

DSO 562: Fraud Analytics

**Supervised Fraud Model on Applications Data**

**Trojan Consulting Team 3**

**Table of Content**

[Executive Summary 3](#_Toc7724503)

[Part I. Data Description 4](#_Toc7724504)

[Data Summary 4](#_Toc7724505)

[Categorical variable 4](#_Toc7724506)

[Key Variables Description 4](#_Toc7724507)

[Part II. Data Cleaning 8](#_Toc7724508)

[Part III. Candidate Variables 8](#_Toc7724509)

[Part IV. Feature Selection 10](#_Toc7724510)

[Part VII. Conclusions 19](#_Toc7724511)

[Part VIII. Future Improvements 19](#_Toc7724512)

[Appendix 21](#_Toc7724513)

**Executive Summary**

**Objective:**

Application fraud is one of the most popular forms of identity fraud. Fraudsters use falsified (either stolen or invented) personal information when applying for credit cards, bank accounts, loans or tax rebates. This fraud costs millions of dollars each year, since the process of tracking them with accuracy, without turning down too many valid customers, is challenging to strike. This report examined an applications dataset to find potential fraud with methods include Feature Selection, Wrapper and Machine Learning Algorithm Modeling. Data is processed and analyzed in Python and R.

**Project Outline:**

The original dataset contains 1,000,000 rows of application records with 9 variables of applicants’ personal information. The general process of analysis step included:

1. **Data cleaning and filing missing values**; We proposed the dataset to optimize the results of the analysis. Although there were no missing values in this dataset, there were multiple frivolous values that were addressed prior to building variables.

2. **Building expert variables and standardizing**; we built many candidate variables and scaled them before utilizing feature selection methods to select the best candidate variables.

3. **Feature selection using KS and FDR**; Both the KS score and the FDR rate help in determining how well candidate variables individually predict fraud. We rank ordered the candidate variables in terms of usefulness for our models.

4. **Applying fraud algorithms**; We used supervised algorithms including a logistic regression, a random forest, AdaBoosting, XGBoosting and Gradient Boosting methods to detect fraud in the application dataset provided.

5. **Recommendations**: Lastly, we will create a threshold for the top 7 percent of applications to reject based on our fraud scoring model to optimize the balance between rejecting legitimate applications and accepting fraudulent ones.

**Part I. Data Description**

**Data Summary**

The dataset contains application information of the New York City. The data is collected to evaluate the possibility of identity fraud. The data has 1,000,000 rows and 10 columns. All fields are categorical.

**Categorical variable**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Missing values | Percentage Populated | Unique Values | Most Common Field Value |
| record | Categorical | 0 | 100.00% | 1000000 | N/A |
| date | Categorical | 0 | 100.00% | 365 | 20160816 |
| ssn | Categorical | 0 | 100.00% | 835819 | 999999999 |
| firstname | Categorical | 0 | 100.00% | 78136 | EAMSTRMT |
| lastname | Categorical | 0 | 100.00% | 177001 | ERJSAXA |
| address | Categorical | 0 | 100.00% | 828774 | 123 MAIN ST |
| zip5 | Categorical | 0 | 100.00% | 26370 | 68138 |
| dob | Categorical | 0 | 100.00% | 42673 | 19070626 |
| homephone | Categorical | 0 | 100.00% | 28244 | 9999999999 |
| fraud\_label | Categorical | 0 | 100.00% | 2 | 0 |

**Key Variables Description**

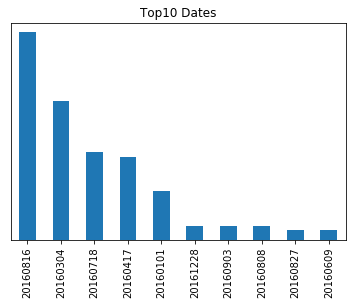
**Field Name: date**

**Description:**

date is a categorical variable representing the date of each transaction.

**Unique Value:**

date has 365 unique values. No missing value exists. The distribution is shown below, top 10 categories are listed below:



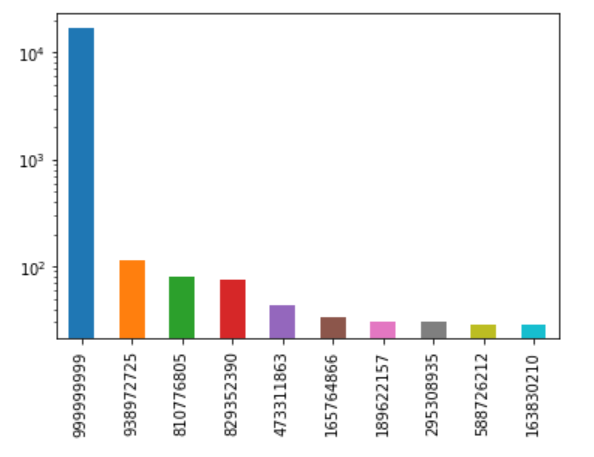
|  |  |
| --- | --- |
| Category | Count |
| 20160816 | 2877 |
| 20160304 | 2861 |
| 20160718 | 2849 |
| 20160417 | 2848 |
| 20160101 | 2840 |
| 20161228 | 2832 |
| 20160903 | 2832 |
| 20160808 | 2832 |
| 20160827 | 2831 |
| 20160609 | 2831 |

**Field Name: ssn (categorical, dtype: int64)**

**Description:**

ssn is a categorical variable representing the social security number of the applicant.

ssn has 835,819 unique values. No missing value exists. The distribution is shown below, top 10 categories are listed below. The following bar chart shows the log count and top 10 ssn.

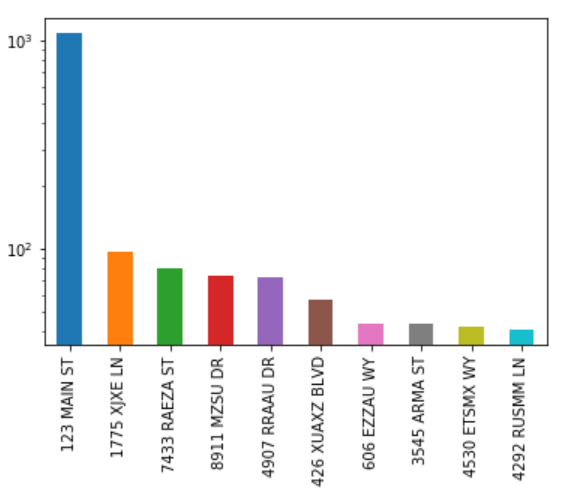


|  |  |
| --- | --- |
| Category | Count |
| 999999999 | 16935 |
| 938972725 | 114 |
| 810776805 | 81 |
| 829352390 | 74 |
| 473311863 | 44 |
| 165764866 | 34 |
| 189622157 | 30 |
| 295308935 | 30 |
| 588726212 | 29 |
| 163830210 | 29 |

**Field Name: address (categorical, dtype: object)**

**Description:**

address is a categorical variable describing the address of the applicant. address has 828,774 unique values. There are no missing values. Below are top 10 categories in descending order. The following bar chart shows the log count and top 10 addresses.

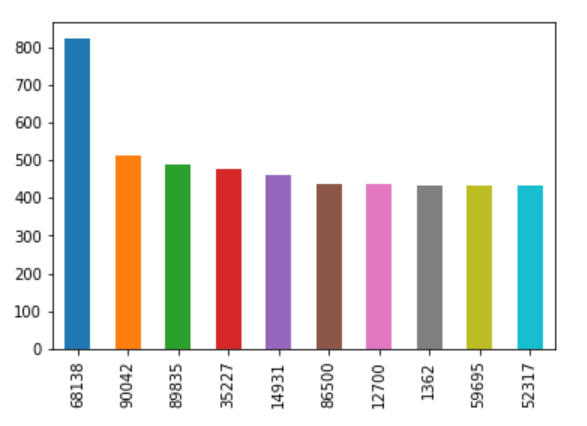


|  |  |
| --- | --- |
| Category | Count |
| 123 MAIN ST | 1079 |
| 1775 XJXE LN | 97 |
| 7433 RAEZA ST | 80 |
| 8911 MZSU DR | 74 |
| 4907 RRAAU DR | 73 |
| 426 XUAXZ BLVD | 57 |
| 606 EZZAU WY | 44 |
| 3545 ARMA ST | 44 |
| 4530 ETSMX WY | 42 |
| 4292 RUSMM LN | 41 |

**Field Name: zip5 (categorical, dtype: int64)**

**Description:**

zip5 is a categorical variable describing the 5-digit zip code of the applicants. zip5 has 26,370 unique values. There are no missing values. Below are top 10 categories. The following bar chart shows the count of top 10 zip codes.

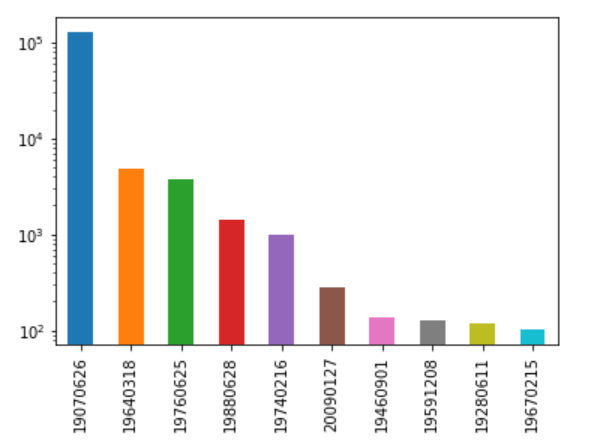


|  |  |
| --- | --- |
| Category | Count |
| 68138 | 823 |
| 90042 | 514 |
| 89835 | 489 |
| 35227 | 478 |
| 14931 | 459 |
| 86500 | 438 |
| 12700 | 436 |
| 1362 | 434 |
| 59695 | 432 |
| 52317 | 432 |

**Field Name: dob (categorical, dtype: int64)**

**Description:**

dob is a categorical variable describing the date of birth of the applicants. dob has 42,673 unique values. There are 0 missing values. Below are top 10 categories in descending order. The following bar chart shows the log count of top 10 dates of birth.



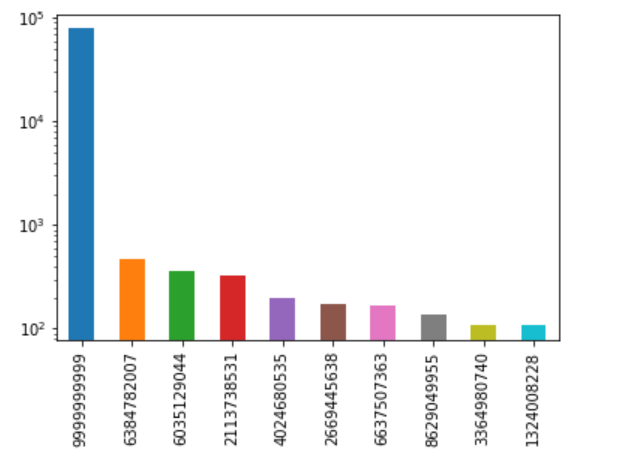
|  |  |
| --- | --- |
| Category | Count |
| 19070626 | 126568 |
| 19640318 | 4818 |
| 19760625 | 3723 |
| 19880628 | 1404 |
| 19740216 | 980 |
| 20090127 | 280 |
| 19460901 | 135 |
| 19591208 | 126 |
| 19280611 | 120 |
| 19670215 | 102 |

**Field Name: homephone**

**Description:**

homephone is a categorical variable describing the home phone number of the applicants.

homephone has 28,244 unique values. There are 0 missing values. Below are top 10 categories.

****

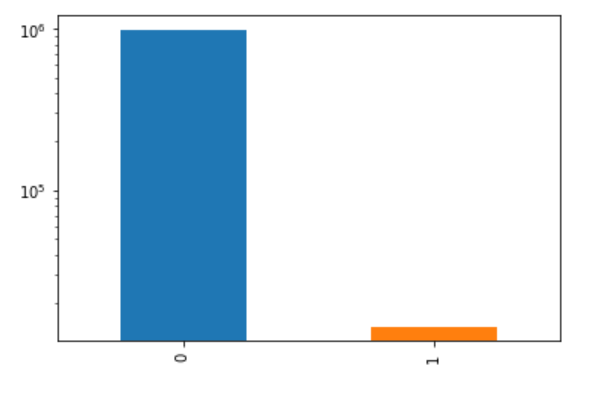
|  |  |
| --- | --- |
| Category | Count |
| 9999999999 | 78512 |
| 6384782007 | 466 |
| 6035129044 | 360 |
| 2113738531 | 331 |
| 4024680535 | 198 |
| 2669445638 | 172 |
| 6637507363 | 169 |
| 8629049955 | 139 |
| 3364980740 | 110 |
| 1324008228 | 108 |

**Field Name: fraud\_label**

**Description:**

fraud\_label is a categorical variable denoting whether an application is fraud.

LTDEPTH has 2 unique values, 0 and 1. There are 985,607 zeros and 14,393 ones. Below is the log distribution of two categories.



**Part II. Data Cleaning**

There are some abnormal values of SSN, Homephone, Address and Date of Birth in the dataset that need to be taken care of as they create unnecessary noise in the data and can hinder with any model trying to make accurate predictions. They must be replaced with a random number. In our case, we have replaced these frivolous values by the corresponding record number. Following are the details of values that have been taken care of:

a) SSN - 999999999

b) HomePhone - 9999999999

c) Address - 123 MAIN ST

d) dob - 19070626

**Part III. Candidate Variables**

To quantify the all the transaction activities and connect these activities to fraud, first we created two new fields:

1. **nameDOB** - combination of ‘firstname’, ‘lastname’, and ‘dob’ fields
2. **Fulladdress** - combination of ‘address’ and ‘zip’ fields

The combination of ‘firstname’, ‘lastname’, and ‘dob’ can be a really good unique identifier of a person rather than only using these entities individually.

Also, there can be several similar addresses, but they can be located at completely different locations, so it’s important to attach ‘Zip code’ with an address value to make it a unique address identifier

**Type of Variables:**

Overall, we created 104 expert variables. Below is the summary of how we went about creating these variables. **For creating ‘Velocity’ and ‘Days since’ variables we have used the whole dataset.**

1. **Velocity variables:** No. of records seen over the past n days, where n = 0, 1, 3, 7, 14, 30. These variables essentially tells us how many times an entity or a combination group is seen over past n days.

**Entity**: [‘ssn’, ‘full address’, ‘nameDOB’, ‘homephone’]

For each entity, we created 6 variables (one for each time stamp). So, overall, we created 24 Entity velocity variables. *For example, ‘ssn0’ means number of applications filed with a SSN on any current date.*

**Combinations:** We took 8 combinations - [‘ssn’, ‘full address’], [‘ssn’, nameDOB’], [‘ssn’, ‘homephone’], [‘ssn’, ‘firstname’], [‘ssn’, ‘lastname’], [‘nameDOB’, ‘homephone’], [‘fulladdress’,’homephone’], [‘fulladdress’, nameDOB’]

For each combination, we created 6 variables (one for each timestamp). So overall, we created 48 Combination velocity variables. *For example, ‘fulladdresshomephone14’ means number of applications filed with a combination of full address and homephone in the last 14 days.*

Hence finally we created 72 Velocity variables

1. **Days since variables -** **For each entity or combination, how many days since we last saw that element.** Wecreated ‘Days since variables’ and in total we have 19 variables

**Entity** - [‘ssn’, ‘full address’, ‘nameDOB’, ‘homephone’]

For each entity, we created 1 ‘Days since’ variables, so overall, we created 4 ‘Days since’ variables. *For example, ‘diff\_date.ssn’ indicates how many days since an application has been filed with a particular SSN.*

**Combinations** - Take 2 combinations out of ['ssn', 'fulladdress', 'firstname', 'lastname', 'nameDOB', 'homephone'], so in total we have 15 combinations.

For each combination, we created 1 ‘Days since’ variable, so overall, we created 15 ‘Days since’ variables. *For example, ‘diff\_date.ssn\_fulladdress’ indicates how many days since an application has been filed with a unique combination of SSN and full address*

Hence, we created 19 ‘Days since’ variables finally.

1. **Risk Variables**

We created 13 risk tables only using the Training and Test data (we took the mean of

Fraud\_Labels for unique entity or combination values mentioned below: (# of bads/ # of

records) for an entity) and then created the following risk variables for both (Train +

Testing) data and OOT data.

*“Risk\_ssn”, “risk\_fulladdress”, “risk\_nameDOB”, “risk\_homephone”, “risk\_ssnfulladdress”, “risk\_ssnnameDOB”, “risk\_ssnhomephone”, “risk\_ssnfirstname”, “risk\_ssnlastname”, “risk\_nameDOBhomephone”, “risk\_fulladdresshomephone” “risk\_fulladdressnameDOB”, “risk\_dayofweek”*

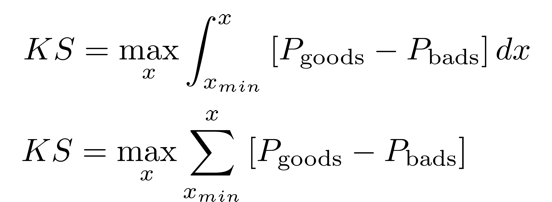
**Part IV. Feature Selection**

1. **Filtering using KS score and Fraud Detection Rate (FDR)**

Before doing KS and FDR, we standardized our candidate variables using Z-scaling

For each of our candidate variables, we calculated Kolmogorov–Smirnov (KS) score and fraud detection rate individually. Both the KS score and the FDR rate will help us determine how well candidate variables individually predict fraud, allowing us to rank order the variables in terms of usefulness for our models.

The KS score is a filter method that helps determine how well a candidate variable separates the goods from the bads, or in this case, the frauds and the not frauds. For each variable, we will use the formula below to calculate a KS score and rank order the variables by the score.



The FDR for each variable be determined at a 3% level. It’s the value representing the % of all frauds caught at a particular examination cutoff. For each variable, we will determine what percent of frauds are captured by the top 3% of the variable and rank order as such.

After getting KS and FDR table for all expert variables, we did the following steps to select around 70 of the top expert variables

1. Sort records by ‘KS’ in a descending order
2. Replace ‘KS’ with the sorted rank order (Highest score getting rank 1 and so on)
3. Sort records by ‘FDR’ in a descending order
4. Replace ‘FDR’ with the sorted rank order (Highest score getting rank 1 and so on)
5. ‘KS’ and ‘FDR’ are on same scale now and we took their average to get ‘Cumulative score’, which is our final importance rank of variables

Though the KS and FDR models prioritized all 13 risk variables on top, but in order to avoid overfitting and try to individually see the impact of each risk variable, we kept the top 4 Risk variables in the filtered list of important 61 variables *(9 risk variables were removed).*

1. **Wrapper Method:**

A wrapper method was implemented to determine the candidate variables that should be used to fit a model. It uses a logic like that of a step-wise approach to determine which variables are valuable for our machine learning model’s prediction. We used the wrapper method to select our best 21 candidate variables for predicting fraud.

**Random Forest Wrapper:**

We used a Random Forest Classifier for feature selection. We performed step forward feature selection using the ‘mlxtend’ library. We started by making two separate columns, one containing all the predictors (candidate variables - in this case we have filtered to 61 out of 104 candidate variables after doing KS and FDR as the Filter method, so we have 61 predictors) and a second column containing the Fraud label. Also, we used the data up to Oct. 31 to perform Feature Selection.

We chose the ‘Number of Estimators’ as 10, essentially the number of decision trees that we want to create during our Feature selection Process. In the feature selection step, we set the ‘Cross Validation’ count as 3 for ‘mlxtend’ that essentially splits the data into 3 parts and choose 1 part as test and other two as the training data. This process continues until each record value is considered as a training and test set during the model training. The model that ran for 2 hours to identify 20 most important variables included all the 4 risk variables [risk\_fulladdressnameDOB’, ‘risk\_nameDOB’, ‘risk\_ssnfulladdress’, ‘risk\_ssnhomephone’] with very high % importance among the selected features.

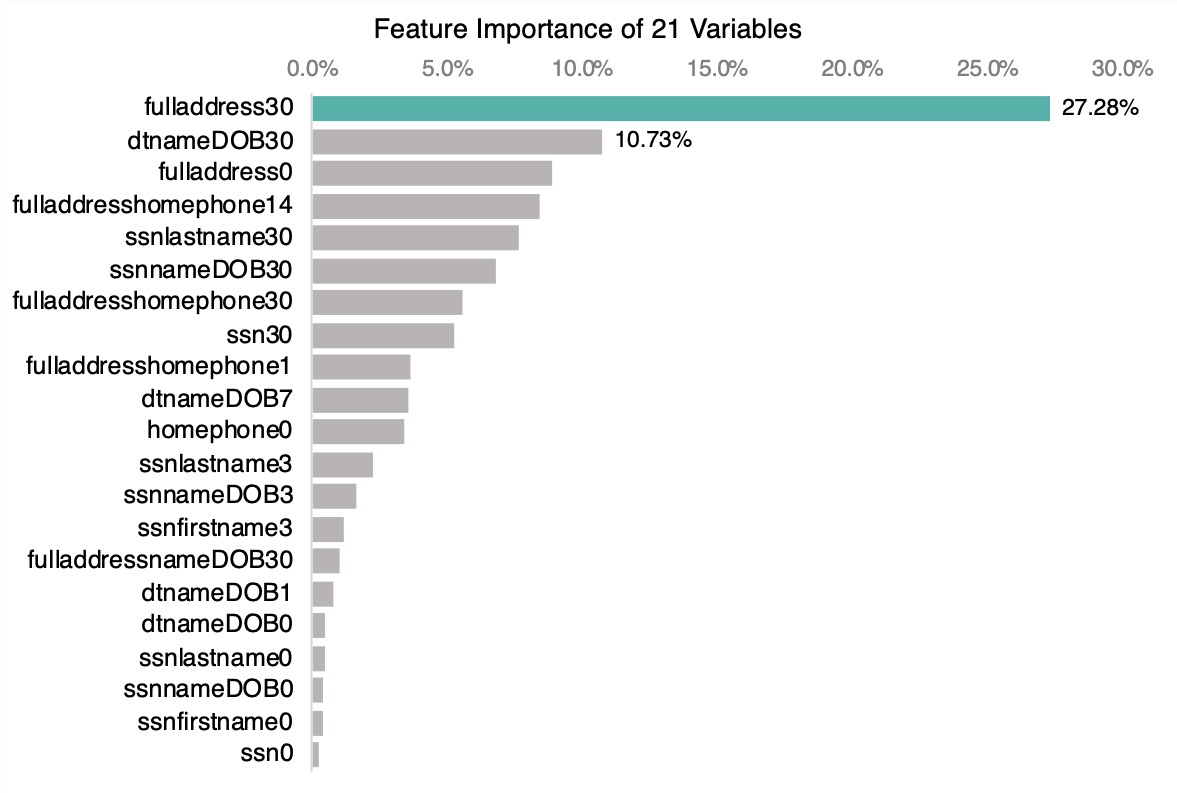
After this to see how these features perform in doing Fraud\_label predictions, we perform predictions using Random Forest Classifier for Train, Test, and OOT data sets.

Following table shows the overfitting issue with including the risk variables in the final set of variables:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **FDR - Random Forest** | | |  |  |
| **# of trees** | **Train** | **Test** | **OOT** | **# of risk variables** | **# of variables** |
| 400 | **100%** | **100%** | **37.20%** | 4 | 20 |
| 400 | **100%** | **100%** | **38.60%** | 1 | 17 |
| 400 | **100%** | **100%** | 52.68% | 0 | 16 |

Therefore, we started removing the risk variables one-by-one and we saw considerable improvement in the OOT FDR, but still performance on Train/ Test data was high. Also, we were left with only 16 variables and since a random forest classifier takes around sqrt (p) variables *(where p is the total number of predictors)* while splitting each individual tree, so a small count of variables might not be an ideal number to build a decision tree. Hence, we performed the feature selection again using the Random Forest Wrapper with risk variables completely removed after the KS and FDR approach and we chose 21 out of the 57 variables *(this took around 3 hours to run).* Detailed performance of different models will be in Results section.

The below graph shows the list of 21 variables with feature % importance that we finally used for building our different models:



**Part V. Model Algorithms**

We attempted to try five model: Logistic Regression, Random Forest, AdaBoost, Two Gradient Boosting models and Support Vector Machine while the SVM model was too computationally costly so we run output for other four models.

**a) Logistic Regression**

A multiple logistic regression uses multiple variables to predict the likelihood of a single binary, target variable. The model creates coefficients for each of the predictor variables using a least squares approach.

We ran a logistic regression using different combinations of our identified 21 wrapper variables (25 minus the 4 risk variables). Although we used the wrapper to identify the top 21 variables, we also needed to use a different tool to identify smaller combinations of variables that would perform best.

We used recursive feature elimination to find the most effective, smaller, combinations of variables to try models of sizes 1-21. The RFE recursively removes attributes and builds a model on the attributes that remain and computes which combinations of attributes contribute the most to predicting the target. After running the RFE, we identified the smaller combinations have used them to predict fraud.

Our model’s top performance occurred with a combination of size 14. The model’s fraud detection rate at 3% threshold was 50.02% for testing, 50.72% for training and 48.52% for the holdout sample. This model would serve as our baseline for to improve upon with more advanced algorithms.

**b) Random Forest Classifier**

While making predictions using random forests as we did in bagging, we build several decision trees on bootstrapped training samples. However, in random forests, when building these decision trees, each time a split in a tree is considered, a random sample of predictors is chosen as split candidates from the full set of predictors. The number of predictors considered at each split is approximately equal to the square root of the total number of predictors.

In other words, in building a random forest, at each split in the tree, the algorithm is not allowed to consider most of the available predictors. A main disadvantage with bagging technique is that if predictors are highly correlated, it will lead to split due to strongest predictor at top and hence all bagged trees will have the tendency to look similar. Hence predictions of response variable from bagging will be close to each other and average of these correlated quantities will not lead to enough reduction in variance.

Random forests take care of this situation by forcing each split to consider a subset of predictors and this helps to reduce the effect of highly correlated predictors. On a long run, this will help to reduce variance when we take average of predicted values.

We used the RandomForestClassifier package from the library sklearn to make the Random Forest model on our reduced set of variables. We used the number of estimators as 50 i.e. no. of trees and then we trained our model on training data. Then we predicted the probability of Fraud over training, test and OOT (validation data).

**c) Adaboost**

In this project, we used AdaBoost for classifying the fraudulent case. AdaBoost is sequentially building a series of ‘weak’ classifier and then combine the results based on their performance. The algorithm is as follows:

1. At first, it assigns an equal “weight” to each training record, which determines the probability that each record will be selected in the training set. Records with higher weights are more likely to be included in the training set. After training a classifier, AdaBoost increases the weight on the misclassified records so that these records and the next classifier will put more emphasis in predicting those misclassified cases right next time.

2. AdaBoost assigns a “weight” to each data point, which determines the probability that each sample should be selected in the training set. Data points with higher weights are more likely to be included in the training set, and vice versa. After training a classifier, AdaBoost increases the weight on the data points that are not predicted incorrectly so that these data points will take up a larger proportion of the next classification training set.

One main reason for having weak learner and built trees sequentially is to prevent overfitting and increase our computation efficiency.

**d) Gradient and XGBoost**

Gradient boosting is the process of training many smaller models additively. Although it is very similar to the AdaBoost model, it differs in that the gradient boost uses gradients in its loss function to train itself over a series of several smaller models. A loss function is a representation of the model’s error in fraud prediction. As a result, the model slowly progresses towards an optimal result over time until it reaches its peak prediction accuracy. Our gradient boosting model boasted an OOT accuracy of 52.56%.

XGBoost is an optimized gradient boost model that is regularly the champion of prediction competitions. The model prioritizes harnessing the complexity of the gradient boost model but on a more scalable level computationally, optimizing results. Our model had a subsample value of 0.8, 0.5, depth of 15, nround of 50 and 21 variables. Our XGBoost model was our top performing model, boasting an OOT accuracy of 52.66%.

**FDR (Train, Test and OOT) of different models:**

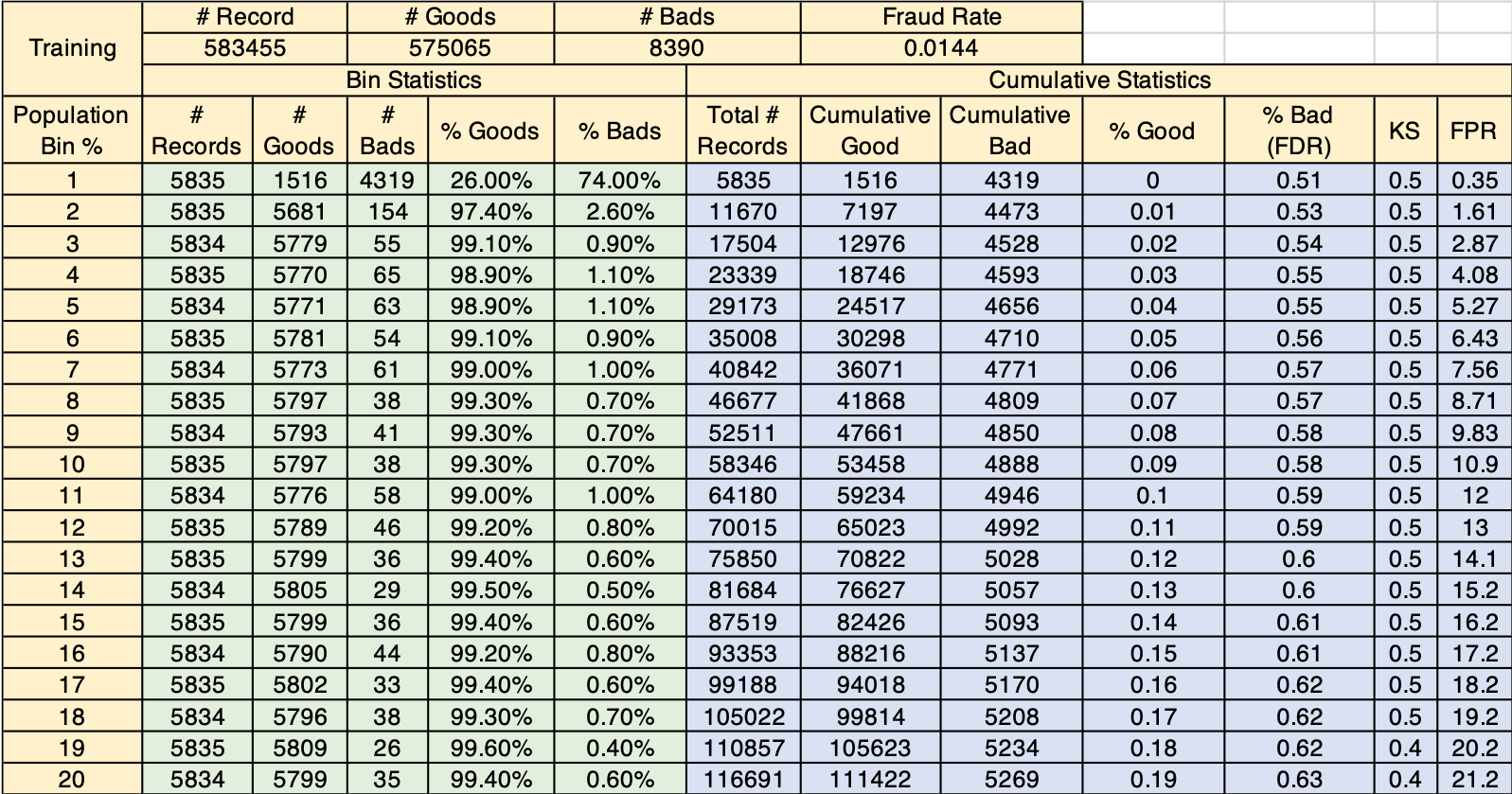


**Part VI. Results**

Our best performing algorithm is XGboost model and we have generated cumulative Good, Bads, % Good, % Bad (FDR), KS and FPR for all three populations (training, testing, and Validation (OOT), and the fraud savings plot.

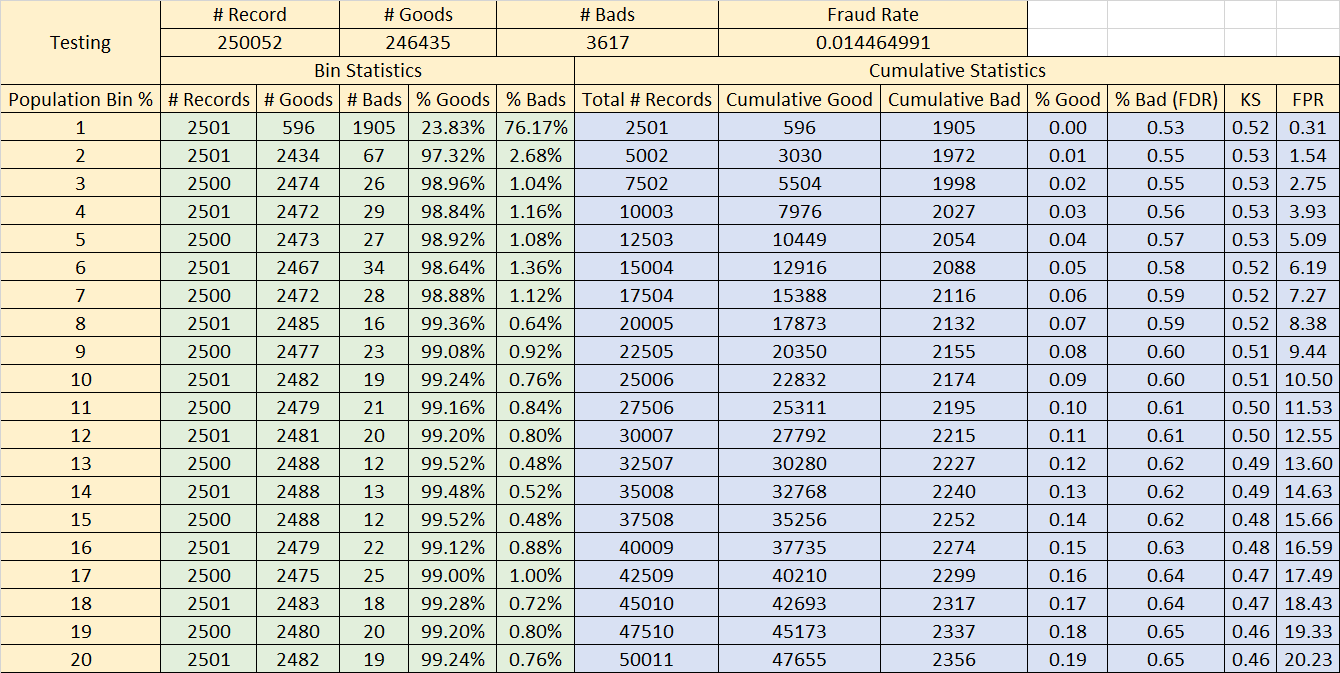
**1.Training Data**

* **Total no. of records in training set – 583,455**
* **Total no. of original Bads – 8,390**



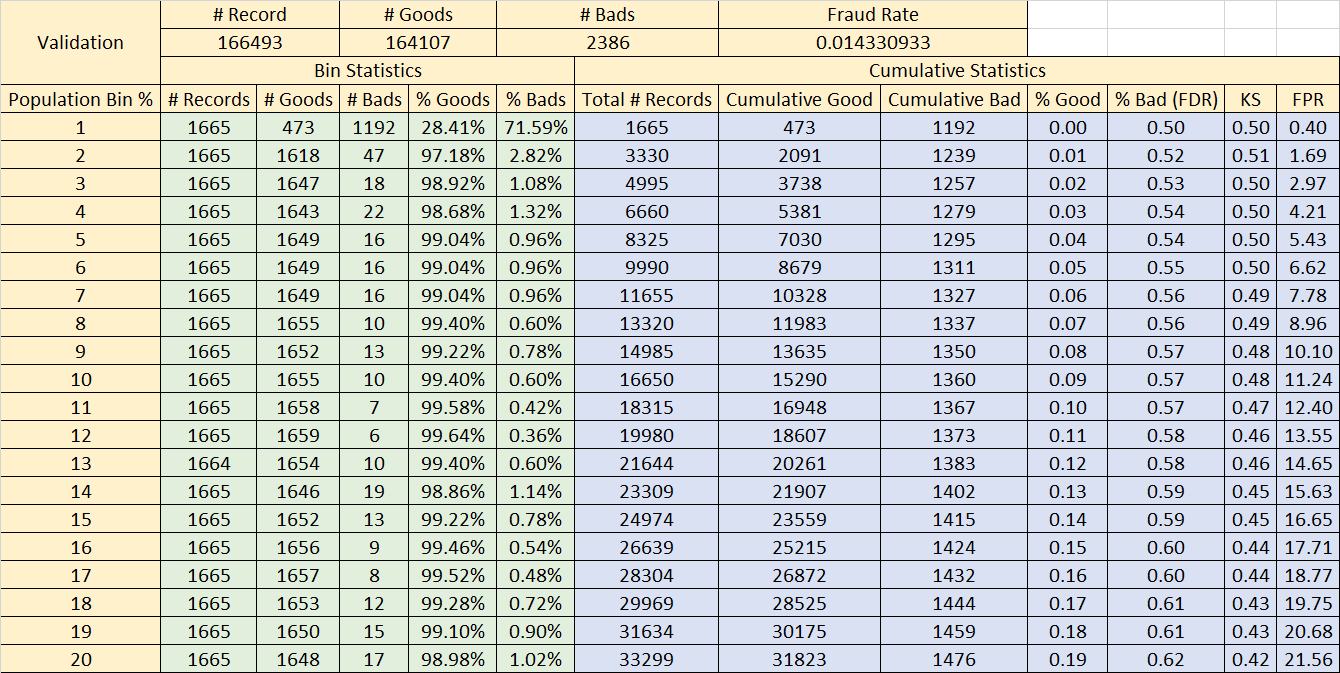
**2.Test Data**

* **Total no. of records in the testing set – 250,052**
* **Total no. of original Bads – 3,617**

****

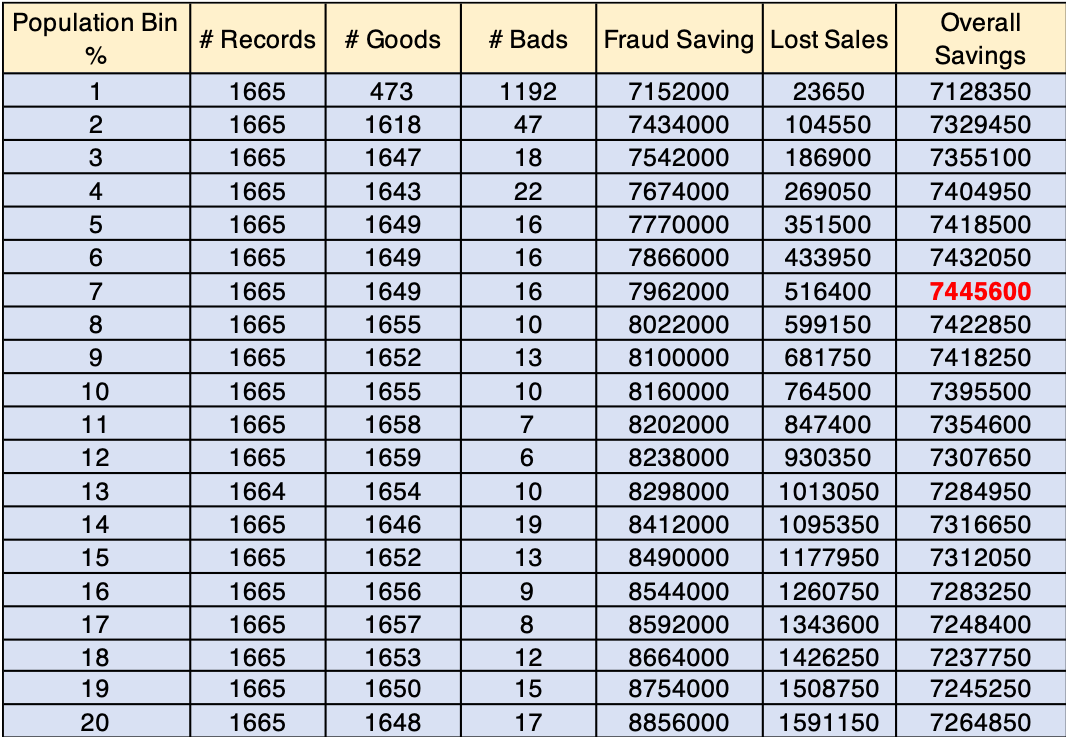
**3. Validation Data**

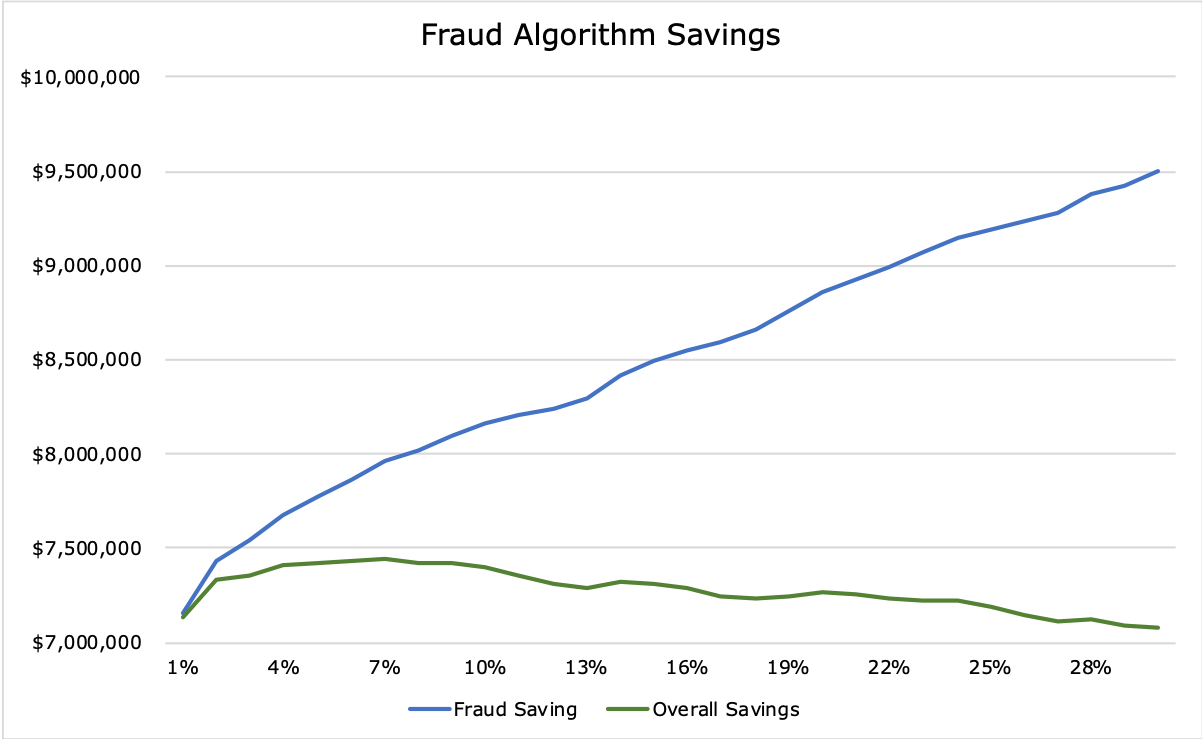
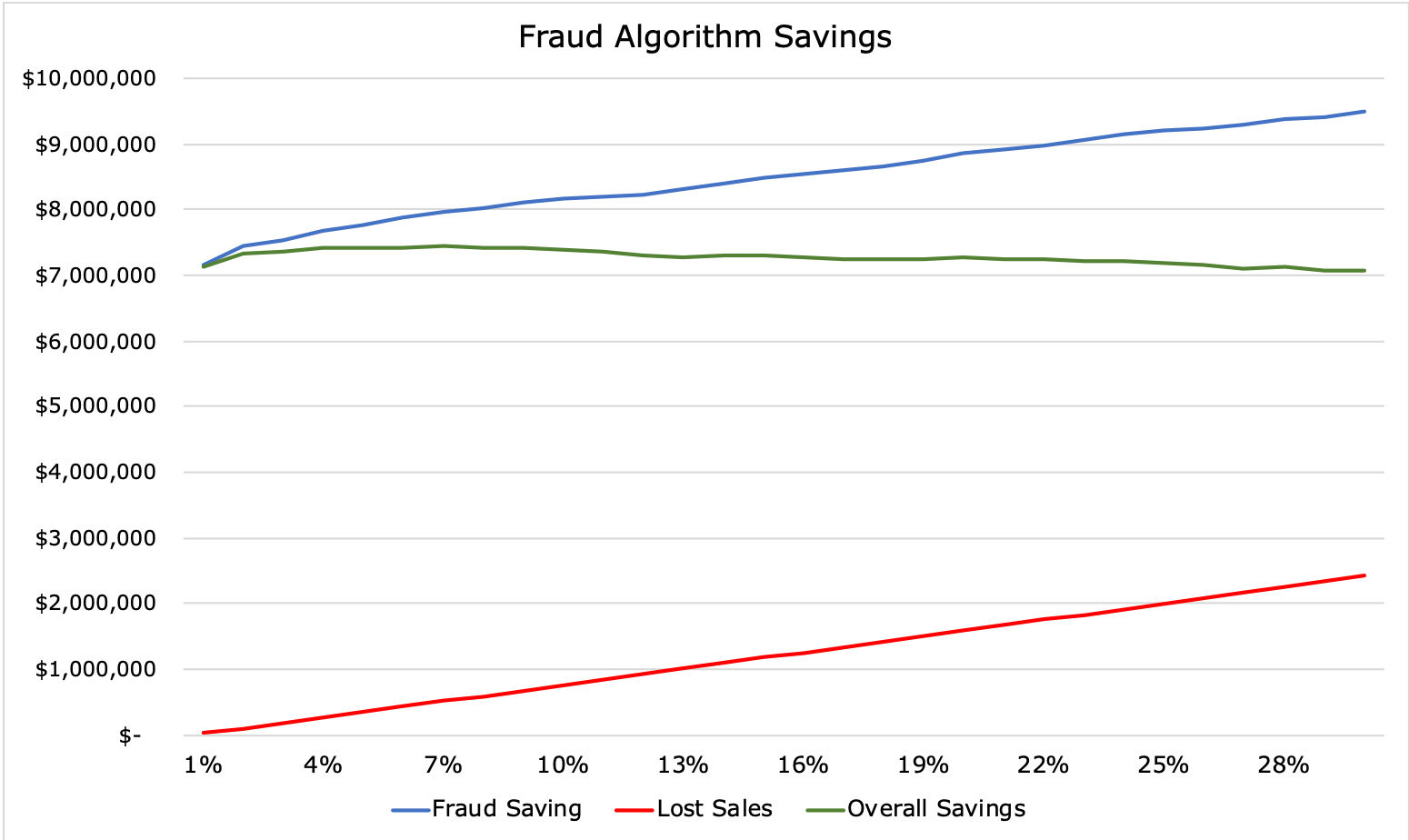
* **Total no. of records in validation set – 166,493**
* **Total no. of original Bads – 2,386**

****

**4. Fraud Saving Table and Plot for the OOT (Validation Data)**

The following plot shows the variation of Fraud Savings (in $) as we go deeper in our data (which is sorted by the Predicted Fraud Label probabilities):

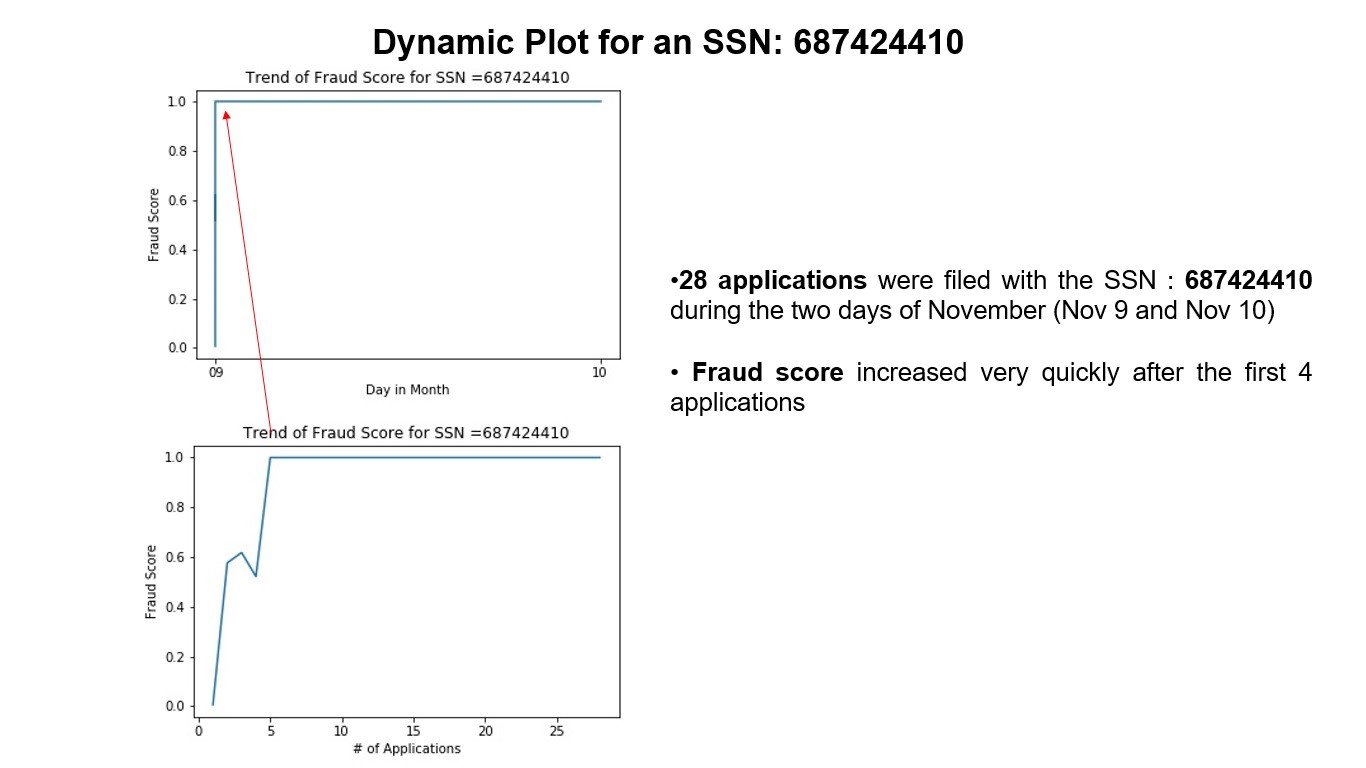




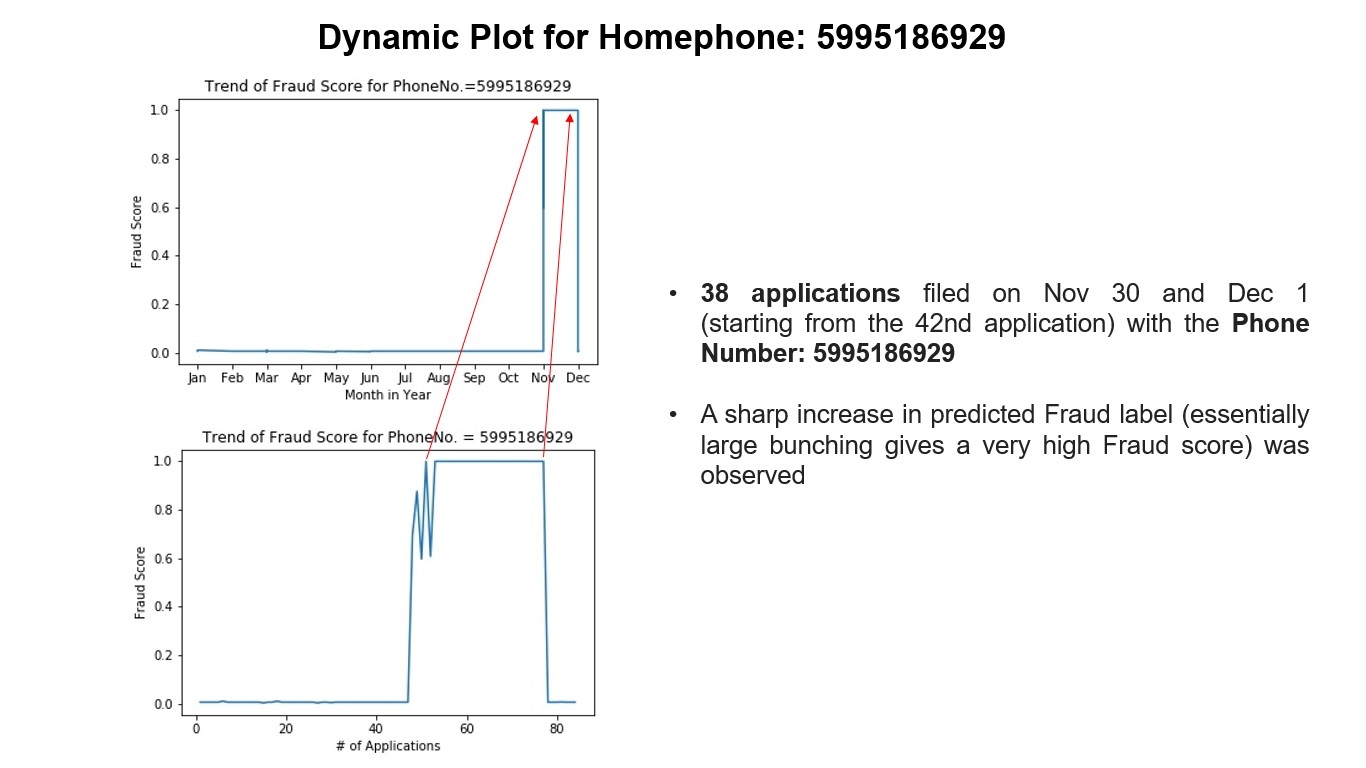
As it is evident from the OOT table, Fraud Savings Table and the graph above, we observe maximum savings by just looking at the top 7% of the OOT population. This gives a cumulative FDR of 56% and we are getting total savings of **$7,445,600.**

**5. Dynamic Plots for Entities**

1. **For an SSN**



1. **For a HomePhone Number:**



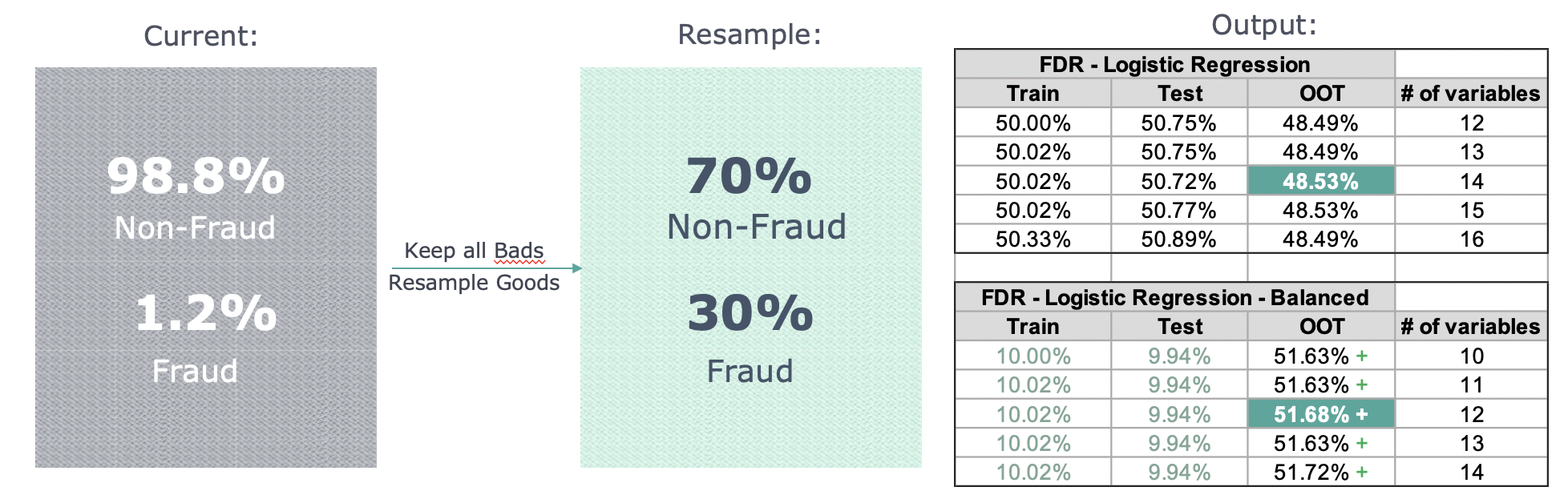
**Part VII. Conclusions**

Application fraud is one of the most common identity frauds. Falsified or stolen personal information is used to apply for cards, accounts, etc. In this report, we have examined the dataset to draw the following conclusion.

Comparing all the above models, we can conclude that XGBoost performed the best. We generated cum. Good, Bad, and percentages for all three sets (training, testing, and validation) as well as fraud savings plots. The FDR on training dataset is 53.97%, 55.24% on test set and 52.68% on the validation dataset. We used supervised algorithms including logistic regression, Adaboost, and gradient boosting. If given more time, we will try more complicated model like Neural Network and compare the result with other models to select the best one.

**Part VIII. Future Improvements**

Our models were trained, tested and validated using the original dataset, which, as mentioned, boasts only 1.2% of rows demonstrating fraud. Weighting a file is a proven method to improve model accuracy. Generally, fraud datafiles are imbalanced, so sub scrambling the goods or unscrambling the bads can help increase prediction accuracy. For this project, we created a weighted dataset featuring 30% fraud rows rather than the 1.2% using subscrambling of the goods. This method showed an increase of about 3% in FDR (3%) for our baseline logistic regression for validation. If given additional time and funding, our consulting team would utilize this more balanced dataset with all our algorithms to see if FDR (3%) prediction accuracy would increase.



Additionally, further gains in FDR could be achieved with the addition of external datasets related to our potential applicants. For example, a dataset from a cell phone company containing accurate name and phone number combinations could make it much easier to identify algorithmically whether or not someone is using falsified information in their application. Similarly, a collection of addresses and the last name of the owner could potentially lead to greater accuracy if utilized correctly. Adding additional variables or information related to the interactions between variables in the dataset could potentially help increase FDR in the future.

**Appendix**

**Data Quality Report on Applications Data**

**High-Level Description of Data**

The dataset contains application information of the New York City. The data is collected to evaluate the possibility of identity fraud. The data has 1,000,000 rows and 10 columns. All fields are categorical.

**Summary Table of All fields**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Missing values | Percentage Populated | Unique Values | Most Common Field Value |
| record | Categorical | 0 | 100.00% | 1000000 | N/A |
| date | Categorical | 0 | 100.00% | 365 | 20160816 |
| ssn | Categorical | 0 | 100.00% | 835819 | 999999999 |
| firstname | Categorical | 0 | 100.00% | 78136 | EAMSTRMT |
| lastname | Categorical | 0 | 100.00% | 177001 | ERJSAXA |
| address | Categorical | 0 | 100.00% | 828774 | 123 MAIN ST |
| zip5 | Categorical | 0 | 100.00% | 26370 | 68138 |
| dob | Categorical | 0 | 100.00% | 42673 | 19070626 |
| homephone | Categorical | 0 | 100.00% | 28244 | 9999999999 |
| fraud\_label | Categorical | 0 | 100.00% | 2 | 0 |

**Short Description and Picture of Each Field**

**Field 1**

**Field Name:** record (int64)

**Description:** record is a categorical variable. It is a unique value to identify each record. Each value in record is unique.

**Field 2**

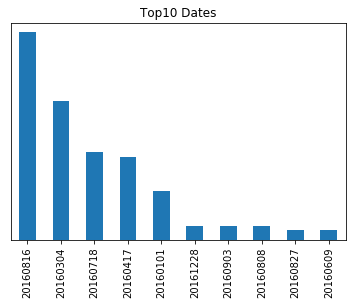
**Field Name:** date

**Description:**

date is a categorical variable representing the date of each transaction.

**Unique Value:**

date has 365 unique values. No missing value exists. The distribution is shown below, top 10 categories are listed below:



|  |  |
| --- | --- |
| Category | Count |
| 20160816 | 2877 |
| 20160304 | 2861 |
| 20160718 | 2849 |
| 20160417 | 2848 |
| 20160101 | 2840 |
| 20161228 | 2832 |
| 20160903 | 2832 |
| 20160808 | 2832 |
| 20160827 | 2831 |
| 20160609 | 2831 |

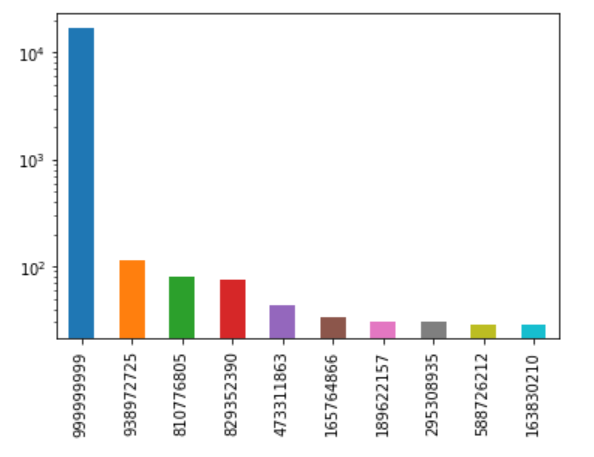
**Field 3**

**Field Name:** ssn (categorical, dtype: int64)

**Description:**

ssn is a categorical variable representing the social security number of the applicant.

ssn has 835819 unique values. No missing value exists. The distribution is shown below, top 10 categories are listed below. The following bar chart shows the log count and top 10 ssn.



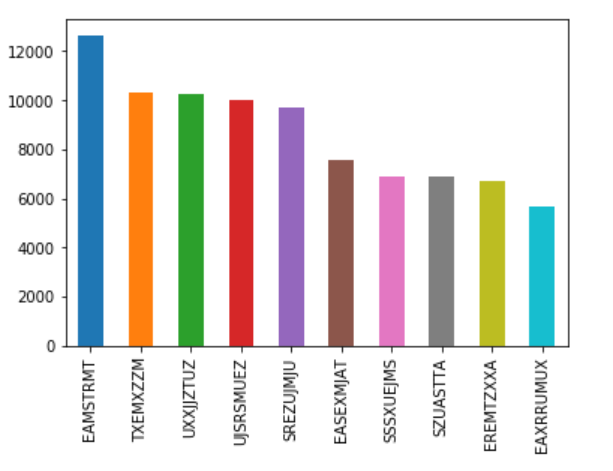
|  |  |
| --- | --- |
| Category | Count |
| 999999999 | 16935 |
| 938972725 | 114 |
| 810776805 | 81 |
| 829352390 | 74 |
| 473311863 | 44 |
| 165764866 | 34 |
| 189622157 | 30 |
| 295308935 | 30 |
| 588726212 | 29 |
| 163830210 | 29 |

**Field 4**

**Field Name:** firstname (categorical, dtype: object)

**Description:**

firstname is a categorical variable representing the first name of the applicants. firstname has 78136 unique values. No missing value exists. The distribution is shown below, top 10 categories are listed below. The following bar chart shows the count and top 10 firstname.



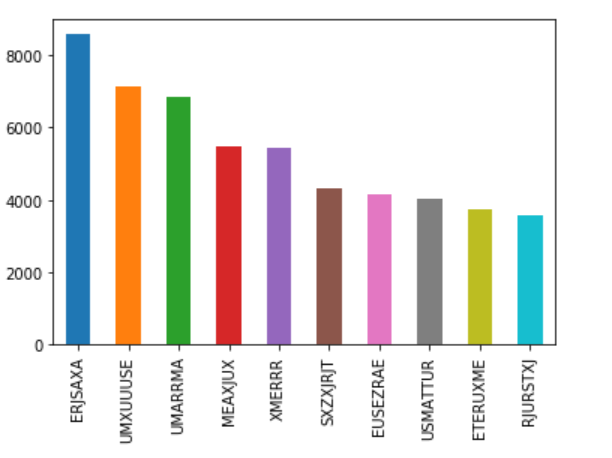
|  |  |
| --- | --- |
| Category | Count |
| EAMSTRMT | 12658 |
| TXEMXZZM | 10297 |
| UXXJJZTUZ | 10235 |
| UJSRSMUEZ | 9994 |
| SREZUJMJU | 9688 |
| EASEXMJAT | 7576 |
| SSSXUEJMS | 6923 |
| SZUASTTA | 6878 |
| EREMTZXXA | 6717 |
| EAXRRUMUX | 5686 |

**Field 5**

**Field Name:** lastname (categorical, dtype: object)

**Description:**

lastname is a categorical variable describing the last name of the applicants. lastname has 177001 unique values. No missing value exists. The distribution is shown below, top 10 categories are listed below. The following bar chart shows the count and top 10 lastname.



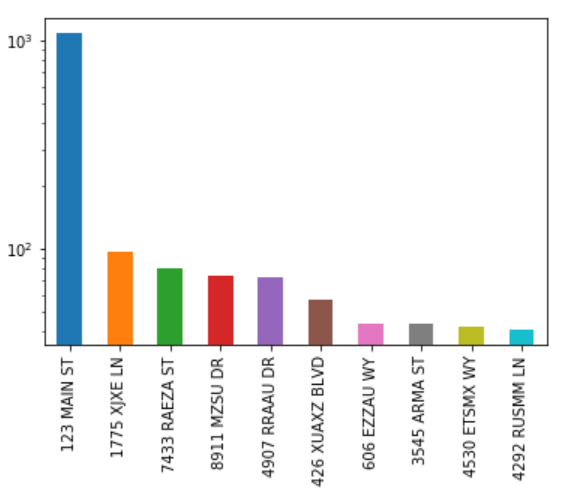
|  |  |
| --- | --- |
| Category | Count |
| ERJSAXA | 8580 |
| UMXUUUSE | 7156 |
| UMARRMA | 6832 |
| MEAXJUX | 5492 |
| XMERRR | 5451 |
| SXZXJRJT | 4340 |
| EUSEZRAE | 4173 |
| USMATTUR | 4036 |
| ETERUXME | 3762 |
| RJURSTXJ | 3575 |

**Field 6**

**Field Name:** address (categorical, dtype: object)

**Description:**

address is a categorical variable describing the address of the applicant. address has 828774 unique values. There are no missing values. Below are top 10 categories in descending order. The following bar chart shows the log count and top 10 addresses.



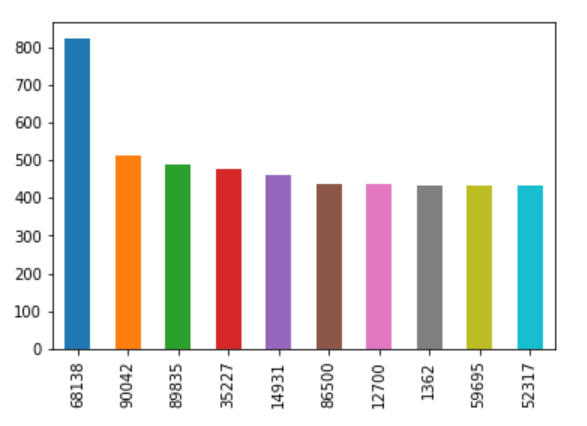
|  |  |
| --- | --- |
| Category | Count |
| 123 MAIN ST | 1079 |
| 1775 XJXE LN | 97 |
| 7433 RAEZA ST | 80 |
| 8911 MZSU DR | 74 |
| 4907 RRAAU DR | 73 |
| 426 XUAXZ BLVD | 57 |
| 606 EZZAU WY | 44 |
| 3545 ARMA ST | 44 |
| 4530 ETSMX WY | 42 |
| 4292 RUSMM LN | 41 |

**Field 7**

**Field Name:** zip5 (categorical, dtype: int64)

**Description:**

zip5 is a categorical variable describing the 5-digit zip code of the applicants. zip5 has 26370 unique values. There are no missing values. Below are top 10 categories. The following bar chart shows the count of top 10 zip codes.



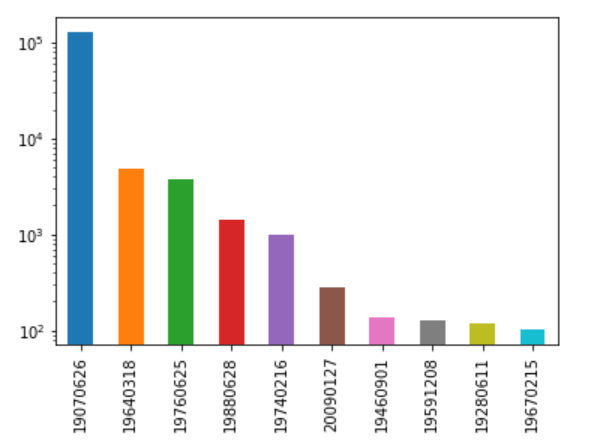
|  |  |
| --- | --- |
| Category | Count |
| 68138 | 823 |
| 90042 | 514 |
| 89835 | 489 |
| 35227 | 478 |
| 14931 | 459 |
| 86500 | 438 |
| 12700 | 436 |
| 1362 | 434 |
| 59695 | 432 |
| 52317 | 432 |

**Field 8**

**Field Name:** dob (categorical, dtype: int64)

**Description:**

dob is a categorical variable describing the date of birth of the applicants. dob has 42673 unique values. There are 0 missing values. Below are top 10 categories in descending order. The following bar chart shows the log count of top 10 dates of birth.



|  |  |
| --- | --- |
| Category | Count |
| 19070626 | 126568 |
| 19640318 | 4818 |
| 19760625 | 3723 |
| 19880628 | 1404 |
| 19740216 | 980 |
| 20090127 | 280 |
| 19460901 | 135 |
| 19591208 | 126 |
| 19280611 | 120 |
| 19670215 | 102 |

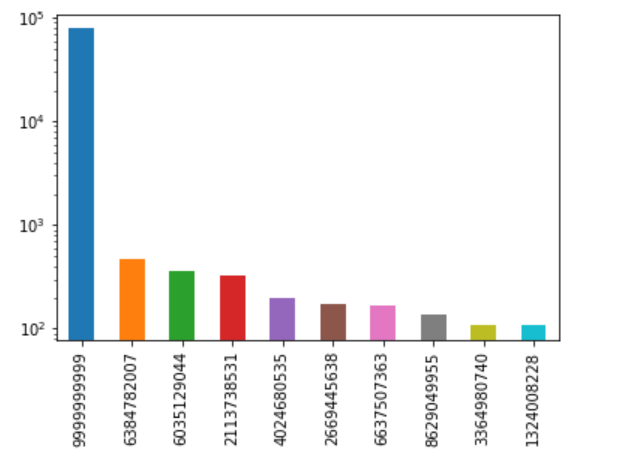
**Field 9**

**Field Name:** homephone

**Description:**

homephone is a categorical variable describing the home phone number of the applicants.

homephone has 28244 unique values. There are 0 missing values. Below are top 10 categories.

****

|  |  |
| --- | --- |
| Category | Count |
| 9999999999 | 78512 |
| 6384782007 | 466 |
| 6035129044 | 360 |
| 2113738531 | 331 |
| 4024680535 | 198 |
| 2669445638 | 172 |
| 6637507363 | 169 |
| 8629049955 | 139 |
| 3364980740 | 110 |
| 1324008228 | 108 |

**Field 10**

**Field Name:** fraud\_label

**Description:**

fraud\_label is a categorical variable denoting whether an application is fraud.

LTDEPTH has 2 unique values, 0 and 1. There are 985607 zeros and 14393 ones. Below is the log distribution of two categories.

