# Detecting Credit Card Fraud

# Importing Datasets

library(ranger)

library(caret)

library(data.table)

library(readxl)

creditcard <- read\_excel("D:/PROJECT/Credit-card-dataset/creditcard.xlsx")

View(creditcard)

creditcard\_data <-creditcard

# Data Exploration

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)

# Data Manipulation

head(creditcard\_data)

creditcard\_data$Amount=scale(creditcard\_data$Amount)

NewData=creditcard\_data[,-c(1)]

head(NewData)

# Data Modelling

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)

# Fitting Logistic Regression Model

Logistic\_Model=glm(Class~.,test\_data,family=binomial())

summary(Logistic\_Model)

# Visualizing summarized model through the following plots

plot(Logistic\_Model)

# ROC Curve to assess the performance of the model

library(pROC)

lr.predict <- predict(Logistic\_Model,test\_data, probability = TRUE)

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

# Fitting a Decision Tree Model

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)

# Artificial Neural Network

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)

# Gradient Boosting (GBM)

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

library(pROC)

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)