Exploring Identity Theft Fraud Through Machine Learning Algorithms

Ziyi Gao, Bingxin Li, David Shin,
Yamato Tadokoro, Lizhu Wang, Zhiyu Zhang

DSO 562 Fraud Analytics

Professor Stephen Cogeshall

March 24, 2022

Table of Contents

Executive Summary **3**

1. Description of Data **4**

1.1 Categorical Fields Summary 4

1.2 Numerical Fields Summary 5

1.3 Field Distributions 6

2. Data Cleaning **9**

2.1 Date Field Reformatting 9

2.2 SSN Frivolous Values 9

2.3 Address Frivolous Values 9

2.4 DOB Frivolous Values 10

2.5 Homephone Frivolous Values 10

3. Create Candidate Variables **11**

3.1 Linking Variables 11

3.2 Velocity 12

3.3 Days since last seen 12

3.4 Relative velocity 12

3.5 Target Encoding 12

3.5.1 Day of Week Field 12

3.5.2 Age_Level Field 12

4. Feature Selection Process **14**

4.1 Kolmogorov-Smirnov filter 14

4.2 Wrapper – Forward Selection 14

4.3 Final 30 Variables with Their Descriptions 15

5. Model Algorithms **19**

5.1 Four Models 20

5.1.1 Logistic Regression 20

5.1.2 Decision Tree 21

5.1.3 Extreme Gradient Boosting (XG-Boost) 22

5.1.4 Neural Network 24

5.2 Model Comparison with 8 variables vs 25 variables 25

6. Result **26**

7. Conclusion **29**

Executive Summary

This project explores a synthetic dataset of 1,000,000 credit card and cell phone applications, and builds a real-time supervised model based on those applications to help companies identify possible application frauds in real time. For the model building, we followed the following steps:

- Data description: We examined the dataset through identifying each variable, calculating minimum, maximum, percentage of null values for the numeric variables, and calculating the number of unique values, and the most common field value for the categorical variables. After the data examination, we wrote a Data Quality Report (DQR) for the dataset.

- Data cleaning: To better prepare the data for model building, we cleaned and organized the data. The first focus was on fixing the data type for the date and zip5 fields. We also fixed frivolous values for the ssn, address, dob (date of birth), and homephone fields. Lastly, we did target encoding for the date field.

- Variable creation: Before the model building and selection, we created as many variables as possible to better capture all possible variables that can be used to build a good predictive model to detect the fraud applications in the future. To achieve this, we created several new fields and also combined some fields together as new variables.

- Feature selection: From the various variables created in the previous step, we selected the 30 best performing variables to further build models and make model selections. For the selection process, we first used the KS method to select 100 variables and used wrapper–forward selection to select the top 30 best variables for the next step.

- Model algorithms: We tried 4 models: Logistic Regression, Decision Tree, Extreme Gradient Boosting Algorithm (XGboost), and Neural Network to find the best predicting model. For each model, we changed hyperparameters to better fit the dataset and compared their performances to select the best model.

- Results: After testing all models and making comparisons, we chose the best performing model with the highest fraud detection rate (FDR) at 3%. We found that the XGboost model had the best performance.

1. Description of Data

The data used in this project is a synthetic dataset of credit card and cell phone applications created for academic purposes. The dataset was made based on analysis done on real U.S. application data over a 10-year period, meaning that it imitates both the format and distributions of real data. The dataset has the following 10 fields:

1. record – The record number or unique index of the dataset
2. date – The date that the application was filled on. All dates are in 2016
3. ssn – The social security number of the applicant
4. firstname – The first name of the applicant
5. lastname – The last name of the applicant
6. address – The listed address on the application
7. zip5 – The zip code of the listed address
8. dob – The date of birth of the applicant
9. homephone – The listed home phone number on the application
10. fraud_label – Indicates whether the application was detected as fraud | 0 = not, 1 = yes

1.1 Categorical Fields Summary

Field Name	% Populated	# Unique Values	Mode
record	100%	1,000,000	1
ssn	100%	835,819	999999999
firstname	100%	78,136	EAMSTRMT
lastname	100%	177,001	ERJSAXA
address	100%	828,774	123 MAIN ST
zip5	100%	26,370	68138
homephone	100%	28,244	9999999999
fraud_label	100%	2	0

1.2 Numerical Fields Summary

Field Name	% Populated	Min	Max	Mode	Mean	Stdev	% Zero
date	100%	2016-01-01	2016-12-31	2016-08-16	N/A	N/A	0
dob	100%	1900-01-01	2016-10-31	1907-06-26	N/A	N/A	0

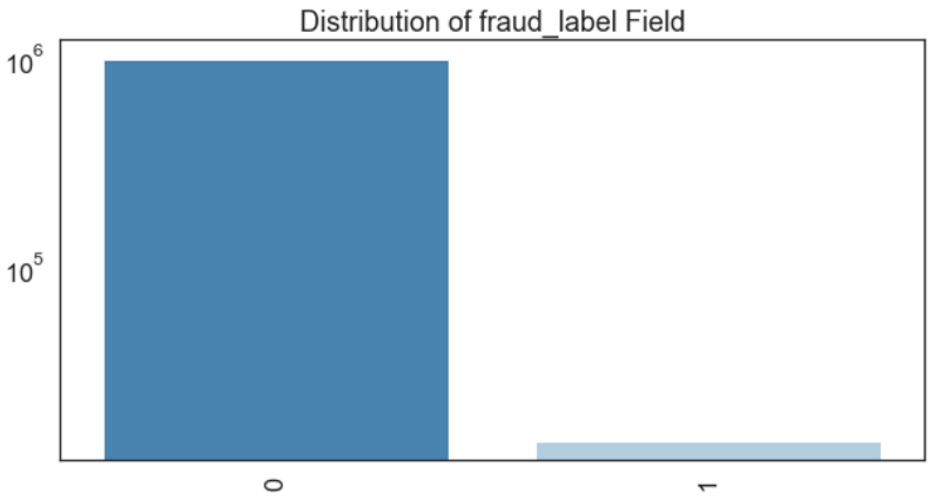


Figure 1. Count-plot of fraud_label Field (scale: log)

There are 14,393 rows in the data with a fraud label of 1, meaning that about 1.4% of all applications were detected as fraud. This means that 985,607 or 98.6% of all applications were not labeled as fraud.

According to the UTICA University Center for Identity Management and Information Protection, the most common identity crimes begin when your name, Social Security Number, date of birth, maiden name or address are exposed(Utica.edu). These fields are undoubtedly part of the most commonly asked for details on applications. The distributions of the top 25 values for the ssn, firstname, last name, address and dob columns for our data set are shown in the following pages.

1.3 Field Distributions

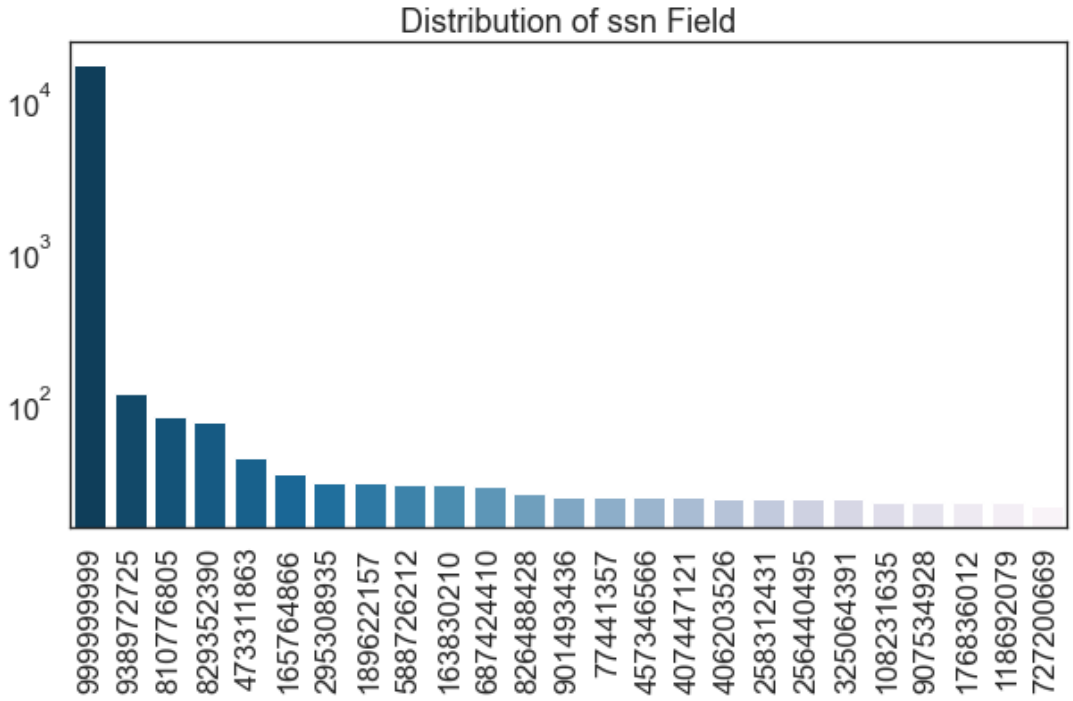


Figure 2. Count-plot of ssn Field (scale: log)

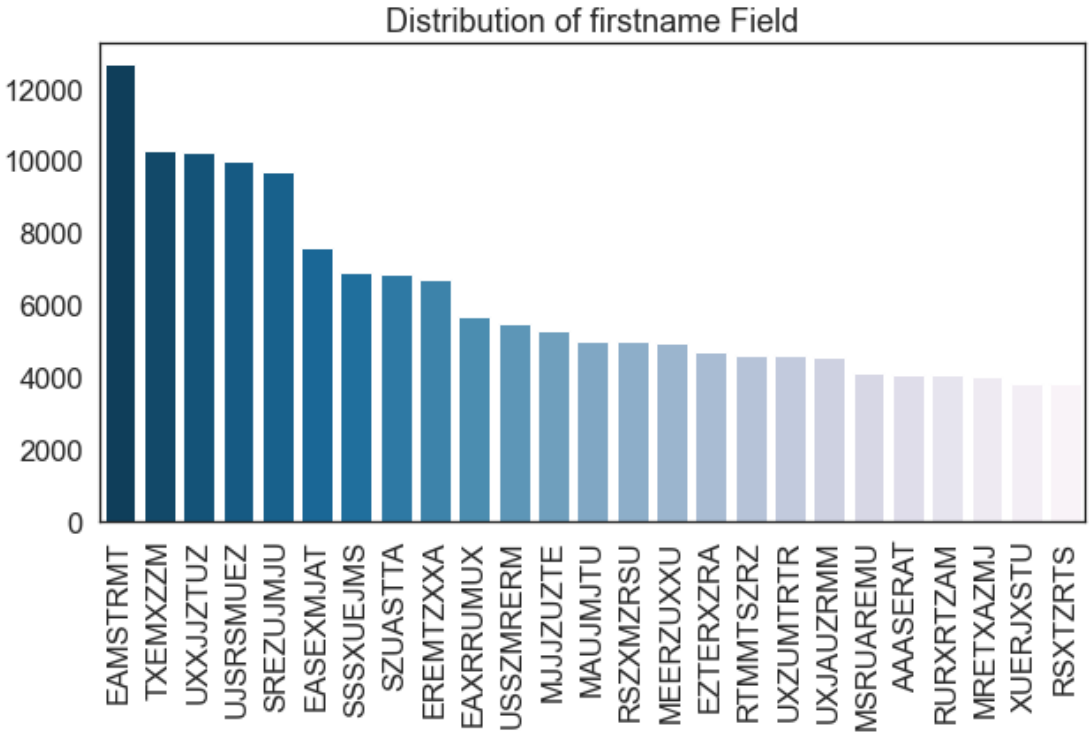


Figure 3. Count-plot of firstname Field

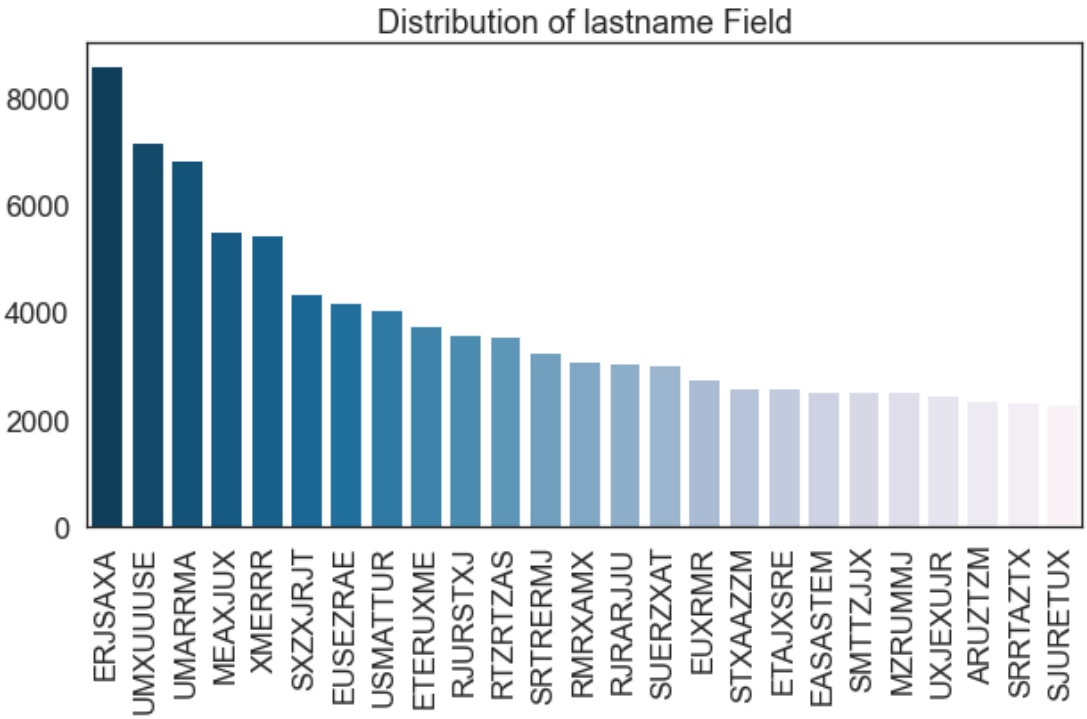


Figure 4. Count-plot of lastname Field

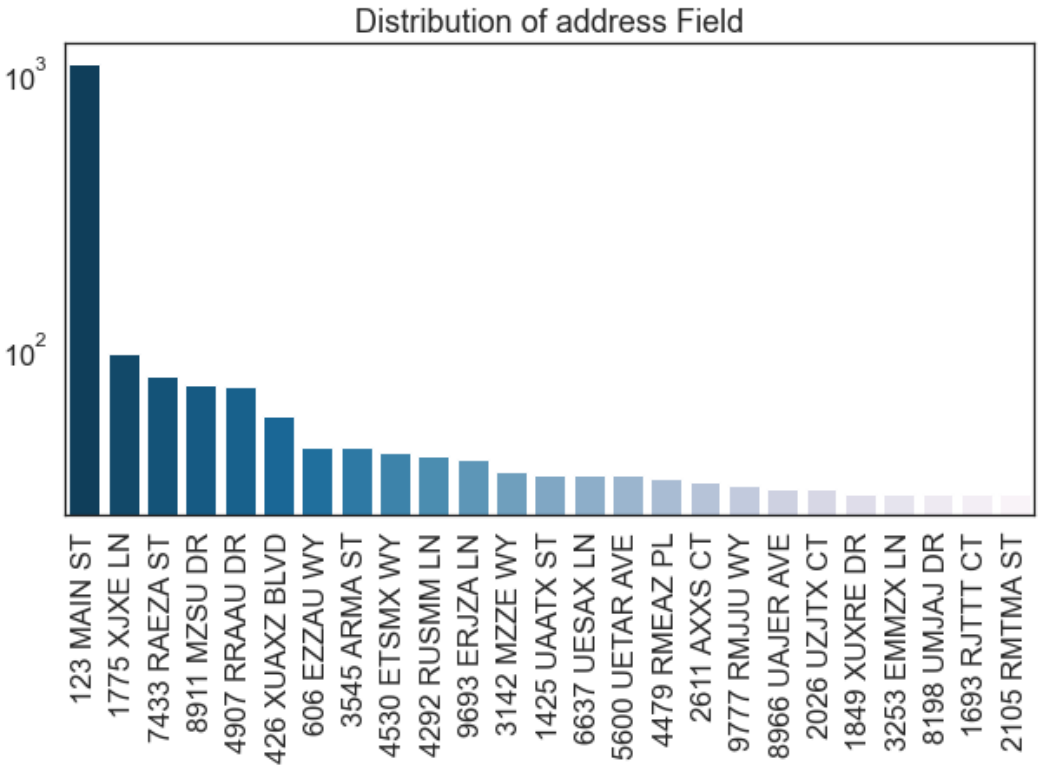


Figure 5. Count-plot of address Field (scale: log)

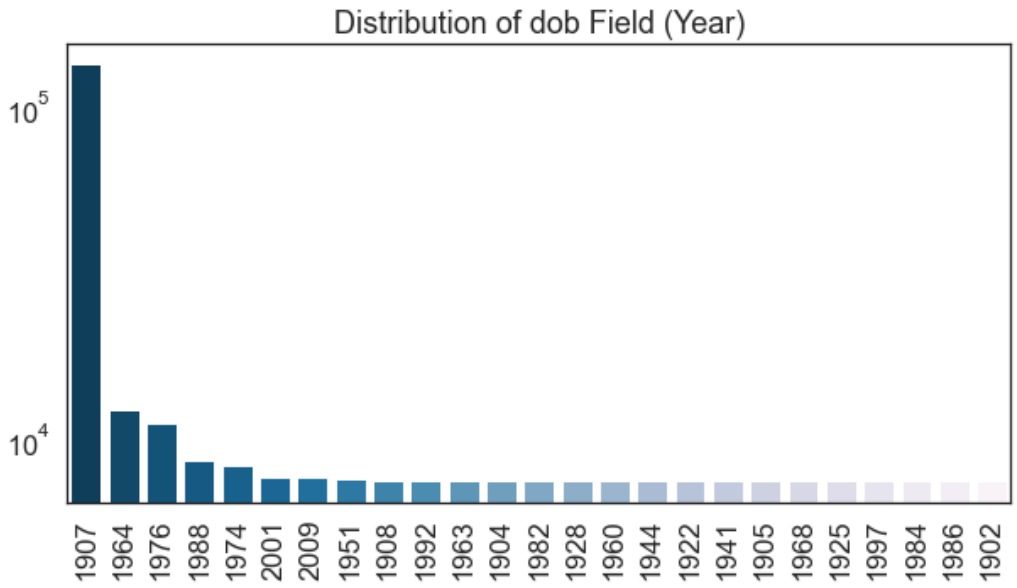


Figure 6. Count-plot of the years of the dob Field (scale: log)

2. Data Cleaning

After exploring and summarizing the applications, we needed to clean the data to handle any outliers, missing fields and frivolous values. All of the fields are 100% populated, meaning that we didn't have to fill any missing values with estimates. However, it is clear from the initial exploration that there are issues with outliers and frivolous values that would skew our research. To fix this, it will be necessary to manipulate and organize the date, zip5, ssn, address, dob and homephone fields.

2.1 Date Field Reformatting

The original values in the date field are of the in64 type and converting this column to the datetime type is pertinent in being able to efficiently organize the data. This was done by first converting the date field to a string and inserting dashes between the year, month and day utilizing the .str operator. The pandas to_datetime function was then used to convert the column from the string to the datetime type.

2.2 SSN Frivolous Values

As seen from Figure 2, the SSN field value of "999999999" was a blatant outlier with 16,935 occurrences in the dataset which accounts for about 1.7% of all applications. Given the frequency and the nature of the SSN, we can assume that "999999999" is a frivolous value that will link together many unrelated cases. Leaving the column as it would lead to unwanted correlations that would substantially impact the performance of our model. To reduce the linking effects, we will substitute the SSN with the negative of their corresponding record number and fill it with zeros to match the SSN length, which will ensure that each of these applications don't link to one another.

2.3 Address Frivolous Values

Looking at Figure 5, we can see that "123 MAIN ST" was a significant outlier, having 1079 occurrences in the dataset. It is apparent that "123 MAIN ST" is a popular but frivolous value that is commonly used in fraudulent situations. To make sure these records don't link with each other, we will need to convert the address to unique values by converting each frivolous value to the combination of the record and the string "record" to ensure each address with "123 MAIN ST" is unique in our dataset.

2.4 DOB Frivolous Values

We can also see frivolous values in the DOB column, where date 1907-06-26 had 126, 568 occurrences. While it could still be a legitimate date of birth for many applications, the fact that it

makes up about 12.7% of all applications indicates that it is a fraudulent and frivolous value. To ensure uniqueness, we approached this similarly as the SSN field. The DOB values with "19070626" were converted to the negative of their record and filled with zeros to match the length of the other DOB values.

2.5 Homephone Frivolous Values

In the homephone field, there were 78,512 instances of the number "9999999999", which made up about 7.9% of all records. The high frequency and redundant nature of the phone number indicate it is a frivolous value that needs to be fixed for further use. We approached it the same way as the SSN and DOB fields by taking the negative of the record number and filling zeros to match the length of the homephone field. Adding zeros ensures that each of our converted values is the same length as its field and that none of our converted values overlap with each other outside of their fields.

3. Create Candidate Variables

Feature creation is an essential step in balancing the inconsistencies and complexities of our data. While there are various types of identity fraud, this project will be focusing on identity theft through exploring 2 significant types of identity theft. One of the most common modes is when an individual fraudster has stolen or bought a list of identity information. As part of the application process, the fraudster uses core identifying information such as the victim's name, social security number or date of birth with the fraudster's own contact information such as address and phone number to apply for application. Another common situation is when a victim's identity was compromised in a data breach and their identifying information is being used by various fraudsters.

To find these two fraud modes, we want to create as many variables as possible using the entity fields (ssn, firstname, lastname, address, zip5, dob, and homephone) from the original dataset. In addition, we created two new fields namedob (firstname + lastname + dob) and fulladdress (address + zip) as unique identifiers to help detecting these two fraud modes. When it comes to the first type of fraud, determining how many different names or date of births are associated with one address would help to identify potential fraud. According to the second type of identity fraud, each breached core identity information was used by many different fraudsters. Therefore, this type of fraud can be detected by counting the number of addresses or phone numbers that are associated with each specific identity data. We also created a new categorical field called age_level by extracting the age from the dob column. We divided age into different categories, then utilized target encoding to convert the field to numerical values usable for model building purposes.

3.1 Linking Variables

From our preliminary exploration, it was apparent that the individual fields for each application would not be a dependable indicator for identifying people, as many applications could have the same name or zip code. In order to create candidate variables, we created combination groups of the identity fields to create linking variables. For instance, we created the following combination groups – ssn_fulladdress, ssn_namedob, ssn_phone, firstname_ssn, lastname_ssn – by concatenating the ssn field with these others. We also combined our new age_level field with the other groupings. Then for each entity and combination group, we created velocity, days since last seen, and relative velocity variables to detect possible frauds.

3.2 Velocity

Velocity refers to how many times a combo group of entities has been viewed over the past n days where n can be 0, 1, 3, 7, 14 or any other numbers. Velocity is a measure of how frequently the combined group appears in the application.

3.3 Days since last seen

Among the combination groups of entities, the days since last seen variable indicates how many days have passed between consecutive appearances of that entity in the data.

3.4 Relative velocity

A relative velocity variable is computed by dividing the number of applications with a specific combination group in the recent past (0 or 1 day) by the number of applications with that same combination group in the past n days where n can be 3, 7, 14 or 30 days. It is a ratio of the short-term velocity to a long-term averaged velocity.

3.5 Target Encoding

3.5.1 Day of Week Field

To look further into the date field, we created a Day of Week (dow) column by extracting the day from the date field. Since we can only use numerical variables, we utilized target encoding to convert each day into an average. Target encoding is a technique that utilizes a risk table to replace categorical variables with the relative mean of the target variable. The risk table variable was built by calculating the fraud proportion averages, processing the values through a statistical smoothing formula and then mapping it onto the Day of Week column. The new variable was called dow_risk.

3.5.2 Age_Level Field

Another variable that we wanted to look further into was the dob column. First, we calculated the age of applicants by subtracting the date of birth from the application date, then separated the ages into the following categories as the age_level variable:

- Age = 0: MISSING
- Age 1 to 4: TODDLER
- Age 5 to 12: CHILD
- Age 13 to 19: TEEN
- Age 20 to 39: ADULT
- Age 40 to 60: MIDDLE AGE ADULT
- Age 60+: SENIOR ADULT

We used target encoding to convert the age_level field into the relative fraud averages for each age_level, labeling the new field as age_level_risk. As previously mentioned, the age_level column was also used in the Linking Variables phase to create combination entities for feature creation.

4. Feature Selection Process

After creating 1883 potential features based on the 10 given variables, we continue for the feature selection. We filtered the features by using Kolmogorov-Smirnov (KS) first to select 100 relatively important features and then used a forward selection wrapper to select top 30 variables for further model building. This is the general workflow of the feature selection process.

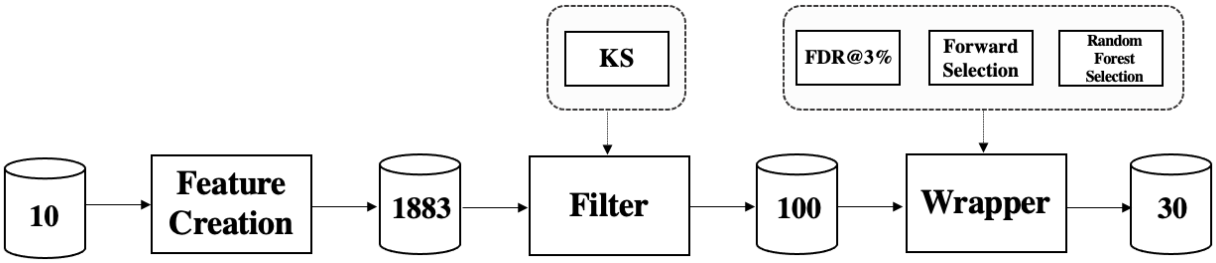


Figure 7. Workflow of feature selection.

4.1 Kolmogorov-Smirnov filter

Kolmogorov-Smirnov (KS) is a robust measure of how well two distributions are separated. For choosing the best variables to build models for fraud application detection, variables that can separate the good applications most from the bad applications are the best variables.

To calculate the separation or the KS value manually, we first removed the last two months of the records as the out-of-time data. And we left out the first two weeks since they didn't perform well. For each possible value of the variable, we substrat the proportion of bad applications from the proportion of good applications to get the difference. Next, we select the max differences among all values' differences as the KS score of this variable. Lastly, after we calculated all KS scores for the variables, we ranked them numerically and selected 100 variables with the highest KS scores.

$$KS = \max_x \left| \int_{x_{min}}^{x} P_{\text{good}}\, dx - \int_{x_{mi}}^{x} P_{\text{bad}}\, dx \right|$$

4.2 Wrapper – Forward Selection

With the selected 100 variables with the highest KS scores, we selected the top 30 variables by using a wrapper of forward selection. For the forward selecting wrapper, we started with an empty model and used random forest to add one more variable each time. The model will keep

the best variables it chose previously, and add one more variable that performs the best. The performance is measured by the fraud detection rate (FDR) at 3%, which is also used for measuring the performance of the models we built. The wrapper not only helps find the most valuable variables to differentiate the good and bad applications, but also helps remove correlated variables by only choosing one of the correlated variables in the model.

For the FDR score at 3%, it is a measure of the performance of a classifier. Higher the FDR score, the better the classifier is. To calculate this score, we first ranked the fraud scores calculated by the classifier of all records in descending order, and we cut the records into 100 bins. We selected the top 3 bins (3%) and compared the total number of fraud labels in these bins with the total number of frauds in the dataset. The higher the ratio is, the better the model is performing. In the wrapper, each time we added a new variable, we calculated the FDR score after adding one of the variables into the dataset and selected the variable that generated the highest FDR score of the classifier.

Lastly, after building the classifier with top 30 performing variables, we ranked these variables according to their rank being added into the model. With these ranked variables, we can more easily select the best performing number of variables for further model building and testing. The figure below shows the workflow of the feature selection process.

4.3 Final 30 Variables with Their Descriptions

Variable	KS Score	Variable	KS Score
address_count_30	0.3327	name_dob_count_0_by_30	0.2070
fulladdress_count_30	0.3320	ssn_age_level_count_0_by_30	0.2065
fulladdress_homephone_count_30	0.2290	ssn_count_0_by_30	0.2063
ssn_dob_count_30	0.2285	ssn_name_dob_count_0_by_30	0.2055
name_dob_count_30	0.2276	ssn_firstname_count_0_by_30	0.2053
ssn_name_dob_count_30	0.2262	ssn_lastname_count_0_by_30	0.2053
ssn_firstname_count_30	0.2261	ssn_name_count_0_by_30	0.2043
ssn_lastname_count_30	0.2260	homephone_count_3	0.1949
ssn_name_count_30	0.2250	name_dob_count_0_by_14	0.1948

name_dob_count_14	0.2153	ssn_count_0_by_14	0.1938
ssn_dob_count_14	0.2149	ssn_age_level_count_0_by_14	0.1935
ssn_count_14	0.2144	ssn_name_dob_count_0_by_14	0.1929
ssn_age_level_count_14	0.2142	ssn_lastname_count_0_by_14	0.1928
ssn_name_dob_count_14	0.2135	ssn_name_dob_count_7	0.1925
ssn_lastname_count_14	0.2134	ssn_name_count_0_by_14	0.1924

Table 1. Final Top 30 Variables with KS Score for Each

- address_count_30: the number of applications for the same address has been viewed over the past 30 days.
- fulladdress_count_30: the number of applications for the same fulladdress has been viewed over the past 30 days.
- fulladdress_homephone_count_30: the number of applications for the same combination of the fulladdress and homephone has been viewed over the past 30 days.
- ssn_dob_count_30: the number of applications for the same combination of ssn and dob has been viewed over the past 30 days.
- name_dob_count_30: the number of applications for the same combination of name and dob has been viewed over the past 30 days.
- ssn_name_dob_count_30: the number of applications for the same combination of ssn, name and dob has been viewed over the past 30 days.
- ssn_firstname_count_30: the number of applications for the same combination of ssn and firstname has been viewed over the past 30 days.
- ssn_lastname_count_30: the number of applications for the same combination of ssn and lastname has been viewed over the past 30 days.
- ssn_name_count_30: the number of applications for the same combination of ssn and name has been viewed over the past 30 days.
- name_dob_count_14: the number of applications for the same combination of name and dob has been viewed over the past 14 days.
- ssn_dob_count_14: the number of applications for the same combination of ssn and dob has been viewed over the past 14 days.
- ssn_count_14: the number of applications for the same ssn has been viewed over the past 14 days.
- ssn_age_level_count_14: the number of applications for the same combination of ssn and age_level has been viewed over the past 14 days.

- ssn_name_dob_count_14: the number of applications for the same combination of ssn, name and dob has been viewed over the past 14 days.
- ssn_lastname_count_14: the number of applications for the same combination of ssn and lastname has been viewed over the past 14 days.
- name_dob_count_0_by_30: the number of applications for the same combination of name and dob has been viewed over the past 0 days divides the number of applications with the same combination group that has been viewed over the past 30 days.
- ssn_age_level_count_0_by_30: the number of applications for the same combination of ssn and age_level has been viewed over the past 0 days divides the number of applications with the same combination group has been viewed over the past 30 days.
- ssn_count_0_by_30: the number of applications for the same ssn that has been viewed over the past 0 days divides the number of applications with the same ssn that has been viewed over the past 30 days.
- ssn_name_dob_count_0_by_30: the number of applications for the same combination of ssn, name, and dob that has been viewed over the past 0 days divides the number of applications with the same combination group that has been viewed over the past 30 days.
- ssn_firstname_count_0_by_30: the number of applications for the same combination of ssn and firstname has been viewed over the past 0 days divides the number of applications with the same combination group that has been viewed over the past 30 days.
- ssn_lastname_count_0_by_30: the number of applications for the same combination of ssn and lastname has been viewed over the past 0 days divides the number of applications with the same combination group that has been viewed over the past 30 days.
- ssn_name_count_0_by_30: the number of applications for the same combination of ssn and name has been viewed over the past 0 days divides the number of applications with the same combination group that has been viewed over the past 30 days.
- homephone_count_3: the number of applications for the same homephone has been viewed over the past 3 days.
- name_dob_count_0_by_14: the number of applications for the same combination of name and dob has been viewed over the past 0 days divides the number of applications with the same combination group that has been viewed over the past 14 days.
- ssn_count_0_by_14: the number of applications for the same ssn has been viewed over the past 0 days divides the number of applications for the same ssn has been viewed over the past 14 days.
- ssn_age_level_count_0_by_14: the number of applications for the same combination of ssn and age_level has been viewed over the past 0 days divides the number of applications with the same combination group that has been viewed over the past 14 days.

- ssn_name_dob_count_0_by_14: the number of applications for the same combinations of ssn, name and dob has been viewed over the past 0 days divides the number of applications with the same combination group that has been viewed over the past 14 days.
- ssn_lastname_count_0_by_14: the number of applications for the same combinations of ssn and lastname has been viewed over the past 0 days divides the number of applications with the same combination group that has been viewed over the past 14 days.
- ssn_name_dob_count_7: the number of applications for the same combinations of ssn, name, and dob has been viewed over the past 7 days.
- ssn_name_count_0_by_14: the number of applications for the same combination of ssn and name has been viewed over the past 0 days divides the number of applications with the same combination group that has been viewed over the past 14 days.

5. Model Algorithms

This project implemented four different models – Logistic Regression, Decision Tree, Extreme Gradient Boosting, and Neural Network. For each model, we have tuned various parameters to achieve the best performance.

Before training the models, data after November 1th, 2016 were reserved as the out-of-time (OOT) data for the evaluation purpose which is about 17% of the whole data. In addition, data before January 14th were also left out, because many of the features would not be effective. This is equivalent to 38512 rows of data. We then randomly split the rest of the data into the training and test data sets, where the training set contains 70% of the data and the testing set contains 30%. Since this project is a classification problem, we used the stratifying method when splitting to ensure that each set has a similar proportion of fraud label records.

After training the models, we obtained the probability of each observation being a fraud (class 1) and sorted them in descending order. Then, we calculated the fraud detection rate in the top 3% of the sorted lists for the training, testing, and OOT dataset respectively to evaluate the model performance.

Since we have 30 variables to build the model, we want to see if there are a sufficient number of variables to develop a predictive model. By using a non-linear algorithm, Neural Network, we visualized where the cut off line for choosing the number of variables is needed to reduce the complexity of the model but also maintain the effectiveness. Based on the line chart shown below, OOT score fluctuates with the change of number of variables in the neural net model. Choosing fewer than 5 variables may easily cause underfitting problems as well as choosing more than 10 variables may make our model redundant and hard to generalize. Therefore, it is better to choose the number of variables between this range. OOT will get a local peak when we choose 8 variables so we choose the first 8 variables to fit the following models.

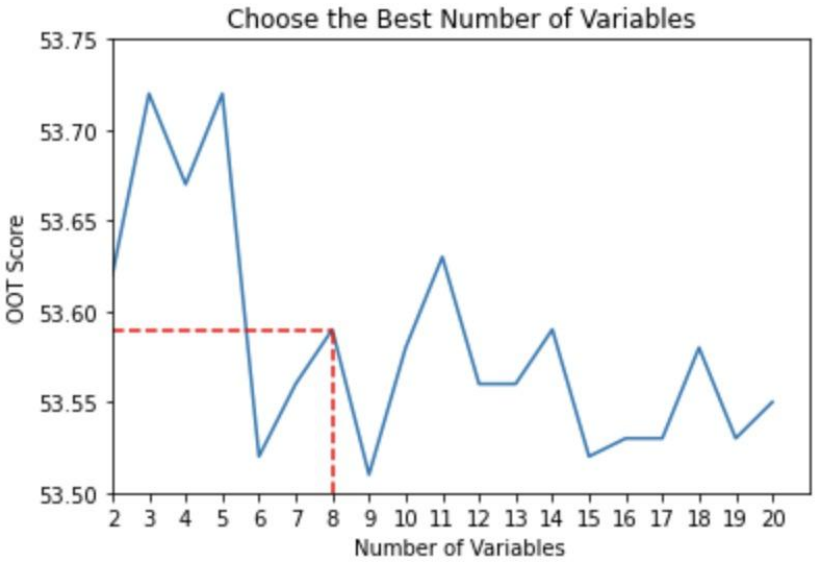


Figure 8. Best Number of Variables for Model Algorithm

5.1 Four Models

5.1.1 Logistic Regression

After selecting 8 variables that have relatively useful information, we choose to use the Logistic Regression model as our baseline model. Due to its simplicity and high efficiency, the Logistic Regression algorithm is widely used in most binary classification problems. It estimates the probability of y=1 using a regression function of x, in other words, P(Y=1| X=x). By giving a user-defined decision boundary threshold, the model can automatically classify the observations based on the output probability. Specifically, in logistic regression, the probability of y=1 regress on x using a function in the inverse logit form. Below is the inverse logit function.

$$P(Y=1|\ X=x) = \frac{e^{\beta_0+\beta_1 x_1}}{1+e^{\beta_0+\beta_1 x_1}}$$

The table below shows the various parameters we used to test the Logistic Regression model.

- penalty: Any modification on the learning algorithm that is intended to reduce its generalization error but not its training error
- C: The inverse of regularization strength
- solver: Solves optimization problems by successively performing approximate minimization along coordinate directions or coordinate hyperplanes

Model		Parameters				Average FDR at 3%		
	Iteration	#Variables	penalty	C	solver	Train	Test	OOT
Logistic Regression	1	8	l2	0.1	lbfgs	53.86	53.8	51.84
	2	8	l2	1	saga	54.14	53.34	51.84
	3	8	l2	0.1	saga	53.85	53.81	51.77
	4	8	l2	5	saga	53.58	53.48	51.77
	5	8	l1	1	saga	53.74	53.75	51.76
	6	8	l2	1	lbfgs	53.71	53.84	51.76
	7	8	l1	5	saga	53.75	53.79	51.73
	8	8	l2	0.6	saga	53.73	53.82	51.73

Table 2. Logistic Regression high-level results

The table above is sorted by how well the model performed based on the OOT dataset. We can conclude that the Logistic Regression model reached the best performance when we use the 'lbfgs' solver and apply L2 regularization with inverted strength of 0.1. Looking at the second best model, the OOT is the same, but the best model has a better performance on the test dataset.

5.1.2 Decision Tree

A decision tree algorithm is one of the most used and intuitive machine learning algorithms. From its name, a Decision Tree consists of a tree of decisions to partition the data space into smaller subspaces, where each subspace is given a label. During the training process, the tree is built by splitting one node at a time, and the algorithm scans all possible splits along all axes to optimize the split. Entropy, information gain, and Gini are examples of metrics to measure how good a split is. These all measure the impurity of the resulting two partitions, and the best split point is found to result from the least impurity among partitions. After the tree is built, we then can traverse the tree according to the rules and get the predicted label for new records.

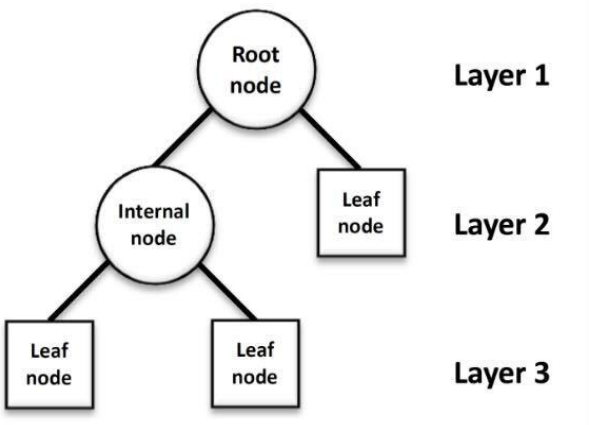


Figure 9. Decision Tree Diagram

A decision tree is commonly used due to its fast training, reliability, and robustness to dirty data; however, it is prone to overfit, where the algorithm can generate over-complex trees that fit training data so perfectly that they do not generalize to unseen data.

There are 4 related hyper-parameters tuned for our decision tree algorithm:

- criterion: The criterion to measure impurity.
- splitter: Cost coefficient for cost-complex pruning.
- max_depth: Maximum depth of the tree.
- min_samples_leaf: Minimal number of samples in each leaf node. If the number is lower, two leaf nodes will be combined.

After tuning, the final setting for parameters are shown in Table 3.

Model		Parameters					Average FDR at 3%		
	Iteration	#Variables	criterion	splitter	max_depth	min_samples_leaf	Train	Test	OOT
Decision Tree	1	8	entropy	random	10	15	55.65	55.93	53.7
	2	8	gini	best	10	20	55.73	55.84	53.69
	3	8	gini	best	10	25	55.86	55.48	53.69
	4	8	gini	best	10	30	55.47	56.35	53.69
	5	8	gini	random	10	40	55.28	55.4	53.2
	6	8	entropy	random	10	30	55.18	54.84	53.05
	7	8	entropy	random	10	40	55.03	55.31	53
	8	8	gini	random	10	15	54.93	54.67	52.77

Table 3. Decision Tree high-level results

The table above is sorted by how well the model performed based on the OOT dataset and listed the top 4 best models and worst 4 models as comparison. Therefore, we can conclude that the Decision Tree model reached the best performance when we use the 'entropy' criterion, random splitter, max depth of trees to be 10 and minimum samples on a leaf to be 15. Surprisingly, comparing the worst model to the best, the only difference is the criterion, entropy vs gini.

5.1.3 Extreme Gradient Boosting (XG-Boost)

Boosting method, another ensemble learning method, is one of the most powerful machine learning algorithms in prediction making. The idea of boosting is combining the predictions of multiple weak learners, which are slightly better than random guesses, sequentially to reduce the bias.

Gradient boosting recasts the boosting method as a numerical optimization problem, where the objective is to minimize the loss by using a gradient-descent-like procedure to add weak learners. In other words, the gradient of the loss function is approximated by a new weak learner at each step. This property allows users to use a self-customized loss function as long as it is differentiable. The figure below shows the process of learning from the weaker models.

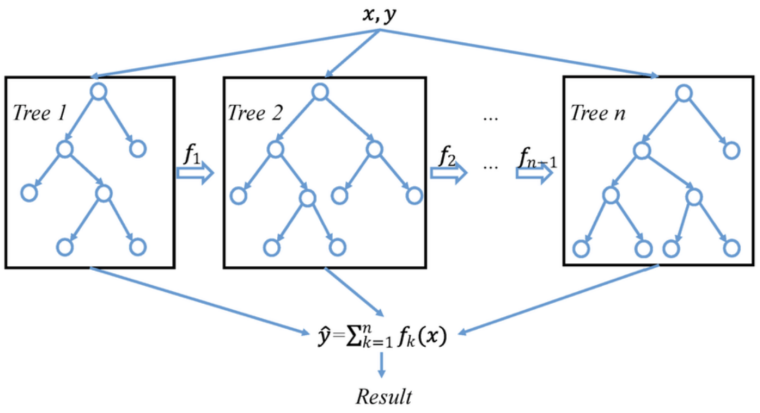


Figure 10. Extreme Gradient Boosting Diagram

Extreme Gradient Boosting is a scalable and highly accurate implementation of gradient boosting that pushes the limits of computing power for boosted tree algorithms, being built largely for energizing machine learning model performance and computational speed. It follows a level-wise strategy, scanning across gradient values and using these partial sums to evaluate the quality of splits at every possible split in the training set.

There are 3 related hyper-parameters tuned for our XG-Boost algorithm:

- n_estimators: The number of trees or rounds in the model.
- learning_rate: A tuning parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a loss function.
- max_depth: Maximum depth of the tree.

Model		**Parameters**				**Average FDR at 3%**		
	Iteration	#Variables	n_estimators	learning_rate	max_depth	Train	Test	OOT
Extreme Gradient Boosting	1	8	500	0.3	5	55.92	55.42	53.79
	2	8	150	0.05	5	55.72	55.69	53.74
	3	8	500	0.1	4	55.8	55.65	53.74
	4	8	200	0.1	5	55.73	55.82	53.73
	5	8	150	0.5	5	55.88	55.41	53.48
	6	8	100	0.2	3	55.74	55.69	53.46
	7	8	200	0.1	4	55.68	55.85	53.46
	8	8	500	0.2	5	55.9	55.27	53.44

Table 4. Extreme Gradient Boosting high-level results

The table above is sorted by how well the model performed based on the OOT dataset. We can conclude that the Extreme Gradient Boosting model reached the best performance when we use 500 estimators, learning rate at 0.3 and max depth of trees to be 5. Overall, the XG-Boost model performed very well despite changing different parameters since the worst model has an OOT of 53.44.

5.1.4 Neural Network

The neural network model is an imitation of the multiple neurons in the human brain. This model is often used in supervised learning to train the model to recognize and distinguish one thing from another through large data sets.

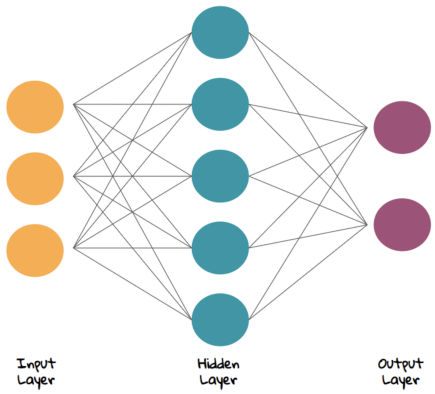


Figure 11. Neural Network Diagram

MLP, which represents Multi-layer Perceptron, is a supervised learning algorithm, and it is the classic neural network model above. In other words, it is a class of feedforward artificial neural networks that use backpropagation for training. It can have multiple layers, and non-linear activation distinguishes multi-layer perceptrons.

There are 3 related hyper-parameters tuned for our Neural Network algorithm:

- Hidden layer size: The number of neurons in the ith hidden layer.
- Max iteration: The maximum number of times a batch of data passed through the algorithm.
- Learning rate: It schedules for weight updates.

Model		Parameters				Average FDR at 3%		
	Iteration	#Variables	hidden_layer_size	max_iteration	learning_rate	Train	Test	OOT
Neural Network	1	8	(10, 5)	20	invscaling	55.52	56	53.66
	2	8	(20, 10)	60	invscaling	55.94	55.13	53.63
	3	8	20	20	constant	55.54	55.98	53.62
	4	8	20	60	constant	55.59	55.85	53.62
	5	8	20	60	invscaling	55.33	55.32	53.24
	6	8	10	60	invscaling	55.27	55.28	53.16
	7	8	10	60	constant	55.1	55.19	53.02
	8	8	10	50	constant	54.85	53.66	52.5

Table 5. Neural Network high-level results

The table above is sorted by how well the model performed based on the OOT dataset. We can conclude that the Neural Network model reached the best performance when we use criterion with 2 layers and 10 & 5 nodes in each layer respectively, 20 splitter and max depth of trees to be

'invscaling'. In addition, having a lower splitter value will more likely make the model perform well on this dataset.

5.2 Model Comparison with 8 variables vs 25 variables

Models	Parameters	Average FDR at 3%		
	#Variables	Train	Test	OOT
Logistic Regression	8	53.86	53.8	51.84
	25	53.82	54.45	52.14
Decision Tree	8	55.65	55.93	53.7
	25	55.57	56.23	53.63
Extreme Gradient Boosting	8	55.92	55.42	53.79
	25	55.94	55.6	53.63
Neural Network	8	55.52	56	53.66
	25	55.99	55.15	53.67

Table 6. Number of Variable Comparison Within Models

This table shows the best OOT performance for each model using 8 and 25 variables. We can see that the Logistic Regression and Neural Network algorithms have a higher OOT when the number of variables are larger, but it does not make a significant difference that will convince people that having more variables is better. In fact, looking at Decision Tree and XG-Boost algorithms, only using 8 variables performs better. Thus, when we decide to create all four models with 8 variables, it will not lose information about the data than using 25 variables.

6. Result

From table 4, the Extreme Gradient Boosting algorithm performed the overall best out of all models. As a matter of fact, this model performed very well in the training and the test dataset as well. For this dataset, it seems non-linear models performed much better than the linear model. The table below shows the best hyperparameters for this model.

Model	Parameters				Average FDR at 3%		
Extreme Gradient Boosting	#Variables	n_estimators	learning_rate	max_depth	Train	Test	OOT
	8	500	0.3	5	55.92	55.42	53.79

Table 7. The Final Best Model

The following tables are training, testing, and OOT dataset's final performance tables:

Training	# Records	# Goods	# Bads	Fraud Rate
	583454	575094	8360	1.45%

Bin Statisitcs						Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total Records	Cumulative Goods	Cumulative Bads	% Goods	FDR	KS	FPR
1	5835	1419	4416	24.32	75.68	5835	1419	4416	0.25	52.82	52.57	0.32
2	5835	5632	203	96.52	3.48	11670	7051	4619	1.23	55.25	54.02	1.53
3	5835	5760	75	98.71	1.29	17505	12811	4694	2.23	55.92	53.92	2.73
4	5835	5786	49	99.16	0.84	23340	18597	4743	3.23	56.73	53.5	3.92
5	5835	5789	46	99.21	0.79	29175	24386	4789	4.24	57.28	53.04	5.09
6	5835	5791	44	99.25	0.75	35010	30177	4833	5.25	57.81	52.56	6.24
7	5835	5777	58	99.01	0.99	40845	35954	4891	6.25	58.5	52.25	7.35
8	5835	5793	42	99.28	0.72	46680	41747	4933	7.26	59.01	51.75	8.46
9	5835	5781	54	99.07	0.93	52515	47528	4987	8.26	59.65	51.39	9.53
10	5835	5786	49	99.16	0.84	58350	53314	5036	9.27	60.24	50.97	10.59
11	5835	5785	50	99.14	0.86	64185	59099	5086	10.28	60.84	50.56	11.62
12	5835	5802	33	99.43	0.57	70020	64901	5119	11.29	61.23	49.94	12.68
13	5835	5800	35	99.4	0.6	75855	70701	5154	12.29	61.65	49.36	13.72
14	5835	5808	27	99.54	0.46	81690	76509	5181	13.3	61.97	48.67	14.77
15	5835	5787	48	99.18	0.82	87525	82296	5229	14.31	62.55	48.24	15.74
16	5835	5803	32	99.45	0.55	93360	88099	5261	15.32	62.93	47.61	16.75
17	5835	5792	43	99.26	0.74	99195	93891	5304	16.33	63.44	47.11	17.7
18	5835	5798	37	99.37	0.63	105030	99689	5341	17.33	63.89	46.56	18.66
19	5835	5794	41	99.3	0.7	110865	105483	5382	18.34	64.38	46.04	19.6
20	5835	5794	41	99.3	0.7	116700	111277	5423	19.35	64.87	45.52	20.52

Table 8. Performance Table for Training Dataset

Test	# Records	# Goods	# Bads	Fraud Rate			
	250053	246406	3647	1.48%			

Bin Statisitcs						Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total Records	Cumulative Goods	Cumulative Bads	% Goods	FDR	KS	FPR
1	2501	612	1889	24.47	75.53	2501	612	1889	0.25	51.8	51.55	0.32
2	2501	2422	79	96.84	3.16	5002	3034	1968	1.23	53.96	52.73	1.54
3	2501	2460	41	98.36	1.64	7503	5494	2009	2.23	55.42	52.86	2.73
4	2501	2479	22	99.12	0.88	10004	7973	2031	3.24	55.69	52.45	3.93
5	2501	2488	13	99.48	0.52	12505	10461	2044	4.25	56.05	51.8	5.12
6	2501	2474	27	98.92	1.08	15006	12935	2071	5.25	56.79	51.54	6.25
7	2501	2478	23	99.08	0.92	17507	15413	2094	6.26	57.42	51.16	7.36
8	2501	2480	21	99.16	0.84	20008	17893	2115	7.26	57.99	50.73	8.46
9	2501	2475	26	98.96	1.04	22509	20368	2141	8.27	58.71	50.44	9.51
10	2501	2478	23	99.08	0.92	25010	22846	2164	9.27	59.34	50.07	10.56
11	2501	2498	3	99.88	0.12	27511	25344	2167	10.29	59.42	49.13	11.7
12	2501	2485	16	99.36	0.64	30012	27829	2183	11.29	59.86	48.57	12.75
13	2501	2475	26	98.96	1.04	32513	30304	2209	12.3	60.57	48.27	13.72
14	2501	2487	14	99.44	0.56	35014	32791	2223	13.31	60.95	47.64	14.75
15	2501	2489	12	99.52	0.48	37515	35280	2235	14.32	61.28	46.96	15.79
16	2501	2473	28	98.88	1.12	40016	37753	2263	15.32	62.05	46.73	16.68
17	2501	2485	16	99.36	0.64	42517	40238	2279	16.33	62.49	46.16	17.66
18	2501	2482	19	99.24	0.76	45018	42720	2298	17.34	63.01	45.67	18.59
19	2501	2486	15	99.4	0.6	47519	45206	2313	18.35	63.42	45.07	19.54
20	2501	2481	20	99.2	0.8	50020	47687	2333	19.35	63.97	44.62	20.44

Table 9. Performance Table for Test Dataset

OOT	# Records	# Goods	# Bads	Fraud Rate			
	166493	164107	2386	1.45%			

Bin Statisitcs						Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total Records	Cumulative Goods	Cumulative Bads	% Goods	FDR	KS	FPR
1	1665	458	1207	27.51	72.49	1665	458	1207	0.28	50.59	50.31	0.38
2	1665	1608	57	96.58	3.42	3330	2066	1264	1.26	52.98	51.72	1.63
3	1665	1644	21	98.74	1.26	4995	3710	1285	2.26	53.79	51.6	2.89
4	1665	1647	18	98.92	1.08	6660	5357	1303	3.26	54.61	51.35	4.11
5	1665	1654	11	99.34	0.66	8325	7011	1314	4.27	55.07	50.8	5.34
6	1665	1647	18	98.92	1.08	9990	8658	1332	5.28	55.83	50.55	6.5
7	1665	1646	19	98.86	1.14	11655	10304	1351	6.28	56.62	50.34	7.63
8	1665	1652	13	99.22	0.78	13320	11956	1364	7.29	57.17	49.88	8.77
9	1665	1656	9	99.46	0.54	14985	13612	1373	8.29	57.54	49.25	9.91
10	1665	1653	12	99.28	0.72	16650	15265	1385	9.3	58.05	48.75	11.02
11	1665	1654	11	99.34	0.66	18315	16919	1396	10.31	58.51	48.2	12.12
12	1665	1653	12	99.28	0.72	19980	18572	1408	11.32	59.01	47.69	13.19
13	1665	1654	11	99.34	0.66	21645	20226	1419	12.32	59.47	47.15	14.25
14	1665	1650	15	99.1	0.9	23310	21876	1434	13.33	60.1	46.77	15.26
15	1665	1649	16	99.04	0.96	24975	23525	1450	14.34	60.77	46.43	16.22
16	1665	1653	12	99.28	0.72	26640	25178	1462	15.34	61.27	45.93	17.22
17	1665	1655	10	99.4	0.6	28305	26833	1472	16.35	61.69	45.34	18.23
18	1665	1648	17	98.98	1.02	29970	28481	1489	17.36	62.41	45.05	19.13
19	1665	1654	11	99.34	0.66	31635	30135	1500	18.36	62.87	44.51	20.09
20	1665	1644	21	98.74	1.26	33300	31779	1521	19.36	63.75	44.39	20.89

Table 10. Performance Table for OOT Dataset

With our best performing Extreme Gradient Boost model, the three tables from above showing FDR and other statistics for each population bin. Records are ranked in descent order based on their probability of being fraud. From tables 8 to 10, we can see that more than 50% of frauds are

caught within the very first percentile bin. Fraud detection rate at 3%, or the cumulative percentage of bad, are 55.92%, 55.42% and 53.79% for training, testing and OOT respectively.

7. Conclusion

We built four different supervised machine learning models on the synthetic credit card application dataset to find identity fraud in this project. We first explore the dataset by calculating the summary statistics and plotting the distribution of the 10 fields. A Data Quality Report was written. Then we cleaned the data by substituting the frivolous values. After the data was cleaned, we created several new fields by combining different identity fields. Also, we created age_level to represent the age of applicants when submitting the application. Then, 1883 candidate variables were generated based on the occurrence of fields and the occurrence of field relationships. Since the dimensionality of the data was too high to fit machine learning models, we used KS as a filter to reduce the number of variables to 100, and then we ran a wrapper to select the top 30 best variables for modeling.

Before training the models, we first removed the last two months of the records as the OOT data. Also, we left out the first two weeks since they are not effective for modeling. We used a Neural Network to visualize the cut off line for choosing the number of variables. We found OOT will get a local peak when we choose 8 variables. We decided to use the first 8 variables to fit the models. Logistic Regression was built firstly as our baseline model. The Logistic Regression reached the best performance of 51.84% FDR at 3% when we use the 'lbfgs' solver and apply L2 regularization. Then we built the Decision Tree model, Extreme Gradient Boosting model, and Neural Network model subsequently, applying various hyperparameters. Among these, the best performance was from the XG-Boosting model, which reached 53.79% when we use 500 estimators, the learning rate at 0.3, and the max depth of trees to be 5. We also compared the model's best performance between models using 8 variables and using 25 variables. It confirmed that using 8 variables will achieve the best performance. Tables of the predicted percentile bins using the Extreme Gradient Boost model were built. We could see that more than 50% of frauds were caught within the very first percentile bin.

There are also aspects of our models that can be improved. Even though the XG-Boost algorithm had the highest OOT, it did not significantly have the highest score. Therefore, by exploring different hyperparameters on the other algorithms, we might be able to find close results to the best model.

Appendix

A.1 File Description

The file is a synthetic dataset of applications for credit cards and cell phones that imitates real application data. The dataset was made based on analysis on real U.S. applications over the last 10 years. The dataset covers applications from Jan 1, 2016 to Dec 31, 2016. It has 1,000,000 records and 10 fields.

A.2 Numerical Fields Summary Statistics

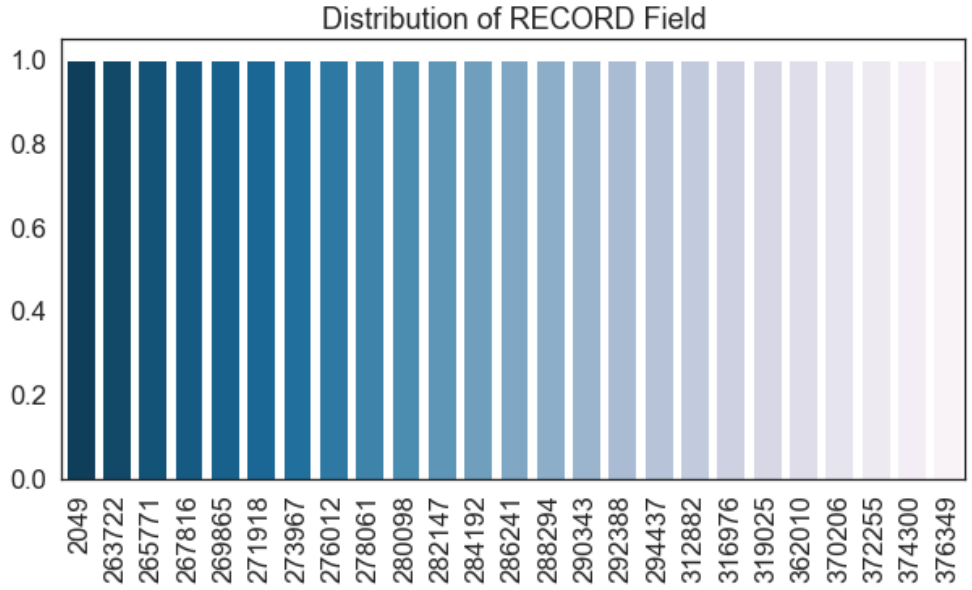
Field Name	% Populated	Min	Max	Mode	Mean	Stdev	% Zero
date	100%	2016-01-01	2016-12-31	2016-08-16	N/A	N/A	0
dob	100%	1900-01-01	2016-10-31	1907-06-26	N/A	N/A	0

A.3 Categorical Fields Summary Statistics

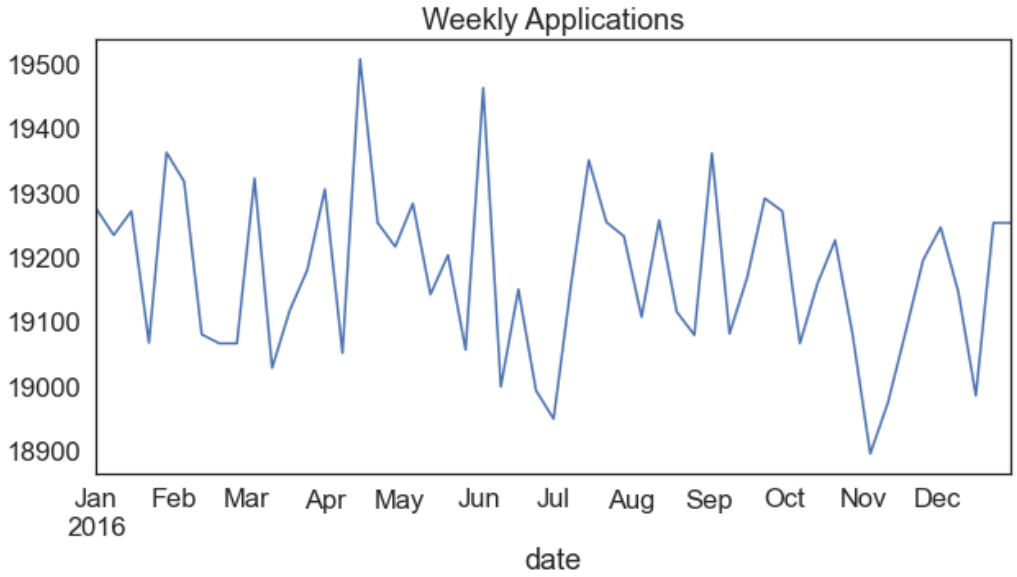
Field Name	% Populated	# Unique Values	Mode
record	100%	1,000,000	1
ssn	100%	835,819	999999999
firstname	100%	78,136	EAMSTRMT
lastname	100%	177,001	ERJSAXA
address	100%	828,774	123 MAIN ST
zip5	100%	26,370	68138
homephone	100%	28,244	9999999999
fraud_label	100%	2	0

A.4 Distributions of Fields

Field 1: Record
Description: The unique index of the dataset. Each application has a unique record number.



Field 2: Date
Description: The date that the application was filed on. The dates range from 2016-01-01 to 2016-12-31. The graph shows weekly application frequencies.



Field 3: SSN
Description: The 9-digit social security number of the applicant. Outlier “999999999” had 16,935 instances.

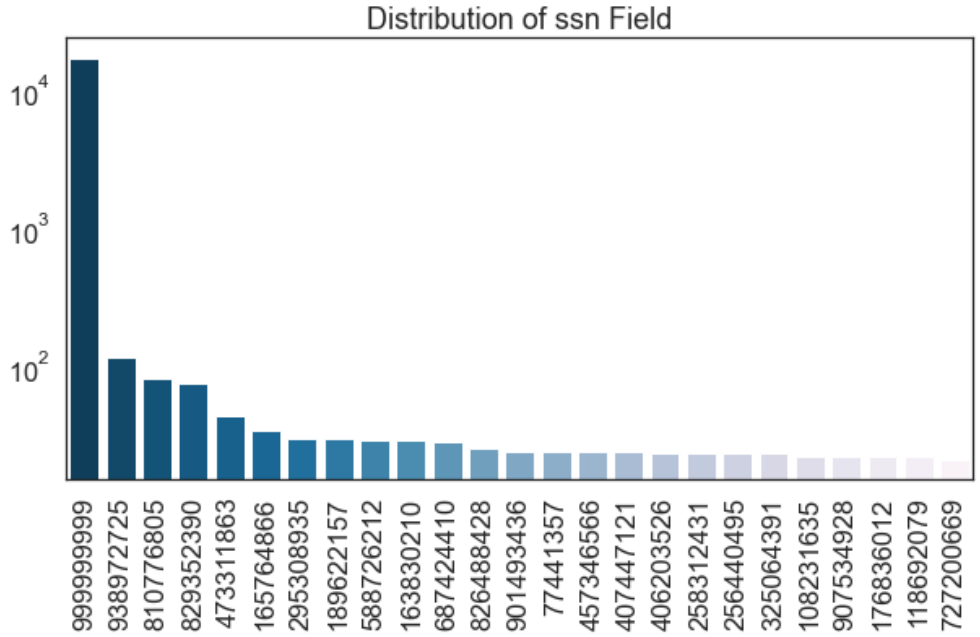


Figure 2. Count-plot of ssn Field (scale: log)

Field 4: First Name
Description: The listed first name of the applicant.

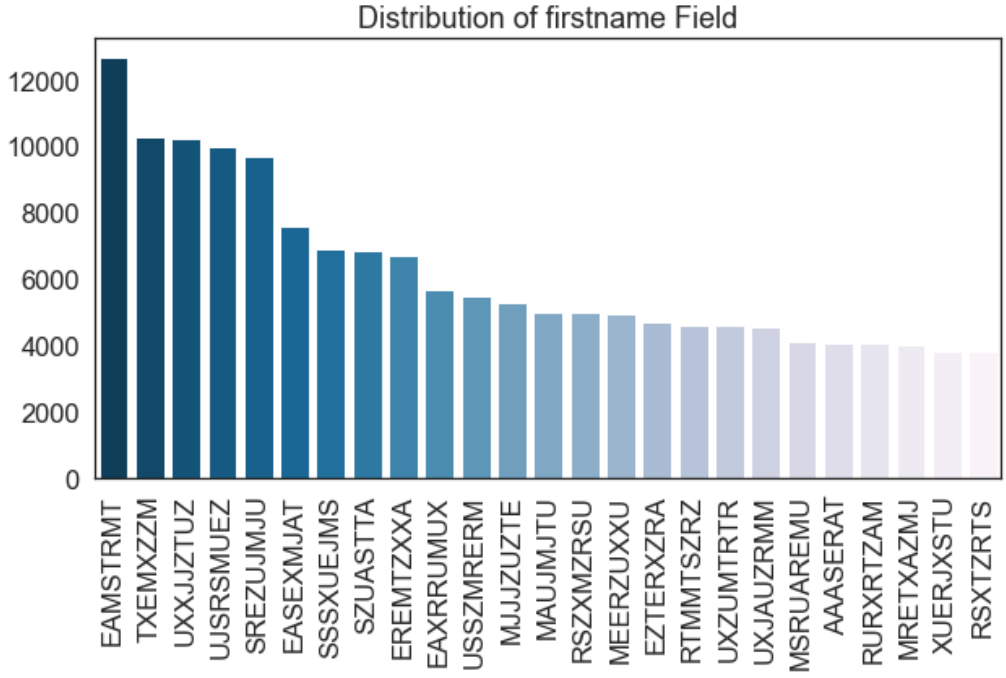


Figure 3. Count-plot of firstname Field

Field 5: Last Name
Description: The listed last name of the applicant.

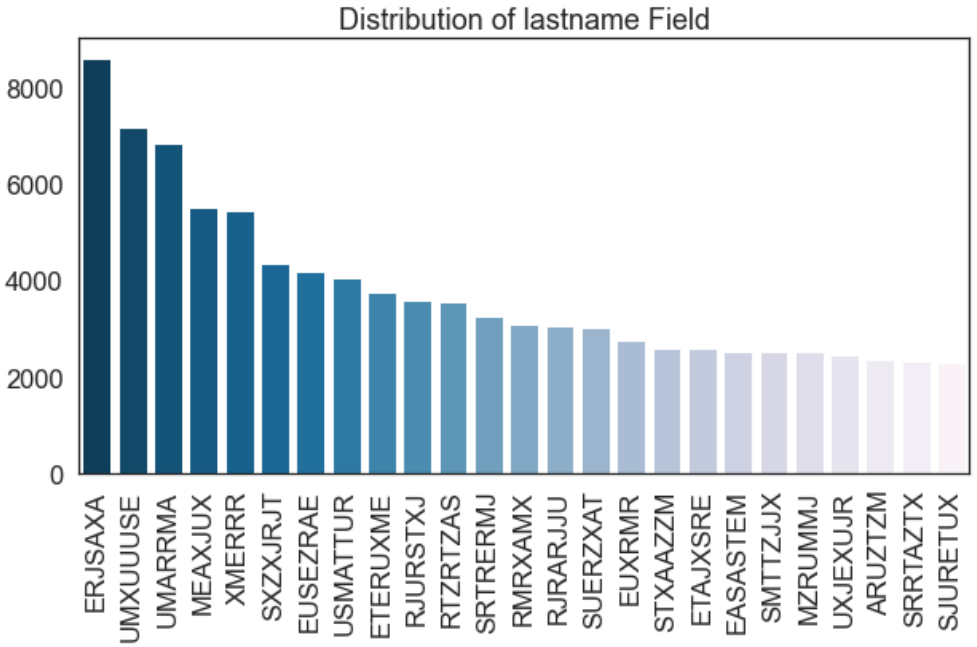


Figure 4. Count-plot of firstname Field

Field 6: Address
Description: The listed address on the application. Outlier "123 Main St" had 1079 instances.

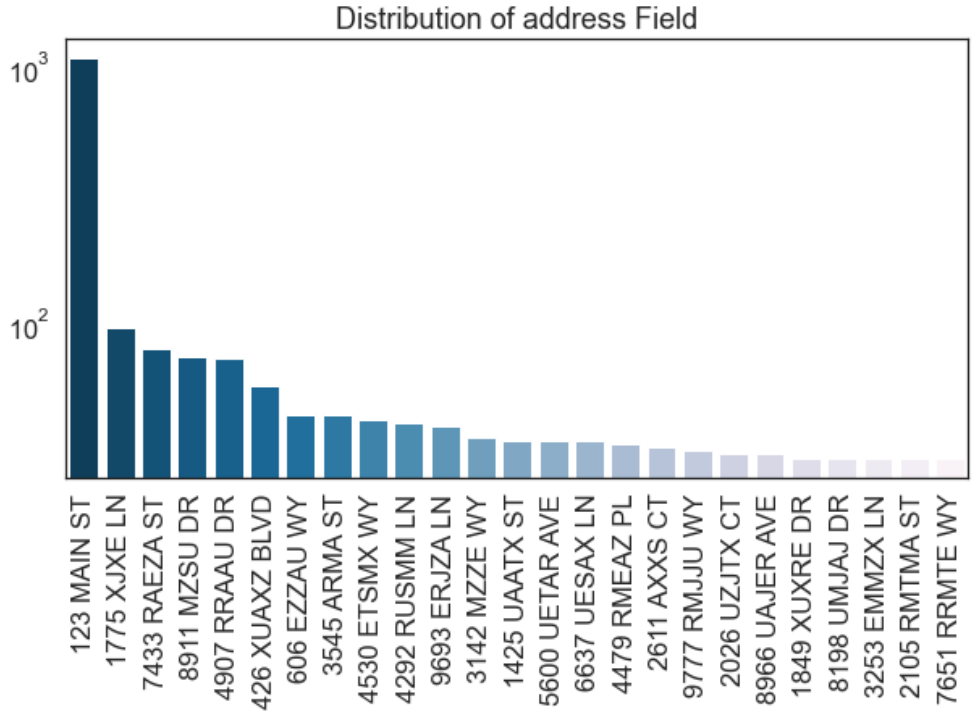
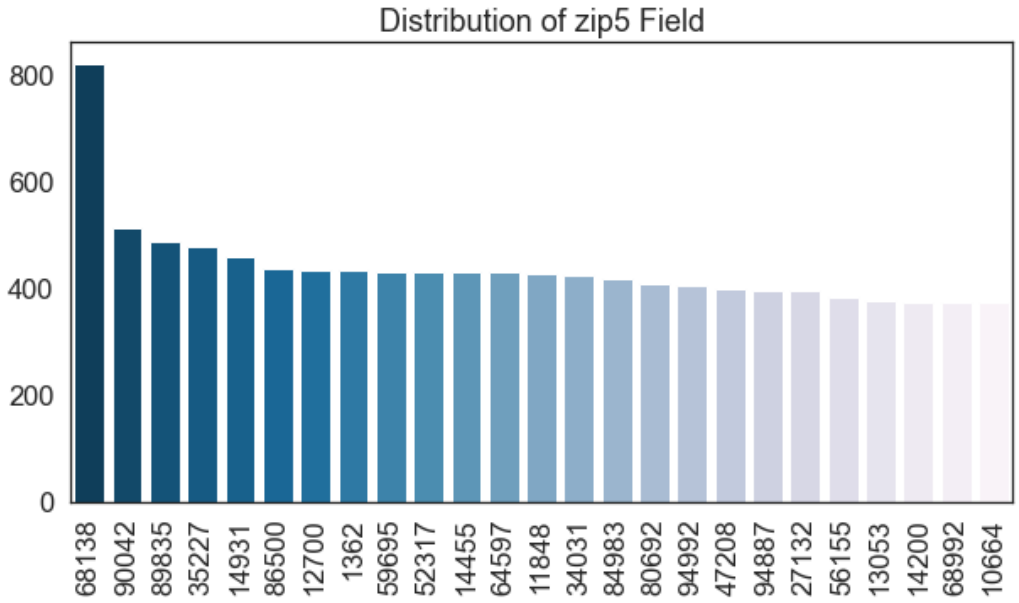


Figure 5. Count-plot of address Field (scale: log)

Field 7: Zip5
Description: The listed zip code of the address on the application.



Field 8: DOB
Description: The date of birth of the applicant. Graph shows years of each "dob". Outlier "1907" had 133,986 instances.

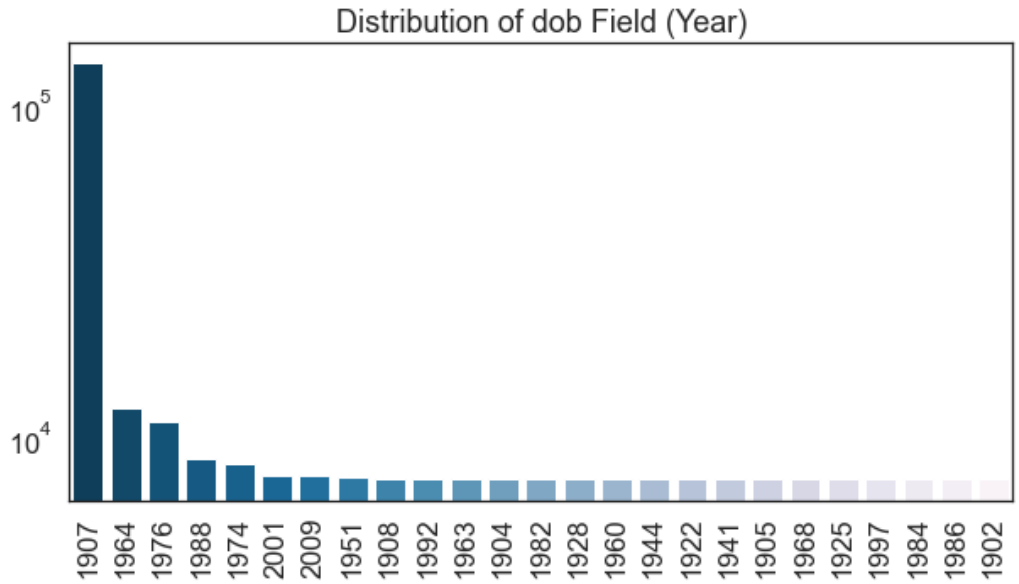
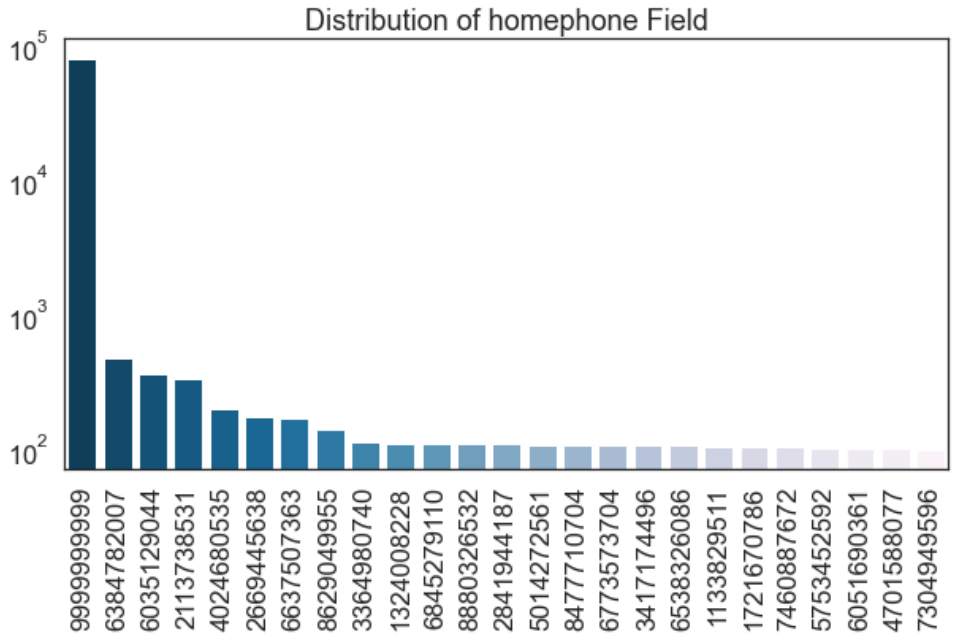
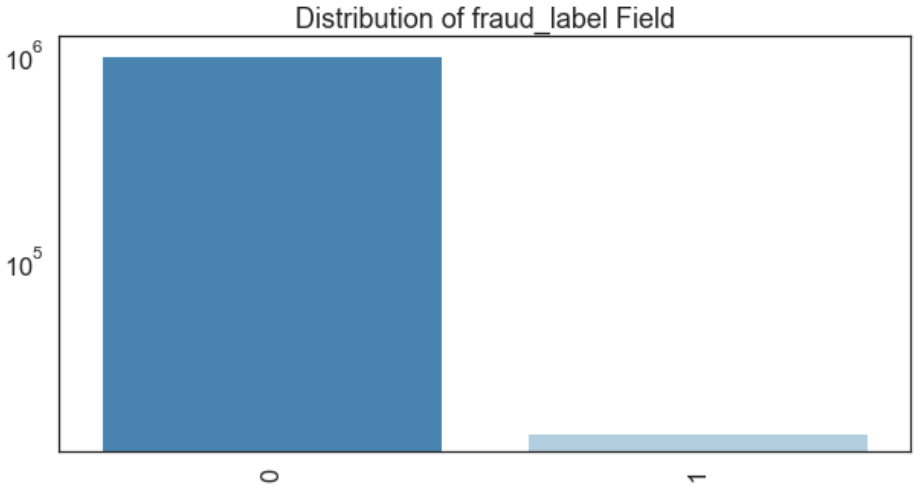


Figure 6. Count-plot of the years of the dob Field (scale: log)

Field 9: Homephone
Description: The listed home phone number of the applicant. Outlier “9999999999” had 78,512 instances.



Field 10: Fraud_label
Description: Whether or not the model detected the application as fraudulent. Values are 1 for fraudulent, 0 for not. 1.4% of records have the label 1.



A.5 Created Variables

Description of Variables	# Variables Created	Approximate CPU Time (Minute)	Names of New Columns
Day of Week of Each Application (including Monday to Sunday, 7 unique values)	1	<0.1 min	dow
Target Encoding for dow column (transferring categorical variables to numerical variables)	1	<0.1 min	Dow_risk
Age of the applicant when they deliver application (treat missing values as 0)	1	<0.1 min	age
Age level of the age filed (7 categories, treat missing values as “MISSING” type)	1	<0.1 min	Age_level
Target Encoding for age column (transferring categorical variables to numerical variables)	1	<0.1 min	Age_level_risk
Full name of applicant (including first and last name)	1	<0.1 min	name
Full address of applicant (including address and 5-digit zip code)	1	<0.1 min	Fulladdress
Combination of name and date of birth, address and home phone number of the applicant	3	<0.1 min	Name_dob Name_fulladdress Name_homephone
Combination of full address and dob, home phone number of the applicant	2	<0.1 min	Fulladdress_dob Fulladdress_homephone
Combination of home phone number and date of birth, name & date of birth of the applicant	2	<0.1 min	Dob_homephone Homephone_name_dob

Combination of age level and address, home phone number of the applicant	2	<0.1 min	Avg_level_address Avg_level_homephone
Combination of social security number and {first name, last name, address, 5-digit zip code, date of birth, home phone number, age level, name, full address, name & date of birth, age level & address, age level & home phone number}	12	<0.1 min	Ssn_ {firstname, lastname, address, zip5, dob, homephone, age_level, name, fulladdress, name _dob, age_level_address, age_level_homephone}
Number of days since an application with that entity has been seen. Entities are {social security number, address, date of birth, home phone number, name, full address, name & date of birth/ full address/ home phone number, full address & date of birth/ home phone number, date of birth & home phone number/ home phone number with name, age level & address/ home phone number, social security number & first name/ last name/ address/ 5-digit zip code/ date of birth/ home phone number/ age level/ name/ full address/ name & date of birth/ age level & address/ age level & home phone number}	27	~ 5 mins	{ssn, address, dob, homephone, name, fulladdress, name_dob, name_fulladdress, name_homephone, fulladdress_dob, fulladdress_homephone, dob_homephone, homephone_name_dob, age_level_address, age_level_homephone, ssn_firstname, ssn_lastname, ssn_address, ssn_zip5, ssn_dob, ssn_homephone, ssn_age_level, ssn_name, ssn_fulladdress, ssn_name_dob, ssn_age_level_address,ssn_age_level_homephone}_day_since
Number of applications at that entity over the past n days. Entities are {social security number, address, date of birth, home phone number, name, full address, name & date of birth/ full	162	~ 47 min	{ssn, address, dob, homephone, name, fulladdress, name_dob, name_fulladdress, name_homephone, fulladdress_dob, fulladdress_homephone, dob_homephone, homephone_name_dob,

address/ home phone number, full address & date of birth/ home phone number, date of birth & home phone number/ home phone number with name, age level & address/ home phone number, social security number & first name/ last name/ address/ 5-digit zip code/ date of birth/ home phone number/ age level/ name/ full address/ name & date of birth/ age level & address/ age level & home phone number}, n is {0,1,3,7,14,30}			age_level_address, age_level_homephone, ssn_firstname, ssn_lastname, ssn_address, ssn_zip5, ssn_dob, ssn_homephone, ssn_age_level, ssn_name, ssn_fulladdress, ssn_name_dob, ssn_age_level_address,ssn_age_level_homephone}_count_{0,1,3,7,14,30}
Number of applications at that entity seen in a short time window {today or past day} compared to number of applications at that entity over a longer time window {past 3,7,14,30 days}. Entities are {social security number, address, date of birth, home phone number, name, full address, name & date of birth/ full address/ home phone number, full address & date of birth/ home phone number, date of birth & home phone number/ home phone number with name, age level & address/ home phone number, social security number & first name/ last name/ address/ 5-digit zip code/ date of birth/ home phone number/ age level/ name/ full address/ name & date of	216	~ 22.58 s	{ssn, address, dob, homephone, name, fulladdress, name_dob, name_fulladdress, name_homephone, fulladdress_dob, fulladdress_homephone, dob_homephone, homephone_name_dob, age_level_address, age_level_homephone, ssn_firstname, ssn_lastname, ssn_address, ssn_zip5, ssn_dob, ssn_homephone, ssn_age_level, ssn_name, ssn_fulladdress, ssn_name_dob, ssn_age_level_address,ssn_age_level_homephone}_count_{0,1}_by_{3,7,14,30}

birth/ age level & address/ age level & home phone number}			
Number of unique identity in the specific field during the past {1,3,7,14,30,60} days. Both entity and field are {social security number, full address, name & date of birth/ full address, full address & date of birth, date of birth & home phone number, age level & address/ home phone number, social security number & last name/ 5-digit zip code/ age level/ name/ full address/ name & date of birth/ age level & address/ age level & home phone number}, but with different values.	1440	~ 88 mins	{ssn, fulladdress, name_dob, name_fulladdress, fulladdress_dob, dob_homephone, age_level_address, age_level_homephone, ssn_lastname, ssn_zip5, ssn_age_level, ssn_name, ssn_fulladdress, ssn_name_dob, ssn_age_level_address,ssn_age_level_homephone}_unique_count_in_{ ssn, fulladdress, name_dob, name_fulladdress, fulladdress_dob, dob_homephone, age_level_address, age_level_homephone, ssn_lastname, ssn_zip5, ssn_age_level, ssn_name, ssn_fulladdress, ssn_name_dob, ssn_age_level_address,ssn_age_level_homephone}_for_{1,3,7,14,30,60}
Total	1873	~ 140 mins	

A.6 References

1) Center for Identity Management and Information Protection. “Identity Crimes - Most Common Schemes.” *CIMIP*, Utica University, https://www.utica.edu/academic/institutes/cimip/idcrimes/schemes.cfm.