ERF Paper

Tegveer Ghura

2023-02-10

```
# df import and subset
df <- read_csv('Data/REKT_Database_Clean_Python.csv')
df <- subset(df, select = -c(...1, token_name, description, name_categories))
#df <- df %>% filter(funds_lost!=0)
```

```
# Removing dictionary values from the scam_type column
df$scam_type <- gsub("[^:]*,[^:]*", "",df$scam_type)
df$scam_type <- gsub("'id'::", "",df$scam_type)
df$scam_type <- gsub("\\{|\\}", "",df$scam_type)
df$scam_type <- gsub("'", "",df$scam_type)
df$scam_type <- gsub("type: ", "",df$scam_type)
df$scam_type <- gsub(" ", "",df$scam_type)

# Removing list brackets from the scamNetworks column
df$scamNetworks <- gsub("\\[|\\]", "", df$scamNetworks)
df$scamNetworks <- gsub("'", '', df$scamNetworks)
df$scamNetworks <- gsub(", +", ",", df$scamNetworks) # remove whitespace after comma for grouping later
```

```
# pooling together scam types into respective types
df <- df %>%
  mutate(scam_type_grouped = if_else(scam_type=="Honeypot" | scam_type=="Rugpull" | scam_type=="Abandone
df <- subset(df, select = -c(scam_type, day_of_week_of_attack, day_of_year_of_attack, date, project_name
table(df$scam_type_grouped)
```

```
##
## Exit Scam   Exploit
##      2677       486
```

```
#only month_of_attack has NA's (1873 of them), we can impute "unknown" for them or get rid of the colum
df$month_of_attack=month.name[df$month_of_attack]
df$month_of_attack[is.na(df$month_of_attack)] <- "Unknown"
#df <- na.omit(df)
```

```
# pooling scamNetworks into 5 levels (Eth, binance, polygon, other centralized, other decentralized)
df <- separate_rows(df,scamNetworks,sep = ",")
df <- df %>%
  mutate(scam_networks_grouped = if_else(scamNetworks == "Avax" | scamNetworks == "Algorand" | scamNetwo
df <- df %>% filter(scam_networks_grouped != "") # remove empty string level
df <- subset(df, select = -c(scamNetworks))
```

```
# specify dtypes before train test split

df$scam_networks_grouped <- as.factor(df$scam_networks_grouped)
df$scam_type_grouped <-as.factor(df$scam_type_grouped)
```

```
df$month_of_attack <-as.factor(df$month_of_attack)

# add +1 because we have zeros in funds_returned and helps avoid negative inf values

df$log_funds_lost <- log(df$funds_lost + 1)
df$log_funds_returned <- log(df$funds_returned + 1)
df <- subset(df, select = -c(funds_lost, funds_returned))
```

```
library(caret)

set.seed(3738)

df <- df[sample(1:nrow(df)), ] # shuffle rows

train.index <- createDataPartition(df$scam_networks_grouped,
                                   p = .8, list = FALSE)
train <- df[ train.index,]
test  <- df[-train.index,]

x_train <- train %>% select(log_funds_lost, log_funds_returned,
                            scam_networks_grouped)
y_train <- train$scam_type_grouped

x_test <- test %>% select(log_funds_lost, log_funds_returned,
                          scam_networks_grouped)
y_test <- test$scam_type_grouped

classifier_RF <- randomForest(x = x_train,
                             y = y_train,
                             ntree = 500)

classifier_RF
```

```
##
## Call:
##  randomForest(x = x_train, y = y_train, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 7.53%
## Confusion matrix:
##           Exit Scam Exploit class.error
## Exit Scam      2052      98   0.0455814
## Exploit          95     318   0.2300242
```

```
# Predicting the Test set results
y_pred = predict(classifier_RF, newdata = x_test)

# Confusion Matrix
confusion_mtx = table(y_test, y_pred)
confusion_mtx
```

```
##            y_pred
## y_test      Exit Scam Exploit
##   Exit Scam       504      23
##   Exploit          37      73
```

```r
library(caret)
set.seed(2377)
train.index <- createDataPartition(df$scam_networks_grouped,
                                   p = .8, list = FALSE)
train <- df[ train.index,]
test  <- df[-train.index,]

train$scam_type_grouped = ifelse(train$scam_type_grouped  == "Exploit", 1, 0)
test$scam_type_grouped = ifelse(test$scam_type_grouped  == "Exploit", 1, 0)
logistic_model <- glm(scam_type_grouped ~ ., data = train, family = "binomial")
summary(logistic_model)
```

```
##
## Call:
## glm(formula = scam_type_grouped ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5795  -0.1964  -0.1090  -0.1090   2.9504
##
## Coefficients:
##                                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)                              -2.72453    0.56944  -4.785 1.71e-06
## month_of_attackAugust                    -1.15261    0.55455  -2.078 0.037666
## month_of_attackDecember                  -1.39598    0.54601  -2.557 0.010567
## month_of_attackFebruary                  -0.70712    0.57729  -1.225 0.220612
## month_of_attackJanuary                   -2.01204    0.58020  -3.468 0.000525
## month_of_attackJuly                      -0.31824    0.55894  -0.569 0.569103
## month_of_attackJune                      -0.47891    0.56253  -0.851 0.394571
## month_of_attackMarch                     -0.79979    0.56472  -1.416 0.156696
## month_of_attackMay                       -0.55392    0.56750  -0.976 0.329028
## month_of_attackNovember                  -1.25996    0.53871  -2.339 0.019342
## month_of_attackOctober                   -0.83424    0.53809  -1.550 0.121054
## month_of_attackSeptember                 -1.21387    0.55567  -2.185 0.028924
## month_of_attackUnknown                   -2.39913    0.62875  -3.816 0.000136
## scam_networks_groupedCentralized          0.97793    0.26961   3.627 0.000287
## scam_networks_groupedEthereum             0.78431    0.19070   4.113 3.91e-05
## scam_networks_groupedOther Decentralized  1.41432    0.32384   4.367 1.26e-05
## scam_networks_groupedPolygon              0.99678    0.40332   2.471 0.013456
## log_funds_lost                            0.23462    0.02149  10.918  < 2e-16
## log_funds_returned                        0.11347    0.03510   3.233 0.001225
##
## (Intercept)                              ***
## month_of_attackAugust                    *
## month_of_attackDecember                  *
## month_of_attackFebruary
## month_of_attackJanuary                   ***
## month_of_attackJuly
## month_of_attackJune
## month_of_attackMarch
## month_of_attackMay
## month_of_attackNovember                  *
## month_of_attackOctober
## month_of_attackSeptember                 *
## month_of_attackUnknown                   ***
## scam_networks_groupedCentralized         ***
## scam_networks_groupedEthereum            ***
## scam_networks_groupedOther Decentralized ***
## scam_networks_groupedPolygon             *
## log_funds_lost                           ***
## log_funds_returned                       **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2246.8  on 2562  degrees of freedom
## Residual deviance: 1064.8  on 2544  degrees of freedom
## AIC: 1102.8
##
## Number of Fisher Scoring iterations: 7
```

```
train_prob_pred <- predict(logistic_model, type = 'response', newdata = train)
test_prob_pred <- predict(logistic_model, type = 'response', newdata = test)
#y_pred = ifelse(prob_pred > 0.5, "Exploit", "Exit Scam")

# Train Confusion Matrix
y_train_pred = ifelse(train_prob_pred > 0.5, 1, 0)
y_train_pred<- as.factor(y_train_pred)
train$scam_type_grouped <- as.factor(train$scam_type_grouped)
(cm = table(train$scam_type_grouped, y_train_pred))
```

```
##    y_train_pred
##        0    1
##   0 2068   87
##   1  128  280
```

```
# Test Confusion Matrix
y_test_pred = ifelse(test_prob_pred > 0.5, 1, 0)
y_test_pred<- as.factor(y_test_pred)
test$scam_type_grouped <- as.factor(test$scam_type_grouped)
(cm = table(test$scam_type_grouped, y_test_pred)) # NAs ignored
```

```
##    y_test_pred
##       0   1
##   0 508  14
##   1  38  77
```

```
#y_pred <- as.factor(unname(y_pred)) # for cfm plot
```

```
# 1. Open jpeg file
#jpeg("Train_CFM.jpg", width = 350, height = 350)

library(scales)
```

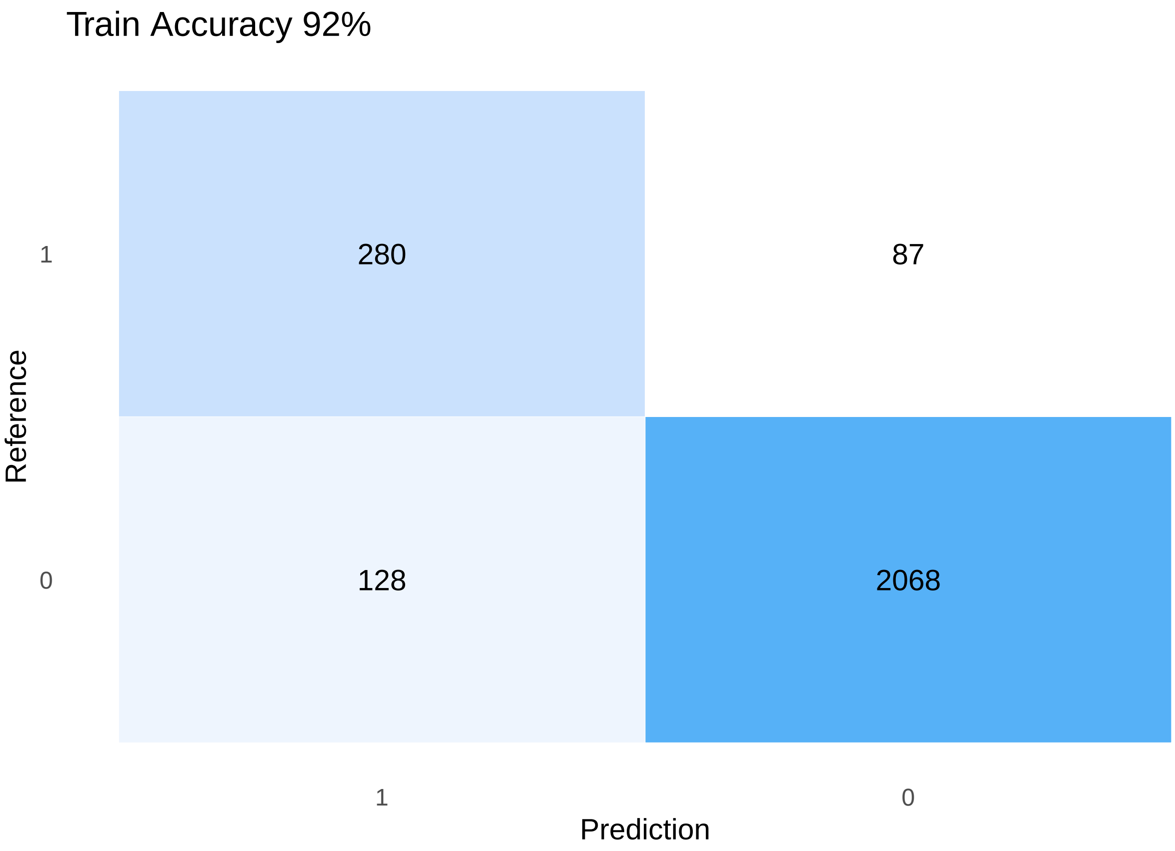
```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
```

```
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

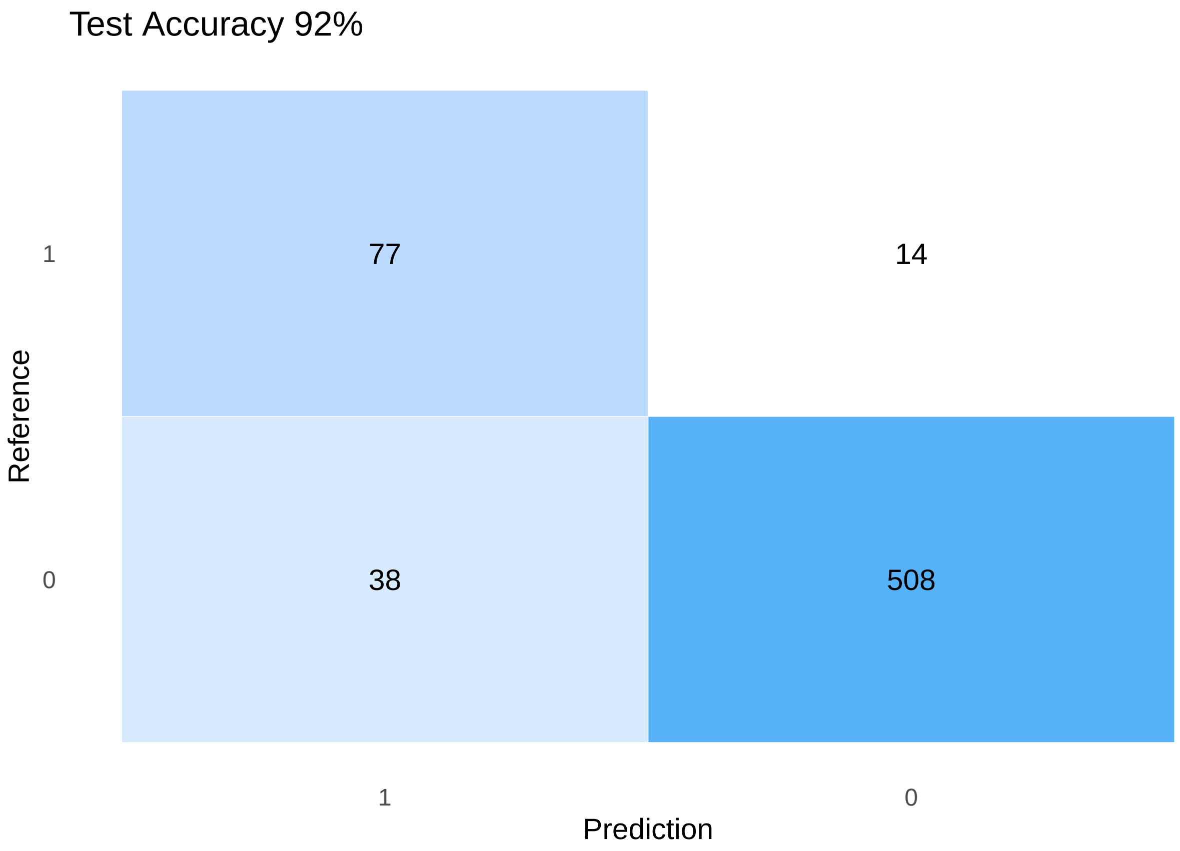
```
ggplotConfusionMatrix <- function(m){
  mytitle <- paste("Train Accuracy", percent_format()(m$overall[1]))
  p <-
    ggplot(data = as.data.frame(m$table) ,
           aes(x = Prediction, y = Reference)) +
    geom_tile(aes(fill = log(Freq)),
              colour = "white", show.legend = FALSE) +
    scale_fill_gradient(low = "white", high = "#56B1F7") +
    geom_text(aes(x = Prediction, y = Reference,
                  label = Freq)) +
    ggtitle(mytitle) +
    scale_x_discrete(limits = rev) +
    theme_minimal() +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank())
  return(p)
}
cfm_train <- confusionMatrix(train$scam_type_grouped, y_train_pred)
ggplotConfusionMatrix(cfm_train)
```



```
# Close the jpeg file
#dev.off()
```

```
# Open jpeg file
#jpeg("Test_CFM.jpg", width = 350, height = 350)

library(scales)
ggplotConfusionMatrix <- function(m){
  mytitle <- paste("Test Accuracy", percent_format()(m$overall[1]))
  p <-
    ggplot(data = as.data.frame(m$table) ,
           aes(x = Prediction, y = Reference)) +
    geom_tile(aes(fill = log(Freq)),
              colour = "white", show.legend = FALSE) +
    scale_fill_gradient(low = "white", high = "#56B1F7") +
    geom_text(aes(x = Prediction, y = Reference,
                  label = Freq)) +
    ggtitle(mytitle) +
    scale_x_discrete(limits = rev) +
    theme_minimal() +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank())
  return(p)
}
cfm_test <- confusionMatrix(test$scam_type_grouped, y_test_pred)
ggplotConfusionMatrix(cfm_test)
```



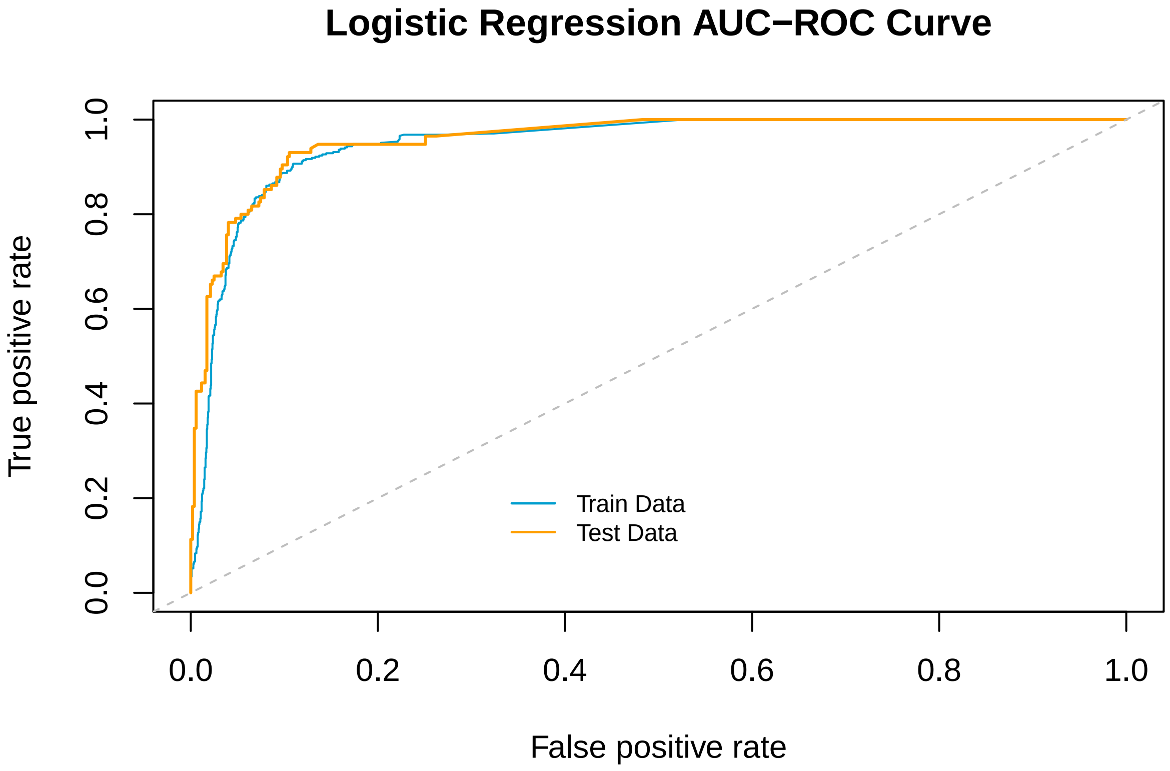
```
# Close the jpeg file
#dev.off()
```

```
# Train and Test Data ROC-AUC Curve
train_pred <- prediction(train_prob_pred, train$scam_type_grouped)
test_pred <- prediction(test_prob_pred, test$scam_type_grouped)

# Create an ROC curve
perf_train <- performance(train_pred, measure = "tpr", x.measure = "fpr")
perf_test <- performance(test_pred, measure = "tpr", x.measure = "fpr")

# Open a pdf and jpeg file
#pdf("ROC.pdf", width = 6.5, height = 4.24)
#jpeg("ROC.jpg", width = 700, height = 350)


# Plot the ROC curve
plot(perf_train, main = "Logistic Regression AUC-ROC Curve",
    col = "#009ECE")
plot(perf_test, add = T, col = "#FF9E00", lwd = 1.5)
legend(0.32, 0.25, c("Train Data", "Test Data"),
       col = c("#009ECE", "#FF9E00"),
       bty = "n", lwd = 1.2, cex = 0.75)
abline(0, 1, lty = 2, col = "gray") # Add y=x line
```



```
# Close the pdf/jpeg file
#dev.off()
```

```
auc.train <- auc(train$scam_type_grouped, train_prob_pred)
cat("Area under the curve for Logistic Regression Train Set is: ", auc.train)
```

```
## Area under the curve for Logistic Regression Train Set is:  0.9500398
```

```r
auc.test <- auc(train$scam_type_grouped, train_prob_pred)
cat("\nArea under the curve for Logistic Regressio  Test Set is: ", auc.test)
```

```
##
## Area under the curve for Logistic Regressio  Test Set is:  0.9500398
```