```
library(ranger)

library(caret)

library(data.table)

creditcard_data <- read.csv("creditcard.csv")

dim(creditcard_data)

head(creditcard_data,6)

tail(creditcard_data,6)

table(creditcard_data$Class)

summary(creditcard_data$Amount)

names(creditcard_data)

var(creditcard_data$Amount)

sd(creditcard_data$Amount)

head(creditcard_data)

creditcard_data$Amount=scale(creditcard_data$Amount)

NewData=creditcard_data[,-c(1)]

head(NewData)

library(caTools)

set.seed(123)

data_sample = sample.split(NewData$Class,SplitRatio=0.80)

train_data = subset(NewData,data_sample==TRUE)

test_data = subset(NewData,data_sample==FALSE)

dim(train_data)

dim(test_data)

creditcard_data$Amount=scale(creditcard_data$Amount)

NewData=creditcard_data[,-c(1)]

head(NewData)

library(caTools)

set.seed(123)

data_sample = sample.split(NewData$Class,SplitRatio=0.80)
```

```
train_data = subset(NewData,data_sample==TRUE)

test_data = subset(NewData,data_sample==FALSE)

dim(train_data)

dim(test_data)

Logistic_Model=glm(Class~.,test_data,family=binomial())

summary(Logistic_Model)

plot(Logistic_Model)

library(pROC)

lr.predict <- predict(Logistic_Model,train_data, probability = TRUE)

auc.gbm = roc(test_data$Class, lr.predict, plot = TRUE, col = "blue")

library(rpart)

library(rpart.plot)

decisionTree_model <- rpart(Class ~ . , creditcard_data, method = 'class')

predicted_val <- predict(decisionTree_model, creditcard_data, type = 'class')

probability <- predict(decisionTree_model, creditcard_data, type = 'prob')

rpart.plot(decisionTree_model)

library(neuralnet)

ANN_model =neuralnet (Class~.,train_data,linear.output=FALSE)

plot(ANN_model)

predANN=compute(ANN_model,test_data)

resultANN=predANN$net.result

resultANN=ifelse(resultANN>0.5,1,0)

library(gbm, quietly=TRUE)

# Get the time to train the GBM model

system.time(

       model_gbm <- gbm(Class ~ .

               , distribution = "bernoulli"

               , data = rbind(train_data, test_data)

               , n.trees = 500

               , interaction.depth = 3
```

```
            , n.minobsinnode = 100
            , shrinkage = 0.01
            , bag.fraction = 0.5
            , train.fraction = nrow(train_data) / (nrow(train_data) + nrow(test_data))
)
)
# Determine best iteration based on test data
gbm.iter = gbm.perf(model_gbm, method = "test")
model.influence = relative.influence(model_gbm, n.trees = gbm.iter, sort. = TRUE)
#Plot the gbm model
plot(model_gbm)
# Plot and calculate AUC on test data
gbm_test = predict(model_gbm, newdata = test_data, n.trees = gbm.iter)
gbm_auc = roc(test_data$Class, gbm_test, plot = TRUE, col = "red")
print(gbm_auc)
```