Fraud Detection

12.05.2022

Bill Kamanzi

BANA 255 - Data Literacy and Analytics

Overview

An established food retailer has introduced a self-scanning system that allows customers to scan their items using a handheld mobile scanner while shopping. This type of payment leaves retailers with the risk that a certain number of customers will take advantage of this freedom to commit fraud by not scanning all the items in their cart. Empirical research conducted by suppliers has shown that discrepancies are found in approximately 5% of all self-scan transactions. The research does not differentiate between the actual fraudulent intent of the customer, inadvertent errors, or technical problems with scanners.

Objective

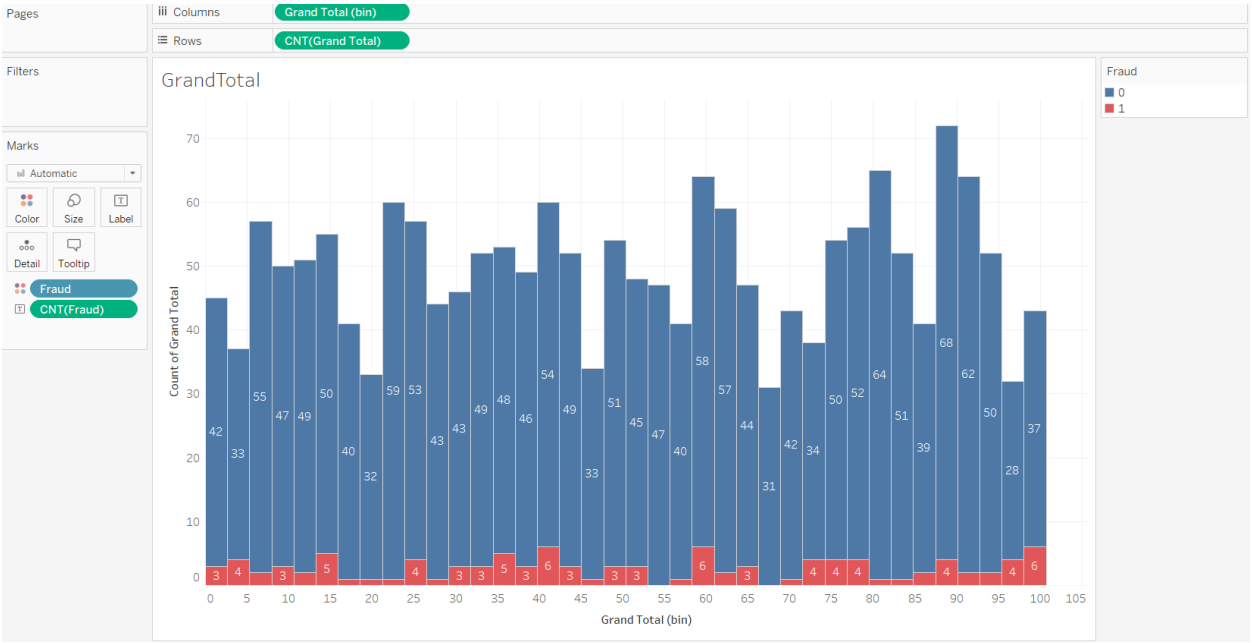
To minimize losses, the food retailer hopes to identify fraud cases using targeted follow-up checks. The challenge here is to keep the number of checks as low as possible to avoid unnecessary added expense as well as to avoid putting off innocent customers due to false accusations. At the same time, however, the goal is to identify as many false scans as possible.

Create a **predictive model** to classify the scans as fraudulent or non-fraudulent. The classification does not consider whether the fraud was committed intentionally or inadvertently.

Explanatory Data Analysis (EDA)

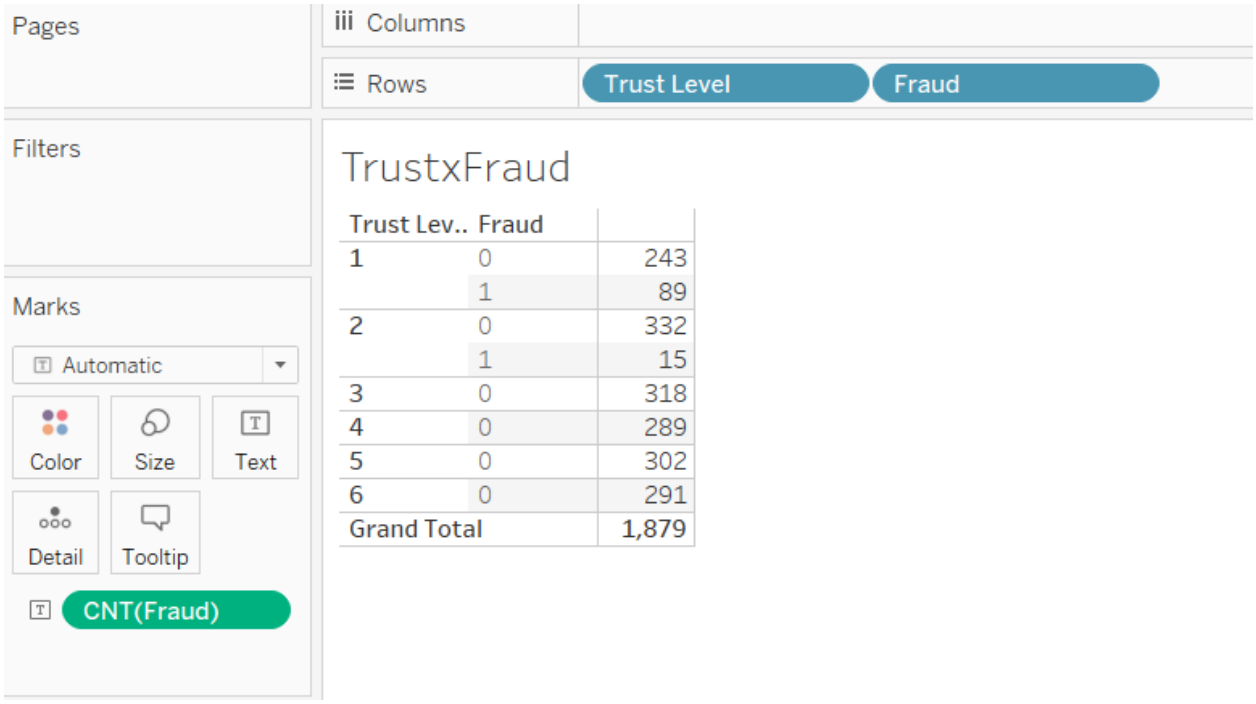
The following are the steps I did to analyze and better understand the data using Tableau:

1. The first graph I assessed was a histogram of the Grand Total of products scanned with colors that indicate how many observations were classified as fraud.



This shows that there are fewer observations that were classified as fraud. For example, under the 25 bin of Grand Total, out of 57 observations that all have 25 products in total scanned, 53 observations were classified as non-fraudulent and 4 were classified as fraudulent.

2. Assess the Relationship between Trust level and Fraud variables:



Trust Lev..	Fraud	
1	0	243
	1	89
2	0	332
	1	15
3	0	318
4	0	289
5	0	302
6	0	291
Grand Total		1,879

As the figure shows, as the trust level of the customer goes up to 6, there are no observations classified as fraudulent, and for those whose trust level is 1 and 2, some of the observations were classified as fraudulent. Thus, the customer's trust level may determine whether some of their scans are classified as fraud since there are no observations classified as fraud for customers with high trust levels.

3. Combine value per second and ScannedLineItemsPerSecond to get averageValueofScannedLineItemsPerSecond

```
#Step 0: EDA (Explanatory Data Analysis) and cleaning.

#Combine valuePerSecond and ScannedLineItemsPerSecond to get averageValueofScannedLineItemsPerSecond
df$averageValueofScannedLineItemsPerSecond = df$valuePerSecond / df$scannedLineItemsPerSecond
df$averageValueofScannedLineItemsPerSecond[1:10]
#check for missing values
colSums(is.na(df))
```

The values in Scanned Line Items Per second are quite big and thus insignificant according to our prediction model in R.

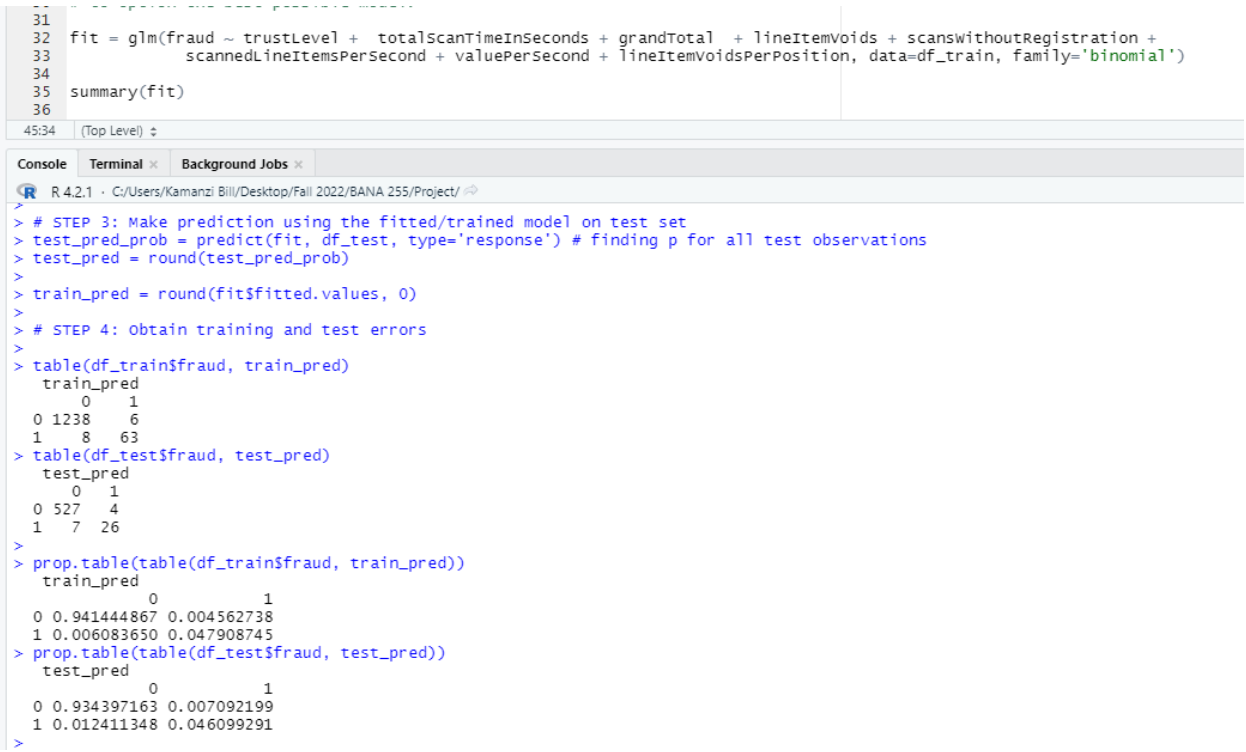
```
Coefficients:
                                          Estimate  Std. Error z value   Pr(>|z|)
(Intercept)                             -9.996190    2.460210  -4.063 0.00004842 ***
trustLevel2                             -9.352543    1.924392  -4.860 0.00000117 ***
trustLevel3                            -37.481253 4313.460957  -0.009    0.993067
trustLevel4                            -29.734345 5536.807503  -0.005    0.995715
trustLevel5                            -36.270896 4477.838306  -0.008    0.993537
trustLevel6                            -34.999010 5118.765949  -0.007    0.994545
totalScanTimeInSeconds                   0.004945    0.001503   3.289    0.001004 **
grandTotal                               0.306238    0.075469   4.058 0.00004954 ***
lineItemVoids                            2.575510    0.571501   4.507 0.00000659 ***
scansWithoutRegistration                 0.650058    0.165645   3.924 0.00008695 ***
scannedLineItemsPerSecond                6.257602    4.664037   1.342    0.179703
valuePerSecond                         -32.188514   11.918970  -2.701    0.006921 **
lineItemVoidsPerPosition               -47.480208   11.833632  -4.012 0.00006013 ***
averageValueofScannedLineItemsPerSecond  -6.006700    1.629650  -3.686    0.000228 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 552.581  on 1314  degrees of freedom
Residual deviance:  58.048  on 1301  degrees of freedom
AIC: 86.048

Number of Fisher Scoring iterations: 23
```

It has zero stars thus making it insignificant in predicting fraudulent scans. So I decided to divide the value per second for scanned items by scanned line items per second to get the average value of scanned items per second. This in turn makes the scanned line items per second more significant in our model for predicting fraudulent scans and also increases the model's accuracy as shown below:



```
31
32 fit = glm(fraud ~ trustLevel +  totalScanTimeInSeconds + grandTotal  + lineItemVoids + scansWithoutRegistration +
33              scannedLineItemsPerSecond + valuePerSecond + lineItemVoidsPerPosition, data=df_train, family='binomial')
34
35 summary(fit)
36
```

45:34 (Top Level)

Console | Terminal | Background Jobs

R 4.2.1 · C:/Users/Kamanzi Bill/Desktop/Fall 2022/BANA 255/Project/

```
> # STEP 3: Make prediction using the fitted/trained model on test set
> test_pred_prob = predict(fit, df_test, type='response') # finding p for all test observations
> test_pred = round(test_pred_prob)
>
> train_pred = round(fit$fitted.values, 0)
>
> # STEP 4: Obtain training and test errors
>
> table(df_train$fraud, train_pred)
   train_pred
      0    1
  0 1238    6
  1    8   63
> table(df_test$fraud, test_pred)
   test_pred
     0   1
  0 527   4
  1   7  26
>
> prop.table(table(df_train$fraud, train_pred))
   train_pred
             0           1
  0 0.941444867 0.004562738
  1 0.006083650 0.047908745
> prop.table(table(df_test$fraud, test_pred))
   test_pred
             0           1
  0 0.934397163 0.007092199
  1 0.012411348 0.046099291
>
```

This is the model excluding the average value of scanned items per second value in the model prediction. The accuracy is about 98.0% and using the cost matrix, it has a monetary value of -$5.

As we add the new variable to the prediction model:

```
fit = glm(fraud ~ trustLevel +  totalScanTimeInSeconds + grandTotal  + lineItemVoids + scansWithoutRegistration +
            scannedLineItemsPerSecond + valuePerSecond + lineItemVoidsPerPosition + averageValueofScannedLineItemsPerSecond,
            data=df_train, family='binomial')

summary(fit)

# STEP 3: Make prediction using the fitted/trained model on test set
test_pred_prob = predict(fit, df_test, type='response') # finding p for all test observations
test_pred = round(test_pred_prob)

train_pred = round(fit$fitted.values, 0)

# STEP 4: Obtain training and test errors

table(df_train$fraud, train_pred)
table(df_test$fraud, test_pred)
```

34:13 (Top Level)

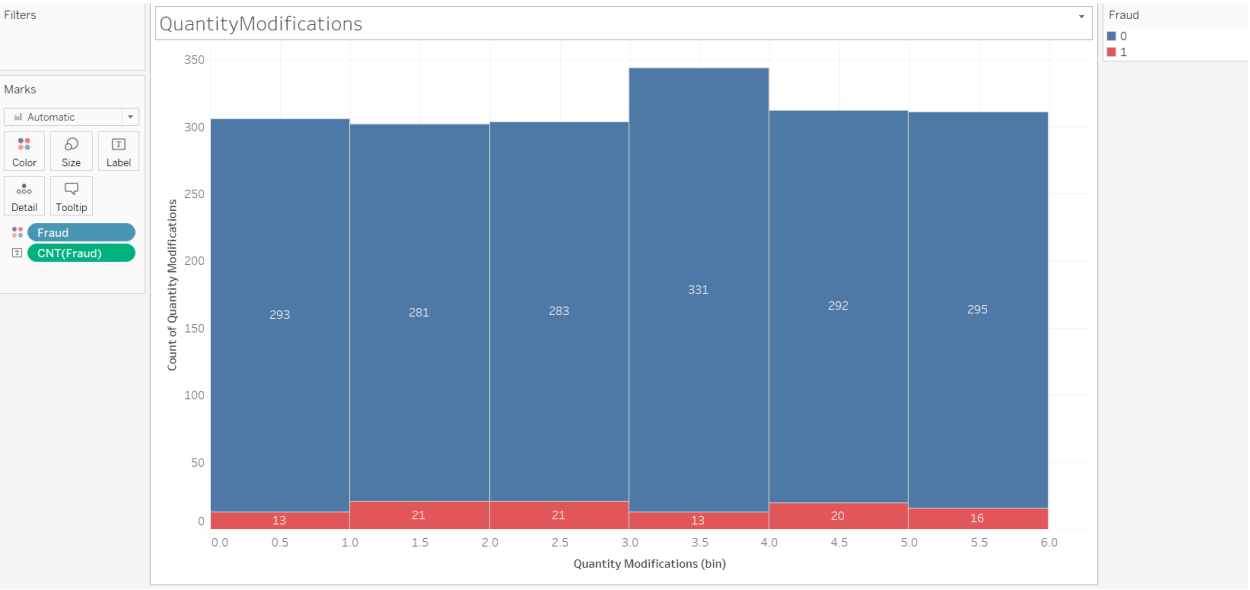
Console | Terminal | Background Jobs

R 4.2.1 · C:/Users/Kamanzi Bill/Desktop/Fall 2022/BANA 255/Project/

```
> # STEP 3: Make prediction using the fitted/trained model on test set
> test_pred_prob = predict(fit, df_test, type='response') # finding p for all test observations
> test_pred = round(test_pred_prob)
>
> train_pred = round(fit$fitted.values, 0)
>
> # STEP 4: Obtain training and test errors
>
> table(df_train$fraud, train_pred)
   train_pred
       0    1
  0 1237    7
  1    8   63
> table(df_test$fraud, test_pred)
   test_pred
      0   1
  0 529   2
  1   4  29
>
> prop.table(table(df_train$fraud, train_pred))
   train_pred
              0           1
  0 0.940684411 0.005323194
  1 0.006083650 0.047908745
> prop.table(table(df_test$fraud, test_pred))
   test_pred
              0           1
  0 0.937943262 0.003546099
  1 0.007092199 0.051418440
>
```

The accuracy of the model increases by a small percentage and also increases the monetary value of the cost matrix, which will be discussed below. Therefore, we have established that the relationship between value per second and scanned line item per second increased the model's accuracy.

4. Assess the relationship between quantity modification and fraud:



Compared to the number of observations for modifications classified as non-fraudulent, the number of observations for quantity modifications classified as fraudulent is very low. Thus, the probability of predicting a fraudulent scan is low judging by the number of quantity of modifications. I believe that this is what makes the quantity modifications variable insignificant in our prediction model.

Modeling

Step 1: Divide the data into training and test sets and Treat Fraud and TrustLevel as categorical Values

```
#Treat Fraud and TrustLevel as categorical Values
df$trustLevel = as.factor(df$trustLevel)
df$fraud = as.factor(df$fraud)


# STEP 1: Divide the data into training and test sets
set.seed(42) # we are setting the seed for the sake of reproducibility.
n = dim(df)[1] #Number of Observations
train = sample(1:n, 0.7*n, replace=FALSE) #The indices of customers that will go to training set
test = setdiff(1:n, train) #The indices of customers that will go to the test set

df_train = df[train, ] # create the training dataset
df_test = df[test, ] # create the test dataset
```

Step 2: Build the model on the training set only. Also, do all the modifications to option the best possible model.

```
# STEP 2: Build the model on training set only. Also, do all the modification
# to option the best possible model.

fit = glm(fraud ~ trustLevel +  totalScanTimeInSeconds + grandTotal  + lineItemVoids + scansWithoutRegistration +
            scannedLineItemsPerSecond + valuePerSecond + lineItemVoidsPerPosition + averageValueofScannedLineItemsPerSecond, data=df_train, family='binomial')

summary(fit)
```

```
Call:
glm(formula = fraud ~ trustLevel + totalScanTimeInSeconds + grandTotal +
    lineItemVoids + scansWithoutRegistration + scannedLineItemsPerSecond +
    valuePerSecond + lineItemVoidsPerPosition + averageValueofScannedLineItemsPerSecond,
    family = "binomial", data = df_train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.093   0.000   0.000   0.000   2.306

Coefficients:
                                        Estimate  Std. Error z value   Pr(>|z|)
(Intercept)                            -9.996190    2.460210  -4.063 0.00004842 ***
trustLevel2                            -9.352543    1.924392  -4.860 0.00000117 ***
trustLevel3                           -37.481253 4313.460957  -0.009   0.993067
trustLevel4                           -29.734345 5536.807503  -0.005   0.995715
trustLevel5                           -36.270896 4477.838306  -0.008   0.993537
trustLevel6                           -34.999010 5118.765949  -0.007   0.994545
totalScanTimeInSeconds                  0.004945    0.001503   3.289   0.001004 **
grandTotal                              0.306238    0.075469   4.058 0.00004954 ***
lineItemVoids                           2.575510    0.571501   4.507 0.00000659 ***
scansWithoutRegistration                0.650058    0.165645   3.924 0.00008695 ***
scannedLineItemsPerSecond               6.257602    4.664037   1.342   0.179703
valuePerSecond                        -32.188514   11.918970  -2.701   0.006921 **
lineItemVoidsPerPosition              -47.480208   11.833632  -4.012 0.00006013 ***
averageValueofScannedLineItemsPerSecond  -6.006700    1.629650  -3.686   0.000228 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 552.581  on 1314  degrees of freedom
Residual deviance:  58.048  on 1301  degrees of freedom
AIC: 86.048

Number of Fisher Scoring iterations: 23
```

Step 3: Make predictions using the fitted/trained model on the test set

```
# STEP 3: Make prediction using the fitted/trained model on test set
test_pred_prob = predict(fit, df_test, type='response') # finding p for all test observations
test_pred = round(test_pred_prob)

train_pred = round(fit$fitted.values, 0)
```

Step 4: Obtain training and test errors

```
# STEP 4: Obtain training and test errors

table(df_train$fraud, train_pred)
table(df_test$fraud, test_pred)

prop.table(table(df_train$fraud, train_pred))
prop.table(table(df_test$fraud, test_pred))
```

Evaluation

1) Monetary value using cost matrix
    a) The food retailer receives a profit of $5 for every correctly identified fraud attempt. However, for every fraud case that goes unexposed, he loses $5. A customer falsely accused of fraud, might not return to this store, which is denoted by a $25 loss for the retailer.

    Using our model:

```
> # STEP 4: Obtain training and test errors
>
> table(df_train$fraud, train_pred)
   train_pred
       0    1
  0 1237    7
  1    8   63
> table(df_test$fraud, test_pred)
   test_pred
      0   1
  0 529   2
  1   4  29
>
```

    Since the food retailer receives a profit of $5, our predicted 29 correct fraud scans on the test data, thus making a total of (29*5) = $145.

    The food retailer loses $25 for every customer that is falsely accused of fraud, and according to our model, it predicted only 2 incorrect fraud scans thus making a loss of $50.

    Lastly, for every fraud that goes undetected, they lose $5, and our model did predict 4 false scans, thus making a loss of $20.

    Total Profit = 145 - (50+20) = $75 dollars

    In conclusion, using our model the food retailer will gain $75 in profit.

2) Accuracy (From the confusion matrix)
    a) Using the model:

```
> prop.table(table(df_train$fraud, train_pred))
   train_pred
                0           1
  0 0.940684411 0.005323194
  1 0.006083650 0.047908745
> prop.table(table(df_test$fraud, test_pred))
   test_pred
                0           1
  0 0.937943262 0.003546099
  1 0.007092199 0.051418440
```

The accuracy of the model is: 0.937943262 + 0.051418440 = 0.989361702 ~ 98.9%. The accuracy is very high, almost 99% accurate. The model looks stable as the two confusion matrices look similar.

The True positive rate (26/ (26+4) = 86.6%, therefore, this model is accurate.

Conclusion

I recommend using a logistic regression model to predict fraud. The model will utilize the confusion matrix features to assess the model's stability and accuracy. This model has a monetary value of $75 and an accuracy of %98.9.