gini=confusionMatrix(pred_gini,data.test$DEATH_EVENT, positive = "Yes")# %82
cf_ent=confusionMatrix(pred_ent,data.test$DEATH_EVENT, positive = "Yes") #85.33
cf_gini
cf_ent

## Possible pruning
# To avoid over-fitting  prune trees based on the value of
# cp that minimizes cross-validated error.

# Cp table for tree_gini
printcp(tree_gini)
printcp(tree_entropy)
# Cross-valiedation error plot for d1
par(mfrow=c(1,2))
plotcp(tree_entropy, sub = "Entropy")
plotcp(tree_gini, sub = "Gini")

# Pruning tree for d1
pruned_gini = prune(tree_gini, cp = tree_gini$cptable[which.min(tree_gini$cptable[,"xerror"]),"CP"])
pruned_ent = prune(tree_entropy, cp =
tree_entropy$cptable[which.min(tree_entropy$cptable[,"xerror"]),"CP"])

par(mfrow = c(1,2))
# Ploting trees for Gini
fancyRpartPlot(tree_gini,
            sub = "Gini")
fancyRpartPlot(pruned_gini,
            sub = "Pruned Gini")

# Ploting trees for Entropy
fancyRpartPlot(tree_entropy,
            sub = "Entropy")
fancyRpartPlot(pruned_ent,
            sub = "Pruned Entropy")
# Assesing pruned trees

# Using our model to predict the test set
pred_pruned_g = predict(pruned_gini,
                    data.test,
                    type = "class")

pred_pruned_e = predict(pruned_ent,
                    data.test,
                    type = "class")

# Construct the confusion matrices
cf_pruned_g=confusionMatrix(pred_pruned_g,data.test$DEATH_EVENT, positive = "Yes")# %82
cf_pruned_e=confusionMatrix(pred_pruned_e,data.test$DEATH_EVENT, positive = "Yes") #81
cf_pruned_g$overall['Accuracy']
cf_pruned_e$overall['Accuracy']

# ROC Curves
## Gini Index
pred_gini = predict(tree_gini,
               data.test,
               type = "prob")[, 2]
test_roc = roc(data.test$DEATH_EVENT ~ pred_gini, plot = TRUE, print.auc = TRUE,
           main = "tree")
as.numeric(test_roc$auc)

pred_pruned_g =predict(pruned_gini,
                 data.test,
                 type = "prob")[, 2]
test_roc = roc(data.test$DEATH_EVENT ~ pred_pruned_g, plot = TRUE, print.auc = TRUE,
           main = "pruned_ tree")
as.numeric(test_roc$auc)

## Entropy Index

pred_ent = predict(tree_entropy,
               data.test,
               type = "prob")[, 2]
test_roc = roc(data.test$DEATH_EVENT ~ pred_ent, plot = TRUE, print.auc = TRUE,
           main = "tree")
as.numeric(test_roc$auc)

pred_pruned_e =predict(pruned_ent,
                 data.test,
                 type = "prob")[, 2]
test_roc = roc(data.test$DEATH_EVENT ~ pred_pruned_e, plot = TRUE, print.auc = TRUE,
           main = "pruned_ tree")
as.numeric(test_roc$auc)
#################################
# bagging classifer
library(randomForest)
set.seed (123)

bag.hf =randomForest(DEATH_EVENT~.,data.train ,
                    mtry=12,ntree=1000, importance =TRUE)
bag.hf
bag.pred=predict (bag.hf ,data.test ,type ="class")
confusionMatrix(bag.pred,data.test[,13],positive = "Yes")
library(vip)
vip::vip(bag.hf, num_features = 15)

# increasing ntree does not lead to overfitting

#random forest classifier

# Fit the model on the training set
set.seed(123)

rf = train(DEATH_EVENT ~., data = data.train, method = "rf",
        trControl = trainControl("cv", number = 10,sampling = "up"),
        metric = performance_metric,tuneGrid=data.frame(.mtry=c(2:6)),
        n.tree=1000,importance = TRUE)


# Best tuning parameter
rf$bestTune #The optimal number of variables sampled at each split is 2
# Final model
rf$finalModel
# Make predictions on the test data
predicted.classes <- rf %>% predict(data.test)
rf
# Compute model accuracy rate

#For a given tree, the out-of-bag (OOB) error is the model error in predicting
#the data left out of the training set for that tree (P. Bruce and Bruce 2017).
#OOB is a very straightforward way to estimate the test error of a bagged model,
#without the need to perform cross-validation or the validation set approach.

# variable importance
importance(rf$finalModel)

par(mfrow=c(1,1))
# Plot MeanDecreaseAccuracy
varImpPlot(rf$finalModel, type = 1)
# Plot MeanDecreaseGini
varImpPlot(rf$finalModel, type = 2)
varImp(rf)
confusionMatrix(predicted.classes,data.test[,13],positive='Yes')
```