Application (Identity) Fraud Analysis

By

Santhoshini Jayachandran

8218065467

Table of Contents

Description of the Data

Dataset Name: Applications Data

Dataset Description: The data is about Application/ Identity Fraud. The data is synthetic application data indicating PII of individual from US applications such as applications for credit card The dataset also contains a field representing the date on which the credit card application was made, as well as Personal Identity Information fields like SSN, name, address, phone number, date of birth, zip code and the fraud label which tells us that the application is fraudulent if the value is 1 and not fraudulent if the value is 0.

Fields: 10

Records: 1,000,000

Time Period: 1st January 2017 - 31st December 2017

1. Summary Table

I. Numerical Table

Field Name	% Populated	Min	Max	Mean	Stdev	%Zero
Date	100.00	2017-01-01	2017-12-31	NA	NA	0.00
Dob	100.00	1900-01-01	2016-10-31	NA	NA	0.00

Table 1: Numerical Table

II. Categorical Table

Field Name	% Populated	#Unique Value	Most Common Value
Record	100.00	1,000,000	Nan
SSN	100.00	835,819	999999999
First Name	100.00	78,136	EAMSTRMT
Last Name	100.00	177,001	ERJSAXA
Address	100.00	828,774	123 MAIN ST
Zip	100.00	26,370	68138
Home Phone	100.00	28,244	9999999999
Fraud Label	100.00	2	0

Table 2: Categorical Table

2. Visualization of Each Field:

(1) Field Name: Record

Description: Record Number: Ordinal Unique positive integer for each record from 1 to 1,000,000.

(2) Field Name: Date

Description: Date: The “date” field indicates the date on which the individual has raised a request for an application. The daily application distribution across time.

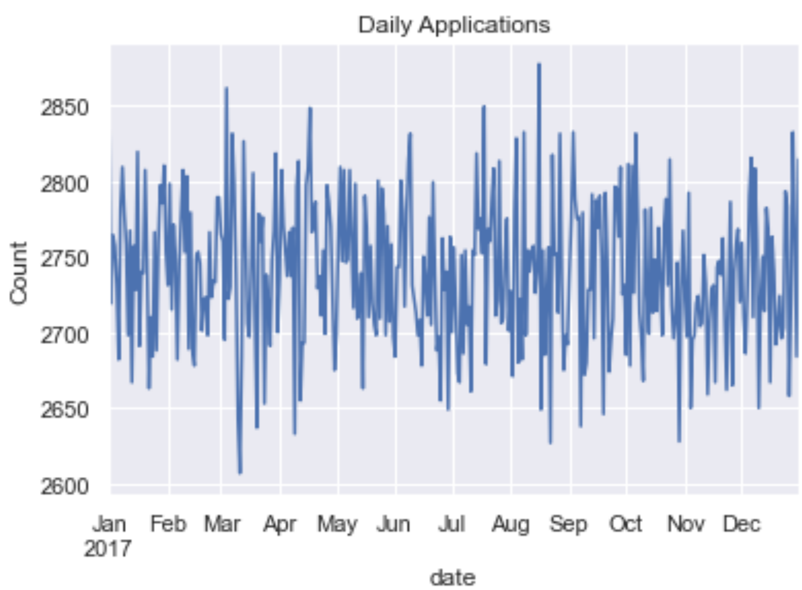

Fig 1. Monthly distribution of Application

(3) Field Name: Ssn

Description: SSN: SSN refers to the social security number of the applicant. The distribution indicates the top 15 most common SSNs used by applicants.

The most common value of the SSN is 999999999, the count is 16,935.

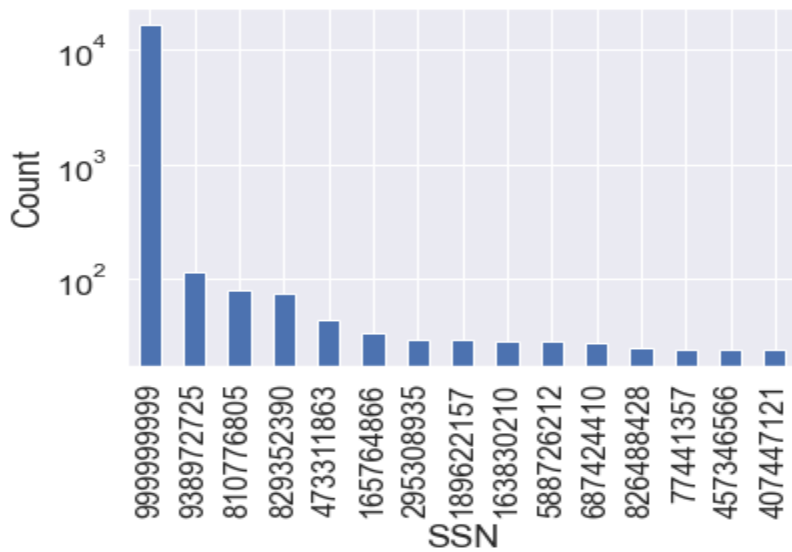

Fig 2. Bar plot displaying distribution of field 'ssn’

(4) Field Name: Firstname

Description: First Name: The field “firstname” refers to the First name of the applicant. The distribution indicates the top 15 most common First name used by applicants.

The most common value of the first name is EAMSTRMT, the count is 12,658.

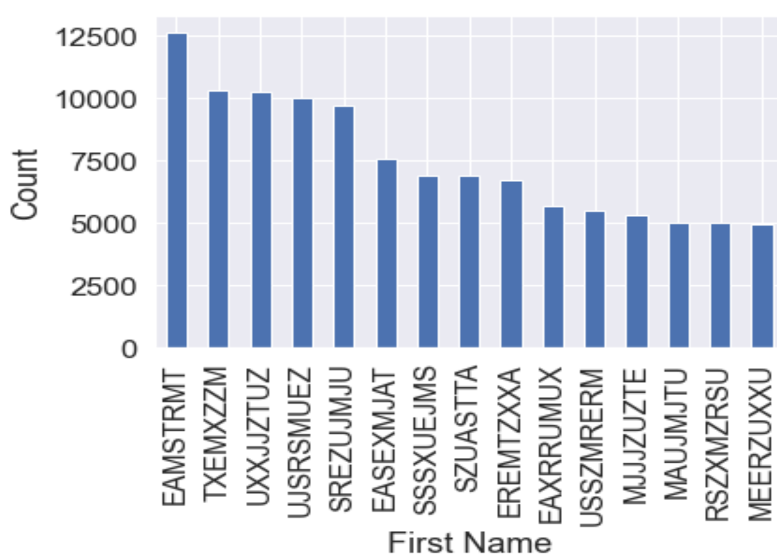

Fig 3. Bar plot displaying distribution of field 'First Name’

(5) Field Name: Lastname

Description: Last Name: The field “lastname” refers to the Last name of the applicant. The distribution indicates the top 15 most common Last name used by applicants.

The most common value of the first name is ERJSAXA, the count is 8,580.

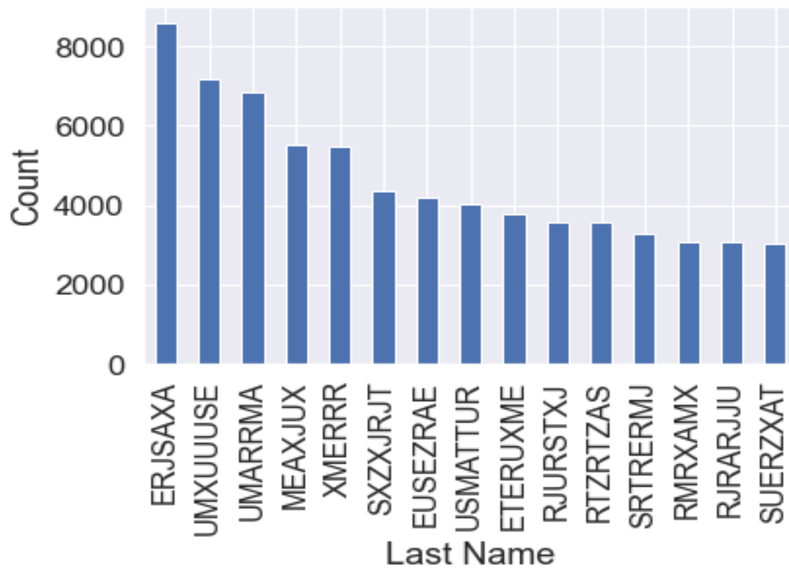

Fig 4. Bar plot displaying distribution of field 'Last Name’

(6) Field name: address

Description: Address: The field “address” refers to the Address of the applicant. The distribution indicates the top 15 most common Last name used by applicants.

The most common value of the first name is 123 MAIN ST, the count is 1,079.

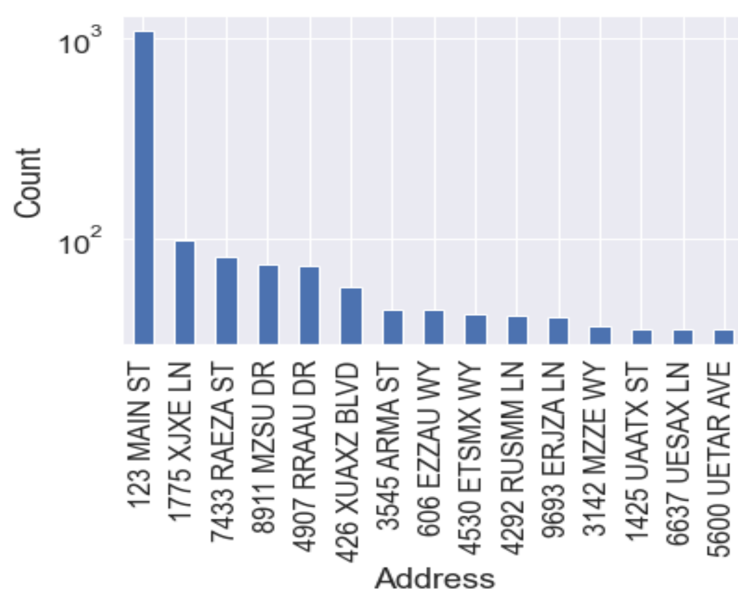

Fig 5. Bar plot displaying distribution of field ‘Address’

(7) Field Name: Zip5

Description: Zip: The field “zip5” refers to the Zip code of the applicant. The distribution indicates the top 10 most common Zip code used by applicants.

The most common value of the Zip code is 68138, the count is 823.

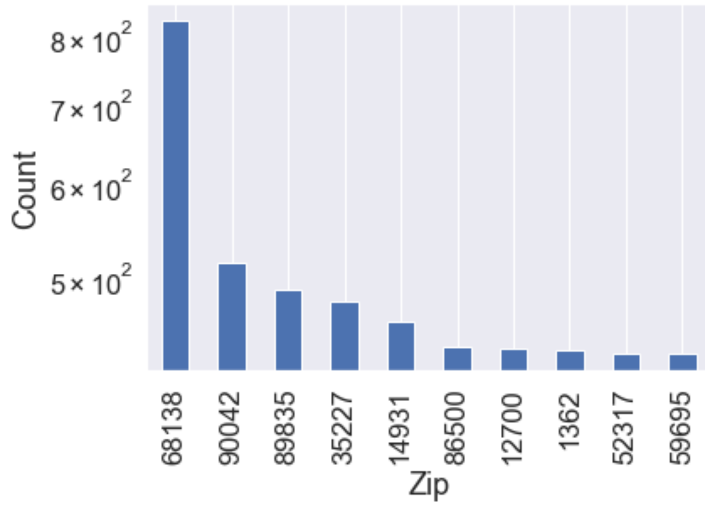

Fig 6. Bar plot displaying distribution of field ‘Zip5’

(8) Field Name: Dob

Description: Date of Birth: The field “dob” refers to the Date of Birth of the applicant. The distribution indicates the top 15 most common Zip code used by applicants.

The most common value of the dob code is 1907-06-26, the count is 26,568.

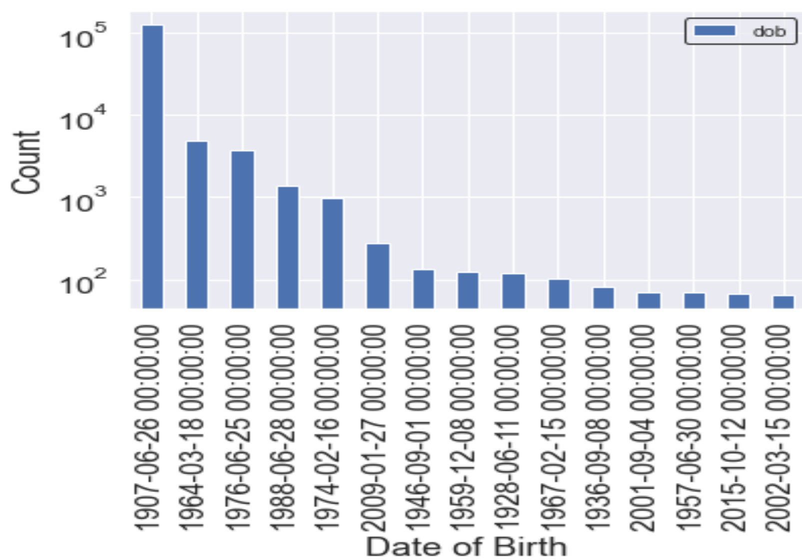

Fig 6. Bar plot displaying distribution of field ‘DoB’

(9) Field Name: Homephone

Description: Home Phone: The field “homephone” refers to the phone number of the applicant. The distribution indicates the top 10 most common home phone used by applicants.

The most common value of the dob code is ‘9999999999’, the count is 78,512.

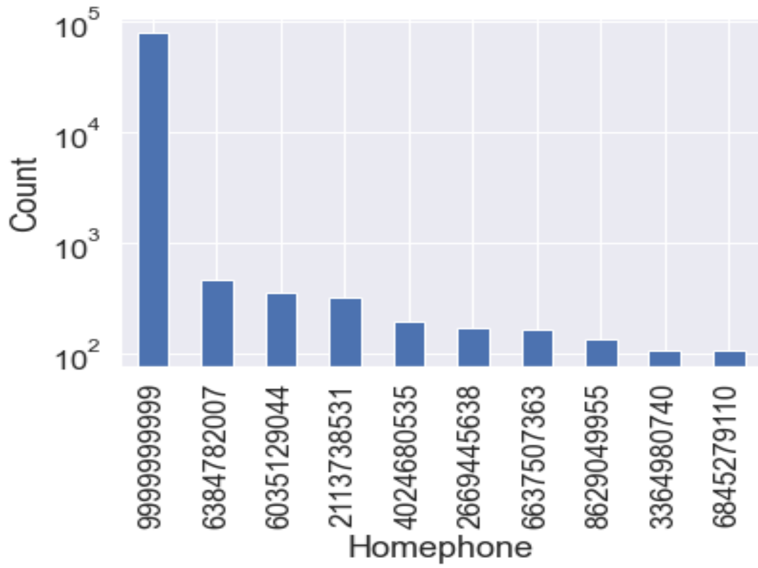

Fig 7. Bar plot displaying distribution of field ‘Homephone’

(10) Field Name: Fraud label

Description: Fraud = 0 (no fraud label), Fraud=1 (fraud label)

The count of fraud_label=0 (985,607) and fraud_label =1 (14,393)

Fig 8. Bar plot displaying distribution of field ‘Homephone’

Data Cleaning

- **Missing field values:**

  There were no missing values in the dataset. Therefore, no action was required.

  The applications dataset did not have any missing values in any of its fields. All fields were 100% populated. Therefore, no action was required.

- **Frivolous Values:**

  We saw that all the data fields were 100% populated; however, we found that four of the data fields contained frivolous values.

Data Field	Frivolous Value	Number of Records
ssn	999999999	16,935
address	123 MAIN ST	1,079
dob	19070626	126,568
homephone	9999999999	78,512

Table 3: Frivolous Values

The presence of frivolous values can lead to distorted outcomes, causing an overinflated count and false alarms for potentially fraudulent activity. As a solution, we opted to substitute these values with a distinct value equivalent to the corresponding record number. For instance, if record number 105 had "999999999" in the "ssn" data field and "123 MAIN ST" in the "address" data field, so these frivolous values were replaced with the number "105" in both data fields.

- **Leading Zeros:**

  There were certain data fields in the dataset that were meant to have a fixed length, such as zip5 field that should have exactly five digits. However, we found that some values in zip5 field had only four digits, and likewise, some values in other fields like SSN, DOB, and Homephone had fewer digits than the required length. To address this issue, we came up with a solution of adding '0' (zeroes) to the left side of the value, thereby padding it to the desired length. For instance, if the value in zip5 field was "1034", we added one zero to the left and made it "01034."

Variable Creation

Variables are created for fuzzy matching so in-depth analysis can be done better to capture fraud application. Two types of candidate variables were built:

- velocity in which applications were seen and the
- number of days since the last time a field was seen in an application.

Prior to creating candidate variables for our analysis and models, additional categorical variables were created.

a. Categorical variables

These categorical variables would be used to create candidate predictor variables. These categorical variables were created by using business knowledge and different combinations of existing fields were made, that fraudsters would most likely use in applications. For our project, we created 18 categorical variables as shown in the table below.

	Variable Name	**Description**
1	name	First name + Last name
2	fulladdress	Address + Zip code
3	name_dob	Name + Date of Birth
4	name_fulladdress	Name + Address
5	name_homephone	Name + Phone number
6	fulladdress_dob	Full Address + Date of Birth
7	fulladdress_homephone	Full Address + Phone number
8	dob_homephone	Date of Birth + Phone number
9	homephone_name_dob	Phone number + Name + Date of Birth
10	ssn_firstname	SSN + First name
11	ssn_lastname	SSN + Last name
12	ssn_address	SSN + address
13	ssn_zip5	SSN + zip5
14	ssn_dob	SSN + Date of Birth
15	ssn_homephone	SSN + Phone number
16	ssn_name	SSN + Name
17	ssn_fulladdress	SSN + Full Address
18	ssn_name_dob	SSN + Name + Date of Birth

Table 4: Categorical Variable

b. Velocity Candidate Variables

The frequency at which the entities (normal variables + categorical variables) appeared in the dataset for a particular application record is referred to as velocity. This measure can aid in identifying and detecting potentially fraudulent applications by assessing their rate of appearance. A higher velocity score would indicate a greater likelihood of fraudulent activity. Our assessment of candidate variables involves evaluating their velocity over various timeframes, namely 0, 1, 3, 7, 14, and 30 days.

Velocity = # of records with the same entity over the last n

days = {0, 1, 3, 7, 14, 30} days

Entity in the above formula refers to the normal variables + various categorical variables created previously (Refer Table 5).

	Variable Name		Variable Name
1	ssn_count_0	13	dob_count_0
2	ssn_count_1	14	dob_count_1
3	ssn_count_3	15	dob_count_3
4	ssn_count_7	16	dob_count_7
5	ssn_count_14	17	dob_count_14
6	ssn_count_30	18	dob_count_30
7	address_count_0	19	homephone_count_0
8	address_count_1	20	homephone_count_1
9	address_count_3	21	homephone_count_3
10	address_count_7	22	homephone_count_7
11	address_count_14	23	homephone_count_14
12	address_count_30	24	homephone_count_30

Table 5: Velocity Candidate Variable

c. Days Since Candidate Variables

The "Days Since" variable refers to the time elapsed since the last appearance of a comparable entity in a particular application record. It is represented by a whole number denoting the number of days that have passed since the last occurrence. In situations where the entity appears multiple times on the same date, the "Days Since" field for that entity in that record will display a value of 1. If the value is small, but not zero, it implies a greater probability of fraudulent behavior.

Days since = # of days since the entity was last seen

	Variable Name		Variable Name
1	ssn_days_since	12	dob_homephone_days_since
2	address_days_since	13	homephone_name_dob_days_since
3	dob_days_since	14	ssn_firstname_days_since
4	homephone_days_since	15	ssn_lastname_days_since
5	name_days_since	16	ssn_address_days_since
6	fulladdress_days_since	17	ssn_zip5_days_since
7	name_dob_days_since	18	ssn_dob_days_since
8	name_fulladdress_days_since	19	ssn_homephone_days_since
9	name_homephone_days_since	20	ssn_name_days_since
10	fulladdress_dob_days_since	21	ssn_fulladdress_days_since
11	fulladdress_homephone_days_since	22	ssn_name_dob_days_since

Table 6: Days Since Candidate Variable

d. Relative Velocity Candidate Variables

Relative velocity refers to the speed at which an entity is seen in the dataset for a particular application record over a short period of time (0 - 1 days) in relation to how often the same entity is seen over a longer period (3 – 30 days). The speed at which these applications come through in a shorter timeframe versus a longer time frame is a way to detect and identify potentially fraudulent applications. A higher value of relative velocity would indicate a greater likelihood of a fraudulent application. For our model, welook at a relative velocity over 3, 7, 14, and 30 days.

x = {0,1} days

n = {3, 7, 14, 30} days

	Variable Name		Variable Name
1	ssn_count_0_by_3	12	address_count_0_by_30
2	ssn_count_0_by_7	13	address_count_1_by_3
3	ssn_count_0_by_14	14	address_count_1_by_7
4	ssn_count_0_by_30	15	address_count_1_by_14
5	ssn_count_1_by_3	16	address_count_1_by_30
6	ssn_count_1_by_7	17	dob_count_0_by_3
7	ssn_count_1_by_14	18	dob_count_0_by_7
8	ssn_count_1_by_30	19	dob_count_0_by_14
9	address_count_0_by_3	20	dob_count_0_by_30
10	address_count_0_by_7	21	dob_count_1_by_3
11	address_count_0_by_14	22	dob_count_1_by_7

Table 7: Relative Velocity Candidate Variable

e. Combination Unique Velocity Candidate Variables

Combination unique velocity refers to the speed at which one entity is seen in relation with another unique entity over a period of time. A higher value of velocity would indicate a greater likelihood of a fraudulent application. For our candidate variables, we look at a velocity over 1, 3, 7, 14, 30, and 60 days.

Combination Unique Velocity = # of Unique Entity2 used against Entity1 over the past ndays

n = {1, 3, 7, 14, 30, 60}

Entity in the above formula refers to the various categorical variables created previously. Some of the combination unique velocity variables can be found in the table below.

	Variable Name		Variable Name
1	ssn_unique_count_for_address_1	12	ssn_unique_count_for_dob_60
2	ssn_unique_count_for_address_3	13	ssn_unique_count_for_homephone_1
3	ssn_unique_count_for_address_7	14	ssn_unique_count_for_homephone_3
4	ssn_unique_count_for_address_14	15	ssn_unique_count_for_homephone_7
5	ssn_unique_count_for_address_30	16	ssn_unique_count_for_homephone_14
6	ssn_unique_count_for_address_60	17	ssn_unique_count_for_homephone_30
7	ssn_unique_count_for_dob_1	18	ssn_unique_count_for_homephone_60
8	ssn_unique_count_for_dob_3	19	ssn_unique_count_for_name_1
9	ssn_unique_count_for_dob_7	20	ssn_unique_count_for_name_3
10	ssn_unique_count_for_dob_14	21	ssn_unique_count_for_name_7
11	ssn_unique_count_for_dob_30	22	ssn_unique_count_for_name_14

Table 8: Combination Unique Velocity Candidate Variable

Feature Selection

Feature selection is the process of selecting a subset of relevant features (also known as variables or predictors) from a larger set of available features in a dataset. In machine learning, feature selection is an important technique for improving the performance of a model, reducing computational time and improving generalization by reducing the dimensionality of the dataset.

Also, features that were either highly correlated with others or not significant to perform an accurate prediction were ignored. Additionally, feature selection enhances model performance.

1. Filter Method: Filter method is generally used as a preprocessing step. The selection of features is independent of any machine learning algorithms. Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable. Variable is useful if has correlation with the output otherwise that the variable can removed.

2. Wrapper Method: These methods select features by using a machine learning algorithm to evaluate the performance of different subsets of features. The algorithm is trained and tested on different subsets of features, and the subset that achieves the best performance is selected.
    a. Forward selection is a feature selection technique that is commonly used in wrapper methods. It is a stepwise procedure that starts with an empty set of features and iteratively adds one feature at a time to the set, based on its impact on the performance of the machine learning algorithm. Forward selection was used and following variable were obtained:

Sorted List of Variables from Forward Selection:

Rank	variable name	avg_score
1	max_count_by_address_30	0.37964656
2	max_count_by_ssn_dob_7	0.54949073
3	max_count_by_homephone_3	0.58387743
4	max_count_by_fulladdress_30	0.59249586

5	zip5_count_3	0.60990685
6	max_count_by_ssn_dob_30	0.61060329
7	max_count_by_homephone_7	0.61138679
8	fulladdress_count_0_by_30	0.61365021
9	max_count_by_fulladdress_homephone_30	0.6142596
10	ssn_dob_day_since	0.61460782
11	max_count_by_address_7	0.61460782
12	address_day_since	0.61460782
13	fulladdress_day_since	0.61460782
14	max_count_by_fulladdress_3	0.61460782
15	max_count_by_address_3	0.61460782
16	address_count_14	0.61460782
17	fulladdress_count_14	0.61460782
18	max_count_by_address_1	0.61460782
19	max_count_by_fulladdress_1	0.61460782
20	address_count_7	0.61460782
21	fulladdress_count_7	0.61460782
22	address_unique_count_for_name_homephone_60	0.61460782
23	address_count_0_by_30	0.61460782
24	address_unique_count_for_homephone_name_dob_60	0.61460782
25	fulladdress_unique_count_for_ssn_homephone_60	0.61460782

Table 7: Top 25 Candidate Variables

Preliminary models exploration

After selecting the top 25 best variables, with each having a wrapper score above 0.5. We start with a logistic regression to get a base line model and then test Decision Tree, Random Forest, Boosted Tree, and Neural Network models with varying hyperparameters to choose the best models by comparing the Fraud Detection Rate (FDR) at 3% for the train, test and out of time (OOT) datasets.

Logistics Regression:

The logistic regression is one of the most popular classification algorithms. In logistic regression, a linear output is converted into a probability between 0 & 1 using the sigmoid function.

For this project's fraud analysis, five versions of logistic regression were created by changing the solver, penalty, and c hyperparameters and training the model with the identified 30 best variables. To understand the used hyperparameters further:

- Solver: This parameter represents which algorithm to use in the optimization problem.
    - liblinear − It is a good choice for small datasets. It also handles L1 penalty. For multiclass problems, it is limited to one-versus-rest schemes.
    - lbfgs − For multiclass problems, it handles multinomial loss. It also handles only L2 penalty.

    Default is 'lbfgs'.
- Penalty: Penalized logistic regression imposes a penalty to the logistic model for having too many variables. This results in shrinking the coefficients of the less contributive variables toward zero. This is also known as regularization. L1 is therefore useful for feature selection, as we can drop any variables associated with coefficients that go to zero. L2, on the other hand, is useful when you have collinear/codependent features. Default is 'L2'
- C: It represents the inverse of regularization strength, which must always be a positive float. Smaller values specify stronger regularization.

The results of the logistic regression are shown below.

Model			Parameter					Average FDR at 3%		
	Iteration	Variable	Penalty	c	solver	l1_ratio	Max_iter	Train	Test	OOT
Logistic Regression	1(default)	10	l2	1	lbfgs	None	20	48.97	48.46	47.38
	2	15	l2	0.1	lbfgs	None	30	48.49	49.35	47.01
	3	15	l1	1	saga	None	3	47.68	48.12	46.31
	4	20	elasticnet	1	saga	0.4	5	47.99	47.52	46.37
	5	20	l1	0.1	saga	None	100	47.44	48.02	46.26

Table 8: Logistic Regression Model Results

The best results were given when solver = lbfgs, penalty = l2, c = 1.

Decision Tree:

In decision analysis, a decision tree can be used to represent decisions and decision making visually and explicitly. The main goal of Decision Trees is to create a model predicting target variable value by learning simple decision rules deduced from the data features. Decision trees have two main entities; one is root node, where the data splits, and other is decision nodes or leaves, where we got final output. In three-dimensional view, decision trees approximate the surface into y = f(x) with steps or platforms. These steps form boxes and each box contains the average of the dependent variable y for its range.

For this project’s fraud analysis, four versions of logistic regression were created by changing the max_depth, min_sample_leaf and min_samples_split hyperparameters and training the model with the identified 30 best variables. To understand the used hyperparameters further:

- max_depth: 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. Default is None
- min_sample_leaf: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression. Default is 1.
- min_samples_split: The minimum number of samples required to split an internal node. Default is 2.

The results of the Decision Trees are shown below.

Model			Parameter						Average FDR at 3%		
	Iteration	Variable	criterion	max_depth	min_sample_split	min_sample_leaf	max_features	splitter	Train	Test	OOT
Decision Tree	1(default)	10	gini	5	50	30	30	best	51.37	51.02	48.89
	2	15	entropy	15	45	25	25	random	53.14	51.89	51
	3	15	gini	30	30	20	20	best	53.97	52.77	50.13
	4	20	gini	25	50	30	30	best	53.95	52.17	50.25
	5	20	gini	None	8	5	5	random	54.34	51.98	49.96
	6	20	gini	20	5	2	2	best	53.97	51.51	49.58

Table 9: Decision Tree model results

The best results were given when max_depth = 30, min_sample_leaf = 20, min_samples_split = 30.

Random Forest

Random forest is an ensemble of many decision trees. Random forests are built using a method called bagging in which each decision trees are used as parallel estimators. If used for a classification problem, the result is based on average prediction from each decision tree. For regression, the prediction of a leaf node is the mean value of the target values in that leaf.
Random forest regression takes mean value of the results from decision trees.
Random forests reduce the risk of overfitting and accuracy is much higher than a single decision tree. Furthermore, decision trees in a random forest run in parallel so that the time does not become a bottleneck.

The success of a random forest highly depends on using uncorrelated decision trees. If we use same or very similar trees, overall result will not be much different than the result of a single decision tree. Random forests achieve to have uncorrelated decision trees.
by bootstrapping and feature randomness.

For this project's fraud analysis, four versions of Random Forests were created by changing the n_estimators, max_depth, max_features, min_samples_leaf, and min_samples_split hyperparameters and training the model with the identified 30 best variables. To understand the used hyperparameters further:

- n_estimators: The number of trees in the forest. default=100
- max_depth: 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. default=None
- max_features: The number of features to consider when looking for the best split. default= "auto"
- min_samples_leaf: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression. Default is 1.
- min_samples_split: The minimum number of samples required to split an internal node. Default is 2.

The results of the Random Forest models are shown below.

Model	Parameter								Average FDR at 3%		
	Variable	bootstrap	n_estimators	max_depth	max_features	min_sample_split	min_samples_leaf	criterion	Train	Test	OOT
Random Forest	10	TRUE	20	5	3	50	30	gini	56.27	51.45	49.02
	15	TRUE	30	10	4	45	28	entropy	52.71	53.23	50.43
	15	TRUE	50	25	6	30	20	gini	53.9	52.29	50.19
	20	TRUE	60	5	8	50	18	gini	51.8	52.41	50.36
	20	TRUE	80	15	10	8	5	entropy	53.67	53	50.37

Table 10: Random Forest model results

The best results were given when n_estimators = 60, max_depth = 5, max_features = 8, min_samples_leaf = 18 and min_samples_split = 50.

Boosted Tree

Gradient boosting algorithm sequentially combines weak learners in way that each new learner fits to the residuals from the previous step so that the model improves. The final model aggregates the results from each step and a strong learner is achieved. Gradient boosted decision trees algorithm uses decision trees as week learners. A loss function is used to detect the residuals. For instance, mean squared error (MSE) can be used for a regression task and logarithmic loss (log loss) can be used for classification tasks. It is worth noting that existing trees in the model do not change when a new tree is added. The added decision tree fits the residuals from the current model.

LightGBM:

Light Gradient Boosting Machine is a popular open-source gradient boosting framework that uses tree-based learning algorithms. It is designed to be efficient, scalable, and optimized for handling large datasets. LightGBM is particularly useful for handling high-dimensional data, where there are many features, as well as for data with a large number of observations.

LightGBM uses a gradient boosting approach to iteratively improve the performance of a decision tree ensemble model. It uses a technique called "leaf-wise" growth, which differs from the traditional "level-wise" growth used by other tree-based algorithms. Leaf-wise growth builds a tree by expanding the leaf with the maximum gain, rather than expanding all the leaves on the same level.

Model	Parameter				Average FDR at 3%		
LightBGM	Variable	n_estimators	max_depth	learning_rate	Train	Test	OOT
	10	20	2	0.1	51.07	51.56	48.85
	15	250	4	0.01	52.28	52.5	50.11
	15	500	6	0.1	53.5	51.92	50.57
	20	750	8	0.1	53.29	52.71	50.45
	20	1000	10	0.01	53.16	52.55	50.74

Table 11: Light Boosted Tree model results

The best results were given when n_estimators = 1000, max_depth = 10 and learning_rate = 0.01

XGBoost:

Extreme Gradient Boosting (XGBoost) is an open-source software library that provides an optimized implementation of the gradient boosting algorithm using decision tree ensembles. It was developed with a focus on scalability, speed, and accuracy and is widely used in machine learning competitions and industry applications.

XGBoost is similar to other tree-based boosting algorithms like LightGBM and AdaBoost, but it incorporates several key optimizations that make it particularly efficient and effective.

Model	Parameter				Average FDR at 3%		
XGBoost	Variable	n_estimators	max_depth	learning_rate	Train	Test	OOT
	10	20	2	0.1	49.51	49.9	47.86
	15	250	4	0.01	51.45	51.25	49.12
	15	500	6	0.1	53.64	52.41	50.36
	20	750	8	0.1	54.09	52.31	50.23
	20	1000	10	0.01	53.47	52.87	50.58

Table 12: Extreme Gradient Boosting Tree model results

The best results were given when n_estimators = 1000, max_depth = 10 and learning_rate = 0.01

Categorical Boosting Tree:

Categorical Boosting Tree (CatBoost) is a machine learning algorithm for gradient boosting on decision trees that is designed to handle categorical features and missing values in data. It was developed by Yandex, a Russian search engine, and is now an open-source software library.

CatBoost includes several features that make it effective for handling categorical data, including:

- Ordered boosting: This technique handles categorical features by considering the natural order of the categories rather than just treating them as unordered. This can improve the accuracy of the model and reduce overfitting.
- Target encoding: This technique encodes categorical features based on the target variable, which can improve the accuracy of the model and reduce the risk of overfitting.
- Feature combination: CatBoost can automatically combine categorical features to create new features, which can improve the accuracy of the model.
- Handling missing values: CatBoost can handle missing values in the data without requiring imputation or removal of observations with missing values.

CatBoost is also optimized for performance and includes features for handling large datasets, such as parallel processing and support for GPU acceleration

Overall, CatBoost is a powerful machine learning algorithm for handling categorical data and missing values. It has been shown to be effective for a variety of tasks, including classification, regression, and ranking, and is widely used in industry applications.

Model	Parameter				Average FDR at 3%		
	Variable	n_estimators	max_depth	learning_rate	Train	Test	OOT
CatBoost	10	20	2	0.1	51.22	51.29	48.86
	15	250	4	0.01	52.32	52.03	49.96
	15	500	6	0.1	52.1	52.35	49.79
	20	750	8	0.1	53.88	52.25	50.4
	20	1000	10	0.01	52.98	52.53	50.48

Table 13: Categorical Boosting Tree model results

The best results were given when n_estimators = 60, max_depth = 5, max_features = 8, min_samples_leaf = 18 and min_samples_split = 50.

Learning rate and n_estimators

Hyperparameters are key parts of learning algorithms which effect the performance and accuracy of a model. Learning rate and n_estimators are two critical hyperparameters for gradient boosting decision trees. Learning rate, denoted as α, simply means how fast the modellearns.

Neural Network

A neural network is a type of machine learning model that is inspired by the structure and function of the human brain. It is a network of interconnected nodes or "neurons" that work together to solve complex problems.

In a neural network, data is input into an input layer, and then it is processed through one or more hidden layers of interconnected neurons. Each neuron performs a simple computation and passes its output to the next layer of neurons. The output layer produces the final prediction or decision.

Neural networks can be used for a wide variety of machine learning tasks, including classification, regression, and image or speech recognition. They have proven to be highly effective for many applications, including natural language processing, autonomous driving, and recommendation systems.

One of the most popular types of neural networks is the deep neural network, which includes multiple hidden layers of neurons. Deep neural networks can learn very complex patterns in data, but they also require large amounts of data and computing power for training.

Overall, neural networks are a powerful machine learning tool that have revolutionized many fields, including computer vision, natural language processing, and speech recognition. They continue to be an active area of research and development, with many new techniques and architectures being developed to improve their performance and applicability.

Model	Parameter								Average FDR at 3%		
	Variable	activation	solver	alpha	learning_rate_init	hidden_layer_size	max_iter	learning_rate	Train	Test	OOT
Neural Network	10	relu	adam	0.0001	0.0001	(5,)	200	constant	52.33	52.45	50.15
	15	relu	adam	0.05	0.05	(10,10,)	250	constant	48.99	48.22	47.19
	15	relu	sgd	0.005	0.005	(20,20,20)	50	adaptive	52.85	52.43	50.46
	20	relu	lbfgs	0.15	0.15	(5,)	300	constant	50.24	50.48	48.52
	20	logistic	lbfgs	0.1	0.1	(20,20,20)	150	adaptive	43.18	42.76	41.34

Table 14: Neural Network model results

The best results were given when activation = ‘relu’, solver =’sgd’, alpha = ‘0.15’, hidden_layer_size = (5,) and learning rate= constant.

Final Model and Results:

The below diagram gives us the results from all models that were tested to predict fraud.

Model			Parameter								Average FDR at 3%		
	Iteration	Variable	Penalty	c		solver		l1_ratio		Max_iter	Train	Test	OOT
Logistic Regression	1(default)	10	l2	1		lbfgs		None		20	48.97	48.46	47.38
	2	15	l2	0.1		lbfgs		None		30	48.49	49.35	47.01
	3	15	l1	1		saga		None		3	47.68	48.12	46.31
	4	20	elasticnet	1		saga		0.4		5	47.99	47.52	46.37
	5	20	l1	0.1		saga		None		100	47.44	48.02	46.26
Decision Tree	Iteration	Variable	criterion	max_depth		min_sample_split	min_sample_leaf	max_features		splitter	Train	Test	OOT
	1(default)	10	gini	5		50	30	30		best	51.37	51.02	48.89
	2	15	entropy	15		45	25	25		random	53.14	51.89	51
	3	15	gini	30		30	20	20		best	53.97	52.77	50.13
	4	20	gini	25		50	30	30		best	53.95	52.17	50.25
	5	20	gini	None		8	5	5		random	54.34	51.98	49.96
	6	20	gini	20		5	2	2		best	53.97	51.51	49.58
Random Forest	Iteration	Variable	bootstrap	n_estimators	max_depth	max_features	min_sample_split	min_samples_leaf		criterion	Train	Test	OOT
	1(default)	10	TRUE	20	5	3	50	30		gini	56.27	51.45	49.02
	2	15	TRUE	30	10	4	45	28		entropy	52.71	53.23	50.43
	3	15	TRUE	50	25	6	30	20		gini	53.9	52.29	50.19
	4	20	TRUE	60	5	8	50	18		gini	51.8	52.41	50.36
	5	20	TRUE	80	15	10	8	5		entropy	53.67	53	50.37
LightBGM	Iteration	Variable	n_estimators			max_depth			learning_rate		Train	Test	OOT
	1(default)	10	20			2			0.1		51.07	51.56	48.85
	2	15	250			4			0.01		52.28	52.5	50.11
	3	15	500			6			0.1		53.5	51.92	50.57
	4	20	750			8			0.1		53.29	52.71	50.45
	5	20	1000			10			0.01		53.16	52.55	50.74
XGBoost	Iteration	Variable	n_estimators			max_depth			learning_rate		Train	Test	OOT
	1(default)	10	20			2			0.1		49.51	49.9	47.86
	2	15	250			4			0.01		51.45	51.25	49.12
	3	15	500			6			0.1		53.64	52.41	50.36
	4	20	750			8			0.1		54.09	52.31	50.23
	5	20	1000			10			0.01		53.47	52.87	50.58
CatBoost	Iteration	Variable	n_estimators			max_depth			learning_rate		Train	Test	OOT
	1(default)	10	20			2			0.1		51.22	51.29	48.86
	2	15	250			4			0.01		52.32	52.03	49.96
	3	15	500			6			0.1		52.1	52.35	49.79
	4	20	750			8			0.1		53.88	52.25	50.4
	5	20	1000			10			0.01		52.98	52.53	50.48
Neural Network	Iteration	Variable	activation	solver		alpha	learning_rate_init	hidden_layer_size	max_iter	learning_rate	Train	Test	OOT
	1(default)	10	relu	adam		0.0001	0.0001	(5,)	200	constant	52.33	52.45	50.15
	2	15	relu	adam		0.05	0.05	(10,10,)	250	constant	48.99	48.22	47.19
	3	15	relu	sgd		0.005	0.005	(20,20,20)	50	adaptive	52.85	52.43	50.46
	4	20	relu	lbfgs		0.15	0.15	(5,)	300	constant	50.24	50.48	48.52
	5	20	logistic	lbfgs		0.1	0.1	(20,20,20)	150	adaptive	43.18	42.76	41.34

Table 15: Final Summary of Models

Final Model: Light Boosting Gradient Model

After comparing the results between logistics regression, boosted trees, random forest, and a neural network, we determined that Light Boosting Gradient Model performed the best. Light Boosting Gradient Model outperformed other models for both testing and out of time validation datasets with 52.55 % and 50.74% respectively.

The chosen hyperparameters are:

- No. of variables: 20
- n_estimators: 1000
- max_depth: 10
- Learning_rate: 0.01

Summary of Results

Summary Table has basically two categories: Bin statistics and cumulative statistics. The Fraud rate on the top of the Table is calculated by #Bads divided by #Goods and #Bads.

% Goods = Cum.Good/ Total Good

The fraud detection rate (%Bads) is calculated to be the number of true frauds in the bin, which are caught by the model, divided by the total number of true frauds exists in the entire dataset. FDR reflects how many frauds can be caught by a model, with a fixed number of predicted positives.

FDR = Cum.Bad / Total Bad

KS is the maximum difference in cumulative fractions of goods and bads flagged at any possible cutoff.

KS= %Bad - % Good

False positive rate (FPR) – Percentage of total legitimate events that are incorrectly predicted as fraud.

FPR= Cum.Good/ Cum.Bad

Light Boosting Gradient Model Train Table:

Training	#Records		#Goods		#Bads		Fraud Rate				
	583454		575068		8386		0.014169				
Bin Statistics					Cumulative Statistics						
#Records	#Goods	#Bads	% Goods	%Bads	Total #Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	1567	4268	26.8551842	73.1448158	5835	1567	4268	0.27248951	50.8943477	50.6218582	0.36715089
2	5691	143	97.5488516	2.45114844	11669	7258	4411	1.26211161	52.5995707	51.3374591	1.64543187
3	5773	62	98.9374464	1.06255356	17504	13031	4473	2.26599289	53.3388982	51.0729053	2.91325732
4	5787	47	99.1943778	0.80562221	23338	18818	4520	3.27230867	53.8993561	50.6270474	4.16327434
5	5790	45	99.2287918	0.77120823	29173	24608	4565	4.27914612	54.4359647	50.1568186	5.3905805
6	5782	52	99.1086733	0.89132671	35007	30390	4617	5.28459243	55.0560458	49.7714534	6.58219623
7	5805	30	99.4858612	0.51413882	40842	36195	4647	6.29403827	55.4137849	49.1197466	7.78889606
8	5792	42	99.2800823	0.71991772	46676	41987	4689	7.30122351	55.9146196	48.6133961	8.95436127
9	5776	59	98.9888603	1.01113967	52511	47763	4748	8.30562647	56.6181731	48.3125467	10.059604
10	5801	33	99.4343504	0.56564964	58345	53564	4781	9.31437673	57.0116861	47.6973094	11.2035139
11	5786	49	99.1602399	0.83976007	64180	59350	4830	10.3205186	57.5959933	47.2754747	12.2877847
12	5800	34	99.4172095	0.58279054	70014	65150	4864	11.329095	58.001431	46.672336	13.3943257
13	5795	40	99.3144816	0.68551842	75849	70945	4904	12.3368019	58.4784164	46.1416145	14.4667618
14	5799	36	99.3830334	0.61696658	81684	76744	4940	13.3452044	58.9077033	45.5624989	15.5352227
15	5786	48	99.1772369	0.82276311	87518	82530	4988	14.3513463	59.4800859	45.1287396	16.5457097
16	5791	44	99.2459297	0.75407027	93353	88321	5032	15.3583576	60.0047699	44.6464122	17.551868
17	5795	39	99.331505	0.66849503	99187	94116	5071	16.3660645	60.4698307	44.1037661	18.5596529
18	5793	42	99.2802057	0.71979434	105022	99909	5113	17.3734237	60.9706654	43.5972417	19.5401917
19	5795	39	99.331505	0.66849503	110856	105704	5152	18.3811306	61.4357262	43.0545956	20.5170807
20	5799	36	99.3830334	0.61696658	116691	111503	5188	19.3895331	61.8650131	42.4754801	21.4924827

Table 15: Train result table

Light Boosting Gradient Model Test Table:

Testing	#Records		#Goods		#Bads		Fraud Rate				
	250053		246432		3621		0.014418				
Bin Statistics					Cumulative Statistics						
#Records	#Goods	#Bads	% Goods	%Bads	Total #Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	663	1838	26.5093962	73.4906038	2501	663	1838	0.26903974	50.7594587	50.490419	0.36071817
2	2456	44	98.24	1.76	5001	3119	1882	1.26566355	51.9745927	50.7089291	1.65727949
3	2483	18	99.2802879	0.71971212	7502	5602	1900	2.27324373	52.4716929	50.1984492	2.94842105
4	2488	12	99.52	0.48	10002	8090	1912	3.28285288	52.8030931	49.5202402	4.23117155
5	2481	20	99.2003199	0.79968013	12503	10571	1932	4.28962148	53.3554267	49.0658052	5.47153209
6	2477	23	99.08	0.92	15003	13048	1955	5.29476691	53.9906103	48.6958434	6.6741688
7	2482	19	99.2403039	0.75969612	17504	15530	1974	6.30194131	54.5153273	48.213386	7.86727457
8	2483	17	99.32	0.68	20004	18013	1991	7.30952149	54.9848108	47.6752893	9.04721246
9	2483	18	99.2802879	0.71971212	22505	20496	2009	8.31710168	55.4819111	47.1648094	10.2020906
10	2480	20	99.2	0.8	25005	22976	2029	9.32346449	56.0342447	46.7107802	11.3238048
11	2484	17	99.3202719	0.67972811	27506	25460	2046	10.3314505	56.5037283	46.1722778	12.4437928
12	2486	14	99.44	0.56	30006	27946	2060	11.340248	56.8903618	45.5501138	13.5660194
13	2489	12	99.5201919	0.47980808	32507	30435	2072	12.350263	57.2217619	44.871499	14.6887066
14	2480	20	99.2	0.8	35007	32915	2092	13.3566258	57.7740956	44.4174698	15.7337476
15	2473	28	98.8804478	1.11955218	37508	35388	2120	14.360148	58.5473626	44.1872146	16.6924528
16	2478	22	99.12	0.88	40008	37866	2142	15.3656993	59.1549296	43.7892303	17.6778711
17	2486	15	99.4002399	0.5997601	42509	40352	2157	16.3744968	59.5691798	43.194683	18.7074641
18	2482	19	99.2403039	0.75969612	45010	42834	2176	17.3816712	60.0938967	42.7122255	19.6847426
19	2488	12	99.52	0.48	47510	45322	2188	18.3912804	60.4252969	42.0340165	20.713894
20	2481	20	99.2003199	0.79968013	50011	47803	2208	19.398049	60.9776305	41.5795815	21.6499094

Table 15: Test result table

Light Boosting Gradient Model out of Time Table:

OOT	#Records		#Goods		#Bads		Fraud Rate				
	166493		164107		2386		0.014539				
Bin Statistics					Cumulative Statistics						
#Records	#Goods	#Bads	% Goods	%Bads	Total #Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	506	1159	30.3903904	69.6096096	1665	506	1159	0.30833542	48.575021	48.2666855	0.43658326
2	1631	34	97.957958	2.04204204	3330	2137	1193	1.30219917	50	48.6978008	1.79128248
3	1649	16	99.039039	0.96096096	4995	3786	1209	2.30703139	50.6705784	48.363547	3.13151365
4	1651	14	99.1591592	0.84084084	6660	5437	1223	3.31308232	51.2573345	47.9442521	4.44562551
5	1650	15	99.0990991	0.9009009	8325	7087	1238	4.31852389	51.8860017	47.5674778	5.72455574
6	1654	11	99.3393393	0.66066066	9990	8741	1249	5.3264029	52.3470243	47.0206214	6.99839872
7	1653	12	99.2792793	0.72072072	11655	10394	1261	6.33367254	52.8499581	46.5162855	8.24266455
8	1648	16	99.0384615	0.96153846	13319	12042	1277	7.3378954	53.5205365	46.1826411	9.42991386
9	1655	10	99.3993994	0.6006006	14984	13697	1287	8.34638376	53.9396479	45.5932642	10.6425796
10	1652	13	99.2192192	0.78078078	16649	15349	1300	9.35304405	54.4844929	45.1314488	11.8069231
11	1653	12	99.2792793	0.72072072	18314	17002	1312	10.3603137	54.9874267	44.627113	12.9588415
12	1652	13	99.2192192	0.78078078	19979	18654	1325	11.366974	55.5322716	44.1652976	14.0784906
13	1651	14	99.1591592	0.84084084	21644	20305	1339	12.3730249	56.1190277	43.7460027	15.1643017
14	1653	12	99.2792793	0.72072072	23309	21958	1351	13.3802946	56.6219614	43.2416669	16.2531458
15	1652	13	99.2192192	0.78078078	24974	23610	1364	14.3869549	57.1668064	42.7798515	17.3093842
16	1656	9	99.4594595	0.54054054	26639	25266	1373	15.3960526	57.5440067	42.1479541	18.4020393
17	1650	15	99.0990991	0.9009009	28304	26916	1388	16.4014941	58.1726739	41.7711798	19.3919308
18	1653	12	99.2792793	0.72072072	29969	28569	1400	17.4087638	58.6756077	41.2668439	20.4064286
19	1646	19	98.8588589	1.14114114	31634	30215	1419	18.4117679	59.4719195	41.0601516	21.2931642
20	1657	8	99.5195195	0.48048048	33299	31872	1427	19.421475	59.8072087	40.3857337	22.3349685

Table 16: Out of time result table