Whether a given credit card transaction will be fraudulent or not?

Marjan Rezvani

Fall 2020

course: Econ B2000, Statistics and Introduction to Econometrics

Abstract

One of the most important responsibilities that a bank or financial institution has is to protect the integrity of the institution by working hard to protect the financial assets that it holds. Bank fraud can be defined as an unethical and/or criminal act by an individual or organization to illegally attempt to possess or receive money from a bank or financial institution.It is anticipated that card frauds would amount to around $30 billion worldwide by 2020. So, how banks can improve security by detecting and obstructing frauds?

In this project, I have applied multiple ML techniques to the problem using card transaction data to identify fraudulent transactions (i.e. Fraud Detection)

I tried to show that my proposed approaches are able to detect fraud transactions with high accuracy and reasonably low number of false positives.

Introduction

Banking Fraud has been an ever-growing issue with huge consequences to banks and customers, in terms of financial losses, trust and credibility. An effective fraud detection system should be able to detect fraudulent transactions with high accuracy and efficiency.

A major challenge in applying a model to fraud detection is presence of highly imbalanced data sets. In many available datasets, majority of transactions are genuine with an extremely small percentage of fraudulent ones. Designing an accurate and efficient fraud detection system that is low on false positives but detects fraudulent activity effectively is a significant challenge.

In this project, I have applied multiple binary classification approaches such as Logistic Regression, Random Forest and Support Vector Machine to solve this problem on a labeled dataset. my models collect information, analyze the data gathered and extract the required features. my goal is to build binary classifiers which are able to separate fraud transactions from non-fraud transactions. Then I compared the effectiveness of these approaches in detecting fraud transactions.

Related Work

There are different types of approaches which have been applied to the problem of fraud detection which gave me some ideas to do my project. Different computational methods

have been stated for detecting the fraud by computing various parameters for each kind of algorithm. and They had taken the different datasets german credit card dataset.

For instance, in paper https://ijarcce.com/upload/2017/june-17/IJARCCE%202.pdf there have been used some of the supervised algorithms to detect the fraud which gives accurate results. first They have explained different types of frauds which are: credit card fraud, financial fraud, mortgage fraud, insurance fraud , telecommunication fraud. Then they used some supervised learning algorithms predictions which are made on the known training dataset. Their learning algorithms are further grouped into regression and classification problems. so they collected the accurate training dataset and then have found the accuracy of the function. It is the machine learning task of inferring a function from supervised training data. in continue, they created small decision trees so that records can be identified after only a few decision tree splitting. then tried to match a hoped for minimalism of the process of decision making.

In the other paper, https://arxiv.org/pdf/1611.06439.pdf, some people investigated difficulties of credit card fraud detection. Then the advantages and disadvantages of fraud detection methods have been enumerated and compared.Furthermore, they have presented a classification of some techniques into two main fraud detection approaches, namely, misuses (supervised) and anomaly detection (unsupervised).

In addition, a group of students in their project http://cs229.stanford.edu/proj2018/report/261.pdf have reviewed and compared such multiple state of the techniques, datasets and evaluation criteria applied to this problem. they have applied multiple binary classification approaches - Logistic regression, Linear SVM and SVM with RBF kernel on a labeled dataset.

Dataset and Explanatory Analysis

In this project I have used a dataset downloaded from capital one data science challenge repository https://github.com/CapitalOneRecruiting/DS which contains a year credit card transaction made in 2016. There are 641914 instances in the dataset. As it's shown below:

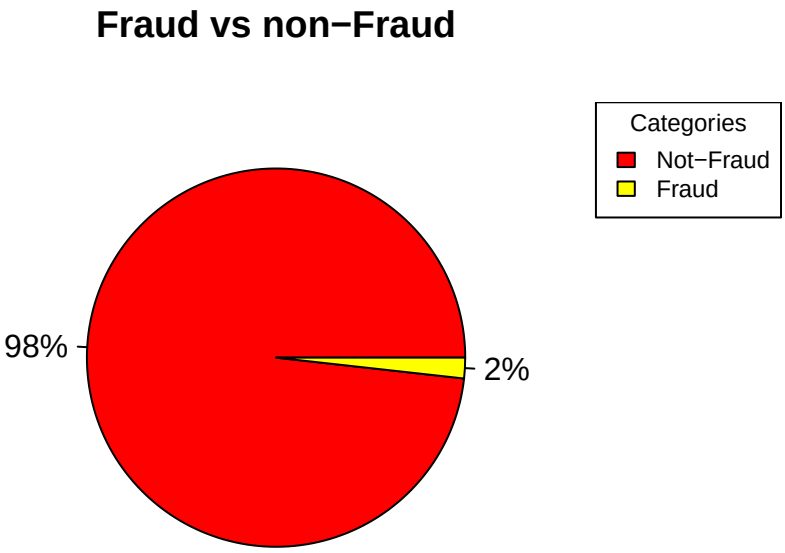
```
raw_data <- read.csv(file='/Users/marjanrezvani/Documents/Fall2020/eco_stat/final_project/data/rawdata.
colnames(raw_data)[order(colnames(raw_data))]
```

```
##  [1] "accountNumber"            "accountOpenDate"
##  [3] "acqCountry"               "availableMoney"
##  [5] "cardCVV"                  "cardLast4Digits"
##  [7] "cardPresent"              "creditLimit"
##  [9] "currentBalance"           "currentExpDate"
## [11] "customerId"               "dateOfLastAddressChange"
## [13] "echoBuffer"               "enteredCVV"
## [15] "expirationDateKeyInMatch" "isFraud"
## [17] "merchantCategoryCode"     "merchantCity"
## [19] "merchantCountryCode"      "merchantName"
## [21] "merchantState"            "merchantZip"
## [23] "posConditionCode"         "posEntryMode"
## [25] "posOnPremises"            "recurringAuthInd"
## [27] "transactionAmount"        "transactionDateTime"
## [29] "transactionType"          "X"
```

This dataset is highly unbalance with a low percentage of fraudulent transactions within several records of non-fraud transactions, which is shown in my pie chart. Out of the 641914 transactions in the dataset, 10892 were fraudulent which means the True frauds account for 2% of all transactions. The feature named 'isFraud' is used to classify the transaction whether it is a fraud or not. There are also features besides being fraud and not fraud that I am going to describe. Some of numerical attributes are like available money, credit limit and categorical attributes like merchant name and transaction type. there are also few attributes such as echoBuffer, merchantCity, merchantState, merchantZip, posOnPremises, recurringAuthInd, which totally have missing values that I will not use these columns.

```
pie(table(raw_data$isFraud),
    labels = paste(round(prop.table(table(raw_data$isFraud))*100), "%", sep = ""),
    col = heat.colors(2), main = "Fraud vs non-Fraud")

legend("topright", legend = c("Not-Fraud", "Fraud"),
       fill = heat.colors(2), title = "Categories", cex = 0.75)
```



Classification results for the unbalanced data could be misleading. So, I first make a balanced dataset. There are various ways to make data balanced. such as downsample majority class or upsample minority class. In this part, I downsample majority class.

```
dim(raw_data[raw_data$isFraud=='True',])[1] # Fraud
```

```
## [1] 11302
```

```
dim(raw_data[raw_data$isFraud=='False',])[1] # non Fraud
```

```
## [1] 630612
```

```
df1 = raw_data[raw_data$isFraud=='True',]
tmp = raw_data[raw_data$isFraud=='False',]
df2 = tmp[sample(nrow(raw_data[raw_data$isFraud=='False',]),15000),]

balanced_df = rbind(df1,df2)
```

then I created a new variable named 'correctCVV'. Its value is True if cardCVV matches enteredCVV

```
balanced_df$correctCVV <- with(balanced_df, ifelse(cardCVV==enteredCVV, 'True', 'False'))

summary(balanced_df)
```

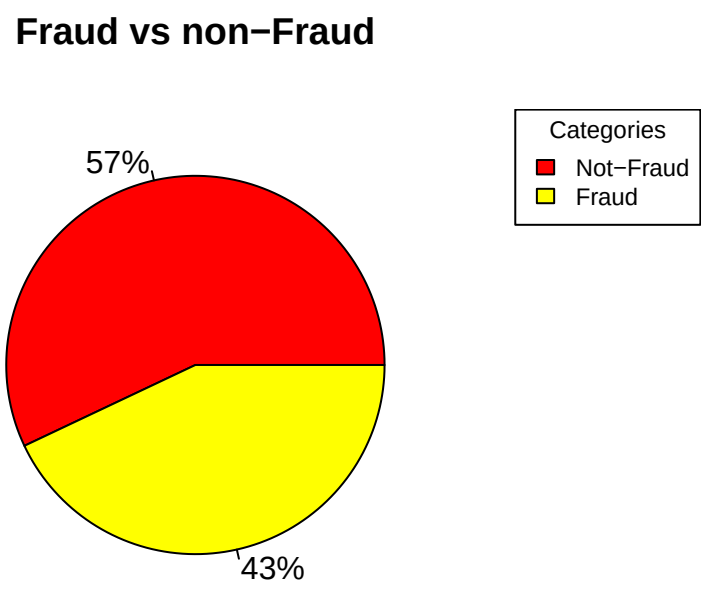
```
##        X          accountNumber         accountOpenDate  acqCountry
##  Min.   :     0   Min.   :100547107   2015-12-11:  518      :  226
##  1st Qu.:161600   1st Qu.:318001076   2014-06-06:  384   CAN:   99
##  Median :315332   Median :537052791   2013-07-04:  301   MEX:  116
##  Mean   :319392   Mean   :549265331   2012-10-05:  279   PR :   54
##  3rd Qu.:481207   3rd Qu.:782744512   2014-01-31:  225   US :25807
##  Max.   :641874   Max.   :999789077   2015-03-12:  208
##                                       (Other)   :24387
##  availableMoney       cardCVV      cardLast4Digits cardPresent
##  Min.   : -894.6   Min.   :100.0   Min.   :   0    False:14423
##  1st Qu.: 1196.3   1st Qu.:327.0   1st Qu.:2315    True :11879
##  Median : 3648.8   Median :572.0   Median :4743
##  Mean   : 6620.8   Mean   :552.3   Mean   :4823
##  3rd Qu.: 8108.5   3rd Qu.:759.0   3rd Qu.:7262
##  Max.   :50000.0   Max.   :998.0   Max.   :9995
##
##   creditLimit    currentBalance    currentExpDate    customerId
##  Min.   :  250   Min.   :    0.0   12/2021:  204   Min.   :100547107
##  1st Qu.: 5000   1st Qu.:  587.2   10/2024:  198   1st Qu.:318001076
##  Median : 7500   Median : 2276.4   08/2026:  197   Median :537052791
##  Mean   :10786   Mean   : 4165.3   11/2020:  196   Mean   :549265331
##  3rd Qu.:15000   3rd Qu.: 5257.9   10/2031:  193   3rd Qu.:782744512
##  Max.   :50000   Max.   :47469.7   05/2025:  186   Max.   :999789077
##                                    (Other):25128
##  dateOfLastAddressChange echoBuffer       enteredCVV
##  2016-01-29:  304        Mode:logical   Min.   :  4.0
##  2016-05-11:  167        NA's:26302     1st Qu.:327.0
##  2016-03-15:  147                       Median :571.0
##  2016-04-25:  145                       Mean   :551.8
##  2016-01-26:  138                       3rd Qu.:759.0
##  2016-03-10:  124                       Max.   :998.0
##  (Other)   :25277
##  expirationDateKeyInMatch  isFraud         merchantCategoryCode merchantCity
##  False:26268               False:15000   online_retail:7320      Mode:logical
##  True :   34               True :11302   fastfood     :4186      NA's:26302
##                                          food         :2996
##                                          entertainment:2515
##                                          rideshare    :2279
##                                          online_gifts :1433
##                                          (Other)      :5573
##  merchantCountryCode      merchantName   merchantState  merchantZip
##     :   94           Uber       : 1144   Mode:logical   Mode:logical
##  CAN:  102           Lyft       : 1135   NA's:26302     NA's:26302
##  MEX:  115           alibaba.com:  721
##  PR :   53           target.com :  667
##  US :25938           sears.com  :  638
##                      oldnavy.com:  620
##                      (Other)    :21377
##  posConditionCode  posEntryMode    posOnPremises  recurringAuthInd
##  Min.   : 1.000   Min.   : 2.000   Mode:logical   Mode:logical
```

```
##  1st Qu.: 1.000   1st Qu.: 2.000   NA's:26302     NA's:26302
##  Median : 1.000   Median : 5.000
##  Mean   : 3.525   Mean   : 9.456
##  3rd Qu.: 1.000   3rd Qu.: 9.000
##  Max.   :99.000   Max.   :90.000
##  NA's   :17       NA's   :316
##  transactionAmount          transactionDateTime             transactionType
##  Min.   :   0.00   2016-04-29T13:56:53:    2                          :   26
##  1st Qu.:  49.11   2016-06-24T11:59:15:    2    ADDRESS_VERIFICATION:  453
##  Median : 124.51   2016-07-18T07:16:59:    2    PURCHASE             :25148
##  Mean   : 176.18   2016-08-05T07:13:58:    2    REVERSAL             :  675
##  3rd Qu.: 250.55   2016-10-07T12:09:39:    2
##  Max.   :1743.51   2016-11-13T03:13:58:    2
##                    (Other)            :26290
##   correctCVV
##  Length:26302
##  Class :character
##  Mode  :character
##
##
##
##
```

```
attach(balanced_df)

pie(table(isFraud), labels = paste(round(prop.table(table(isFraud))*100), "%", sep = ""),
    col = heat.colors(2), main = "Fraud vs non-Fraud")

legend("topright", legend = c("Not-Fraud", "Fraud"),
       fill = heat.colors(2), title = "Categories", cex = 0.75)
```



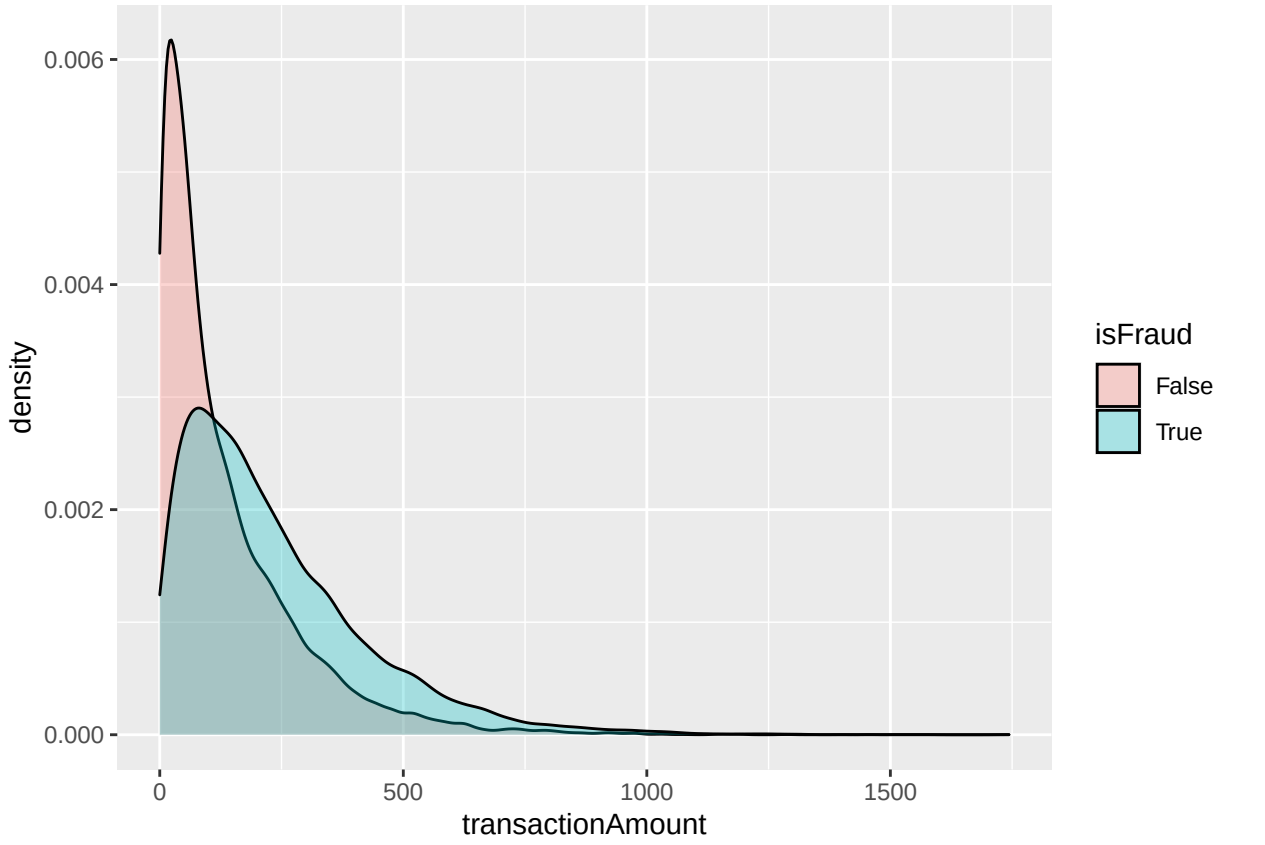
So now from the pie chart we observe that we have kind of balanced dataset.

Let's do some plots to examine the dataset.

First thing I am interested in is to explore and see if there is a difference between the distribution of transactionamount across Fraud or non-Fraud.

Plot below shows that Fradualant transavctions have higher transaction amount (e.g. compare the area under the curve for range of $500 to $1000)

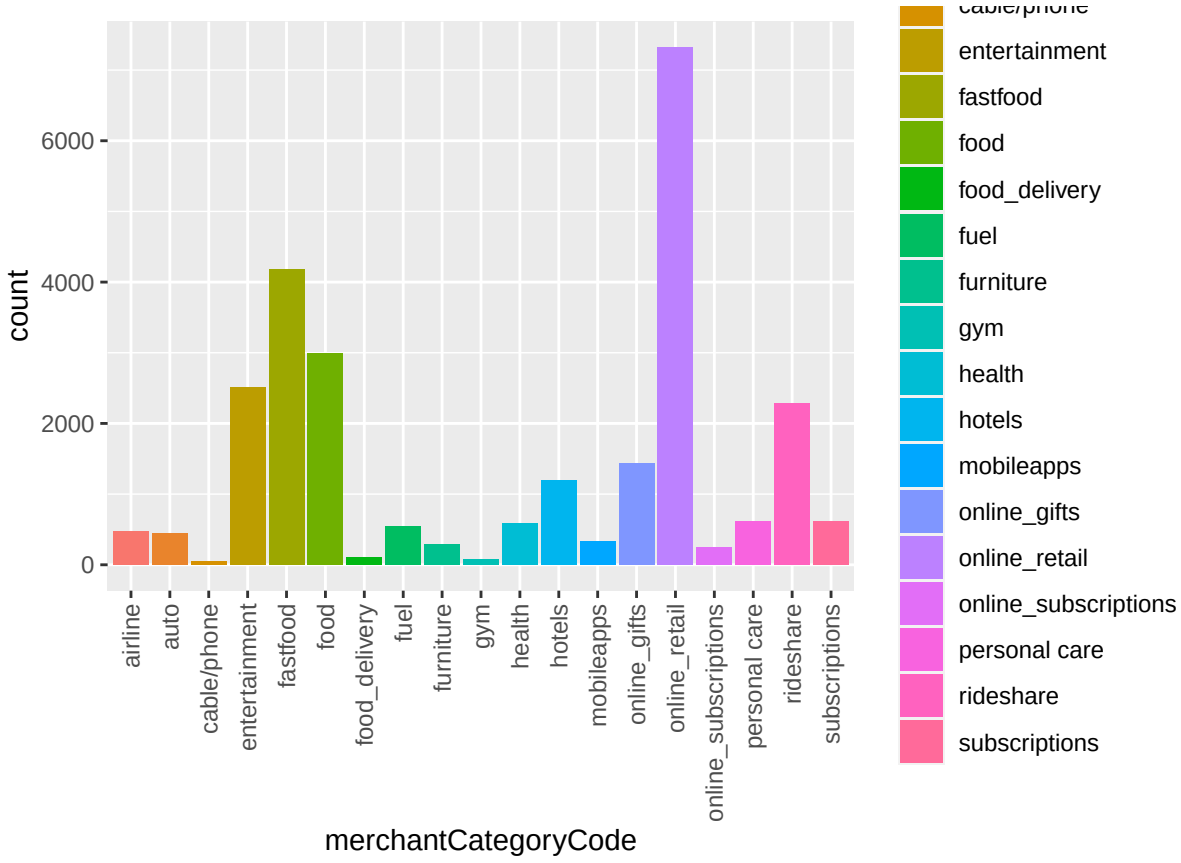
```
library(ggplot2)
ggplot(data=balanced_df, aes(x = transactionAmount, fill = isFraud)) +
  geom_density(alpha = .3) #alpha used for filling the density
```



Let's see if how the transactions are distributed across different merchants.

Plot below shows that majority of the transactions are from online_retail. fastfood, and food, and entertainment are the other 3 merchants with high number of transactions.

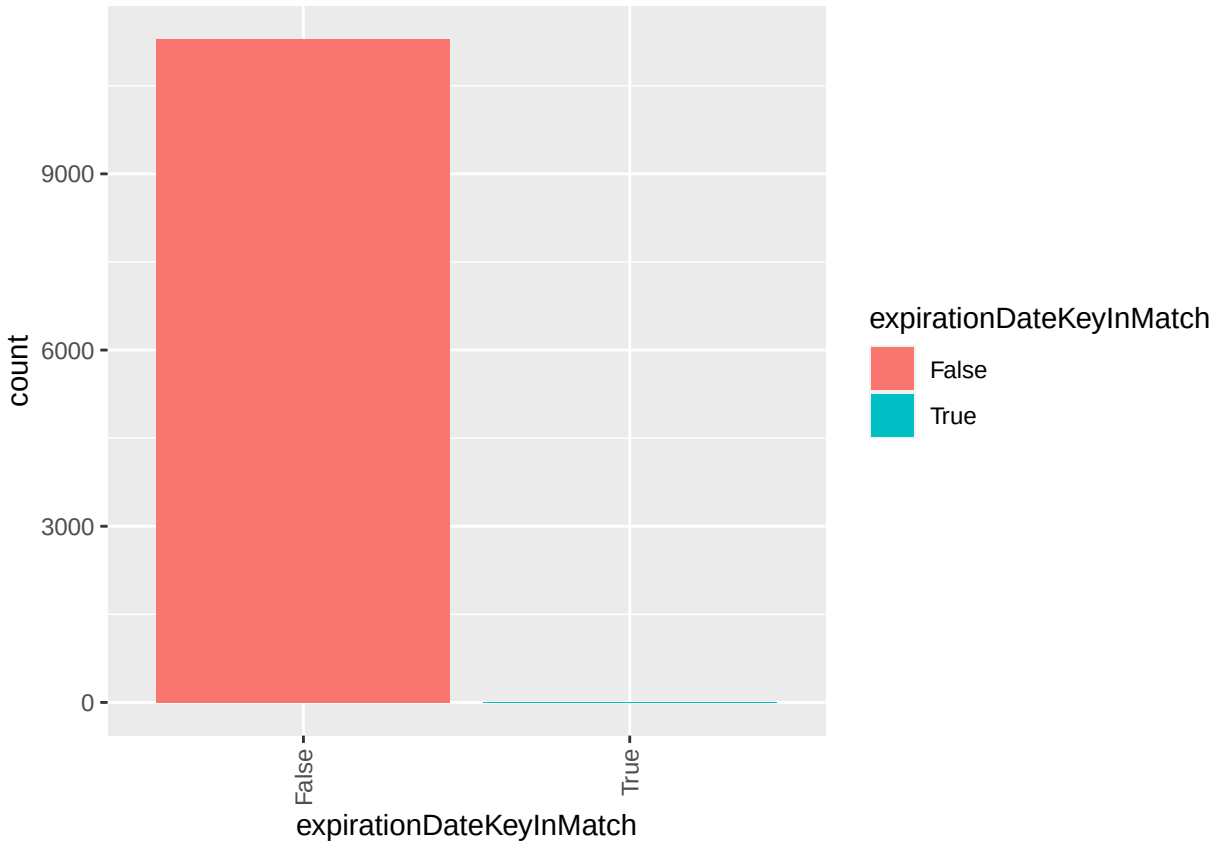
```
(ggplot(balanced_df, aes(x=merchantCategoryCode, fill=merchantCategoryCode))
  + geom_bar()
  + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
)
```



Another interesting feature is 'expirationDateKeyInMatch' which shows whether the entered expiration date matched the one on the card.

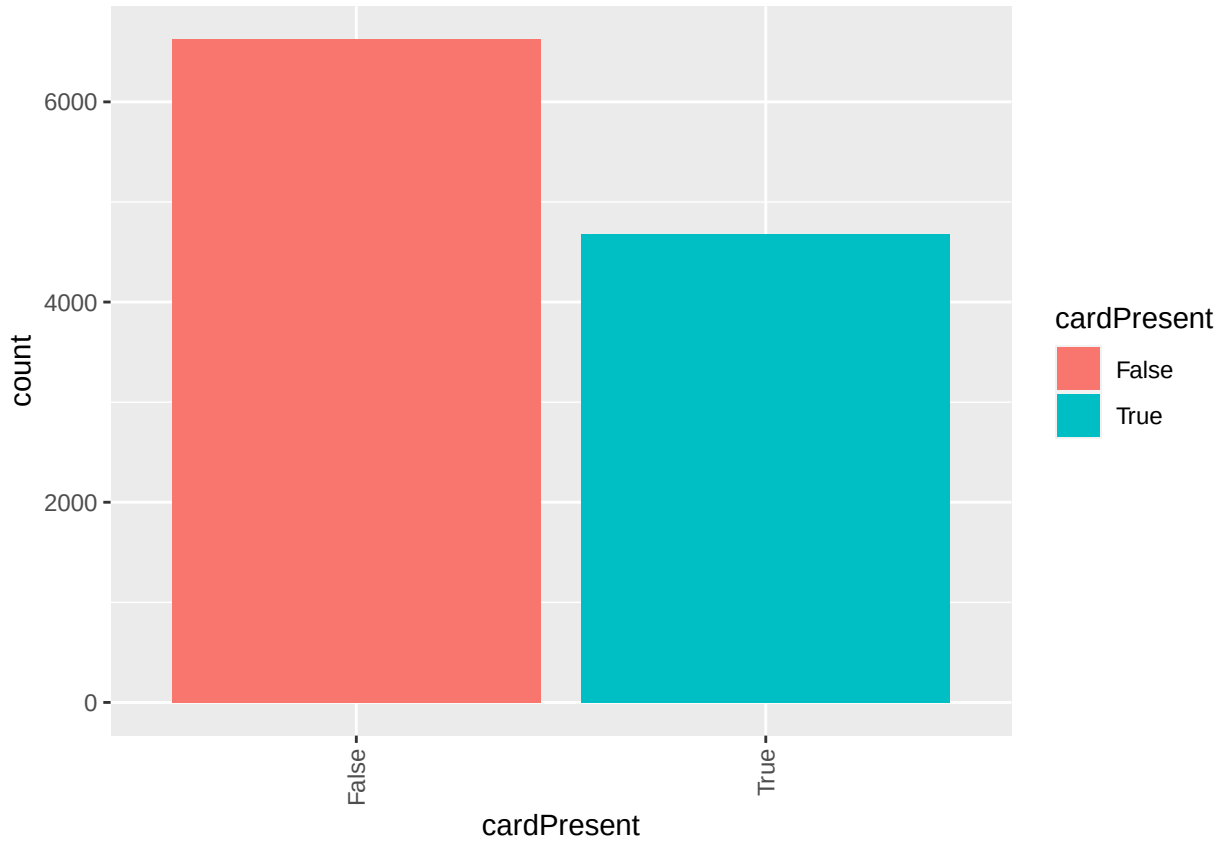
Plot below shows that for fradulent transactions, majority of them expiration date does not match

```
(ggplot(balanced_df[balanced_df$isFraud=='True',], aes(x=expirationDateKeyInMatch, fill=expirationDateK
  + geom_bar()
  + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
)
```



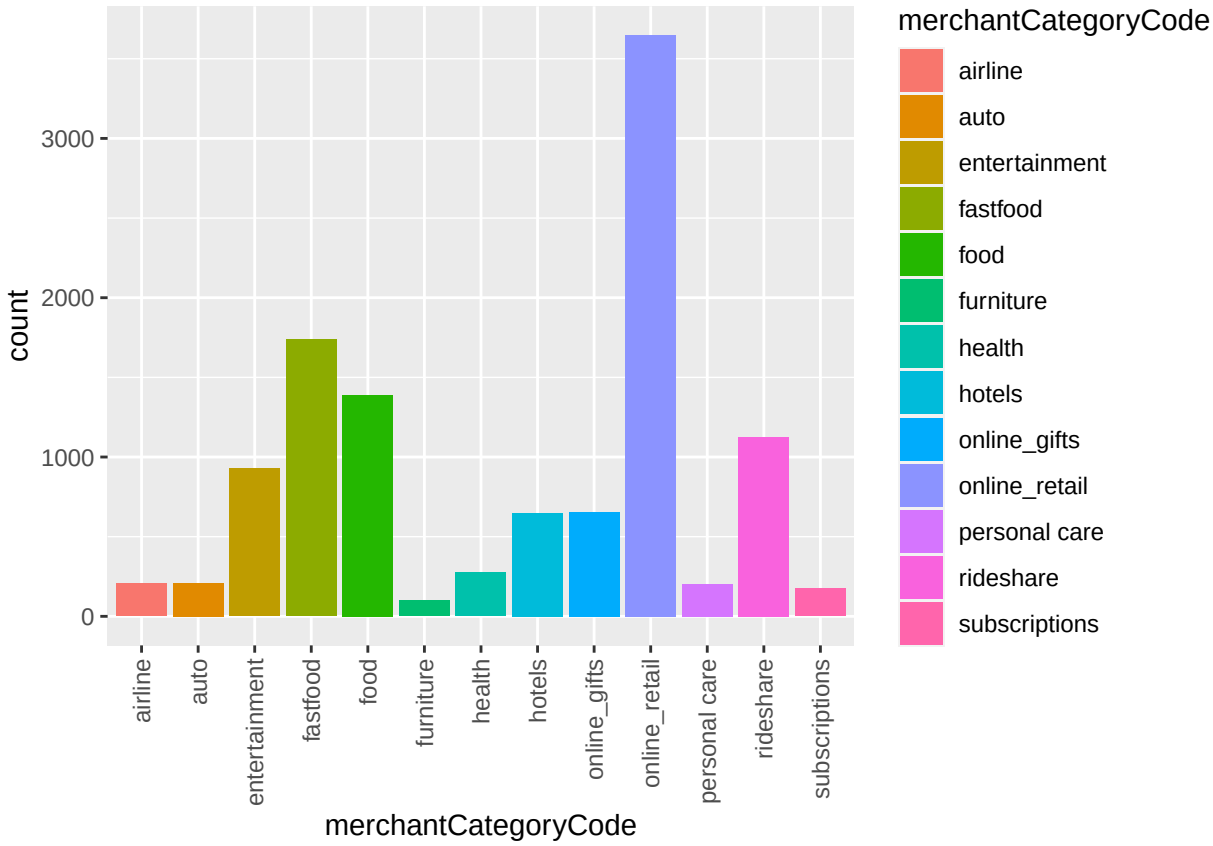
Similarly, we can look at cardPresent featute. It tells whether card was present during transaction or not. Plot below indicates that for Fradulent transactions, most of the transactioins card was not present (in other words most of them was online. Actually, we can confirm this by looking at distribution of Fradulent transactions across different merchants)

```
(ggplot(balanced_df[balanced_df$isFraud=='True',], aes(x=cardPresent, fill=cardPresent))
 + geom_bar()
 + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
)
```



And Finally plot below shows that, we have most of the Fradulent transactions heppening in online_retails.

```
(ggplot(balanced_df[balanced_df$isFraud=='True',], aes(x=merchantCategoryCode, fill=merchantCategoryCod
  + geom_bar()
  + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
)
```



Methods

my goal is to detect fraud and non-fraud transactions by building appropriate binary classifiers. I have built some models using logistic regression, Random Forest and Support Vector machine. then I compare the effectiveness of these approaches in detecting fraud transactions.

With these models, my goal is to (1) learn a model that, given the features, can predict which transactions have this label, (2) evaluate the model and estimate its generalizable accuracy metric, and (3) identify the features most important for prediction.

then to be able to test the performance of my algorithms, I split data into train test split by the ration of 0.25 and feed into model. Then I discuss results obtained in training, validation and testing phases. and evaluate performance of my models by computing accuracy metrics.

```
balanced_df$merchantCategoryCode <- as.factor(balanced_df$merchantCategoryCode)
balanced_df$transactionType <- as.factor(balanced_df$transactionType)
balanced_df$correctCVV <- as.factor(balanced_df$correctCVV)
balanced_df$cardPresent <- as.factor(balanced_df$cardPresent)

## 75% of the sample size
smp_size <- floor(0.75 * nrow(balanced_df))

## set the seed to make the partition reproducible
set.seed(123)
```

```
train_ind <- sample(seq_len(nrow(balanced_df)), size = smp_size)

train <- balanced_df[train_ind, ]
test <- balanced_df[-train_ind, ]

vartokeep <- c('isFraud','transactionAmount','merchantCategoryCode','transactionType',
               'correctCVV', 'cardPresent', 'currentBalance', 'creditLimit')
train <- train[,vartokeep]
test <- test[,vartokeep]
```

Logistic Regression

I fit my logistic regression model to my data and get the following classification report and confusion matrix. The number of correct and incorrect predictions are summarized with count values and broken down by each class 'Fraud' and 'Not Fraud'.

```
model_logit <- glm(isFraud ~ transactionAmount + merchantCategoryCode + transactionType+
                     correctCVV + cardPresent + currentBalance + creditLimit,
                   family = binomial, data = train)

summary(model_logit)
```

```
##
## Call:
## glm(formula = isFraud ~ transactionAmount + merchantCategoryCode +
##     transactionType + correctCVV + cardPresent + currentBalance +
##     creditLimit, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6681  -1.0026  -0.6817   1.1624   2.1480
##
## Coefficients:
##                                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)                            -7.056e-01  4.937e-01  -1.429 0.152997
## transactionAmount                       3.316e-03  1.017e-04  32.612  < 2e-16
## merchantCategoryCodeauto                8.213e-01  1.704e-01   4.818 1.45e-06
## merchantCategoryCodecable/phone        -1.599e+01  4.134e+02  -0.039 0.969151
## merchantCategoryCodeentertainment       2.768e-01  1.374e-01   2.014 0.044009
## merchantCategoryCodefastfood            5.622e-01  1.364e-01   4.121 3.78e-05
## merchantCategoryCodefood                7.912e-01  1.386e-01   5.709 1.14e-08
## merchantCategoryCodefood_delivery      -1.590e+01  2.887e+02  -0.055 0.956080
## merchantCategoryCodefuel               -1.520e+01  1.188e+02  -0.128 0.898208
## merchantCategoryCodefurniture           5.437e-02  1.997e-01   0.272 0.785478
## merchantCategoryCodegym                -1.578e+01  3.239e+02  -0.049 0.961132
## merchantCategoryCodehealth              2.259e-01  1.512e-01   1.494 0.135180
## merchantCategoryCodehotels              9.513e-01  1.423e-01   6.686 2.29e-11
## merchantCategoryCodemobileapps         -1.574e+01  1.496e+02  -0.105 0.916198
## merchantCategoryCodeonline_gifts        1.225e-01  1.290e-01   0.949 0.342529
## merchantCategoryCodeonline_retail       1.988e-01  1.166e-01   1.705 0.088115
## merchantCategoryCodeonline_subscriptions -1.578e+01  1.783e+02  -0.089 0.929473
## merchantCategoryCodepersonal care       3.063e-01  1.674e-01   1.830 0.067300
## merchantCategoryCoderideshare           2.276e-01  1.237e-01   1.840 0.065825
```

```
## merchantCategoryCodesubscriptions        -6.837e-01  1.596e-01  -4.285 1.83e-05
## transactionTypeADDRESS_VERIFICATION      -8.331e-01  4.905e-01  -1.698 0.089438
## transactionTypePURCHASE                   3.282e-01  4.674e-01   0.702 0.482574
## transactionTypeREVERSAL                   2.076e-01  4.765e-01   0.436 0.662977
## correctCVVTrue                           -4.729e-01  1.292e-01  -3.659 0.000253
## cardPresentTrue                          -6.883e-01  7.554e-02  -9.112  < 2e-16
## currentBalance                            5.568e-06  3.267e-06   1.705 0.088277
## creditLimit                              -5.271e-07  1.758e-06  -0.300 0.764343
##
## (Intercept)
## transactionAmount                        ***
## merchantCategoryCodeauto                 ***
## merchantCategoryCodecable/phone
## merchantCategoryCodeentertainment        *
## merchantCategoryCodefastfood             ***
## merchantCategoryCodefood                 ***
## merchantCategoryCodefood_delivery
## merchantCategoryCodefuel
## merchantCategoryCodefurniture
## merchantCategoryCodegym
## merchantCategoryCodehealth
## merchantCategoryCodehotels               ***
## merchantCategoryCodemobileapps
## merchantCategoryCodeonline_gifts
## merchantCategoryCodeonline_retail        .
## merchantCategoryCodeonline_subscriptions
## merchantCategoryCodepersonal care        .
## merchantCategoryCoderideshare            .
## merchantCategoryCodesubscriptions        ***
## transactionTypeADDRESS_VERIFICATION      .
## transactionTypePURCHASE
## transactionTypeREVERSAL
## correctCVVTrue                           ***
## cardPresentTrue                          ***
## currentBalance                           .
## creditLimit
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 26949  on 19725  degrees of freedom
## Residual deviance: 24054  on 19699  degrees of freedom
## AIC: 24108
##
## Number of Fisher Scoring iterations: 15
```

```
logit.pred <- predict(model_logit, test)
logit.pred_label <- (logit.pred > 0.5)

logitpredtable <- table(pred = logit.pred_label, true = test$isFraud)
logitpredtable
```

```
##        true
## pred    False True
```

```
##   FALSE  3457 2228
##   TRUE    283  608
```

```
accuracy.logit <- sum((prop.table(logitpredtable)[1,1])+(prop.table(logitpredtable)[2,2]))
print(accuracy.logit)
```

```
## [1] 0.6181569
```

In my Logistic regression model, I observed 0.6199818 accuracy. as you can see in the summary, some attribute play an important role and are significant, such as transactionAmount, merchantCategoryCodeauto, merchantCategoryCodefastfood, merchantCategoryCodefood, merchantCategoryCodehotels, merchantCategoryCodesubscriptions, transactionTypeADDRESS_VERIFICATION, correctCVVTrue, and cardPresentTrue.

Random Forest

```
require('randomForest')
```

```
## Loading required package: randomForest
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(54321)
model_randFor <- randomForest( as.factor(isFraud) ~ transactionAmount + merchantCategoryCode +
                                 transactionType + correctCVV +
                                 cardPresent + currentBalance + creditLimit,
                               data = train, importance=TRUE, proximity=TRUE)

print(model_randFor)
```

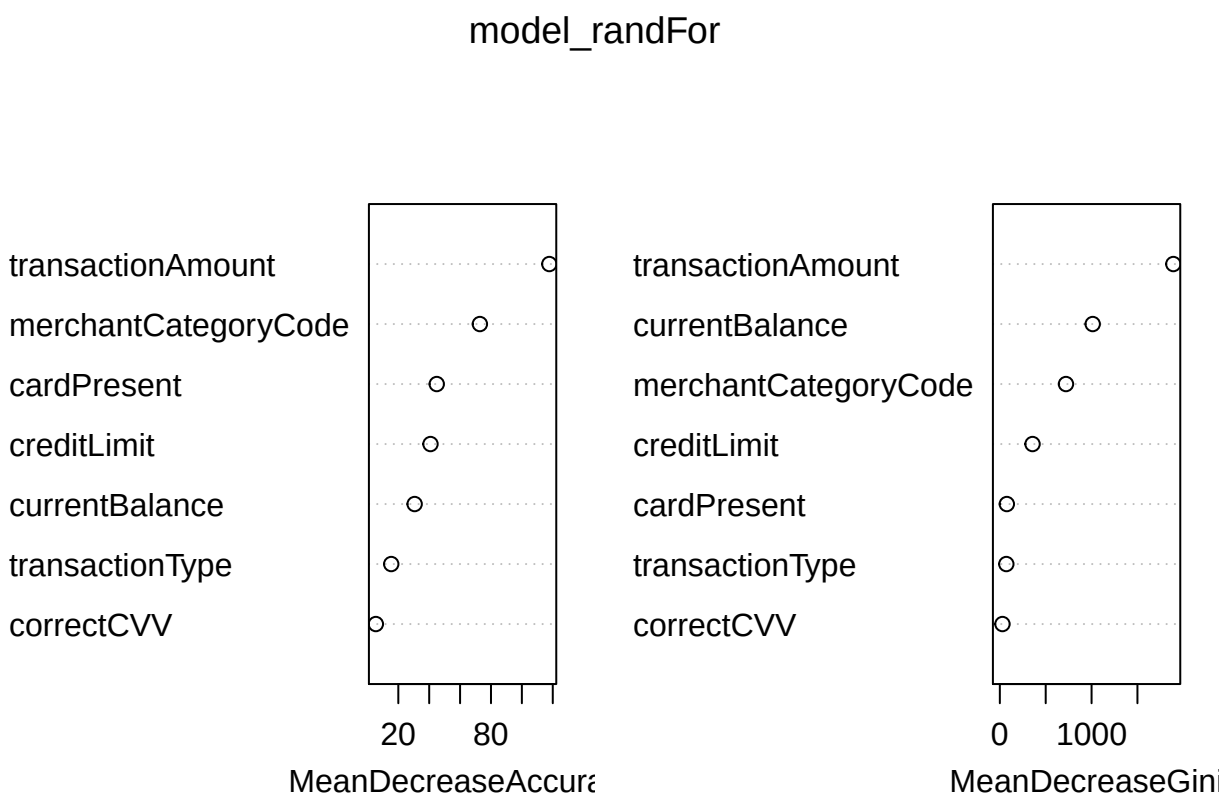
```
##
## Call:
##  randomForest(formula = as.factor(isFraud) ~ transactionAmount +      merchantCategoryCode + transac
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 33.7%
## Confusion matrix:
##       False True class.error
## False  8455 2805   0.2491119
## True   3842 4624   0.4538153
```

```
round(importance(model_randFor),2)
```

```
##                      False   True MeanDecreaseAccuracy MeanDecreaseGini
## transactionAmount    68.77 133.38               117.77          1891.26
```

```
## merchantCategoryCode 49.53  -0.93                72.81          720.77
## transactionType       9.22   7.95                15.42           70.72
## correctCVV            8.17  -0.25                 5.46           28.29
## cardPresent          45.99 -19.35                44.89           76.04
## currentBalance       13.40  13.46                30.57         1012.23
## creditLimit           9.32  26.67                40.83          356.25
```

```
varImpPlot(model_randFor)
```




```
rf.pred <- predict(model_randFor, test)


RFpredtable <- table(pred = rf.pred, true = test$isFraud)
RFpredtable
```

```
##        true
## pred    False True
##   False  2751 1282
##   True    989 1554
```

```
accuracy.rf <- sum((prop.table(RFpredtable)[1,1])+(prop.table(RFpredtable)[2,2]))
print(accuracy.rf)
```

```
## [1] 0.6546533
```

SVM

```
require(e1071)
```

```
## Loading required package: e1071
```

```
svm.model <- svm(as.factor(isFraud) ~ transactionAmount + merchantCategoryCode +
                   transactionType + correctCVV +
                   cardPresent + currentBalance + creditLimit,
                   data = train, cost = 10, gamma = 0.1, probability=TRUE)

svm.pred <- predict(svm.model, test)
SVMpredtable <- table(pred = svm.pred, true = test$isFraud)
SVMpredtable
```

```
##        true
## pred    False True
##   False  2781 1341
##   True    959 1495
```

```
SVMproppred <- prop.table(SVMpredtable)
SVMproppred
```

```
##        true
## pred        False      True
##   False 0.4229015 0.2039234
##   True  0.1458333 0.2273418
```

```
SVMgoodpred <- sum((SVMproppred[1,1])+(SVMproppred[2,2]))
SVMgoodpred
```

```
## [1] 0.6502433
```

Result Comparison

Logistic Regression is a linear classifier. If the decision boundary is non-linear other methods would outperform. Random Forest is an ensemble method which is capable of handeling non-linear boundary. SVM can also be useful when we have non-linear boundary since by using kernel functions (such as RBF) we can handle this non-lineartity.

the results I obtained shows that SVM outperform logistic regression and Random Forest.

Conclusion and Future Work

In fraud detection, we often deal with highly imbalanced datasets. For the chosen dataset, I showed that my proposed approaches are able to detect fraud transactions with reasonable accuracy which relates to low false positive rate. I found that the resulting Random forest and SVM were a little stronger and more robust than the other methods (I tried some other methods such as KNN, which did not give me better result, and they were perform weakly. I did not keep them here, because of memory usage)

As demonstrated before, the precision of the algorithms increases when the size of dataset is increased. Hence, more data will surely make the model more accurate in detecting frauds and reduce the number of false positives. We can further improve our techniques by using algorithms like Decision trees to leverage categorical features associated with accounts/users in this dataset. We can create user specific models - which are based on user's previous transactional behavior - and use them to further improve our decision-making process. All of these, I believe, can be very effective in improving my classification quality on this dataset, and there is some room for improvement here.

Github link : https://github.com/marjanrez/Econometrics-Final-Project

Bibliography

ftp://ftp.cdc.gov/pub/Health_Statistics/NCHS/Dataset_Documentation/NHIS/2014/personsx_layout.pdf

ftp://ftp.cdc.gov/pub/Health_Statistics/NCHS/Dataset_Documentation/NHIS/2014/personsx_layout.pdf

http://cs229.stanford.edu/proj2018/report/261.pdf

paper https://ieeexplore.ieee.org/abstract/document/1297040

http://www.ecmlpkdd2018.org/wp-content/uploads/2018/09/567.pdf