**Supervised Learning Analyses on Survival of Heart Failure Patients**

Name : Ahmad Luqman Al Hakim

Surname : bin Shamsudin

**Abstract**

This report employs supervised learning algorithms to analyse the Survival of Heart Failure Patients. The dataset used contains 13 variables of patients physical and lifestyle information. The analyses are performed using R programming language. We employ two different supervised learning algorithms for the analyses which are the logistic regression and the random forest. The methods are implemented for prediction and the accuracy of each method is measured and compared. According to the findings, both algorithms perform very well at predicting the survival of heart failure patients.

**Section 1: Data Descriptions**

In this paper, we analyse a dataset of 299 patients with heart failure collected in 2015 sourced from the Public Library of Science[[1]](#footnote-1), under the Creative Commons license. The dataset contains the medical records of 299 heart failure patients collected at the Faisalabad Institute of Cardiology and at the Allied Hospital in Faisalabad (Pakistan), during April–December 2015. The patients consisted of 105 women and 194 men, aged between 40 and 95 years old. All patients had left ventricular systolic dysfunction and had previous heart failures.

The dataset contains 13 features, which report clinical, body, and lifestyle information, that we briefly describe here. Some features are binary e.g., anaemia, high blood pressure, diabetes, sex, and smoking.

| **Feature** | **Explanation** | **Measurement** | **Range** |
| --- | --- | --- | --- |
| Age | Age of the patient | Years | [40,..., 95] |
| Anaemia | Decrease of red blood cells or hemoglobin | Boolean | 0, 1 |
| High blood pressure | If a patient has hypertension | Boolean | 0, 1 |
| Creatinine phosphokinase | Level of the CPK enzyme in the blood | micrograms per litre | [23,..., 7861] |
| Diabetes | If the patient has diabetes | Boolean | 0, 1 |
| Ejection fraction | Percentage of blood leaving  the heart at each contraction | Percentage | [14,..., 80] |
| Sex | Woman or man | Binary | 0, 1 |
| Platelets | Platelets in the blood | kiloplatelets/microlitre | [25.01,..., 850.00] |
| Serum creatinine | Level of creatinine in the blood | mg/decilitre | [0.50,..., 9.40] |
| Serum sodium | Level of sodium in the blood | miliequivalents/litre | [114,..., 148] |
| Smoking | If the patient smokes | Boolean | 0, 1 |
| Time | Follow-up period | Days | [4,...,285] |
| Death event | If the patient died during the follow-up period | Boolean | 0, 1 |

Here, we briefly explain the meaning behind key features such as anaemia, creatinine\_phosphokinase, ejection\_fraction, serum\_creatinine and serum\_sodium.

The hospital physician considered a patient having anaemia (anaemia = 1) if haematocrit level is lower than 36%.

Next, the creatinine phosphokinase (CPK) represents the level of the CPK enzyme in blood. When a muscle tissue gets damaged, CPK flows into the blood. Therefore, high levels of CPK in the blood of a patient might indicate a heart failure or injury.

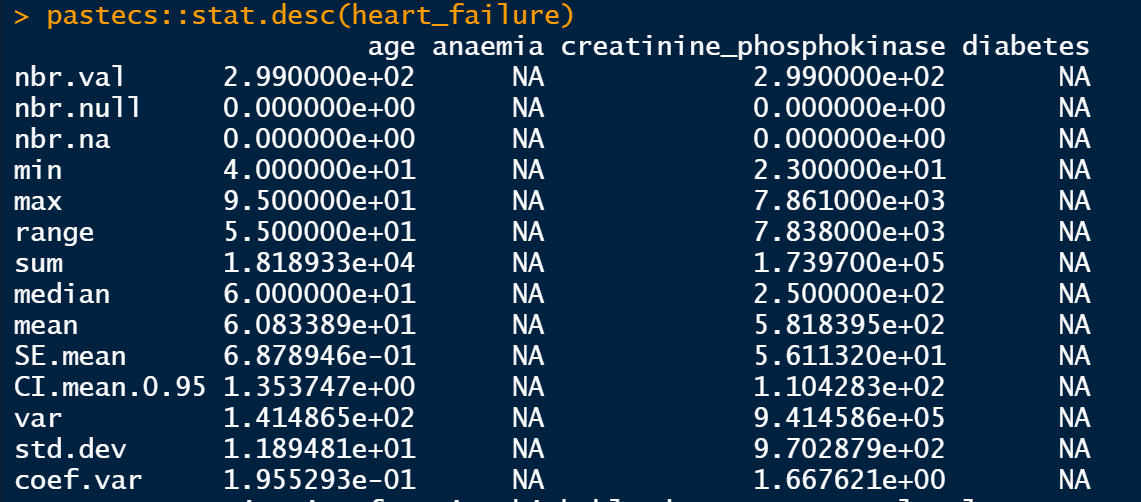
The ejection fraction states the percentage of how much blood the left ventricle pumps out with each contraction.

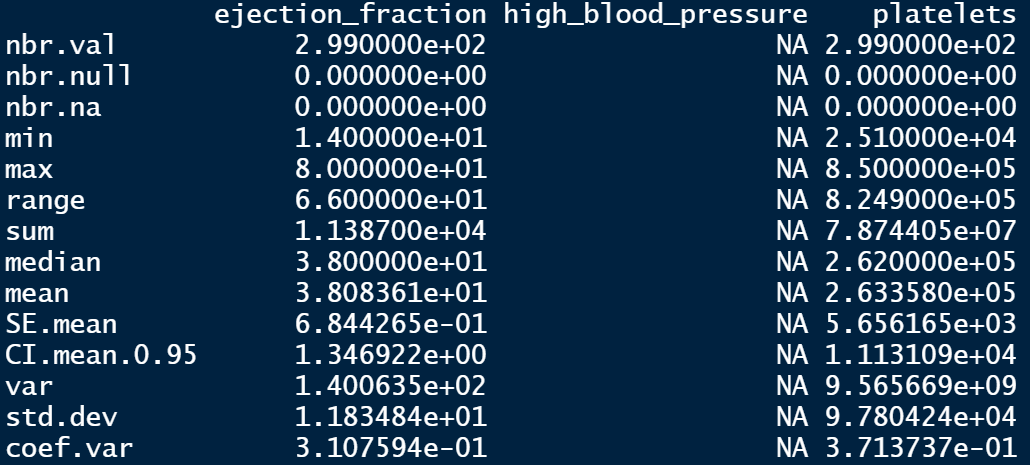
The serum creatinine is a waste product generated by creatine when a muscle breaks down. Especially, doctors focus on serum creatinine in blood to check kidney function. If a patient has high levels of serum creatinine, it may indicate renal dysfunction.

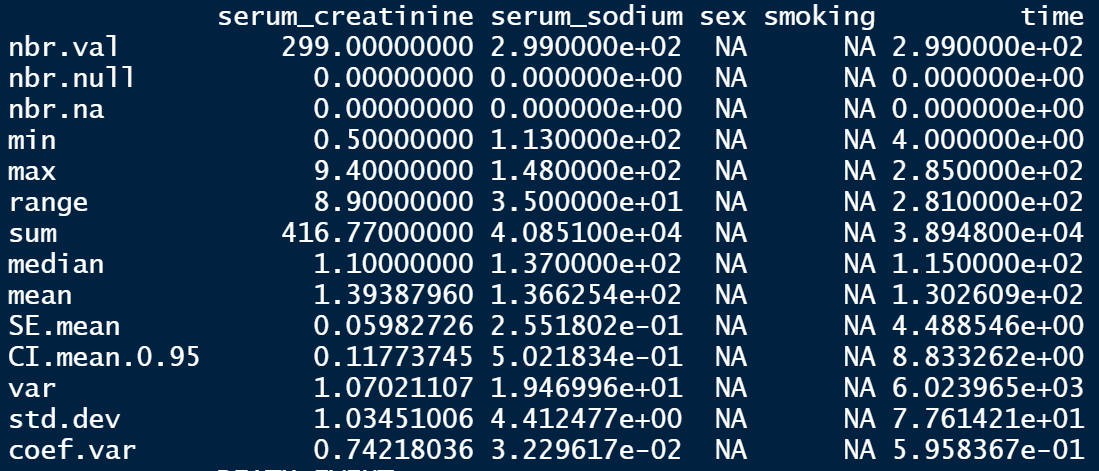
Sodium is a mineral that serves for the correct functioning of muscles and nerves. The serum sodium test is a routine blood exam that indicates if a patient has normal levels of sodium in the blood. An abnormally low level of sodium in the blood might be caused by heart failure.

The death event feature, that we use as the dependent variable, represents if the patient died or survived before the end of the follow-up period, which is 130 days on average.

Before proceeding to the modelling part, we present the summary of the dataset. In the table below, each column represents the variable of the dataset, and each row represents the descriptive statistics for each variable. These statistics are produced using the function stat.desc of ‘pastecs’ library of R. Looking at the values in the table, we inspect whether this dataset has any issues such as missing values etc. There are no missing values. And, ‘NA’ values are printed for features that are in binary format.







All independent variable columns are also rescaled since the unit of measurements are not uniform. To ensure that the dataset is ready for analysis, we also plot the distribution of each column. The distributions are as we expected. Thus, no further data pre-processing needed.

Timeline

Description automatically generated

Before proceeding to the modelling stage, we separate the dataset into training and test set, in 2/3 and 1/3 ratio respectively. The reason behind this ratio is that due to the moderate size of the dataset (n = 299), we think it is appropriate for the test set not to go lower than 1/3 of the original dataset size.

In Section 2, we provide models to analyse this dataset. Since our dependent variable ‘DEATH\_EVENT’ is in binary form, that narrows down our model choices. Therefore, we are going with logistic regression and random forest.

**Section 2: Theoretical Backgrounds and Analyses**

Cardiovascular diseases kill approximately 17 million people globally every year, and they mainly exhibit as myocardial infarctions and heart failures. Myocardial infarctions usually happen when a blood clot blocks blood flow to the heart. Without enough blood, tissue loses oxygen and dies. And heart failures occur when the heart cannot pump enough blood to meet the needs of the body.

Available electronic medical records that quantify symptoms, body features, and clinical laboratory test values can be used to perform biostatistical analysis aimed at highlighting patterns and correlations otherwise undetected by medical doctors. Furthermore, machine learning can be utilised to predict patients’ survival from their data, allowing physicians to personalise the patients’ care and thus maximising the chance of survival of the patients.

Modelling survival for heart failure is still a problem nowadays, both in terms of achieving high prediction accuracy and identifying the driving factors. Most of the models developed for this purpose reach only modest accuracies, with limited interpretability from the predicting variables. More recent models show improvements, especially if the survival outcome is coupled with additional targets (for example, hospitalization). Although scientists have identified a broad set of predictors and indicators, there is no shared consensus on their relative impact on survival prediction.

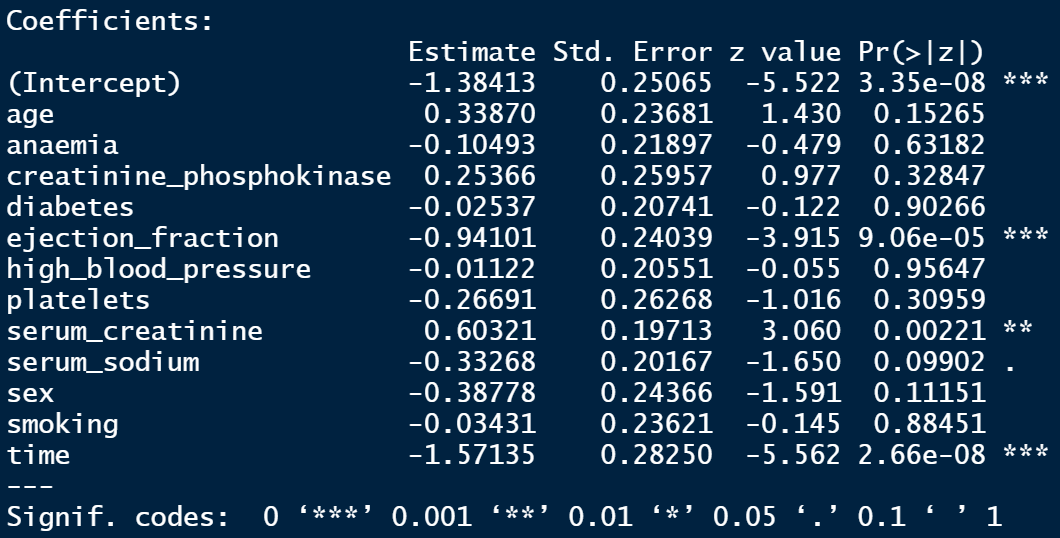
As pointed out by Sakamoto et al. in 2018, this situation is largely due to a lack of reproducibility, which prevents drawing definitive conclusions about the importance of the detected factors. Further, this lack of reproducibility strongly affects model performances: generalisation to external validation datasets is often inconsistent and achieves only modest discrimination. Consequently, risk factors statistical significances from the models suffer similar problems, limiting their reliability.

In this paper, we provide 2 reproducible machine learning algorithms that can be used to model the survival of heart attack patients and identify the most important factors determining the survivals. The first model is the logistic regression which is presented in Section 2 (a). We also employ the random forest algorithm on the dataset, and it is presented in Section 2 (b).

**Section 2 (a): Logistic Regression**

Logistic regression is used to model the probability of a certain event. In this case, we refer to the probability of surviving a heart attack event. This event is recorded in DEATH\_EVENT column, where 0 means the patients survives and 1 means the patients deceases.

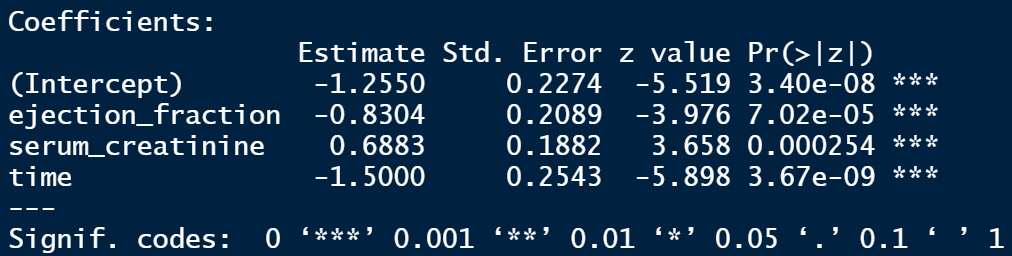
Firstly, we specify the logistic regression model of interest. Then we fit the training set onto model specified. In this paper, we specify logistic\_regression1 as the primary choice of model. The first model includes every column in the dataset. In this model, we would like to see which variables are significant to predict the survival of heart attack patients. Presented below is the trained model for this logistic regression.



At a glance, we notice that there are only 3 statistically significant variables. Ejection fraction and time are statistically significant at 100% confidence level, whilst serum creatinine is statistically significant at 99.9% confidence level. Moving onto the coefficients, we observe that the signs of the statistically significant variables are as expected. Such as that the negative sign of ejection fraction signifies that this clinical laboratory test value is negatively correlated with a patient’s death. The same sign is observed for time. Meanwhile, a positive sign is observed for serum creatinine.

Generally, the results of logistic\_regresison1 makes sense. However, we would like to improve the model and thus we re-specify another logistic regression model. This second model (logistic\_regression2) is only made up of the statistically significant dependent variables from the first specification. I.e., ejection fraction, serum creatinine and time.

Presented below is the refined model. We see that now all the estimated coefficients more or less the same as before. (Same signs and magnitudes) Only now, all the estimates are statistically significant at 100% confidence level.



Next, we perform several tests to compare the 2 logistic regression models and determine the better model based on the test results and interpretability of the models. The table below summarises the results of the tests performed on both logistic regression models. Based on the results of the following tests, we conclude that **logistic\_regression2 is the better logistic regression model for this dataset**.

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Logistic\_regression1 | Logistic\_regression2 | Better model |
| Mean Squared Error | 6.581201 | 5.242297 | Logistic\_regression2 |
| Multicollinearity test | No multicollinearity | No multicollinearity | Indifferent |
| Akaike Information Criterion | 177.82 | 170.63 | Logistic\_regression2 |
| Fisher Scoring Iterations | 5 | 5 | Indifferent |

Taking logistic\_regression2 as the model of choice, we proceed to test the model on the test dataset, which was separated from the original dataset in the pre-processing stage. This testing stage is mainly performed using the R library ‘InformationValue’. Upon mapping the model on the test dataset, we obtain the confusion matrix showing the difference between the predicted and the real value of DEATH\_EVENT = 0 (survives) and DEATH\_EVENT = 1 (deceased). Presented below is the confusion matrix obtained:

|  |  |  |
| --- | --- | --- |
|  | Actual survive | Actual deceased |
| Predicted survive | 66 | 12 |
| Predicted deceased | 2 | 20 |

As shown in the confusion matrix above, our logistic regression model performed quite well in predicting the survival rate of heart attack patients. The accuracy rate thus can be calculated from this matrix.

Accuracy rate = ((66+20) / (66+12+2+20)) \* 100 = 86%

We also calculate other matrices such as the sensitivity rate, specificity rate and the misclassification rate.

Sensitivity rate = 62.50%

Specificity rate = 97.05%

Misclassification rate = 13.00%

Furthermore, presented below is the Receiver operating characteristic curve (ROC) curve for the logistic regression model. The Area Under the ROC curve (AUROC) is an aggregated metric that evaluates how well a logistic regression model classifies positive and negative outcomes at all possible cut offs. It can range from 0.5 to 1, and the larger it is the better. In this case, our AUROC = 0.9, which is very good and close to 1.

Chart

Description automatically generated

To conclude this section, we plot the chosen logistic regression model. To do this, we first create a data frame of the predicted probabilities of surviving from a heart attack and the actual survival status. Secondly, we sort this data frame by ascending order of the predicted probabilities of surviving. Thirdly, we add another column ranking the order in the data frame, which will be useful when plotting the logistic regression model. Shown below is the plot:

Chart, scatter chart

Description automatically generated

**Section 2 (b): Random Forest**

In attempting to further improve the predictive power of our analyses, we employ the random forest method. Three main R libraries are used for this stage. I.e., ‘ggplot2’, ‘cowplot’, ‘randomForest’. In order to compare the random forest model with its adjacent logistic regression model that is built using the same variables, we specify the first random forest model with all the variables in the dataset. The second random forest model is specified with the same variables chosen for the logistic regression model 2.

Thus, we start our modelling by specifying the first model with all the variables using the function randomForest. Then, we fit the training data on the first model which is called random\_forest1. At this stage, we go with the default number of trees = 500. And the number of variables tried at each split = 3. For the first random forest model, we obtain the Out-Of-Bag (OOB) Error Rate of 15.58%, with the resulting confusion matrix as below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Actual survive | Actual deceased | Class error |
| Predicted survive | 123 | 12 | 0.08888889 |
| Predicted deceased | 19 | 45 | 0.29687500 |

123 patients predicted to survive survived while 12 patients predicted deceased survive. 45 patients predicted deceased deceased and 19 patients predicted survived deceased. Thus, the accuracy rate for our first random forest model is 84.42%.

Accuracy rate = 100% – OOB error rate = 100% - 15.58% = 84.42%

The second random forest model with the same number of trees = 500 and the number of variable tried at each split = 1. For this model, the OOB error rate is 18.59%, with the resulting confusion matrix as below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Actual survive | Actual deceased | Class error |
| Predicted survive | 120 | 15 | 0.1111111 |
| Predicted deceased | 22 | 42 | 0.3437500 |

120 patients precited to survive survived while 15 patients predicted deceased survive. 42 patients predicted deceased deceased, 22 patients predicted survived deceased. Thus, the accuracy rate for the second random forest model is 81.41%.

Accuracy rate = 100% - OOB error rate = 100% - 18.59% = 81.41%

With respect to the accuracy rate, we observe that the first random forest model which includes all the variables performs slightly better. One way to see whether it is possible to improve the accuracy is by examining the OOB error rate evolution as the number of trees used changes. To examine this, we created a new data frame consisting of the extracted values of error by OOB, survived and deceased error. The chart below shows the evolution of error rates for random\_forest1.

Chart

Description automatically generated

For random forest model 1, the error rates for all 3 categories do not stabilise up until trees = 400, where we get the OOB error rate of 20%.

A picture containing timeline

Description automatically generated

For random forest model 2, the error rates do not stabilise even until trees = 500. Therefore, we will run an adjusted model where trees = 1000 to see whether the error rates stabilise.

After these observations, we respecified both random forest models to have trees = 1000, double from the original number of trees. For the adjusted models, we obtained the following OOB error rates and confusion matrices.

**For random\_forest1new:**

OOB error rate = 14.57%, Accuracy rate = 85.43%

|  |  |  |  |
| --- | --- | --- | --- |
|  | Actual survive | Actual deceased | Class error |
| Predicted survive | 123 | 12 | 0.08888889 |
| Predicted deceased | 17 | 47 | 0.26562500 |

**For random\_forest2new:**

OOB error rate = 18.09%, Accuracy rate = 81.91%

|  |  |  |  |
| --- | --- | --- | --- |
|  | Actual survive | Actual deceased | Class error |
| Predicted survive | 121 | 14 | 0.1037037 |
| Predicted deceased | 22 | 42 | 0.3437500 |

Chart

Description automatically generated

Chart

Description automatically generated with low confidence

Extending the information that we have on the OOB error rates evolution on the number of trees used in the random forest models, we can see in the diagrams above that the OOB error rates for both models stabilise around trees = 625. Thus, the new models with trees = 1000 are more stable than the ones with trees = 500.

Considering all information we have on the performance of the random forest models, random\_forest1 would be slightly preferred due to its lower OOB error rate. However, we see that the improvement in the accuracy rate is not that high by going with the model with more variables – only 3.5% improvement.

To conclude this section, we take into account the accuracy-interpretability trade-off between the first and the second random forest model. On that, we think the 3.5% reduction in the accuracy rate is not enough to take on a significantly more complicated model that can be heavy on computational costs. Therefore, the preferred random forest model is random\_forest2new, with accuracy rate 81.91%.

Going with that information, we plot the MDS plot of random\_forest2new below. In the plot below, the blue patients which mostly centred on the left side represent the deceased patients. And the red patients are the patients that survived heart attacks. There are some deceased patients that were predicted to survive such as patient 111, patient 109 and patient 126. This shows that our chosen model is not 100% accurate. Improvements is expected if given a dataset with larger sample size.

Chart, scatter chart

Description automatically generated

**Section 3: Conclusion**

In conclusion, both logistic regression algorithm (logistic\_regression2) and random forest algorithm (random\_forest2new) perform very well (more than 80% accuracy rate) in predicting which patients survive and which do not, based on only 3 key variables (ejection fraction, serum creatinine and time). These insights should shed new lights for biostatisticians to design survival analysis experiments that focus on these important variables. And since these medical laboratory measures are only taken when a patient experiences a heart attack, these new insights should motivate physicians to take these measures more regularly in order to identify how well the patients are living with their cardiovascular diseases and thus avoiding premature death of the patients.

**References**

Chicco, D., & Jurman, G. (2020). Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. *BMC medical informatics and decision making*, *20*(1), 1-16.

Sakamoto, M., Fukuda, H., Kim, J., Ide, T., Kinugawa, S., Fukushima, A., ... & Kitakaze, M. (2018). The impact of creating mathematical formula to predict cardiovascular events in patients with heart failure. *Scientific reports*, *8*(1), 1-12.

**Appendix**

#personal project 1 - supervised learning

##data cleaning part##

#load heart failure data

library(readr)

heart\_failure <- read\_csv("heart\_failure.csv")

#summary statistics

pastecs::stat.desc(heart\_failure)

#plot data distributions

library('tidyverse')

ggplot(gather(heart\_failure), aes(value)) +

geom\_histogram(bins = 20) +

facet\_wrap(~key, scales = 'free\_x')

#rescaling variables

heart\_failure[,1:12] <- scale(heart\_failure[,1:12])

#split data into train and test

library(caTools)

set.seed(123)

split <- sample.split(heart\_failure$DEATH\_EVENT, SplitRatio = 2/3)

training\_set <- subset(heart\_failure, split == TRUE)

test\_set <- subset(heart\_failure, split == FALSE)

##Section 2(a) - logistic regression##

#train model 1 (with all variables)

logistic\_train1 <- glm(DEATH\_EVENT ~ ., data = training\_set, family = "binomial")

summary(logistic\_train1)

#train model 2 (with statistically significant variables)

logistic\_train2 <- glm(DEATH\_EVENT ~ ejection\_fraction + serum\_creatinine + time, data = training\_set, family = "binomial")

summary(logistic\_train2)

#calculate mse

library(modelr)

mse(logistic\_train1, training\_set)

mse(logistic\_train2, training\_set)

#compute variable importance - higher value indicates higher importance

library(caret)

varImp(logistic\_train1)

varImp(logistic\_train2)

#calculate VIF values for dependent variables

# VIF > 5 suggests multicollinearity

library(car)

vif(logistic\_train1)

vif(logistic\_train2)

#no multicollinearity in either model

#proceed with model logistic\_train2

library(InformationValue)

predicted <- predict(logistic\_train2, test\_set, type = "response")

optimal <- optimalCutoff(test\_set$DEATH\_EVENT, predicted)[1]

#the optimal probability cutoff for death is 0.5758 = 58%

#create confusion matrix prediction compared to actual deaths

confusionMatrix(test\_set$DEATH\_EVENT, predicted)

#calculate sensitivity

sensitivity(test\_set$DEATH\_EVENT, predicted)

#calculate specificity

specificity(test\_set$DEATH\_EVENT, predicted)

#calculate total misclassification error rate

misClassError(test\_set$DEATH\_EVENT, predicted, threshold = optimal)

#13% - the lower the better

#plot ROC curve

plotROC(test\_set$DEATH\_EVENT, predicted)

#to plot the graph logistic\_train2

#create data frame of probabilities of dying from heart attack and the actual status

predicted.data <- data.frame(

probability.of.death = logistic\_train2$fitted.values,

death\_event = training\_set$DEATH\_EVENT

)

#sort data frame from low probabilities to high probabilities

predicted.data <- predicted.data[

order(predicted.data$probability.of.death, decreasing = FALSE),]

#add a new column to the data frame that ranks each sample from low to high prob

predicted.data$rank <- 1:nrow(predicted.data)

library(ggplot2)

library(cowplot)

ggplot(data=predicted.data, aes(x=rank, y=probability.of.death)) +

geom\_point(aes(colour=death\_event), alpha=1, shape=4, stroke=2) +

xlab("Index") +

ylab("Predicted probability of not surviving heart attack") +

ggtitle("Logistic Regression Model of Survival of Heart Failure Patients")

##Section 2(b) - random forest##

library(ggplot2)

library(cowplot)

library(randomForest)

set.seed(24)

heart\_failure$DEATH\_EVENT <- as.factor(heart\_failure$DEATH\_EVENT)

training\_set$DEATH\_EVENT <- as.factor(training\_set$DEATH\_EVENT)

test\_set$DEATH\_EVENT <- as.factor(test\_set$DEATH\_EVENT)

#build random forest models

random\_forest1 <- randomForest(DEATH\_EVENT ~ ., data = training\_set, proximity = TRUE)

random\_forest1

#OBB error rate = 15.58%

#124 patients predicted to survive survive, 11 patients predicted deceased survive

#47 patients predicted deceased deceased, 17 patients predicted survive deceased

random\_forest2 <- randomForest(DEATH\_EVENT ~ ejection\_fraction + serum\_creatinine + time, data = training\_set, proximity = TRUE)

random\_forest2

#OBB error rate = 18.59%

#random\_forest1 performs better with respect to OOB error rate

#to observe evolution of error rate as the number of trees changes

#for random\_forest1

oob.error.data1 <- data.frame(

Trees = rep(1:nrow(random\_forest1$err.rate), times = 3),

Type = rep(c("OOB", "Survived", "Deceased"), each = nrow(random\_forest1$err.rate)),

Error = c(random\_forest1$err.rate[,"OOB"],

random\_forest1$err.rate[,"0"],

random\_forest1$err.rate[,"1"]))

#plot the graph

ggplot(data = oob.error.data1, aes(x=Trees, y=Error)) +

geom\_line(aes(colour=Type)) +

ggtitle("Error Rate Evolution for random\_forest1")

#for random\_forest2

oob.error.data2 <- data.frame(

Trees = rep(1:nrow(random\_forest2$err.rate), times = 3),

Type = rep(c("OOB", "Survived", "Deceased"), each = nrow(random\_forest2$err.rate)),

Error = c(random\_forest2$err.rate[,"OOB"],

random\_forest2$err.rate[,"0"],

random\_forest2$err.rate[,"1"]))

#plot the graph

ggplot(data = oob.error.data2, aes(x=Trees, y=Error)) +

geom\_line(aes(colour=Type)) +

ggtitle("Error Rate Evolution for random\_forest2")

#from graphs, tree = 145 is sufficient (save computational power) and optimal (any higher, error rate does not decrease)

#both random\_forest1 and random\_forest2 error rates evolve similarly (no improvements after trees=200)

#up to this point, random\_forest1 is slightly preferred due to general OBB error rate

#however, considering interpretability-accuracy trade-off, we'll go with random\_forest2, which only include certain variables, instead of all

#respecify random forest models with tree=1000 to check OOB

random\_forest1new <- randomForest(DEATH\_EVENT ~ ., data = training\_set, ntree=1000, proximity = TRUE)

random\_forest1new

#OOB error rate = 14.07%

random\_forest2new <- randomForest(DEATH\_EVENT ~ ejection\_fraction + serum\_creatinine + time, data = training\_set, ntree=1000, proximity = TRUE)

random\_forest2new

#OOB error rate 18.59%

#to observe evolution of error rate as the number of trees increase (for testing)

#for random\_forest1new

oob.error.data1new <- data.frame(

Trees = rep(1:nrow(random\_forest1new$err.rate), times = 3),

Type = rep(c("OOB", "Survived", "Deceased"), each = nrow(random\_forest1new$err.rate)),

Error = c(random\_forest1new$err.rate[,"OOB"],

random\_forest1new$err.rate[,"0"],

random\_forest1new$err.rate[,"1"]))

#plot the graph

ggplot(data = oob.error.data1new, aes(x=Trees, y=Error)) +

geom\_line(aes(colour=Type)) +

ggtitle("Error Rate Evolution for random\_forest1 (trees=1000)")

#for random\_forest2new

oob.error.data2new <- data.frame(

Trees = rep(1:nrow(random\_forest2new$err.rate), times = 3),

Type = rep(c("OOB", "Survived", "Deceased"), each = nrow(random\_forest2new$err.rate)),

Error = c(random\_forest2new$err.rate[,"OOB"],

random\_forest2new$err.rate[,"0"],

random\_forest2new$err.rate[,"1"]))

#plot the graph

ggplot(data = oob.error.data2new, aes(x=Trees, y=Error)) +

geom\_line(aes(colour=Type)) +

ggtitle("Error Rate Evolution for random\_forest2 (trees=1000)")

#random\_forest2 appears more stable after increasing tree to 1000

#now we test the optimal number of variables at each internal node in the tree

#random\_forest1 vs random\_forest2

#create empty vector that can hold 10 values

#for random\_forest1

obb.values1 <- vector(length = 10)

#create a loop that tests different number of variables at each step

for(i in 1:10){

temp.model1 <- randomForest(DEATH\_EVENT~., data = training\_set, mtry = i, ntree = 500)

obb.values1[i] <- temp.model1$err.rate[nrow(temp.model1$err.rate),1]

}

obb.values1

#for random\_forest2

obb.values2 <- vector(length = 10)

for(i in 1:10){

temp.model2 <- randomForest(DEATH\_EVENT ~ ejection\_fraction + serum\_creatinine + time, data = training\_set, mtry = i, ntree = 500)

obb.values2[i] <- temp.model2$err.rate[nrow(temp.model2$err.rate),1]

}

obb.values2

#stick with 1 variable, instead of 9 suggested, since we only have 4 independent variables in random\_forest2

#best random forest model = random\_forest2

#MDS plots#

#create distance matrix: 1-proximity matrix

distance.matrix <- dist(1-random\_forest2$proximity)

#run cmd (classical multi-dimensional scaling) on the distance matrix

mds.stuff <- cmdscale(distance.matrix, eig=TRUE, x.ret=TRUE)

#then calculate percentage of variation in the distancce matrix

mds.var.per <- round(mds.stuff$eig/sum(mds.stuff$eig)\*100, 1)

#format data for ggplot

mds.values <- mds.stuff$points

mds.data <- data.frame(Sample=rownames(mds.values),

X=mds.values[,1],

Y=mds.values[,2],

Status=training\_set$DEATH\_EVENT)

#plot the graph

ggplot(data=mds.data, aes(x=X, y=Y, label=Sample)) +

geom\_text(aes(colour=Status))+

theme\_bw() +

xlab(paste("MDS1 -", mds.var.per[1], "%", sep="")) +

ylab(paste("MDS2 -", mds.var.per[2], "%", sep=""))+

ggtitle("MDS plot using (1- Random Forest Proximities)")

#the patients in blue are deceased patients

#the patients in red are those who survived

#random\_forest2 did a good job predicting the outcome

#there are some exceptions (blue patients on the right side of the graph)

#Conclusion: random forest did a good job predicting, but imperfect.

#There are other factors in the play: expert opinions, other characteristics of the patients.

1. <https://plos.figshare.com/articles/dataset/Survival_analysis_of_heart_failure_patients_A_case_study/5227684/1> [↑](#footnote-ref-1)