

Fraud Detection

Santi

2022-06-20

```
loadLibraries() #This is just a function that loads essential stats libraries.

## [1] "The following libraries were loaded: ISLR2,ggplot2, dplyr,\n      glmnet, leaps, boot, gam, tree,
fraud <- read.csv('fraud_detection.csv', header = TRUE)
head(fraud)

##   step      type    amount   nameOrig oldbalanceOrg newbalanceOrig
## 1    1 PAYMENT 9839.64 C1231006815        170136     160296.36
## 2    1 PAYMENT 1864.28 C1666544295        21249      19384.72
## 3    1 TRANSFER 181.00 C1305486145         181        0.00
## 4    1 CASH_OUT 181.00 C840083671         181        0.00
## 5    1 PAYMENT 11668.14 C2048537720        41554     29885.86
## 6    1 PAYMENT 7817.71 C90045638        53860     46042.29
##   nameDest oldbalanceDest newbalanceDest isFraud isFlaggedFraud
## 1 M1979787155          0          0        0        0
## 2 M2044282225          0          0        0        0
## 3 C553264065           0          0        1        0
## 4 C38997010          21182          0        1        0
## 5 M1230701703          0          0        0        0
## 6 M573487274           0          0        0        0

# Look at data dims.
dim(fraud)

## [1] 6362620      11

# See if there are any NA's and duplicate entries.
sum(is.na(fraud))

## [1] 0

sum(duplicated(fraud))

## [1] 0
```

Data Summary and Graphs

Let's make a summary for the data and convert the *isFraud* and *isFlaggedFraud* to factors, so we can run the classification models.

```
summary(fraud)

##      step      type            amount   nameOrig
##  Min.   : 1.0  Length:6362620   Min.   :    0  Length:6362620
##  1st Qu.:156.0 Class :character  1st Qu.: 13390 Class :character
```

```

## Median :239.0   Mode  :character   Median : 74872   Mode  :character
## Mean   :243.4               Mean   : 179862
## 3rd Qu.:335.0               3rd Qu.: 208721
## Max.   :743.0               Max.   :92445517
## oldbalanceOrg    newbalanceOrig      nameDest
## Min.    :     0   Min.    :     0   Length:6362620
## 1st Qu.:     0   1st Qu.:     0   Class :character
## Median : 14208   Median :     0   Mode   :character
## Mean   : 833883  Mean   : 855114
## 3rd Qu.: 107315  3rd Qu.: 144258
## Max.   :59585040  Max.   :49585040
## oldbalanceDest   newbalanceDest      isFraud
## Min.    :     0   Min.    :     0   Min.   :0.000000
## 1st Qu.:     0   1st Qu.:     0   1st Qu.:0.000000
## Median : 132706  Median : 214661  Median :0.000000
## Mean   : 1100702  Mean   : 1224996  Mean   :0.001291
## 3rd Qu.: 943037  3rd Qu.: 1111909  3rd Qu.:0.000000
## Max.   :356015889 Max.   :356179279  Max.   :1.000000
## isFlaggedFraud
## Min.   :0.0e+00
## 1st Qu.:0.0e+00
## Median :0.0e+00
## Mean   :2.5e-06
## 3rd Qu.:0.0e+00
## Max.   :1.0e+00

fraud$isFraud <- as.factor(fraud$isFraud)

```

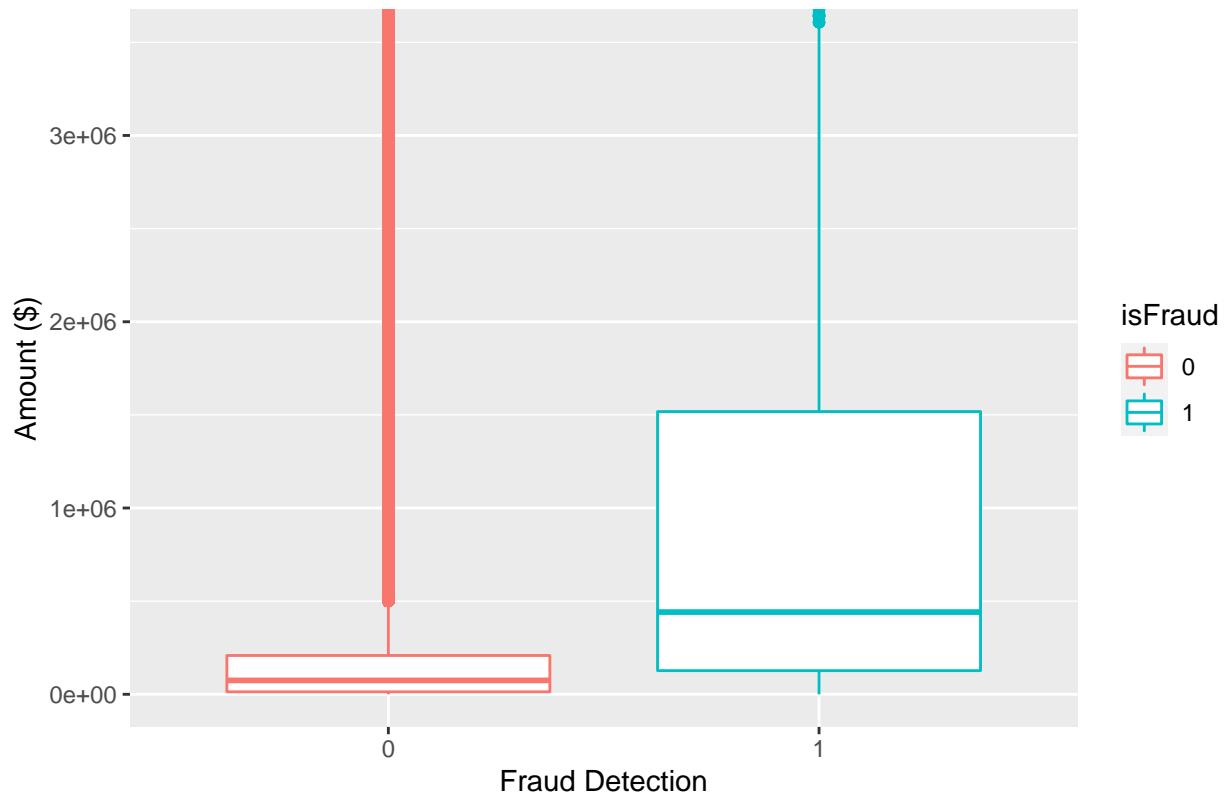
Now we can start our exploratory analysis. The transaction amount has a large number of outliers, so we will restrict the amount value to \$3,500,000.

```

ggplot(data = fraud, aes(x = isFraud, y = amount, color = isFraud)) +
  coord_cartesian(ylim = c(0, 3500000)) +
  geom_boxplot() +
  ylab('Amount ($)') +
  xlab('Fraud Detection') +
  ggtitle('Boxplot of Transaction Amounts and Fraud Detection')

```

Boxplot of Transaction Amounts and Fraud Detection



```
flagged_table <- table(fraud$isFraud, fraud$isFlaggedFraud)
flagged_table

##          0      1
## 0 6354407 0
## 1 8197    16

correct_flag <- table(fraud$isFraud, fraud$isFlaggedFraud)[2,2] /
                  sum(table(fraud$isFraud, fraud$isFlaggedFraud)[2,])
cat(correct_flag * 100, "% of the fraudulent transactions were flagged as such.")

## 0.1948131 % of the fraudulent transactions were flagged as such.
```

Note the system we currently have does poorly when at detecting fraudulent transactions (0.19%). The names of the person making and receiving the transactions are also recorded but they offer no significance so we remove those from the data set.

```
fraudNew <- subset(fraud, select = - c(nameOrig, nameDest))
attach(fraudNew)

## The following objects are masked from fraudNew (pos = 3):
## 
##   amount, isFlaggedFraud, isFraud, newbalanceDest, newbalanceOrig,
##   oldbalanceDest, oldbalanceOrg, step, type

## The following objects are masked from fraudNew (pos = 4):
## 
##   amount, isFlaggedFraud, isFraud, newbalanceDest, newbalanceOrig,
```

```

##      oldbalanceDest, oldbalanceOrg, step, type
## The following objects are masked from fraudNew (pos = 5):
##      amount, isFlaggedFraud, isFraud, newbalanceDest, newbalanceOrig,
##      oldbalanceDest, oldbalanceOrg, step, type
dim(fraudNew)

## [1] 6362620      9

```

Now lets run some data analysis. First let's make a train and test set. Let's randomly select 100,000 observations as our training data and the rest as test data. Why? Well if I do a larger amount my computer really slows down.

```

set.seed(3)

train <- sample(1:nrow(fraudNew), 100000)
test <- (-train)

fraud_train <- fraudNew[train, ]
fraud_test <- fraudNew[test, ]

isFraud_test <- fraudNew[test, 'isFraud']

```

#Logistic Regression

First let's do logistic regression to detect fraudulent transactions. Since this data has a high null rate, we know we will run into issues.

```

null_tab <- table(isFraud)

null_rate <- table(isFraud_test)[1]/length(isFraud_test)
null_rate

##          0
## 0.9987101

set.seed(3)

fraud_glm = glm(isFraud ~ ., data = fraud_train,
                 family = binomial)

## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
summary(fraud_glm)

##
## Call:
## glm(formula = isFraud ~ ., family = binomial, data = fraud_train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -8.49     0.00     0.00     0.00     8.49
##
## Coefficients: (1 not defined because of singularities)
##                  Estimate Std. Error   z value Pr(>|z|)
## (Intercept) -2.117e+15 7.425e+05 -2.851e+09 <2e-16 ***

```

```

## step      1.892e+11  1.501e+03  1.261e+08  <2e-16 ***
## typeCASH_OUT 1.810e+15  7.458e+05  2.427e+09  <2e-16 ***
## typeDEBIT   -2.185e+15  2.726e+06 -8.017e+08  <2e-16 ***
## typePAYMENT  1.778e+15  7.412e+05  2.399e+09  <2e-16 ***
## typeTRANSFER -1.888e+15  1.020e+06 -1.852e+09  <2e-16 ***
## amount      1.399e+07  6.951e-01  2.013e+07  <2e-16 ***
## oldbalanceOrg 8.560e+08  1.583e+00  5.409e+08  <2e-16 ***
## newbalanceOrig -9.814e+08 1.587e+00 -6.184e+08  <2e-16 ***
## oldbalanceDest 8.134e+07  4.332e-01  1.878e+08  <2e-16 ***
## newbalanceDest -8.770e+07 4.287e-01 -2.046e+08  <2e-16 ***
## isFlaggedFraud NA        NA        NA        NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 2053.9 on 99999 degrees of freedom
## Residual deviance: 11678.1 on 99989 degrees of freedom
## AIC: 11700
##
## Number of Fisher Scoring iterations: 25
# Predict the responders that are diagnosed with heart disease.
glm_pred = predict(fraud_glm, data = fraudNew[test, ], type = 'response')
vec = rep(0, length(isFraud))
vec[glm_pred >= 0.2] = 1

table(vec, isFraud)

## isFraud
## vec      0      1
##   0 6345563    8203
##   1    8844     10

```

#Bagging

Let's first do bagging with 100 trees.

```

set.seed(3)
bag_fraud <- randomForest(isFraud ~ ., data = fraudNew, subset = train,
                           mtry = 8, ntree = 100, importance = TRUE)

```

Once that is done, predict the remaining fraud cases. We use `type = 'class'` since we are doing classification. This will allow us to make a table to determine accuracy (see below).

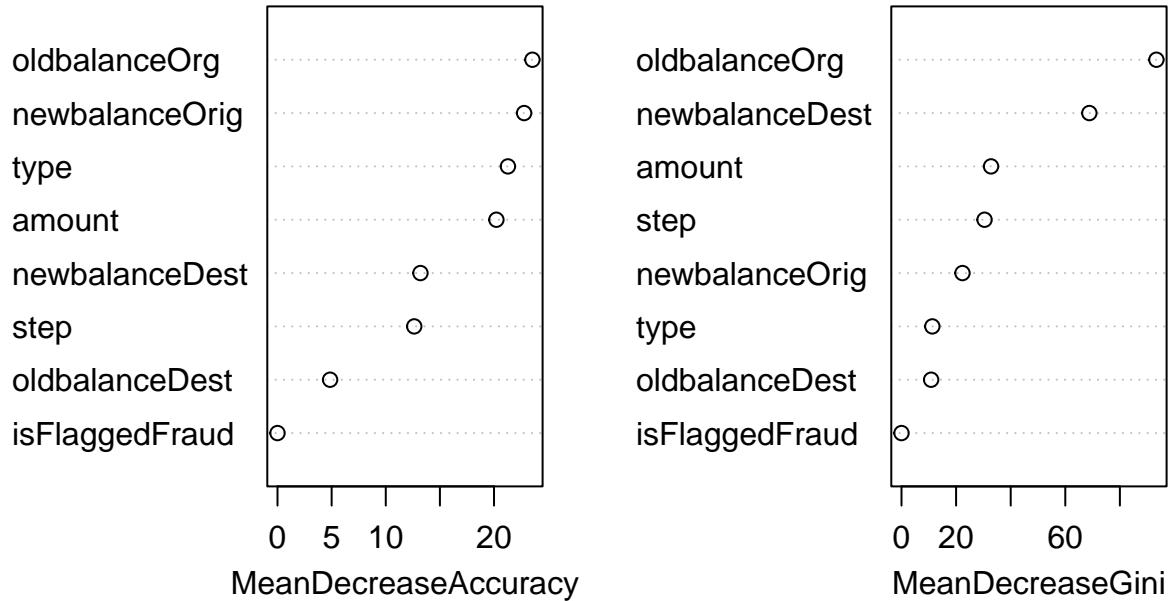
```

set.seed(3)

yhat_bag <- predict(bag_fraud, newdata = fraud_test, type = 'class')
varImpPlot(bag_fraud, main = 'Predictor Importance Using Bagging')

```

Predictor Importance Using Bagging



In this case note that Mean Decrease Accuracy is low for the first 4 predictors: old balance of the account initiating the transaction (*oldbalanceOrg*) , the new balance of the destination account holder, (*newbalanceDest*), the type of transaction *type* and the transaction amount (*amount*) and then follow a large jump.

```
# Table
bag_table <- table('Prediction' = yhat_bag, 'True Value' = isFraud_test)
bag_table
```

```
##          True Value
## Prediction      0      1
##             0 6254275  2203
##             1     267  5875
sum(diag(bag_table))/ sum(bag_table)
```

```
## [1] 0.9996056
bag_table[2,2]/ sum(bag_table[2, ])
```

```
## [1] 0.9565288
```

```
#Random Forests
```

Let's try random forests on the training data set. We allow the model to pick $\sqrt{8} \approx 3$ features at each node. With 1,000 trees.⁶

```
set.seed(3)
rf_fraud <- randomForest(isFraud ~ ., data = fraudNew, subset = train,
                           mtry = 3, ntree = 100, importance = TRUE)
```

```

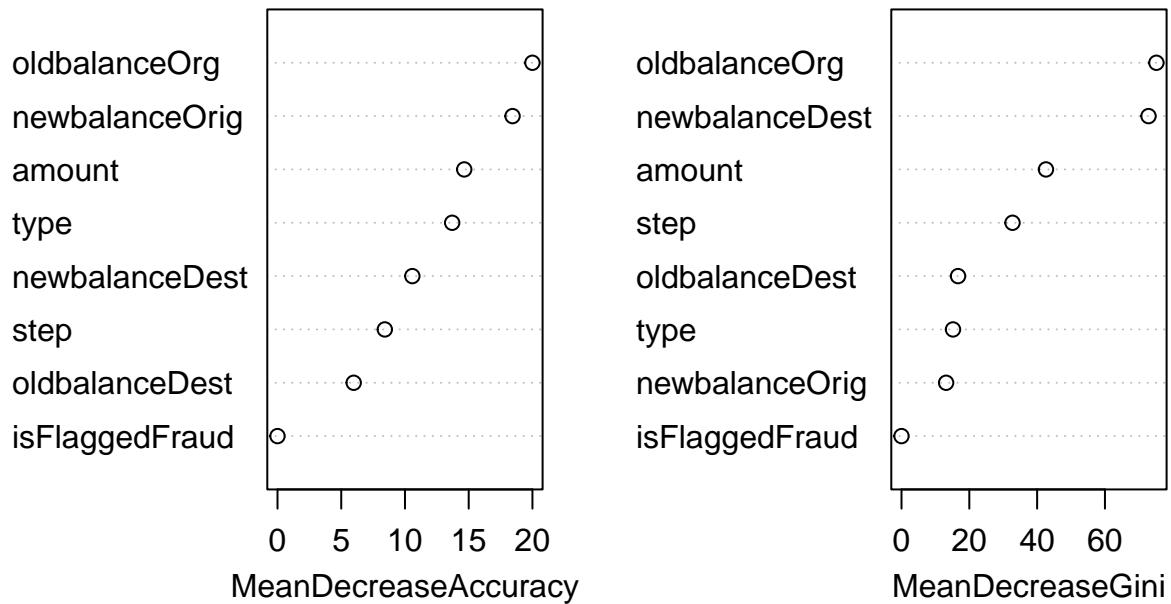
set.seed(3)

yhat_rf <- predict(rf_fraud, newdata = fraud_test, type = 'class')

varImpPlot(rf_fraud, main ='R.F with mtry = 3 and 100 trees')

```

R.F with mtry = 3 and 100 trees



Here the most important predictors are The old balance of the account initiating the transaction, (*oldbalanceOrg*) , the new balance of the destination account holder, (*newbalanceDest*), and the transaction amount (*amount*)

```

# Table
rf_table <- table('Prediction' = yhat_rf, 'True Value' = isFraud_test)
rf_table

##          True Value
## Prediction   0      1
##             0 6254444  2277
##             1      98  5801
sum(diag(rf_table))/ sum(rf_table)

## [1] 0.9996208
rf_table[2,2]/ sum(rf_table[2, ])
## [1] 0.983387

```

Now as mentioned before this data set has a large null rate. So our overall error would be small if we simply made our predictions all be zero. Instead we focus our results on the fraud detection rate. That is to

Flagged	Bagging	Boosting
0.19%	95.65%	98.34%

Table 1: Fraud Detection Rate

say, we measure how well our model is at detection fraud itself. Below are such rates for the flagged rate $isFlaggedFraud$, bagging, and boosting.