DSO 562: Fraud Analytics

Supervised Fraud Detection: Credit Card Transactions

Team 4

Jasleen Kaur Ahuja

Aishwarya Joshi

Dhwani Kapadia

Mahesh Pandit

Shashank Ravi Shankar

Shubham Rishishwar

Shringar Sharan

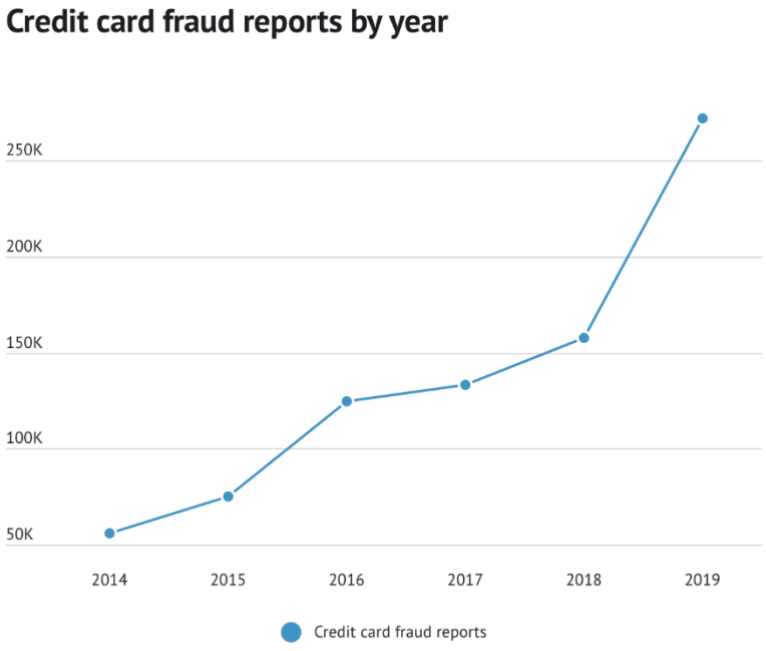
Table of Contents

1.0 EXECUTIVE SUMMARY

Payment card fraud is a growing concern around the world, not only for financial institutions, but also for individuals whose identities may be stolen to commit the fraud. Payment card fraud can be broadly classified into two categories:

1. Application Fraud
2. Transaction Fraud

According to the Federal Trade Commission, there were more than 250,000 reports of credit card fraud in the year 2019. The graph below shows the yearly trend of credit card fraud in the United States for the past five years



Data source: Federal Trade Commission (2020).

In this project, our objective is to curtail credit card transaction fraud by assigning a real-time fraud score to each and every transaction using supervised classification models built using Logistic Regression, Neural Networks, Boosted Trees and Random Forest. The data we used in this project consists of credit card transaction data for all the days in the year 2010, with each record identified either as a fraudulent or genuine transaction.

Given below is a high-level outline of the steps we took to achieve the objective of the project:

1. **Exploratory data analysis and data cleaning**. The data consists of 10 fields and 96753 records. We only retained the records with transaction type 'P'(Payment) and imputed

some missing values for the fields merchant number, merchant zip code and merchant state.

2. **Feature engineering and feature selection**. We engineered features by combining fields such as card number and merchant number with the average/median amount spent in each transaction. We also created days since variables and velocity variables for the combination variables. Among the 343 variables that were engineered, the 30 most relevant features were selected using a random forest wrapper.
3. **Building classification models and evaluating performance**. We divided the data into training, testing and out of time (OOT) sets, and built four classification models using Logistic Regression, Neural Networks, Boosted Trees and Random Forest. The performance of each model was evaluated using the Fraud Detection Rate in the top 3% of the OOT data.
4. **Determine the optimal score cutoff to maximize savings**. We assumed a $2000 savings for every fraudulent transaction that was caught and a $50 loss for every false positive. For our best performing model, we calculated the fraud savings, lost sales and overall savings at each score cutoff, above which the transaction will be flagged. The most profitable score cutoff was determined.

The result of the project was that the best classifier was built using a Random Forest Model, which had an average **FDR of 59.2%** in the top 3% of the OOT data. We recommend a score cutoff at the top 3% of the OOT population, which will result in yearly savings of over $2.2 million.

2.0 DATA DESCRIPTION

Credit Card Transactions Data contains details for credit card transactions for the period of one year from January 1, 2010 to December 31, 2010. We will use this data to create a supervised fraud model to detect credit card transactions fraud.

Following are some of the characteristics of this dataset:

- Dataset Name: Credit Card Transactions Data
- Number of Records/Rows: 96753
- Number of Fields/Columns: 10
- 9 fields in the dataset are categorical whereas only one is numerical (Amount).
- Each record in the dataset represents a credit card transaction and the characteristics of the transaction have been captured in fields such as Card number, Date, Transaction Type, Merchant number, Amount, Merchant Description, Zip code of merchant etc. The fraud label field indicates whether the record represents a fraudulent transaction or not. Here, 1 corresponds to a fraudulent activity whereas 0 does not.

2.1 SUMMARY OF DATA FIELDS

There are nine categorical fields and one numerical field.

Table 1: Summary Characteristics of All Fields

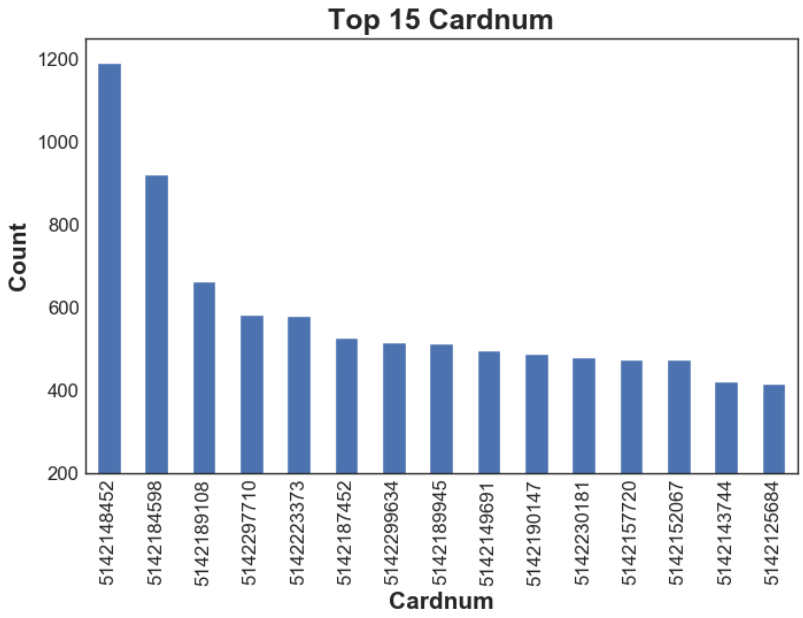
Fields	# records with value	Type	% populated	# unique values	#records with zero as a value	Mean	SD	Min Value	Max Value	Most common value (MCV)	Freq. of MCV
Recnum	96753	Categorical	100%	96753	0	NA	NA	NA	NA	NA	NA
Cardnum	96753	Categorical	100%	1645	0	NA	NA	NA	NA	5142148452	1192
Date	96753	Categorical	100%	365	0	NA	NA	NA	NA	2010-02-28	684
Merchnum	93378	Categorical	96.5%	13092	0	NA	NA	NA	NA	930090121224	9310
Merch description	96753	Categorical	100%	13126	0	NA	NA	NA	NA	GSA-FSS-ADV	1688
Merch State	95558	Categorical	98.76%	227	0	NA	NA	NA	NA	TN	12035
Merch Zip	92097	Categorical	95.18%	4568	0	NA	NA	NA	NA	38118	11868
Transtype	96753	Categorical	100%	4	0	NA	NA	NA	NA	P	96398
Amount	96753	Numerical	100%	34909	0	4.27e+02	1.0e+04	1.0e-02	3.1e+06	NA	NA
Fraud	96753	Categorical	100%	2	95694	NA	NA	NA	NA	0	95694

2.2 DESCRIPTION OF PRIMARY FIELDS

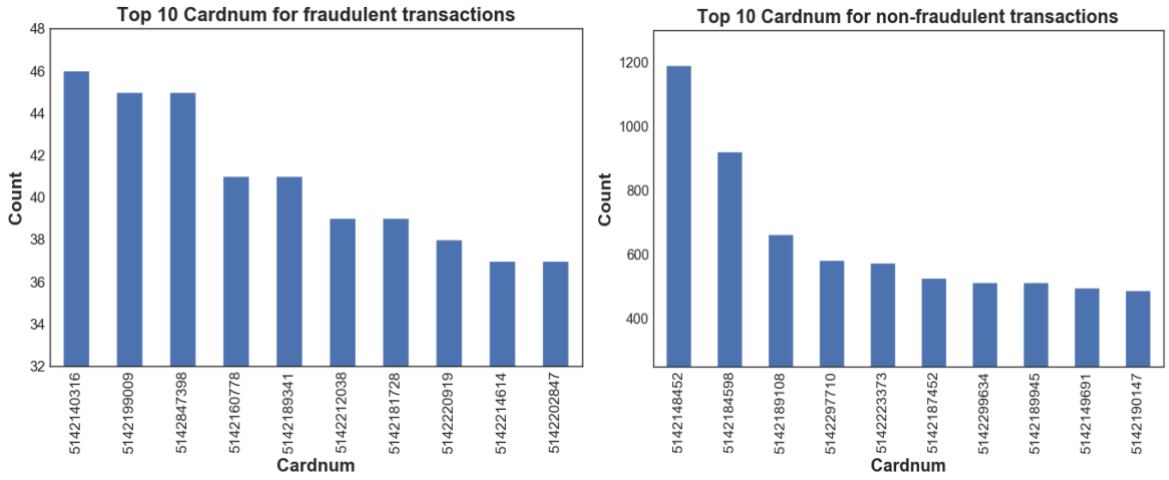
Now, we will explore in detail, the characteristics of a few important fields by plotting various graphs based on the category of the field.

2.2.1 Field Name: Cardnum

Description: This is a categorical field and corresponds to the credit card number used for each transaction. There are 1645 unique values and there are no missing values for this field. We can see that the card number '5142148452' is repeated 1192 times in the dataset. The top 15 card numbers according to the count of transactions are shown as follows in the table below.

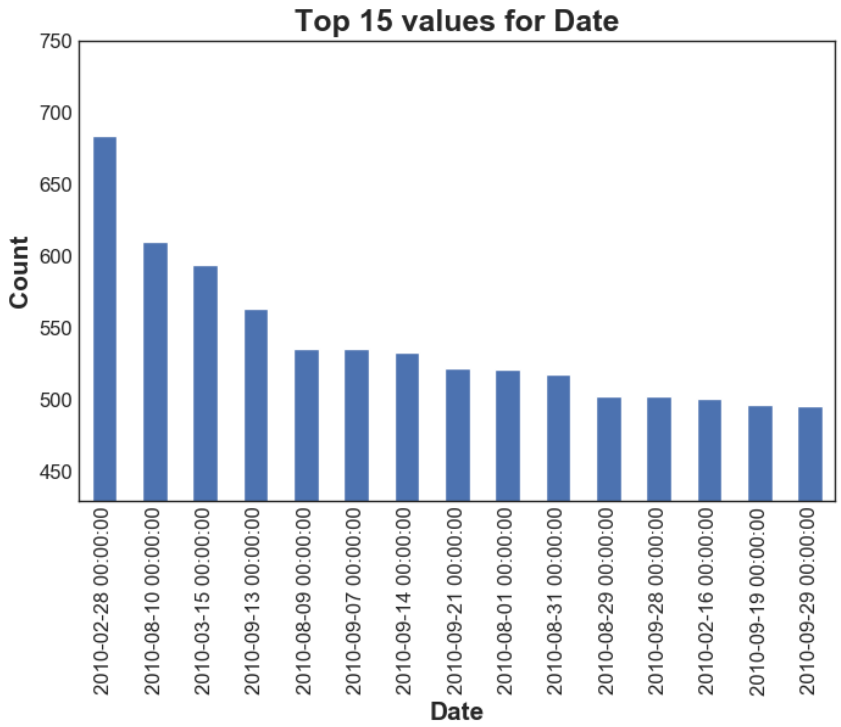


The following graphs show the count of top 10 credit card numbers used for fraudulent as well non-fraudulent transactions.

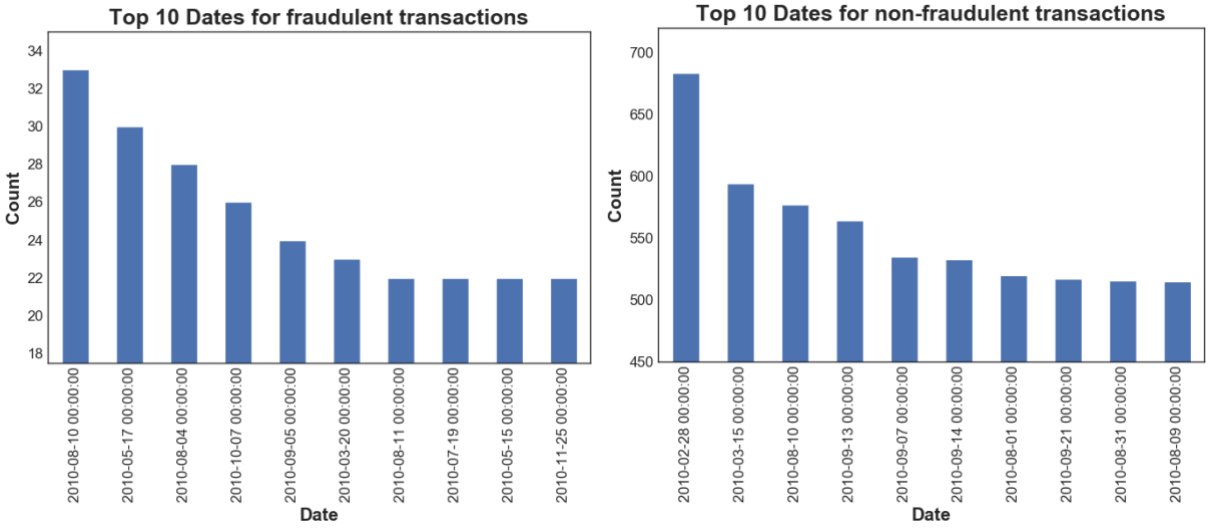


2.2.2 Field Name: Date

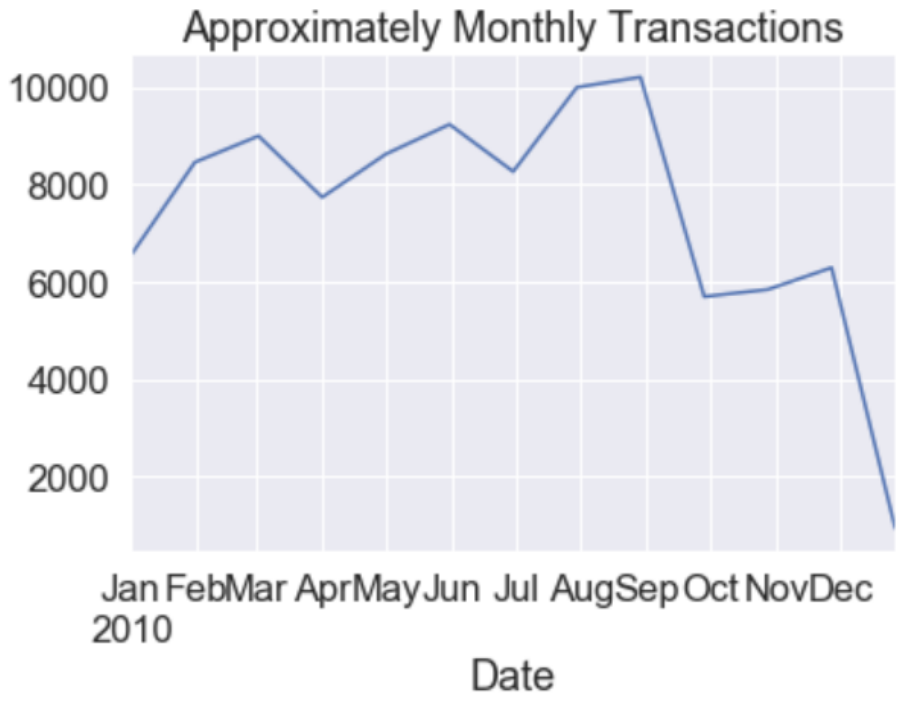
Description: This field contains the Date of the transaction and is of type datetime. There are 365 unique values corresponding to each day of the year 2010.



The following graphs show the count of top 10 dates for fraudulent as well non-fraudulent transactions.

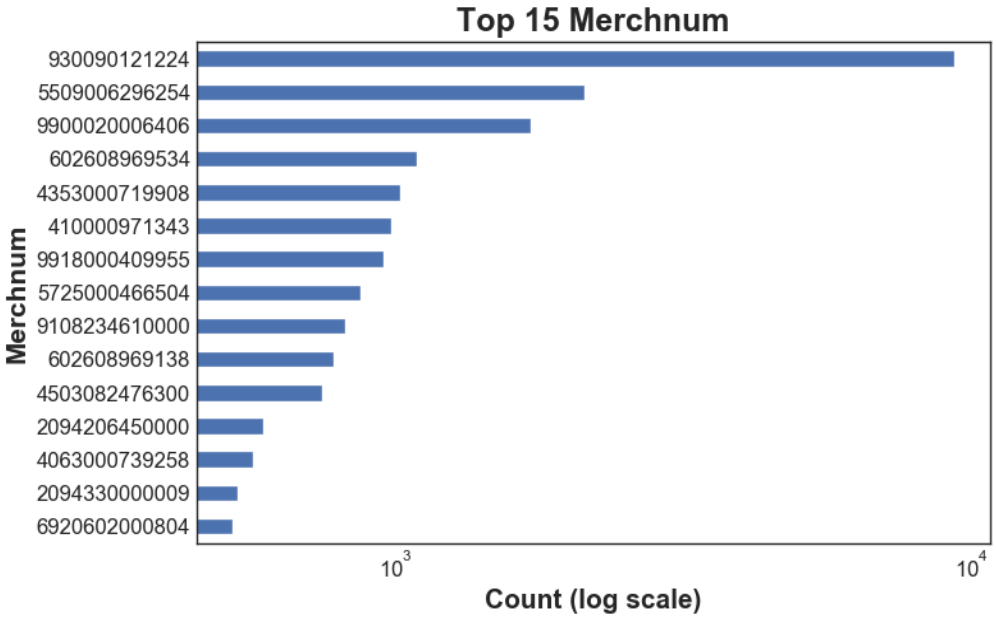


There is a drop in monthly transactions at the beginning of the fiscal year between Oct-Dec, when government employees are more conservative with spends. There is a spike at the end of the year around September when they need to spend the remainder of allocated budget.



2.2.3 Field Name: Merchnum

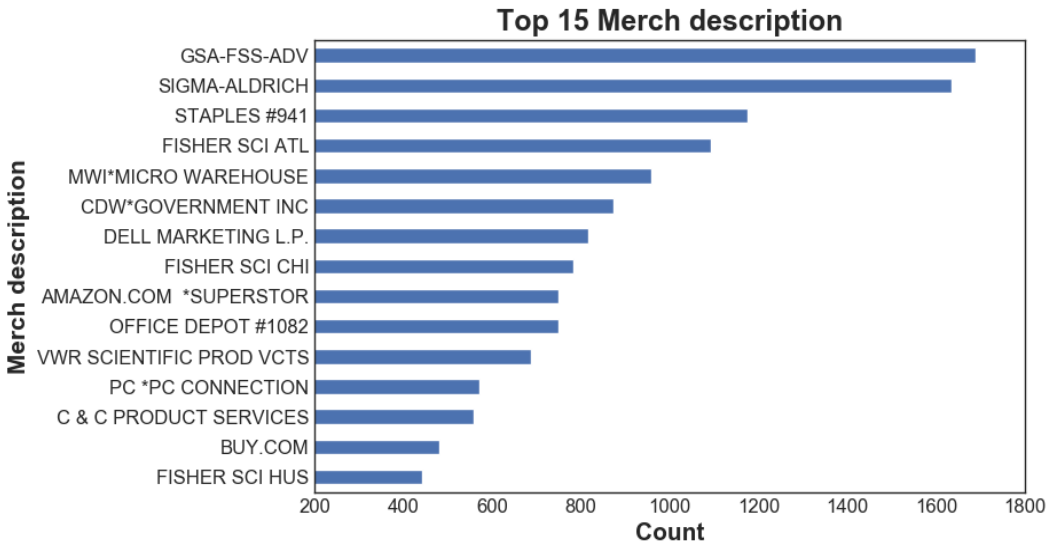
Description: This is a categorical field and around 96.5% values are filled. The bar graph and the table show top 15 Merchnum by count.



Merchnum	Count
930090121224	9310
5509006296254	2131
9900020006406	1714
602608969534	1092
4353000719908	1020
410000971343	982
9918000409955	956
5725000466504	872
9108234610000	817
602608969138	783
4503082476300	746
2094206450000	590
4063000739258	568
2094330000009	533
6920602000804	523

2.2.4 Field Name: Merch description

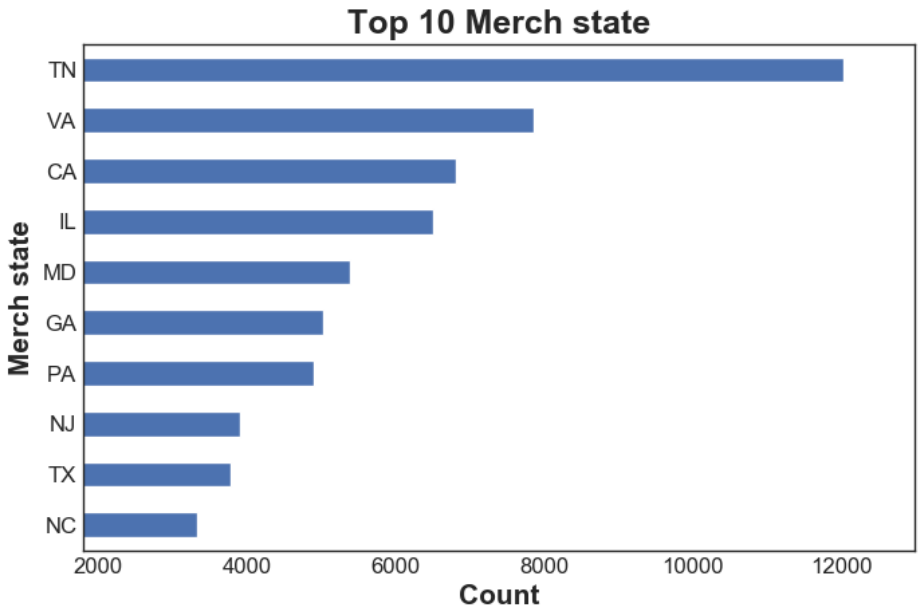
Description: This field contains the merchant description and is a categorical field. It gives information about the location of each merchant. There are 13126 unique values, and it is 100% populated. The bar graph shows the top 15 Merch description by Count.



Merch description	Count
GSA-FSS-ADV	1688
SIGMA-ALDRICH	1635
STAPLES #941	1174
FISHER SCI ATL	1093
MWI*MICRO WAREHOUSE	958
CDW*GOVERNMENT INC	872
DELL MARKETING L.P.	816
FISHER SCI CHI	783
AMAZON.COM *SUPERSTOR	750
OFFICE DEPOT #1082	748
VWR SCIENTIFIC PROD VCTS	688
PC *PC CONNECTION	570
C & C PRODUCT SERVICES	558
BUY.COM	481
FISHER SCI HUS	442

2.2.5 Field Name: Merch state

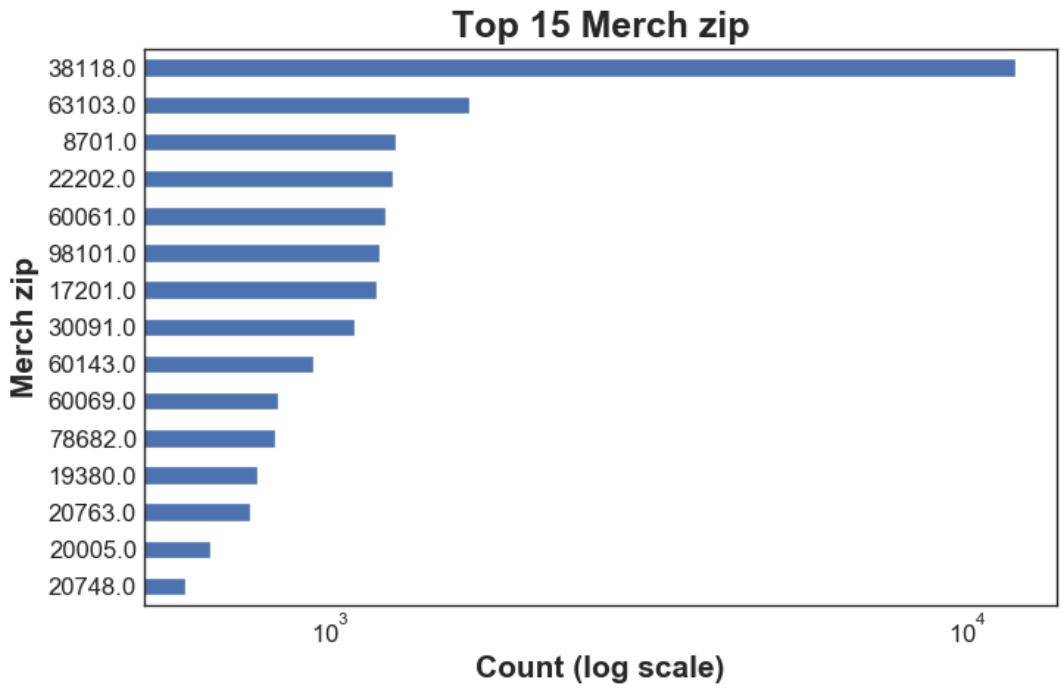
Description: This field contains the merchant state which indicates the location of each merchant. The bar graph shows the top 10 Merchant states by count.



Merchant state	Count
TN	12035
VA	7872
CA	6817
IL	6508
MD	5398
GA	5025
PA	4899
NJ	3912
TX	3790
NC	3322

2.2.6 Field Name: Merch zip

Description: This is a categorical field contains the zip codes of the merchants for each transaction. It has 4568 unique values in the dataset. The bar graph shows the top 15 values by count (log scale).



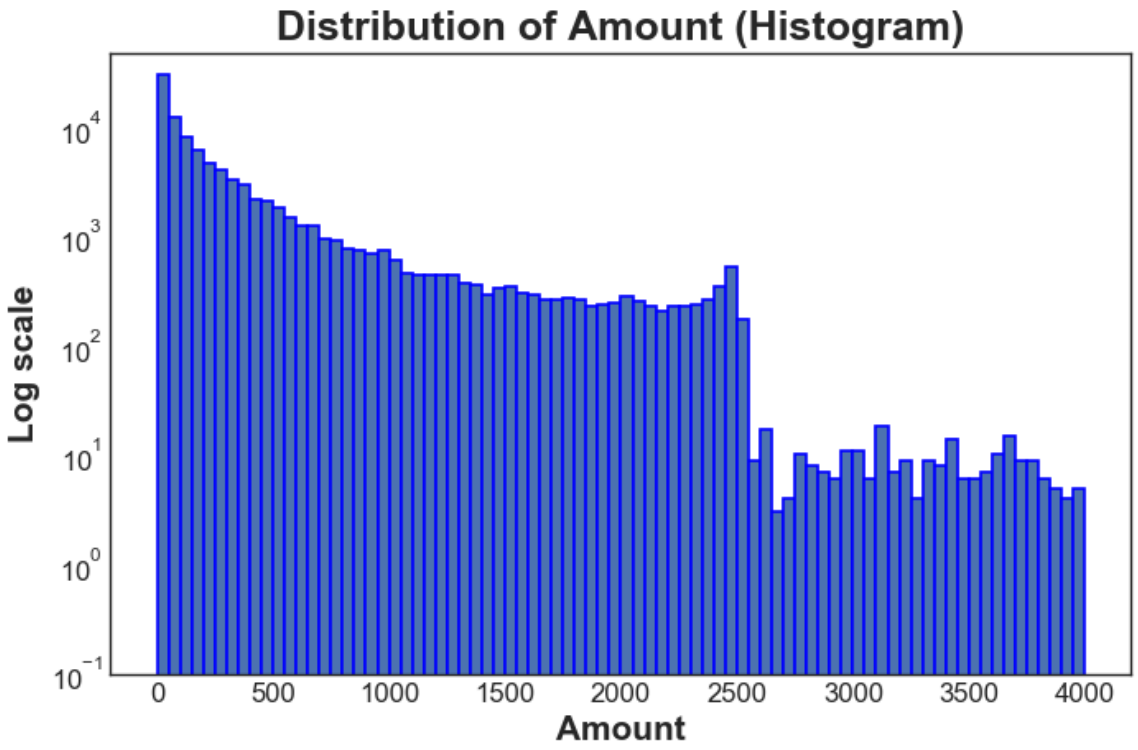
Merch zip	Count
38118	11868
63103	1650
8701	1267
22202	1250
60061	1221
98101	1197
17201	1180
30091	1092
60143	942
60069	826
78682	817
19380	769
20763	749
20005	648
20748	592

2.2.7 Field Name: Amount

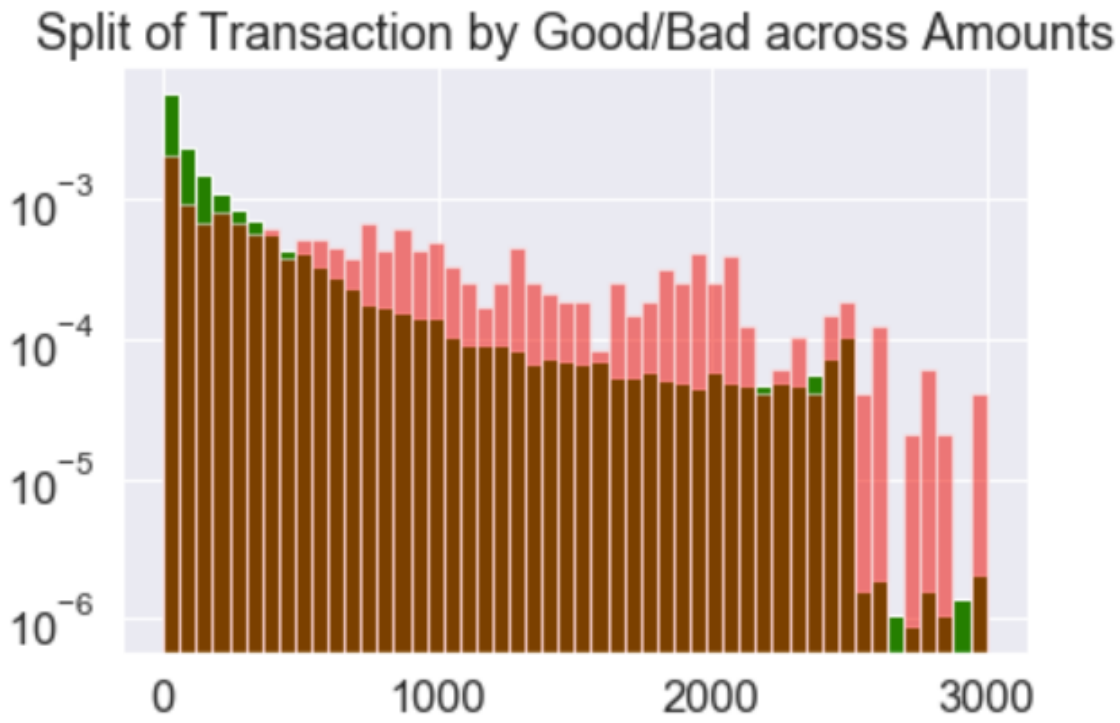
Description: This is a numeric field and contains the amount of the particular transaction. The histogram shows the distribution of this field on a log y scale.

Count	9.675300e+04
Mean	4.278857e+02
Std	1.000614e+04
Min	1.000000e-02
25%	3.348000e+01
50%	1.379800e+02
75%	4.282000e+02
Max	3.102046e+06

We can see here that there are fewer transactions with higher amounts, likely due to some cap on government spending after a certain threshold.



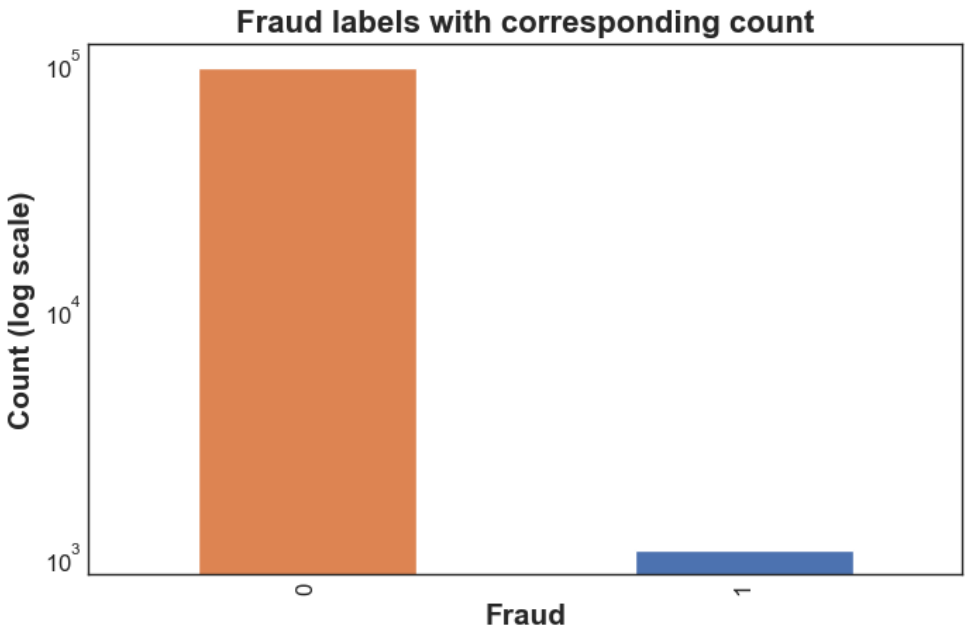
Also, there are more fraudulent cases at higher transaction amounts.



2.2.8 Field Name: Fraud

Description: This field contains the label which indicates whether the particular record is identified as fraud or not. It has two unique values, 0 which indicates the particular record is not fraud whereas 1 indicates that the record is fraudulent. There are 95694 zeros and 1059 ones. This shows that around 1.1% of the dataset consists of fraudulent transactions. Below is the log distribution of these two categories.

Value	Count	Percent
0	95694	98.9%
1	1059	1.1%


Fraud labels with corresponding count
Count (log scale)
10^5
10^4
10^3
0
1
Fraud

3.0 DATA CLEANING

We hereby describe the logic for imputing the missing values in the data. We took the following steps to clean the data.

1. Retained only type P transactions and removed others
2. Removed 1 outlier record with amount > 3 million since it was recorded in pesos and not converted to USD.
3. Imputed NA values in three columns: Merchnum, Merch zip, Merch state

We observed that for some of the transactions, for which the Merch description was 'RETAIL DEBIT ADJUSTMENT' or 'RETAIL CREDIT ADJUSTMENT', there was no information in any of these 3 columns (namely, Merchnum, Merch state, Merch zip) to impute them by. Hence we decided to group these transactions as well as other transactions which had missing values in all these columns by the Merchant description and fill all of these columns with the first 'Recnum' of the group.

We then looked individually at the three fields and used a similar logic to fill all of them.

3.1 MERCHNUM

- First, we replaced values having '0' with NaN
- We then grouped by Merch description and Merch zip and then used the mode of the group to fill NA values in Merchnum
- For the rows that were left, we grouped by Merch description and used the first 'Recnum' of the group to fill NA values in Merchnum

We used a similar logic for Merch Zip and Merch State.

3.2 MERCH ZIP

- Grouped by Merchnum and Merch state and used the mode of the group to fill NA values in Merch zip
- For the rows that were left, we grouped by Merchnum and used the first 'Recnum' of the group to fill the NA values in Merch zip

3.3 MERCH STATE

- Grouped by Merchnum and Merch zip and used the mode of the group to fill NA values in Merch state
- For the rows that were left, we grouped by Merchnum and used the first 'Recnum' of the group to fill the NA values in Merch state

4.0 VARIABLE CREATION

We have built candidate variables on the basis of 5 entities- Cardnumber (cardnum), Merchant number (Merchnum), and three new fields that we created by combining Cardnum with Merchnum, Merch zip and Merch state: Cardnum_Merchnum, Cardnum_Merch zip, Cardnum_Merch state.

Next, we built 6 time-adjusted windows (0 days, 1 days, 3 days, 7 days, 14 days, 30 days) to create the variables that captured the relation between time and fraud.

Using these entities over 6 time windows, we have created 4 types of candidate variables based on amount, frequency, days since the card was last seen and the velocity.

Expert variables for each group:

1. **Amount Variables**

- We utilized several aggregation approaches for transaction amount such as average, maximum, median, sum and the ratio of actual amount and aggregated metrics
- Using these aggregation approaches, we built variables over the past n days, where n = 0, 1, 3, 7, 14, 30. For each entity and combination variable, we created 48 variables (one for each time stamp)
- For example, 'Amount_Cardnum_sum_0d' is the sum of the amount of transactions for a particular card number on the same day
- Hence, since we had 5 fields, we created 240 Amount variables

2. **Frequency Variables**

- Number of transactions seen over the past n days, where n = 0, 1, 3, 7, 14, 30 for each of the entities. These variables tell us how many times an entity or a combination group is seen over past n days.
- For each entity and combination variable, we created 6 variables (one for each time stamp).
- For example, 'Cardnum_count_0d' is the frequency of transactions for a particular card on the same day.
- Hence, since we had 5 fields, we created 30 Frequency variables

3. **Days Since Variables**

- For each entity or combination, this variable denotes the number of days since we last saw that element.
- For example, 'Days_since_Cardnum' is the number of days since a card number was last seen

- If we see multiple Card numbers on the same day, the count for the second occurrence becomes zero, as we had already seen that Card number earlier in the day
- Since there was only 1 'days_since' variable for each existing field, we created 5 'days since' variables overall.

4. Velocity Variables

- Relative number and amount of transactions seen over the past i days compared to the past j days, where i = 0, 1 and j = 3, 7, 14, 30. These variables tell us how the velocity variables fluctuate over time and if any day has an unusually high number of transactions (compared to the average of the past j days) for Card number and Merchant number
- data[numerator] / (data[denominator]/j)
  where,
  numerator = velocity variables for n=1
  denominator = velocity variables for n = 3, 7, 14, 30
  j = 3, 7, 14, 30
- For example, 'count_Cardnum_0d/mean_count_Cardnum_14d' is the frequency of transactions for a particular card number on the same day over the average daily frequency of transactions for that card number over 14 days
- We have created 48 Velocity variables

We also created **Benford's law variable** which is the unusualness score and also created **day of week risk table variable** for merchant number and card number.

Benford's Law is an observation about the distribution of the first digits of many measurements occurring in real life. If a distribution does not follow this pattern, then it's likely that the numbers have been tampered with and that is how it is used to identify fraud.

Lastly, the day of the week risk table is the rate of fraud for each of the weekdays.

In total, we created **325** candidate variables.

5.0 FEATURE SELECTION

5.1 Approach 1

5.1.1 Filtering using KS score and Fraud Detection Rate (FDR)

KS score and the FDR rate helped us determine how well candidate variables separates the 'goods' from the 'bads' thereby individually predicting fraud. This allows us to rank order the variables in terms of usefulness for our models.

First, we z-scaled our candidate variables. For each of our candidate variable, we calculated Kolmogorov–Smirnov (KS) score and fraud detection rate individually.

$$KS = \max_x \int_{x_{min}}^{x} [P_{\text{goods}} - P_{\text{bads}}]\, dx$$

$$KS = \max_x \sum_{x_{min}}^{x} [P_{\text{goods}} - P_{\text{bads}}]$$

The FDR for each variable was determined at a 3% level. It is the value representing the percentage of all frauds caught at an examination cutoff. For each variable, we determined what percentage of frauds are captured by the top 3% of the variable and ranked order as such.

After getting KS and FDR table for all variables, we followed the below steps to reduce the number of variables from 325 to 80 of the top variables:

1. Sorted records by KS in the descending order

2. Replaced KS with the sorted rank order (Highest score getting rank 1 and so on)

3. Sorted records by FDR in the descending order

4. Replaced FDR with the sorted rank order (Highest score getting rank 1 and so on)

5. Next, we took their average to get a cumulative score, which is our final importance rank of variables

5.1.2 Recursive Feature Elimination Wrapper

Recursive Feature Elimination is a wrapper method that performs a backward feature selection of the variables. This method starts by fitting a model with all the variables and computes an importance score for each variable from the model. The variable with the least importance is eliminated and the model is run again with the remaining variables. This process is repeated recursively until the number of variables reaches a pre-defined number of variables and retains only the top N variables that the model finds important to predict the target.

For this wrapper, if a linear model is used it might only retain the variables that have a linear relationship with the target. This could limit any further model training on the data as all the variables with a non-linear relationship with the target maybe lost. However, using a non-linear model in the wrapper might lead to a huge computational overload. Thus, choosing a model for this wrapper is more of a trade-off between computational time and retaining relationships with the target.

To implement the wrapper for our data, we built two instances of the wrapper: one with a random forest model and another with gradient boosting model; both scored on FDR at 3%. From the two wrappers we obtained a rank for the features, both ranks, one from each wrapper, were averaged and the top 30 variables were obtained out of the 80 from the previous method.

5.2 Approach 2

In this approach were directly used the Recursive Feature Elimination Wrapper from 4.1.2 on our feature set of 325 variables. This approach directly gave us the top 30 variables from the initial feature set. On building final models for both Approach 1 and Approach 2, we discovered that the FDRs we got from the second approach was significantly higher, i.e. we got an FDR of 51.39 from Approach 1 and an FDR of 59.77 from Approach 2. **So, we decided to go ahead with Approach 2.**

In both approaches, we separated the data after Oct 31, 2010 as the Out of Time Data, whereas the records up to Oct 31, 2010 were taken for feature selection and for further model building.

6.0 ALGORITHMS

In order to predict fraud, we built four machine learning models: Logistic Regression, Random Forest, Boosted Trees and Neural Net

The goal was to evaluate the performance of each of these models on our dataset (based on 3% FDR) and then choose the best one for deployment.

6.1 Logistic Regression

Working:

Logistic Regression is a classification algorithm that uses a sigmoid function and creates a linear decision boundary to separate classes.
We have a binary output variable Y (fraud=1/not fraud=0), and we want to model the conditional probability $Pr(Y = 1|X = x)$ as a function of x; any unknown parameters in the function are to be estimated by maximum likelihood.
The equation for the Logistic Regression model is:
$\log p(x)/1 - p(x) = \beta 0 + x \cdot \beta$
We should predict $Y = 1$ when $p \geq 0.5$ and $Y = 0$ when $p < 0.5$. This means guessing 1 whenever $\beta 0 + x \cdot \beta$ is non-negative, and 0 otherwise.
This is one of the simplest kinds of algorithms used in the project, mainly to create a baseline model against which we could compare the performance of our non-linear models, which ideally should be better.

Implementation:

We used cross-validation to have 10 iterations of the model with each set of variables. Then we average the 3% FDR obtained from each set of over the 10 iterations to get the overall result. We repeat this over top 30, 25, 20, 16, 10 and 6 variables and select the best one.

This exercise is repeated with different combinations of variables and is summarized in section 6.5.

Result:

Although the average OOT 3% FDR with 25 variables and 30 variables is similar at 42.5%, the test/train/OOT balance is better achieved with 25 variables.

6.2 Neural Networks

Working:

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns.

Deep learning maps inputs to outputs. It finds correlations. It is known as a “universal approximator”, because it can learn to approximate an unknown function f(x) = y between any input x and any output y, assuming they are related at all (by correlation or causation, for example).

A typical neural net consists of an input layer, some number of hidden layers and an output layer.

- All the independent variables (x’s) form the input layer.
- The dependent variable y is the output layer.
- The hidden layer is a set of nodes or “neurons”

Each node in the hidden layer receives weighted signals from all the nodes in the previous layer and does a transform on this linear combination of signals.

The transform/activation function can be a logistic function (sigmoid), or something else.

Parameters: # hidden layers, # nodes/layer, transform/activation function

Implementation:

We created the Neural Nets using the scikit-learn library and MLP Classifier function. We used the following parameters:

- # hidden layers: 1
- # nodes/layer: 6
- transformation/activation function: Rectified Linear Unit

We used the top 30 variables and ran 10 iterations of the model.

Results:

The average OOT FDR with 30 variables is 33.2%.

6.3 Boosted Tree Model

Working:

A boosted tree is a series of weak decision tree learners to result in a strong learner. Each tree attempts to minimize the errors of the previous tree. This makes boosted trees a highly efficient and accurate model. Each new tree fits to the residuals from the previous step, so the overall model improves. A loss function is used to detect the residuals. In this model, trees are added sequentially hence the model learns slowly. Each added tree modifies the overall model. The magnitude of this modification is controlled by learning rate, that implies, how fast the model learns. The lower the learning rate, the slower the model learns. The advantage of slower learning rate is that the model becomes more robust and generalized. However, slow learning takes more time to train the model and we need more trees to train the model. But adding too many trees can take us back to the root problem, overfitting. So, the number of trees in a boosted tree model can be decided using a cross-validation approach.

Implementation:

In the current implementation of boosted tree, we have utilized Catboost gradient boosting model which leverages gradient descent method to optimize the loss function.

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. We have used gradient descent to update the parameters of this model.

Next, we used cross-validation to run 10 iterations of the model with each set of variables. Then we average the 3% FDR obtained from each set of over the 10 iterations to get the overall result. We repeat this over top 30, 28, 26, 24, 22 and 20 variables models and select the best one. We observed model with 30 variables gave us best results in terms of balance between train/test/OOT FDR.

In the following step we aimed to tune the hyperparameters of the model with 30 variables. So, we used Bayesian Hyperparameter Optimization technique as described below.

Bayesian Hyperparameter Optimization is a probabilistic model based approach which finds a tuple of hyperparameters that yields an optimal model to minimize a predefined loss function on a given independent data.

We had set hyperparameter optimization technique to use 1000 trials to find the best hyperparameters for our 30 variables model and the best hyperparameters returned were 500 trees, depth of 6 and a learning rate of 0.47.

Results:

Average FDR is 48.3% on OOT data for 30 variables selection and the best set of hyperparameters mentioned above.

6.4 Random Forest

Working:

A random forest is an ensemble of strong learners, each is a full model to predict the output. The final output is an average or vote across all the strong models.

A large number of relatively uncorrelated models operating as a committee will outperform any constituent models. It is important for the trees to have very little correlation between them, so they are protected from each other's errors.

Random Forest achieves this low correlation in two ways:

- Random forest uses **bagging**- allowing each individual tree to randomly sample from the dataset with replacement, resulting in different trees. The number of predictors considered at each split is approximately equal to the square root of the total number of predictors
- **Feature randomness**- In a normal decision tree, when it is time to split a node, we consider every possible feature and pick the one that produces the most separation between the observations in the left node vs. those in the right node. In contrast, each tree in a random forest can pick only from a random subset of features. This forces even more variation amongst the trees in the model and ultimately results in lower correlation across trees and more

Implementation:

We implemented XGBoost Random Forest Classifier. XGBoost stands for "Extreme Gradient Boosting". XGBoost Random forests use the same model representation and inference, as gradient-boosted decision trees, but a different training algorithm. One can use XGBoost to train a standalone random forest or use random forest as a base model for gradient boosting. Here we focused on training standalone random forest.

We used cross-validation to run 10 iterations of the model with each set of variables. Then we average the 3% FDR obtained from each set of over the 10 iterations to get the overall result. We repeat this over top 30, 28, 26, 24, 22 and 20 variables and select the best one. Once again, we observed model with 30 variables gave us best results in terms of balance between train/test/OOT FDR.

Finally, we optimized hyperparameters for the 30 variables model by implementing Bayesian Hyperparameter Optimization technique using 1000 trials. The best set of hyperparameters returned were 285 trees, max_depth of 10 and a learning rate of 1.

Results:

Average FDR is 59.7% on OOT data for 30 variables selection and the best set of hyperparameters mentioned above.

6.5 FDR (TRAIN, TEST and OOT) OF DIFFERENT MODELS

FDR-Logistic Regression				
Model	**Train**	**Test**	**OOT**	**# Variables**
1	54.4%	54.3%	42.6%	30
2	**56.3%**	**56.2%**	**42.5%**	**25**
3	56.4%	55.2%	42.4%	20
4	54.4%	55.4%	42.3%	16
5	54.2%	54.2%	42.3%	10
6	56.1%	55.1%	42.2%	6

FDR- Neural Net							
Model	**Layer**	**Nodes**	**Epoch**	**Train**	**Test**	**OOT**	**# Variables**
1	**1**	**6**	**20**	**71.7%**	**68.9%**	**33.2%**	**30**
2	1	10	50	70.5%	67.2%	29.9%	30
3	1	30	50	71.3%	66.1%	29.1%	30
4	1	40	20	69.9%	67.9%	30.9%	30
5	1	10	50	70.7%	67.6%	29.4%	30
6	1	30	50	71.6%	68.4%	29.2%	30

FDR-Boosted Trees							
Model	**# Trees**	**Depth**	**Learning Rate**	**Train**	**Test**	**OOT**	**# Variables**
1	**500**	**6**	**0.47**	**91.3%**	**60.6%**	**48.3%**	**30**
2	500	6	0.47	91.3%	59.3%	48.1%	28
3	500	6	0.47	89.7%	58.2%	47.5%	26
4	500	6	0.47	89.2%	57.4%	47.8%	24
5	500	6	0.47	89.2%	57.2%	47.2%	22
6	500	6	0.47	88.8%	56.1%	46.4%	20

FDR-Random Forest							
Model	**# Trees**	**Max Depth**	**Learning Rate**	**Train**	**Test**	**OOT**	**# Variables**
1	**285**	**10**	**1**	**86.7%**	**82.9%**	**59.7%**	**30**
2	285	10	1	86.6%	79.2%	59.5%	28
3	285	10	1	85.9%	78.9%	58.4%	26
4	285	10	1	85.3%	79.4%	59.1%	24
5	285	10	1	85.2%	77.1%	58.3%	22
6	285	10	1	84.9%	78.7%	58.9%	20

7.0 Results

The best performing algorithm is Random Forest (**FDR 59.7%**) with the following parameters:

- Max Depth = 10
- Number of trees = 285
- 30 variables

The tables below summarize the cumulative Good, Cumulative Bads, % Good, % Bad (FDR), KS and FPR for all three populations - training, testing, and Validation (OOT)

Training Data:

Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods2	% Bads (FDR)	KS	FPR
1	645	123	522	19.07%	80.93%	645	123	522	0.19%	77.68%	77.49%	0.24
2	645	605	40	93.80%	6.20%	1290	728	562	1.14%	83.63%	82.49%	1.30
3	645	640	5	99.22%	0.78%	1935	1368	567	2.14%	84.38%	82.23%	2.41
4	645	643	2	99.69%	0.31%	2580	2011	569	3.15%	84.67%	81.52%	3.53
5	645	640	5	99.22%	0.78%	3225	2651	574	4.15%	85.42%	81.26%	4.62
6	645	638	7	98.91%	1.09%	3870	3289	581	5.15%	86.46%	81.31%	5.66
7	645	644	1	99.84%	0.16%	4515	3933	582	6.16%	86.61%	80.45%	6.76
8	645	641	4	99.38%	0.62%	5160	4574	586	7.17%	87.20%	80.04%	7.81
9	645	641	4	99.38%	0.62%	5805	5215	590	8.17%	87.80%	79.63%	8.84
10	645	642	3	99.53%	0.47%	6450	5857	593	9.18%	88.24%	79.07%	9.88
11	645	637	8	98.76%	1.24%	7095	6494	601	10.17%	89.43%	79.26%	10.81
12	645	639	6	99.07%	0.93%	7740	7133	607	11.17%	90.33%	79.15%	11.75
13	645	635	10	98.45%	1.55%	8385	7768	617	12.17%	91.82%	79.65%	12.59
14	645	642	3	99.53%	0.47%	9030	8410	620	13.18%	92.26%	79.09%	13.56
15	645	645	0	100.00%	0.00%	9675	9055	620	14.19%	92.26%	78.08%	14.60
16	645	644	1	99.84%	0.16%	10320	9699	621	15.19%	92.41%	77.22%	15.62
17	645	643	2	99.69%	0.31%	10965	10342	623	16.20%	92.71%	76.51%	16.60
18	645	639	6	99.07%	0.93%	11610	10981	629	17.20%	93.60%	76.40%	17.46
19	645	640	5	99.22%	0.78%	12255	11621	634	18.21%	94.35%	76.14%	18.33
20	645	642	3	99.53%	0.47%	12900	12263	637	19.21%	94.79%	75.58%	19.25

Testing Data:

Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods2	% Bads (FDR)	KS	FPR
1	161	36	125	22.36%	77.64%	161	36	125	0.23%	63.78%	63.55%	0.29
2	161	140	21	86.96%	13.04%	322	176	146	1.10%	74.49%	73.39%	1.21
3	161	151	10	93.79%	6.21%	483	327	156	2.05%	79.59%	77.54%	2.10
4	161	158	3	98.14%	1.86%	644	485	159	3.04%	81.12%	78.08%	3.05
5	161	161	0	100.00%	0.00%	805	646	159	4.05%	81.12%	77.07%	4.06
6	161	161	0	100.00%	0.00%	966	807	159	5.07%	81.12%	76.06%	5.08
7	161	159	2	98.76%	1.24%	1127	966	161	6.06%	82.14%	76.08%	6.00
8	161	158	3	98.14%	1.86%	1288	1124	164	7.06%	83.67%	76.62%	6.85
9	161	160	1	99.38%	0.62%	1449	1284	165	8.06%	84.18%	76.12%	7.78
10	161	159	2	98.76%	1.24%	1610	1443	167	9.06%	85.20%	76.15%	8.64
11	161	160	1	99.38%	0.62%	1771	1603	168	10.06%	85.71%	75.65%	9.54
12	161	158	3	98.14%	1.86%	1932	1761	171	11.05%	87.24%	76.19%	10.30
13	161	161	0	100.00%	0.00%	2093	1922	171	12.06%	87.24%	75.18%	11.24
14	161	158	3	98.14%	1.86%	2254	2080	174	13.06%	88.78%	75.72%	11.95
15	161	161	0	100.00%	0.00%	2415	2241	174	14.07%	88.78%	74.71%	12.88
16	161	161	0	100.00%	0.00%	2576	2402	174	15.08%	88.78%	73.70%	13.80
17	161	159	2	98.76%	1.24%	2737	2561	176	16.08%	89.80%	73.72%	14.55
18	161	161	0	100.00%	0.00%	2898	2722	176	17.09%	89.80%	72.71%	15.47
19	161	161	0	100.00%	0.00%	3059	2883	176	18.10%	89.80%	71.70%	16.38
20	161	158	3	98.14%	1.86%	3220	3041	179	19.09%	91.33%	72.24%	16.99

Out of Time (OOT) Data:

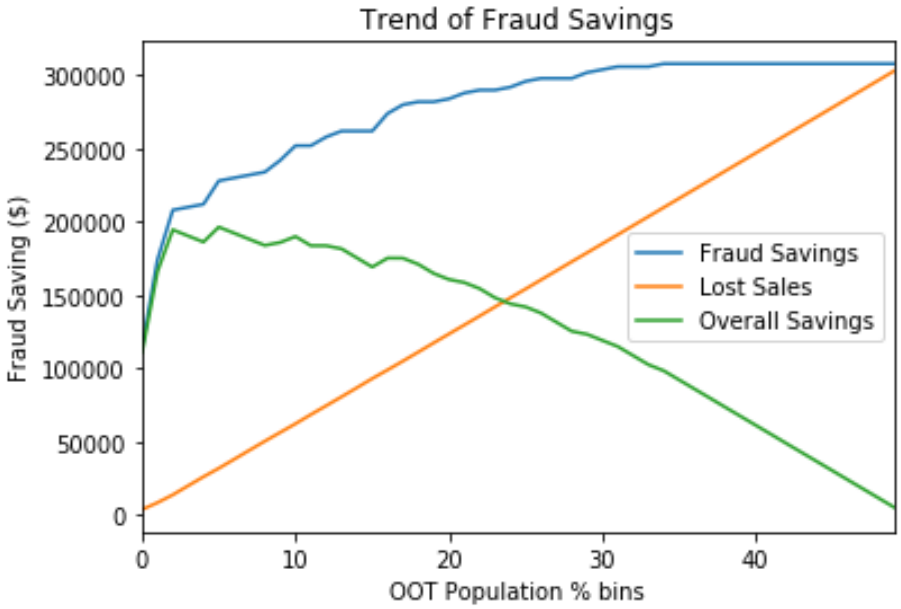
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods2	% Bads (FDR)	KS	FPR
1	124	62	62	50.00%	50.00%	124	62	62	0.51%	34.64%	34.13%	1.00
2	124	99	25	79.84%	20.16%	248	161	87	1.31%	48.60%	47.29%	1.85
3	124	105	19	84.68%	15.32%	372	266	106	2.17%	59.22%	57.05%	2.51
4	124	124	0	100.00%	0.00%	496	390	106	3.18%	59.22%	56.03%	3.68
5	124	124	0	100.00%	0.00%	620	514	106	4.20%	59.22%	55.02%	4.85
6	124	123	1	99.19%	0.81%	744	637	107	5.20%	59.78%	54.58%	5.95
7	124	123	1	99.19%	0.81%	868	760	108	6.21%	60.34%	54.13%	7.04
8	124	117	7	94.35%	5.65%	992	877	115	7.16%	64.25%	57.09%	7.63
9	124	123	1	99.19%	0.81%	1116	1000	116	8.16%	64.80%	56.64%	8.62
10	124	120	4	96.77%	3.23%	1240	1120	120	9.14%	67.04%	57.89%	9.33
11	124	119	5	95.97%	4.03%	1364	1239	125	10.12%	69.83%	59.72%	9.91
12	124	124	0	100.00%	0.00%	1488	1363	125	11.13%	69.83%	58.70%	10.90
13	124	120	4	96.77%	3.23%	1612	1483	129	12.11%	72.07%	59.96%	11.50
14	124	122	2	98.39%	1.61%	1736	1605	131	13.10%	73.18%	60.08%	12.25
15	124	120	4	96.77%	3.23%	1860	1725	135	14.08%	75.42%	61.34%	12.78
16	124	124	0	100.00%	0.00%	1984	1849	135	15.10%	75.42%	60.32%	13.70
17	124	123	1	99.19%	0.81%	2108	1972	136	16.10%	75.98%	59.88%	14.50
18	124	124	0	100.00%	0.00%	2232	2096	136	17.11%	75.98%	58.86%	15.41
19	124	124	0	100.00%	0.00%	2356	2220	136	18.13%	75.98%	57.85%	16.32
20	124	123	1	99.19%	0.81%	2480	2343	137	19.13%	76.54%	57.41%	17.10

Fraud Savings Table and Plot:

The overall savings in the following table are calculated by assuming a $2000 gain for every fraud caught and a $50 loss for every false positive:

Pop Bin %	# Records	# Goods	# Bads	Fraud Saving	Lost Sales	Overall Savings
1	125	68	57	114000	3400	110600
2	124	94	30	174000	8100	165900
3	124	107	17	208000	13450	194550
4	125	124	1	210000	19650	190350
5	124	123	1	212000	25800	186200
6	124	116	8	228000	31600	196400
7	124	123	1	230000	37750	192250
8	125	124	1	232000	43950	188050
9	124	123	1	234000	50100	183900
10	124	120	4	242000	56100	185900

The following plot shows the variation of Fraud Savings (in $) as we go deeper in our data



As seen in the plot and the table, since the fraud savings at 3% and 6% population (OOT) are equivalent, the final threshold was at **3%** as the goal is to get as small a cutoff as possible while maximizing savings.

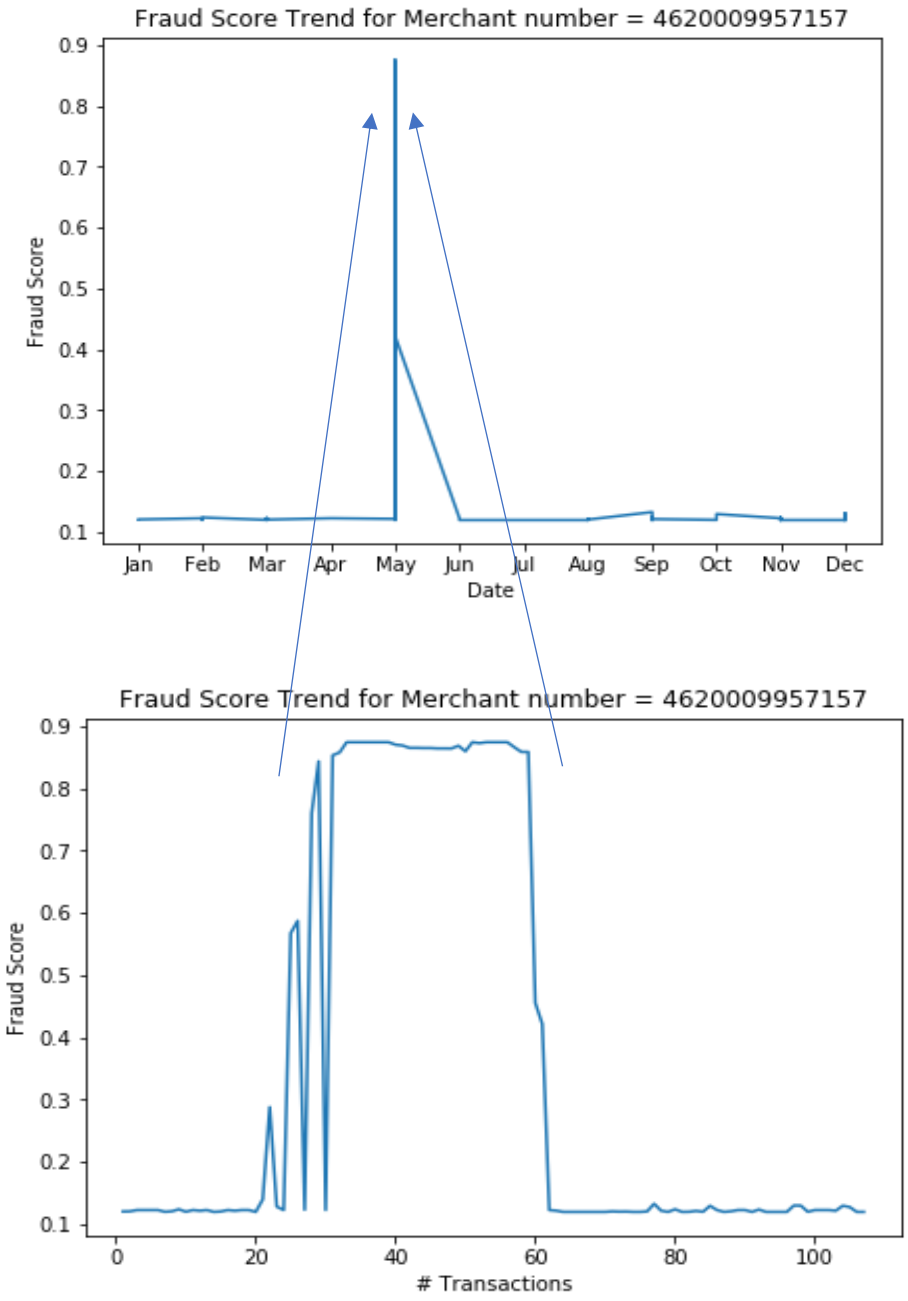
Fraud Savings per 1000 transactions: $15,655.

Dynamic Plots:

Dynamic Plot for Merchant Number = 4620009957157

45 Transactions were made in the month of May, out of which 39 Transactions were within 3 days (14th-16th May)

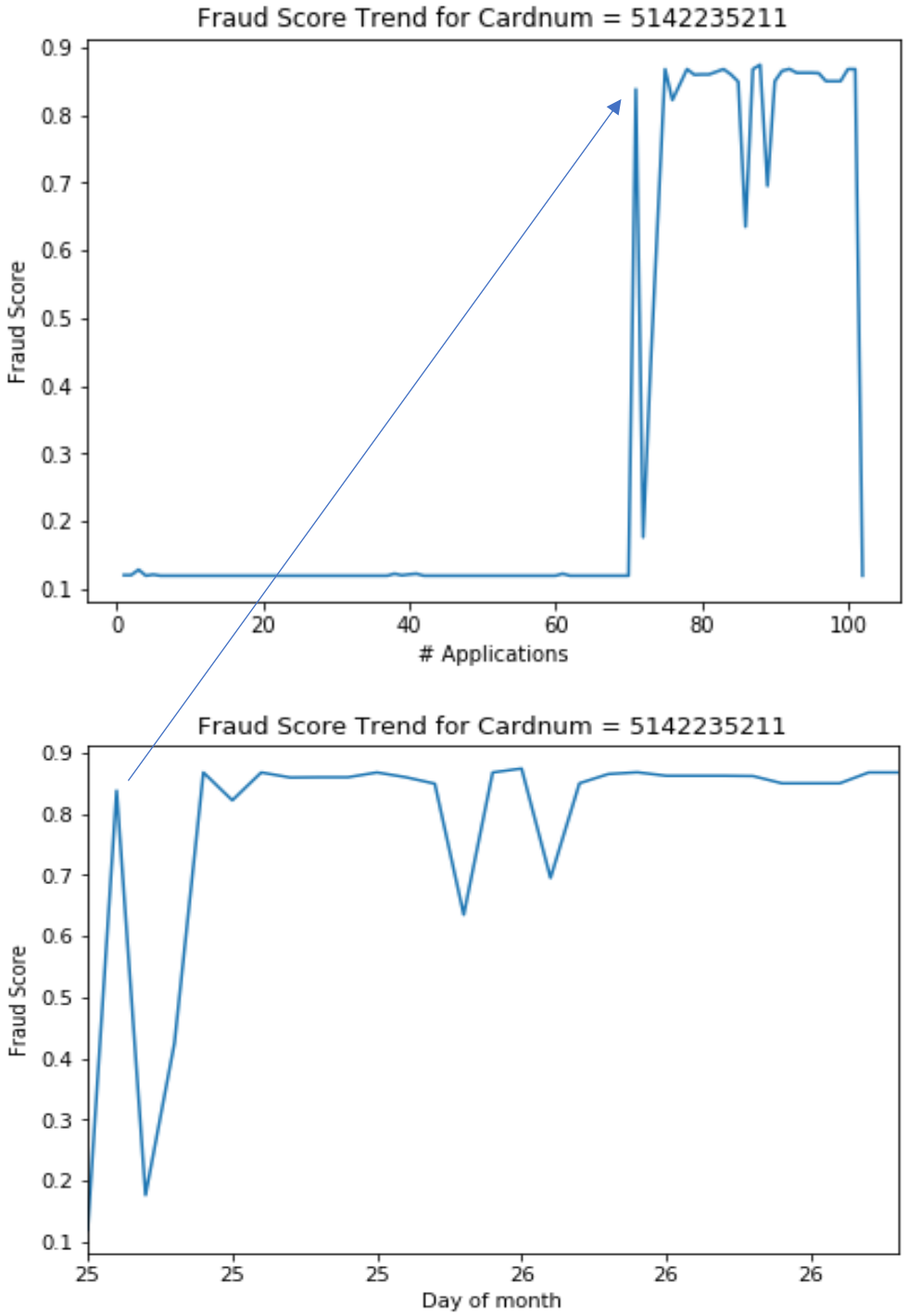
This abnormal behavior led to a spike in the fraud score as you can see in the following two images:



Dynamic Plot for Card Number = 5142235221

32 transactions were made from this card within 2 days, in the month of November.

A sharp increase in predicted Fraud Score was observed.


Fraud Score Trend for Cardnum = 5142235211
Fraud Score
Applications
Fraud Score Trend for Cardnum = 5142235211
Fraud Score
Day of month

8.0 CONCLUSION

In this project, we created a classification model in order to assign a real-time fraud score to credit card transactions. To achieve this, we trained and tested our model using credit card transactions data from the year 2010. An extensive process was followed, which included data cleaning, feature engineering, feature selection, building classification models and determining the optimal fraud score cutoff to maximize savings.

Python libraries and excel were used as effective tools during the project. After performing data cleaning, we engineered 325 variables using the fields that were provided in the data and selected the 30 most relevant features using a random forest wrapper. Classification models were built using logistic regression, neural networks, boosted trees and random forest. The performance of each model was evaluated by calculating the average Fraud Detection Rate at 3% of the training, testing and out of time data.

After comparing the performance of all the models, we found that the random forest classifier had the most balanced performance in training, testing and out of time data. The FDR for this model in the top 3% of the three datasets is as follows:

Scenario	Train	Test	OOT
Best	86.9%	82.4%	62.0%
Average	86.7%	82.9%	59.7%

The optimal cutoff point for the fraud score was calculated to be at the top 3% of the OOT population. The total savings using this model in the out of time period of two months was $194550. Therefore, the model can potentially help save more than $2.2 million annually.

Recommendations

We hereby provide some recommendations to improve the process of fraud detection using classification models:

- **Obtain more data to train the models**: Our model has been trained and tested on the transactions data of a single calendar year. Training the model on more data and exposing it to more fraudulent transactions might improve the performance.
- **Consult subject matter experts**: We created over 300 variables by making various combination of fields, using Benford's Law variables, velocity variables and last seen variables. Consulting some experts in the fraud detection domain might help us engineer better variables.
- **Account for seasonality**: The dataset we used does not show a seasonal pattern for credit card fraud. Statistically, credit card fraud increases during certain parts of the year, for example, during Christmas. Obtaining datasets that show such seasonal patterns and training our model to recognize them will improve scalability.

9.0 APPENDIX

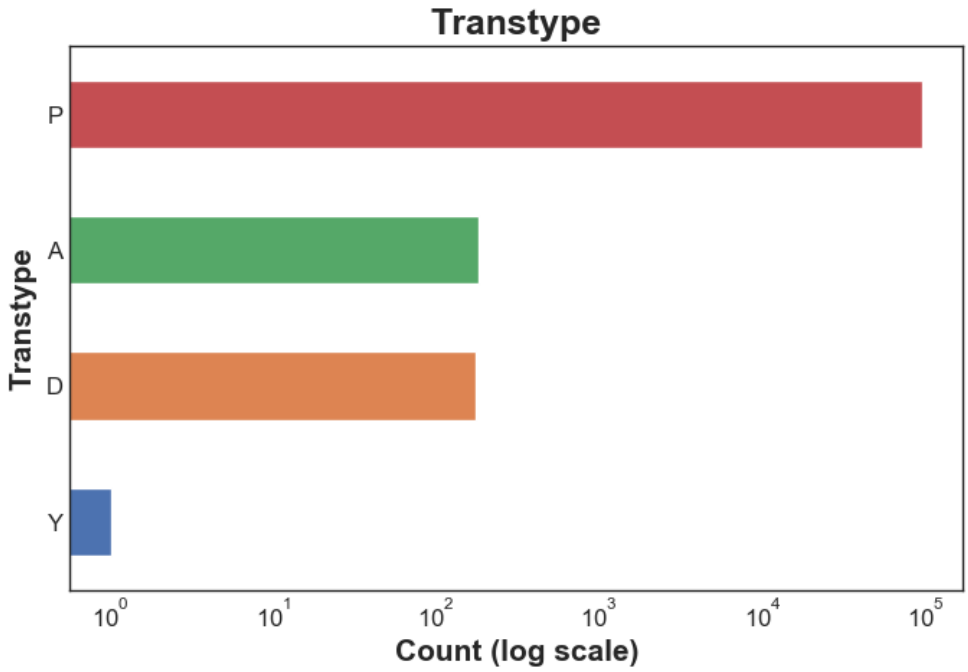
The Appendix includes the Data Quality Report related to the description of the remaining fields of the dataset.

9.1 Field Name: Recnum

Description: This field represents a unique identifier for each record or row in the dataset. Values in this field have a unique number for each of the 96753 rows in the dataset.

9.2 Field Name: Transtype

Description: This field is categorical and contains the type of each transaction corresponding to each record in the dataset. There are 4 types of transactions in the dataset. We note that transtype 'P' is the most common with a frequency of 96398. The bar graph shows count (log scale) of each transtype.



Transtype	Count
P	96398
A	181
D	173
Y	1