Detecting a Niche of Fraudsters

Sharon Colson, Breanna Moore, Gavin Mcduffee, Seth Tompson

1.Problem Statement and Background

The problem is to find an optimal way to process and analyze data from retailer partners in order to detect fraud cases, so that fraudsters can be found and refused service in the future. The dataset is a list of financial purchases from retailers. The list is filled with categorical and numerical variables such as the item purchased, cost of the item, code for the item, number purchased, and many other features. The success measure for this project would be accuracy on predicting which purchases are fraud without misclassifying too many real transactions as fraud. Companies, banks, and even consumers all care about fraudsters being caught. This type of solution would allow banks to save quite the amount of money, with companies having less issues with fraudsters. Of course, consumers also benefit with better methods. With better methods, there are far less chances of customers being falsely accused of being a fraudster with a false positive. Nobody wants to be stuck in another state with their cards frozen due to fraud suspicion, so better algorithms will help save money (and keep anxiety down) substantially. Related work with this data problem is credit card companies. They take the data of transactions that a person has over a period of time, and then they can take new transactions from the person to see if they are real, or if they believe based on the data that it is fraud.

2.Data and Exploratory Analysis

The original source of the data can be viewed with a free account here:
https://challengedata.ens.fr/participants/challenges/104/

However, it is also available on the our project github repository in the Datasets directory:
https://github.com/SColson82/Detecting_Fraudsters_ML.git

The data is split into 4 sections corresponding to training and testing sets with the target variables for each already having been split from the feature datasets. The training data represents eighty percent of the total with the final twenty percent having been partitioned as a part of the analysis setup. The training set alone consists of 92,790 data points and 148 features with a combination of character, double, and integer data types. Each data represents one shopper's "cart" with the ability to add up to twenty-four unique items, each unique item has a corresponding feature indicating how many of the item has been placed into the cart, the price, model, make, and goods code. In addition to specific information for each unique item there is a feature for the total number of items purchased. The target feature, "fraud_flag" is represented in the training data set by 0 for non-fraudulent purchase and 1 indicating a fraudulent purchase. However, the target feature presented by the

testing set is presented as a ratio. We feel that this points to an initial algorithm of logistic regression.

Due to the nature of the data presentation there are MANY empty strings and NA values in the data set as the majority of the data represent less than ten unique items in the cart. All of the data features imported as double may be converted to integer but the more concerning features are the twenty-four goods code features. The majority of these are integer data, however there are values for “fullfilment”, “service”, and “warranty” preventing these features from being interpreted as integer.

Analysis of the data was completed using tidyverse, dplyr, and ggplot2 beginning with a summary (summary()) of the data and scatter plots, bar charts, and box plots in an effort to understand the data shape and confirm outliers. As might be imagined with data consisting of numbers bought and prices, the data will need to be scaled. Some features hit their max at less than fifty while others see maximums in the tens of thousands. We are still trying to fully understand the visualizations we’ve created and feel that others may be warranted but preliminary analysis indicates that most carts tend toward lower dollar and lower amounts purchased. Additionally, there are FAR fewer fraudulent data points (1319 of the total 92,790) than non-fraudulent data points so this will most likely need to be considered. Fraudulent data points do not seem to be specific to low or high numbers or items, number of unique items, type of item, or low or high price. In other words, we have not found a specific pattern yet in our analysis.

Data Dictionary:

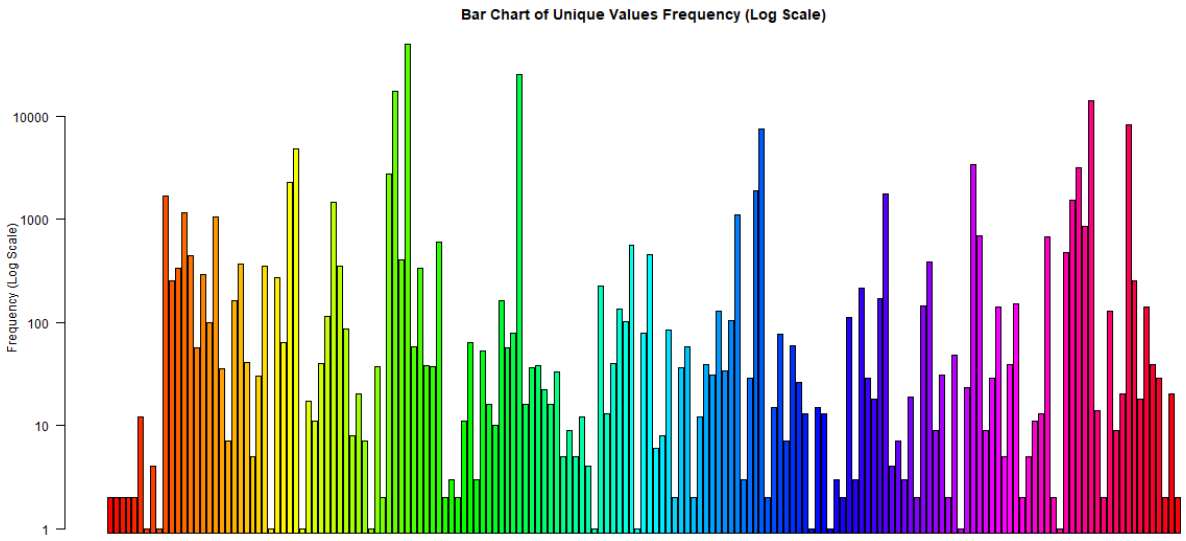
Variable	Description	Example
ID(Num) Item 1 - Item 24 (Char)	Unique Identifier Goods category for item 1 to item 24	1 Computer
Cash_price1 - Cash_price24 (Num)	Cash price for item 1 to item 24	850
Make1 - Make24 (Char)	Manufacturer for item 1 to item 24	Apple
Model1 - Model24(Char)	Model description for item 1 to item 24	Apple iphone XX
Goods_code1 - Goods_code24 (Char)	Retailers code for item 1 to item 24	2378284364

Nbr_of_prod_purchase1 to Nbr_or_prod_purchase24 (Num)	Number of products for item 1 to item 24	2
Nb_of_Items(Num)	Total number of items	7
Fraud_flag	Purchase fraudulence/ non-fraudulence	1 0

Truncated Data Sample:

	item1	item2	item3	item4
85517	COMPUTERS	NA	NA	NA
51113	COMPUTER PERIPHERALS ACCESSORIES	NA	NA	NA
83008	TELEVISIONS HOME CINEMA	NA	NA	NA
78712	COMPUTERS	COMPUTER PERIPHERALS ACCESSORIES	NA	NA
77846	TELEVISIONS HOME CINEMA	NA	NA	NA
86994	COMPUTER PERIPHERALS ACCESSORIES	NA	NA	NA
113204	TELEVISIONS HOME CINEMA	SERVICE	NA	NA
41798	COMPUTERS	NA	NA	NA
83365	COMPUTERS	NA	NA	NA
39361	COMPUTERS	NA	NA	NA
85472	TELEVISIONS HOME CINEMA	NA	NA	NA
107968	TELEVISIONS HOME CINEMA	NA	NA	NA

Unique Items by Frequency:



3.Methods

Initially, the dataset read in with empty strings in the character features and NAs in the numeric. Empty strings were replaced with NA using the apply() function. The target feature was read in separately and appeared as three columns: index, ID, and fraud_flag. ID (present in the x-train set as well) was found to be unique for each row of the data so we indexed both sets by it, removed the extraneous index column of y-train, and merged x-train with y-train using cbind().

```
xTrainData <- read.csv("Datasets/X_train_G3tdtEn.csv",row.names = "ID")
xTrainData <- apply(xTrainData, 2, function(x) ifelse(x == "", NA, x))
yTrainData <- read.csv("Datasets/Y_train_2_XPXJDyy.csv")
yTrainData <- yTrainData %>%
  select(ID, fraud_flag)
rownames(yTrainData) <- yTrainData$ID
yTrainData <- yTrainData[, "fraud_flag", drop = FALSE]
train_data <- cbind(xTrainData, y = yTrainData)
train_data
summary(train_data)
glimpse(train_data)
```

Once this had been completed, we determined that all columns with the strings “cash” and “Nb” in the name could be converted to integer data types. This was done using the mutate() function.

```
keywords <- c("cash", "Nb")

train_data <- train_data %>%
  mutate_at(vars(contains(keywords)), as.integer)
colnames(train_data) <- make.names(colnames(train_data))
```

The models that we considered are logistic regression, naive bayes, support vector machines, classification trees, and random forest. Initially, the naive bayes model was rejected due to the fact that it assumes features are independent and equally important. Logistic regression seemed like the ideal candidate but because its most basic form deletes rows with NA values, we moved away from this. It would cause us to disregard most of our data, so we decided against using it. To add onto that, the NA values in the dataset actually hold meaning for the problem we are trying to solve. Support Vector Machines was rejected for the same reason.
We wanted to find a model that could account for the number of NA values in the dataset without having to do a large amount of transformation. We attempted a model using classification trees, but we ran into problems due to how it handles missing data by imputing it. We did not wish to do this, as it would likely interact poorly with our dataset, and cause bias in our problem statement.
Random Forest was able to handle the missing values through Hot One encoding, however the number of unique values across our categorical features caused a

stack overflow error. Therefore, due to the limitations on computational resources and the nature of this short term project, we are currently considering other ways that we might do some additional processing of the data without losing data integrity. The data is sectioned into items, item price, make, model, goods code, number of each item sold, and total number of items sold. In the next iteration of this project we intend to attempt the following transformations:
We are trying to cut several parts of the dataset and merge them into one column. The main parts we decided to combine were the 24 cash_price columns. There are 24 of the columns, one for each of the possible items that people could add to their online cart. We want to create a dummy items column to have the columns based on the numbers of each make inside the cart. (looks to see if a make, like apple, is inside the cart at any point and adds the number of times that make is inside the cart. If a make is not inside the cart it has the value 0.

Truncated Initial View of Data as It Was Read In:

```
Rows: 92,790
Columns: 145
$ item1        <chr> "COMPUTERS", "COMPUTER PERIPHERALS ACCESSORIES", "TELEVISIONS HOME CINEMA", "COMPUTERS",
$ item2        <chr> "", "", "", "COMPUTER PERIPHERALS ACCESSORIES", "", "", "SERVICE", "", "", "", "", "", "
$ item3        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "FULFILMENT CHARGE", "", "",
$ item4        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item5        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item6        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item7        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item8        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item9        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item10       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item11       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item12       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item13       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item14       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item15       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item16       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item17       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item18       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item19       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item20       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item21       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item22       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item23       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ item24       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
$ cash_price1  <dbl> 889, 409, 1399, 689, 1199, 369, 1299, 1187, 1899, 898, 1178, 799, 569, 379, 1899, 717, 2
$ cash_price2  <dbl> NA, NA, NA, 119, NA, NA, 125, NA, NA, NA, NA, NA, NA, NA, NA, 119, 17, 200, NA, 550, NA,
$ cash_price3  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 7, NA, NA, NA, 0, NA, NA, NA
$ cash_price4  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
```

4.Tools

The tools that we used to implement the methods were: cbind(), na.omit(), various parts of the tidyverse, dplry(), randomForrest(), summary(), apply(), mutate(), and classification trees. cbind() worked well for us because it allowed us to combine column values into new columns used to evaluate the performance measure of detecting fraud in the carts more accurately. Another tool that did work for us was the apply() function. It helped us to replace empty strings in the dataset with NA to make it more readable and easier to manipulate in the methods we used.
How we implemented the apply(), and cbind() in our methods was with the following code.

```
xTrainData <- read.csv("Datasets/X_train_G3tdtEn.csv",row.names = "ID")
xTrainData <- apply(xTrainData, 2, function(x) ifelse(x == "", NA, x))
yTrainData <- read.csv("Datasets/Y_train_2_XPXJDyy.csv")
yTrainData <- yTrainData %>%
  select(ID, fraud_flag)
rownames(yTrainData) <- yTrainData$ID
yTrainData <- yTrainData[, "fraud_flag", drop = FALSE]
train_data <- cbind(xTrainData, y = yTrainData)
train_data
summary(train_data)
glimpse(train_data)
```

The mutate function also worked for us to manipulate certain data types to become integers, making it easier to run models on. We implemented the tool mutate() in our methods by using it in the method to turn certain data into integer values.

```
keywords <- c("cash", "Nb")

train_data <- train_data %>%
  mutate_at(vars(contains(keywords)), as.integer)
colnames(train_data) <- make.names(colnames(train_data))
```

With these two functions we can use the method, random forest, to see what is fraud and what is not fraud. These tools make the data more concise and manipulable, so running models is easier. This makes it more manageable to figure out the problem statement of detecting fraud in the test data. We could improve model functionality by getting the data smaller, so the models can run more efficiently and be more productive.

One tool that didn't work well for our methods was na.omit() because we have many NA values in the dataset that hold value in the data. Due to this, we can not simply omit rows with NA values. If we were to simply remove the rows with NAs, the data would be biased, and we would not get an accurate result to the problem statement. Classification trees also did not work for the dataset due to how similar it is to na.omit(). It would get rid of the missing values which would cause us to run into many bias-based issues.

The tools apply() and mutate() helped us get the data to a semi-manageable state where we are able to see what the next steps are in the project. However, the data is still very large and too difficult to run efficiently on our computers. We are brainstorming more tools that can possibly be used to improve the size of the dataset to a more controllable size when implementing our methods without causing bias in the problem statement.

5.Results

Evaluation metric

The objective is to identify fraudulent operations within a population by predicting a fraud risk/probability. Hence, the metric that will be used is the area under the precision-recall curve, also noted the PR-AUC.

The precision-recall curve is obtained by plotting the precision ($\frac{TP}{TP+FN}$) on the y-axis and the recall ($\frac{TP}{TP+FP}$) on the x-axis for all values of the probability thresholds between 0 and 1.

This metric is very useful for properly evaluating a model's performance on the minority class in severely imbalanced classification problems.

The higher the PR-AUC, the better the model is at correctly detecting the minority class.

In this challenge, the PR-AUC will be calculated as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight:

$$\text{PR-AUC} = \Sigma_n (R_n - R_{n-1}) P_n \,,$$

where P_n and R_n are the precision and recall at the nth threshold.

N.B. This implementation corresponds to the ***average_precision_score*** metric of *sklearn*.

Hence, your **submission file** should have **the following format**:

Variable	Description
ID (Num)	Unique identifier
fraud_flag (Num)	Probability estimate (positive float between 0 and 1) for the minority class (fraud). The higher the riskier.

You can use the *Y_test_random* and *Y_test_benchmark* .csv files as examples to verify the exact format and size of a valid submission file for this challenge.

[Give a detailed summary of the results of your work. Here is where you specify the exact performance measures you used. Usually there will be some kind of accuracy or quality measure. There may also be a performance (runtime or throughput) measure. Please use visualizations whenever possible. Include links to interactive visualizations if you built them. You should attempt to evaluate a primary model and in addition a "baseline" model. The baseline is typically the simplest model that's applicable to that data problem, e.g. Naive Bayes for classification, or K-means on raw feature data for clustering. If there isn't a plausible automatic baseline model, you can e.g. compare with human performance by having someone hand-solve your problem on a small subset of data. You won't expect to achieve this level of performance, but it establishes a scale by which to measure your project's

performance. Compare the performance of your baseline model and primary model and explain the differences.]

6.Summary and Conclusions

[In this section give a high-level summary of your results. If the reader only reads one section of the report, this one should be it, and it should be self-contained. You can refer back to the "Results" section for elaborations. This section should be less than a page. In particular, emphasize any results that were surprising.]

7.Appendix

Include the link to your github/gitlab repository (that I can access) containing your R programs/scripts, and link to the data.

The link to the github repository we made:
https://github.com/SColson82/Detecting_Fraudsters_ML.git
The link to the challenge website(you need to create an account to see the datasets):
https://challengedata.ens.fr/participants/challenges/104/