

DSO 562 Fraud Analytics  
Project 2 Report  
Professor Coggeshall  
March 25th, 2021



# Application Fraud Detection

A Report on Building a Supervised Machine Learning Model on Credit  
Product Application Data

## Team

Anushka Kapur - [anushka.kapur.2021@marshall.usc.edu](mailto:anushka.kapur.2021@marshall.usc.edu)

Benjamin (Ben) Lee - [benjamin.lee.2021@marshall.usc.edu](mailto:benjamin.lee.2021@marshall.usc.edu)

Christopher (Kai) Painton - [painton.christopher.2021@marshall.usc.edu](mailto:painton.christopher.2021@marshall.usc.edu)

Joyce Yam - [joyce.yam.2021@marshall.usc.edu](mailto:joyce.yam.2021@marshall.usc.edu)

Kyu Tae (Jayden) Cho - [kyutae.cho.2021@marshall.usc.edu](mailto:kyutae.cho.2021@marshall.usc.edu)

Shun (Bowen) Wong - [shun.wong.2021@marshall.usc.edu](mailto:shun.wong.2021@marshall.usc.edu)

Yutong Wanyan - [yutong.wanyan.2021@marshall.usc.edu](mailto:yutong.wanyan.2021@marshall.usc.edu)

## Table of Contents

<b>Executive Summary</b>	<b>3</b>
<b>Description of Data</b>	<b>4</b>
<b>Data Cleaning</b>	<b>7</b>
<b>Candidate Variables</b>	<b>8</b>
<b>Feature Selection Process</b>	<b>10</b>
Filter Method	10
Wrapper Method	11
<b>Model Algorithms</b>	<b>14</b>
Logistic Regression	14
Random Forest	15
XGBoost	15
Neural Networks	16
<b>Results</b>	<b>17</b>
<b>Conclusions</b>	<b>19</b>
<b>Appendix</b>	<b>20</b>

## Executive Summary

This project focuses on using advanced analytical modeling and techniques to find identity fraud. Identity fraud is the act of misrepresenting who you are in order to improperly receive services or to remain “hidden” while conducting illegal activity. There are three core methods of identity fraud: identity theft, identity manipulation, and using synthetic identity. Our goal was to use only identity fields to build a model for flagging potential application/identity fraud in real time.

Our dataset was synthetic generated from analysis of over one billion real US applications over a ten year period. The data was created to attempt to mimic real application data and contained over one million records with various personally identifiable information fields including social security number, first and last name, address, date of birth, and home phone number.

We started by cleaning the data set of any frivolous fields, common values used to fill missing fields, and replaced them with unique values. We created over 350 candidate variables by linking applications through different combinations of identity fields. These candidate variables included a risk table for the day of the week, days since last seen for a specific attribute, velocity which captures the number of times an attribute appears in the past n days, and relative velocity, a ratio between recent occurrences and longer time periods for an attribute.

In order to prepare our dataset for model building, we used a filter and wrapper feature selection process to reduce the total number of variables. First, we ran a filter, which selected the variables with the highest average KS and FDR rank. Next, we used RFECV, a backward selection wrapper, to rank and narrow our field of variables. Lastly, we ran RFE, without cross validation, to get a well-ranked list of 30 different candidate variables.

Using our 30 most-important variables, we created 43 unique models from four different binary classification algorithms: Logistic Regression, Random Forest, XGBoost, and Neural Networks. Each of these models used an unique combination of sampling strategies and hyperparameters. Based on the best performance on the test set, we selected XGBoost, a gradient boosting algorithm, as our final model with 600 estimators, a max depth of 3 for each estimator, a learning rate of 0.1, and a subsample ratio of 0.8. This model produced a fraud detection rate for 3% of the population at 53.56%. The success of our final model demonstrates a real business opportunity to implement our solution as an effective way of catching identity fraud.

## Description of Data

Our dataset was synthetically generated from analysis of over one billion real credit card and cell phone applications over a ten year period. This data contained one million records within a one year time period, from January 1st, 2016 to December 31st, 2016. There are a total of ten unique fields containing identity information like social security number, applicant's name, address, homephone, and date of birth. Each record had a fraud label with 1 flagging an application as identity fraud and 0 representing a normal application. Below is a summary table detailing the percent of fields populated, the number of unique values, and the most common field value.

Column Name	# Records	% Populated	Unique Values	Min	Max	Most Common Field Value
record	1,000,000	100	1000000	N/A	N/A	N/A
date	1,000,000	100	365	20160101	20161231	20160816
ssn	1,000,000	100	835819	N/A	N/A	999999999
firstname	1,000,000	100	78136	N/A	N/A	EAMSTRMT
lastname	1,000,000	100	177001	N/A	N/A	ERJSAXA
address	1,000,000	100	828774	N/A	N/A	123 MAIN ST
zip5	1,000,000	100	26370	N/A	N/A	68138
dob	1,000,000	100	42673	19000101	20161031	19070626
homephone	1,000,000	100	28244	N/A	N/A	999999999
fraud_label	1,000,000	100	2	N/A	N/A	0

Table 1.1: Summary Table For All Fields in Dataset

For each field, we plotted the distribution and the most frequent occurrences to better understand the nature of our data before creating any candidate variables. Our plots were organized into a data quality report, which can be found in the appendix. The following distributions and bar charts highlight some important characteristics of our dataset.

### Date

The volume of applications were fairly similar throughout the year with hikes around the beginning of each new quarter.

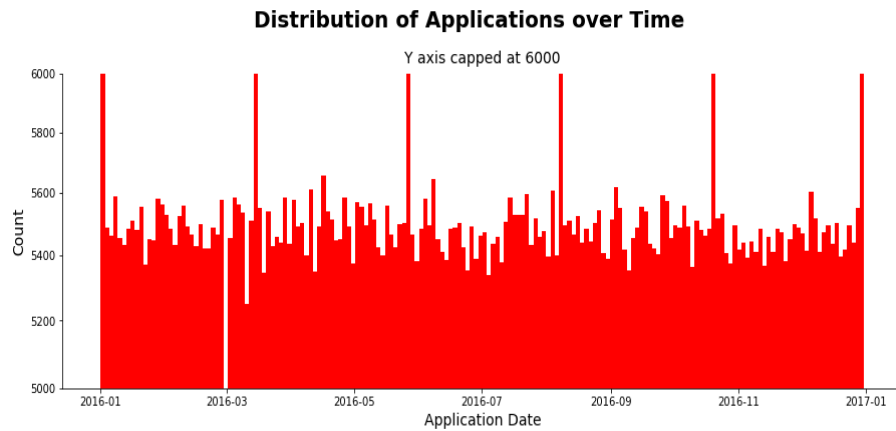


Figure 1.1: Distribution of Application Dates Over Time

## SSN

Social security number '999999999' was most frequently used and led us to conclude that it is likely a frivolous value, a value that was originally missing, but filled with a new common value.

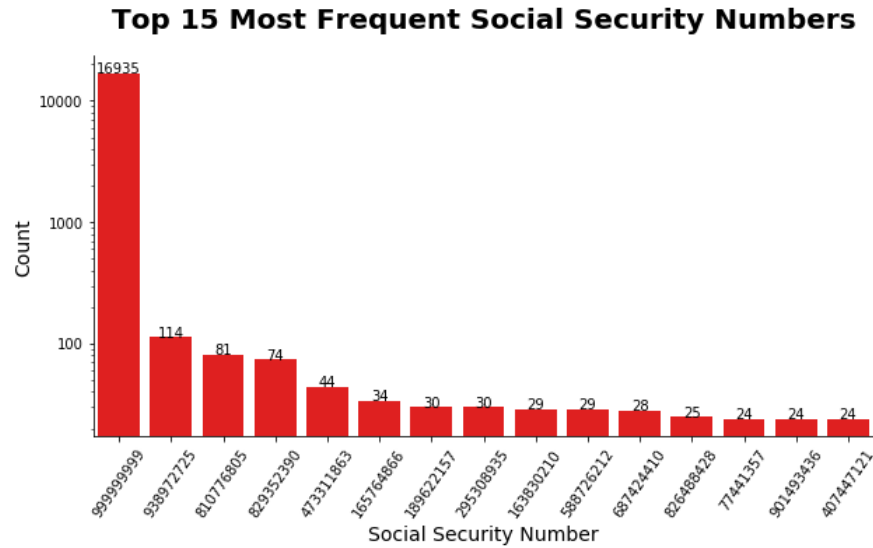


Figure 1.2: Most Frequent Social Security Numbers

## Address

'123 Main Street' appeared significantly more than the other addresses and led us to believe that it was also a frivolous value.

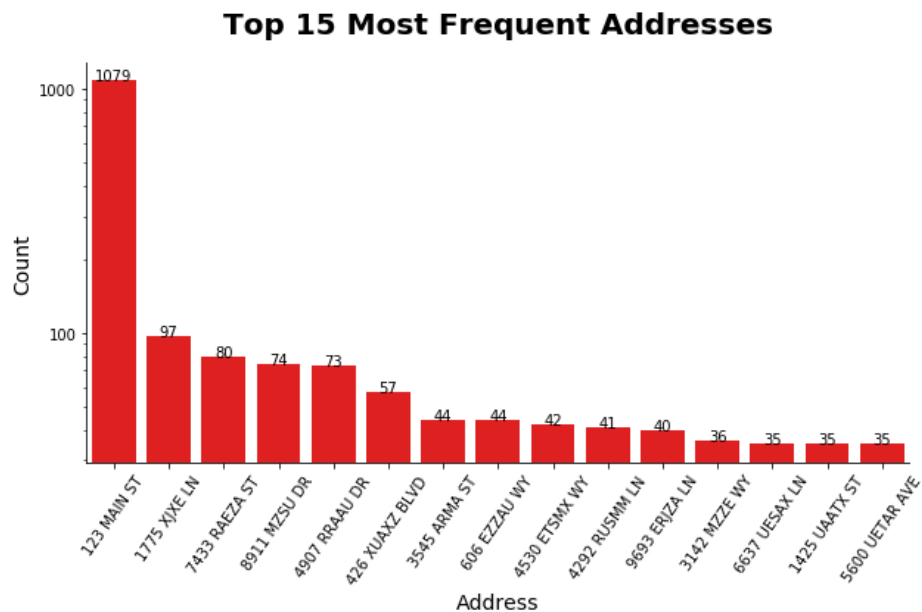


Figure 1.3 Most Frequent Applicant Address

## Zip Code

Postal code '68138' was most frequently used, but the total count was not much larger than other postal codes so we decided to not classify it as a frivolous value.

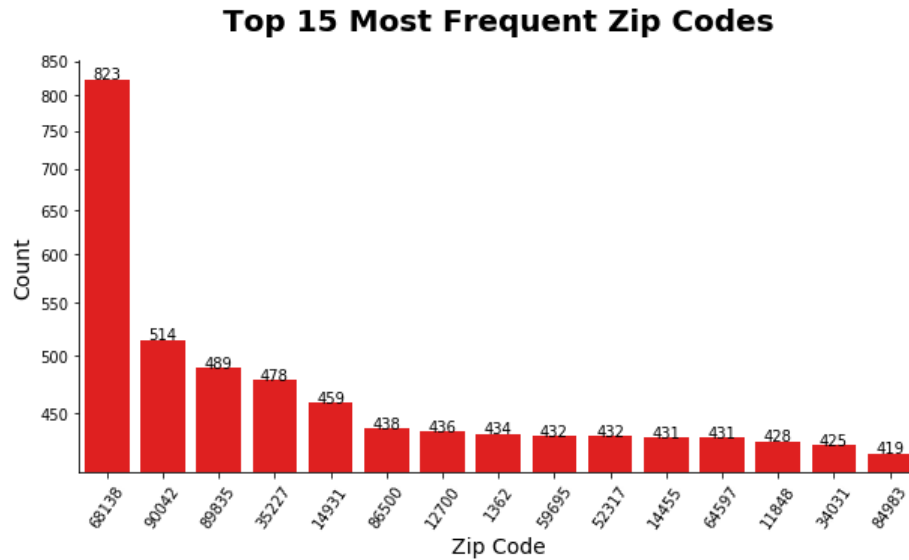


Figure 1.4 Most Frequent Zip Codes

## Homephone

Home phone number '999999999' was by far the most frequently used, indicating that it is likely a frivolous value.

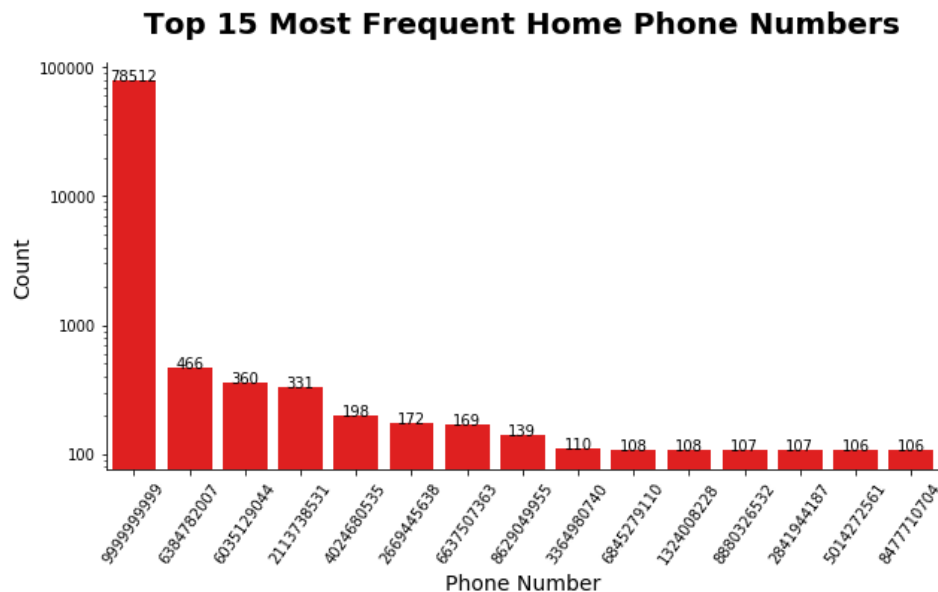


Figure 1.5 Most Frequent Phone Numbers

## Data Cleaning

After initial investigation into the dataset, we discovered that all fields were 100% populated, therefore we did not need to treat any missing values. However, we discovered that there were frivolous values in four of the fields, shown in Table 2.1.

Field	Frivolous Value	Count
<i>ssn</i>	999999999	16,935
<i>address</i>	123 MAIN ST	1,079
<i>dob</i>	19070626	126,568
<i>homephone</i>	999999999	78,512

**Table 2.1:** Count of Frivolous Values

Frivolous values are made-up values that are used to fill in missing values in a credit application. These values are risky as they could accidentally link two completely unrelated applications. Hence, before proceeding with any further actions on our dataset, frivolous values must be cleaned and replaced in order to avoid undesired results and false conclusions.

There are different approaches to treat frivolous values. For this project, we decided to replace these values with a unique value that will not link.

We chose to replace each frivolous value in field *ssn*, *dob* and *homephone* with the negative of its corresponding unique record number and added leading zeros. For instance, if record # 100 has a value of “999999999” in the *ssn* field, a value of “19070626” in the *dob* field and a value of “999999999” in the *homephone* field, these frivolous values were replaced with “-000000100”.

For each frivolous value in the *address* field, we chose to replace it with its corresponding unique record number, and append the string “RECORD” to the end. For instance, if record # 101 has a value of “123 MAIN ST” in the *address* field, the frivolous value was replaced with “101 RECORD”.

To maintain data type consistency, all four aforementioned data fields were converted into string types after treating the frivolous values. The *date* and *zip5* fields were also converted into datetime and string type respectively.

## Candidate Variables

After handling frivolous values, we built 393 new variables from the original identity fields from four different categories.

Apart from the original identity fields, we derived four new fields, which were used when creating variables. We first created an Age variable which is the date difference between the date of birth and application date. Then, we created age bins with eight buckets with boundaries of [ $<13.75$ ,  $<28.5$ ,  $<43.25$ ,  $<58$ ,  $<72.75$ ,  $<87.5$ ,  $<102.25$ ,  $<117$ ]. An area code was extracted from the first three digits of the home phone number. Lastly, we found the dow, which is an abbreviation of day of week, for the weekday an application was submitted.

Next, we created two new identity fields: *namedob* and *fulladdress*. *Namedob* was created by appending applicant's first name and last name, followed by their date of birth. Full address was made by appending address and zip code.

We created unique attributes using unique combinations of two or more identity fields and linking entities. The entities used for linking were *ssn*, *fulladdress*, *namedob*, and *homephone*. A total of 26 new attributes were created from this process. One attribute example is *ssn\_name\_dob* where applicant's ssn, name, and date of birth are concatenated.

The final step was to create numerical variables. The first variable created was *dow\_risk*, which uses targeted encoding to assign each weekday the fraud proportion on that specific day of the week. To avoid overfitting, we used a smoothing formula with  $c=4$  and  $nmid=20$ . Another similar variable we created was the *Age\_bin\_risk*, which encoded our age bins with the fraud proportion for each age group. We chose the smoothing parameter  $c=4$  and  $nmid=20$  for *age\_bin\_risk*.

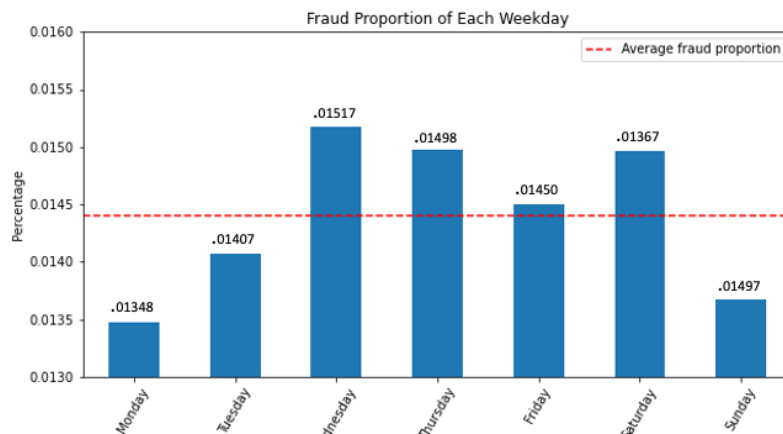


Figure 3.1 Day of Week Risk Table



Our next set of variables focused on using time windows to capture the fraudulent application activity over a period of time. The  $\{attribute\}_{days\_since}$  variables show how many days have passed since that attribute was last observed. In case that attribute was not seen before, or in other words, this is the first time this attribute has appeared, the value was filled with 365. One example of this variable is  $name\_dob\_days\_since$ , which is the number of days since that same  $namedob$  was observed. A total of 26 days since variables were created.

The next variable type looked at the velocity,  $\{attribute\}_{count\_n}$ , which captures the number of times that the same attribute has appeared in the past  $n$  days. For  $n$ , we chose 6 different time windows: 0, 1, 3, 7, 14, 30 days. As an example,  $name\_dob\_count\_7$  represents the number of times that the identity with the same name and date of birth was seen in the last 7 days. We created 156 total variables of the velocity type, which were all the combinations of our 26 attributes and 6 time windows.

$$\frac{\text{\# apps with that } \mathbf{group} \text{ seen in the recent past}}{\text{\# apps with that } \mathbf{same\ group} \text{ seen in the past } \{1, 3, 7, 14, 30\} \text{ days}}$$

**Figure 3.2 Relative Velocity Formula**

The last type is the relative velocity,  $\{attribute\}_{count\_n\_recent}_{n\_past}$ , which captures the ratio of the amount of times an attribute was seen in the last 0 and 1 days ( $n\_recent$ ) to the amount of times that same attributes was seen in past 3, 7, 14, 30 days ( $n\_past$ ). In other words, it finds the relative recent activity of an attribute compared to longer past periods. For instance,  $name\_dob\_count\_0\_by\_7$  is the proportion that an attribute with the same name and date of birth was seen in the past 0 day over the last 7 days. Using the combinations of the 26 attributes, two  $n\_recent$  days, and four  $n\_past$  days, we created 208 ( $26*2*4$ ) variables of this type.

A full list of all the created candidate variables can be found in the appendix.

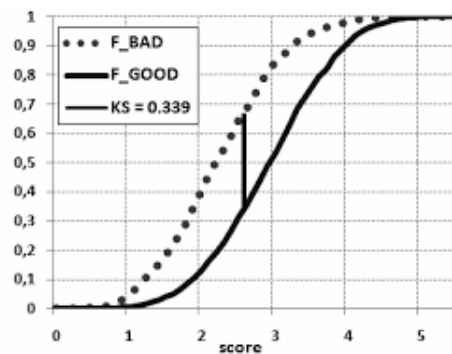
## Feature Selection Process

After creating 393 candidate variables, it became necessary to reduce dimensionality and find the most informative, substantial variables for our prediction purposes. By reducing the number of features, our model would have fewer chances of fitting data noise and thus improve overall prediction accuracy. In this feature selection process, we implemented a 2-step method to obtain a final list of 30 expert variables to use for our model building.

### Filter Method

Our first step for feature selection was to apply a filter on our 393 variables to reduce to a ranked list of the top 80. Our filter method ranked all variables in order of how individually important each was to predicting the fraud label. The filter method is fast and stable, which enables us to quickly sort and rank the potentially useful candidate variables without running any time-consuming algorithms.

We calculated two metrics, Kolmogorov-Smirnov (KS) statistic and Fraud Detection Rate (FDR) when only using 3% of data, and created a synthetic score by averaging their ranks in order of importance. The KS score is a measure of how separated two classes are given one candidate variable. Therefore, a variable is considered more important to our prediction target if it has a higher KS statistic, as it can help us better distinguish data points between two classes (fraud and non-fraud). The FDR at 3%, on the other hand, measures the percentage of all fraud found for a single, sorted candidate variable. Fraud is by nature extremely hard to detect and also an event that happens relatively rarely in the real-life situation, so a strong candidate variable has to perform well even if only a small amount of data is given. Therefore, we calculated KS and FDR and averaged them to get a balanced measure of importance.



**Figure 4.1: KS Statistic Explanation** - A measure of how separated two classes are given each candidate variable

To make sure our scoring was reliable, we added two special variables to the list of candidate variables: a copy of the fraud label and a randomly generated variable. As we saw in the output,

the fraud label copy was ranked as 1 out of all 395 (393 candidates plus two) variables, and the random variable ranked near the bottom. This indicated we had precise calculations and ensured the reliability of the sorted list.

Candidate Variables	Scores	
	ks	fdr
fraud_label	1	1
age	0.0757792	0.0426606
dow_risk	0.0220876	0.0392652
age_bin_risk	0.0712698	0.0418771
ssn_day_since	0.227255	0.260839
...	...	...
ssn_homephone_name_dob_count_1_by_3	0.0096551	0.0491033
ssn_homephone_name_dob_count_1_by_7	0.0254564	0.0646004
ssn_homephone_name_dob_count_1_by_14	0.0446558	0.0812293
ssn_homephone_name_dob_count_1_by_30	0.0600989	0.0939404
random	0.00955982	0.0363921

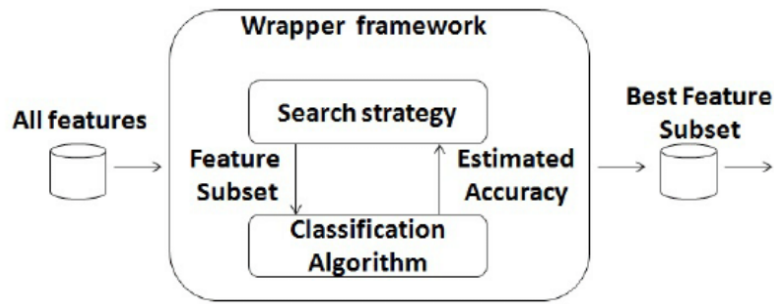
Table 4.1: Unsorted Calculation of KS and FDR Score for All Candidate Variables

Candidate Variables	Scores		Ranks		
	ks	fdr	rank_ks	rank_fdr	average_rank
fraud_label	1	1	395	395	395
address_day_since	0.334096	0.366794	394	394	394
fulladdress_day_since	0.33321	0.364531	393	393	393
fulladdress_count_30	0.332032	0.36427	391	392	391.5
address_count_30	0.332725	0.363399	392	391	391.5
...	...	...	...	...	...
fulladdress_dob_count_0	0.001812	0.044315	8	8	8
ssn_fulladdress_count_0	0.001774	0.044315	5.5	8	6.75
ssn_name_fulladdress_count_0	0.001774	0.044315	5.5	8	6.75
ssn_fulladdress_dob_count_0	0.001729	0.044315	3	8	5.5
ssn_name_homephone_count_0	0.001708	0.044228	1	5	3

Table 4.2: Sorted Rank of Candidate Variables Based on Average Score of KS and FDR

## Wrapper Method

After finding the top 80 candidate variables from the KS-FDR filter, we further reduced the number of candidate variables to 30 by applying a wrapper method. A wrapper method, specifically a backward selection method, builds a single classification model using all variables and removes one variable at a time, starting with the least important variable on the list. We decided to use logistic regression as the underlying model due to its effectiveness and simpleness.



**Figure 4.2: Wrapper Method Explanation** - This method uses an algorithm to select the best candidate variables

We first applied this wrapper method with the built-in function from the `sklearn.feature_selection` package that returned the best number of features based on recursive feature elimination and cross-validated selection.

However, our initial list ranked many variables as 1, which indicated our method couldn't distinguish the importance of candidate variables. We came up with two strategies for making changes to the wrapper. The first alternative was to use an advanced non-linear model like a decision tree, as they might capture non-linear patterns in our dataset compared to logistic regression. The second alternative was to skip cross-validation in the wrapper process, as the cross-validation would improve model performance as much as possible, thus the algorithm couldn't effectively rank candidates. We implemented the second alternative and created a well-ranked list of 30 variables that could best contribute to the prediction of fraud probability.

Final Variables			
	ranking	variable	Description
0	1	address_count_0	Number of times that the same address has occurred in the same day
1	1	address_count_0_by_7	The ratio that the same address was seen in the last 0 day to it seen in the past 7 days
2	1	address_count_1_by_7	The ratio that the same address was seen in the last 1 day to it seen in the past 7 days
3	1	address_count_7	Number of times that the same address has occurred in the past 7 days
4	1	fulladdress_count_0_by_7	The ratio that 'fulladdress_homephone' was seen in the last 1 day to it seen in the past 7 days
5	1	fulladdress_homephone_count_3	Number of times that the same fulladdress_homephone has occurred in the past 3 days
6	1	fulladdress_homephone_count_7	Number of times that the same fulladdress_homephone has occurred in the past 7 days
7	1	homephone_count_1	Number of times that the same homephone has occurred in the past 1 days
8	1	name_dob_count_14	Number of times that the same name_dob has occurred in the past 14 days
9	1	name_dob_count_30	Number of times that the same name_dob has occurred in the past 30 days
10	1	ssn_count_14	Number of times that the same ssn has occurred in the past 14 days
11	1	ssn_dob_count_14	Number of times that the same ssn_dob has occurred in the past 14 days
12	1	ssn_firstname_count_7	Number of times that the same ssn_firstname has occurred in the past 7 days
13	2	fulladdress_count_0	Number of times that the same fulladdress has occurred in the same day
14	4	fulladdress_count_30	Number of times that the same fulladdress has occurred in the past 30 days
15	10	fulladdress_count_1	Number of times that the same fulladdress has occurred in the past 1 day
16	12	fulladdress_count_0_by_14	The ratio that the same fulladdress was seen in the last 0 day to it seen in the past 14 days
17	13	ssn_count_3	Number of times that the same ssn has occurred in the past 3 days
18	14	fulladdress_count_0_by_3	The ratio that the same fulladdress was seen in the last 0 day to it seen in the past 3 days
19	17	ssn_dob_count_3	Number of times that the same ssn_dob has occurred in the past 3 days
20	20	ssn_count_7	Number of times that the same ssn has occurred in the past 7 days
21	21	ssn_dob_count_0_by_14	The ratio that the same ssn_dob was seen in the last 0 day to it seen in the past 14 days
22	23	ssn_firstname_count_3	Number of times that the same ssn_firstname has occurred in the past 3 days
23	26	address_count_0_by_3	The ratio that the same address was seen in the last 0 day to it seen in the past 3 days
24	27	name_dob_count_3	Number of times that the same name_dob has occurred in the past 3 days
25	28	address_count_1	Number of times that the same address has occurred in the past 1 day
26	29	ssn_firstname_count_30	Number of times that the same ssn_firstname has occurred in the past 30 days
27	31	ssn_count_30	Number of times that the same ssn has occurred in the past 30 days
28	33	ssn_firstname_count_14	Number of times that the same ssn_firstname has occurred in the past 14 days
29	34	ssn_dob_count_30	Number of times that the same dob has occurred in the past 30 days

**Table 4.3: A Final List of 30 Candidate Variables Selected**

Based on the output of the wrapper method

## Model Algorithms

As we are attempting to identify applications as potential fraud, the most appropriate modeling approach was to build a binary classification model. Before training our models, we split our data into two sets: a train and test set made up of applications prior to October 31<sup>st</sup>, 2016 and an out-of-time (OOT) set made of applications that we received after October 31<sup>st</sup>, 2016. Our train and test data were split into a 75% training group and a 25% testing group.

Since roughly 1.5% of our fields were flagged as frauds, we had a potential imbalanced data issue. As a remedy, we utilized three different sampling techniques: undersampling, SMOTE, and tuning the weight parameter within the algorithm. Undersampling is a technique where you down sample the majority class to meet a specific ratio with the minority class. Synthetic minority oversampling technique (SMOTE) can be used to synthetically generate more minority classes, by taking the nearest neighbors of all “bad” records and creating similar ones. Lastly, some algorithms have a weight parameter to inform the model of different class weights. We used a ratio of 10 “goods” to 1 “bad” as our parameter setting for some models to place more emphasis on the “bad” records by minimizing the ratio between the two classes.

We tested four different algorithms across many different hyperparameters to determine which method would be the most effective in finding identity fraud.

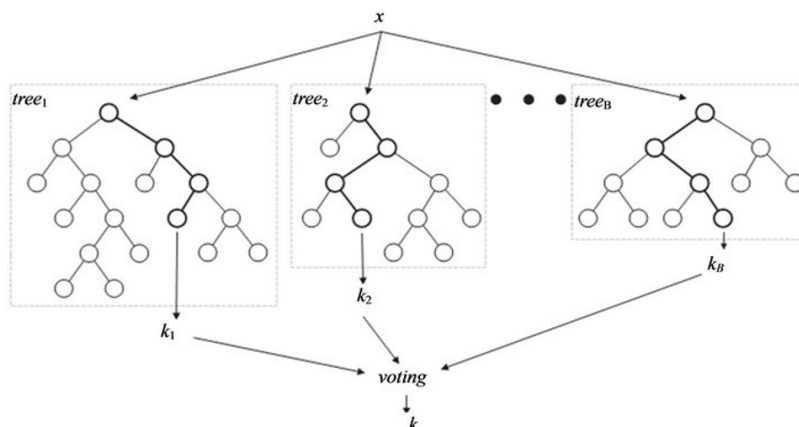
### Logistic Regression

Our first algorithm used logistic regression, a linear classifier algorithm. This algorithm outputs a probability of a record being identified as fraud by using the logistic sigmoid function which forms a sideways S curve on a graph. In order to map these probabilities to a prediction, we use a linear decision boundary to assign records to specific classes. The hyperparameters we controlled for in our model were the penalty parameter which controls regularization, C another regularization term, and the solver which is used for optimizing the algorithm.

Model	Parameters					Average FDR at 3%		
	Iteration	Sampling Strategy (Majority:Minority)	penalty	C	solver	Train	Test	OOT
Logistic Regression	1	No Sampling	l2	1	liblinear	52.51	52.14	50.00
	2	Class weight Argument (10:1)	l2	1	liblinear	52.81	52.48	50.84
	3	SMOTE (70:30)	l2	1	liblinear	53.50	53.15	51.30
	4	SMOTE (60:40)	l2	1	liblinear	53.94	53.41	51.30
	5	SMOTE (50:50)	l2	1	liblinear	53.95	53.41	51.30
	6	Undersample (70:30)	l2	1	liblinear	54.43	53.92	52.14
	7	Undersample (60:40)	l2	1	liblinear	54.20	53.88	52.14
	8	Undersample (50:50)	l2	1	liblinear	54.53	53.85	52.10
	9	No Sampling	l2	0.1	saga	54.55	53.82	51.72
	10	Undersample (70:30)	l2	0.1	saga	54.65	53.92	52.10
	11	SMOTE (70:30)	l2	0.1	saga	54.14	53.55	51.26
	12	No Sampling	l2	0.01	lbfgs	53.33	52.54	50.80
	13	Undersample (70:30)	l2	0.01	lbfgs	53.89	53.18	51.34

Table 5.1 Logistic Regression Hyperparameter Performance Table

## Random Forest



**Figure 5.1: Random Forest Explanation:** A collection of many strong models with averaged scoring

The next modeling algorithm we tried was a Random Forest, an ensemble method that averages many strong decision tree models to build a more accurate and stable prediction. On a high level, a decision tree works by carving up spaces into boxes and assigns a model output for each carved box. Random Forest builds a collection of many of these decision tree models, selecting a random subset of variables or records, and votes based on the outputs of each model to flag a record as fraud or not. We tried a wide range of hyperparameters, including the number of trees, the depth of each tree, the criterion to measure the quality of each split, and the minimum number of samples required to split an internal node.

Model	Parameters						Average FDR at 3%		
	Iteration	Sampling Strategy (Majority:Minority)	criterion	n_estimators	max_depth	min_samples_split	Train	Test	OOT
Random Forest	1	No Sampling	gini	100	20	300	55.59	55.06	53.35
	2	Class Weight Argument (10:1)	gini	100	20	300	55.66	55.09	53.56
	3	SMOTE (70:30)	gini	100	20	300	55.75	55.09	51.55
	4	Undersample (70:30)	gini	100	20	300	55.38	54.60	52.98
	5	Class Weight Argument (10:1)	entropy	150	10	200	55.51	54.76	53.35
	6	Class Weight Argument (10:1)	entropy	150	25	300	55.73	55.06	53.48
	7	Class Weight Argument (10:1)	entropy	150	30	400	55.67	55.12	53.52
	8	Class Weight Argument (10:1)	gini	100	20	400	55.71	55.12	53.52
	9	Class Weight Argument (10:1)	gini	50	25	300	55.65	55.06	53.44
	10	Class Weight Argument (10:1)	gini	50	25	300	55.69	54.99	53.56

**Table 5.2 Random Forest Hyperparameter Performance Table**

## XGBoost

Another modeling algorithm we tested was XGBoost, a gradient boosting algorithm. XGBoost iteratively trains a series of weak models, correcting the previous errors to result in stronger predictions. These additional models will continue to be added sequentially until no further improvements can be made. It uses a gradient descent algorithm as a way of minimizing the errors for each new tree that is added to the model. We focused on tuning the number of trees, the learning rate for preventing overfitting, the max depth of each tree, and the ratio for sampling the training data prior to building any trees.

Model	Parameters						Average FDR at 3%		
	Iteration	Sampling Strategy (Majority:Minority)	learning_rate	n_estimators	max_depth	subsample	Train	Test	OOT
XGBoost	1	No Sampling	0.1	400	3	1	57.61	55.22	53.31
	2	Scale Pos Weight = 10	0.1	400	3	1	62.56	55.02	53.02
	3	SMOTE (70:30)	0.1	400	3	1	58.28	54.89	49.75
	4	Undersample (70:30)	0.1	400	3	1	57.33	54.89	52.85
	5	No Sampling	0.05	500	5	0.7	56.92	55.35	53.10
	6	No Sampling	0.1	750	5	0.8	59.49	55.02	52.68
	7	No Sampling	0.1	750	5	1	58.54	54.92	53.10
	8	No Sampling	0.01	1000	3	1	55.65	55.29	53.25
	9	No Sampling	0.1	600	3	0.8	57.04	55.46	53.19
	10	No Sampling	0.05	600	5	1	57.11	55.05	53.44

Table 5.3 XGBoost Hyperparameter Performance Table

## Neural Networks

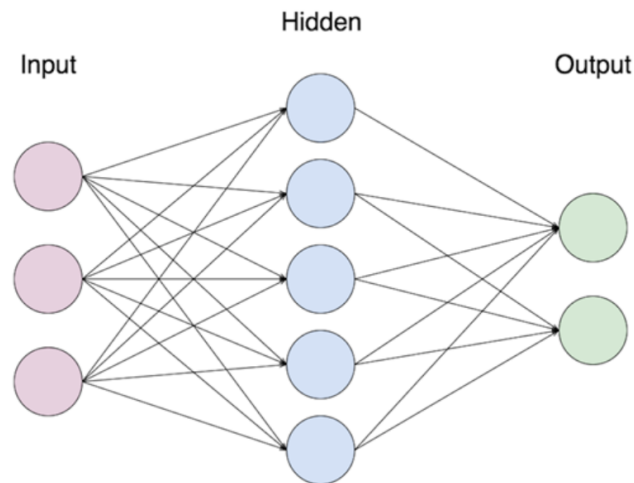


Figure 5.2 Simple Neural Network with One Hidden Layer and 5 Nodes

Our last modeling algorithm was a neural network, a method that attempts to mimic human brain activity by using a set of hidden layers and nodes. Each node receives a weighted signal from all the previous layer's nodes and performs a transformation on the linear combinations of all these signals. The data is passed through this model many times, called epochs, and each epoch attempts to reduce the error of the previous run. During each epoch, the weights of the signals are adjusted from each data record, until the weights are tuned to the local optimal point. The hyperparameters that we set were the number of epochs, the number of nodes, and the optimization function.

Model	Parameters							Average FDR at 3%		
	Iteration	Sampling Strategy (Majority:Minority)	optimizer	layers	nodes	epochs	activation	Train	Test	OOT
Neural Networks	1	No Sampling	adam	1	20	10	relu	55.22	54.46	52.18
	2	SMOTE (70:30)	adam	1	20	10	relu	55.60	54.73	52.56
	3	Undersample (70:30)	adam	1	20	10	relu	54.44	54.14	52.01
	4	SMOTE (70:30)	sgd	1	10	20	relu	54.47	54.01	52.05
	5	SMOTE (70:30)	sgd	1	30	20	relu	55.25	54.53	52.31
	6	SMOTE (70:30)	sgd	1	30	30	relu	55.25	54.53	52.26
	7	SMOTE (70:30)	adam	1	50	30	relu	55.59	54.73	53.19
	8	SMOTE (70:30)	adam	1	30	40	relu	55.59	55.09	52.72
	9	SMOTE (70:30)	adam	1	50	40	relu	55.72	55.12	53.02
	10	SMOTE (70:30)	adam	1	50	20	relu	55.61	54.76	53.23

Table 5.4 Neural Network Hyperparameter Performance Table



## Results

After creating 43 models across four different base algorithms, we discovered that our best performing model was an XGBoost algorithm with a 55.46% fraud detection rate at 3% for the test data set. The hyperparameters were set to:

- 600 Estimators
- Max depth of 3
- Learning rate of 0.1
- Random Subsample ratio at 0.8

The full model performance table can be found in the appendix.

Model	Parameters							Average FDR at 3%		
Logistic Regression	Iteration	Sampling Strategy (Majority:Minority)	penalty	C	solver			Train	Test	OOT
	1	No Sampling	l2	1	liblinear			52.51	52.14	50.00
XGBoost	Iteration	Sampling Strategy (Majority:Minority)	learning_rate	n_estimators	max_depth	subsample		Train	Test	OOT
	1	No Sampling	0.1	400	3	1		55.37	56.26	53.56
Random Forest	Iteration	Sampling Strategy (Majority:Minority)	criterion	n_estimators	max_depth	min_samples_split		Train	Test	OOT
	1	No Sampling	gini	100	20	300		55.59	55.06	53.35
Neural Networks	Iteration	Sampling Strategy (Majority:Minority)	optimizer	layers	nodes	epochs	activation	Train	Test	OOT
	1	No Sampling	adam	1	20	10	relu	55.22	54.46	52.18

**Table 6.1 Performance Table of the Four Best Models:** One iteration selected from each model

In the training set the top population bin was comprised of over 75.6% fraudulent records, with similar results across test, and out of time. This model helps ensure that investigators have to sift through minimal numbers of false positives.

Training	# Records	# Goods	# Bads	Fraud Rate								
	625130	616125	9005	0.01440500376								
Bin Statistics						Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	6250	1525	4725	24.40%	75.60%	6250	1525	4725	0.25%	52.47%	52.22	0.32
2	6251	6070	181	97.10%	2.90%	12501	7595	4906	1.23%	54.48%	53.25	1.55
3	6251	6171	80	98.72%	1.28%	18752	13766	4986	2.23%	55.37%	53.13	2.76
4	6252	6205	47	99.25%	0.75%	25004	19971	5033	3.24%	55.89%	52.65	3.97
5	6251	6195	56	99.10%	0.90%	31255	26166	5089	4.25%	56.51%	52.27	5.14
6	6251	6198	53	99.15%	0.85%	37506	32364	5142	5.25%	57.10%	51.85	6.29
7	6252	6213	39	99.38%	0.62%	43758	38577	5181	6.26%	57.53%	51.27	7.45
8	6251	6212	39	99.38%	0.62%	50009	44789	5220	7.27%	57.97%	50.70	8.58
9	6251	6213	38	99.39%	0.61%	56260	51002	5258	8.28%	58.39%	50.11	9.70
10	6251	6198	53	99.15%	0.85%	62511	57200	5311	9.28%	58.98%	49.69	10.77
11	6252	6207	45	99.28%	0.72%	68763	63407	5356	10.29%	59.48%	49.19	11.84
12	6251	6212	39	99.38%	0.62%	75014	69619	5395	11.30%	59.91%	48.61	12.90
13	6251	6196	55	99.12%	0.88%	81265	75815	5450	12.31%	60.52%	48.22	13.91
14	6252	6197	55	99.12%	0.88%	87517	82012	5505	13.31%	61.13%	47.82	14.90
15	6251	6201	50	99.20%	0.80%	93768	88213	5555	14.32%	61.69%	47.37	15.88
16	6251	6205	46	99.26%	0.74%	100019	94418	5601	15.32%	62.20%	46.87	16.86
17	6252	6209	43	99.31%	0.69%	106271	100627	5644	16.33%	62.68%	46.34	17.83
18	6251	6214	37	99.41%	0.59%	112522	106841	5681	17.34%	63.09%	45.75	18.81
19	6251	6205	46	99.26%	0.74%	118773	113046	5727	18.35%	63.60%	45.25	19.74
20	6251	6217	34	99.46%	0.54%	125024	119263	5761	19.36%	63.98%	44.62	20.70

**Table 6.2 Training Results**

Our process was validated by an equally impressive test dataset prediction, with excellent results across FDR, purity, and KS. Of the 208377 records, almost 1700 of the 3000 frauds were caught in the 1% population bin.

Testing	# Records	# Goods		# Bads		Fraud Rate						
	208377	205375		3002		0.01440658038						
	Bin Statistics					Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	2082	471	1611	22.62%	77.38%	2082	471	1611	0.23%	53.66%	53.43	0.29
2	2084	2034	50	97.60%	2.40%	4166	2505	1661	1.22%	55.33%	54.11	1.51
3	2084	2056	28	98.66%	1.34%	6250	4561	1689	2.22%	56.26%	54.04	2.70
4	2084	2064	20	99.04%	0.96%	8334	6625	1709	3.23%	56.93%	53.70	3.88
5	2083	2068	15	99.28%	0.72%	10417	8693	1724	4.23%	57.43%	53.20	5.04
6	2084	2070	14	99.33%	0.67%	12501	10763	1738	5.24%	57.89%	52.65	6.19
7	2084	2065	19	99.09%	0.91%	14585	12828	1757	6.25%	58.53%	52.28	7.30
8	2084	2069	15	99.28%	0.72%	16669	14897	1772	7.25%	59.03%	51.77	8.41
9	2083	2069	14	99.33%	0.67%	18752	16966	1786	8.26%	59.49%	51.23	9.50
10	2084	2068	16	99.23%	0.77%	20836	19034	1802	9.27%	60.03%	50.76	10.56
11	2084	2069	15	99.28%	0.72%	22920	21103	1817	10.28%	60.53%	50.25	11.61
12	2084	2072	12	99.42%	0.58%	25004	23175	1829	11.28%	60.93%	49.64	12.67
13	2084	2063	21	98.99%	1.01%	27088	25238	1850	12.29%	61.63%	49.34	13.64
14	2083	2067	16	99.23%	0.77%	29171	27305	1866	13.30%	62.16%	48.86	14.63
15	2084	2069	15	99.28%	0.72%	31255	29374	1881	14.30%	62.66%	48.36	15.62
16	2084	2063	21	98.99%	1.01%	33339	31437	1902	15.31%	63.36%	48.05	16.53
17	2084	2071	13	99.38%	0.62%	35423	33508	1915	16.32%	63.79%	47.48	17.50
18	2083	2063	20	99.04%	0.96%	37506	35571	1935	17.32%	64.46%	47.14	18.38
19	2084	2074	10	99.52%	0.48%	39590	37645	1945	18.33%	64.79%	46.46	19.35
20	2084	2069	15	99.28%	0.72%	41674	39714	1960	19.34%	65.29%	45.95	20.26

Table 6.3 Testing Results

To ensure our model would perform well on completely unseen data, we withheld the last 2 months of the year completely from model building. Our model still performed excellently on this out-of-time data with over 53.56% FDR at 3%.

OOT	# Records	# Goods	# Bads	Fraud Rate								
	163772	161427	2345	0.01431868696								
	Bin Statistics					Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	1636	442	1194	27.02%	72.98%	1636	442	1194	0.27%	50.92%	50.64	0.37
2	1638	1603	35	97.86%	2.14%	3274	2045	1229	1.27%	52.41%	51.14	1.66
3	1638	1611	27	98.35%	1.65%	4912	3656	1256	2.26%	53.56%	51.30	2.91
4	1637	1627	10	99.39%	0.61%	6549	5283	1266	3.27%	53.99%	50.71	4.17
5	1638	1629	9	99.45%	0.55%	8187	6912	1275	4.28%	54.37%	50.09	5.42
6	1638	1623	15	99.08%	0.92%	9825	8535	1290	5.29%	55.01%	49.72	6.62
7	1638	1628	10	99.39%	0.61%	11463	10163	1300	6.30%	55.44%	49.14	7.82
8	1637	1621	16	99.02%	0.98%	13100	11784	1316	7.30%	56.12%	48.82	8.95
9	1638	1627	11	99.33%	0.67%	14738	13411	1327	8.31%	56.59%	48.28	10.11
10	1638	1628	10	99.39%	0.61%	16376	15039	1337	9.32%	57.01%	47.70	11.25
11	1637	1623	14	99.14%	0.86%	18013	16662	1351	10.32%	57.61%	47.29	12.33
12	1638	1617	21	98.72%	1.28%	19651	18279	1372	11.32%	58.51%	47.18	13.32
13	1638	1625	13	99.21%	0.79%	21289	19904	1385	12.33%	59.06%	46.73	14.37
14	1638	1621	17	98.96%	1.04%	22927	21525	1402	13.33%	59.79%	46.45	15.35
15	1637	1624	13	99.21%	0.79%	24564	23149	1415	14.34%	60.34%	46.00	16.36
16	1638	1624	14	99.15%	0.85%	26202	24773	1429	15.35%	60.94%	45.59	17.34
17	1638	1623	15	99.08%	0.92%	27840	26396	1444	16.35%	61.58%	45.23	18.28
18	1637	1631	6	99.63%	0.37%	29477	28027	1450	17.36%	61.83%	44.47	19.33
19	1638	1630	8	99.51%	0.49%	31115	29657	1458	18.37%	62.17%	43.80	20.34
20	1638	1628	10	99.39%	0.61%	32753	31285	1468	19.38%	62.60%	43.22	21.31

Table 6.4 Out of Time Results

## Conclusions

Our project used a synthetic dataset of one million applications to build a real-time model for finding identity fraud. The original dataset contained various identity fields including social security number, first and last name, address, date of birth, and home phone number. Our first step was to replace frivolous fields, common values used to fill missing fields, with a unique value. Afterwards, we created attributes on different combinations of the identity fields to build over 350 candidate variables focusing on the relative velocity of occurrences and the frequency of specific combinations of the identity fields over time.

After creating our candidate variables, we used a two-step feature selection process to reduce our collection of variables. First, we ran a filter method, selecting variables with the highest average rank of KS and FDR at 3%, to narrow our list of variables to 80. Next, we used a backward feature selection process, with a base logistic regression model, to assign an initial ordering to our most important variables. Lastly, we ran our backward selection process again, without cross validation, to get a well-ranked list of the 30 most important variables.

Using these 30 variables, we created 43 unique models from four different binary classification algorithms: Logistic Regression, Random Forest, XGBoost, and Neural Networks. For each iteration, we tested different sampling strategies and hyperparameters with successful results determined by the highest fraud detection rate on our test dataset. The best performing model used XGBoost, a gradient boosting algorithm, with the hyperparameters tuned to 600 estimators, a max depth of 3, a learning rate of 0.1, and a random subsample ratio of 0.8. This final model produced a fraud detection rate of 53.56% at 3% of the population on the out of time data.

Given more time, we would have liked to create more candidate variables, specifically observing the uniqueness of different attributes and their relationships to application frequency. Another area of interest was to try a different feature selection technique, as RFECV did not work as we originally expected. Lastly, we would have liked to spend more time fine tuning our model's performance by training our model on different subsets containing the misclassified applications. However, our models ability to positively identify 53% of fraud for 3% of the population demonstrates a real business opportunity to utilize our algorithm as an effective way to catch identity fraud in real-time.

## Appendix

### DATA QUALITY REPORT FOR APPLICATIONS DATA

#### Description

**Dataset Name:** Product Application Data

**Dataset Purpose:** A synthetic dataset meant to mimic real application data from credit cards and cell phone applications. This dataset was created from analysis of over one billion real US applications over 10 years. The data has similar statistical properties to real data characteristics, such as frequency of similar fields and occurrences of relationships to other fields. The purpose of this dataset is to explore and identify application and identity fraud with only an applicant's identity fields.

**Data Source:** Synthetic Data (ID Analytics)

**Time Period:** 2016-01-01 – 2016-12-31

**Number of Fields:** #10

**Number of Records:** 1,000,000

#### Summary Table

Column Name	# Records	% Populated	Unique Values	Min	Max	Most Common Field Value
record	1,000,000	100	1000000	N/A	N/A	N/A
date	1,000,000	100	365	20160101	20161231	20160816
ssn	1,000,000	100	835819	N/A	N/A	999999999
firstname	1,000,000	100	78136	N/A	N/A	EAMSTRMT
lastname	1,000,000	100	177001	N/A	N/A	ERJSAXA
address	1,000,000	100	828774	N/A	N/A	123 MAIN ST
zip5	1,000,000	100	26370	N/A	N/A	68138
dob	1,000,000	100	42673	19000101	20161031	19070626
homephone	1,000,000	100	28244	N/A	N/A	999999999
fraud_label	1,000,000	100	2	N/A	N/A	0

#### Data Field Exploration:

Field 1:

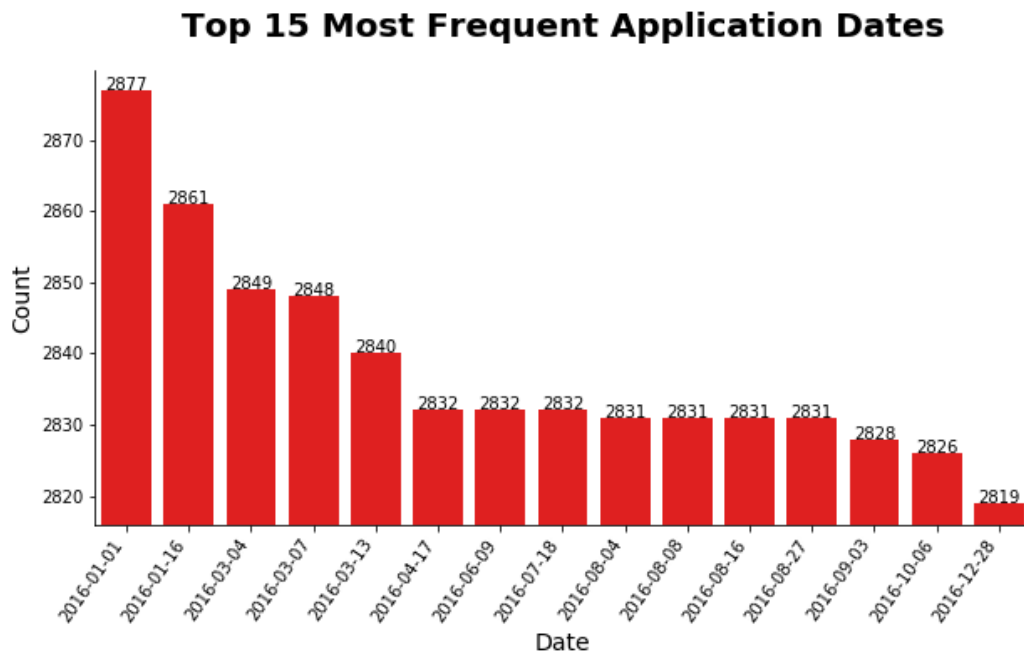
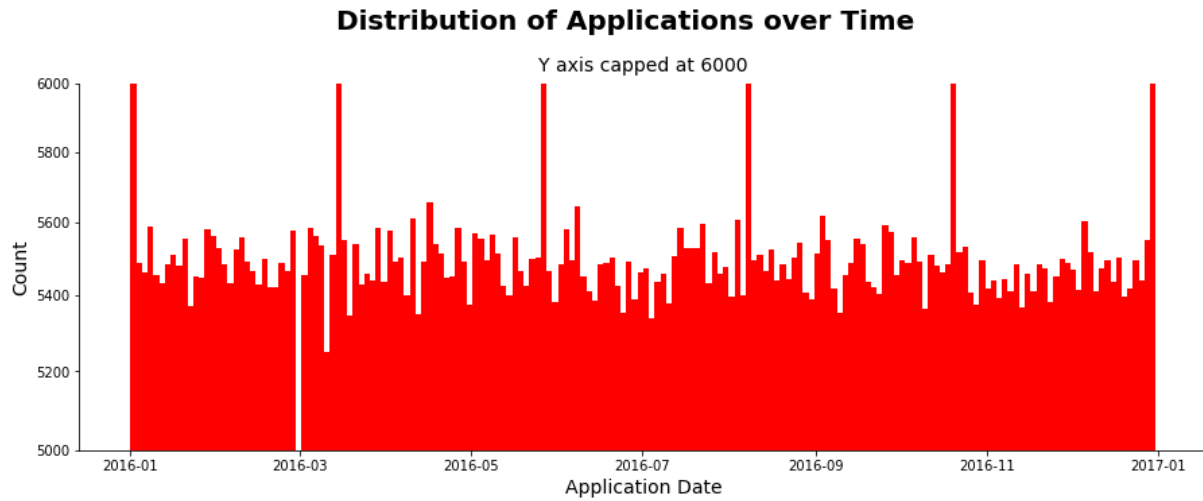
Name: record

Description: Unique identifier of each entry in the data.

Field 2:

Name: date

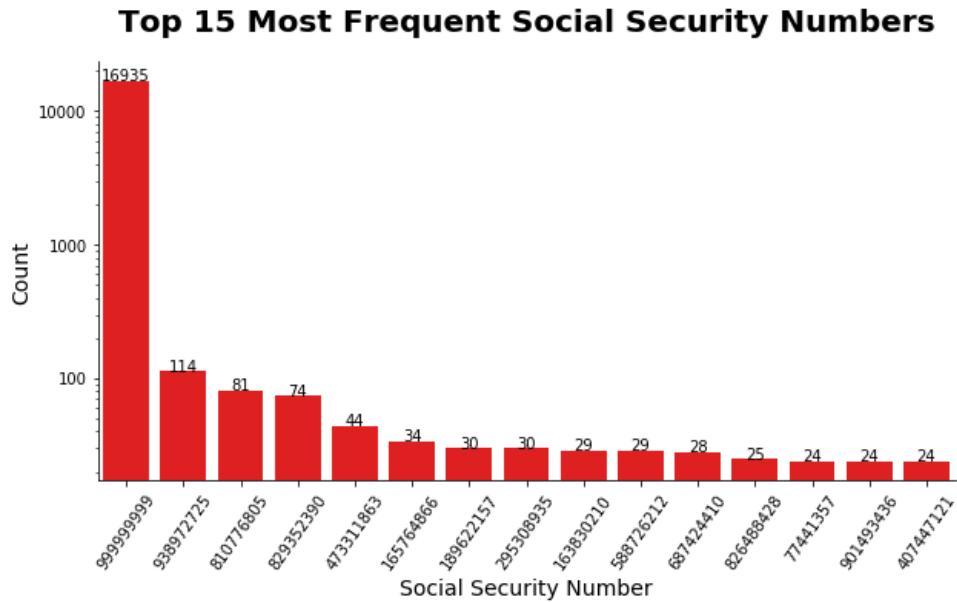
Description: Date of application



Field 3:

Name: ssn

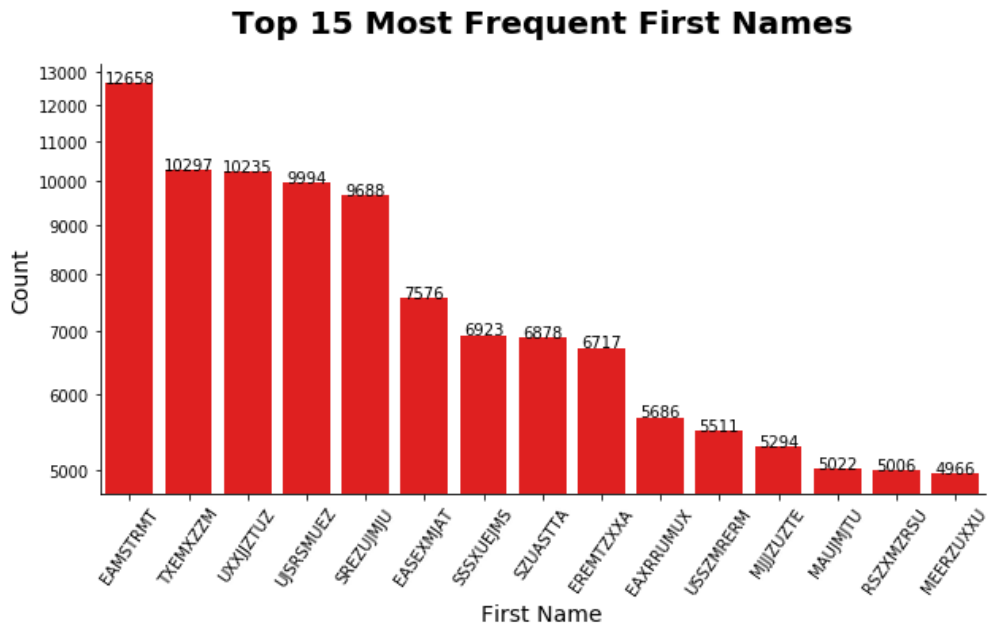
Description: Social security number of the applicant



Field 4:

Name: firstname

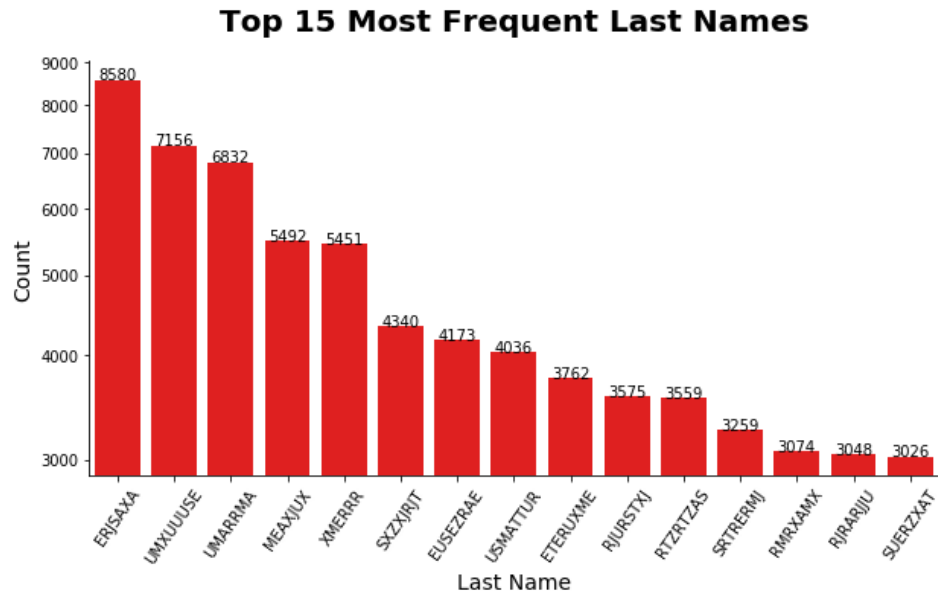
Description: First name of the applicant



Field 5:

Name: lastname

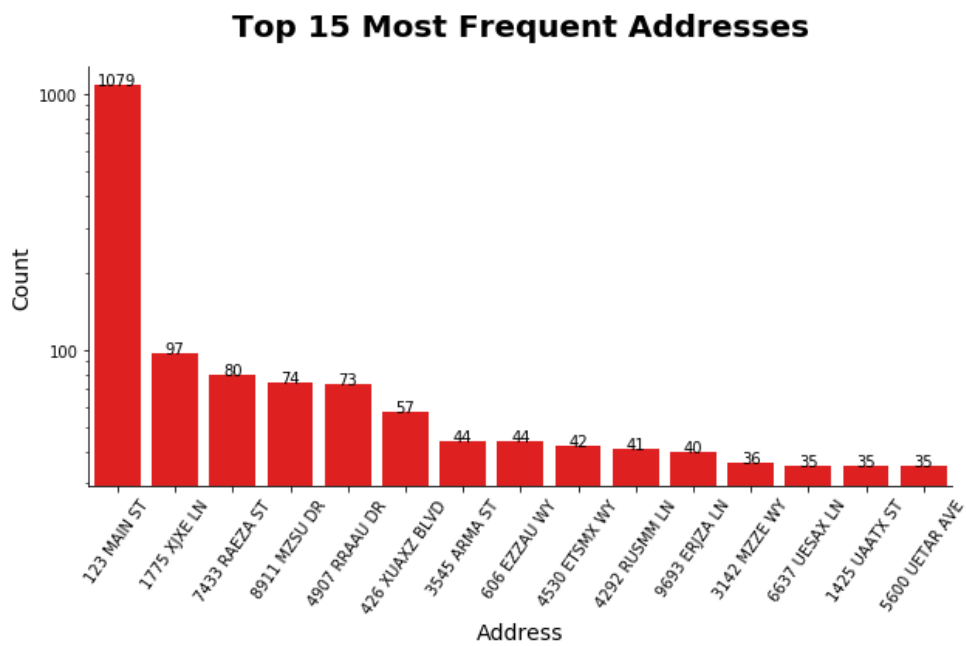
Description: Last name of the applicant



Field 6:

Name: address

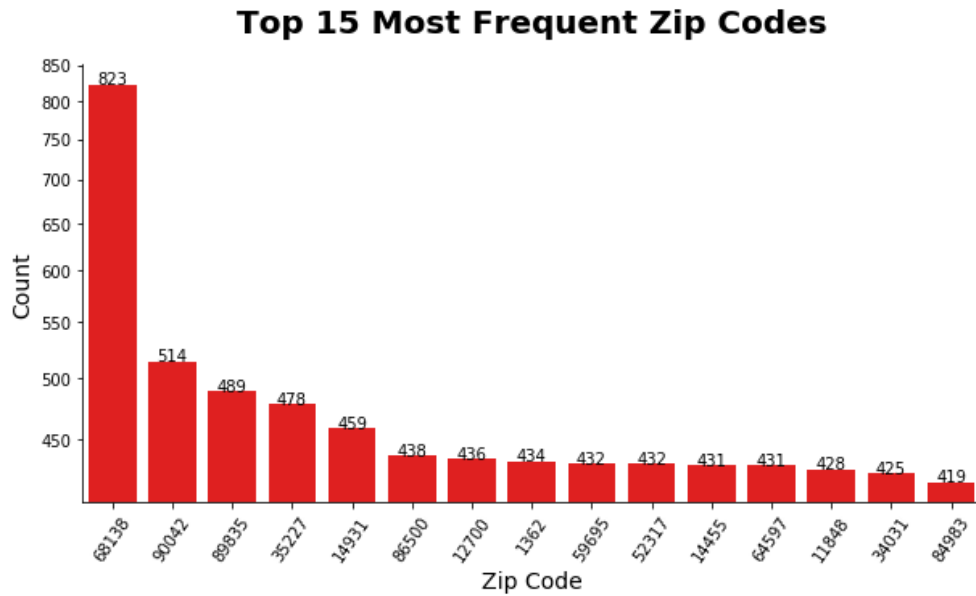
Description: Address of the applicant



Field 7:

Name: zip5

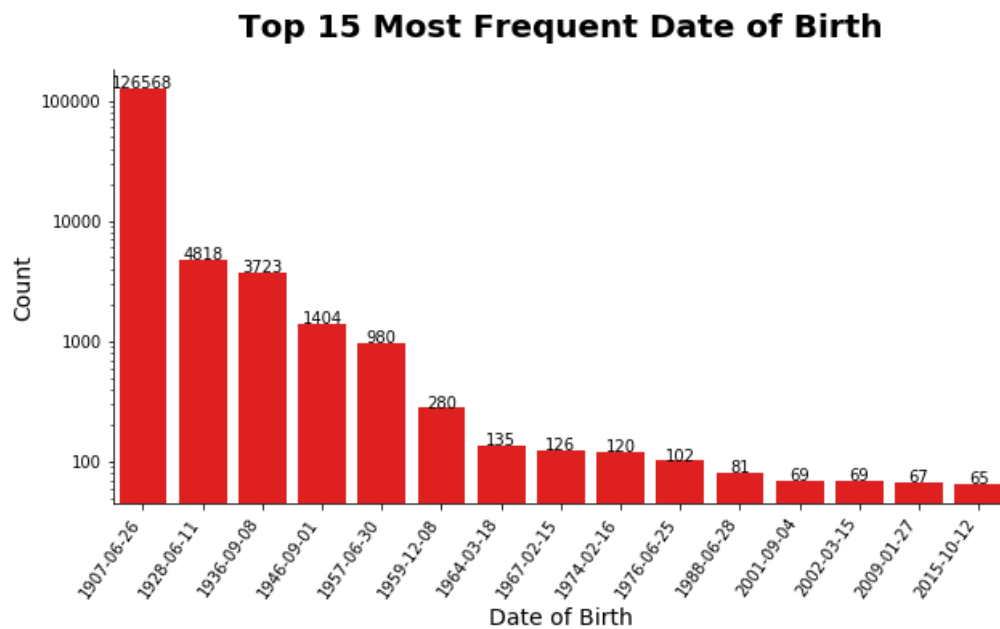
Description: Postal code of the applicant



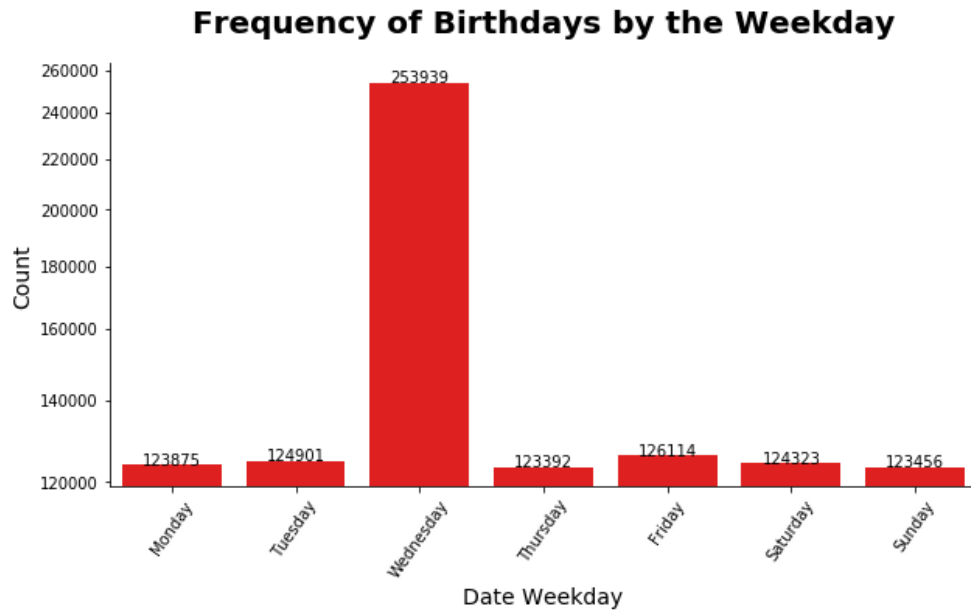
Field 8:

Name: dob

Description: Date of birth of the applicant



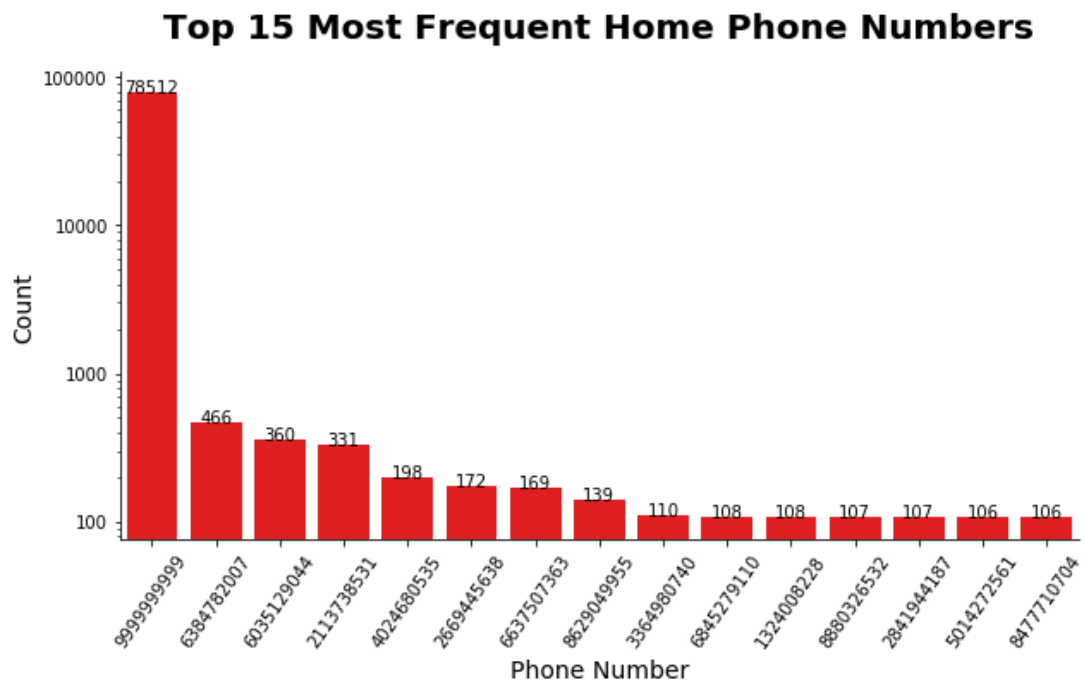




Field 9:

Name: homephone

Description: Home phone number of the applicant



Field 10:

Name: fraud\_label

Description: Binary label for each record in the data to classify fraud vs no fraud



## Candidate Variables

Index	Variable Name	Index	Variable Name
1	age	50	name_fulladdress_count_3
2	age_bin	51	name_fulladdress_count_7
3	dow_risk	52	name_fulladdress_count_14
4	age_bin_risk	53	name_fulladdress_count_30
5	ssn_day_since	54	name_homephone_day_since
6	ssn_count_0	55	name_homephone_count_0
7	ssn_count_1	56	name_homephone_count_1
8	ssn_count_3	57	name_homephone_count_3
9	ssn_count_7	58	name_homephone_count_7
10	ssn_count_14	59	name_homephone_count_14
11	ssn_count_30	60	name_homephone_count_30
12	address_day_since	61	fulladdress_dob_day_since
13	address_count_0	62	fulladdress_dob_count_0
14	address_count_1	63	fulladdress_dob_count_1
15	address_count_3	64	fulladdress_dob_count_3
16	address_count_7	65	fulladdress_dob_count_7
17	address_count_14	66	fulladdress_dob_count_14
18	address_count_30	67	fulladdress_dob_count_30
19	dob_day_since	68	fulladdress_homephone_day_since
20	dob_count_0	69	fulladdress_homephone_count_0
21	dob_count_1	70	fulladdress_homephone_count_1
22	dob_count_3	71	fulladdress_homephone_count_3
23	dob_count_7	72	fulladdress_homephone_count_7
24	dob_count_14	73	fulladdress_homephone_count_14
25	dob_count_30	74	fulladdress_homephone_count_30
26	homephone_day_since	75	dob_homephone_day_since
27	homephone_count_0	76	dob_homephone_count_0
28	homephone_count_1	77	dob_homephone_count_1
29	homephone_count_3	78	dob_homephone_count_3
30	homephone_count_7	79	dob_homephone_count_7
31	homephone_count_14	80	dob_homephone_count_14
32	homephone_count_30	81	dob_homephone_count_30
33	fulladdress_day_since	82	homephone_name_dob_day_since
34	fulladdress_count_0	83	homephone_name_dob_count_0
35	fulladdress_count_1	84	homephone_name_dob_count_1
36	fulladdress_count_3	85	homephone_name_dob_count_3
37	fulladdress_count_7	86	homephone_name_dob_count_7
38	fulladdress_count_14	87	homephone_name_dob_count_14
39	fulladdress_count_30	88	homephone_name_dob_count_30
40	name_dob_day_since	89	areacode_name_fulladdress_day_since
41	name_dob_count_0	90	areacode_name_fulladdress_count_0
42	name_dob_count_1	91	areacode_name_fulladdress_count_1
43	name_dob_count_3	92	areacode_name_fulladdress_count_3
44	name_dob_count_7	93	areacode_name_fulladdress_count_7
45	name_dob_count_14	94	areacode_name_fulladdress_count_14
46	name_dob_count_30	95	areacode_name_fulladdress_count_30
47	name_fulladdress_day_since	96	ssn_firstname_day_since
48	name_fulladdress_count_0	97	ssn_firstname_count_0
49	name_fulladdress_count_1	98	ssn_firstname_count_1

Index	Variable Name	Index	Variable Name
99	ssn_firstname_count_3	148	ssn_name_fulladdress_count_3
100	ssn_firstname_count_7	149	ssn_name_fulladdress_count_7
101	ssn_firstname_count_14	150	ssn_name_fulladdress_count_14
102	ssn_firstname_count_30	151	ssn_name_fulladdress_count_30
103	ssn_lastname_day_since	152	ssn_name_homephone_day_since
104	ssn_lastname_count_0	153	ssn_name_homephone_count_0
105	ssn_lastname_count_1	154	ssn_name_homephone_count_1
106	ssn_lastname_count_3	155	ssn_name_homephone_count_3
107	ssn_lastname_count_7	156	ssn_name_homephone_count_7
108	ssn_lastname_count_14	157	ssn_name_homephone_count_14
109	ssn_lastname_count_30	158	ssn_name_homephone_count_30
110	ssn_address_day_since	159	ssn_fulladdress_dob_day_since
111	ssn_address_count_0	160	ssn_fulladdress_dob_count_0
112	ssn_address_count_1	161	ssn_fulladdress_dob_count_1
113	ssn_address_count_3	162	ssn_fulladdress_dob_count_3
114	ssn_address_count_7	163	ssn_fulladdress_dob_count_7
115	ssn_address_count_14	164	ssn_fulladdress_dob_count_14
116	ssn_address_count_30	165	ssn_fulladdress_dob_count_30
117	ssn_zip5_day_since	166	ssn_fulladdress_homephone_day_since
118	ssn_zip5_count_0	167	ssn_fulladdress_homephone_count_0
119	ssn_zip5_count_1	168	ssn_fulladdress_homephone_count_1
120	ssn_zip5_count_3	169	ssn_fulladdress_homephone_count_3
121	ssn_zip5_count_7	170	ssn_fulladdress_homephone_count_7
122	ssn_zip5_count_14	171	ssn_fulladdress_homephone_count_14
123	ssn_zip5_count_30	172	ssn_fulladdress_homephone_count_30
124	ssn_dob_day_since	173	ssn_dob_homephone_day_since
125	ssn_dob_count_0	174	ssn_dob_homephone_count_0
126	ssn_dob_count_1	175	ssn_dob_homephone_count_1
127	ssn_dob_count_3	176	ssn_dob_homephone_count_3
128	ssn_dob_count_7	177	ssn_dob_homephone_count_7
129	ssn_dob_count_14	178	ssn_dob_homephone_count_14
130	ssn_dob_count_30	179	ssn_dob_homephone_count_30
131	ssn_fulladdress_day_since	180	ssn_homephone_name_dob_day_since
132	ssn_fulladdress_count_0	181	ssn_homephone_name_dob_count_0
133	ssn_fulladdress_count_1	182	ssn_homephone_name_dob_count_1
134	ssn_fulladdress_count_3	183	ssn_homephone_name_dob_count_3
135	ssn_fulladdress_count_7	184	ssn_homephone_name_dob_count_7
136	ssn_fulladdress_count_14	185	ssn_homephone_name_dob_count_14
137	ssn_fulladdress_count_30	186	ssn_homephone_name_dob_count_30
138	ssn_name_dob_day_since	187	ssn_count_0_by_3
139	ssn_name_dob_count_0	188	ssn_count_0_by_7
140	ssn_name_dob_count_1	189	ssn_count_0_by_14
141	ssn_name_dob_count_3	190	ssn_count_0_by_30
142	ssn_name_dob_count_7	191	ssn_count_1_by_3
143	ssn_name_dob_count_14	192	ssn_count_1_by_7
144	ssn_name_dob_count_30	193	ssn_count_1_by_14
145	ssn_name_fulladdress_day_since	194	ssn_count_1_by_30
146	ssn_name_fulladdress_count_0	195	address_count_0_by_3
147	ssn_name_fulladdress_count_1	196	address_count_0_by_7

Index	Variable Name	Index	Variable Name
197	address_count_0_by_14	246	name_homephone_count_0_by_30
198	address_count_0_by_30	247	name_homephone_count_1_by_3
199	address_count_1_by_3	248	name_homephone_count_1_by_7
200	address_count_1_by_7	249	name_homephone_count_1_by_14
201	address_count_1_by_14	250	name_homephone_count_1_by_30
202	address_count_1_by_30	251	fulladdress_dob_count_0_by_3
203	dob_count_0_by_3	252	fulladdress_dob_count_0_by_7
204	dob_count_0_by_7	253	fulladdress_dob_count_0_by_14
205	dob_count_0_by_14	254	fulladdress_dob_count_0_by_30
206	dob_count_0_by_30	255	fulladdress_dob_count_1_by_3
207	dob_count_1_by_3	256	fulladdress_dob_count_1_by_7
208	dob_count_1_by_7	257	fulladdress_dob_count_1_by_14
209	dob_count_1_by_14	258	fulladdress_dob_count_1_by_30
210	dob_count_1_by_30	259	fulladdress_homephone_count_0_by_3
211	homephone_count_0_by_3	260	fulladdress_homephone_count_0_by_7
212	homephone_count_0_by_7	261	fulladdress_homephone_count_0_by_14
213	homephone_count_0_by_14	262	fulladdress_homephone_count_0_by_30
214	homephone_count_0_by_30	263	fulladdress_homephone_count_1_by_3
215	homephone_count_1_by_3	264	fulladdress_homephone_count_1_by_7
216	homephone_count_1_by_7	265	fulladdress_homephone_count_1_by_14
217	homephone_count_1_by_14	266	fulladdress_homephone_count_1_by_30
218	homephone_count_1_by_30	267	dob_homephone_count_0_by_3
219	fulladdress_count_0_by_3	268	dob_homephone_count_0_by_7
220	fulladdress_count_0_by_7	269	dob_homephone_count_0_by_14
221	fulladdress_count_0_by_14	270	dob_homephone_count_0_by_30
222	fulladdress_count_0_by_30	271	dob_homephone_count_1_by_3
223	fulladdress_count_1_by_3	272	dob_homephone_count_1_by_7
224	fulladdress_count_1_by_7	273	dob_homephone_count_1_by_14
225	fulladdress_count_1_by_14	274	dob_homephone_count_1_by_30
226	fulladdress_count_1_by_30	275	homephone_name_dob_count_0_by_3
227	name_dob_count_0_by_3	276	homephone_name_dob_count_0_by_7
228	name_dob_count_0_by_7	277	homephone_name_dob_count_0_by_14
229	name_dob_count_0_by_14	278	homephone_name_dob_count_0_by_30
230	name_dob_count_0_by_30	279	homephone_name_dob_count_1_by_3
231	name_dob_count_1_by_3	280	homephone_name_dob_count_1_by_7
232	name_dob_count_1_by_7	281	homephone_name_dob_count_1_by_14
233	name_dob_count_1_by_14	282	homephone_name_dob_count_1_by_30
234	name_dob_count_1_by_30	283	areacode_name_fulladdress_count_0_by_3
235	name_fulladdress_count_0_by_3	284	areacode_name_fulladdress_count_0_by_7
236	name_fulladdress_count_0_by_7	285	areacode_name_fulladdress_count_0_by_14
237	name_fulladdress_count_0_by_14	286	areacode_name_fulladdress_count_0_by_30
238	name_fulladdress_count_0_by_30	287	areacode_name_fulladdress_count_1_by_3
239	name_fulladdress_count_1_by_3	288	areacode_name_fulladdress_count_1_by_7
240	name_fulladdress_count_1_by_7	289	areacode_name_fulladdress_count_1_by_14
241	name_fulladdress_count_1_by_14	290	areacode_name_fulladdress_count_1_by_30
242	name_fulladdress_count_1_by_30	291	ssn_firstname_count_0_by_3
243	name_homephone_count_0_by_3	292	ssn_firstname_count_0_by_7
244	name_homephone_count_0_by_7	293	ssn_firstname_count_0_by_14
245	name_homephone_count_0_by_14	294	ssn_firstname_count_0_by_30

Index	Variable Name	Index	Variable Name
295	ssn_firstname_count_1_by_3	344	ssn_name_dob_count_1_by_7
296	ssn_firstname_count_1_by_7	345	ssn_name_dob_count_1_by_14
297	ssn_firstname_count_1_by_14	346	ssn_name_dob_count_1_by_30
298	ssn_firstname_count_1_by_30	347	ssn_name_fulladdress_count_0_by_3
299	ssn_lastname_count_0_by_3	348	ssn_name_fulladdress_count_0_by_7
300	ssn_lastname_count_0_by_7	349	ssn_name_fulladdress_count_0_by_14
301	ssn_lastname_count_0_by_14	350	ssn_name_fulladdress_count_0_by_30
302	ssn_lastname_count_0_by_30	351	ssn_name_fulladdress_count_1_by_3
303	ssn_lastname_count_1_by_3	352	ssn_name_fulladdress_count_1_by_7
304	ssn_lastname_count_1_by_7	353	ssn_name_fulladdress_count_1_by_14
305	ssn_lastname_count_1_by_14	354	ssn_name_fulladdress_count_1_by_30
306	ssn_lastname_count_1_by_30	355	ssn_name_homephone_count_0_by_3
307	ssn_address_count_0_by_3	356	ssn_name_homephone_count_0_by_7
308	ssn_address_count_0_by_7	357	ssn_name_homephone_count_0_by_14
309	ssn_address_count_0_by_14	358	ssn_name_homephone_count_0_by_30
310	ssn_address_count_0_by_30	359	ssn_name_homephone_count_1_by_3
311	ssn_address_count_1_by_3	360	ssn_name_homephone_count_1_by_7
312	ssn_address_count_1_by_7	361	ssn_name_homephone_count_1_by_14
313	ssn_address_count_1_by_14	362	ssn_name_homephone_count_1_by_30
314	ssn_address_count_1_by_30	363	ssn_fulladdress_dob_count_0_by_3
315	ssn_zip5_count_0_by_3	364	ssn_fulladdress_dob_count_0_by_7
316	ssn_zip5_count_0_by_7	365	ssn_fulladdress_dob_count_0_by_14
317	ssn_zip5_count_0_by_14	366	ssn_fulladdress_dob_count_0_by_30
318	ssn_zip5_count_0_by_30	367	ssn_fulladdress_dob_count_1_by_3
319	ssn_zip5_count_1_by_3	368	ssn_fulladdress_dob_count_1_by_7
320	ssn_zip5_count_1_by_7	369	ssn_fulladdress_dob_count_1_by_14
321	ssn_zip5_count_1_by_14	370	ssn_fulladdress_dob_count_1_by_30
322	ssn_zip5_count_1_by_30	371	ssn_fulladdress_homephone_count_0_by_3
323	ssn_dob_count_0_by_3	372	ssn_fulladdress_homephone_count_0_by_7
324	ssn_dob_count_0_by_7	373	ssn_fulladdress_homephone_count_0_by_14
325	ssn_dob_count_0_by_14	374	ssn_fulladdress_homephone_count_0_by_30
326	ssn_dob_count_0_by_30	375	ssn_fulladdress_homephone_count_1_by_3
327	ssn_dob_count_1_by_3	376	ssn_fulladdress_homephone_count_1_by_7
328	ssn_dob_count_1_by_7	377	ssn_fulladdress_homephone_count_1_by_14
329	ssn_dob_count_1_by_14	378	ssn_fulladdress_homephone_count_1_by_30
330	ssn_dob_count_1_by_30	379	ssn_dob_homephone_count_0_by_3
331	ssn_fulladdress_count_0_by_3	380	ssn_dob_homephone_count_0_by_7
332	ssn_fulladdress_count_0_by_7	381	ssn_dob_homephone_count_0_by_14
333	ssn_fulladdress_count_0_by_14	382	ssn_dob_homephone_count_0_by_30
334	ssn_fulladdress_count_0_by_30	383	ssn_dob_homephone_count_1_by_3
335	ssn_fulladdress_count_1_by_3	384	ssn_dob_homephone_count_1_by_7
336	ssn_fulladdress_count_1_by_7	385	ssn_dob_homephone_count_1_by_14
337	ssn_fulladdress_count_1_by_14	386	ssn_dob_homephone_count_1_by_30
338	ssn_fulladdress_count_1_by_30	387	ssn_homephone_name_dob_count_0_by_3
339	ssn_name_dob_count_0_by_3	388	ssn_homephone_name_dob_count_0_by_7
340	ssn_name_dob_count_0_by_7	389	ssn_homephone_name_dob_count_0_by_14
341	ssn_name_dob_count_0_by_14	390	ssn_homephone_name_dob_count_0_by_30
342	ssn_name_dob_count_0_by_30	391	ssn_homephone_name_dob_count_1_by_3
343	ssn_name_dob_count_1_by_3	392	ssn_homephone_name_dob_count_1_by_7
		393	ssn_homephone_name_dob_count_1_by_14
		394	ssn_homephone_name_dob_count_1_by_30

## Model Hyperparameters Performance

Model	Parameters						Average FDR at 3%			
Logistic Regression	Iteration	Sampling Strategy (Majority:Minority)	penalty	C	solver		Train	Test	OOT	
	1	No Sampling	l2	1	liblinear		52.51	52.14	50.00	
	2	Class weight Argument (10:1)	l2	1	liblinear		52.81	52.48	50.84	
	3	SMOTE (70:30)	l2	1	liblinear		53.50	53.15	51.30	
	4	SMOTE (60:40)	l2	1	liblinear		53.94	53.41	51.30	
	5	SMOTE (50:50)	l2	1	liblinear		53.95	53.41	51.30	
	6	Undersample (70:30)	l2	1	liblinear		54.43	53.92	52.14	
	7	Undersample (60:40)	l2	1	liblinear		54.20	53.88	52.14	
	8	Undersample (50:50)	l2	1	liblinear		54.53	53.85	52.10	
	9	No Sampling	l2	0.1	saga		54.55	53.82	51.72	
	10	Undersample (70:30)	l2	0.1	saga		54.65	53.92	52.10	
	11	SMOTE (70:30)	l2	0.1	saga		54.14	53.55	51.26	
	12	No Sampling	l2	0.01	lbfgs		53.33	52.54	50.80	
	13	Undersample (70:30)	l2	0.01	lbfgs		53.89	53.18	51.34	
XGBoost	Iteration	Sampling Strategy (Majority:Minority)	learning_rate	n_estimators	max_depth	subsample	Train	Test	OOT	
	1	No Sampling	0.1	400	3	1	57.61	55.22	53.31	
	2	Scale Pos Weight = 10	0.1	400	3	1	62.56	55.02	53.02	
	3	SMOTE (70:30)	0.1	400	3	1	58.28	54.89	49.75	
	4	Undersample (70:30)	0.1	400	3	1	57.33	54.89	52.85	
	5	No Sampling	0.05	500	5	0.7	56.92	55.35	53.1	
	6	No Sampling	0.1	750	5	0.8	59.49	55.02	52.68	
	7	No Sampling	0.1	750	5	1	58.54	54.92	53.1	
	8	No Sampling	0.01	1000	3	1	55.65	55.29	53.25	
	9	No Sampling	0.1	600	3	0.8	57.04	55.46	53.19	
10	No Sampling	0.05	600	5	1	57.11	55.05	53.44		
Random Forest	Iteration	Sampling Strategy (Majority:Minority)	criterion	n_estimators	max_depth	min_samples_split	Train	Test	OOT	
	1	No Sampling	gini	100	20	300	55.59	55.06	53.35	
	2	Class Weight Argument (10:1)	gini	100	20	300	55.66	55.09	53.56	
	3	SMOTE (70:30)	gini	100	20	300	55.75	55.09	51.55	
	4	Undersample (70:30)	gini	100	20	300	55.38	54.6	52.98	
	5	Class Weight Argument (10:1)	entropy	150	10	200	55.51	54.76	53.35	
	6	Class Weight Argument (10:1)	entropy	150	25	300	55.73	55.06	53.48	
	7	Class Weight Argument (10:1)	entropy	150	30	400	55.67	55.12	53.52	
	8	Class Weight Argument (10:1)	gini	100	20	400	55.71	55.12	53.52	
	9	Class Weight Argument (10:1)	gini	50	25	300	55.65	55.06	53.44	
10	Class Weight Argument (10:1)	gini	50	25	300	55.69	54.99	53.56		
Neural Networks	Iteration	Sampling Strategy (Majority:Minority)	optimizer	layers	nodes	epochs	activation	Train	Test	OOT
	1	No Sampling	adam	1	20	10	relu	55.22	54.46	52.18
	2	SMOTE (70:30)	adam	1	20	10	relu	55.6	54.73	52.56
	3	Undersample (70:30)	adam	1	20	10	relu	54.44	54.14	52.01
	4	SMOTE (70:30)	sgd	1	10	20	relu	54.47	54.01	52.05
	5	SMOTE (70:30)	sgd	1	30	20	relu	55.25	54.53	52.31
	6	SMOTE (70:30)	sgd	1	30	30	relu	55.25	54.53	52.26
	7	SMOTE (70:30)	adam	1	50	30	relu	55.59	54.73	53.19
	8	SMOTE (70:30)	adam	1	30	40	relu	55.59	55.09	52.72
	9	SMOTE (70:30)	adam	1	50	40	relu	55.72	55.12	53.02
10	SMOTE (70:30)	adam	1	50	20	relu	55.61	54.76	53.23	