Machine Learning Project - Kaggle Heart Failure

HEART FAILURE PREDICTION

This project involves using heart failure data from different patients to create a machine learning model which can be used for predicting whether a patient is likely to have a fatal heart failure event. There are several steps taken in this project. Step 1 involves loading, visualising and transforming the data, step 2 involves splitting the data into a training and test set which will be used during the modeling steps. Step 3 involves trying out different models and comparing the accuracy, ROC plot and AUC. Step 4 is short listing the best models which will then be further tuned to improve the performance (hyperparameters), step 5 will involve feature engineering to attempt to further improve performance of the models. A final best model will be chosen.

STEP 1: Adding required libraries, loading dataset, transforming and visualling data -

```
library(caret)
library(tidyverse)
library(randomForest)
library(corrplot)
library(ggplot2)
library(pROC)
library(ROCR)
library(xgboost)
library(rattle)
library(dplyr)
library(MLmetrics)
library(patchwork)
library(lemon)
```

The required libraries are imported.

```
file_path <- "~/Desktop/Graduate Life/Machine Learning/Kaggle Datasets"
raw_dataset <- read.csv(paste(file_path,"heart_failure_clinical_records_dataset.csv",sep="/"), header=TRUE) #Load
CSV
glimpse(raw_dataset)
```

```
## Rows: 299
## Columns: 13
## $ age                      <dbl> 75, 55, 65, 50, 65, 90, 75, 60, 65, 80, 75, 6…
## $ anaemia                  <int> 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, …
## $ creatinine_phosphokinase <int> 582, 7861, 146, 111, 160, 47, 246, 315, 157, …
## $ diabetes                 <int> 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, …
## $ ejection_fraction        <int> 20, 38, 20, 20, 20, 40, 15, 60, 65, 35, 38, 2…
## $ high_blood_pressure      <int> 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, …
## $ platelets                <dbl> 265000, 263358, 162000, 210000, 327000, 20400…
## $ serum_creatinine         <dbl> 1.90, 1.10, 1.30, 1.90, 2.70, 2.10, 1.20, 1.1…
## $ serum_sodium             <int> 130, 136, 129, 137, 116, 132, 137, 131, 138, …
## $ sex                      <int> 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, …
## $ smoking                  <int> 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, …
## $ time                     <int> 4, 6, 7, 7, 8, 8, 10, 10, 10, 10, 10, 10, 11,…
## $ DEATH_EVENT              <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, …
```

The kaggle dataset for the heart failure clinical records is the loaded and a glimpse of the dataset is presented. Here we can see that there are binary features which are currently of the wrong datatype (should be factors). This is fixed by mutating the columns. There are also 299 patient data rows made up of 13 columns (12 feature variables and 1 response variable).

```
dataset <- raw_dataset %>%
  mutate(anaemia = if_else(anaemia==1,"YES","NO"),
         diabetes = if_else(diabetes==1,"YES","NO"),
         high_blood_pressure = if_else(high_blood_pressure==1,"YES","NO"),
         sex = if_else(sex==1,"MALE","FEMALE"),
         smoking = if_else(smoking==1,"YES","NO"),
         DEATH_EVENT = if_else(DEATH_EVENT==1,"YES","NO"),
         age = as.integer(age),
         platelets = as.integer(platelets)
  ) %>%
  mutate_if(is.character,as.factor) %>%
  dplyr::select(DEATH_EVENT,anaemia,diabetes,high_blood_pressure,sex,smoking,everything())
```

The features 'anaemia', 'diabetes', high_blood_pressure', 'sex', 'smoking', 'DEATH_EVENT' are converted to factors where 1 = 'YES' and 0 = 'NO'. 'Sex' is also converted to a factor with 1 = 'MALE' and 0 = 'FEMALE'. Finally 'age' and 'platelets' are converted to intergers as they are currently doubles.

We then glimpse the transformed dataset to ensure this has worked and then have a look at the first 6 rows of the dataset.

```
glimpse(dataset)
```

```
## Rows: 299
## Columns: 13
## $ DEATH_EVENT              <fct> YES, YES, YES, YES, YES, YES, YES, YES, YES, …
## $ anaemia                  <fct> NO, NO, NO, YES, YES, YES, YES, YES, NO, YES,…
## $ diabetes                 <fct> NO, NO, NO, NO, YES, NO, NO, YES, NO, NO, NO,…
## $ high_blood_pressure      <fct> YES, NO, NO, NO, NO, YES, NO, NO, NO, YES, YE…
## $ sex                      <fct> MALE, MALE, MALE, MALE, FEMALE, MALE, MALE, M…
## $ smoking                  <fct> NO, NO, YES, NO, NO, YES, NO, YES, NO, YES, Y…
## $ age                      <int> 75, 55, 65, 50, 65, 90, 75, 60, 65, 80, 75, 6…
## $ creatinine_phosphokinase <int> 582, 7861, 146, 111, 160, 47, 246, 315, 157, …
## $ ejection_fraction        <int> 20, 38, 20, 20, 20, 40, 15, 60, 65, 35, 38, 2…
## $ platelets                <int> 265000, 263358, 162000, 210000, 327000, 20400…
## $ serum_creatinine         <dbl> 1.90, 1.10, 1.30, 1.90, 2.70, 2.10, 1.20, 1.1…
## $ serum_sodium             <int> 130, 136, 129, 137, 116, 132, 137, 131, 138, …
## $ time                     <int> 4, 6, 7, 7, 8, 8, 10, 10, 10, 10, 10, 10, 11,…
```

```
head(dataset)
```

	DEATH_EVENT <fct>	anaemia <fct>	diabetes <fct>	high_blood_pressure <fct>	sex <fct>	smoking <fct>	a... <int>	creatinine_phosphokinase <int>
1	YES	NO	NO	YES	MALE	NO	75	582
2	YES	NO	NO	NO	MALE	NO	55	7861
3	YES	NO	NO	NO	MALE	YES	65	146
4	YES	YES	NO	NO	MALE	NO	50	111
5	YES	YES	YES	NO	FEMALE	NO	65	160
6	YES	YES	NO	YES	MALE	YES	90	47

6 rows | 1-9 of 14 columns

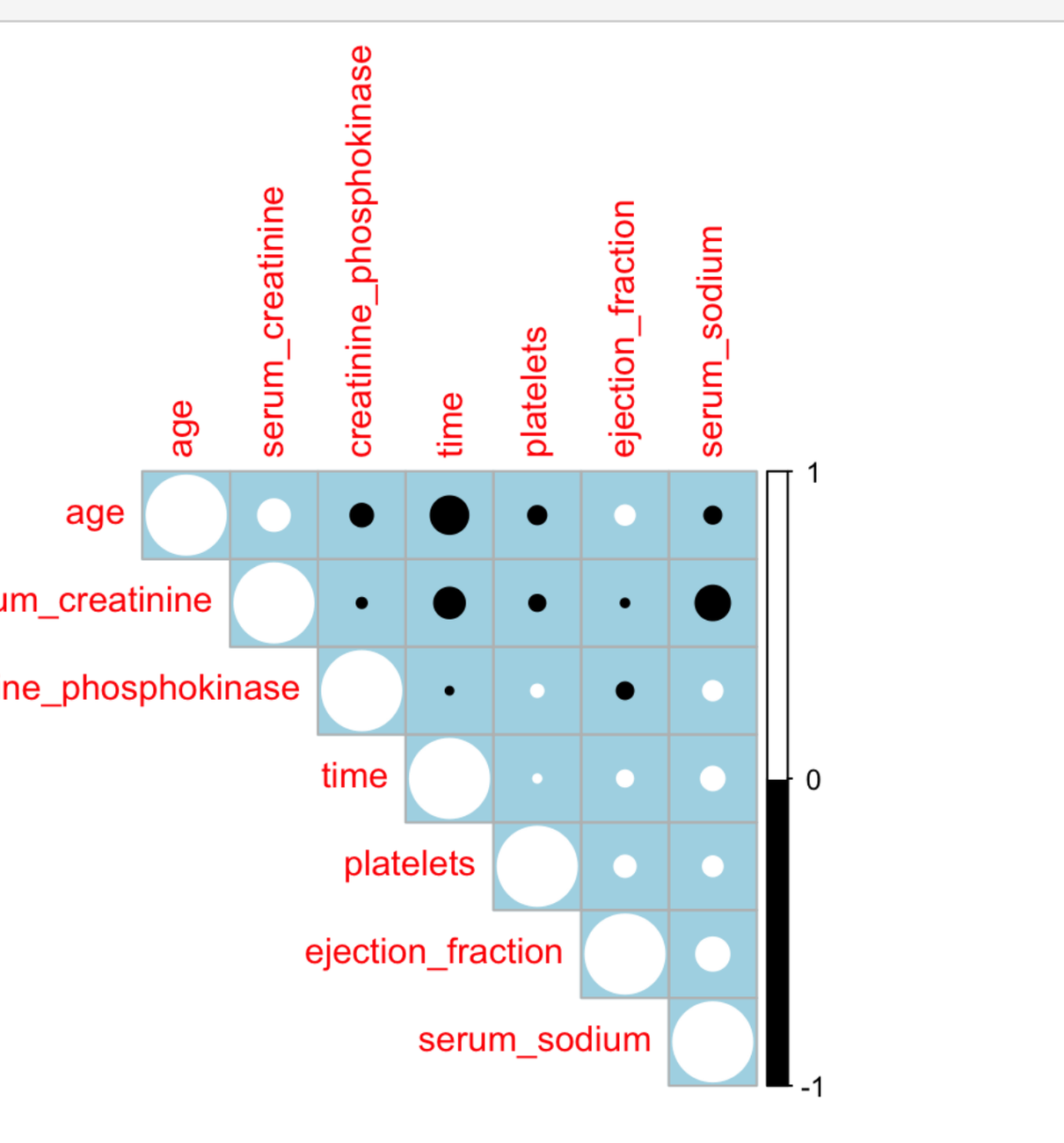
We can take a look at the percentage breakdown of those who have had a fatal heart failure vs those patients who have not. Here we see approx. 32% of patients suffered a fatal event, whilst 68% did not.

```
percentage <- prop.table(table(dataset$DEATH_EVENT)) * 100
cbind(freq=table(dataset$DEATH_EVENT), percentage=percentage)
```

```
##     freq percentage
## NO   203   67.89298
## YES   96   32.10702
```

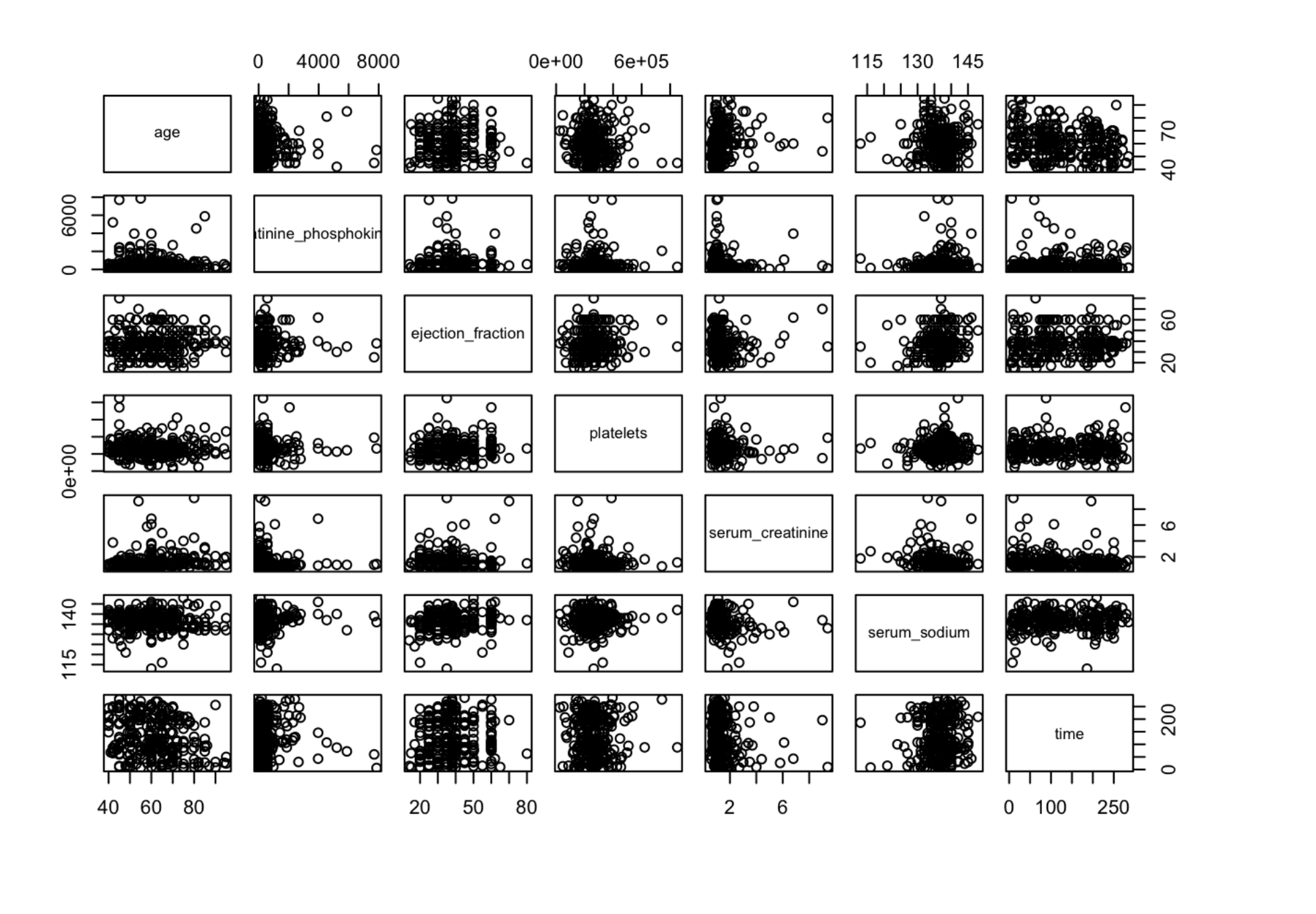
A correlation plot of the non-categorical variables shows there is not much correlation between the different features.

```
corr_data <- cor(dataset[7:13])
corrplot(corr_data, order="hclust", col=c("black", "white"), bg="lightblue", type="upper")
```



This can further seen in the plots below which show that there is little correlation between the different continuous numerical features.

```
plot(dataset[7:13])
```



...

STEP 2: splitting dataset into training and test sets.

The dataset is then split into a training sample and a test sample. The training:test split will be 80:20. The training data will be exclusively used to train the different models (using K Fold Cross Validation) and the test data is used exclusively for measuring the performance of the models.

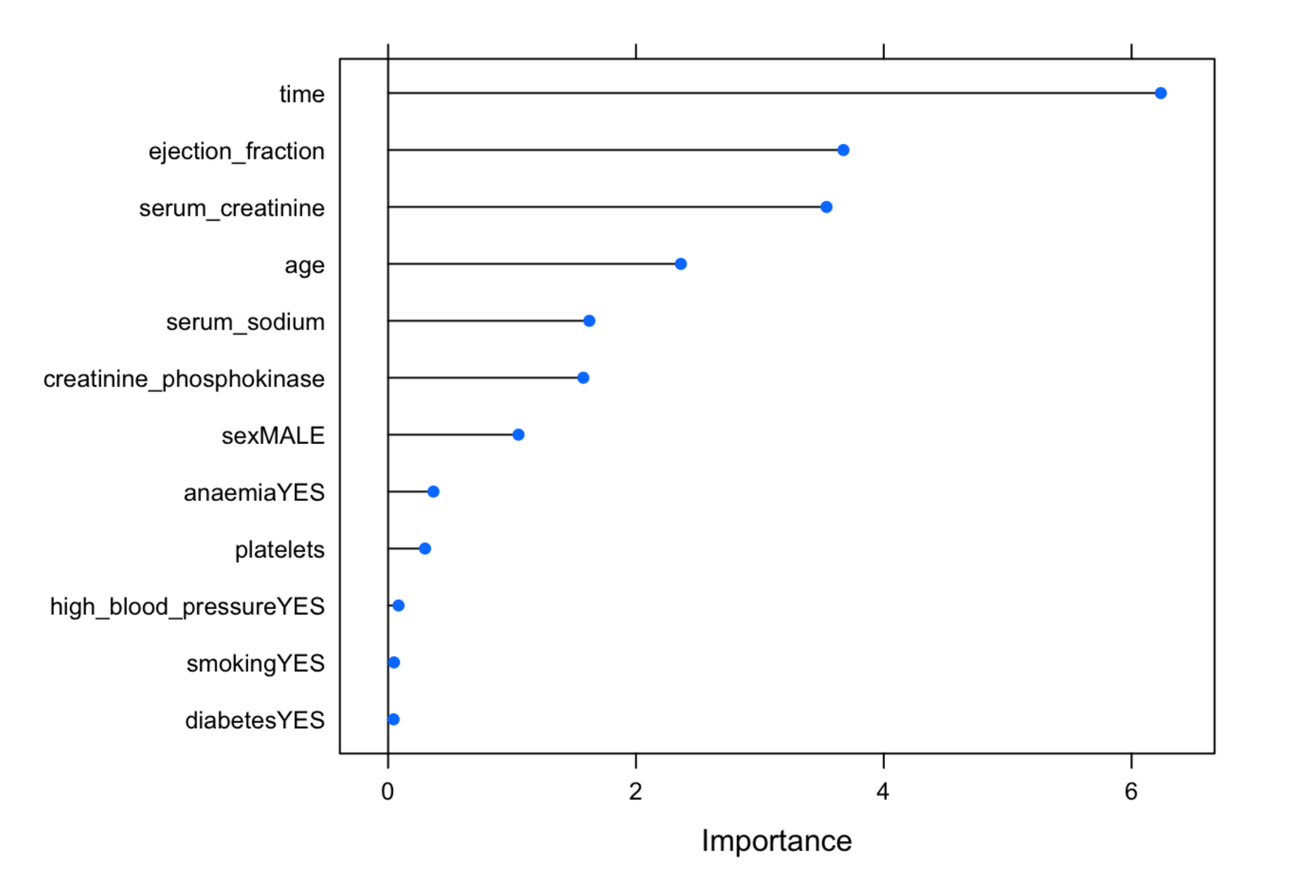
```
set.seed(1995)
training_samples <- createDataPartition(dataset$DEATH_EVENT, p=0.80, list=FALSE)
train_data <- dataset[training_samples,]
test_data <- dataset[-training_samples,]
```

STEP 3: Training the different models

Here we will go through and train each of the models we will compare for the classification of heart failure. The models we will be using are - Logistic Regression, K Nearest Neighbor, Linear Support Vector Machines, Non-Linear Support Vector Machines, Decision Tree, Random Forest Ensemble, Stochastic Gradient Boosting, Extreme Gradient Boosting and Neural Networks.

We have selected 'age', 'ejection_fraction', 'serum_creatinine' and 'time' as these were shown to lead to the best accuracy in the prior run and also have the highest importance in the importance plots -

```
set.seed(692)
train_control <- trainControl(method="repeatedcv", number=10, repeats = 3)
log_model <- train(DEATH_EVENT~., data=train_data, trControl=train_control, method="glm", family="binomial")
importance <- varImp(log_model, scale=FALSE)
plot(importance)
```



Logistic Regression:

```
set.seed(692)
train_control <- trainControl(method="repeatedcv", number=10, repeats = 3)
log_model <- train(DEATH_EVENT~ age + ejection_fraction + serum_creatinine + time,
                   data=train_data, trControl=train_control, method="glm", family="binomial")
log_pred <- predict(log_model,newdata = test_data)
```

K Nearest Neighbors:

```
set.seed(200)
train_control <- trainControl(method="repeatedcv", number=10, repeats = 3, search="grid")
tune_grid <- expand.grid(k=5)
knn_model <- train(DEATH_EVENT ~ age + ejection_fraction + serum_creatinine + time, data = train_data, method =
"knn", trControl=train_control, preProcess = c("center", "scale"), tuneGrid=tune_grid)
knn_pred <- predict(knn_model, newdata = test_data)
```

Support Vector Machine:

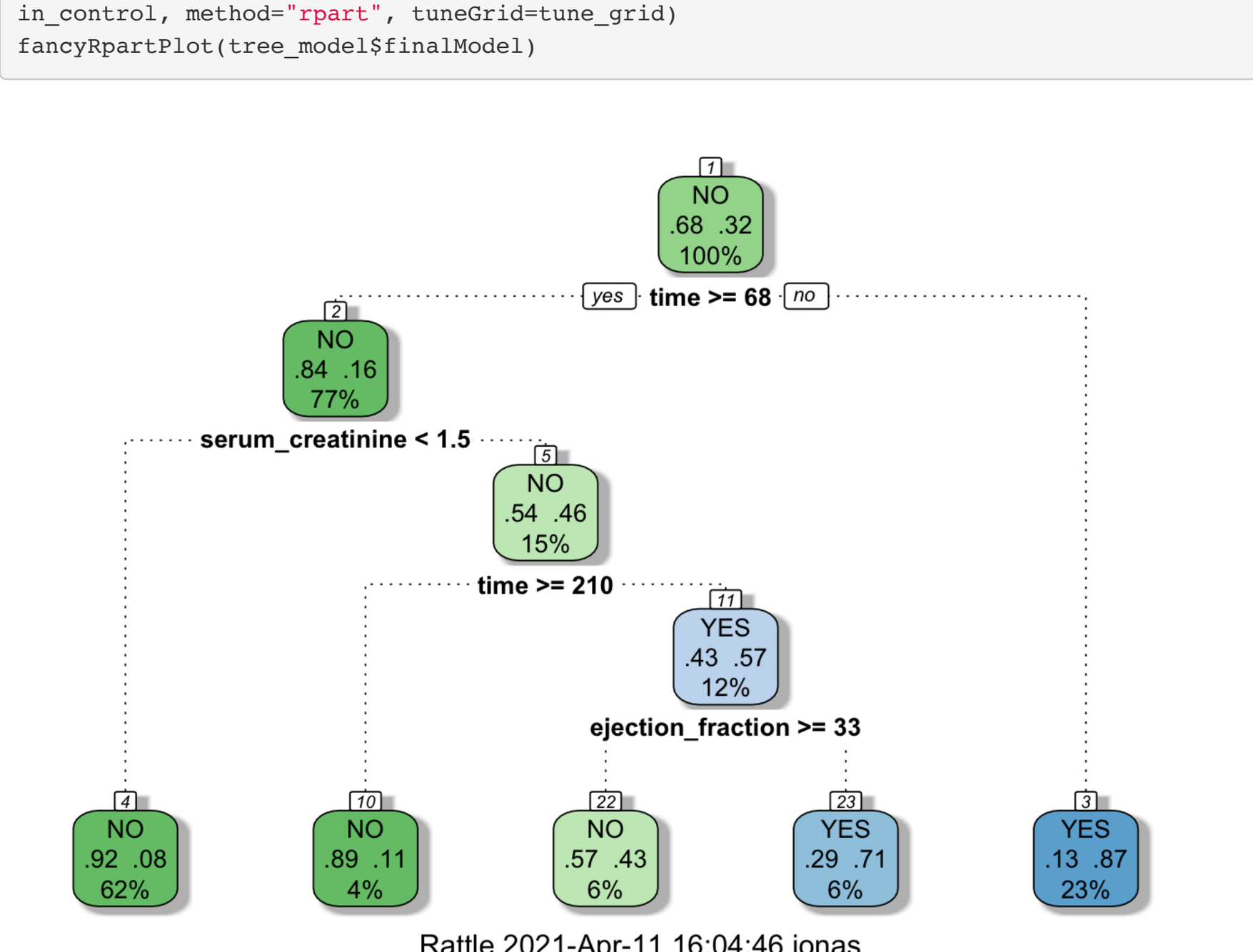
```
set.seed(723)
train_control <- trainControl(method="repeatedcv", number=10, classProbs =  TRUE, repeats = 3, search = "grid")
tune_grid <- expand.grid(C=39)
svm_model <- train(DEATH_EVENT ~age + ejection_fraction + serum_creatinine + time, data = train_data, method = "s
vmLinear", trControl = train_control, preProcess = c("center","scale"), tuneGrid=tune_grid)
svm_pred <- predict(svm_model, newdata = test_data)
```

SVM using Non-Linear Kernel

```
set.seed(139)
train_control <- trainControl(method="repeatedcv", number=10, classProbs =  TRUE, repeats = 3)
non_svm_model <- train(DEATH_EVENT~ age + ejection_fraction + serum_creatinine + time, data=train_data, trControl
=train_control, method="svmRadial", preProcess=c("center", "scale"), tunelength=10)
non_svm_pred <- predict(non_svm_model, newdata = test_data)
```

Decision Tree:

```
set.seed(823)
train_control <- trainControl(method="repeatedcv", number=10, repeats = 3, search = "grid")
tune_grid <- expand.grid(cp=0)
tree_model <- train(DEATH_EVENT~age + ejection_fraction + serum_creatinine + time, data=train_data, trControl=tra
in_control, method="rpart", tuneGrid=tune_grid)
fancyRpartPlot(tree_model$finalModel)
```



Rattle 2021-Apr-11 16:04:46 jonas

```
tree_pred <- predict(tree_model, newdata = test_data)
```

Random Forest Ensemble:

```
set.seed(536)
train_control <- trainControl(method="repeatedcv", number=10, repeats = 3, search="grid")
tune_grid <- expand.grid(mtry=2)
rf_model <- train(DEATH_EVENT~., data=train_data, trControl=train_control, method="rf",tuneGrid=tune_grid)
rf_pred <- predict(rf_model,newdata = test_data)
```

Stochastic Gradient Boosting:

```
set.seed(629)
train_control <- trainControl(method="repeatedcv", number=10, repeats = 10, search="grid")
tune_grid <- expand.grid(n.trees=350,interaction.depth=5,shrinkage=0.01,n.minobsinnode=10)
gbm_model <- train(DEATH_EVENT~age + ejection_fraction + serum_creatinine + time, data=train_data, trControl=trai
n_control, method="gbm", verbose=0, tuneGrid=tune_grid)
gbm_pred <- predict(gbm_model, newdata = test_data)
```

Extreme Gradient Boosting:

```
set.seed(467)
train_control <- trainControl(method="repeatedcv", number=10, repeats = 3, search = "grid")
tune_grid <- expand.grid(nrounds=150, max_depth=3, eta=0.3, gamma=0, colsample_bytree=0.8, min_child_weight=1, su
bsample=1)
xgb_model <- train(DEATH_EVENT~., data=train_data, trControl=train_control, method="xgbTree", tuneGrid=tune_grid)
xgb_pred <- predict(xgb_model,newdata = test_data)
```

Neural Networks:

```
set.seed(888)
train_control <- trainControl(method = 'repeatedcv', number = 10, repeats = 3, classProbs = TRUE, verboseIter = F
ALSE, summaryFunction = twoClassSummary, preProcOptions = list(thresh = 0.75, ICAcomp = 3, k = 5), search="grid")
tune_grid <- expand.grid(size=3, decay=0.1)
nn_model <- train(DEATH_EVENT~anaemia + age + ejection_fraction + serum_creatinine + time, data = train_data, met
hod = 'nnet', preProcess = c('center', 'scale'), trControl = train_control, metric = "ROC", trace=FALSE, tuneGrid
=tune_grid)
nn_pred <- predict(nn_model, newdata = test_data)
```

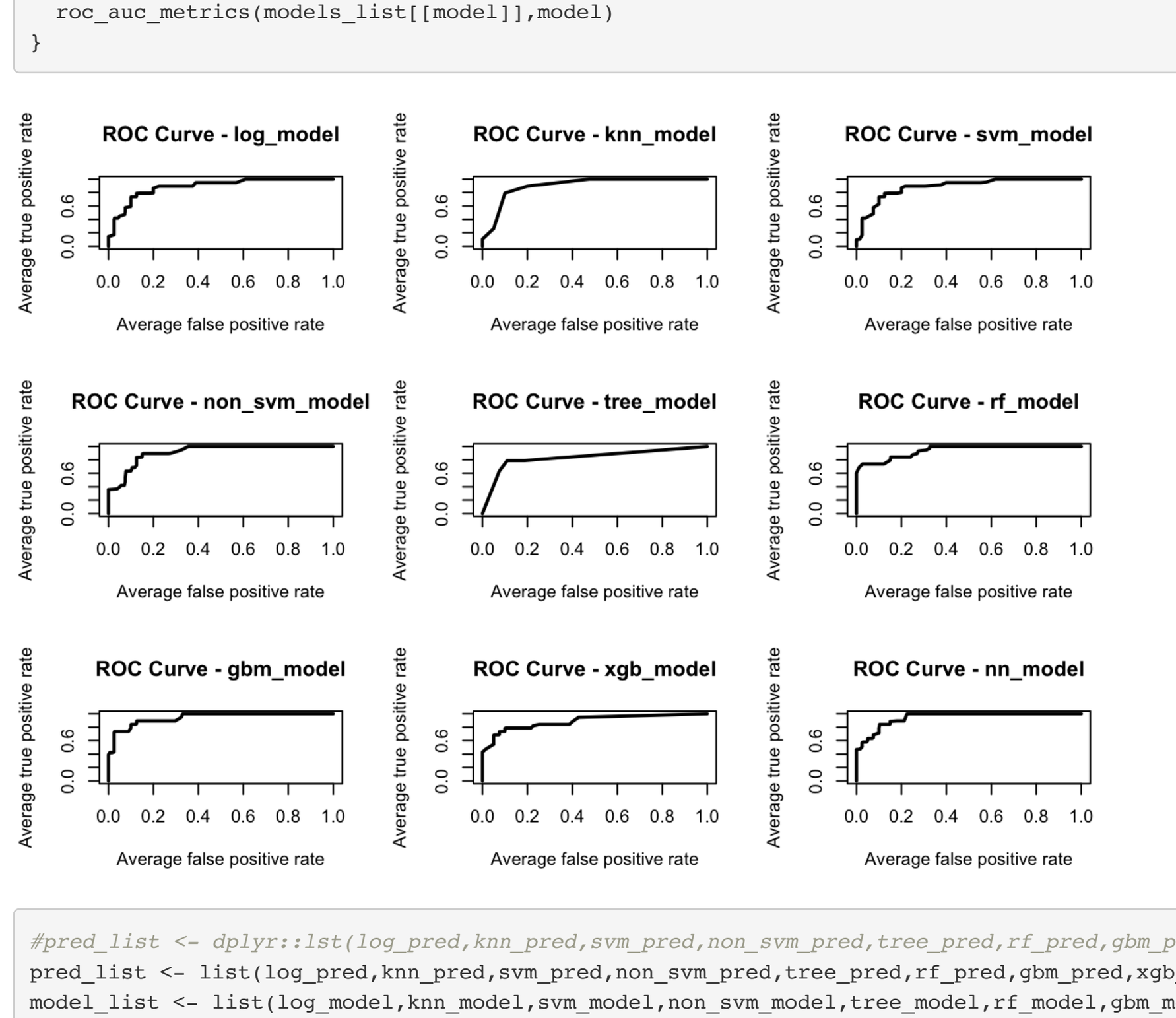
STEP 4: Performance Metrics

To compare the performance of the different models, in order to select a shortlist of the best models, accuracy, sensitivity, specificity and AUC are used to compare.

We first write a function to compute the ROC curve and AUC value for each model to compare the models based on how well they correctly classify the true positives and true negatives.

```
roc_auc_metrics <- function(model, model_name){
  model_pred <- predict(model,newdata = test_data, type="prob")
  pred <- prediction(model_pred[,2], test_data$DEATH_EVENT)
  perf <- performance(pred, measure = "tpr", x.measure = "fpr") #tpr - True Positive Rate/Sensitivity
  plot(perf,avg= "threshold", lwd= 2, main= paste("ROC Curve -",model_name,sep = " "))
  #return(print(auc(test_data$DEATH_EVENT,model_pred[,2])))
  }

models_list <- dplyr::lst(log_model,knn_model,svm_model,non_svm_model,tree_model,rf_model,gbm_model,xgb_model, nn
_model)
par(mfrow=c(3,3))
for(model in names(models_list)){
  roc_auc_metrics(models_list[[model]],model)
}
```




```
#pred_list <- dplyr::lst(log_pred,knn_pred,svm_pred,non_svm_pred,tree_pred,rf_pred,gbm_pred,xgb_pred,nn_pred)
pred_list <- list(log_pred,knn_pred,svm_pred,non_svm_pred,tree_pred,rf_pred,gbm_pred,xgb_pred,nn_pred)
model_list <- list(log_model,knn_model,svm_model,non_svm_model,tree_model,rf_model,gbm_model,xgb_model,nn_model)
algorithms <- c("Logistic Regression","KNN","Linear-SVM","Non-Linear-SVM","Decision Tree","Random Forest","Stocha
stic Gradient Boosting","XGBoost","Neural Networks")

#Metric Scores
accuracy_scores <- c()
kappa_scores <- c()
recall_scores <- c()
specificity_scores <- c()
precision_scores <- c()
f1_scores <- c()
for(model_pred in pred_list){
  confmat <- confusionMatrix(model_pred, test_data$DEATH_EVENT)
  accuracy_scores <- c(accuracy_scores,round(confmat$overall[["Accuracy"]]*100, digits=2))
  kappa_scores <- c(kappa_scores, round(confmat$overall[['Kappa']]*100, digits = 2))
  recall_scores <- c(recall_scores, round(confmat$byClass[["Sensitivity"]]*100, digits=2))
  specificity_scores <- c(specificity_scores, round(confmat$byClass[["Specificity"]]*100, digits=2))
  precision_scores <- c(precision_scores, round(confmat$byClass[["Precision"]]*100, digits=2))
  f1_scores <- c(f1_scores, round(confmat$byClass[["F1"]]*100, digits=2))
}
# AUC Scores
auc_scores <- c()
for(mod in model_list){
  model_pred <- predict(mod,newdata = test_data, type="prob")
  score <- auc(test_data$DEATH_EVENT,model_pred[,2])
  auc_scores <- c(auc_scores, round(score[[1]]*100, digits=2))
}
metrics_results <- data.frame(algorithms, accuracy_scores, kappa_scores, recall_scores, specificity_scores, preci
sion_scores, f1_scores, auc_scores)
metrics_results$algorithms <- as.factor(metrics_results$algorithms)
names(metrics_results) <- c("Models", "Accuracy", "Kappa", "Recall", "Specificity", "Precision", "F1", "AUC")
```

```
library(knitr)
kable(metrics_results)
```

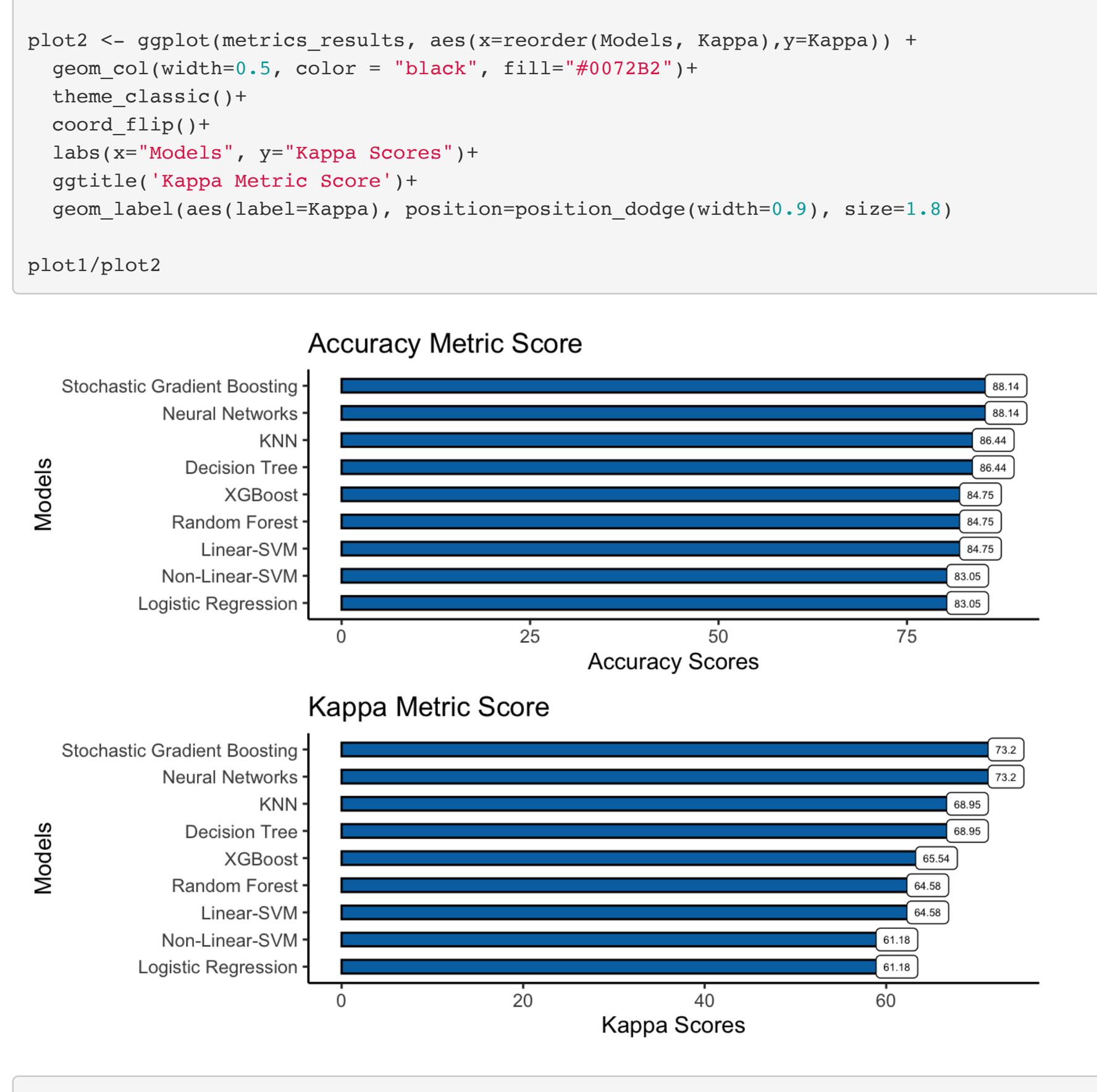
Models	Accuracy	Kappa	Recall	Specificity	Precision	F1	AUC
Logistic Regression	83.05	61.18	87.5	73.68	87.50	87.50	88.95
KNN	86.44	68.95	90.0	78.95	90.00	90.00	90.53
Linear-SVM	84.75	64.58	90.0	73.68	87.80	88.89	88.82
Non-Linear-SVM	83.05	61.18	87.5	73.68	87.50	87.50	91.58
Decision Tree	86.44	68.95	90.0	78.95	90.00	90.00	83.62
Random Forest	84.75	64.58	90.0	73.68	87.80	88.89	93.95
Stochastic Gradient Boosting	88.14	73.20	90.0	84.21	92.31	91.14	94.21
XGBoost	84.75	65.54	87.5	78.95	89.74	88.61	90.26
Neural Networks	88.14	73.20	90.0	84.21	92.31	91.14	94.34

The table breakdown is summarized in the following graphs.

```
# Plotting results
plot1<-ggplot(metrics_results, aes(x=reorder(Models, Accuracy),y=Accuracy)) +
  geom_col(width=0.5, color = "black", fill="#0072B2")+
  theme_classic()+
  coord_flip()+
  labs(x="Models", y="Accuracy Scores")+
  ggtitle('Accuracy Metric Score')+
  geom_label(aes(label=Accuracy), position=position_dodge(width=0.7), size=1.8)

plot2 <- ggplot(metrics_results, aes(x=reorder(Models, Kappa),y=Kappa)) +
  geom_col(width=0.5, color = "black", fill="#0072B2")+
  theme_classic()+
  coord_flip()+
  labs(x="Models", y="Kappa Scores")+
  ggtitle('Kappa Metric Score')+
  geom_label(aes(label=Kappa), position=position_dodge(width=0.9), size=1.8)

plot1/plot2
```

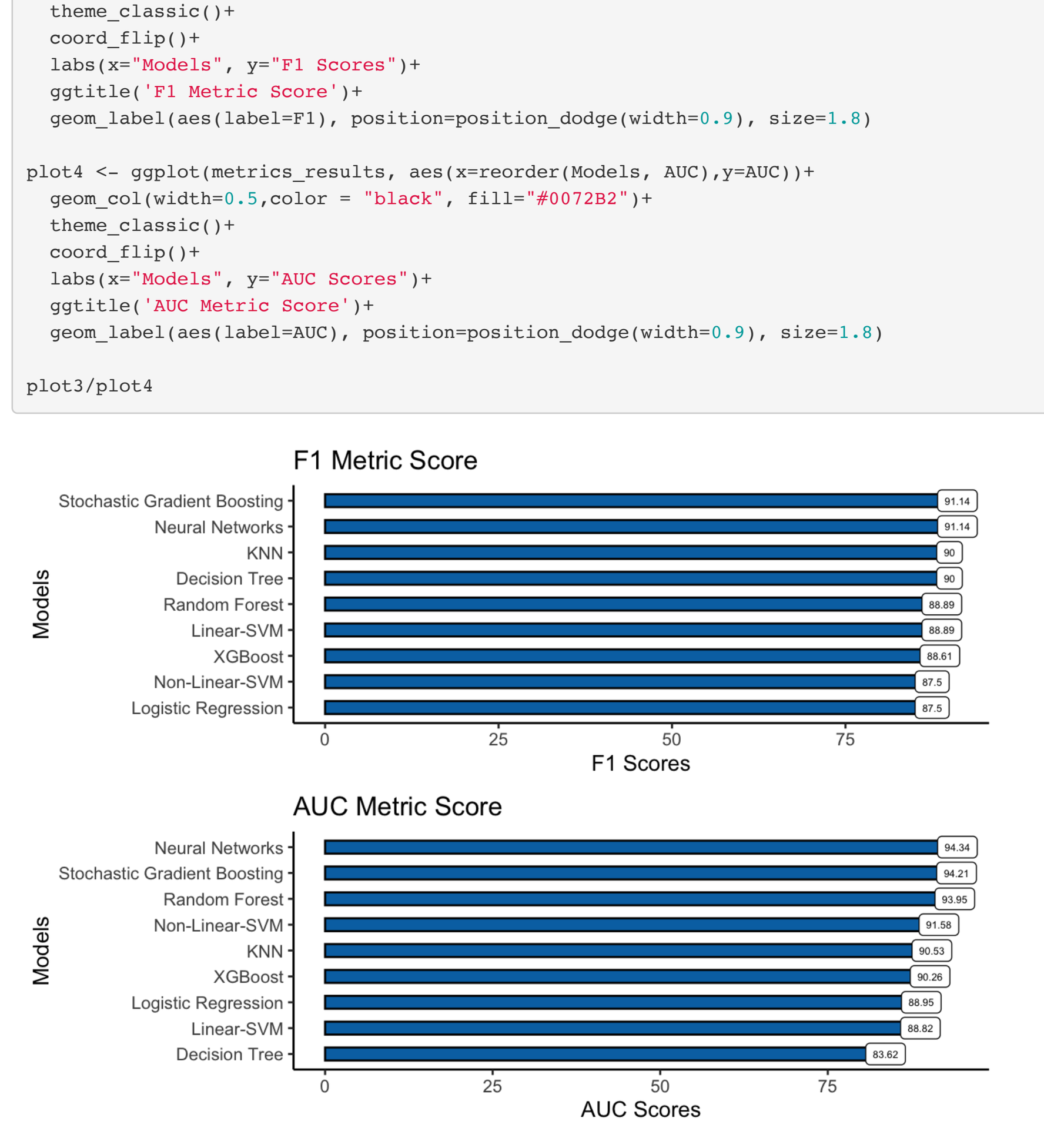


Accuracy Metric Score
Stochastic Gradient Boosting
Neural Networks
KNN
Decision Tree
XGBoost
Random Forest
Linear-SVM
Non-Linear-SVM
Logistic Regression
88.14
88.14
86.44
86.44
84.75
84.75
84.75
83.05
83.05
Models
Accuracy Scores
Kappa Metric Score
73.2
73.2
68.95
68.95
65.54
64.58
64.58
61.18
61.18
Kappa Scores


```
plot3 <- ggplot(metrics_results, aes(x=reorder(Models, F1),y=F1))+
  geom_col(width=0.5, color = "black", fill="#0072B2")+
  theme_classic()+
  coord_flip()+
  labs(x="Models", y="F1 Scores")+
  ggtitle('F1 Metric Score')+
  geom_label(aes(label=F1), position=position_dodge(width=0.9), size=1.8)

plot4 <- ggplot(metrics_results, aes(x=reorder(Models, AUC),y=AUC))+
  geom_col(width=0.5,color = "black", fill="#0072B2")+
  theme_classic()+
  coord_flip()+
  labs(x="Models", y="AUC Scores")+
  ggtitle('AUC Metric Score')+
  geom_label(aes(label=AUC), position=position_dodge(width=0.9), size=1.8)

plot3/plot4
```



F1 Metric Score
Stochastic Gradient Boosting
Neural Networks
KNN
Decision Tree
Random Forest
Linear-SVM
XGBoost
Non-Linear-SVM
Logistic Regression
91.14
91.14
90
90
88.89
88.89
88.61
87.5
87.5
Models
F1 Scores
AUC Metric Score
Neural Networks
Stochastic Gradient Boosting
Random Forest
Non-Linear-SVM
KNN
XGBoost
Logistic Regression
Linear-SVM
Decision Tree
94.34
94.21
93.95
91.58
90.53
90.26
88.95
88.82
83.62
AUC Scores