# FRAUD ANALYTICS PROJECT 2

Product Application Data


## ABSTRACT

Design a supervised fraud model on the product application data



## Team 9

Anni Cai, Suraj Patel, Yuyao Shen, Nanchun Shi, Bingru Xue

# Contents

# Executive Summary

The purpose of this project is to look for identity fraud in the "Product Application Data" file using supervised models. Identity fraud is the act of misrepresenting which person you are. The purpose of conducting identity fraud is to improperly get products or services or to stay “unidentified” while behaving badly.

We built several linear and non-linear models to predict fraud applications. We used our linear model (logistic regression) as the baseline for measuring the performance of other non-linear models. To quantitatively assess all model performances, we calculated the fraud detection rate (FDR) at a 3% examination cutoff location for each model. Fraud detection rate (FDR) is the percentage of fraud applications caught by looking at a certain percentage of overall applications. The higher the fraud detection rate at lower examination cut off the lower the cost of rejecting a smaller population of applications while avoiding the high rate of fraud applications. A good fraud detection model will help in quicker risk assessment in applications which in turn might lead to enhanced automation resulting in fewer manual reviews and fewer fraudulent gain of products or service.

The raw dataset "Product Application Data" file is a dataset that consists of historical application records for a product from January 1, 2016, to December 31, 2016. The dataset contains Personal Identifying Information of applicants, including name, social security number, date of birth, address, and home phone number. Each application record has a fraud label attached indicating whether it is a fraudulent application or not.

The report starts with a detailed description of the dataset. The full data quality report is attached as an appendix for reference. In our data quality report, we find that the dataset has 10 fields and 1,000,000 records. All of these 10 fields are categorical fields. All of these fields have no missing information. But these fields have frivolous values which are most likely created at the time of data gathering to replace the missing value information.

Hence, after data description, the report explains the data cleaning process. For this particular dataset, as we know there are no missing values in any of the fields, but some fields contain frivolous elements. We identify and replace these frivolous field values with unique, unlinked values to avoid getting false alarms. We have carefully decided on this unique unlinked value so that it does not carry any meaning or trigger false alarms in our model building process. To build a strong model we need expert variables, expert variables are variables that combine multiple variables and provide important information to build predictive models.

After the data cleaning process, we created 244 candidate variables using the initial 10 variables. First, we created entities for linking like Social Security Number, full address i.e. address with zip code, name-dob which is a combination of name and date of birth, and phone number. We also created various combination identity groups by combining two entities for linking like Social Security Number and full address and so on. Then, we created velocity variables, day since variables, and relative velocity variables based on these groups. We also created a risk table variable for the day of the week, that is the probability of a fraud application on a particular day of the week. We have built these expert variables to allow our models to better classify the data and identify trends quickly.

As there are many expert variables and a limited number of records there is a need for dimension reduction. Hence, we run a series of feature selection methods to reduce dimensionality. We first filter variables by calculating their univariate KS and FDR. Then we rank them based on the KS, FDR and their combined score. Then we continue reducing variables by using the forward selection logistic regression method.

Then we separate the whole dataset by reserving the last 2 months' records as validation/Out of Time data (oot) and the rest as modeling (training and test) data. Then, we build a baseline Logistic regression model on the training dataset and test the FDR at a 3% cutoff population for training, test and oot validation sample. Then, we explore some nonlinear models with parameters tuning on the training dataset, including Logistic Regression, KNN, Decision Tree, Neural Networks, Boosted Trees, and Random Forest and the predict the FDR at 3% cutoff population similar to logistic regression. We find that the Boosted Trees gives the highest fraud detection rate of 51.8% at a 3% penetration rate on the out-of-time validation dataset.

# Description of Data

This is a product application dataset that contains artificial identity information about applicants and a fraud label for each application. The time period this dataset covers is from January 1, 2016 to December 31, 2016. It has 10 fields and 1,000,000 records. The 10 fields are:

1. Record - represents the record number in the dataset
2. date - date of the application
3. ssn - social security number of the applicant
4. firstname - first name of the applicant
5. lastname - last name of the applicant
6. address - resident address of the applicant
7. zip5 - 5-digit zip code pertaining to the address of the applicant
8. dob - date of birth of the applicant
9. homephone - the home phone number of the applicant
10. fraud_label - Label indicating whether the application is fraud. (0: Not Fraud, 1: Fraud)

**Summary Table**

| Field Name | # of Records | % Populated | # Unique Values | Most Common Value |
|---|---|---|---|---|
| record | 1000000 | 100% | 1000000 | - |
| date | 1000000 | 100% | 365 | 20160816 |
| ssn | 1000000 | 100% | 835819 | 999999999 |
| firstname | 1000000 | 100% | 78136 | EAMSTRMT |
| lastname | 1000000 | 100% | 177001 | ERJSAXA |
| address | 1000000 | 100% | 828774 | 123 MAIN ST |
| zip5 | 1000000 | 100% | 26370 | 68138 |
| dob | 1000000 | 100% | 42673 | 19070626 |
| homephone | 1000000 | 100% | 28244 | 9999999999 |
| fraud_label | 1000000 | 100% | 2 | 0 |

**Fraud label Distribution**

| Fraud Label | Count |
|---|---|
| 0 | 985,607 |
| 1 | 14,393 |

The detailed field distribution and description of fraud label and other variables can be found in appendix.

# Data Cleaning

All fields are 100% populated in this particular dataset. However, after visualizing the distribution of each field, we identified some common made-up values i.e. frivolous values in certain fields that appeared extremely more frequently than others. For example:

- 123 main street (address)
- 999999999 (SSN)
- 9999999999 (homephone)
- 19070626 (dob)

We do not want these made-up values to trigger false alarms. Therefore, we replaced these frivolous values with their record numbers. The philosophy of this replacement is that each record number is unique and is unlinked to values in other fields. Also, as we have a maximum of 1000000 records which is a 7-digit letter there is no possibility of duplication for any of these variables as either they are more than 7-digit numbers or alpha-numeric combination.

After the data cleaning process, our dataset does not contain any outliers, missing fields or frivolous field values.

# Candidate Variables

We started by creating three new concatenated fields to avoid two people accidentally share the same full name or the same address but from different cities:

**name** = firstname + lastname

**namedob** = firstname + lastname + dob

**full address** = address + zip

One mode of identity fraud is identity theft. Some common signals for identity fraud would be one address associated with multiple names, dob, and ssn. Thus, we created following combination groups as unique identifiers:

**name_fulladdress** = name + full address

**name_homephone** = name + home phone

**fulladdress_dob** = full address + date of birth

**fulladdress_homephone** = full address + home phone

**homephone_name_dob** = home phone + name + date of birth

**ssn_firstname** = social security number + first name

**ssn_lastname** = social security number + last name

And for each combination group, we created the following variables:

**1. Days since last seen**: Day since variables represent the count of days since we last saw a particular value of the entity or combination group. We believe a small-time gap between appearances of the same entity could indicate a high possibility of fraud. For example, if ssn_day_since equals to zero (the # of days between when this record happened and the beginning of 2016), it means that we have never encountered this Social Security Number before. Following are some examples of Day since variables and a complete list is in the appendix.

- **ssn_day_since:** number of days since we last saw this social security number
- **address_day_since:** number of days since we last saw this address
- **name_dob_day_since:** number of days since we last saw this name and date of birth combination group
- **ssn_zip5_day_since:** number of days since we last saw this social security number and 5-digit zip code combination group

**2. Velocity**: Velocity variables represent how many times we have seen that entity or combination group in a particular time window. We set different time windows, including 1, 3, 7, 14, 30 days before the record happened. We believe repeated appearance of entities could indicate a high possibility of being fraudulent. For example, if address_count_7 is equal to 3, it indicates that this is the third time we have seen this address in the last 7 days. Following are some examples of Velocity variables and a complete list of variables is in the appendix.

- **name_count_3:** how many times we have seen this applicant name is the past three days
- **name_homephone_count_30:** how many times have we seen this name and home phone number combination group in the past 30 days
- **ssn_fulladdress_count_14:** How many times have we seen this social security number and full address combination in the past 14 days

**3. Relative velocity**: The formula to calculate relative velocity is defined as follows. We would like to see the comparison between a specific applicant's recent activity and long-term activity. If an applicant has much more frequent applications than his/her historical average in the recent period, it sets an alarm for potential fraud.

$$Relative\ Velocity = \frac{\#app\ with\ that\ group\ seen\ in\ the\ recent\ past}{\#apps\ with\ that\ same\ group\ seen\ over\ the\ past\{3,7,14,30\}\ days}$$

**4. Risk table variables (denoted as dow_risk)**. We created a risk table variable "day of week" that represents the percentage of frauds for each day of week. We want to introduce our out-of-time set (oot) here. We decided to keep all records happening in and after November 2016 as out-of-time set, which we won't use any information on nor train on. The purpose of this is to hold out data from machine learning models training. By doing so, the prediction accuracies on this set would be a good measure of model performance on data that it has never seen before.

To calculate risk table variables, we only applied aggregation on records that are not in the out-of-time set. In other words, we calculated percentages of frauds for each day of the week across all records except out of time records.

We used the following smooth transition function:

$$\text{Value} = Y_{\text{low}} + \frac{Y_{\text{high}} - Y_{\text{low}}}{1 + e^{-(n - n_{mid})/c}}$$

where $Y_{low}$ is the fraud rate over all records (except oot), $\boldsymbol{Y_{high}}$ is the fraud rate for that day of the week, $\boldsymbol{n}$ is the total number of records in each group, e.g. the number of records happened on Fridays, $\boldsymbol{n_{mid}}$ and $\boldsymbol{c}$ are parameters to adjust the shape of the function.

The rationale for this is when a group has not enough records, the aggregation values might become misrepresentative. For example, if we only have 5 records happening on Friday, it would be dangerous if we simply calculated the fraud rate for Friday on those 5 records. Instead, we would like to use the overall fraud rate for future use. So basically, the function would return the overall rate when there are too few records in a group.

The risk table we got is as follows:

| Risk Table | |
|---|---|
| **Weekday** | Risk |
| **Sunday** | 0.01367 |
| **Monday** | 0.01348 |
| **Tuesday** | 0.01407 |
| **Wednesday** | 0.01517 |
| **Thursday** | 0.01498 |
| **Friday** | 0.0145 |
| **Saturday** | 0.01497 |

Below is a glimpse of our feature summary table. You could find the complete version in appendix.

| Features | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **fraud_label** | 1000000 | 0.01439 | 0.1191 | 0 | 0 | 0 | 0 | 1 |
| **dow_risk** | 1000000 | 0.01441 | 0.00062 | 0.01348 | 0.01367 | 0.0145 | 0.01498 | 0.01517 |
| **ssn_day_since** | 1000000 | 163.702 | 105.164 | 0 | 72 | 155 | 251 | 365 |
| **ssn_count_0** | 1000000 | 1.00731 | 0.22336 | 1 | 1 | 1 | 1 | 21 |
| **ssn_count_1** | 1000000 | 1.01492 | 0.38121 | 1 | 1 | 1 | 1 | 34 |
| **ssn_count_3** | 1000000 | 1.02014 | 0.42319 | 1 | 1 | 1 | 1 | 34 |
| **ssn_count_7** | 1000000 | 1.02635 | 0.45356 | 1 | 1 | 1 | 1 | 34 |
| **ssn_count_14** | 1000000 | 1.03459 | 0.47765 | 1 | 1 | 1 | 1 | 34 |
| **ssn_count_30** | 1000000 | 1.05083 | 0.51333 | 1 | 1 | 1 | 1 | 34 |
| **address_day_since** | 1000000 | 160.231 | 105.49 | 0 | 67 | 150 | 248 | 365 |
| **address_count_0** | 1000000 | 1.01194 | 0.28329 | 1 | 1 | 1 | 1 | 24 |
| **address_count_1** | 1000000 | 1.02467 | 0.48295 | 1 | 1 | 1 | 1 | 30 |
| **address_count_3** | 1000000 | 1.0331 | 0.5443 | 1 | 1 | 1 | 1 | 30 |
| **address_count_7** | 1000000 | 1.04255 | 0.58575 | 1 | 1 | 1 | 1 | 30 |

# Feature Selection Process

After variable creation, we had 244 candidate variables. We have fixed records of 1000000 applications. If we have to build a model with 244 variables and 1000000 rows there will be a problem of dimensionality as the rule of thumb says we need 10^x records for x variables in the dataset. Hence, to reduce dimensionality we applied two feature selection methods: **Filter** and **Wrapper**.

Filters are univariate functions. They only look at one single feature at time and measure how good it is for predicting the labels. In this project, our team applied two common filters: Kolmogorov Smirnov (KS) and Fraud Detection Rate (FDR).

## I. Filter

**Kolmogorov Smirnov**

KS looks at distributions of good (non-fraudulent) records and bad (fraudulent) records. More specifically, KS plots goods and bads over the current variable value and sees how well the two distributions separated. The KS score ranges from 0 to 1. If the score is close to 1, it means the distributions are well separated and this indicates the current feature is potentially good for predicting the label. The formula for KS is as follows:

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

In short, KS returns the maximum difference between the cumulative distribution functions (CDF) for the two distributions.

We then ranked the candidate features by **KS** scores, and here are the top 10:

| Field | KS | FDR | KS rank | FDR rank | Average rank |
|---|---|---|---|---|---|
| label | 1.000000 | 1.000000 | 245.0 | 245.0 | 245.0 |
| address_count_30 | 0.332725 | 0.353300 | 244.0 | 243.0 | 243.5 |
| fulladdress_count_30 | 0.332032 | 0.354954 | 243.0 | 244.0 | 243.5 |
| address_day_since | 0.324627 | 0.348076 | 242.0 | 241.0 | 241.5 |
| fulladdress_day_since | 0.323543 | 0.349382 | 241.0 | 242.0 | 241.5 |
| address_count_14 | 0.322252 | 0.345812 | 240.0 | 240.0 | 240.0 |
| fulladdress_count_14 | 0.321756 | 0.342330 | 239.0 | 239.0 | 239.0 |
| address_count_7 | 0.301445 | 0.320999 | 238.0 | 238.0 | 238.0 |
| fulladdress_count_7 | 0.301368 | 0.319955 | 237.0 | 237.0 | 237.0 |
| fulladdress_count_3 | 0.278488 | 0.297493 | 236.0 | 235.0 | 235.5 |

**Fraud Detection Rate**

FDR is the percentage of real fraud caught by examining a certain percentage of the population by using only one feature. The higher the percentage is, the higher the importance of predicting its label.

More specifically, we ranked real fraud labels by the value of a feature and calculated how many fraud labels caught by this ranking among the x% of the population. As suggested by the field expert, we used x = 3 in this project. We divided the number of fraud caught in the 3% of the population by the total number of fraud records to get the FDR. Here, since we didn't know if a feature is positively or negatively related to the fraud label, we compared the FDR among the top 3% and the FDR among the bottom 3% and picked the higher value. FDR is a value from 0 to 1. If the FDR at 3% of a feature is higher compared to the others, it would mean the feature is potentially better for predicting the label.

We then ranked the candidate features by **FDR** at 3%, and here are the top 10:

| Field | KS | FDR | KS rank | FDR rank | Average rank |
|---|---|---|---|---|---|
| label | 1.000000 | 1.000000 | 245.0 | 245.0 | 245.0 |
| fulladdress_count_30 | 0.332032 | 0.354954 | 243.0 | 244.0 | 243.5 |
| address_count_30 | 0.332725 | 0.353300 | 244.0 | 243.0 | 243.5 |
| fulladdress_day_since | 0.323543 | 0.349382 | 241.0 | 242.0 | 241.5 |
| address_day_since | 0.324627 | 0.348076 | 242.0 | 241.0 | 241.5 |
| address_count_14 | 0.322252 | 0.345812 | 240.0 | 240.0 | 240.0 |
| fulladdress_count_14 | 0.321756 | 0.342330 | 239.0 | 239.0 | 239.0 |
| address_count_7 | 0.301445 | 0.320999 | 238.0 | 238.0 | 238.0 |
| fulladdress_count_7 | 0.301368 | 0.319955 | 237.0 | 237.0 | 237.0 |
| address_count_3 | 0.278445 | 0.299060 | 235.0 | 236.0 | 235.5 |

We also ranked by the **average** of the two rankings:

| Field | KS | FDR | KS rank | FDR rank | Average rank |
|---|---|---|---|---|---|
| label | 1.000000 | 1.000000 | 245.0 | 245.0 | 245.0 |
| address_count_30 | 0.332725 | 0.353300 | 244.0 | 243.0 | 243.5 |
| fulladdress_count_30 | 0.332032 | 0.354954 | 243.0 | 244.0 | 243.5 |
| fulladdress_day_since | 0.323543 | 0.349382 | 241.0 | 242.0 | 241.5 |
| address_day_since | 0.324627 | 0.348076 | 242.0 | 241.0 | 241.5 |
| address_count_14 | 0.322252 | 0.345812 | 240.0 | 240.0 | 240.0 |
| fulladdress_count_14 | 0.321756 | 0.342330 | 239.0 | 239.0 | 239.0 |
| address_count_7 | 0.301445 | 0.320999 | 238.0 | 238.0 | 238.0 |
| fulladdress_count_7 | 0.301368 | 0.319955 | 237.0 | 237.0 | 237.0 |
| fulladdress_count_3 | 0.278488 | 0.297493 | 236.0 | 235.0 | 235.5 |

We noticed that there was no significant variance between the results from the two rankings, which assured us that our calculations were reliable.

## II. Wrapper

While filter functions are univariate, which means they only take care of relationship between a single variable and the fraud labels, we would like to select a group of features that could work well collaboratively on predicting the labels. To achieve this, we used wrappers. Wrappers use machine learning models to measure the performance of a certain combination of features. In other words, given a specified estimator model, wrappers could compare the performance of different combinations of features on predicting labels using different evaluation criterions. We used two kinds of wrappers in this project: **Stepwise selection** and **Forward selection**.

**Stepwise selection:**

Using the ranking result from filters, we selected the top 80 variables since we didn't know how many variables needed for optimizing prediction. We started by applying a stepwise logistic wrapper, where it used a logistic regression model as the estimator and accuracy scores as evaluation criterion. More specifically, the wrapper starts by building single variable logistic regression of available features, so in our case, there are 244 of them, and then it picks the feature with the best accuracy. Then it builds two-variable logistic regression using the one it just picked with all the other features and picked the best combination and so on. While adding a new feature, it will check the significance of already added features and if any of the variables become insignificant in terms of the prediction accuracy, the wrapper will remove it. In short, stepwise wrapper will not keep features that are highly correlated to the others.

The method we used is RFECV from scikit learn. For each iteration, the function would calculate the cross-validation score, a measure of how well the model performance evaluated by prediction accuracy. We plot the number of features kept against cross validations scores and found that 15-25 variables would be good enough for predictions.

**Forward selection:**

A drawback of RFECV is we could not know the order of importance of each selected feature. Therefore, we could not determine the final list of features to use since the wrapper would always return a number of features more than needed. Knowing the optimal number of features to keep, we then used a forward selection function from SequentialFeatureSelector of mlxtend library, as it allows us to specify the number of features to keep. The logic behind forward selection is similar to stepwise selection. It will start by building simple models and keep adding significant features. As results, it will return an optimal list containing a specified number of features.

We used logistic regression again as the estimator. Since we would build nonlinear models in our modeling stage, here we wanted to keep it simple using only linear models. We specified the wrapper to keep 25 features, which is the upper bound of the optimal number of features we got from the previous step to be conservative. Below is the out final list of features:

| | Final Features |
|---|---|
| 0 | address_count_30 |
| 1 | fulladdress_count_30 |
| 2 | fulladdress_count_3 |
| 3 | address_count_3 |
| 4 | fulladdress_count_1 |
| 5 | ssn_name_dob_count_30 |
| 6 | fulladdress_homephone_day_since |
| 7 | fulladdress_homephone_count_14 |
| 8 | name_dob_day_since |
| 9 | ssn_name_dob_day_since |
| 10 | name_dob_count_14 |
| 11 | ssn_name_day_since |
| 12 | ssn_dob_count_14 |
| 13 | ssn_count_14 |
| 14 | ssn_firstname_count_14 |
| 15 | ssn_lastname_count_14 |
| 16 | ssn_name_count_14 |
| 17 | address_count_1_by_14 |
| 18 | fulladdress_count_1_by_14 |
| 19 | name_count_14 |
| 20 | name_day_since |
| 21 | fulladdress_count_1_by_30 |
| 22 | fulladdress_count_1_by_7 |
| 23 | ssn_name_count_1_by_30 |
| 24 | dob_count_30 |

# Model Algorithms

We first applied the upsampling method to improve the distribution of our data because we noticed the dataset was imbalanced, with only approximately 1.4% fraudulent records. Strongly imbalanced dataset could cause the models to spend too much time on learning one class while not learning enough from the other. Therefore, we upsampled the fraudulent records by randomly duplicating existing them and produced a new training sample with approximately 35% bads. We used this treated training set to train our subsequent models.

## Logistic Regression

The first algorithm we applied is the logistic regression with L1 and L2 (Lasso and Ridge) regularization feature selection.

Ridge Regression

$$\sum_{i=1}^{M}(y_i-\hat{y}_i)^2=\sum_{i=1}^{M}\left(y_i-\sum_{j=0}^{p}w_j\times x_{ij}\right)^2+\lambda\sum_{j=0}^{p}w_j^2$$

Ridge regression puts constraint on the coefficients (w). The penalty term (lambda) regularizes the coefficients such that if the coefficients take values the optimization function is penalized. So, ridge function shrinks the coefficients and it helps to reduce the model complexity and multicollinearity.

Lasso Regression

$$\sum_{i=1}^{M}(y_i-\hat{y}_i)^2=\sum_{i=1}^{M}\left(y_i-\sum_{j=0}^{p}w_j\times x_{ij}\right)^2+\lambda\sum_{j=0}^{p}|w_j|$$

Just like Ridge regression, Lasso Regression also shrinks the coefficient. The only difference is that instead of taking the square of the coefficients, magnitudes are taken into account. This type of regularization can lead to zero coefficients. So, Lasso regression not only helps in reducing overfitting but it can help us in feature selection.

To choose from L1 and L2, we first tried the two types of regularization at the default inverse of regularization strength (C), which equals to 1. It turns out that L1, or Lasso Regression, works better on detecting fraud at 3% level, therefore we decided to go for this methodology.

Then we used SelectFromModel() function with Lasso Regression as the estimator to perform feature selection. We trained the model on our balanced training dataset at a C level of 1, 0.05, 0.01, 0.005 and 0.001 respectively, which correspond to 25, 22, 20, 18 and 13 resulting variables. We transformed the training sample and used the transformed one to train a regular logistic regression model to predict the probability of fraud. We sorted the train, test and oot samples by the probability of fraud in a descending manner and calculated the FDR rate at 3% level. It turns out that 20 variables will result in the highest FDR at 3% level for train, test and oot samples, which are 0.5159, 0.5048 and 0.4904.

| Model | Parameters | | Average FDR (%) at 3% | | |
|---|---|---|---|---|---|
| **Logistic Regression** | **Total Variables** | **Number of Variables Selected** | **Train** | **Test** | **OOT** |
| **1** | 25 | 25 | 51.47 | 50.31 | 48.91 |
| **2** | 25 | 22 | 51.33 | 50.2 | 48.53 |
| **3** | 25 | 20 | 51.59 | 50.48 | 49.04 |
| **4** | 25 | 18 | 50.74 | 49.58 | 48.16 |
| **5** | 25 | 13 | 49.51 | 48.59 | 48.41 |

## KNN

We also tried K-Nearest Neighbors which is an algorithm based on feature similarity. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

| Model | Parameters | | FDR (%) at 3% | | |
|---|---|---|---|---|---|
| **KNN** | **Number of Variables** | **n** | **Train** | **Test** | **OOT** |
| **1** | 25 | 10 | 62.41 | 48.79 | 47.53 |
| **2** | 25 | 20 | 61.96 | 49.3 | 48.03 |
| **3** | 25 | 50 | 58.99 | 49.8 | 48.49 |
| **4** | 25 | 100 | 55.09 | 50.06 | 48.91 |
| **5** | 25 | 150 | 53.98 | 50.23 | 48.83 |
| **6** | 25 | 200 | 53.71 | 50.34 | 48.78 |

## Boosted trees

Then we applied Gradient Boosting tree, which is a one of the leading ensemble algorithms by training a series of weak learners (tree) to result in a strong learner. Gradient boosting algorithm uses gradient descent method to optimize the loss function, for our case, logistic regression for classification. What we do here is fitting an additive model (ensemble) $\sum_t p_t h_t(x)$ in a forward stage-wise manner to the

original model. In each stage, we introduced a weak learner to compensate the shortcomings (residual error) of existing weak learners. There are three primary parameters we have been tuning: number of trees, learning rate and max depth. The number of sequential trees is usually high to make the model robust but as we can see here, when the number of trees greater than 800, the FDR decreases. Learning rate determines the impact of each tree on the final outcome. Lower values are usually better for generalization but require higher number of trees. Max depth indicates the maximum depth of a tree. Higher depth might lead to overfitting.

| Model | Parameters | | | | FDR (%) at 3% | | |
|---|---|---|---|---|---|---|---|
| Boosted Tree | Number of Variables | # of Trees | Max Depth | Learning Rate | Train | Test | OOT |
| **1** | 25 | 600 | 4 | 0.01 | 53.62 | 51.8 | 50.96 |
| **2** | 25 | 600 | 4 | 0.005 | 52.61 | 51.1 | 50.34 |
| **3** | 25 | 600 | 2 | 0.1 | 53.97 | 52.14 | 51.51 |
| **4** | 25 | 800 | 4 | 0.01 | 54.09 | 51.89 | 51.63 |
| **5** | 25 | 800 | 3 | 0.01 | 53.53 | 51.92 | 51.8 |
| **6** | 25 | 800 | 5 | 0.005 | 53.54 | 51.92 | 51.68 |
| **7** | 25 | 800 | 5 | 0.002 | 51.89 | 51.96 | 49.58 |
| **8** | 25 | 800 | 3 | 0.1 | 55.78 | 51.89 | 50.96 |
| **9** | 25 | 1000 | 4 | 0.01 | 54.43 | 51.94 | 51.63 |
| **10** | 25 | 1000 | 3 | 0.001 | 51.44 | 50.2 | 48.87 |

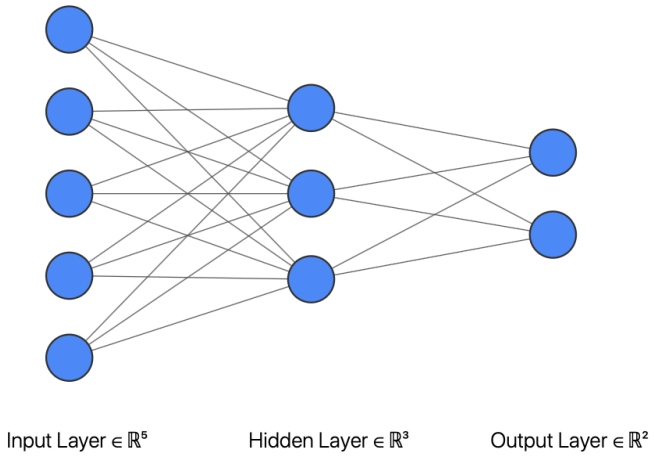
## Random forest

Then we built a random forest, which is another ensemble model using trees but unlike boosted trees random forest is an ensemble of strong learners. Random forest can be used for classification as well as regression. The random forest is a combination of hundreds of or millions of multiple trees, trains each one on a slightly different set of observations, splitting nodes in each tree considering a limited number of features. The final prediction of the tree is made by averaging the prediction of each individual tree. We have tried multiple combinations of different parameters to tune the model like number of trees which represents the total number of trees in the random forest, mtry which represents the maximum number of features to consider while looking for best split and min node size which represents the minimum number of samples required to be at a node.

| Model | Parameters | | | | FDR (%) at 3% | | |
|---|---|---|---|---|---|---|---|
| Random Forest | Number of Variables | Number of trees | Max features | Min samples leaf | Train | Test | OOT |
| **1** | 25 | 200 | 10 | 3 | 63.99 | 50.7 | 49.58 |
| **2** | 25 | 200 | 5 | 3 | 63.94 | 50.79 | 49.71 |
| **3** | 25 | 400 | 10 | 3 | 64.02 | 50.73 | 49.41 |
| **4** | 25 | 400 | 5 | 3 | 63.94 | 50.82 | 49.79 |

## Neural networks

We then built simple neural networks with 1 or 2 hidden layers. The mechanism of neural networks is matrix multiplication. The model takes out data as input, applies linear transformation using weights, catches nonlinear patterns by applying activation functions, and finally outputs the results. A simple example diagram of neural networks could be the following:



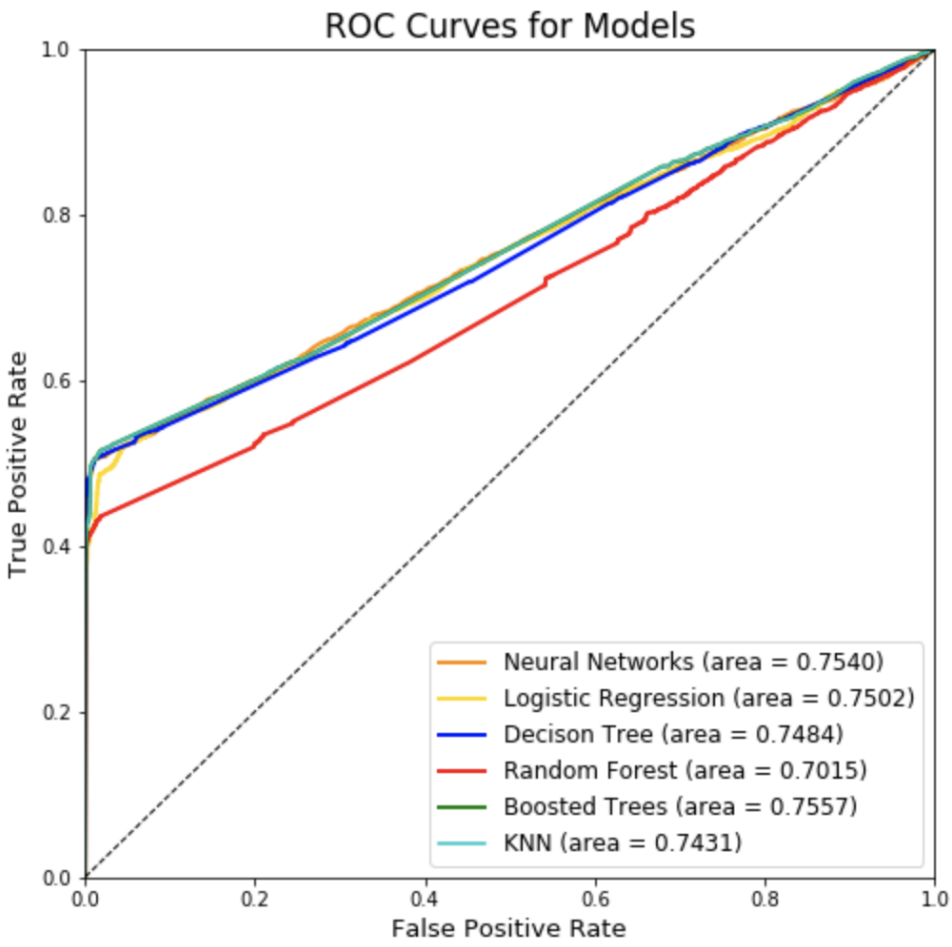

Each edge has a weight, and values in the current layer come from the matrix transformation from previous layers. In the project, we used two nodes in the output layer since we were conducting a binary classification, where the predicted probabilities for each class would be a node. To optimize the weights to get the best performance on predicting labels, the model used the idea of gradient descent. This concept is commonly used for finding global minima of a loss function. The loss function measures how far are the model's predictions away from real labels, and therefore the lower the value the better the performance. The model iterated many times through the training set and learned the direction of gradient of the loss function each time. It has been mathematically proven that the direction of the gradient of a function at a point is the direction with the largest change. Therefore, the model calculates the gradient at each iteration and updates all weights by moving the previous weights to the opposite of the gradient (since we want to find the minima).

A complex architecture of networks could result in overfitting issues. Since our input dimension is relatively small, we focused on the number of nodes in (6,20) and didn't go beyond 2 layers. Here is the result table of our neural network models:

| Model | Parameters | | | FDR (%) at 3% | | |
|---|---|---|---|---|---|---|
| **Neural Networks** | **Number of Variables** | **Nodes** | **Layers** | **Train** | **Test** | **OOT** |
| **1** | 30 | 6 | 1 | 52.62 | 53.23 | 51.63 |
| **2** | 30 | 15 | 1 | 52.5 | 53.26 | 51.22 |
| **3** | 25 | 20 | 1 | 52.72 | 53.01 | 51.8 |
| **4** | 25 | 6 | 1 | 52.44 | 53.18 | 51.72 |
| **5** | 25 | 15 | 1 | 52.67 | 53.12 | 51.63 |
| **6** | 20 | 20 | 1 | 52.7 | 53.15 | 51.26 |
| **7** | 20 | 6 | 1 | 52.21 | 53.26 | 51.51 |
| **8** | 20 | 10 | 1 | 52.43 | 53.12 | 51.59 |
| **9** | 25 | 13 | 1 | 52.78 | 53.48 | 51.8 |
| **10** | 25 | 6,6 | 2 | 52.39 | 53.21 | 51.63 |
| **11** | 25 | 10,6 | 2 | 52.38 | 53.29 | 51.42 |
| **12** | 25 | 10,10 | 2 | 52.57 | 53.09 | 51.51 |
| **13** | 25 | 15,10 | 2 | 52.46 | 53.37 | 51.09 |

Finally, we made a ROC curve with all the models we tried above. ROC curve is the curve of False Positive Rate (Type I error) against True Positive Rate (1-Type II error). The best performance a model could achieve is at point (0, 1), where Type I and Type II errors both become 0. Thus, the closer the curve is to that point, the better we generally believe the model of that curve is. Here each line represents one type of model, as shown in the legend window. Each point on a curve represents the value of the two errors given a certain threshold. A threshold is a cutoff value decided to classify a record given its predicted probabilities. Every value above the threshold will be labeled as one class. For example, for a record with predicted probability of being class 1 (the fraudulent class in our case) is 0.6 and we set threshold to be 0.5, then the record would be labeled as a fraud.

To measure how close the curve is to the best point, we also looked at AUC (area under the curve). It's obtained by taking integrals of the ROC curves. Typically, the larger the area, the better the model. As seen below, Boosted Trees performed slightly better than any other model. This was also confirmed by the OOT FDR results we got above. Note the AUC for Boosted Trees may not be significantly different from neural networks' due to some randomness, but for the purpose of this project, we won't do further investigation for better distinguishment. We would like to leave it for future work. Therefore, our team decided to use **Boosted Trees** as our final model.


ROC Curves for Models
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Neural Networks (area = 0.7540)
Logistic Regression (area = 0.7502)
Decison Tree (area = 0.7484)
Random Forest (area = 0.7015)
Boosted Trees (area = 0.7557)
KNN (area = 0.7431)

# Results

After comparing all models, we selected the Gradient Boosting Tree algorithm which has FDR of 51.8% at 3% with 800 trees, 3 max depth and 0.01 learning rate on out-of-time dataset. Thus, we composed this final performance table for train, test and out-of-time populations separately:

| Training | # Records | # Goods | # Bads | Fraud Rate |
|---|---|---|---|---|
| | 583454 | 574997 | 8457 | 0.0145 |

| | Bin Statistics | | | | | Cumulative Statistics | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Population Bin% | # Records | # Goods | # Bads | % Goods | % Bads | Total # Records | Cumulative Goods | Cumulative Bads | % Goods | % Bads (FDR) | KS | FPR |
| 1 | 5834 | 1547 | 4287 | 26.52% | 73.48% | 5834 | 1547 | 4287 | 0.27% | 50.69% | 0.5042 | 0.36 |
| 2 | 5835 | 5690 | 145 | 97.51% | 2.49% | 11669 | 7237 | 4432 | 1.26% | 52.41% | 0.5115 | 1.63 |
| 3 | 5834 | 5740 | 94 | 98.39% | 1.61% | 17503 | 12977 | 4526 | 2.26% | 53.52% | 0.5126 | 2.87 |
| 4 | 5835 | 5774 | 61 | 98.95% | 1.05% | 23338 | 18751 | 4587 | 3.26% | 54.24% | 0.5098 | 4.09 |
| 5 | 5834 | 5778 | 56 | 99.04% | 0.96% | 29172 | 24529 | 4643 | 4.27% | 54.90% | 0.5063 | 5.28 |
| 6 | 5835 | 5765 | 70 | 98.80% | 1.20% | 35007 | 30294 | 4713 | 5.27% | 55.73% | 0.5046 | 6.43 |
| 7 | 5834 | 5765 | 69 | 98.82% | 1.18% | 40841 | 36059 | 4782 | 6.27% | 56.54% | 0.5027 | 7.54 |
| 8 | 5835 | 5763 | 72 | 98.77% | 1.23% | 46676 | 41822 | 4854 | 7.27% | 57.40% | 0.5013 | 8.62 |
| 9 | 5834 | 5784 | 50 | 99.14% | 0.86% | 52510 | 47606 | 4904 | 8.28% | 57.99% | 0.4971 | 9.71 |
| 10 | 5835 | 5777 | 58 | 99.01% | 0.99% | 58345 | 53383 | 4962 | 9.28% | 58.67% | 0.4939 | 10.76 |
| 11 | 5834 | 5774 | 60 | 98.97% | 1.03% | 64179 | 59157 | 5022 | 10.29% | 59.38% | 0.4909 | 11.78 |
| 12 | 5835 | 5769 | 66 | 98.87% | 1.13% | 70014 | 64926 | 5088 | 11.29% | 60.16% | 0.4887 | 12.76 |
| 13 | 5835 | 5771 | 64 | 98.90% | 1.10% | 75849 | 70697 | 5152 | 12.30% | 60.92% | 0.4862 | 13.72 |
| 14 | 5834 | 5780 | 54 | 99.07% | 0.93% | 81683 | 76477 | 5206 | 13.30% | 61.56% | 0.4826 | 14.69 |
| 15 | 5835 | 5789 | 46 | 99.21% | 0.79% | 87518 | 82266 | 5252 | 14.31% | 62.10% | 0.4779 | 15.66 |
| 16 | 5834 | 5784 | 50 | 99.14% | 0.86% | 93352 | 88050 | 5302 | 15.31% | 62.69% | 0.4738 | 16.61 |
| 17 | 5835 | 5789 | 46 | 99.21% | 0.79% | 99187 | 93839 | 5348 | 16.32% | 63.24% | 0.4692 | 17.55 |
| 18 | 5834 | 5772 | 62 | 98.94% | 1.06% | 105021 | 99611 | 5410 | 17.32% | 63.97% | 0.4665 | 18.41 |
| 19 | 5835 | 5790 | 45 | 99.23% | 0.77% | 110856 | 105401 | 5455 | 18.33% | 64.50% | 0.4617 | 19.32 |
| 20 | 5834 | 5783 | 51 | 99.13% | 0.87% | 116690 | 111184 | 5506 | 19.34% | 65.11% | 0.4577 | 20.19 |

| Test | # Records | # Goods | # Bads | Fraud Rate |
|---|---|---|---|---|
| | 250053 | 246503 | 8457 | 0.0338 |

| | Bin Statistics | | | | | Cumulative Statistics | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Population Bin% | # Records | # Goods | # Bads | % Goods | % Bads | Total # Records | Cumulative Goods | Cumulative Bads | % Goods | % Bads (FDR) | KS | FPR |
| 1 | 2500 | 737 | 1763 | 29.48% | 70.52% | 2500 | 737 | 1763 | 0.30% | 49.66% | 0.4936 | 0.42 |
| 2 | 2501 | 2457 | 44 | 98.24% | 1.76% | 5001 | 3194 | 1807 | 1.30% | 50.90% | 0.496 | 1.77 |
| 3 | 2500 | 2464 | 36 | 98.56% | 1.44% | 7501 | 5658 | 1843 | 2.30% | 51.92% | 0.4962 | 3.07 |
| 4 | 2501 | 2480 | 21 | 99.16% | 0.84% | 10002 | 8138 | 1864 | 3.30% | 52.51% | 0.4921 | 4.37 |
| 5 | 2500 | 2477 | 23 | 99.08% | 0.92% | 12502 | 10615 | 1887 | 4.31% | 53.15% | 0.4884 | 5.63 |
| 6 | 2501 | 2480 | 21 | 99.16% | 0.84% | 15003 | 13095 | 1908 | 5.31% | 53.75% | 0.4844 | 6.86 |
| 7 | 2500 | 2472 | 28 | 98.88% | 1.12% | 17503 | 15567 | 1936 | 6.32% | 54.54% | 0.4822 | 8.04 |
| 8 | 2501 | 2475 | 26 | 98.96% | 1.04% | 20004 | 18042 | 1962 | 7.32% | 55.27% | 0.4795 | 9.20 |
| 9 | 2500 | 2476 | 24 | 99.04% | 0.96% | 22504 | 20518 | 1986 | 8.32% | 55.94% | 0.4762 | 10.33 |
| 10 | 2501 | 2471 | 30 | 98.80% | 1.20% | 25005 | 22989 | 2016 | 9.33% | 56.79% | 0.4746 | 11.40 |
| 11 | 2500 | 2479 | 21 | 99.16% | 0.84% | 27505 | 25468 | 2037 | 10.33% | 57.38% | 0.4705 | 12.50 |
| 12 | 2501 | 2477 | 24 | 99.04% | 0.96% | 30006 | 27945 | 2061 | 11.34% | 58.06% | 0.4672 | 13.56 |
| 13 | 2500 | 2480 | 20 | 99.20% | 0.80% | 32506 | 30425 | 2081 | 12.34% | 58.62% | 0.4628 | 14.62 |
| 14 | 2501 | 2486 | 15 | 99.40% | 0.60% | 35007 | 32911 | 2096 | 13.35% | 59.04% | 0.4569 | 15.70 |
| 15 | 2500 | 2482 | 18 | 99.28% | 0.72% | 37507 | 35393 | 2114 | 14.36% | 59.55% | 0.4519 | 16.74 |
| 16 | 2501 | 2484 | 17 | 99.32% | 0.68% | 40008 | 37877 | 2131 | 15.37% | 60.03% | 0.4466 | 17.77 |
| 17 | 2501 | 2481 | 20 | 99.20% | 0.80% | 42509 | 40358 | 2151 | 16.37% | 60.59% | 0.4422 | 18.76 |
| 18 | 2500 | 2485 | 15 | 99.40% | 0.60% | 45009 | 42843 | 2166 | 17.38% | 61.01% | 0.4363 | 19.78 |
| 19 | 2501 | 2481 | 20 | 99.20% | 0.80% | 47510 | 45324 | 2186 | 18.39% | 61.58% | 0.4319 | 20.73 |
| 20 | 2500 | 2482 | 18 | 99.28% | 0.72% | 50010 | 47806 | 2204 | 19.39% | 62.08% | 0.4269 | 21.69 |

| OOT | # Records | # Goods | # Bads | Fraud Rate |
|---|---|---|---|---|
| | 166493 | 164107 | 2386 | 0.0143 |

| | Bin Statistics | | | | | Cumulative Statistics | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Population Bin% | # Records | # Goods | # Bads | % Goods | % Bads | Total # Records | Cumulative Goods | Cumulative Bads | % Goods | % Bads (FDR) | KS | FPR |
| 1 | 1664 | 503 | 1161 | 30.23% | 69.77% | 1664 | 503 | 1161 | 0.31% | 48.66% | 0.4835 | 0.43 |
| 2 | 1665 | 1620 | 45 | 97.30% | 2.70% | 3329 | 2123 | 1206 | 1.29% | 50.54% | 0.4925 | 1.76 |
| 3 | 1665 | 1635 | 30 | 98.20% | 1.80% | 4994 | 3758 | 1236 | 2.29% | 51.80% | 0.4951 | 3.04 |
| 4 | 1665 | 1655 | 10 | 99.40% | 0.60% | 6659 | 5413 | 1246 | 3.30% | 52.22% | 0.4892 | 4.34 |
| 5 | 1665 | 1656 | 9 | 99.46% | 0.54% | 8324 | 7069 | 1255 | 4.31% | 52.60% | 0.4829 | 5.63 |
| 6 | 1665 | 1648 | 17 | 98.98% | 1.02% | 9989 | 8717 | 1272 | 5.31% | 53.31% | 0.48 | 6.85 |
| 7 | 1665 | 1651 | 14 | 99.16% | 0.84% | 11654 | 10368 | 1286 | 6.32% | 53.90% | 0.4758 | 8.06 |
| 8 | 1665 | 1653 | 12 | 99.28% | 0.72% | 13319 | 12021 | 1298 | 7.33% | 54.40% | 0.4707 | 9.26 |
| 9 | 1665 | 1656 | 9 | 99.46% | 0.54% | 14984 | 13677 | 1307 | 8.33% | 54.78% | 0.4645 | 10.46 |
| 10 | 1665 | 1652 | 13 | 99.22% | 0.78% | 16649 | 15329 | 1320 | 9.34% | 55.32% | 0.4598 | 11.61 |
| 11 | 1665 | 1658 | 7 | 99.58% | 0.42% | 18314 | 16987 | 1327 | 10.35% | 55.62% | 0.4527 | 12.80 |
| 12 | 1665 | 1652 | 13 | 99.22% | 0.78% | 19979 | 18639 | 1340 | 11.36% | 56.16% | 0.448 | 13.91 |
| 13 | 1665 | 1650 | 15 | 99.10% | 0.90% | 21644 | 20289 | 1355 | 12.36% | 56.79% | 0.4443 | 14.97 |
| 14 | 1665 | 1649 | 16 | 99.04% | 0.96% | 23309 | 21938 | 1371 | 13.37% | 57.46% | 0.4409 | 16.00 |
| 15 | 1664 | 1655 | 9 | 99.46% | 0.54% | 24973 | 23593 | 1380 | 14.38% | 57.84% | 0.4346 | 17.10 |
| 16 | 1665 | 1652 | 13 | 99.22% | 0.78% | 26638 | 25245 | 1393 | 15.38% | 58.38% | 0.43 | 18.12 |
| 17 | 1665 | 1651 | 14 | 99.16% | 0.84% | 28303 | 26896 | 1407 | 16.39% | 58.97% | 0.4258 | 19.12 |
| 18 | 1665 | 1660 | 5 | 99.70% | 0.30% | 29968 | 28556 | 1412 | 17.40% | 59.18% | 0.4178 | 20.22 |
| 19 | 1665 | 1647 | 18 | 98.92% | 1.08% | 31633 | 30203 | 1430 | 18.40% | 59.93% | 0.4153 | 21.12 |
| 20 | 1665 | 1657 | 8 | 99.52% | 0.48% | 33298 | 31860 | 1438 | 19.41% | 60.27% | 0.4086 | 22.16 |

# Conclusions

We have built a model with a 51.8% fraud detection rate at a 3% penetration rate on oot data to identify fraud transactions in the product application dataset. This means that we can catch 51.8% of fraud transactions by flagging 3% of the overall dataset.

We first built a data quality report for the raw dataset of product applications that contained 10 variables and 1000000 records. In the data quality report, we defined the variables and looked at the properties and distributions of all the variables. We also found out that the data had no missing values.

Then, upon deep diving into the data cleaning process, we identified and replaced frivolous field values with unique, unlinked values to avoid getting false alarms. frivolous values are values added in the data gathering process to treat the missing values and usually don't carry any meaning but during our model building process, it can be interpreted as an actual input. Hence, we have carefully decided on this unique unlinked value so that it does not carry any meaning or trigger false alarms in our model building process.

After the data cleaning process, we created 244 expert variables by using the initial 10 variables of the dataset. First, we created various combinations of identity groups, such as entities for linking, and then we created velocity variables, day since variables, and relative velocity variables based on these groups. We also created a risk table variable for the day of the week, that is the probability of a fraud application on a particular day of the week.

As we now have 244 variables and just 1000000 records there is a problem of high dimensionality and there is a need to reduce the dimensionality of the dataset. Hence, we built a series of feature selection methods to reduce dimensionality. We calculated the KS and FDR score and ranked the variables based on their combined ranking and subset the dataset with the highest 100 variables. Then we further reduced the dimensionality by using the forward selection method. Then we reduced to 25 variables by using a forward selection logistic regression model. The forward selection logistic regression model runs iterations of models starting from the most informative variable and adds the next best variable until the predictability of the model increases with the addition of the variable.

Then we separate the whole dataset by reserving the last 2 months' records as validation/Out of Time data (oot) and the rest as modeling (training and test) data. we first built a linear model i.e. logistic regression on the training data and predicted the training, test and oot sample data. This linear model acts as a base for our prediction. We then built several non-linear models like KNN, Decision Tree, Boosted Tree, Random Forest and Neural Network on the training data and predicted the FDR at a 3% cutoff population for training, test and oot sample data. Based on these results the Boosted Trees has the highest FDR of 51.8% at a 3% oot population. Hence, the business can potentially reduce 51.8% of fraud transactions by rejecting 3% of applications. For the further scope of the project, we could have built an ensemble model by combining the results of all the models to predict the fraud transactions.

# Appendix

## Variables DQR

### Description

This dataset is a synthesized application data which contains the information about the applicant associated with each application. It has 1,000,000 records and 10 variables.

### Summary Table

| Field name | Field type | # of non-NA values | % populated | # unique values | Most Common Field |
|---|---|---|---|---|---|
| **record** | Categorical | 1,000,000 | 100 | 1,000,000 | N/A |
| **date** | Categorical | 1,000,000 | 100 | 365 | 2016-08-16 |
| **ssn** | Categorical | 1,000,000 | 100 | 835,819 | 999999999 |
| **firstname** | Categorical | 1,000,000 | 100 | 78,136 | EAMSTRMT |
| **lastname** | Categorical | 1,000,000 | 100 | 177,001 | ERJSAXA |
| **address** | Categorical | 1,000,000 | 100 | 828,774 | 123 MAIN ST |
| **zip5** | Categorical | 1,000,000 | 100 | 26,370 | 68138 |
| **dob** | Categorical | 1,000,000 | 100 | 42,673 | 19070626 |
| **homephone** | Categorical | 1,000,000 | 100 | 28,244 | 9999999999 |
| **fraud_label** | Categorical | 1,000,000 | 100 | 2 | 0 |

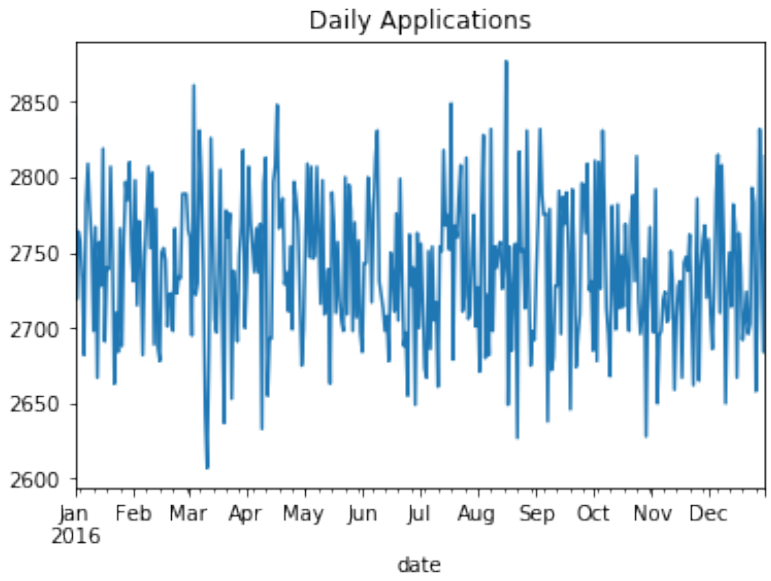
### Detailed Field Information

record:

Each record has a unique record number serving as an identification method.

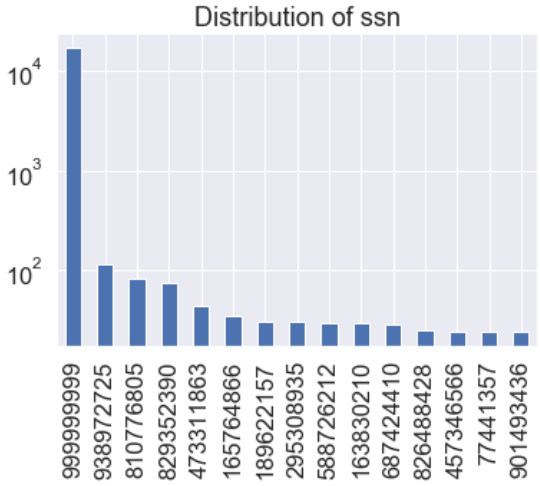
date:

The date of that application occurred on.




ssn:

Social Security Number of the applicant associated with this record.




firstname:

First name of the applicant associated with this record.

| First name | Count (top 10) |
|---|---|
| EAMSTRMT | 12658 |
| TXEMXZZM | 10297 |
| UXXJJZTUZ | 10235 |
| UJSRSMUEZ | 9994 |
| SREZUJMJU | 9688 |
| EASEXMJAT | 7576 |
| SSSXUEJMS | 6923 |

| | |
|---|---|
| SZUASTTA | 6878 |
| EREMTZXXA | 6717 |
| EAXRRUMUX | 5686 |

lastname:

Last name of the applicant.

| Last Name | Count (top 10) |
|---|---|
| ERJSAXA | 8580 |
| UMXUUUSE | 7156 |
| UMARRMA | 6832 |
| MEAXJUX | 5492 |
| XMERRR | 5451 |
| SXZXJRJT | 4340 |
| EUSEZRAE | 4173 |
| USMATTUR | 4036 |
| ETERUXME | 3762 |
| RJURSTXJ | 3575 |

address:

The street address of the applicant.

| Address | Count (top 10) |
|---|---|
| 123 MAIN ST | 1079 |
| 1775 XJXE LN | 97 |
| 7433 RAEZA ST | 80 |
| 8911 MZSU DR | 74 |
| 4907 RRAAU DR | 73 |
| 426 XUAXZ BLVD | 57 |
| 3545 ARMA ST | 44 |
| 606 EZZAU WY | 44 |
| 4530 ETSMX WY | 42 |
| 4292 RUSMM LN | 41 |
| 606 EZZAU WY | 44 |

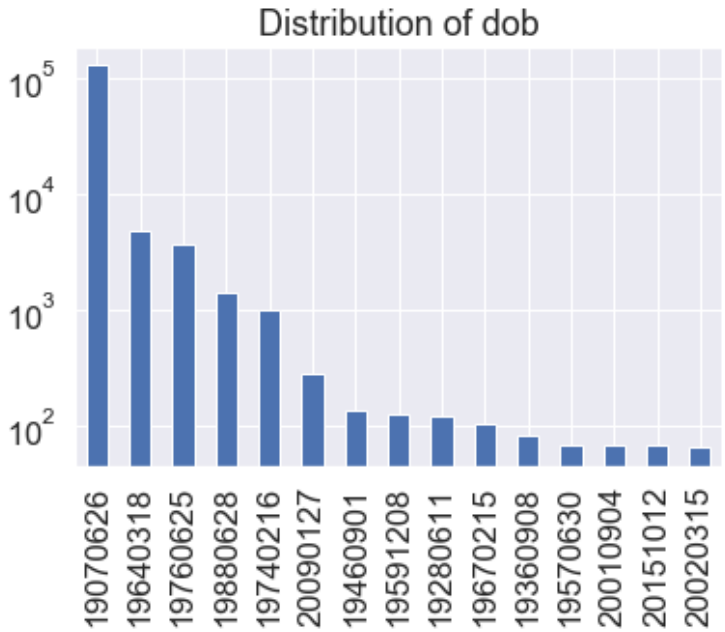
zip5:

Five-digit zip code of the applicant.

| Zip5 | Count (top10) |
|---|---|
| 68138 | 823 |
| 90042 | 514 |
| 89835 | 489 |
| 35227 | 478 |
| 14931 | 459 |
| 86500 | 438 |
| 12700 | 436 |
| 1362 | 434 |
| 59695 | 432 |
| 52317 | 432 |

dob:

Date of birth of the applicant associated with each record.




homephone:

The home phone number of the applicant associated with each record.

| Homephone | Count (top 10) |
|---|---|
| 9999999999 | 78512 |
| 6384782007 | 466 |
| 6035129044 | 360 |
| 2113738531 | 331 |
| 4024680535 | 198 |
| 2669445638 | 172 |
| 6637507363 | 169 |
| 8629049955 | 139 |
| 3364980740 | 110 |
| 1324008228 | 108 |

fraud_label:

Fraud label equal to 1 indicates that record is fraudulent.

| Fraud label | Count |
|---|---|
| **1** | 14393 |
| **0** | 985607 |

## Summary Table for all Candidate Features:

| Features | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| fraud_label | 1000000 | 0.01439 | 0.1191 | 0 | 0 | 0 | 0 | 1 |
| dow_risk | 1000000 | 0.01441 | 0.00062 | 0.01348 | 0.01367 | 0.0145 | 0.01498 | 0.01517 |
| ssn_day_since | 1000000 | 163.702 | 105.164 | 0 | 72 | 155 | 251 | 365 |
| ssn_count_0 | 1000000 | 1.00731 | 0.22336 | 1 | 1 | 1 | 1 | 21 |
| ssn_count_1 | 1000000 | 1.01492 | 0.38121 | 1 | 1 | 1 | 1 | 34 |
| ssn_count_3 | 1000000 | 1.02014 | 0.42319 | 1 | 1 | 1 | 1 | 34 |
| ssn_count_7 | 1000000 | 1.02635 | 0.45356 | 1 | 1 | 1 | 1 | 34 |
| ssn_count_14 | 1000000 | 1.03459 | 0.47765 | 1 | 1 | 1 | 1 | 34 |
| ssn_count_30 | 1000000 | 1.05083 | 0.51333 | 1 | 1 | 1 | 1 | 34 |
| address_day_since | 1000000 | 160.231 | 105.49 | 0 | 67 | 150 | 248 | 365 |
| address_count_0 | 1000000 | 1.01194 | 0.28329 | 1 | 1 | 1 | 1 | 24 |
| address_count_1 | 1000000 | 1.02467 | 0.48295 | 1 | 1 | 1 | 1 | 30 |
| address_count_3 | 1000000 | 1.0331 | 0.5443 | 1 | 1 | 1 | 1 | 30 |
| address_count_7 | 1000000 | 1.04255 | 0.58575 | 1 | 1 | 1 | 1 | 30 |
| address_count_14 | 1000000 | 1.05471 | 0.61795 | 1 | 1 | 1 | 1 | 30 |
| address_count_30 | 1000000 | 1.07817 | 0.66412 | 1 | 1 | 1 | 1 | 30 |
| dob_day_since | 1000000 | 37.9394 | 68.6359 | 0 | 5 | 14 | 32 | 365 |
| dob_count_0 | 1000000 | 1.08569 | 0.73068 | 1 | 1 | 1 | 1 | 23 |
| dob_count_1 | 1000000 | 1.2496 | 1.78946 | 1 | 1 | 1 | 1 | 40 |
| dob_count_3 | 1000000 | 1.56808 | 3.96874 | 1 | 1 | 1 | 1 | 72 |
| dob_count_7 | 1000000 | 2.19215 | 8.33149 | 1 | 1 | 1 | 2 | 129 |
| dob_count_14 | 1000000 | 3.26616 | 15.9025 | 1 | 1 | 1 | 2 | 236 |
| dob_count_30 | 1000000 | 5.63336 | 32.7863 | 1 | 1 | 2 | 3 | 446 |
| homephone_day_since | 1000000 | 23.7079 | 57.8814 | 0 | 2 | 6 | 14 | 365 |
| homephone_count_0 | 1000000 | 1.07109 | 0.36483 | 1 | 1 | 1 | 1 | 22 |
| homephone_count_1 | 1000000 | 1.20165 | 0.62657 | 1 | 1 | 1 | 1 | 31 |
| homephone_count_3 | 1000000 | 1.4493 | 0.86162 | 1 | 1 | 1 | 2 | 32 |
| homephone_count_7 | 1000000 | 1.93312 | 1.21308 | 1 | 1 | 2 | 2 | 33 |
| homephone_count_14 | 1000000 | 2.76381 | 1.74409 | 1 | 1 | 2 | 4 | 36 |
| homephone_count_30 | 1000000 | 4.59752 | 2.86356 | 1 | 2 | 4 | 6 | 50 |
| name_day_since | 1000000 | 142.51 | 105.784 | 0 | 48 | 124 | 227 | 365 |
| name_count_0 | 1000000 | 1.0096 | 0.22879 | 1 | 1 | 1 | 1 | 21 |
| name_count_1 | 1000000 | 1.02164 | 0.39139 | 1 | 1 | 1 | 1 | 34 |
| name_count_3 | 1000000 | 1.03547 | 0.44615 | 1 | 1 | 1 | 1 | 34 |
| name_count_7 | 1000000 | 1.0589 | 0.50934 | 1 | 1 | 1 | 1 | 34 |
| name_count_14 | 1000000 | 1.09624 | 0.60383 | 1 | 1 | 1 | 1 | 34 |
| name_count_30 | 1000000 | 1.17792 | 0.8424 | 1 | 1 | 1 | 1 | 34 |
| fulladdress_day_since | 1000000 | 162.705 | 105.247 | 0 | 70 | 153 | 250 | 365 |
| fulladdress_count_0 | 1000000 | 1.0117 | 0.28285 | 1 | 1 | 1 | 1 | 24 |
| fulladdress_count_1 | 1000000 | 1.02395 | 0.48217 | 1 | 1 | 1 | 1 | 30 |
| fulladdress_count_3 | 1000000 | 1.03143 | 0.54244 | 1 | 1 | 1 | 1 | 30 |
| fulladdress_count_7 | 1000000 | 1.039 | 0.58123 | 1 | 1 | 1 | 1 | 30 |
| fulladdress_count_14 | 1000000 | 1.04797 | 0.60686 | 1 | 1 | 1 | 1 | 30 |
| fulladdress_count_30 | 1000000 | 1.06462 | 0.63383 | 1 | 1 | 1 | 1 | 30 |
| name_dob_day_since | 1000000 | 165.711 | 105.379 | 0 | 73 | 158 | 254 | 365 |
| name_dob_count_0 | 1000000 | 1.00723 | 0.22316 | 1 | 1 | 1 | 1 | 21 |
| name_dob_count_1 | 1000000 | 1.01469 | 0.38087 | 1 | 1 | 1 | 1 | 34 |
| name_dob_count_3 | 1000000 | 1.0196 | 0.42229 | 1 | 1 | 1 | 1 | 34 |
| name_dob_count_7 | 1000000 | 1.02518 | 0.45126 | 1 | 1 | 1 | 1 | 34 |

| Features | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| name_dob_count_14 | 1000000 | 1.03236 | 0.47176 | 1 | 1 | 1 | 1 | 34 |
| name_dob_count_30 | 1000000 | 1.0463 | 0.49691 | 1 | 1 | 1 | 1 | 34 |
| name_fulladdress_day_since | 1000000 | 164.6 | 104.967 | 0 | 73 | 156 | 252 | 365 |
| name_fulladdress_count_0 | 1000000 | 1.00051 | 0.0228 | 1 | 1 | 1 | 1 | 3 |
| name_fulladdress_count_1 | 1000000 | 1.00148 | 0.03855 | 1 | 1 | 1 | 1 | 3 |
| name_fulladdress_count_3 | 1000000 | 1.0035 | 0.05972 | 1 | 1 | 1 | 1 | 5 |
| name_fulladdress_count_7 | 1000000 | 1.00734 | 0.08768 | 1 | 1 | 1 | 1 | 6 |
| name_fulladdress_count_14 | 1000000 | 1.01401 | 0.12273 | 1 | 1 | 1 | 1 | 8 |
| name_fulladdress_count_30 | 1000000 | 1.02866 | 0.1825 | 1 | 1 | 1 | 1 | 11 |
| name_homephone_day_since | 1000000 | 165.51 | 105.095 | 0 | 73 | 157 | 253 | 365 |
| name_homephone_count_0 | 1000000 | 1.0005 | 0.0224 | 1 | 1 | 1 | 1 | 3 |
| name_homephone_count_1 | 1000000 | 1.00141 | 0.03768 | 1 | 1 | 1 | 1 | 3 |
| name_homephone_count_3 | 1000000 | 1.00334 | 0.05835 | 1 | 1 | 1 | 1 | 5 |
| name_homephone_count_7 | 1000000 | 1.00701 | 0.08575 | 1 | 1 | 1 | 1 | 6 |
| name_homephone_count_14 | 1000000 | 1.01337 | 0.12002 | 1 | 1 | 1 | 1 | 8 |
| name_homephone_count_30 | 1000000 | 1.02727 | 0.17844 | 1 | 1 | 1 | 1 | 11 |
| fulladdress_dob_day_since | 1000000 | 166.401 | 105.195 | 0 | 74 | 158 | 255 | 365 |
| fulladdress_dob_count_0 | 1000000 | 1.00047 | 0.02181 | 1 | 1 | 1 | 1 | 3 |
| fulladdress_dob_count_1 | 1000000 | 1.00134 | 0.03676 | 1 | 1 | 1 | 1 | 3 |
| fulladdress_dob_count_3 | 1000000 | 1.00318 | 0.05703 | 1 | 1 | 1 | 1 | 5 |
| fulladdress_dob_count_7 | 1000000 | 1.00667 | 0.08382 | 1 | 1 | 1 | 1 | 6 |
| fulladdress_dob_count_14 | 1000000 | 1.01274 | 0.11747 | 1 | 1 | 1 | 1 | 8 |
| fulladdress_dob_count_30 | 1000000 | 1.026 | 0.17505 | 1 | 1 | 1 | 1 | 11 |
| fulladdress_homephone_day_since | 1000000 | 165.047 | 105.334 | 0 | 73 | 157 | 253 | 365 |
| fulladdress_homephone_count_0 | 1000000 | 1.00742 | 0.2237 | 1 | 1 | 1 | 1 | 21 |
| fulladdress_homephone_count_1 | 1000000 | 1.0152 | 0.37831 | 1 | 1 | 1 | 1 | 30 |
| fulladdress_homephone_count_3 | 1000000 | 1.02045 | 0.42573 | 1 | 1 | 1 | 1 | 30 |
| fulladdress_homephone_count_7 | 1000000 | 1.02629 | 0.45906 | 1 | 1 | 1 | 1 | 30 |
| fulladdress_homephone_count_14 | 1000000 | 1.03378 | 0.48199 | 1 | 1 | 1 | 1 | 30 |
| fulladdress_homephone_count_30 | 1000000 | 1.04821 | 0.50768 | 1 | 1 | 1 | 1 | 30 |
| dob_homephone_day_since | 1000000 | 166.581 | 105.239 | 0 | 74 | 159 | 255 | 365 |
| dob_homephone_count_0 | 1000000 | 1.00047 | 0.02184 | 1 | 1 | 1 | 1 | 3 |
| dob_homephone_count_1 | 1000000 | 1.00134 | 0.0367 | 1 | 1 | 1 | 1 | 3 |
| dob_homephone_count_3 | 1000000 | 1.00317 | 0.05695 | 1 | 1 | 1 | 1 | 5 |
| dob_homephone_count_7 | 1000000 | 1.00662 | 0.08346 | 1 | 1 | 1 | 1 | 6 |
| dob_homephone_count_14 | 1000000 | 1.01263 | 0.1169 | 1 | 1 | 1 | 1 | 8 |
| dob_homephone_count_30 | 1000000 | 1.02576 | 0.17412 | 1 | 1 | 1 | 1 | 11 |
| homephone_name_dob_day_since | 1000000 | 166.808 | 105.25 | 0 | 75 | 159 | 255 | 365 |
| homephone_name_dob_count_0 | 1000000 | 1.00046 | 0.02161 | 1 | 1 | 1 | 1 | 3 |
| homephone_name_dob_count_1 | 1000000 | 1.00131 | 0.03628 | 1 | 1 | 1 | 1 | 3 |
| homephone_name_dob_count_3 | 1000000 | 1.00311 | 0.0564 | 1 | 1 | 1 | 1 | 5 |
| homephone_name_dob_count_7 | 1000000 | 1.00651 | 0.08281 | 1 | 1 | 1 | 1 | 6 |
| homephone_name_dob_count_14 | 1000000 | 1.01243 | 0.11605 | 1 | 1 | 1 | 1 | 8 |
| homephone_name_dob_count_30 | 1000000 | 1.02535 | 0.17287 | 1 | 1 | 1 | 1 | 11 |
| ssn_firstname_day_since | 1000000 | 163.91 | 105.158 | 0 | 72 | 155 | 252 | 365 |
| ssn_firstname_count_0 | 1000000 | 1.00727 | 0.22325 | 1 | 1 | 1 | 1 | 21 |
| ssn_firstname_count_1 | 1000000 | 1.01482 | 0.38103 | 1 | 1 | 1 | 1 | 34 |
| ssn_firstname_count_3 | 1000000 | 1.01989 | 0.42263 | 1 | 1 | 1 | 1 | 34 |
| ssn_firstname_count_7 | 1000000 | 1.02584 | 0.45192 | 1 | 1 | 1 | 1 | 34 |

| Features | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| ssn_firstname_count_14 | 1000000 | 1.03362 | 0.47293 | 1 | 1 | 1 | 1 | 34 |
| ssn_firstname_count_30 | 1000000 | 1.0489 | 0.49903 | 1 | 1 | 1 | 1 | 34 |
| ssn_lastname_day_since | 1000000 | 163.916 | 105.161 | 0 | 72 | 155 | 252 | 365 |
| ssn_lastname_count_0 | 1000000 | 1.00726 | 0.22325 | 1 | 1 | 1 | 1 | 21 |
| ssn_lastname_count_1 | 1000000 | 1.01482 | 0.38103 | 1 | 1 | 1 | 1 | 34 |
| ssn_lastname_count_3 | 1000000 | 1.01989 | 0.42263 | 1 | 1 | 1 | 1 | 34 |
| ssn_lastname_count_7 | 1000000 | 1.02583 | 0.4519 | 1 | 1 | 1 | 1 | 34 |
| ssn_lastname_count_14 | 1000000 | 1.0336 | 0.47291 | 1 | 1 | 1 | 1 | 34 |
| ssn_lastname_count_30 | 1000000 | 1.0489 | 0.49902 | 1 | 1 | 1 | 1 | 34 |
| ssn_address_day_since | 1000000 | 164.751 | 104.997 | 0 | 73 | 156 | 252 | 365 |
| ssn_address_count_0 | 1000000 | 1.00051 | 0.02267 | 1 | 1 | 1 | 1 | 3 |
| ssn_address_count_1 | 1000000 | 1.00147 | 0.03843 | 1 | 1 | 1 | 1 | 3 |
| ssn_address_count_3 | 1000000 | 1.00346 | 0.05948 | 1 | 1 | 1 | 1 | 5 |
| ssn_address_count_7 | 1000000 | 1.00729 | 0.08737 | 1 | 1 | 1 | 1 | 6 |
| ssn_address_count_14 | 1000000 | 1.01389 | 0.12218 | 1 | 1 | 1 | 1 | 8 |
| ssn_address_count_30 | 1000000 | 1.02841 | 0.18152 | 1 | 1 | 1 | 1 | 11 |
| ssn_zip5_day_since | 1000000 | 164.701 | 104.992 | 0 | 73 | 156 | 252 | 365 |
| ssn_zip5_count_0 | 1000000 | 1.00051 | 0.02269 | 1 | 1 | 1 | 1 | 3 |
| ssn_zip5_count_1 | 1000000 | 1.00147 | 0.03842 | 1 | 1 | 1 | 1 | 3 |
| ssn_zip5_count_3 | 1000000 | 1.00347 | 0.05952 | 1 | 1 | 1 | 1 | 5 |
| ssn_zip5_count_7 | 1000000 | 1.00731 | 0.0875 | 1 | 1 | 1 | 1 | 6 |
| ssn_zip5_count_14 | 1000000 | 1.01393 | 0.12233 | 1 | 1 | 1 | 1 | 8 |
| ssn_zip5_count_30 | 1000000 | 1.02849 | 0.18173 | 1 | 1 | 1 | 1 | 11 |
| ssn_dob_day_since | 1000000 | 165.797 | 105.4 | 0 | 73 | 158 | 254 | 365 |
| ssn_dob_count_0 | 1000000 | 1.00723 | 0.22316 | 1 | 1 | 1 | 1 | 21 |
| ssn_dob_count_1 | 1000000 | 1.01469 | 0.38085 | 1 | 1 | 1 | 1 | 34 |
| ssn_dob_count_3 | 1000000 | 1.01958 | 0.42226 | 1 | 1 | 1 | 1 | 34 |
| ssn_dob_count_7 | 1000000 | 1.02513 | 0.45116 | 1 | 1 | 1 | 1 | 34 |
| ssn_dob_count_14 | 1000000 | 1.03226 | 0.47155 | 1 | 1 | 1 | 1 | 34 |
| ssn_dob_count_30 | 1000000 | 1.0461 | 0.49631 | 1 | 1 | 1 | 1 | 34 |
| ssn_homephone_day_since | 1000000 | 165.603 | 105.119 | 0 | 74 | 157 | 254 | 365 |
| ssn_homephone_count_0 | 1000000 | 1.00049 | 0.02222 | 1 | 1 | 1 | 1 | 3 |
| ssn_homephone_count_1 | 1000000 | 1.0014 | 0.03756 | 1 | 1 | 1 | 1 | 3 |
| ssn_homephone_count_3 | 1000000 | 1.00332 | 0.05824 | 1 | 1 | 1 | 1 | 5 |
| ssn_homephone_count_7 | 1000000 | 1.00699 | 0.08562 | 1 | 1 | 1 | 1 | 6 |
| ssn_homephone_count_14 | 1000000 | 1.01331 | 0.11979 | 1 | 1 | 1 | 1 | 8 |
| ssn_homephone_count_30 | 1000000 | 1.02718 | 0.17816 | 1 | 1 | 1 | 1 | 11 |
| ssn_name_day_since | 1000000 | 163.971 | 105.161 | 0 | 72 | 155 | 252 | 365 |
| ssn_name_count_0 | 1000000 | 1.00726 | 0.22324 | 1 | 1 | 1 | 1 | 21 |
| ssn_name_count_1 | 1000000 | 1.01482 | 0.38102 | 1 | 1 | 1 | 1 | 34 |
| ssn_name_count_3 | 1000000 | 1.01988 | 0.42261 | 1 | 1 | 1 | 1 | 34 |
| ssn_name_count_7 | 1000000 | 1.02581 | 0.45187 | 1 | 1 | 1 | 1 | 34 |
| ssn_name_count_14 | 1000000 | 1.03356 | 0.47286 | 1 | 1 | 1 | 1 | 34 |
| ssn_name_count_30 | 1000000 | 1.04878 | 0.49889 | 1 | 1 | 1 | 1 | 34 |
| ssn_fulladdress_day_since | 1000000 | 164.797 | 104.998 | 0 | 73 | 156 | 252 | 365 |
| ssn_fulladdress_count_0 | 1000000 | 1.00051 | 0.0226 | 1 | 1 | 1 | 1 | 3 |
| ssn_fulladdress_count_1 | 1000000 | 1.00146 | 0.03835 | 1 | 1 | 1 | 1 | 3 |
| ssn_fulladdress_count_3 | 1000000 | 1.00346 | 0.05941 | 1 | 1 | 1 | 1 | 5 |
| ssn_fulladdress_count_7 | 1000000 | 1.00727 | 0.08727 | 1 | 1 | 1 | 1 | 6 |

| Features | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| ssn_fulladdress_count_14 | 1000000 | 1.01386 | 0.12201 | 1 | 1 | 1 | 1 | 8 |
| ssn_fulladdress_count_30 | 1000000 | 1.02832 | 0.18127 | 1 | 1 | 1 | 1 | 11 |
| ssn_name_dob_day_since | 1000000 | 165.87 | 105.4 | 0 | 74 | 158 | 254 | 365 |
| ssn_name_dob_count_0 | 1000000 | 1.00722 | 0.22315 | 1 | 1 | 1 | 1 | 21 |
| ssn_name_dob_count_1 | 1000000 | 1.01468 | 0.38084 | 1 | 1 | 1 | 1 | 34 |
| ssn_name_dob_count_3 | 1000000 | 1.01956 | 0.42224 | 1 | 1 | 1 | 1 | 34 |
| ssn_name_dob_count_7 | 1000000 | 1.0251 | 0.45112 | 1 | 1 | 1 | 1 | 34 |
| ssn_name_dob_count_14 | 1000000 | 1.03219 | 0.47148 | 1 | 1 | 1 | 1 | 34 |
| ssn_name_dob_count_30 | 1000000 | 1.04594 | 0.49615 | 1 | 1 | 1 | 1 | 34 |
| ssn_count_1_by_3 | 1000000 | 2.99526 | 0.0837 | 0.42857 | 3 | 3 | 3 | 3 |
| ssn_count_1_by_7 | 1000000 | 6.97342 | 0.3097 | 0.5 | 7 | 7 | 7 | 7 |
| ssn_count_1_by_14 | 1000000 | 13.8989 | 0.85324 | 0.875 | 14 | 14 | 14 | 14 |
| ssn_count_1_by_30 | 1000000 | 29.5693 | 2.5569 | 1.76471 | 30 | 30 | 30 | 30 |
| address_count_1_by_3 | 1000000 | 2.99288 | 0.10221 | 0.42857 | 3 | 3 | 3 | 3 |
| address_count_1_by_7 | 1000000 | 6.96122 | 0.37494 | 0.58333 | 7 | 7 | 7 | 7 |
| address_count_1_by_14 | 1000000 | 13.8564 | 1.02268 | 0.93333 | 14 | 14 | 14 | 14 |
| address_count_1_by_30 | 1000000 | 29.4135 | 3.01934 | 1.30435 | 30 | 30 | 30 | 30 |
| dob_count_1_by_3 | 1000000 | 2.84428 | 0.46307 | 0.08108 | 3 | 3 | 3 | 3 |
| dob_count_1_by_7 | 1000000 | 6.0604 | 1.64379 | 0.09211 | 4.66667 | 7 | 7 | 7 |
| dob_count_1_by_14 | 1000000 | 10.5213 | 4.01433 | 0.08589 | 7 | 14 | 14 | 14 |
| dob_count_1_by_30 | 1000000 | 17.4221 | 9.29114 | 0.07634 | 10 | 15 | 30 | 30 |
| homephone_count_1_by_3 | 1000000 | 2.6916 | 0.61206 | 0.12 | 3 | 3 | 3 | 3 |
| homephone_count_1_by_7 | 1000000 | 5.19552 | 1.96711 | 0.21212 | 3.5 | 7 | 7 | 7 |
| homephone_count_1_by_14 | 1000000 | 7.98108 | 4.21822 | 0.41176 | 4.66667 | 7 | 14 | 14 |
| homephone_count_1_by_30 | 1000000 | 11.6649 | 8.76278 | 0.65217 | 6 | 7.5 | 15 | 30 |
| name_count_1_by_3 | 1000000 | 2.98352 | 0.15661 | 0.42857 | 3 | 3 | 3 | 3 |
| name_count_1_by_7 | 1000000 | 6.89834 | 0.60482 | 0.46667 | 7 | 7 | 7 | 7 |
| name_count_1_by_14 | 1000000 | 13.614 | 1.68271 | 0.82353 | 14 | 14 | 14 | 14 |
| name_count_1_by_30 | 1000000 | 28.4637 | 4.94373 | 1.5 | 30 | 30 | 30 | 30 |
| fulladdress_count_1_by_3 | 1000000 | 2.9942 | 0.09196 | 0.42857 | 3 | 3 | 3 | 3 |
| fulladdress_count_1_by_7 | 1000000 | 6.96971 | 0.33126 | 0.58333 | 7 | 7 | 7 | 7 |
| fulladdress_count_1_by_14 | 1000000 | 13.8892 | 0.89421 | 0.93333 | 14 | 14 | 14 | 14 |
| fulladdress_count_1_by_30 | 1000000 | 29.5398 | 2.64186 | 1.30435 | 30 | 30 | 30 | 30 |
| name_dob_count_1_by_3 | 1000000 | 2.99566 | 0.07995 | 0.42857 | 3 | 3 | 3 | 3 |
| name_dob_count_1_by_7 | 1000000 | 6.97616 | 0.29329 | 0.5 | 7 | 7 | 7 | 7 |
| name_dob_count_1_by_14 | 1000000 | 13.9097 | 0.80554 | 0.875 | 14 | 14 | 14 | 14 |
| name_dob_count_1_by_30 | 1000000 | 29.6169 | 2.41101 | 1.76471 | 30 | 30 | 30 | 30 |
| name_fulladdress_count_1_by_3 | 1000000 | 2.997 | 0.06707 | 0.75 | 3 | 3 | 3 | 3 |
| name_fulladdress_count_1_by_7 | 1000000 | 6.97978 | 0.26593 | 1.16667 | 7 | 7 | 7 | 7 |
| name_fulladdress_count_1_by_14 | 1000000 | 13.9143 | 0.774 | 1.75 | 14 | 14 | 14 | 14 |
| name_fulladdress_count_1_by_30 | 1000000 | 29.6087 | 2.41479 | 2.72727 | 30 | 30 | 30 | 30 |
| name_homephone_count_1_by_3 | 1000000 | 2.99714 | 0.06551 | 0.75 | 3 | 3 | 3 | 3 |
| name_homephone_count_1_by_7 | 1000000 | 6.98069 | 0.25994 | 1.16667 | 7 | 7 | 7 | 7 |
| name_homephone_count_1_by_14 | 1000000 | 13.9183 | 0.75606 | 1.75 | 14 | 14 | 14 | 14 |
| name_homephone_count_1_by_30 | 1000000 | 29.628 | 2.35649 | 2.72727 | 30 | 30 | 30 | 30 |
| fulladdress_dob_count_1_by_3 | 1000000 | 2.99726 | 0.06404 | 0.75 | 3 | 3 | 3 | 3 |
| fulladdress_dob_count_1_by_7 | 1000000 | 6.98165 | 0.25356 | 1.16667 | 7 | 7 | 7 | 7 |
| fulladdress_dob_count_1_by_14 | 1000000 | 13.9222 | 0.73814 | 1.75 | 14 | 14 | 14 | 14 |
| fulladdress_dob_count_1_by_30 | 1000000 | 29.6459 | 2.30125 | 2.72727 | 30 | 30 | 30 | 30 |

| Features | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| fulladdress_homephone_count_1_by_3 | 1000000 | 2.99553 | 0.08122 | 0.5 | 3 | 3 | 3 | 3 |
| fulladdress_homephone_count_1_by_7 | 1000000 | 6.97539 | 0.29781 | 0.58333 | 7 | 7 | 7 | 7 |
| fulladdress_homephone_count_1_by_14 | 1000000 | 13.9066 | 0.81839 | 0.93333 | 14 | 14 | 14 | 14 |
| fulladdress_homephone_count_1_by_30 | 1000000 | 29.6029 | 2.45145 | 1.30435 | 30 | 30 | 30 | 30 |
| dob_homephone_count_1_by_3 | 1000000 | 2.99727 | 0.06395 | 0.75 | 3 | 3 | 3 | 3 |
| dob_homephone_count_1_by_7 | 1000000 | 6.98181 | 0.25237 | 1.16667 | 7 | 7 | 7 | 7 |
| dob_homephone_count_1_by_14 | 1000000 | 13.9229 | 0.7347 | 1.75 | 14 | 14 | 14 | 14 |
| dob_homephone_count_1_by_30 | 1000000 | 29.6491 | 2.29072 | 2.72727 | 30 | 30 | 30 | 30 |
| homephone_name_dob_count_1_by_3 | 1000000 | 2.99731 | 0.06345 | 0.75 | 3 | 3 | 3 | 3 |
| homephone_name_dob_count_1_by_7 | 1000000 | 6.98207 | 0.25058 | 1.16667 | 7 | 7 | 7 | 7 |
| homephone_name_dob_count_1_by_14 | 1000000 | 13.924 | 0.72942 | 1.75 | 14 | 14 | 14 | 14 |
| homephone_name_dob_count_1_by_30 | 1000000 | 29.6547 | 2.27303 | 2.72727 | 30 | 30 | 30 | 30 |
| ssn_firstname_count_1_by_3 | 1000000 | 2.99542 | 0.08221 | 0.42857 | 3 | 3 | 3 | 3 |
| ssn_firstname_count_1_by_7 | 1000000 | 6.97427 | 0.30413 | 0.5 | 7 | 7 | 7 | 7 |
| ssn_firstname_count_1_by_14 | 1000000 | 13.9017 | 0.83846 | 0.875 | 14 | 14 | 14 | 14 |
| ssn_firstname_count_1_by_30 | 1000000 | 29.5793 | 2.51913 | 1.76471 | 30 | 30 | 30 | 30 |
| ssn_lastname_count_1_by_3 | 1000000 | 2.99542 | 0.08221 | 0.42857 | 3 | 3 | 3 | 3 |
| ssn_lastname_count_1_by_7 | 1000000 | 6.97431 | 0.30388 | 0.5 | 7 | 7 | 7 | 7 |
| ssn_lastname_count_1_by_14 | 1000000 | 13.9019 | 0.83799 | 0.875 | 14 | 14 | 14 | 14 |
| ssn_lastname_count_1_by_30 | 1000000 | 29.5794 | 2.51893 | 1.76471 | 30 | 30 | 30 | 30 |
| ssn_address_count_1_by_3 | 1000000 | 2.99703 | 0.0667 | 0.75 | 3 | 3 | 3 | 3 |
| ssn_address_count_1_by_7 | 1000000 | 6.97994 | 0.26489 | 1.16667 | 7 | 7 | 7 | 7 |
| ssn_address_count_1_by_14 | 1000000 | 13.915 | 0.77067 | 1.75 | 14 | 14 | 14 | 14 |
| ssn_address_count_1_by_30 | 1000000 | 29.6118 | 2.40485 | 2.72727 | 30 | 30 | 30 | 30 |
| ssn_zip5_count_1_by_3 | 1000000 | 2.99702 | 0.0668 | 0.75 | 3 | 3 | 3 | 3 |
| ssn_zip5_count_1_by_7 | 1000000 | 6.97986 | 0.26544 | 1.16667 | 7 | 7 | 7 | 7 |
| ssn_zip5_count_1_by_14 | 1000000 | 13.9147 | 0.77199 | 1.75 | 14 | 14 | 14 | 14 |
| ssn_zip5_count_1_by_30 | 1000000 | 29.6107 | 2.40821 | 2.72727 | 30 | 30 | 30 | 30 |
| ssn_dob_count_1_by_3 | 1000000 | 2.99568 | 0.07975 | 0.42857 | 3 | 3 | 3 | 3 |
| ssn_dob_count_1_by_7 | 1000000 | 6.97626 | 0.29259 | 0.5 | 7 | 7 | 7 | 7 |
| ssn_dob_count_1_by_14 | 1000000 | 13.9102 | 0.80309 | 0.875 | 14 | 14 | 14 | 14 |
| ssn_dob_count_1_by_30 | 1000000 | 29.6186 | 2.40453 | 1.76471 | 30 | 30 | 30 | 30 |
| ssn_homephone_count_1_by_3 | 1000000 | 2.99715 | 0.06541 | 0.75 | 3 | 3 | 3 | 3 |
| ssn_homephone_count_1_by_7 | 1000000 | 6.98075 | 0.25957 | 1.16667 | 7 | 7 | 7 | 7 |
| ssn_homephone_count_1_by_14 | 1000000 | 13.9186 | 0.75451 | 1.75 | 14 | 14 | 14 | 14 |
| ssn_homephone_count_1_by_30 | 1000000 | 29.6292 | 2.35276 | 2.72727 | 30 | 30 | 30 | 30 |
| ssn_name_count_1_by_3 | 1000000 | 2.99543 | 0.0821 | 0.42857 | 3 | 3 | 3 | 3 |
| ssn_name_count_1_by_7 | 1000000 | 6.97437 | 0.30355 | 0.5 | 7 | 7 | 7 | 7 |
| ssn_name_count_1_by_14 | 1000000 | 13.9021 | 0.83685 | 0.875 | 14 | 14 | 14 | 14 |
| ssn_name_count_1_by_30 | 1000000 | 29.581 | 2.51429 | 1.76471 | 30 | 30 | 30 | 30 |
| ssn_fulladdress_count_1_by_3 | 1000000 | 2.99704 | 0.06664 | 0.75 | 3 | 3 | 3 | 3 |
| ssn_fulladdress_count_1_by_7 | 1000000 | 6.97999 | 0.2646 | 1.16667 | 7 | 7 | 7 | 7 |
| ssn_fulladdress_count_1_by_14 | 1000000 | 13.9152 | 0.76973 | 1.75 | 14 | 14 | 14 | 14 |
| ssn_fulladdress_count_1_by_30 | 1000000 | 29.613 | 2.40126 | 2.72727 | 30 | 30 | 30 | 30 |
| ssn_name_dob_count_1_by_3 | 1000000 | 2.9957 | 0.07964 | 0.42857 | 3 | 3 | 3 | 3 |
| ssn_name_dob_count_1_by_7 | 1000000 | 6.97635 | 0.29205 | 0.5 | 7 | 7 | 7 | 7 |
| ssn_name_dob_count_1_by_14 | 1000000 | 13.9106 | 0.80138 | 0.875 | 14 | 14 | 14 | 14 |
| ssn_name_dob_count_1_by_30 | 1000000 | 29.6208 | 2.39807 | 1.76471 | 30 | 30 | 30 | 30 |