##**Fraud prevention Model using Neural Network**

```
install.packages("caret")
install.packages("MASS")
install.packages("forecast", dependencies = TRUE)
install.packages("MLmetrics")
install.packages("leaps")
install.packages("DALEX")
install.packages("InformationValue")
install.packages("ROCR")
install.packages("gains")
install.packages("neuralnet")
library(neuralnet)
library(nnet)
library(ROCR)
library(InformationValue)
library(DALEX)
library(leaps)
library(MLmetrics)
library(forecast)
library(caret)
library(readxl)
fraud_Mix <- read_excel("~/Desktop/myproject/fraud_Mix.xlsx")
View(fraud_Mix)
summary(fraud_Mix)
mean(fraud_Mix$Sales)
fraud_Mix$new_y <- ifelse(fraud_Mix$Sales > 697.1303, 0, 1)

fraud_nn<- fraud_Mix[-c(1,2)]
fraud_nn

# create detect outlier function
detect_outlier <- function(x) {
  Quantile1 <- quantile(x, probs=.25)
  Quantile3 <- quantile(x, probs=.75)
  IQR = Quantile3-Quantile1
  x > Quantile3 + (IQR*1.5) | x < Quantile1 - (IQR*1.5)
}

remove_outlier <- function(dataframe,
                           columns=names(dataframe)) {
  for (col in columns) {
    dataframe <- dataframe[!detect_outlier(dataframe[[col]]), ]
  }
```

```
  # return dataframe
  print("Remove outliers")
  print(dataframe)
}
fraud_nn_outlier <- remove_outlier(fraud_nn, c(6))
nrow(fraud_nn_outlier)



##### Data preprocessing using standardization
fraud_nn_outlier$one <- fraud_nn_outlier$new_y ==1
fraud_nn_outlier$zero <- fraud_nn_outlier$new_y ==0

set.seed(1)
sample_data <- sample(c(1:647), 457)
fraud_train_data <- fraud_nn_outlier[sample_data, ]
fraud_test_data <- fraud_nn_outlier[-sample_data, ]

#-----------------------------------------------------------------------
#first Iteration
set.seed(1)
frd.nn.1 <- neuralnet(new_y ~ No_transaction + frequency + average_trans + time + Location +
IP_address
             + V1 + V2, data = fraud_train_data, linear.output = F, hidden = c(6,3))
frd.nn.1$weights

# display predictions
prediction(frd.nn.1)


# plot network
plot(frd.nn.1, rep="best")

# Getting out the predictions
pred.1 <- predict(frd.nn.1, fraud_train_data)

tt<-table(ifelse(pred.1 > 0.70, 1, 0), fraud_train_data$new_y)

sum(diag(tt))/sum(tt)

# The Performance Metrics
performance_Metrics1 <- t(data.frame("Accuracy" = sum(diag(tt))/sum(tt),
                  "Error" = 1 - (sum(diag(tt))/sum(tt)),
```

```
                    "Sensitivity" = tt[2,2] / colSums(tt)[2],
                    "Specificity" = tt[1,1] / colSums(tt)[1],
                    "Precision" = tt[2,2] / rowSums(tt)[2],
                    "F1_Score" = 2 * ((tt[2,2] / colSums(tt)[2])*(tt[2,2] / rowSums(tt)[2]))/
                      ((tt[2,2] / colSums(tt)[2])+(tt[2,2] / rowSums(tt)[2])),
                    "Success_Class" = 1,
                    "Success_Prob" = 0.5))

colnames(performance_Metrics1) <- ""
pMetrics.1


plotROC(fraud_test_data$new_y, log_pred)

#second iteration
#---------------------------------------------------------------
# Preprocessing Normalization
train_data_scaled <- cbind(scale(fraud_train_data[, 1:8]), fraud_train_data[, 9:11])
train_data_scaled

frd.nn.2 <- neuralnet(new_y ~ No_transaction + frequency + average_trans + time + Location +
IP_address
                        +V1 + V2, data = train_data_scaled, linear.output = F, hidden = c(10,8))

frd.nn.2$weights

# display predictions
prediction(frd.nn.2)


# plot network
plot(frd.nn.2, rep="best")

# Getting out the predictions



plot(frd.nn.2, rep="best")

# Getting out the predictions
pred.2 <- predict(frd.nn.2, train_data_scaled)

tt2<-table(ifelse(pred.2 > 0.58, 1, 0), train_data_scaled$new_y)

sum(diag(tt2))/sum(tt2)
```

```
# The Performance Metrics
performance_Metrics2 <- t(data.frame("Accuracy" = sum(diag(tt2))/sum(tt2),
                                     "Error" = 1 - (sum(diag(tt2))/sum(tt2)),
                                     "Sensitivity" = tt2[2,2] / colSums(tt2)[2],
                                     "Specificity" = tt2[1,1] / colSums(tt2)[1],
                                     "Precision" = tt2[2,2] / rowSums(tt2)[2],
                                     "F1_Score" = 2 * ((tt2[2,2] / colSums(tt2)[2])*(tt2[2,2] / rowSums(tt2)[2]))/
                                       ((tt2[2,2] / colSums(tt2)[2])+(tt2[2,2] / rowSums(tt2)[2])),
                                     "Success_Class" = 1,
                                     "Success_Prob" = 0.5))

colnames(performance_Metrics2) <- ""
performance_Metrics2

plotROC(test_data_scaled$new_y, pred.2)

####----------------------------------------------------------------
##Third iteration(with validation dataset)

test_data_scaled <- cbind(scale(fraud_test_data[, 1:8]), fraud_test_data[, 9:11])
test_data_scaled

pred.3 <- predict(frd.nn.2, test_data_scaled)
pred.3

tt3<-table(ifelse(pred.3 > 0.58, 1, 0), test_data_scaled$new_y)
sum(diag(tt3))/sum(tt3)

performance_Metrics_test <- t(data.frame("Accuracy" = sum(diag(tt3))/sum(tt3), "Error" = 1 -
(sum(diag(tt3))/sum(tt3)),
"Sensitivity" = tt3[2,2] / colSums(tt3)[2],
"Specificity" = tt3[1,1] / colSums(tt3)[1],
 "Precision" = tt3[2,2] / rowSums(tt3)[2],
 "F1_Score" = 2 * ((tt3[2,2] / colSums(tt3)[2])*(tt3[2,2] / rowSums(tt3)[2]))/((tt3[2,2] /
colSums(tt3)[2])*(tt3[2,2] / rowSums(tt3)[2])),
"Success_Class" = 1,
 "Success_Prob" = 0.58))

performance_Metrics_test

colnames(performance_Metrics_test) <- ""
performance_Metrics_test
```

```
# The ROC Chart for the Validation Data
plotROC(test_data_scaled$new_y, pred.3)

# The lift Chart for the validation data

gain2 <- gains(actual=test_data_scaled$new_y, predicted=pred.3, groups = 10)
plot(c(0, gain2$cume.pct.of.total* sum(test_data_scaled$new_y))~c(0,gain2$cume.obs),
xlab=" # cases",ylab="cumulative", main="Gain Chart(Performance) of a Nueral Network",type =
"l", col = 'red')
lines (c(0, sum(test_data_scaled$new_y))~c(0, dim(test_data_scaled)[1]), lty = 2)
plotROC(fraud_train_data$new_y, pred.1)
```