USC Marshall
School of Business

DSO 562 Fraud Analytics Project 2
Application (Identity) Fraud

Team 3
Session: 16237
Boyang Han, Leon Man, Ziqing (Juno) Wen, Dakota Wu, Chutong Yan, Yangzi Zhang,
March 15, 2020

Table of Content

Executive Summary

This project is commissioned to develop a supervised fraud detection model based on historical product application data from the "application data.csv" dataset. The dataset stores one million application records dated between 2016/1/1 and 2016/12/31, and 10 fields including the date of the application, the name, date of birth, Social Security Number, and contact information of the applicant, as well as a label specifying if the application is a fraud.

Two types of fraud identifications exist in this dataset: 1) an individual fraudster using many victims' core identity information and his/her own contact information; 2) a victim's core identity information being used by many fraudsters. The model learned to predict fraud by capturing common characteristics of identity fraud such as the frequent occurrence of the same Personally Identifiable Information (PII), or the frequent change of the combination of PII. Fraud Detection Rate (FDR) was used as the evaluation metric for hyperparameter tuning and model selection.

To get a better idea of how well the model predicts on new, previously unseen data, the whole dataset was divided into three parts, training, testing, and out-of-time (OOT) validation:

- **Training data:** randomly picked 80% of the first ten months of data
- **Testing data:** randomly picked 20% of the first ten months of data
- **OOT data:** the last two months of data

Only the training data was used in the model fitting process. The performance on the testing data provides an estimate of how well the model predicts on new data drawn from the same time period where the train data was drawn from, while the performance on the OOT data provides an estimate of model performance on new data drawn from a future time period occurred sometime after the one where the training data was drawn from.

The project was carried out following the steps below:

- **Data Exploration:** understand the characteristics of the data and recognize the necessary data transformation and manipulation to be performed
- **Data Cleaning:** fix frivolous values to avoid wrongful linkage between applications
- **Create Candidate Variables:** build 407 variables that quantify typical fraud behaviors
- **Scale All Variables**: bring all variables to the same scale to be comparable
- **Feature Selection:** use filter and wrapper methods to pick the top 25 variables with the best predictive potential to reduce dimensionality and speed up computation

- **Model Fitting:** try different algorithms such as Logistic Regression, Gradient Boosting Tree, Random Forest, Neural Network, Single Tree, K Nearest Neighbors, and Support Vector Machine with different parameter combinations and compare model performance
- **Model Selection:** select the best model based on the performance on the OOT data

The final model is a Gradient Boosting Tree classifier (*learning_rate* = 0.01, *n_estimators* = 800, *max_depth* = 5, *max_features* = 5, *min_samples_leaf* = 30, *min_samples_split* = 1500, *subsample* = 0.7) that achieved an FDR of 57.17% at a 3% rejection threshold on the training data, 55.77% on the testing data, and 54.07% on the OOT data.

1 Data Exploration

The “applications data.csv” is a simulated dataset that stores 1,000,000 computer-generated records mirroring the real-life information submitted for product applications such as opening a credit card, a bank account or a cell phone line. It covers 10 fields including application date, applicant’s name, social security number, address, zip code, date of birth, phone number, and a label specifying if the application is a fraud. The dataset was shared by Professor Stephen Coggeshall in February 2020.

1.1 Summary Table

All 10 fields in this dataset are categorical variables and are fully populated. Key statistics of these fields are summarized in the following table.

Table 1.1: Summary Statistics of Fields

Field Name	Data Type	# Records w/ Vales	% Populated	# Unique Values	Most Common Vale
record	integer	1,000,000	100	1,000,000	n/a**
date	integer*	1,000,000	100	365	20160816
ssn	integer	1,000,000	100	835,819	999999999
firstname	text	1,000,000	100	78,136	EAMSTRMT
lastname	text	1,000,000	100	177,001	ERJSAXA
address	text	1,000,000	100	828,774	123 MAIN ST
zip5	integer	1,000,000	100	26,370	68138
dob	integer*	1,000,000	100	42,673	19070626
homephone	integer	1,000,000	100	28,244	9999999999
fraud_label	integer	1,000,000	100	2	0

* The *date* and *dob* fields should be of the datetime data type but are stored as integers in the dataset.

** All values in the *record* field are unique, thus no one value is more common than others.

1.2 Field Examples

1.2.1 Field "date"

The *date* field logs the date on which the application was submitted. It is a categorical and ordinal field that takes 365 values, ranging from 2016/01/01 to 2016/12/31. There are records on every day of the year 2016 except 2016/02/29. Daily application counts are plotted in Figure 1.2.1.1. August 16, 2016 has the most daily applications of 2,877 counts.

Figure 1.2.1.1: Daily Applications

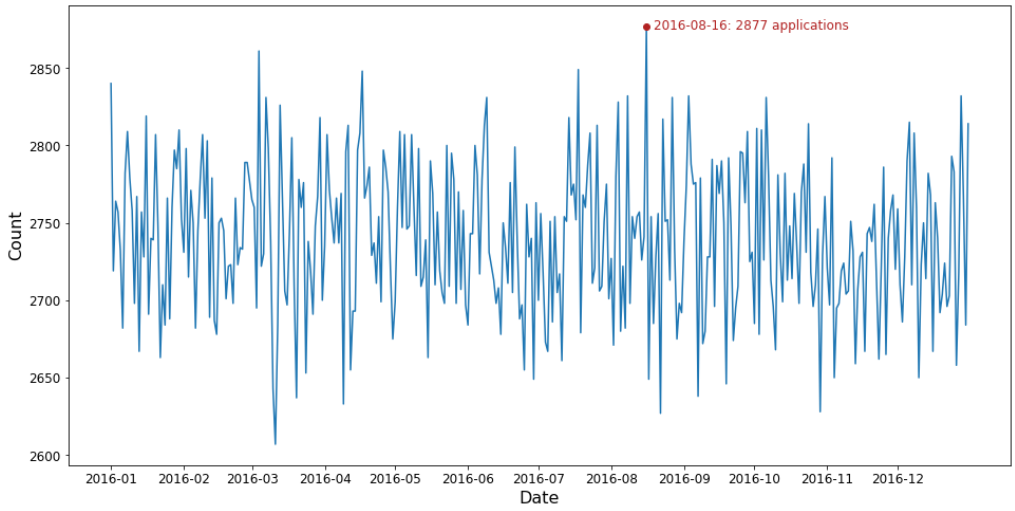
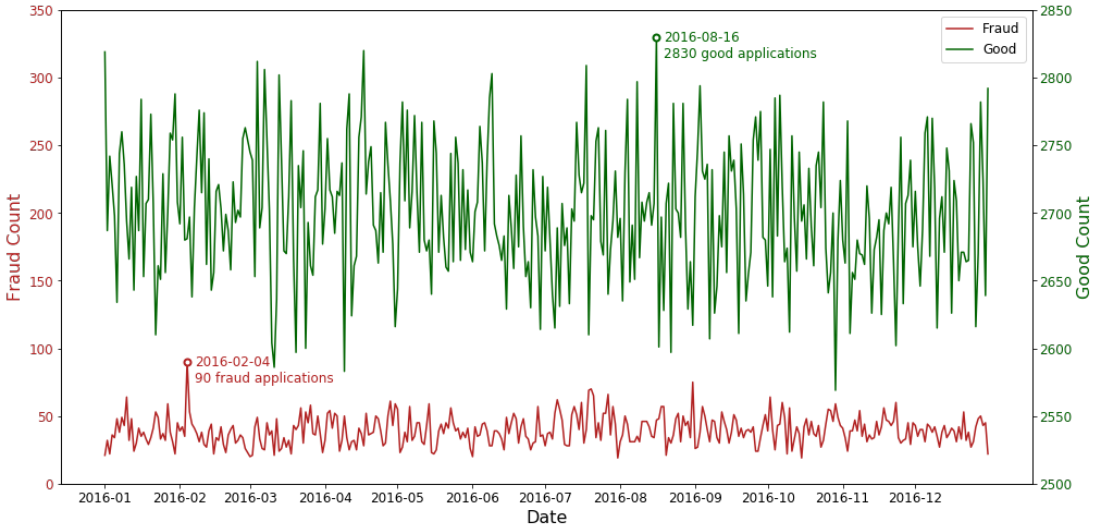


Figure 1.2.1.2 below breaks down the daily applications by their fraud label and plot the counts of the two categories against each other. February 4, 2016 has the most daily fraud applications of 90 counts.

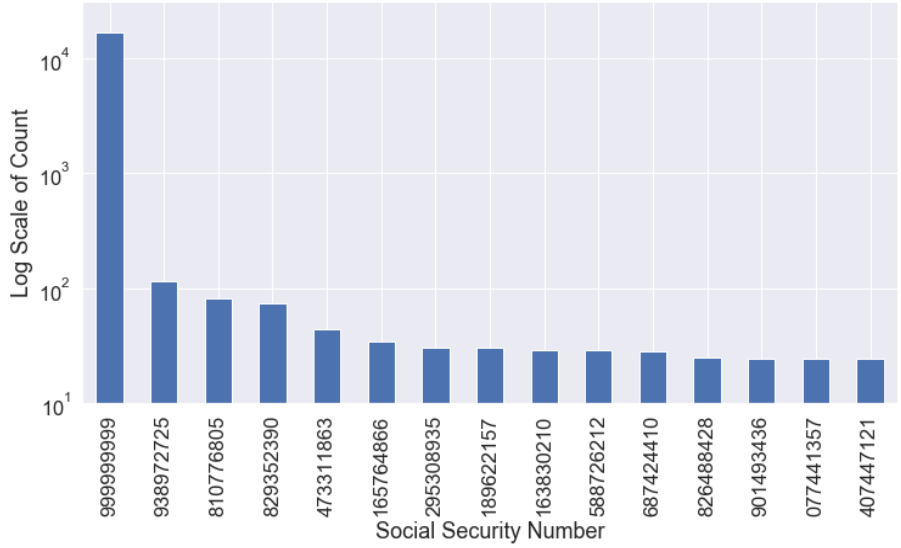
Figure 1.2.1.2: Daily Applications by Fraud and Good Records



1.2.2 Field “ssn”

The *ssn* field logs the Social Security Number (SSN) of the applicant. Value “999999999” occurred the most in this field for 16,935 times (see Figure 1.2.2) and is a frivolous value.

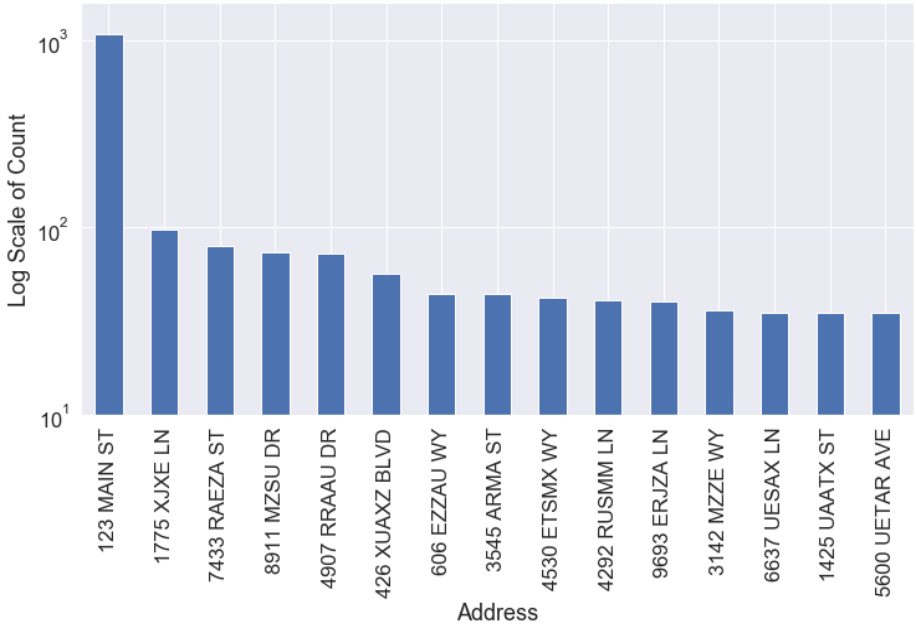
Figure 1.2.2: Distribution of the *ssn* Field (Top 15 Most Common Values)



1.2.3 Field “address”

The *address* field logs the address of the applicant. Address “123 MAIN ST” occurred the most for 1,079 times (see Figure 1.2.3) and is a frivolous value.

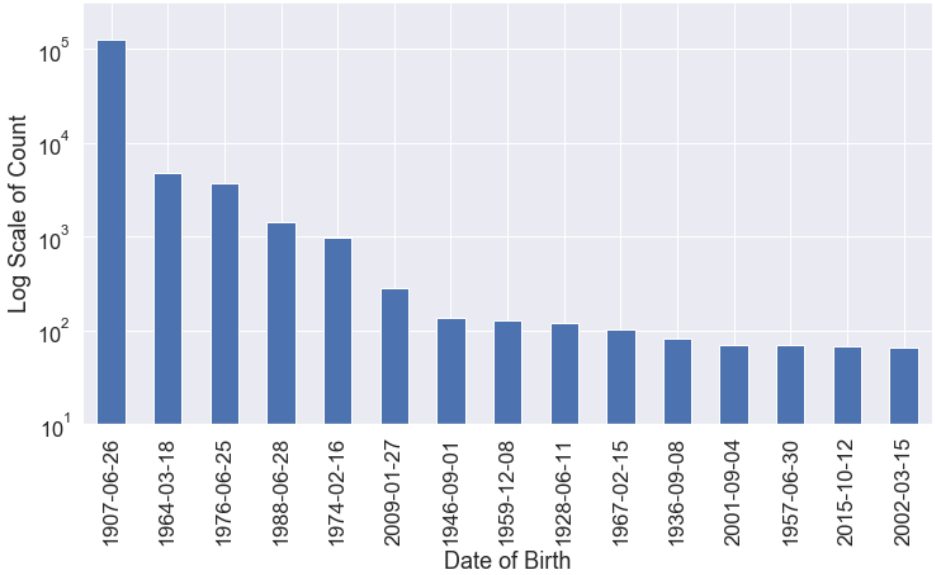
Figure 1.2.3: Distribution of the *address* Field (Top 15 Most Common Values)



1.2.4 Field “dob”

The *dob* field logs the date of birth of the applicant. The field is categorical and ordinal. Date “1907/06/26” occurred the most for 126,568 times (see Figure 1.2.4) and is a frivolous value.

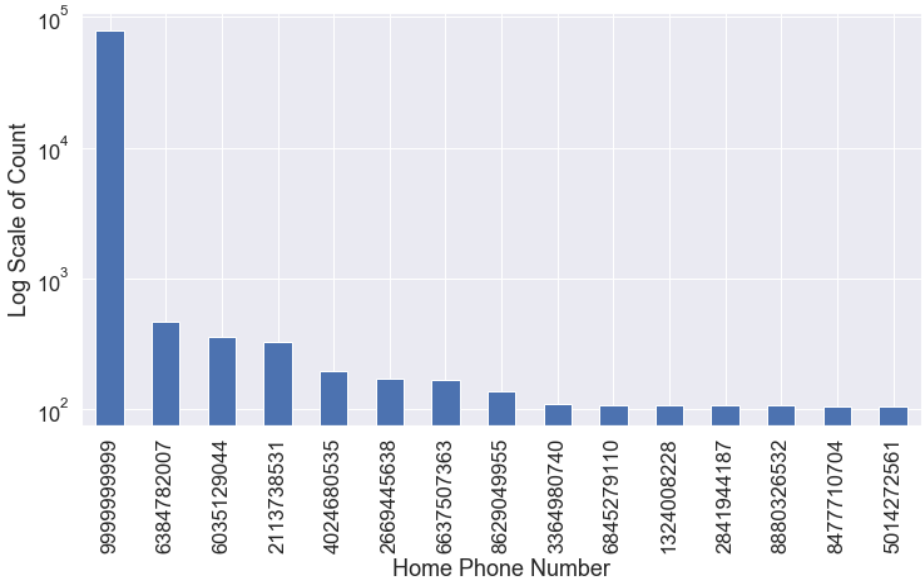
Figure 1.2.4: Distribution of the *dob* Field (Top 15 Most Common Values)



1.2.5 Field “homephone”

The *homephone* field logs the home phone number of the applicant. Number “9999999999” occurred the most for 78,512 times (see Figure 1.2.5) and is a frivolous value.

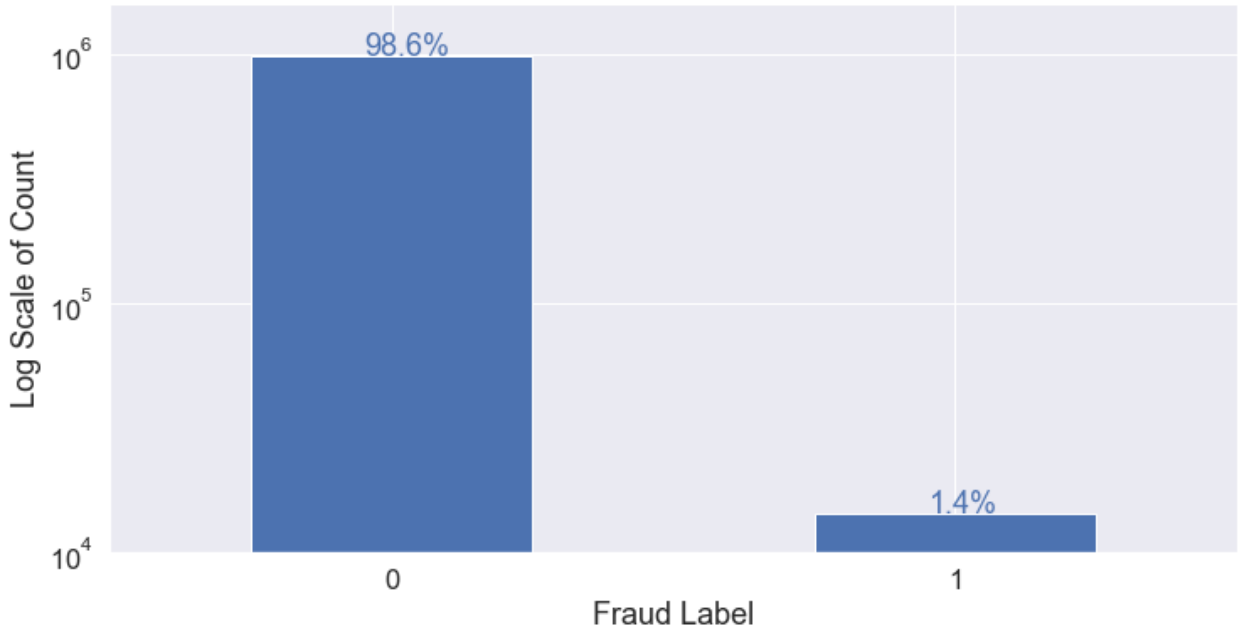
Figure 1.2.5: Distribution of the *homephone* Field (Top 15 Most Common Values)



1.2.6 Field “fraud_label”

The *fraud_label* field takes two values: 0 or 1. Value 1 means the record is a fraud, while value 0 means the record is not a fraud. A total of 14,393 (or 1.4%) records are labeled as fraud (see Figure 1.2.6).

Figure 1.2.6: Distribution of the *fraud_label* Field



2 Data Cleaning

The core design logic of the fraud detection model in this project is to link applications that used the same or similar identity information. Given this logic, values like '999999999' in the *ssn* field or '123 MAIN ST' in the *address* field, which obviously is just placeholder when the field is not collected or populated, would mistakenly connect unrelated applications and confuse the model. Therefore, such values need to be taken care of before modeling. Table 2.0 below lists all frivolous values recognized in this dataset and the values that they are replaced with.

Table 2.0: Frivolous Values

Field	Frivolous Value	Replaced With
ssn	999999999	Negative record number. For example, the *ssn* field of record 976612 is '999999999', and therefore was replaced with '-976612'.
address	123 MAIN ST	
dob	1907-06-26	
homephone	9999999999	

3 Create Candidate Variables

A total of 407 candidate variables were created to quantify the characteristics of fraud behaviors. Table 3.0 below summarizes the description of each variable category and the number of variables created under that category. See Appendix B for the full list of variables.

Table 3.0: Variable Creation Summary

Category	Description	# Variables Created
Weekday Risk	Average likelihood of fraud on a given day of the week	1
Velocity	Occurrence of the same PII or PII combinations within a certain time frame before an application	156
Relative Velocity	Sudden increase of applications using the same PII in the short term in relative to the long term frequency	104
Days Since	Number of days since the last appearance of the same PII	26
Cross Entity Velocity	Frequency of change in the combination of PII within a certain time frame before an application	120

3.1 Weekday Risk

Considering the possibility that the likelihood of someone committing fraud may vary depending on the day of the week, a categorical variable that indicates the weekday on which the application happened was added. Each category in the variable (i.e. Monday) was then replaced with the average probability of fraud in this category.

When calculating the average probabilities, the last two months of data (OOT data) were excluded to avoid overfitting. The average probabilities of fraud by weekday are summarized in Table 3.1 below. For example, all records that happened on a Monday will have a value of 0.0135 in this variable. The resulting new variable was named *weekday_risk.*

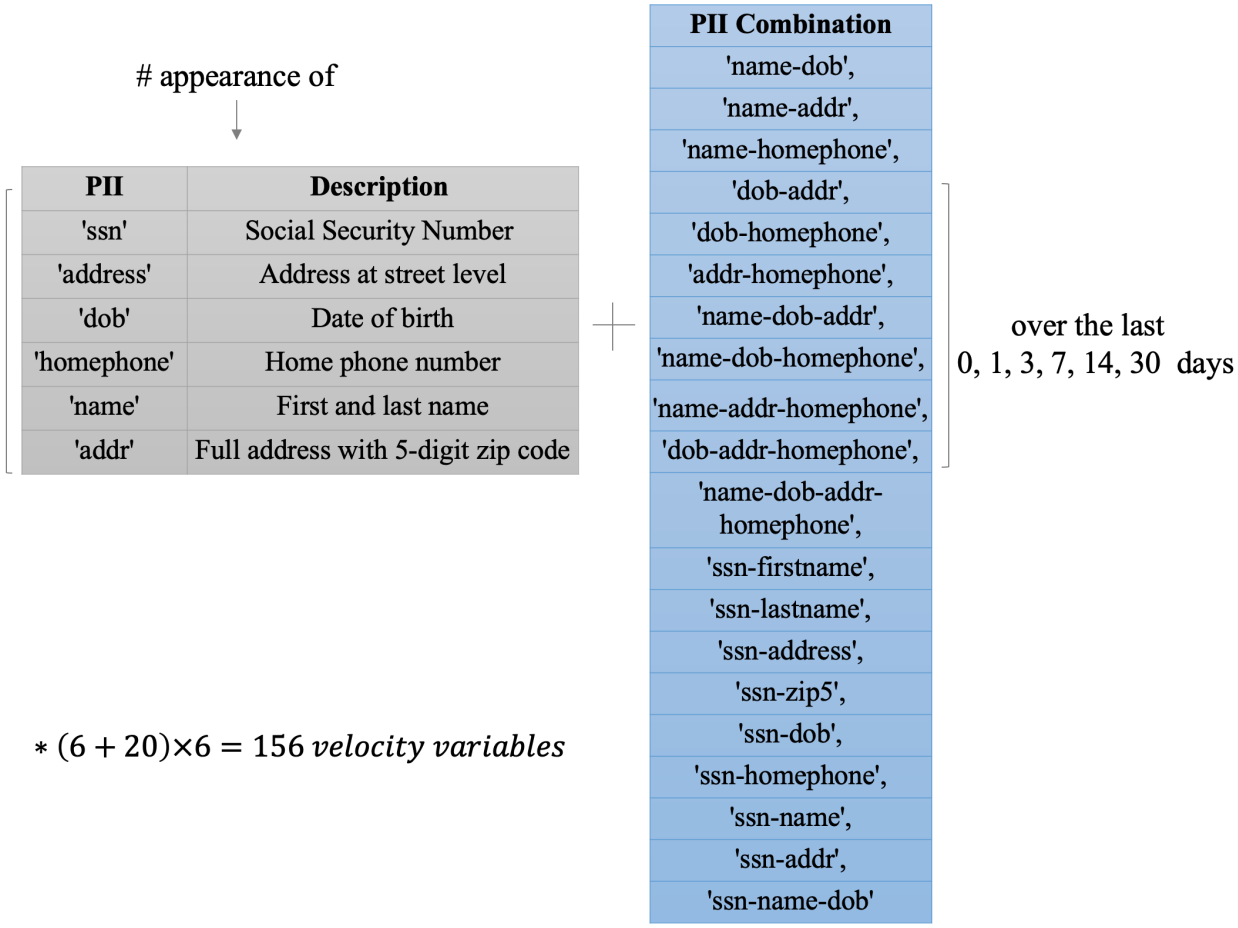
Table 3.1: Average probability of Fraud by Weekday

Weekday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Probability of Fraud	0.0135	0.0141	0.0152	0.0150	0.0145	0.0150	0.0137

3.2 Velocity Variables

The identity fraud in this dataset happened in the form of either one fraudster using multiple victims' core PII or multiple fraudsters using one victim's core PII. Therefore, the frequency of a PII being used is a useful indicator of fraudulent behaviors. Velocity variables, representing the occurrence of the same PII or PII combinations within a certain time frame before a given application, were created to capture such activities. The time frame used were 0, 1, 3, 7, 14 and 30 days, with 0 day meaning the same day up until the time of the application. As shown in Figure 3.2 below, six PII and 20 PII combinations were used to create 156 variables in total.

Figure 3.2: Creation of Velocity Variables



3.3 Relative Velocity Variables

In addition to the velocity variables, relative velocity variables were created to help capture the sudden increase of applications using the same PII in the short term relative to the long term frequency. The rationale behind this is that given the same number of occurrences of the same PII in a set time period (i.e. a 30-day window), a situation where all the occurrences happened on the same day is more likely to be a fraud than another situation where the occurrences were spread out on different dates.

For the PII or PII combination under concern, relative velocity was calculated as the number of the appearance of that PII in the recent past, namely the past 1 day, divided by its average daily appearance in the longer term (see the formula). The longer-term is defined as a time frame of 3, 7, 14 or 30 days. The same 26 PII and PII combinations used in creating the velocity variables were used here. A total of 104 variables were created.

$$\text{Relative velocity} = \frac{\#\text{ apps with that } \textbf{\textit{combination}} \text{ seen in the past 1 day}}{\#\text{ daily apps with that } \textbf{\textit{same combination}} \text{ seen in the past } \{3, 7, 14, 30\} \text{ days}}$$

$* \ 26 \times 4 = 104 \ \textit{relative velocity variables}$

3.4 Days Since Variables

Another way to capture the frequency of appearance of the same PII is to count the number of days since its last appearance. Twenty-six *days since* variables were created to track the 26 baseline PIIs and PII combinations. A smaller value means a closer last appearance of the same PII, and therefore a higher likelihood of fraud. If a PII has never occurred in the dataset before, its corresponding *day since* variable would take the number of days since the earliest date in the dataset, 2016/1/1 to be specific, as its value.

According to this policy, records on later dates would naturally have higher values in the *days since* variables, which would introduce a systematic bias to the data and result in different distributions of the modeling and the validation data that could compromise model performance. Two measures were taken to mitigate such bias:

- All *days since* variables were capped at 60, meaning that values greater than 60 were replaced with 60.
- The first two weeks of data were excluded from model fitting or prediction.

In the real world, fraudsters usually exploit the stolen identity information in a short period of time before the information gets banned (either detected by the bank/fraud detection agency or reported by the victim). Thus a fraud record should have relatively small *days since* values. The pattern shown in the data is in line with this empirical speculation.

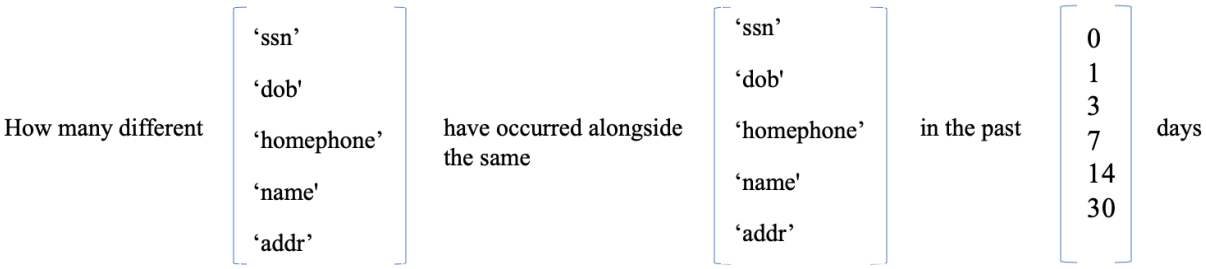
Before capping, 75% fraud records in this dataset have *SSN days since* values no greater than 38, *address days since* values no greater than 15.5, *homephone days since* values no greater than 9, and *name days since* values no greater than 52. Sixty is considered to be a safe upper bound to cut the long right tail of the distribution that would mess up the model while preserving enough information to tell fraud records apart from good records.

On the other hand, records that happened during the first couple of weeks of 2016 have small values in the *days since* variables because the largest possible values these variables can take are the number of days since 2016/1/1, which are small for early records. A record on 2016/1/1 would have value 0 in all its *days since* variables, not because the identity information used was already seen before on the same day, which would be a flag for fraudulent behavior, but simply because there was no data before this date to trace back. To avoid such misleading information being fed into the model, the first two weeks of records were excluded from modeling.

3.5 Cross Entity Velocity Variables

The rationale of creating cross entity velocity variables is to detect the frequent change of the PII combinations. For example, if the same SSN is frequently used along with different addresses or date of birth for applications in a short period of time, such applications signal plausible identity theft. Figure 3.5.1 below demonstrates how cross entity variables were created on core PII.

Figure 3.5.1: Creation of Cross Entity Variables



* A field does not cross with itself
** 5 x 4 x 6 = 120 variables

Table 3.5.2 provides an example with made-up information. Suppose that record 50 is the first time SSN "123456789" appeared in the dataset and the applicant name appeared in the same application is Man, Leon. The next time this SSN appeared was in record 150, alongside with name Han, Boyang. Therefore, the 7-day SSN-name cross entity velocity variable *ssn_lag7_name* takes value 2, as there were two different names used along with that SSN within the past seven days. The *ssn_lag7_name* of record 200 remained 2 because this application didn't use a new name with the same SSN "123456789". Value 2 and 1are assigned to record 250 and 123712 respectively following the same logic. It's worth noting that, the greater the cross entity velocity variables are, the more likely an application is a potential fraud.

Table 3.5.2: 7-Day SSN-Name Cross Entity Velocity Variable Example

record	date	ssn	name	ssn_lag7_name
50	3/1/16	123456789	Man, Leon	1
……	……	……	……	……
150	3/3/16	123456789	Han, Boyang	2
……	……	……	……	……
200	3/5/16	123456789	Han, Boyang	2
……	……	……	……	……
250	3/7/16	123456789	Wen, Juno	3
……	……	……	……	……
123712	10/1/16	123456789	Zhang, Yangzi	1

4 Feature Selection

Feature selection is the process of selecting the features that contribute the most to predicting the output of interest. Section 4.1 below explains why it is performed, section 4.2 lists the preparation steps, and sections 4.3 and 4.4 elaborate in detail how it is performed. Univariate filter and backward-elimination wrapper methods were used in this project. Out of 407 candidate variables, 25 were selected for modeling.

4.1 Why is Feature Selection Necessary

Having high dimensionality of too many features in the data can decrease model performance and increase computation time, hence feature selection is a necessary step before the modeling.

When analyzing data in high-dimensional space, various phenomena arise make machine learning challenging. Typically, such phenomena is referred to as the curse of dimensionality. As dimensionality increases, data quickly becomes sparse and all objects appear to be dissimilar, which makes it harder for a model to learn patterns. Moreover, with higher dimensionality comes more noise, and thus more data is needed to tell the true pattern from noise. And this need for data grows exponentially as the number of features grows.

On the other hand, a large number of features usually mean more irrelevant or correlated features, which could result in a dumb model with less accurate predictions. Such redundant features should be removed for both performance and efficiency concerns.

For all aforementioned reasons, feature selection was performed for this project, which went through the list of candidate variables and found the best predictors to be used for modeling

4.2 Preparation for Feature Selection

The following steps were taken to prepare the data for feature selection:

1. **Z-scale all features:** transform all variables to z-scores; bring all variables to the same scale so that any distance based algorithm would not be mislead
2. **Exclude first two weeks of data:** avoid biased *days since* variables caused by insufficient data; details explained in section 3.4
3. **Split the dataset into training, testing and out-of-time validation data:**
    a. Make possible to evaluate model performance on new data drawn from the same or different time period;
    b. Table 4.2 below summarizes how the split was performed and the basic statistics of the resulting data
    c. Feature selection was only performed on the modeling data to avoid overfitting

Table 4.2: Training, Testing, OOT Data Split Summary

Data		Criteria	# Records	# Fraud	% Fraud
Modeling	**Training**	randomly picked 80% of data between 2016/1/15 and 2016/10/31	635,996	9,189	1.44%
	Testing	randomly picked 20% of data between 2016/1/15 and 2016/10/31	159,000	2,297	1.44%
Out of Time/Validation		data between 2016/10/1 and 2016/12/31	166,493	2,386	1.43%
Total			961,489	13,872	1.44%

4.3 Filter

A filter method is a fast feature selection method that examines how good each independent variable by itself predicts the target. It is independent of any modeling method. Common filter methods for binary classification problems include Pearson Correlation, Fisher Score, Mutual Information, univariate Kolmogorov-Smirnov (KS) Score and Fraud Detection Rate (FDR).

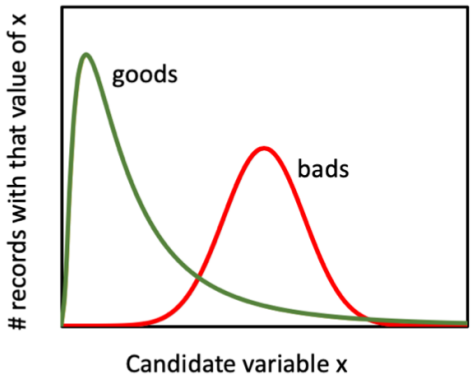
For the purpose of this project, the univariate KS Score and FDR were used to rank the 407 candidate variables. The average ranking by two scores was then used to filter out the top 80 variables to be moved on to the next step of feature selection.

4.3.1 Univariate Kolmogorov-Smirnov (KS)

Why is KS a good filter?

The importance of a feature in supervised fraud detection problems can be measured by how well it separates the bad records from the good ones. Figure 4.3.1.1 below plots the density curve of the good records against the density curve of the bad records for a given variable x. The more disparate the curves are, the better the variable is as a predictor.

Figure 4.3.1.1 Distribution of Goods vs Bads Based on Candidate Variable x

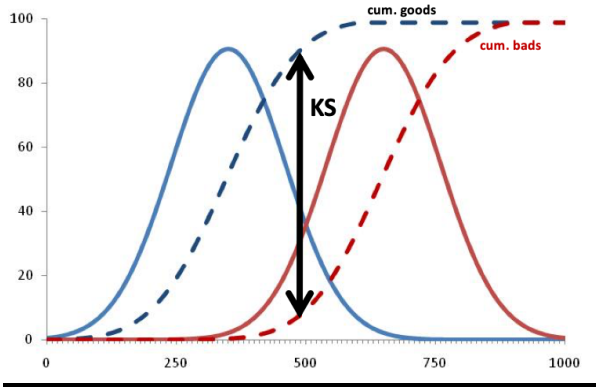


The distance between two curves can be nicely measured by the Kolmogorov-Smirnov Score, a simple, robust and scale-independent statistic. The higher the KS score, the more important a feature is in predicting fraud.

How does it work?

Figure 4.3.1.2 below visualizes how univariate KS is calculated for each candidate variable. It is the maximum of the difference between the cumulative goods and bads.

Figure 4.3.1.2: Demonstration of KS



The mathematical expression of KS is shown in the formula below:

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

4.3.2 Fraud Detection Rate (FDR)

Why is FDR a good filter?

Another way to determine the importance of a feature in fraud detection problems is to measure how well it catches frauds at a given threshold. FDR is such a robust statistic that calculates the percentage of total frauds caught at a given rejection threshold. For example, suppose that there are 10 frauds in a total of 1000 applications. If we reject 3% of the applications, namely 30 applications, and find out that 6 out of the 30 are actually frauds, then the FDR at 3% is calculated as 6 divided by 10, which is 0.6. The higher the FDR is, the better the variable is as a predictor for fraud.

How does it work?

For each feature X, FDR at 3% is computed as the percentage of total frauds caught in the top 3% of the population sorted by X, with the top rows being most likely to be a fraud and the bottom rows the opposite (see the formula below).

$$\text{FDR} = \frac{\#\,frauds\ caught\ at\ 3\%\ rejection\ rate}{total\ \#\,frauds\ in\ the\ dataset}$$

4.3.3 Combine KS and FDR outputs

To balance the weakness of individual filters, the output of KS and FDR were combined to determine the selection of variables to move on with. Records were first ranked by the KS and FDR scores respectively and then the average ranking of the two scores was used as the final filter score (see the formula below).

$$\text{Average Rank of Scores} = \frac{\text{Rank based on KS (Descending)} + \text{Rank based on FDR (Descending)}}{2}$$

As the desired number of input variables for modeling is between 20 and 30, the top 80 candidate variables with the highest final filter scores were selected to be passed on to the next stage of feature selection. A full list of the top 80 variables, together with their scores and rankings, can be found in Appendix C.

4.4 Wrapper

4.4.1 What is a Wrapper

A wrapper is a stepwise feature selection method that evaluates the features of a specific machine learning algorithm to find an optimal feature combination. It follows a greedy search approach by evaluating all the possible combinations of features against the evaluation criterion. For computation time concerns, Logistic Regression was chosen as the machine learning algorithm to be wrapped around.

In addition, because the fraud dataset is very imbalanced and accuracy can be misleading in such case, ROC-AUC (Area Under the Receiver Operating Characteristic Curve) was used as the evaluation criterion. AUC is very sensitive to class imbalance. When there is a minority class in the data, the fraud class in the case of this project, the AUC score will be strongly impacted by the minority class, meaning that it is less likely the fraud records will be misclassified to maximize the correct classification of the majority class and to minimize overall misclassification error. Hence the results are more accurate and reliable.

There are three commonly used wrapper techniques: forward selection, backward selection, and general stepwise selection. For this project, a backward selection wrapper was adopted. The backward selection method starts with a full model, which includes all the features, and removes one feature at a time. A feature is chosen to be removed if its removal results in the highest improvement or the least damage to the classifier performance (i.e. AUC score). Moreover, cross-validation was incorporated to ensure more reliable results. This process repeats until the model degradation, resulted by variable elimination, is below an acceptable amount and then, the optimal feature subset is found.

4.4.2 How Does the Wrapper Work

Considering the randomness embedded in the wrapper method, the wrapper was ran six times and the key results are shown in Table 4.4.2.1 (not in the order of execution):

Table 4.4.2.1: Wrapper Results by Iteration

Iteration	Beginning # Features	# Features Remaining					
		Round 1	Round 2	Round 3	Round 4	Round 5	Round 6
1	80	34 (cv=2)	23 (cv=4)				
2	80	45 (cv=2)	41 (cv=2)	39 (cv=2)	38 (cv=2)	32 (cv=4)	25 (cv=4)
3	80	59 (cv=2)	35 (cv=2)	29 (cv=2)	21 (cv=2)	20 (cv=4)	
4	80	76 (cv=2)	61 (cv=2)	56 (cv=2)	34 (cv=2)	21 (cv=4)	
5	80	38 (cv=2)	37 (cv=10)	32 (cv=7)	22 (cv=5)		
6	80	49 (cv=2)	38 (cv=5)	30 (cv=8)	25 (cv=10)		

In the case of highly correlated variables, a wrapper could choose any one of the correlated variables when deciding which variable to remove while getting similar results (i.e. AUC score). Therefore, the resulting ranking of features from each iteration is slightly different. To narrow down the top 20-30 best features for modeling, the top 30 features from each iteration were examined and the most frequent subset among all iterations was picked to be used for modeling. The final features selected are listed in Table 4.4.2.2 below.

Table 4.4.2.2: Final Candidate Variables

Rank	Variable	Variable Description
1	addr_lag30_name	# different **full name** appeared alongside the given **full address** in the past 30 days
2	addr_lag7_name	# different **full name** appeared alongside the given **full address** in the past 7 days
3	addr_lag1_ssn	# different **SSN** appeared alongside the given **full address** in the past 1 day
4	addr_lag7_ssn	# different **SSN** appeared alongside the given **full address** in the past 7 days
5	addr_lag1_count	# appearance of the given **full address** in the past 1 day
6	addr_lag30_count	# appearance of the given **full address** in the past 30 days
7	addr_lag7_count	# appearance of the given **full address** in the past 7 days
8	address_#days_since	# days since the last appearance of the given **street address**
9	address_lag7_count	# appearance of the given **street address** in the past 7 days
10	homephone_lag7_name	# different **full name** appeared alongside the given **home phone** in the past 7 days
11	homephone_lag3_ssn	# different **SSN** appeared alongside the given **home phone** in the past 3 days
12	homephone_lag7_ssn	# different **SSN** appeared alongside the given **home phone** in the past 7 days
13	homephone_lag7_count	# appearance of the given **home phone** in the past 7 days
14	name-dob_#days_since	# days since the last appearance of the given **full name + DOB** combination
15	name-dob_lag30_count	# appearance of the given **full name + DOB** combination in the past 30 days
16	ssn-dob_#days_since	# days since the last appearance of the given **SSN + DOB** combination
17	ssn-name-dob_#days_since	# days since the last appearance of the given **SSN + full name + DOB** combination
18	ssn-name_#days_since	# days since the last appearance of the given **SSN + full name** combination
19	ssn-name_lag7_count	# appearance of the given **SSN + full name** combination in the past 7 days
20	addr_lag7_dob	# different **DOB** appeared alongside the given **full address** in the past 7 days
21	name-dob_lag14_count	# appearance of the given **full name + DOB** combination in the past 14 days
22	addr_lag14_count	# appearance of the given **full address** in the past 14 days
23	addr_lag14_name	# different **full name** appeared alongside the given **full address** in the past 14 days
24	addr_lag0_dob	# different **DOB** appeared alongside the given **full address** in the past 0 day
25	addr_#days_since	# days since the last appearance of the given **full address**

5 Model Fitting

The modeling process is composed of three major steps: data preprocessing, model training and model testing. The data preprocessing applied Capping to all data and weighting to the training data. After that, as shown in Table 5.0, the team fitted a number of machine learning models and fine-tuned hyperparameters of each model respectively. Models include Logistics Regression, Gradient Boosting Tree, Random Forest, Neural Network, K-Nearest Neighbors, Decision Tree, and Support Vector Machine. FDR at a 3% rejection rate was used to evaluate model performance and was calculated as the average of 5 independent fittings. Gradient Boosting Tree (iteration 2 highlighted in black) has the highest performance on the OOT data and therefore was selected as the final model. The details of the final model will be elaborated in the Results section (Section 6).

Table 5.0: Model Performance Summary

Model	Parameters										Average FDR at 3%		
Logistics Regression	**Iteration**	**#Variables**	**penalty**	**C**	**solver**	**r1_ratio**					**Train**	**Test**	**OOT**
	1 (default)	25	l1	1	liblinear	N/A					55.61	54.24	52.62
	2	25	l2	1	lbfgs	N/A					55.58	54.23	52.63
	3	25	l2	10	lbfgs	N/A					55.6	54.25	52.63
	4	25	l2	0.1	lbfgs	N/A					55.61	54.29	52.68
	5	25	ElasticNet	1	sage	0.2					55.6	54.29	52.68
	6	25	ElasticNet	1	sage	0.4					55.61	54.29	52.68
	7	25	ElasticNet	1	sage	0.6					55.61	54.29	52.68
	8	25	ElasticNet	1	sage	0.8					55.61	54.29	52.68
	9	20	l1	1	liblinear	N/A					55.54	54.16	52.62
	10	20	l2	1	lbfgs	N/A					55.57	54.16	52.64
	11	20	ElasticNet	1	sage	0.2					55.57	54.16	52.64
	12	20	ElasticNet	1	sage	0.4					55.56	54.16	52.64
	13	20	ElasticNet	1	sage	0.6					55.57	54.16	52.64
	14	20	ElasticNet	1	sage	0.8					55.57	54.16	52.64
Gradient Boosting Tree	**Iteration**	**learning_rate**	**n_estimators**	**max_depth**	**max_features**	**min_samples_leaf**	**min_samples_split**			**subsample**	**Train**	**Test**	**OOT**
	1 (default)	0.1	100	3	None	1	2			1	56.73	55.68	53.96
	2	**0.01**	**800**	**5**	**5**	**30**	**1500**			**0.7**	**57.17**	**55.77**	**54.07**
	3	0.05	210	5	5	30	1100			0.7	56.83	55.68	53.96
	4	0.05	240	5	25	30	500			0.7	57.05	55.64	53.94
	5	0.02	240	5	25	30	500			0.7	56.68	55.51	53.98
	6	0.001	4000	5	25	40	500			0.7	56.4	55.29	53.94
Random Forest	**Iteration**	**bootstrap**	**n_estimators**	**max_depth**	**max_features**	**min_samples_leaf**	**min_samples_split**			**criterion**	**Train**	**Test**	**OOT**
	1 (default)	TRUE	100	None	5	1	2			gini	58.61	55.01	53.65
	2	TRUE	50	20	5	30	500			entropy	56.33	54.9	53.31
	3	TRUE	50	20	5	30	300			gini	56.66	55.32	53.73
Neural Network	**Iteration**	**layer**	**nodes**	**max_iter**	**activation**	**optimizer**	**alpha**	**learning_rate**	**learning_rate_init**	**momentum**	**nesterovs_momentum**	**Train**	**Test**	**OOT**
	1 (default)	1	100	200	relu	adam	0.0001	N/A	N/A	N/A	N/A	56.78	55.71	53.88
	2	1	100	500	relu	sgd	0.0001	constant	0.005	0.9	TRUE	56.49	55.59	53.81
	3	1	100	50	relu	adam	0.0001	N/A	0.01	N/A	N/A	56.76	55.85	53.99
	4	1	5	100	relu	adam	0.0001	N/A	0.005	N/A	N/A	56.42	55.45	53.72
	5	1	50	500	relu	sgd	0.001	adaptive	0.02	0.9	TRUE	56.66	55.7	53.91
	6	1	100	200	relu	sgd	0.0001	adaptive	0.005	0.9	TRUE	56.52	55.62	53.86
	7	1	100	200	relu	sgd	0.0003	constant	0.0759	0.1	FALSE	56.59	55.53	53.86
	8	1	100	300	logistic	adam	0.0008	invscaling	0.0397	N/A	N/A	56.41	55.81	53.73
K-Nearest Neighbors	**Iteration**	**neighbors**					**weights**					**Train**	**Test**	**OOT**
	1 (default)	5					uniform					56.92	53.53	52.23
	2	5					distance					56.97	53.17	52.05
	3	7					uniform					57.27	54.65	53.24
	4	12					uniform					57.1	54.92	53.32
Decision Tree	**Iteration**	**criterion**	**max_depth**	**min_samples_leaf**	**min_samples_split**	**splitter**						**Train**	**Test**	**OOT**
	1 (default)	gini	None	1	2	"best"						58.85	53.33	52.72
	2	gini	20	60	310	random						56.39	54.89	53.55
	3	gini	20	60	300	best						56.75	55.07	53.89
	4	gini	default	79	1350	random						56.08	54.68	53.48
Support Vector Machine	**Iteration**	**kernel**										**Train**	**Test**	**OOT**
	1	linear										54.92	54.41	53.12

5.1 Data Preprocessing

The purpose of data preprocessing is to help the algorithms better learn from the data. Two methods, capping, and weighting, were adopted to achieve this goal.

5.2.1 Capping

How does it work?

All z-scaled variables were capped at 6 z-scores. Values more than 6 standard deviations away from the variable mean were transformed to ±6. Capping was applied to all training, testing and OOT data.

Why does it help?

Capping forces the model to focus on value ranges that best differentiate the good and the fraud records. Many variables have extremely high values indicating "very likely to be fraud", but frequently only "slightly likely to be fraud" is necessary for the model to learn. Practically, models perform better by focusing on areas that really matter.

5.2.2 Weighting

How does it work?

The good records were subsampled so that the good-to-bad ratio was reduced from 68:1 to 10:1. In this way, a relatively balanced dataset was obtained without losing too many data points. Resampling was performed on the training data every time a model was fitted to average the variances resulted from the randomness in subsampling.

Why does it help?

It moderates the problem brought by imbalanced data. Machine learning classifiers typically fail to cope with imbalanced training datasets as they aim to minimize the overall misclassification error instead of paying special attention to the correct classification of minority classes (frauds) that this project is interested in.

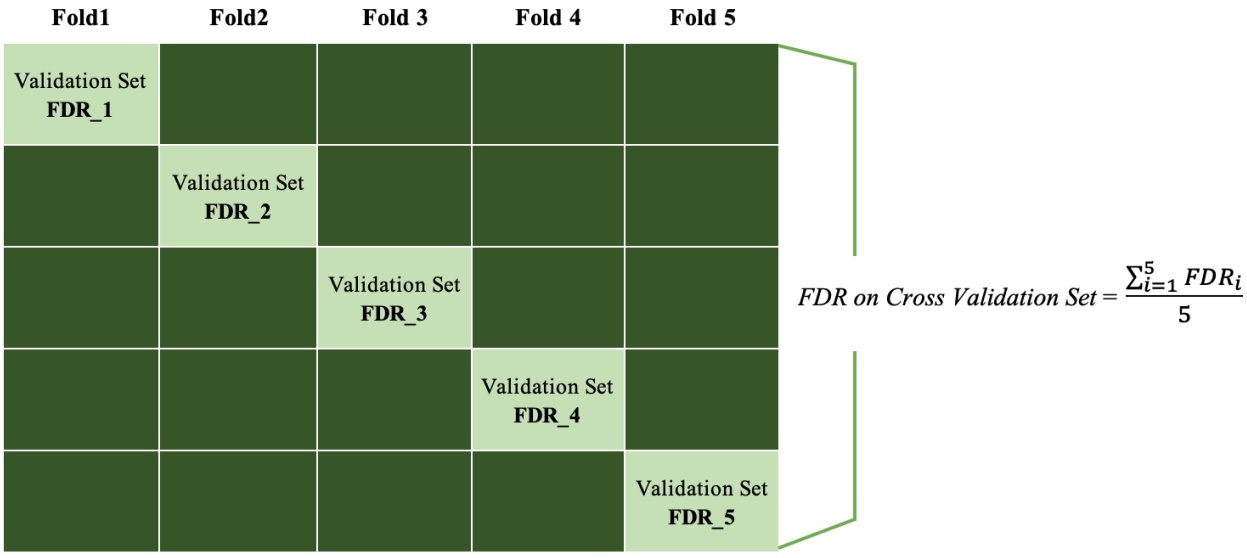
5.3 Model Training

After data preprocessing, the training dataset is ready for model fitting. During the process, a handful of different machine learning algorithms were tried. Hyperparameter tuning techniques such as random search and grid search were performed to find the optimal parameters for each model that achieves the best performance on new data.

Hyperparameters are properties that are set before the learning process begins and govern the entire training process. Some examples of hyperparameters include learning rate, regularization strength, and a number of epochs. The team paid special attention to adjust the regularization parameters and the best regularization strength is identified to avoid overfitting.

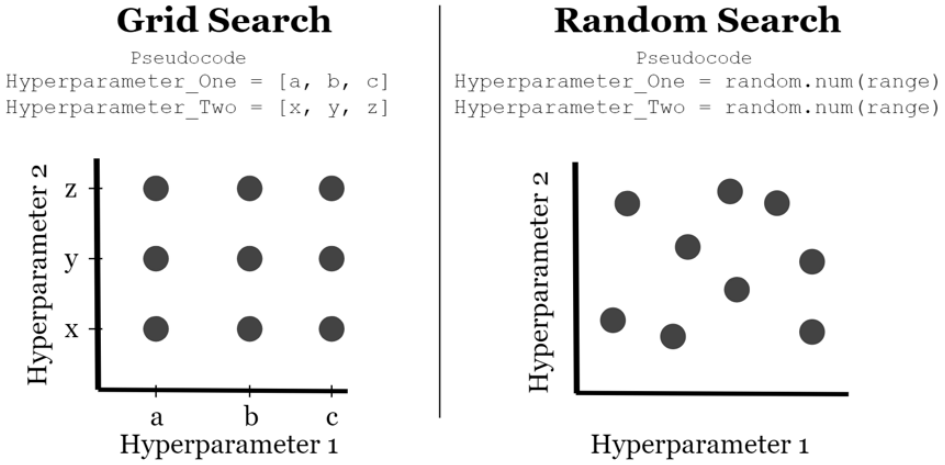
Hyperparameter tuning is the process of choosing the optimal sets of hyperparameters for each learning algorithm. The five-fold cross-validation method is introduced to assist hyperparameter tuning as it offers a more realistic estimate of the model performance by taking the variance into account (as illustrated in Figure 5.3.1). For the purpose of this project, optimal parameters are defined as the combination of hyperparameters that yields the highest average FDR on the validation sets of the training data.

Figure 5.3.1: Measure FDR Performance Through Cross Validation



The methods adopted for tuning are random search and grid search. As illustrated in Figure 5.3.2, grid search works by searching exhaustively through a specific subset of hyperparameters, which on the one hand, guarantees to find the optimal combination. But on the other hand, it can be very time consuming and computationally expensive. The random search method is introduced to combat this drawback. The technique searches hyperparameters randomly instead of exhaustively, with the biggest benefits being the decrease in processing time.

Figure 5.3.2: Visual Illustration of Grid Search and Random Search



The team combined the two methods in the effort to find optimal or near-optimal hyperparameters within reasonable processing time. The process involves first exploring a broader range (or all possible options) of hyperparameters with random search, and subsequently refining the search with grid search within small regions that look promising. Details of hyperparameter combinations tried are enclosed in Appendix C.

5.4 Model Testing

After the best candidate combinations of parameters for each learning algorithm were obtained in the model fitting phase, all classifiers were moved to the testing phase. Model testing aims at estimating how well the model generalizes on unseen data by examining the model performance on the testing and the OOT data.

All candidate models were re-trained on weighted training data (good-to-bad ratio set to 10:1) and then tested on the full training, testing and OOT data. FDR at a 3% rejection rate was used for model comparison. Due to the variance caused by noise and the randomness in the fitting process, estimates of model performances vary at each iteration. Therefore, the training and testing procedure was conducted 5 times to obtain the average FDR score of each model on the training, testing, and OOT data respectively.

6 Results

Out of all the models, Gradient Boosting Tree (*learning_rate* = 0.01, *n_estimators* = 800, *max_depth* = 5, *max_features* = 5, *min_samples_leaf* = 30, *min_samples_split* = 1500, *subsample* = 0.7) was selected as the best and the final model for that it achieved the highest FDR at 3% rejection rate on the OOT data. The FDR scores of this model on the training, testing, and OOT data are 57.17%, 55.77%, and 54.07% respectively. The detailed statistics of the model performance on the training, testing, and oot data are shown in Table 7.1-7.3.

For each of the three sets, records were split into 100 bins after being sorted in the descending order based on the predicted probability of being a fraud. The first bin in each table has the highest number of frauds ("# Bads"), indicating that the model effectively captured the majority of the fraudulent applications. The green shaded area summarizes the basic statistics within each bin, while the blue shaded area reflects cumulative statistics of all the bins above, including the current bin.

Table 6.1: Key Statistics of Top 20 Bins in Training Data

Training	# Records	# Goods	# Bads	Fraud Rate
	635996	626807	9189	0.014448204

	Bin Statistics					Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	6360	1460	4900	22.96%	77.04%	6360	1460	4900	0.23%	53.32%	53.09	0.30
2	6360	6094	266	95.82%	4.18%	12720	7554	5166	1.21%	56.22%	55.01	1.46
3	6360	6273	87	98.63%	1.37%	19080	13827	5253	2.21%	57.17%	54.96	2.63
4	6360	6277	83	98.69%	1.31%	25440	20104	5336	3.21%	58.07%	54.86	3.77
5	6360	6303	57	99.10%	0.90%	31800	26407	5393	4.21%	58.69%	54.48	4.90
6	6360	6298	62	99.03%	0.97%	38160	32705	5455	5.22%	59.36%	54.15	6.00
7	6360	6302	58	99.09%	0.91%	44520	39007	5513	6.22%	60.00%	53.77	7.08
8	6360	6310	50	99.21%	0.79%	50880	45317	5563	7.23%	60.54%	53.31	8.15
9	6360	6307	53	99.17%	0.83%	57240	51624	5616	8.24%	61.12%	52.88	9.19
10	6360	6308	52	99.18%	0.82%	63600	57932	5668	9.24%	61.68%	52.44	10.22
11	6360	6317	43	99.32%	0.68%	69960	64249	5711	10.25%	62.15%	51.90	11.25
12	6360	6309	51	99.20%	0.80%	76320	70558	5762	11.26%	62.71%	51.45	12.25
13	6360	6305	55	99.14%	0.86%	82680	76863	5817	12.26%	63.30%	51.04	13.21
14	6360	6312	48	99.25%	0.75%	89040	83175	5865	13.27%	63.83%	50.56	14.18
15	6360	6308	52	99.18%	0.82%	95400	89483	5917	14.28%	64.39%	50.12	15.12
16	6360	6306	54	99.15%	0.85%	101760	95789	5971	15.28%	64.98%	49.70	16.04
17	6360	6312	48	99.25%	0.75%	108120	102101	6019	16.29%	65.50%	49.21	16.96
18	6360	6309	51	99.20%	0.80%	114480	108410	6070	17.30%	66.06%	48.76	17.86
19	6360	6322	38	99.40%	0.60%	120840	114732	6108	18.30%	66.47%	48.17	18.78
20	6360	6315	45	99.29%	0.71%	127200	121047	6153	19.31%	66.96%	47.65	19.67

Different from the green shaded area, the "% Goods" column in the blue area represents the percentage of total good records that fall in the current bin and the bins above. The "% Bads" column calculated the percentage of total bad records caught so far, starting from the top, which essentially is the FDR at the threshold. Moreover, the "KS" column is the difference between the cumulative "% Bads" and the cumulative "% Goods", and the "FPR" (False Positive Rate) column is the ratio of "Cumulative Goods" to "Cumulative Bads".

Table 6.2: Key Statistics of Top 20 Bins in Testing Data

Testing	# Records	# Goods	# Bads	Fraud Rate
	159000	156703	2297	0.014446541

	Bin Statistics					Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	1590	396	1194	24.91%	75.09%	1590	396	1194	0.25%	51.98%	51.73	0.33
2	1590	1527	63	96.04%	3.96%	3180	1923	1257	1.23%	54.72%	53.50	1.53
3	1590	1566	24	98.49%	1.51%	4770	3489	1281	2.23%	55.77%	53.54	2.72
4	1590	1570	20	98.74%	1.26%	6360	5059	1301	3.23%	56.64%	53.41	3.89
5	1590	1570	20	98.74%	1.26%	7950	6629	1321	4.23%	57.51%	53.28	5.02
6	1590	1579	11	99.31%	0.69%	9540	8208	1332	5.24%	57.99%	52.75	6.16
7	1590	1578	12	99.25%	0.75%	11130	9786	1344	6.24%	58.51%	52.27	7.28
8	1590	1575	15	99.06%	0.94%	12720	11361	1359	7.25%	59.16%	51.91	8.36
9	1590	1575	15	99.06%	0.94%	14310	12936	1374	8.26%	59.82%	51.56	9.41
10	1590	1577	13	99.18%	0.82%	15900	14513	1387	9.26%	60.38%	51.12	10.46
11	1590	1577	13	99.18%	0.82%	17490	16090	1400	10.27%	60.95%	50.68	11.49
12	1590	1575	15	99.06%	0.94%	19080	17665	1415	11.27%	61.60%	50.33	12.48
13	1590	1578	12	99.25%	0.75%	20670	19243	1427	12.28%	62.12%	49.84	13.48
14	1590	1579	11	99.31%	0.69%	22260	20822	1438	13.29%	62.60%	49.32	14.48
15	1590	1578	12	99.25%	0.75%	23850	22400	1450	14.29%	63.13%	48.83	15.45
16	1590	1576	14	99.12%	0.88%	25440	23976	1464	15.30%	63.74%	48.44	16.38
17	1590	1585	5	99.69%	0.31%	27030	25561	1469	16.31%	63.95%	47.64	17.40
18	1590	1578	12	99.25%	0.75%	28620	27139	1481	17.32%	64.48%	47.16	18.32
19	1590	1583	7	99.56%	0.44%	30210	28722	1488	18.33%	64.78%	46.45	19.30
20	1590	1578	12	99.25%	0.75%	31800	30300	1500	19.34%	65.30%	45.97	20.20

Table 6.3: Key Statistics of Top 20 Bins in OOT Data

OOT	# Records	# Goods	# Bads	Fraud Rate
	166493	164107	2386	0.014330933

	Bin Statistics					Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	1665	458	1207	27.51%	72.49%	1665	458	1207	0.28%	50.59%	50.31	0.38
2	1665	1604	61	96.34%	3.66%	3330	2062	1268	1.26%	53.14%	51.89	1.63
3	1665	1643	22	98.68%	1.32%	4995	3705	1290	2.26%	54.07%	51.81	2.87
4	1665	1649	16	99.04%	0.96%	6660	5354	1306	3.26%	54.74%	51.47	4.10
5	1665	1649	16	99.04%	0.96%	8325	7003	1322	4.27%	55.41%	51.14	5.30
6	1665	1651	14	99.16%	0.84%	9990	8654	1336	5.27%	55.99%	50.72	6.48
7	1665	1651	14	99.16%	0.84%	11655	10305	1350	6.28%	56.58%	50.30	7.63
8	1665	1652	13	99.22%	0.78%	13320	11957	1363	7.29%	57.12%	49.84	8.77
9	1665	1648	17	98.98%	1.02%	14985	13605	1380	8.29%	57.84%	49.55	9.86
10	1665	1654	11	99.34%	0.66%	16650	15259	1391	9.30%	58.30%	49.00	10.97
11	1665	1652	13	99.22%	0.78%	18315	16911	1404	10.30%	58.84%	48.54	12.04
12	1665	1645	20	98.80%	1.20%	19980	18556	1424	11.31%	59.68%	48.37	13.03
13	1665	1654	11	99.34%	0.66%	21645	20210	1435	12.32%	60.14%	47.83	14.08
14	1665	1657	8	99.52%	0.48%	23310	21867	1443	13.32%	60.48%	47.15	15.15
15	1665	1657	8	99.52%	0.48%	24975	23524	1451	14.33%	60.81%	46.48	16.21
16	1665	1653	12	99.28%	0.72%	26640	25177	1463	15.34%	61.32%	45.97	17.21
17	1665	1650	15	99.10%	0.90%	28305	26827	1478	16.35%	61.94%	45.60	18.15
18	1665	1655	10	99.40%	0.60%	29970	28482	1488	17.36%	62.36%	45.01	19.14
19	1665	1652	13	99.22%	0.78%	31635	30134	1501	18.36%	62.91%	44.55	20.08
20	1665	1647	18	98.92%	1.08%	33300	31781	1519	19.37%	63.66%	44.30	20.92

Looking across the three tables, the model performs the best on the training data (highest FDR in each bin). It then slightly deteriorates on the testing data and performs the worst on the OOT data. This pattern is expected as models usually perform better on data that they've seen before and on data that are drawn from a similar time period. The modest degradation from training to testing, and then to OOT indicates that the model generalizes well.

7 Conclusion

The report documents the process of building a real-time fraud detection system to capture identity fraud, namely identity theft, that happens during the product application process. The project can be broken down to five main parts:

- **Data exploration:** understood the characteristics of the given dataset
- **Data cleaning**: replaced frivolous values with unique numbers that would not trigger false linkage between irrelevant applications
- **Variable creation**: created 407 expert variables that captured 1) the frequent occurrence of the same PII or PII combination, 2) the frequent change of PII combination, and 3) the likelihood of fraud on a given day of the week
- **Feature selection**: applied two univariate filters (KS and FDR) to narrow down candidate variables from 407 to 80, and then a backward selection wrapper to select the final 25 variables for modeling
- **Statistical modeling**: built, tuned and tested a handful of machine learning algorithms including Logistic Regression, Gradient Boosting Tree, Random Forest, Neural Network, K-Nearest Neighbors, Single Tree, and Support Vector Machine

Based on the performance on the OOT data, a Gradient Boosting Tree classifier with the following parameters was selected as the best and final model: *learning_rate* = 0.01, *n_estimators* = 800, *max_depth* = 5, *max_features* = 5, *min_samples_leaf* = 30, *min_samples_split* = 1500, and *subsample* = 0.7. The FDR scores of this model on the training, testing, and OOT data are 57.17%, 55.77%, and 54.07% respectively.

Due to the time constraint of the project, there were only a limited amount of things that could be explored. However, if more time was given, the final model could have been improved through the ways suggested below:

- Consult experts in the field to create more powerful variables that can separate fraud and non-fraud records better, which could lead to better model performance
- Conduct more thorough hyperparameter searching and try more classifiers to optimize model performance
- Use a higher number of folds (i.e. 10) in cross-validation during hyperparameter tuning to get more reliable scores
- Fit and test on the testing and OOT data more times to further reduce variance and get more reliable results

Appendix A: Data Quality Report

The “applications data.csv” is a simulated dataset that stores 1,000,000 computer generated records mirroring the real-life information submitted for applications such as opening a credit card, a bank account or a cell phone line. It covers 10 fields including application date, applicant’s name, social security number, address, zip code, date of birth, phone number, and a label specifying if the application is a fraud. The dataset is shared by Professor Stephen Coggeshall in February 2020.

1. Summary Table

All 10 fields in this dataset are categorical variables and are fully populated. Key statistics of these fields are summarized in the following table.

Table 7.1.1: Summary Statistics of Fields

Field Name	Data Type	# Records w/ Vales	% Populated	# Unique Values	Most Common Vale
record	integer	1,000,000	100	1,000,000	n/a**
date	integer*	1,000,000	100	365	20160816
ssn	integer	1,000,000	100	835,819	999999999
firstname	text	1,000,000	100	78,136	EAMSTRMT
lastname	text	1,000,000	100	177,001	ERJSAXA
address	text	1,000,000	100	828,774	123 MAIN ST
zip5	integer	1,000,000	100	26,370	68138
dob	integer*	1,000,000	100	42,673	19070626
homephone	integer	1,000,000	100	28,244	9999999999
fraud_label	integer	1,000,000	100	2	0

* The *date* and *dob* fields should be of the datetime data type but are stored as integers in the dataset.

** All values in the *record* field are unique, thus no one value is more common than others.

2. Field Description and Distribution

2.1. record

Categorical. Unique integer label for each record.

2.2. date

Categorical and ordinal. The date on which the application was submitted. Takes 365 values that ranges from 2016/01/01 to 2016/12/31. There are records on every day of year 2016 except 2016/02/29. Daily application counts are plotted in Figure 7.2.2.1. August 16, 2016 has the most daily applications of 2,877 counts.

Figure 7.2.2.1: Daily Applications

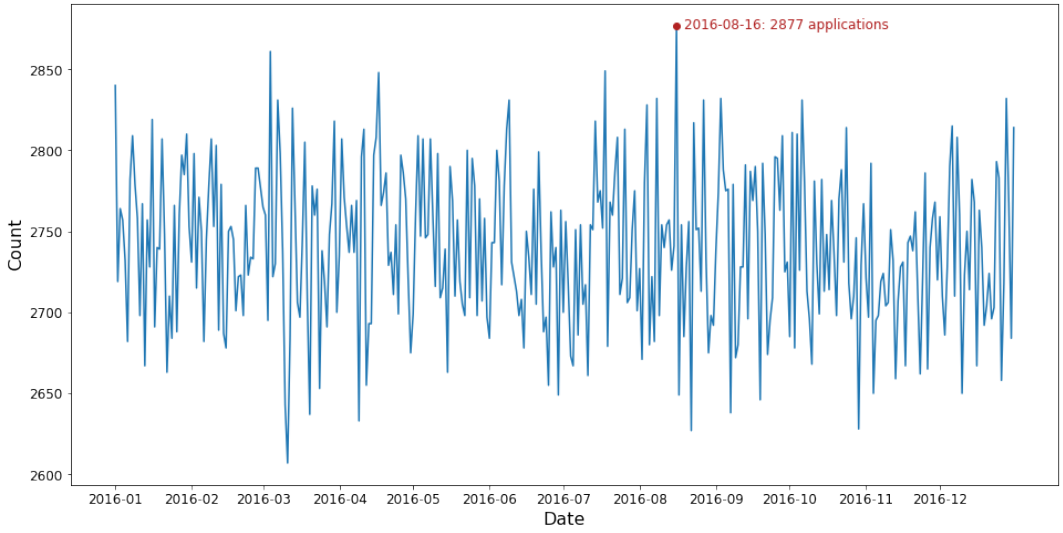
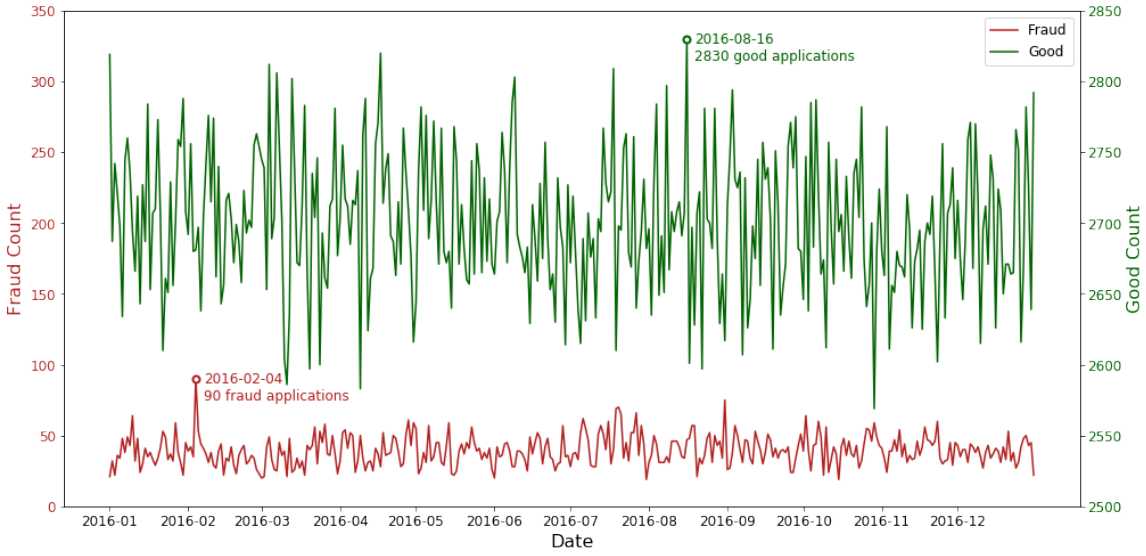


Figure 7.2.2.2 below breaks down the daily applications by their fraud label and plot the counts of the two categories against each other. February 4, 2016 has the most daily fraud applications of 90 counts.

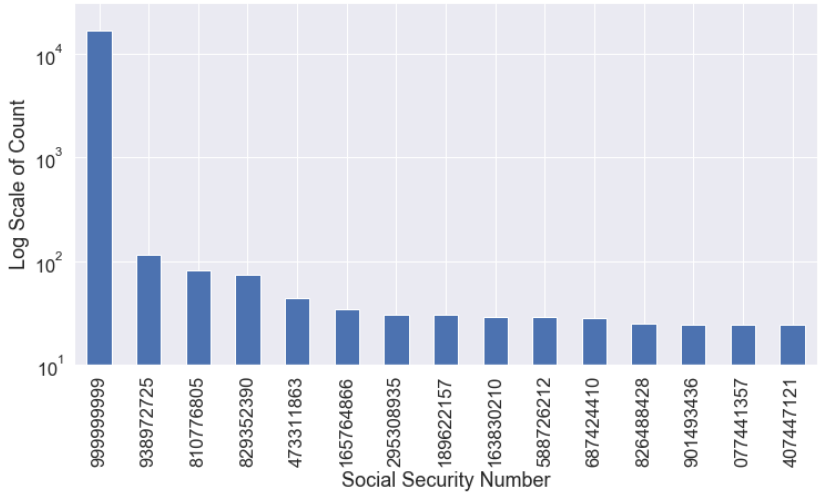
Figure 7.2.2.2: Daily Applications (Fraud vs. Good)



2.3. ssn

Categorical. The social security number of the applicant. Value “999999999” occurred the most in this field for 16,935 times (see Figure 7.2.3 below).

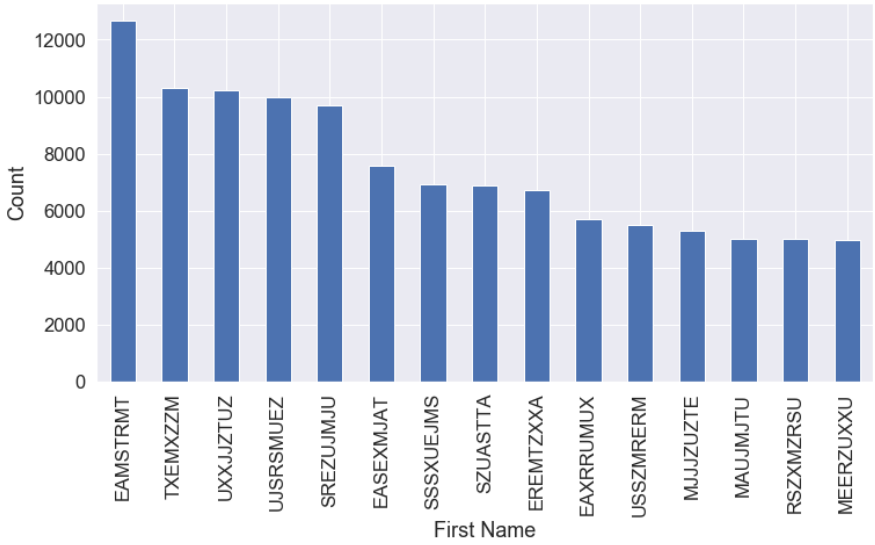
Figure 7.2.3: Distribution of the *ssn* Field
(Top 15 Most Common Values)



2.4. firstname

Categorical. The first name of the applicant. Name “EAMSTRMT” occurred the most for 12,658 times (see Figure 7.2.4 below).

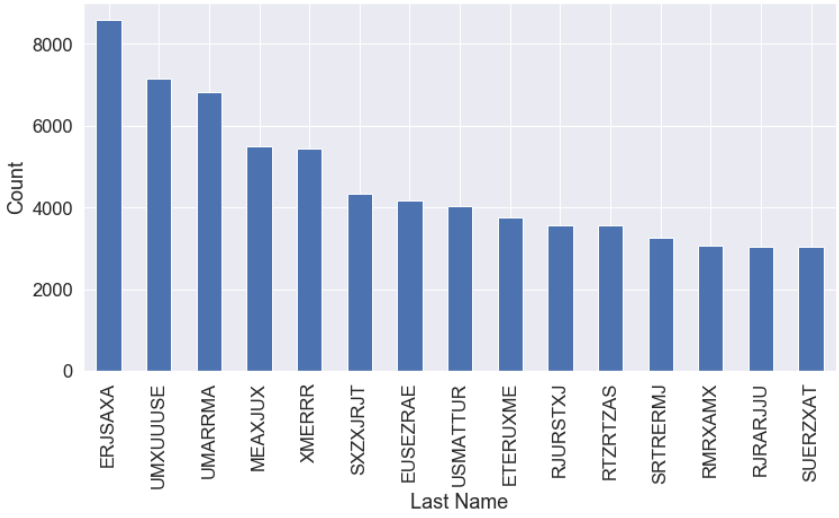
Figure 7.2.4: Distribution of the *firstname* Field
Top 15 Most Common Values



2.5. lastname

Categorical. The last name of the applicant. Name “ERJSAXA” occurred the most for 8,580 times (see Figure 7.2.5 below).

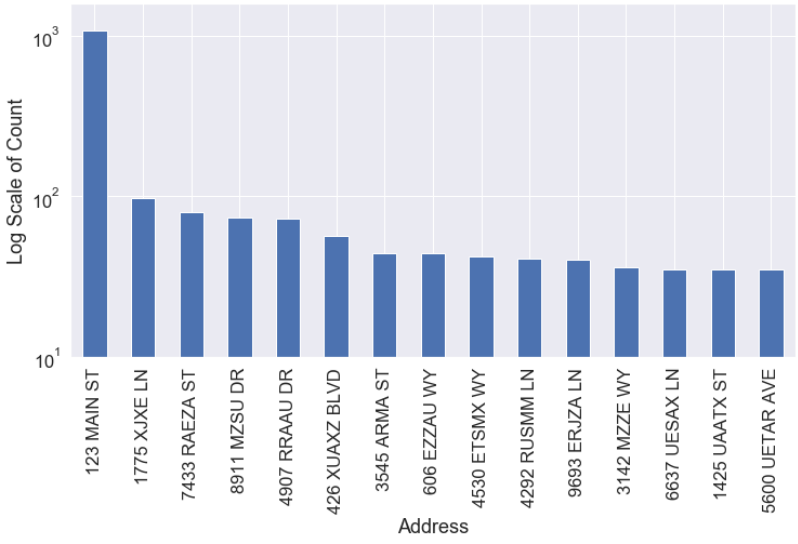
Figure 7.2.5: Distribution of the *lastname* Field
Top 15 Most Common Values



2.6. address

Categorical. The address of the applicant. Address “123 MAIN STREET” occurred the most for 1,079 times (see Figure 7.2.6 below).

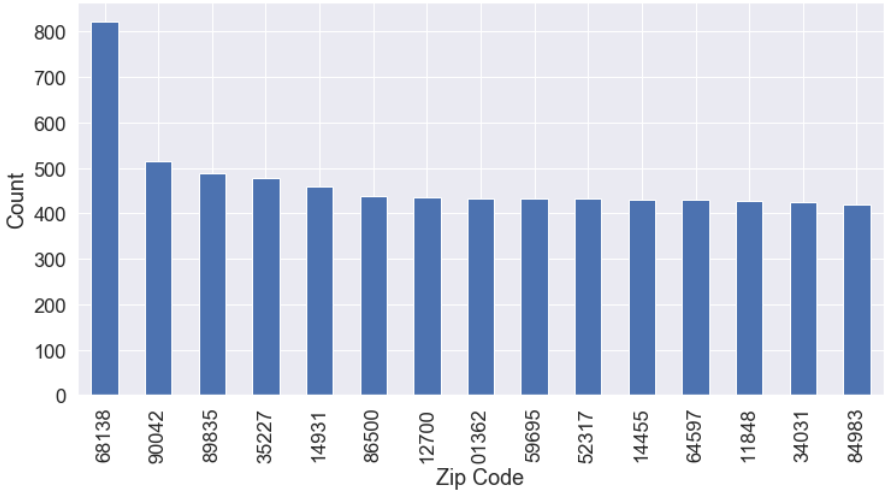
Figure 7.2.6: Distribution of the *address* Field
Top 15 Most Common Values



2.7. zip5

Categorical. The 5-digit zip code of the applicant. Zip code “68138” occurred the most for 823 times (see Figure 7.2.7 below).

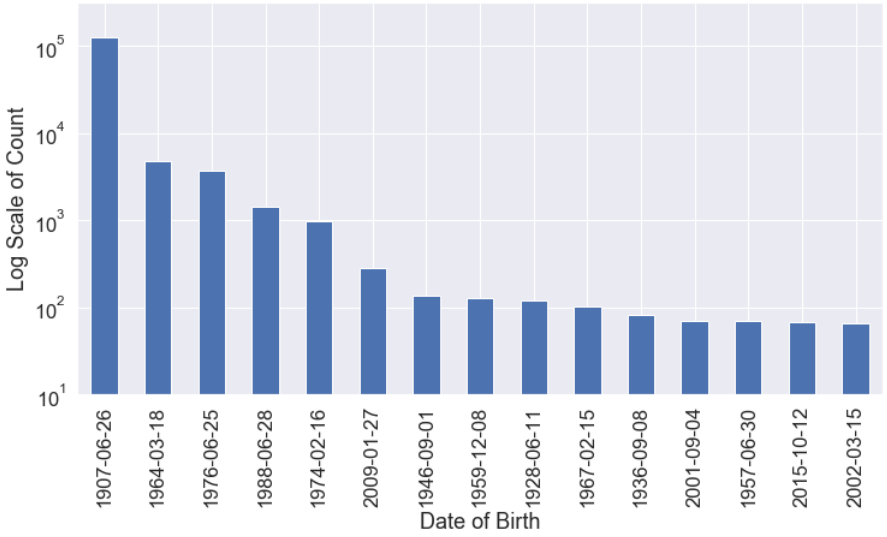
Figure 7.2.7: Distribution of the *zip5* Field
Top 15 Most Common Values



2.8. dob

Categorical and ordinal. The date of birth of the applicant. Date “1907/06/26” occurred the most for 126,568 times (see Figure 7.2.8 below).

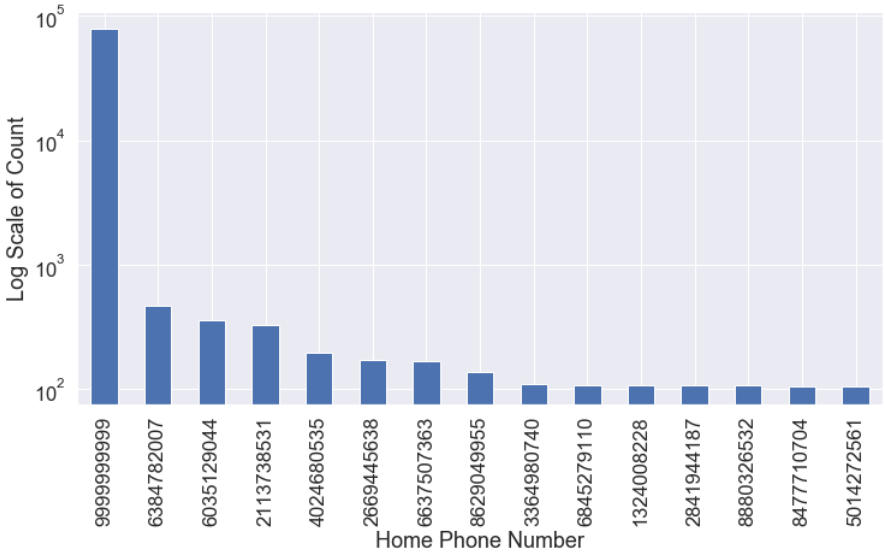
Figure 7.2.8: Distribution of the *dob* Field
Top 15 Most Common Values



2.9. homephone

Categorical. The home phone number of the applicant. Number “9999999999” occurred the most for 78,512 times (see Figure 7.2.9 below).

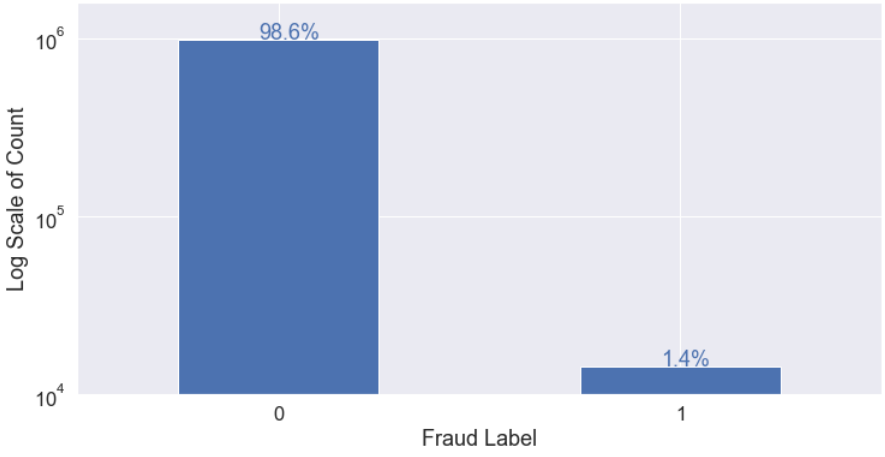
Figure 7.2.9: Distribution of the *homephone* Field
Top 15 Most Common Values



2.10. fraud_label

Categorical. Takes two values: 0 or 1. Value 1 means the record is a fraud, while value 0 means the record is not a fraud. A total of 14,393 (or 1.4%) records are labeled as fraud (see Figure 7.2.10 below).

Figure 7.2.10: Distribution of the *fraud_label* Field



Appendix B: 407 Candidate Variables

1. Weekday Risk: 1 total variable

Weekday_risk: average likelihood of fraud on a given day of the week

2. Velocity Variables: 156 total variables

No.	Variable Name	Description
1	addr_lag0_count	# appearance of the given **full address** in the past 0, 1, 3, 7, 14, 30 days
2	addr_lag1_count	
3	addr_lag3_count	
4	addr_lag7_count	
5	addr_lag14_count	
6	addr_lag30_count	
7	addr-homephone_lag0_count	# appearance of the given **full address + home phone** combination in the past 0, 1, 3, 7, 14, 30 days
8	addr-homephone_lag1_count	
9	addr-homephone_lag3_count	
10	addr-homephone_lag7_count	
11	addr-homephone_lag14_count	
12	addr-homephone_lag30_count	
13	address_lag0_count	# appearance of the given **street address** in the past 0, 1, 3, 7, 14, 30 days
14	address_lag1_count	
15	address_lag3_count	
16	address_lag7_count	
17	address_lag14_count	
18	address_lag30_count	
19	dob_lag0_count	# appearance of the given **date of birth (DOB)** in the past 0, 1, 3, 7, 14, 30 days
20	dob_lag1_count	
21	dob_lag3_count	
22	dob_lag7_count	
23	dob_lag14_count	
24	dob_lag30_count	

No.	Variable Name	Description
25	dob-addr_lag0_count	# appearance of the given **DOB + full address** combination in the past 0, 1, 3, 7, 14, 30 days
26	dob-addr_lag1_count	
27	dob-addr_lag3_count	
28	dob-addr_lag7_count	
29	dob-addr_lag14_count	
30	dob-addr_lag30_count	
31	dob-addr-homephone_lag0_count	# appearance of the given **DOB + full address + home phone** combination in the past 0, 1, 3, 7, 14, 30 days
32	dob-addr-homephone_lag1_count	
33	dob-addr-homephone_lag3_count	
34	dob-addr-homephone_lag7_count	
35	dob-addr-homephone_lag14_count	
36	dob-addr-homephone_lag30_count	
37	dob-homephone_lag0_count	# appearance of the given **DOB + home phone** combination in the past 0, 1, 3, 7, 14, 30 days
38	dob-homephone_lag1_count	
39	dob-homephone_lag3_count	
40	dob-homephone_lag7_count	
41	dob-homephone_lag14_count	
42	dob-homephone_lag30_count	
43	homephone_lag0_count	# appearance of the given **home phone** in the past 0, 1, 3, 7, 14, 30 days
44	homephone_lag1_count	
45	homephone_lag3_count	
46	homephone_lag7_count	
47	homephone_lag14_count	
48	homephone_lag30_count	
49	name_lag0_count	# appearance of the given **full name** in the past 0, 1, 3, 7, 14, 30 days
50	name_lag1_count	
51	name_lag3_count	
52	name_lag7_count	
53	name_lag14_count	
54	name_lag30_count	

No.	Variable Name	Description
55	name-addr_lag0_count	# appearance of the given **full name + full address** combination in the past 0, 1, 3, 7, 14, 30 days
56	name-addr_lag1_count	
57	name-addr_lag3_count	
58	name-addr_lag7_count	
59	name-addr_lag14_count	
60	name-addr_lag30_count	
61	name-addr-homephone_lag0_count	# appearance of the given **full name + full address + home phone** combination in the past 0, 1, 3, 7, 14, 30 days
62	name-addr-homephone_lag1_count	
63	name-addr-homephone_lag3_count	
64	name-addr-homephone_lag7_count	
65	name-addr-homephone_lag14_count	
66	name-addr-homephone_lag30_count	
67	name-dob_lag0_count	# appearance of the given **full name + DOB** combination in the past 0, 1, 3, 7, 14, 30 days
68	name-dob_lag1_count	
69	name-dob_lag3_count	
70	name-dob_lag7_count	
71	name-dob_lag14_count	
72	name-dob_lag30_count	
73	name-dob-addr_lag0_count	# appearance of the given **full name + DOB+ full address** combination in the past 0, 1, 3, 7, 14, 30 days
74	name-dob-addr_lag1_count	
75	name-dob-addr_lag3_count	
76	name-dob-addr_lag7_count	
77	name-dob-addr_lag14_count	
78	name-dob-addr_lag30_count	
79	name-dob-addr-homephone_lag0_count	# appearance of the given **full name + DOB + full address + home phone** combination in the past 0, 1, 3, 7, 14, 30 days
80	name-dob-addr-homephone_lag1_count	
81	name-dob-addr-homephone_lag3_count	
82	name-dob-addr-homephone_lag7_count	
83	name-dob-addr-homephone_lag14_count	
84	name-dob-addr-homephone_lag30_count	

No.	Variable Name	Description
85	name-dob-homephone_lag0_count	# appearance of the given **full name + DOB + home phone** combination in the past 0, 1, 3, 7, 14, 30 days
86	name-dob-homephone_lag1_count	
87	name-dob-homephone_lag3_count	
88	name-dob-homephone_lag7_count	
89	name-dob-homephone_lag14_count	
90	name-dob-homephone_lag30_count	
91	name-homephone_lag0_count	# appearance of the given **full name + home phone** combination in the past 0, 1, 3, 7, 14, 30 days
92	name-homephone_lag1_count	
93	name-homephone_lag3_count	
94	name-homephone_lag7_count	
95	name-homephone_lag14_count	
96	name-homephone_lag30_count	
97	ssn_lag0_count	# appearance of the given **Social Security Number (SSN)** in the past 0, 1, 3, 7, 14, 30 days
98	ssn_lag1_count	
99	ssn_lag3_count	
100	ssn_lag7_count	
101	ssn_lag14_count	
102	ssn_lag30_count	
103	ssn-addr_lag0_count	# appearance of the given **SSN + full address** combination in the past 0, 1, 3, 7, 14, 30 days
104	ssn-addr_lag1_count	
105	ssn-addr_lag3_count	
106	ssn-addr_lag7_count	
107	ssn-addr_lag14_count	
108	ssn-addr_lag30_count	
109	ssn-address_lag0_count	# appearance of the given **SSN + street address** combination in the past 0, 1, 3, 7, 14, 30 days
110	ssn-address_lag1_count	
111	ssn-address_lag3_count	
112	ssn-address_lag7_count	
113	ssn-address_lag14_count	
114	ssn-address_lag30_count	

No.	Variable Name	Description
115	ssn-dob_lag0_count	# appearance of the given **SSN + DOB** combination in the past 0, 1, 3, 7, 14, 30 days
116	ssn-dob_lag1_count	
117	ssn-dob_lag3_count	
118	ssn-dob_lag7_count	
119	ssn-dob_lag14_count	
120	ssn-dob_lag30_count	
121	ssn-firstname_lag0_count	# appearance of the given **SSN + first name** combination in the past 0, 1, 3, 7, 14, 30 days
122	ssn-firstname_lag1_count	
123	ssn-firstname_lag3_count	
124	ssn-firstname_lag7_count	
125	ssn-firstname_lag14_count	
126	ssn-firstname_lag30_count	
127	ssn-homephone_lag0_count	# appearance of the given **SSN + home phone** combination in the past 0, 1, 3, 7, 14, 30 days
128	ssn-homephone_lag1_count	
129	ssn-homephone_lag3_count	
130	ssn-homephone_lag7_count	
131	ssn-homephone_lag14_count	
132	ssn-homephone_lag30_count	
133	ssn-lastname_lag0_count	# appearance of the given **SSN + last name** combination in the past 0, 1, 3, 7, 14, 30 days
134	ssn-lastname_lag1_count	
135	ssn-lastname_lag3_count	
136	ssn-lastname_lag7_count	
137	ssn-lastname_lag14_count	
138	ssn-lastname_lag30_count	
139	ssn-name_lag0_count	# appearance of the given **SSN + full name** combination in the past 0, 1, 3, 7, 14, 30 days
140	ssn-name_lag1_count	
141	ssn-name_lag3_count	
142	ssn-name_lag7_count	
143	ssn-name_lag14_count	
144	ssn-name_lag30_count	

No.	Variable Name	Description
145	ssn-name-dob_lag0_count	# appearance of the given **SSN + full name + DOB** combination in the past 0, 1, 3, 7, 14, 30 days
146	ssn-name-dob_lag1_count	
147	ssn-name-dob_lag3_count	
148	ssn-name-dob_lag7_count	
149	ssn-name-dob_lag14_count	
150	ssn-name-dob_lag30_count	
151	ssn-zip5_lag0_count	# appearance of the given **SSN + zip code** combination in the past 0, 1, 3, 7, 14, 30 days
152	ssn-zip5_lag1_count	
153	ssn-zip5_lag3_count	
154	ssn-zip5_lag7_count	
155	ssn-zip5_lag14_count	
156	ssn-zip5_lag30_count	

3. Relative Velocity Variables

No	Variable Name	Description
1	addr_lag1_lag3_avg	# appearance of the given **full address** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
2	addr_lag1_lag7_avg	
3	addr_lag1_lag14_avg	
4	addr_lag1_lag30_avg	
5	addr-homephone_lag1_lag3_avg	# appearance of the given **full address + home phone** combination in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
6	addr-homephone_lag1_lag7_avg	
7	addr-homephone_lag1_lag14_avg	
8	addr-homephone_lag1_lag30_avg	
9	address_lag1_lag3_avg	# appearance of the given **street address** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
10	address_lag1_lag7_avg	
11	address_lag1_lag14_avg	
12	address_lag1_lag30_avg	
13	dob_lag1_lag3_avg	# appearance of the given **DOB** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
14	dob_lag1_lag7_avg	
15	dob_lag1_lag14_avg	
16	dob_lag1_lag30_avg	

No	Variable Name	Description
17	dob-addr_lag1_lag3_avg	# appearance of the given **DOB + full address** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
18	dob-addr_lag1_lag7_avg	
19	dob-addr_lag1_lag14_avg	
20	dob-addr_lag1_lag30_avg	
21	dob-addr-homephone_lag1_lag3_avg	# appearance of the given **DOB + full address + home phone** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
22	dob-addr-homephone_lag1_lag7_avg	
23	dob-addr-homephone_lag1_lag14_avg	
24	dob-addr-homephone_lag1_lag30_avg	
25	dob-homephone_lag1_lag3_avg	# appearance of the given **DOB + home phone** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
26	dob-homephone_lag1_lag7_avg	
27	dob-homephone_lag1_lag14_avg	
28	dob-homephone_lag1_lag30_avg	
29	homephone_lag1_lag3_avg	# appearance of the given **home phone** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
30	homephone_lag1_lag7_avg	
31	homephone_lag1_lag14_avg	
32	homephone_lag1_lag30_avg	
33	name_lag1_lag3_avg	# appearance of the given **full name** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
34	name_lag1_lag7_avg	
35	name_lag1_lag14_avg	
36	name_lag1_lag30_avg	
37	name-addr_lag1_lag3_avg	# appearance of the given **full name + full address** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
38	name-addr_lag1_lag7_avg	
39	name-addr_lag1_lag14_avg	
40	name-addr_lag1_lag30_avg	
41	name-addr-homephone_lag1_lag3_avg	# appearance of the given **full name + full address + home phone** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
42	name-addr-homephone_lag1_lag7_avg	
43	name-addr-homephone_lag1_lag14_avg	
44	name-addr-homephone_lag1_lag30_avg	
45	name-dob_lag1_lag3_avg	# appearance of the given **full name + DOB** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
46	name-dob_lag1_lag7_avg	
47	name-dob_lag1_lag14_avg	
48	name-dob_lag1_lag30_avg	

No	Variable Name	Description
49	name-dob-addr_lag1_lag3_avg	# appearance of the given **full name** + **DOB** + **full address** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
50	name-dob-addr_lag1_lag7_avg	
51	name-dob-addr_lag1_lag14_avg	
52	name-dob-addr_lag1_lag30_avg	
53	name-dob-addr-homephone_lag1_lag3_avg	# appearance of the given **full name** + **DOB** + **full address** + **home phone** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
54	name-dob-addr-homephone_lag1_lag7_avg	
55	name-dob-addr-homephone_lag1_lag14_avg	
56	name-dob-addr-homephone_lag1_lag30_avg	
57	name-dob-homephone_lag1_lag3_avg	# appearance of the given **full name** + **DOB** + **home phone** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
58	name-dob-homephone_lag1_lag7_avg	
59	name-dob-homephone_lag1_lag14_avg	
60	name-dob-homephone_lag1_lag30_avg	
61	name-homephone_lag1_lag3_avg	# appearance of the given **full name** + **home phone** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
62	name-homephone_lag1_lag7_avg	
63	name-homephone_lag1_lag14_avg	
64	name-homephone_lag1_lag30_avg	
65	ssn_lag1_lag3_avg	# appearance of the given **SSN** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
66	ssn_lag1_lag7_avg	
67	ssn_lag1_lag14_avg	
68	ssn_lag1_lag30_avg	
69	ssn-addr_lag1_lag3_avg	# appearance of the given **SSN** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
70	ssn-addr_lag1_lag7_avg	
71	ssn-addr_lag1_lag14_avg	
72	ssn-addr_lag1_lag30_avg	
73	ssn-address_lag1_lag3_avg	# appearance of the given **SSN** + **full address** in the past 1 day / # daily appearance of which in the past 3, 7, 14, 30 days
74	ssn-address_lag1_lag7_avg	
75	ssn-address_lag1_lag14_avg	
76	ssn-address_lag1_lag30_avg	

No	Variable Name	Description
77	ssn-dob_lag1_lag3_avg	# appearance of the given **SSN** + **date of birth** in the past 1 day # daily appearance of which in the past 3, 7, 14, 30 days
78	ssn-dob_lag1_lag7_avg	
79	ssn-dob_lag1_lag14_avg	
80	ssn-dob_lag1_lag30_avg	
81	ssn-firstname_lag1_lag3_avg	# appearance of the given **SSN** + **first name** in the past 1 day # daily appearance of which in the past 3, 7, 14, 30 days
82	ssn-firstname_lag1_lag7_avg	
83	ssn-firstname_lag1_lag14_avg	
84	ssn-firstname_lag1_lag30_avg	
85	ssn-homephone_lag1_lag3_avg	# appearance of the given **SSN** + **home phone** in the past 1 day # daily appearance of which in the past 3, 7, 14, 30 days
86	ssn-homephone_lag1_lag7_avg	
87	ssn-homephone_lag1_lag14_avg	
88	ssn-homephone_lag1_lag30_avg	
89	ssn-lastname_lag1_lag3_avg	# appearance of the given **SSN** + **last name** in the past 1 day # daily appearance of which in the past 3, 7, 14, 30 days
90	ssn-lastname_lag1_lag7_avg	
91	ssn-lastname_lag1_lag14_avg	
92	ssn-lastname_lag1_lag30_avg	
93	ssn-name_lag1_lag3_avg	# appearance of the given **SSN** + **full name** in the past 1 day # daily appearance of which in the past 3, 7, 14, 30 days
94	ssn-name_lag1_lag7_avg	
95	ssn-name_lag1_lag14_avg	
96	ssn-name_lag1_lag30_avg	
97	ssn-name-dob_lag1_lag3_avg	# appearance of the given **SSN** + **full name** + **DOB** in the past 1 day # daily appearance of which in the past 3, 7, 14, 30 days
98	ssn-name-dob_lag1_lag7_avg	
99	ssn-name-dob_lag1_lag14_avg	
100	ssn-name-dob_lag1_lag30_avg	
101	ssn-zip5_lag1_lag3_avg	# appearance of the given **SSN** + **zip** in the past 1 day # daily appearance of which in the past 3, 7, 14, 30 days
102	ssn-zip5_lag1_lag7_avg	
103	ssn-zip5_lag1_lag14_avg	
104	ssn-zip5_lag1_lag30_avg	

4. Days Since Variables

No.	Variable Name	Description
1	addr_#days_since	# days since the last appearance of the given **full address**
2	addr-homephone_#days_since	# days since the last appearance of the given **full address** + **home phone** combination
3	address_#days_since	# days since the last appearance of the given **street address**
4	dob_#days_since	# days since the last appearance of the given **DOB**
5	dob-addr_#days_since	# days since the last appearance of the given **DOB** + **full address** combination
6	dob-addr-homephone_#days_since	# days since the last appearance of the given **DOB** + **full address** + **home phone** combination
7	dob-homephone_#days_since	# days since the last appearance of the given **DOB** + **home phone** combination
8	homephone_#days_since	# days since the last appearance of the given **home phone**
9	name_#days_since	# days since the last appearance of the given **full name**
10	name-addr_#days_since	# days since the last appearance of the given **full name** + **full address** combination
11	name-addr-homephone_#days_since	# days since the last appearance of the given **full name** + **full address** + **home phone** combination
12	name-dob_#days_since	# days since the last appearance of the given **full name** + **DOB** combination
13	name-dob-addr_#days_since	# days since the last appearance of the given **full name** + **DOB** + **full address** combination
14	name-dob-addr-homephone_#days_since	# days since the last appearance of the given **full name** + **DOB** + **full address** + **home phone** combination
15	name-dob-homephone_#days_since	# days since the last appearance of the given **full name** + **DOB** + **home phone** combination
16	name-homephone_#days_since	# days since the last appearance of the given **full name** + **home phone** combination
17	ssn_#days_since	# days since the last appearance of the given **SSN**
18	ssn-addr_#days_since	# days since the last appearance of the given **SSN** + **full address** combination
19	ssn-address_#days_since	# days since the last appearance of the given **SSN** + **street address** combination
20	ssn-dob_#days_since	# days since the last appearance of the given **SSN** + **DOB** combination
21	ssn-firstname_#days_since	# days since the last appearance of the given **SSN** + **first name** combination
22	ssn-homephone_#days_since	# days since the last appearance of the given **SSN** + **home phone** combination
23	ssn-lastname_#days_since	# days since the last appearance of the given **SSN** + **last name** combination
24	ssn-name_#days_since	# days since the last appearance of the given **SSN** + **full name** combination

No.	Variable Name	Description
25	ssn-name-dob_#days_since	# days since the last appearance of the given **SSN + full name + DOB** combination
26	ssn-zip5_#days_since	# days since the last appearance of the given **SSN + zip code** combination

5. Cross Entity Velocity Variables

No.	Variable Name	Description
1	addr_lag0_dob	# different DOB appeared alongside the given full address in the past 0, 1, 3, 7, 14, 30 days
2	addr_lag1_dob	
3	addr_lag3_dob	
4	addr_lag7_dob	
5	addr_lag14_dob	
6	addr_lag30_dob	
7	addr_lag0_homephone	# different home phone appeared alongside the given full address in the past 0, 1, 3, 7, 14, 30 days
8	addr_lag1_homephone	
9	addr_lag3_homephone	
10	addr_lag7_homephone	
11	addr_lag14_homephone	
12	addr_lag30_homephone	
13	addr_lag0_name	# different full name appeared alongside the given full address in the past 0, 1, 3, 7, 14, 30 days
14	addr_lag1_name	
15	addr_lag3_name	
16	addr_lag7_name	
17	addr_lag14_name	
18	addr_lag30_name	
19	addr_lag0_ssn	# different SSN appeared alongside the given full address in the past 0, 1, 3, 7, 14, 30 days
20	addr_lag1_ssn	
21	addr_lag3_ssn	
22	addr_lag7_ssn	
23	addr_lag14_ssn	
24	addr_lag30_ssn	
25	dob_lag0_addr	

No.	Variable Name	Description
26	dob_lag1_addr	# different full address appeared alongside the given DOB in the past 0, 1, 3, 7, 14, 30 days
27	dob_lag3_addr	
28	dob_lag7_addr	
29	dob_lag14_addr	
30	dob_lag30_addr	
31	dob_lag0_homephone	# different home phone appeared alongside the given DOB in the past 0, 1, 3, 7, 14, 30 days
32	dob_lag1_homephone	
33	dob_lag3_homephone	
34	dob_lag7_homephone	
35	dob_lag14_homephone	
36	dob_lag30_homephone	
37	dob_lag0_name	# different full name appeared alongside the given DOB the past 0, 1, 3, 7, 14, 30 days
38	dob_lag1_name	
39	dob_lag3_name	
40	dob_lag7_name	
41	dob_lag14_name	
42	dob_lag30_name	
43	dob_lag0_ssn	# different SSN appeared alongside the given DOB in the past 0, 1, 3, 7, 14, 30 days
44	dob_lag1_ssn	
45	dob_lag3_ssn	
46	dob_lag7_ssn	
47	dob_lag14_ssn	
48	dob_lag30_ssn	
49	homephone_lag0_addr	# different full address appeared alongside the given home phone in the past 0, 1, 3, 7, 14, 30 days
50	homephone_lag1_addr	
51	homephone_lag3_addr	
52	homephone_lag7_addr	
53	homephone_lag14_addr	
54	homephone_lag30_addr	

No.	Variable Name	Description
55	homephone_lag0_dob	# different date of birth appeared alongside the given home phone in the past 0, 1, 3, 7, 14, 30 days
56	homephone_lag1_dob	
57	homephone_lag3_dob	
58	homephone_lag7_dob	
59	homephone_lag14_dob	
60	homephone_lag30_dob	
61	homephone_lag0_name	# different full name appeared alongside the given home phone in the past 0, 1, 3, 7, 14, 30 days
62	homephone_lag1_name	
63	homephone_lag3_name	
64	homephone_lag7_name	
65	homephone_lag14_name	
66	homephone_lag30_name	
67	homephone_lag0_ssn	# different SSN appeared alongside the given home phone in the past 0, 1, 3, 7, 14, 30 days
68	homephone_lag1_ssn	
69	homephone_lag3_ssn	
70	homephone_lag7_ssn	
71	homephone_lag14_ssn	
72	homephone_lag30_ssn	
73	name_lag0_addr	# different full address appeared alongside the given full name in the past 0, 1, 3, 7, 14, 30 days
74	name_lag1_addr	
75	name_lag3_addr	
76	name_lag7_addr	
77	name_lag14_addr	
78	name_lag30_addr	
79	name_lag0_dob	# different DOB appeared alongside the given full name in the past 0, 1, 3, 7, 14, 30 days
80	name_lag1_dob	
81	name_lag3_dob	
82	name_lag7_dob	
83	name_lag14_dob	
84	name_lag30_dob	

No.	Variable Name	Description
85	name_lag0_homephone	# different home phone appeared alongside the given full name in the past 0, 1, 3, 7, 14, 30 days
86	name_lag1_homephone	
87	name_lag3_homephone	
88	name_lag7_homephone	
89	name_lag14_homephone	
90	name_lag30_homephone	
91	name_lag0_ssn	# different SSN appeared alongside the given full name in the past 0, 1, 3, 7, 14, 30 days
92	name_lag1_ssn	
93	name_lag3_ssn	
94	name_lag7_ssn	
95	name_lag14_ssn	
96	name_lag30_ssn	
97	ssn_lag0_addr	# different full address appeared alongside the given SSN in the past 0, 1, 3, 7, 14, 30 days
98	ssn_lag1_addr	
99	ssn_lag3_addr	
100	ssn_lag7_addr	
101	ssn_lag14_addr	
102	ssn_lag30_addr	
103	ssn_lag0_dob	# different DOB appeared alongside the given SSN in the past 0, 1, 3, 7, 14, 30 days
104	ssn_lag1_dob	
105	ssn_lag3_dob	
106	ssn_lag7_dob	
107	ssn_lag14_dob	
108	ssn_lag30_dob	
109	ssn_lag0_homephone	# different home phone appeared alongside the given SSN in the past 0, 1, 3, 7, 14, 30 days
110	ssn_lag1_homephone	
111	ssn_lag3_homephone	
112	ssn_lag7_homephone	
113	ssn_lag14_homephone	
114	ssn_lag30_homephone	

No.	Variable Name	Description
115	ssn_lag0_name	# different full name appeared alongside the given SSN in the past 0, 1, 3, 7, 14, 30 days
116	ssn_lag1_name	
117	ssn_lag3_name	
118	ssn_lag7_name	
119	ssn_lag14_name	
120	ssn_lag30_name	

Appendix C: 80 Variables Selected after Filtering

	Candidate Variable	KS	FDR	KS Rank	FDR Rank	Average Rank
1	addr_lag30_count	0.3320	0.3563	407	408	407.5
2	address_lag30_count	0.3327	0.3536	408	407	407.5
3	addr_#days_since	0.3235	0.3484	405	406	405.5
5	address_#days_since	0.3246	0.3469	406	405	405.5
6	address_lag14_count	0.3223	0.3462	404	404	404
7	addr_lag14_count	0.3218	0.3429	403	403	403
8	address_lag7_count	0.3014	0.3217	402	402	402
9	addr_lag7_count	0.3014	0.3200	401	401	401
10	addr_lag30_ssn	0.2818	0.2996	400	400	400
11	addr_lag30_dob	0.2787	0.2969	398	399	398.5
12	address_lag3_count	0.2784	0.2964	396	398	397
13	addr_lag30_name	0.2796	0.2960	399	397	398
14	addr_lag3_count	0.2785	0.2960	397	397	397
15	addr_lag14_ssn	0.2764	0.2938	395	395	395
16	addr_lag14_dob	0.2746	0.2931	393	394	393.5
17	addr_lag14_name	0.2752	0.2925	394	393	393.5
18	addr_lag7_ssn	0.2730	0.2902	392	392	392
19	addr_lag7_dob	0.2720	0.2897	390	391	390.5
20	addr_lag7_name	0.2721	0.2892	391	390	390.5
21	addr_lag3_ssn	0.2639	0.2812	389	389	389
22	addr_lag3_dob	0.2635	0.2810	388	388	388
23	addr_lag3_name	0.2634	0.2808	387	387	387
24	address_lag1_count	0.2493	0.2679	386	386	386
25	addr_lag1_count	0.2491	0.2674	385	385	385
26	addr_lag1_dob	0.2435	0.2614	384	384	384
27	addr_lag1_ssn	0.2434	0.2613	383	383	383
28	addr_lag1_name	0.2434	0.2612	382	382	382
29	addr-homephone_lag30_count	0.2290	0.2554	381	381	381
30	ssn-dob_lag30_count	0.2285	0.2547	380	380	380
31	name-dob_lag30_count	0.2276	0.2534	379	379	379
32	ssn_lag30_count	0.2270	0.2532	378	378	378
33	ssn-lastname_lag30_count	0.2260	0.2524	374	377	375.5

	Candidate Variable	KS	FDR	KS Rank	FDR Rank	Average Rank
34	ssn-firstname_lag30_count	0.2261	0.2521	375	376	375.5
35	ssn-name-dob_lag30_count	0.2262	0.2520	377	375	376
36	ssn-name_lag30_count	0.2250	0.2511	373	374	373.5
37	addr-homephone_#days_since	0.2262	0.2480	376	373	374.5
38	ssn-dob_#days_since	0.2196	0.2438	372	372	372
39	name-dob_#days_since	0.2193	0.2430	371	371	371
40	ssn_#days_since	0.2185	0.2419	369	370	369.5
41	addr-homephone_lag14_count	0.2189	0.2417	370	369	369.5
42	ssn-lastname_#days_since	0.2175	0.2413	365	368	366.5
43	ssn-firstname_#days_since	0.2178	0.2412	367	367	367
44	ssn-name-dob_#days_since	0.2176	0.2412	366	367	366.5
45	ssn-name_#days_since	0.2167	0.2407	364	365	364.5
46	ssn-dob_lag14_count	0.2149	0.2399	361	364	362.5
47	name_lag30_count	0.2139	0.2396	359	363	361
48	name-dob_lag14_count	0.2153	0.2393	362	362	362
49	ssn-firstname_lag14_count	0.2138	0.2389	358	361	359.5
50	ssn-lastname_lag14_count	0.2134	0.2386	356	360	358
51	ssn_lag14_count	0.2144	0.2379	360	359	359.5
52	ssn-name-dob_lag14_count	0.2135	0.2376	357	358	357.5
53	address_lag1_lag14_avg	0.2108	0.2374	354	357	355.5
54	ssn-name_lag14_count	0.2130	0.2364	355	356	355.5
55	addr_lag1_lag14_avg	0.2091	0.2352	353	355	354
56	addr-homephone_lag7_count	0.1998	0.2217	350	354	352
57	ssn-dob_lag7_count	0.1931	0.2184	346	353	349.5
58	name-dob_lag7_count	0.1941	0.2177	347	352	349.5
59	ssn_lag7_count	0.1930	0.2170	345	351	348
60	ssn-firstname_lag7_count	0.1927	0.2170	344	351	347.5
61	ssn-lastname_lag7_count	0.1926	0.2169	343	349	346
62	ssn-name-dob_lag7_count	0.1925	0.2168	342	348	345
63	ssn-name_lag7_count	0.1924	0.2166	341	347	344
64	name_lag14_count	0.2045	0.2103	351	346	348.5
65	address_lag1_lag7_avg	0.1851	0.2099	329	345	337
66	name_#days_since	0.2053	0.2098	352	344	348

	Candidate Variable	KS	FDR	KS Rank	FDR Rank	Average Rank
67	addr_lag1_lag7_avg	0.1852	0.2097	330	343	336.5
68	name_lag7_count	0.1885	0.2087	336	342	339
69	homephone_lag7_count	0.1942	0.2080	348	341	344.5
70	addr_lag0_count	0.1868	0.2076	334	340	337
71	address_lag0_count	0.1868	0.2071	335	339	337
72	homephone_lag3_count	0.1949	0.2066	349	338	343.5
73	addr_lag0_dob	0.1850	0.2058	327	337	332
74	addr_lag0_ssn	0.1850	0.2057	328	336	332
75	addr_lag0_name	0.1850	0.2056	326	335	330.5
76	homephone_lag7_ssn	0.1865	0.2049	333	334	333.5
77	homephone_lag7_name	0.1864	0.2049	332	333	332.5
78	homephone_lag7_dob	0.1863	0.2045	331	332	331.5
79	homephone_lag3_name	0.1900	0.2032	338	331	334.5
80	homephone_lag3_dob	0.1901	0.2016	339	330	334.5

Appendix D: Models and Hyperparameter Tuning

1. Logistic Regression (LR)

In an LR model, data is fit into a logistic function to predict the target-dependent variable and the output of the model is used to infer how confident can predicted value be actual value given an input X.

Package: sklearn.linear_model.LogisticRegression

Table 8.1: Essential Parameters in Logistic Regression

parameter name	values	description
penalty	l1, l2, elasticnet (three different norms of penalties)	● Used to specify the norm used in the penalization. ● l1 is lasso regularization, l2 is ridge regularization, ● elasticnet is the combination of l1 and l2 regularization
C	the strength of logistics regression	● C = 1/λ, default=1.0 ● Inverse of regularization strength, C= 1/λ, λ is the regularization parameters ● a smaller value means stronger regularization and heavy penalty on model complexity
solver	'lbfgs', 'liblinear', 'saga'	Algorithm to use in the optimization problem. Only 'saga' supports "ElasticNet" (we have to use sage when penalty = "elasticnet")
l1_ratio - float, default=None		● only applicable when penalty = 'elasticnet' ● l1_ratio=0 is equivalent to using penalty='l2', while setting l1_ratio=1 is equivalent to using penalty='l1'. ● For 0 < l1_ratio <1, the penalty is a combination of L1 and L2.

Steps to tune the parameters:

- The logistics regression models with default parameters were trained.
- Based on the performance of logistic regression (default parameters), three forms of regularization (L1, L2, and ElasticNet) were applied separately to see which regularization can have the best performance on FDR. It was found that L2 could fit a better model with the highest average FDR on OOT dataset.
- Given the best model created in step 3, the strength of regularization was fine-tuned to train a better model. Specifically, both 0.1 and 10 were used in parameter "C" to train a new logistics regression model with L2 regularization. Indeed, when C is equal to 0.1, the performance is better than that when C is equal to 10, because the strength of regularization is stronger, and in turn, the complexity of the model should be lower. However, this improvement was not significant at this point (slightly up from 52.63% to 52.68%).

2. Single Decision Tree (DT)

The model is a flowchart-like structure consisting of nodes and branches. Each time a split on data is performed and generates more branches until almost all data belong to the same class and further splits are no longer possible. Typically results in the lowest impurity should be chosen.

Package: sklearn.tree.DecisionTreeClassifier

Table 8.2: Essential Parameters in Single Decision Tree

parameter name	values	description
max_depth	10, 15, 20, ... 45, None	The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
min_samples_split	100, 150, 200, 250, ... 1500	The minimum number of samples required to split an internal node
min_samples_leaf	1, 2, 3, 4, ... 110	The minimum number of samples required to be at a leaf node

Steps to tune the parameters:

- Firstly using the random search method to secure candidate parameter combinations
- Using the grid search method to narrow down the scope of combinations

3. Boosted Tree (BT)

The boosted tree is a method of training a series of weak trees to result in a stronger model, where each weak learner is trained to predict the residual error of the current sum. Based on a suitable loss function, the boosting process is optimized by iteratively choosing a function until a single stronger learner is obtained.

Package: sklearn.ensemble.GradientBoostingClassifier

Table 8.3: Essential Parameters in Boosted Tree

parameter name	values	description
loss	deviance', 'exponential'	Loss function to be optimized
learning_rate	0.001, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2	It shrinks the contribution of each tree by learning_rate; Used to regularize Random Forest
n_estimators	60, 80, 100, 120, 190, 200, 210, 240, 400, 600, 800, 1000, 1200, 4000	The number of boosting stages to perform;
max_depth	3, 4, 5, 10, 15, None	The maximum depth of the tree

min_samples_split	2, 500, 900, 1000, 1100, 1200, 1500, 1800	The minimum number of samples required to split an internal node
min_samples_leaf	1, 20, 30, 50	The minimum number of samples required to be at a leaf node
max_features	5, 10, 15, 25	The number of features to consider when looking for the best split
subsample	0.6, 0.7, 0.8	The fraction of samples to be used for fitting the individual base learners

Steps to tune the parameters:

- Given a 'learning_rate' large enough as 0.1 or 0.2, decide the best 'n_estimators'
- Decide the best parameter combination using the random search method
- Narrow down the 'learning_rate' gradually and increase 'n_estimators' using the grid search method

4. Random Forest (RF)

The random forest is an algorithm ensembling many strong decision trees together. The model uses bagging and feature randomness when building each individual tree to try to create an uncorrelated 'forest' of trees whose prediction by committee is more accurate than that of any individual tree.

Package: sklearn.ensemble.RandomForestClassifier

Table 8.4: Essential Parameters in Random Forest

parameter name	values	description
n_estimators	20, 30, 50, 80, 100	The number of trees in the forest; Used to regularize Random Forest
criterion	'gini', 'entropy'	The function to measure the quality of a split
max_depth	10, 15, 20, None	The maximum depth of the tree; Used to regularize Random Forest
min_samples_split	2, 300, 400, 500, 800, 1000, 1500	The minimum number of samples required to split an internal node
min_samples_leaf	1, 30, 50, 100	The minimum number of samples required to be at a leaf node
max_features	5, 10, 15, 25	The number of features to consider when looking for the best split; Used to regularize Random Forest
bootstrap	True, False	Whether bootstrap samples are used when building trees

Steps to tune the parameters:

- Iterate through candidate values to secure the best ‘bootstrap’, ‘criterion’ as gini
- Use the random search method on a large scale and the grid search method on a small scale to find the best parameters

5. K Nearest Neighbors (KNN)

Based on the assumption that similar things exist in close proximity, the KNN algorithm assigns weights to the contributions of the neighbors both for classification so that the nearer neighbors contribute, the more to the averages are than the distant ones.

Package: sklearn.neighbors.KNeighborsClassifier

Table 8.5: Essential Parameters in K Nearest Neighbors

parameter name	values	description
n_neighbors	3, 4, 5, ..., 15	Number of neighbors to use by default for kneighbors queries.
weights	‘uniform', 'distance'	weight function used in prediction. Possible values: ‘uniform’ : uniform weights. All points in each neighborhood are weighted equally. ‘distance’ : weight points by the inverse of their distance. In this case, closer neighbors of a query point will have a greater influence than neighbors which are further away. [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

Steps to tune the parameters:

- Iterate ‘n_neighbors’ from three to 15 to find the elbow point, using ‘weights’ = ‘uniform’, and choose three best ‘n_neighbors’
- Given the best three ‘n_neighbors’, try all six combinations of ‘n_neighbors’ and ‘weights’ to find the best combination of parameters

6. Neural Network (NN)

A neural net consists of an input layer, some number of hidden layers and an output layer. In NN implementations, the output of each neuron is computed by some non-linear function of the sum of its inputs. Neurons and the connections, which are called edges, typically have a weight that adjusts as learning proceeds. Each node in the hidden layer receives weighted signals from all nodes in the previous layer and does a transform on this linear combination of signals.

Package: sklearn.neural_network.MLPClassifier

Table 8.6: Essential Parameters in Neural Network

parameter name	values	description
hidden_layer_sizes	1, 5, 10, 50, 100	The number of neurons in the ith hidden layer.
activation	'relu', ‘tanh’ 'logistic', ‘identity’	Activation function for the hidden layer.
solver	'sgd', 'adam'	The solver for weight optimization.
alpha	0.00005, np.arange (0, 0.1, 0.0001)	L2 penalty (regularization term) parameter.
learning_rate	'constant', 'adaptive' ‘invscaling’	Only used when solver='sgd'. Learning rate schedule for weight updates.
learning_rate_init	0.001, 0.003, 0.005,0.007, 0.01, 0.02	Only used when solver=’sgd’ or ‘adam’. The initial learning rate used
momentum	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1	Only used when solver=’sgd’. Momentum for gradient descent update.
nesterovs_momentum	True, False	Only used when solver=’sgd’ and momentum > 0. Whether to use Nesterov’s momentum.
max_iter	50, 100, 150, 200, 250, 300, 400, 500	Maximum number of iterations. The solver iterates until convergence (determined by ‘tol’) or this number of iterations

Steps to tune the parameters:

- Deciding on the network topology
- Adjust ‘learning_rate’. Start with 'sgd’, a ‘learning_rate’ of 0.01 and a ‘momentum’ of 0.8
- Plot the accuracy and loss function as a function of epochs for the training and test sets to see how the network performed
- Decide on the nodes, layer, activation function, alpha using RandomSearch and refine the search using GridSearch at promising regions

7. Support Vector Classification (SVC)

The support vector machine (SVM) searches a hyperplane in an N-dimensional space that distinctly classifies the data points, which can be used for both classification and regression problems. Given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples in the dataset. Based on SVM, SVC is applying SVM to classification.

Package: sklearn.svm.SVC

Table 8.7: Essential Parameters in Support Vector Classification

parameter name	values	description
kernel	‘linear'	
C	1	Regularization parameter; Smaller C -> stronger regularization

Steps to tune the parameters:

- Set ‘kernel’ = ‘linear’ to speed up processing