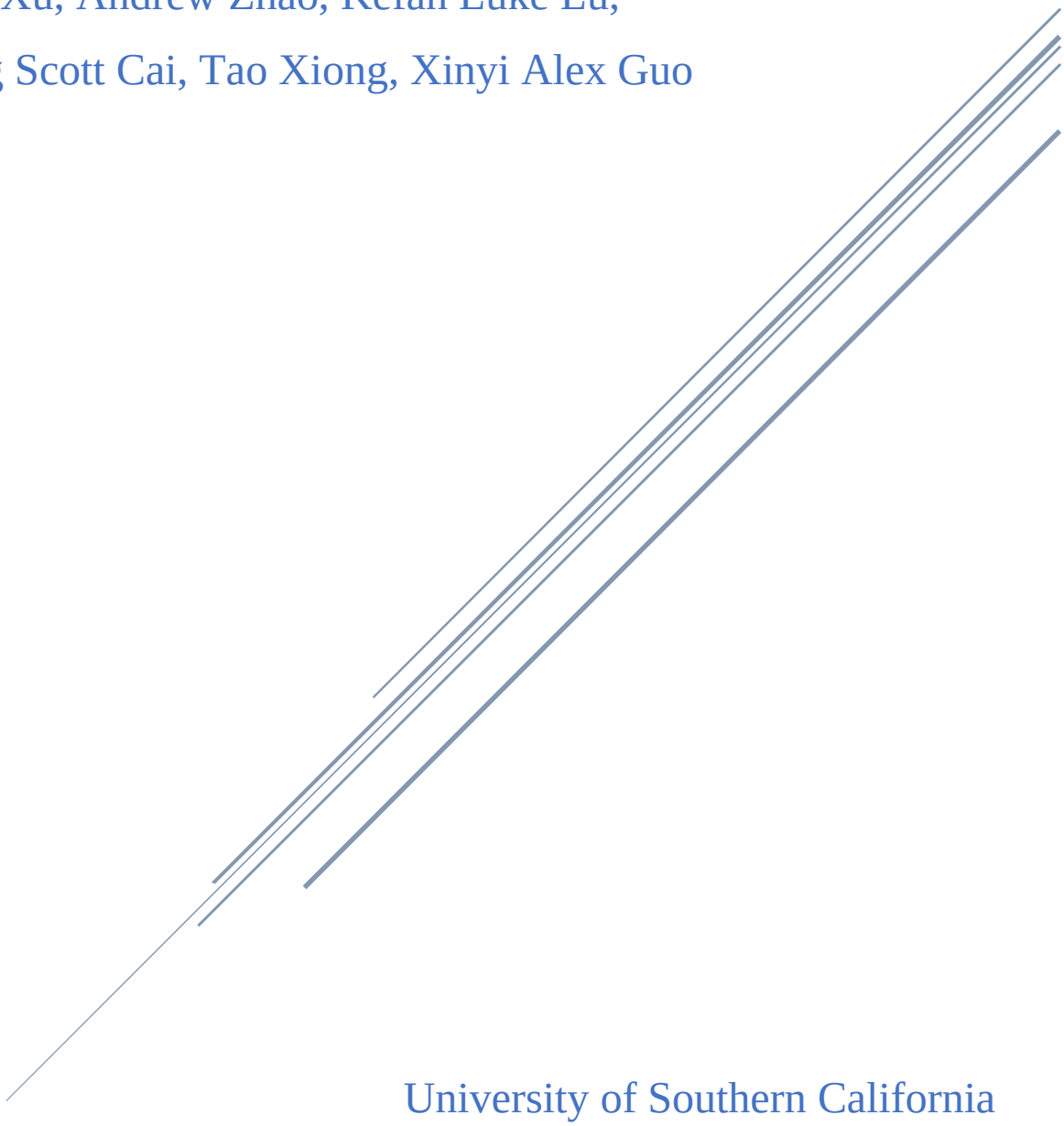
PRODUCT APPLICATION DATA FRUAD DETECTION REPORT

Group 6

Daniel Xu, Andrew Zhao, Kefan Luke Lu,

Yuxing Scott Cai, Tao Xiong, Xinyi Alex Guo



University of Southern California

DSO 562 Fraud Analytics

CONTENTS

EXECUTIVE SUMMARY

This report evaluates The Product Application Data for the purpose of identity fraud detection utilizing supervised machine learning methods. The tools used for analysis are Python and Microsoft Excel, and the modeling methodologies implemented include Logistic Regression, Decision Tree, Random Forest, Gradient Boosted Trees, and Neural Network.

The original application data set comprises 1,000,000 unique records containing information about the date of the application, social security number, name, address, five-digit zip code, date of birth, home phone number that each applicant reported as of the end of 2016, and a fraud label indicating whether each application is found fraudulent or not. The general analysis process includes preliminary data cleaning, creating expert variables, in-time and out-of-time data separation, feature selection (filter and wrapper methods), train-test splitting, applying supervised fraud algorithms, comparing model results, and deciding & evaluating the best-performing model.

By feeding the selected expert variables into models and examining the performance of the aforementioned supervised algorithms at a 3% penetration rate, we found that Neural Network yielded the optimal model on the out-of-time dataset with a fraud detection rate of 55.62%, while Logistic Regression produced the lowest accuracy on the out-of-time data with a detection rate of only 52.75%. Overall, both the Gradient Boosted Trees and Neural Network methods demonstrated high accuracy in fraud detection with relatively stable performance across all experiments.

DATA OVERVIEW

The product application dataset includes information about the date of the application, social security number, name, address, five-digit zip code, date of birth, home phone number that each applicant reported as of the end of 2016, and a fraud label indicating whether each application is found fraudulent or not. The dataset consists of 1,000,000 records and 10 categorical fields.

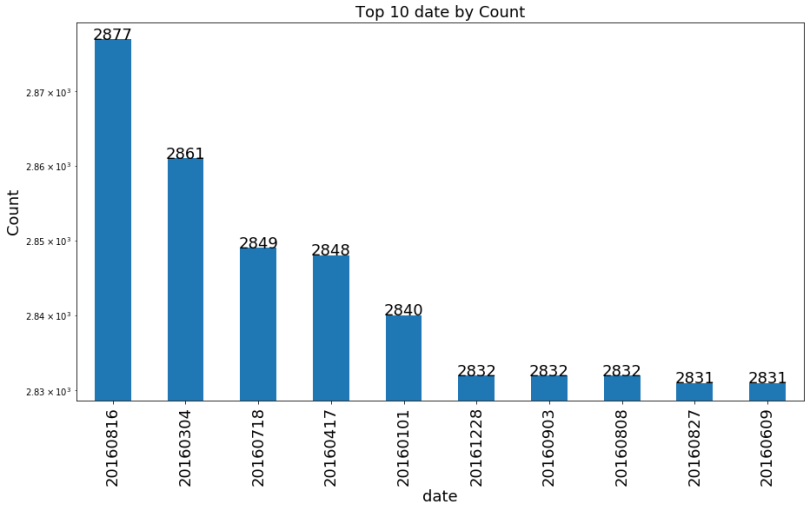
Below are summary tables and descriptions of the variables considered to be most pertinent to our analysis, excerpted from the full data quality report which can be found in the Appendix.

SUMMARY STATISTICS OF KEY VARIABLES

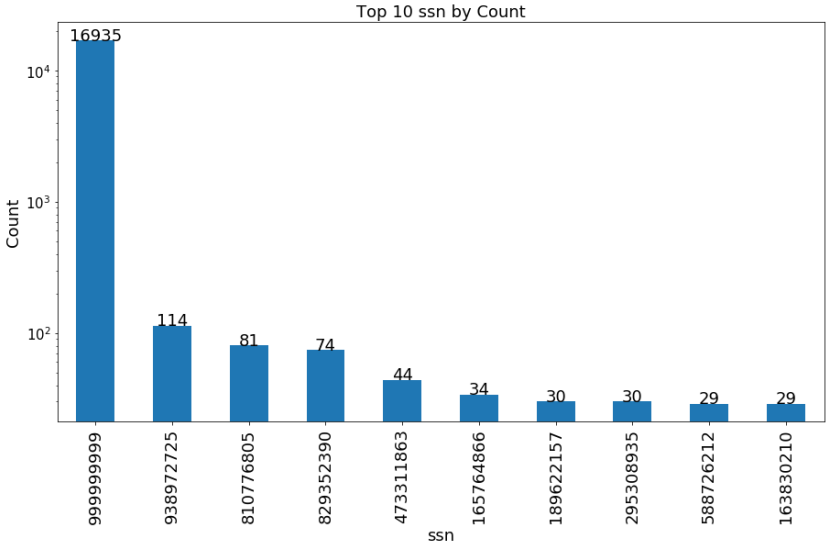
Field name	Field type	# of non-NA values	% populated	# unique values	Most Common Field
record	Categorical	1,000,000	100%	1,000,000	NA
date	Date	1,000,000	100%	365	20160816
ssn	Categorical	1,000,000	100%	835,819	999999999
firstname	Categorical	1,000,000	100%	78,136	EAMSTRMT
lastname	Categorical	1,000,000	100%	177,001	ERJSAXA
address	Categorical	1,000,000	100%	828,774	123 MAIN ST
zip5	Categorical	1,000,000	100%	26,370	68138
dob	Date	1,000,000	100%	42,673	19070626
homephone	Categorical	1,000,000	100%	28,244	9999999999
fraud_label	Boolean	1,000,000	100%	2	0

record: A unique record number for referencing each product application in the data, ranging from 1 to 1,000,000.

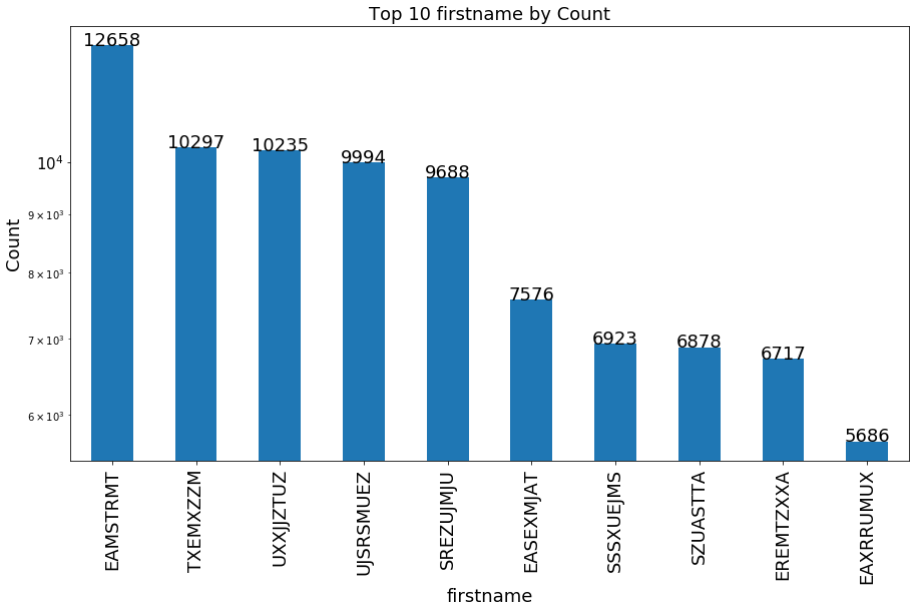
date: Date of the application in the format of Year-Month-Day, with 365 unique values ranging from 2016-01-01 to 2016-12-31.



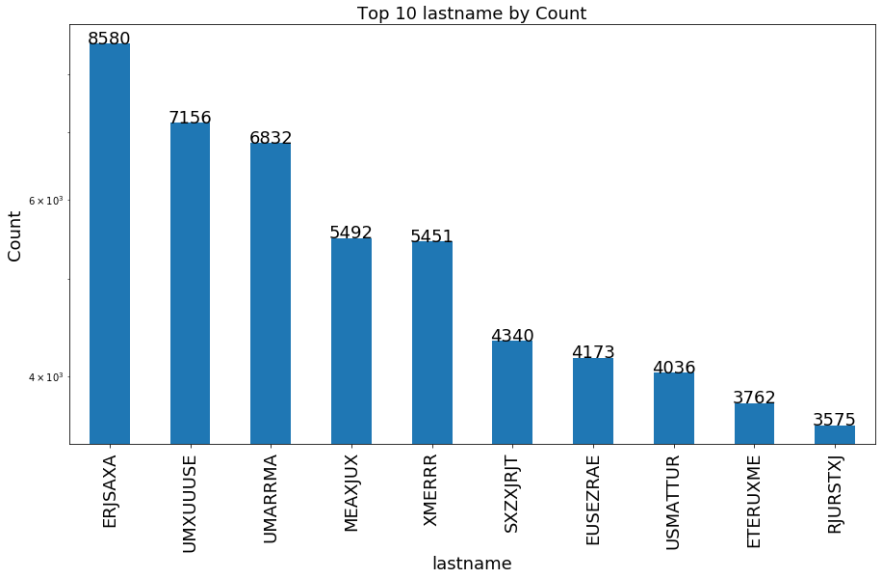
ssn: social security number of the applicant, with 835,819 distinct values ranging from 36 to 999999999 and 999999999 being the most common.



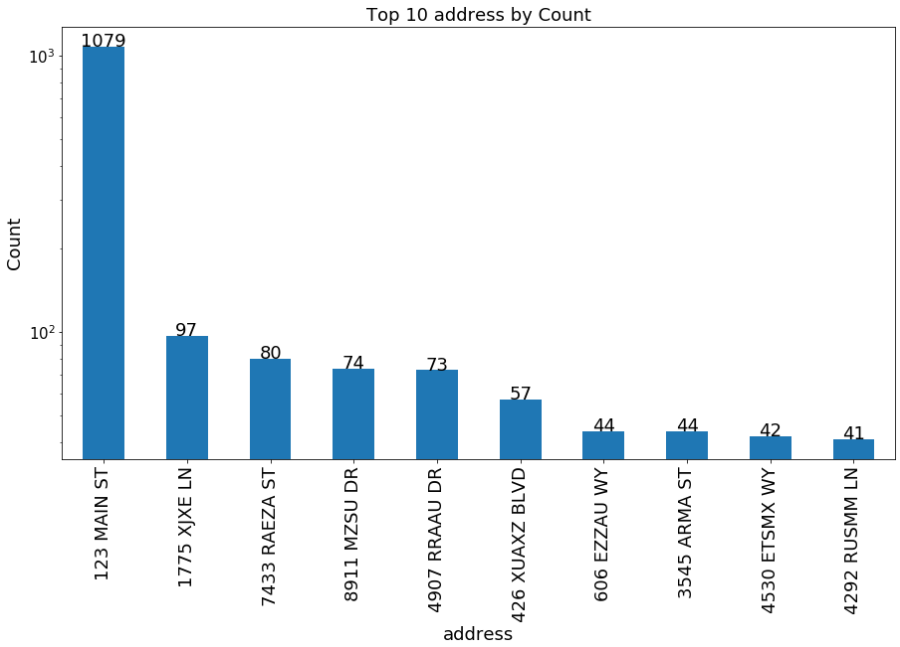
firstname: denotes the first name of the applicant, with 78,136 distinct values and EAMSTRMT being the most common.



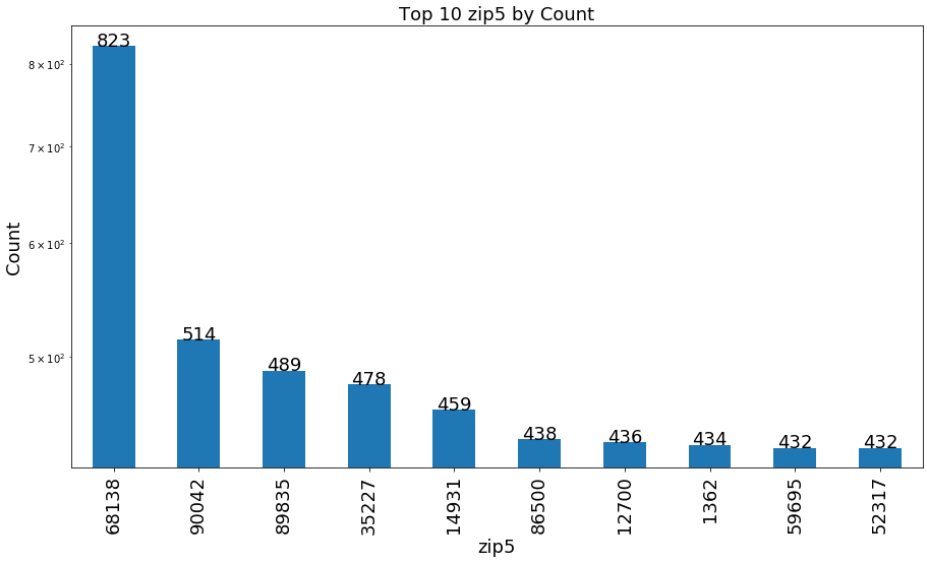
lastname: stands for the last name of the applicant, with 177,001 distinct values and ERJSAXA being the most common.



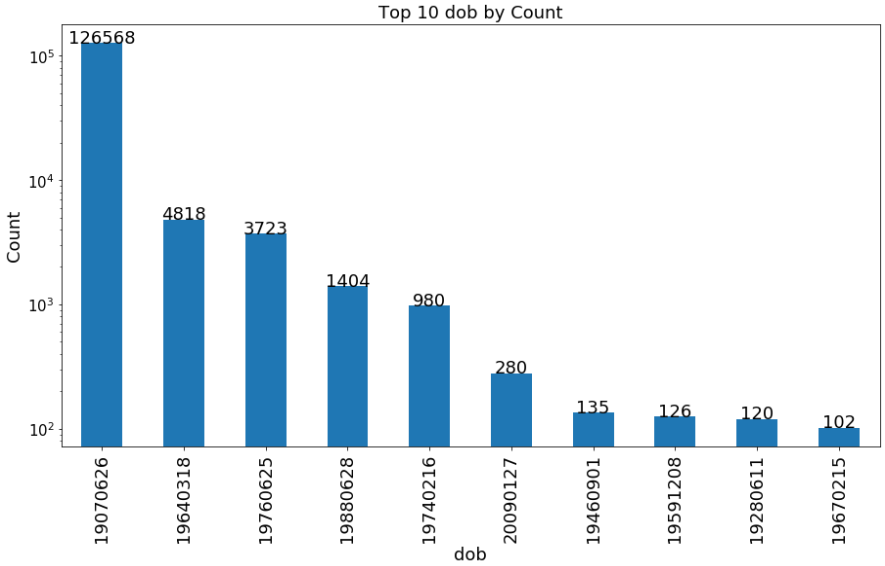
address: shows the home address of the applicant, with 828,774 unique addresses and 123 MAIN ST being the most common.



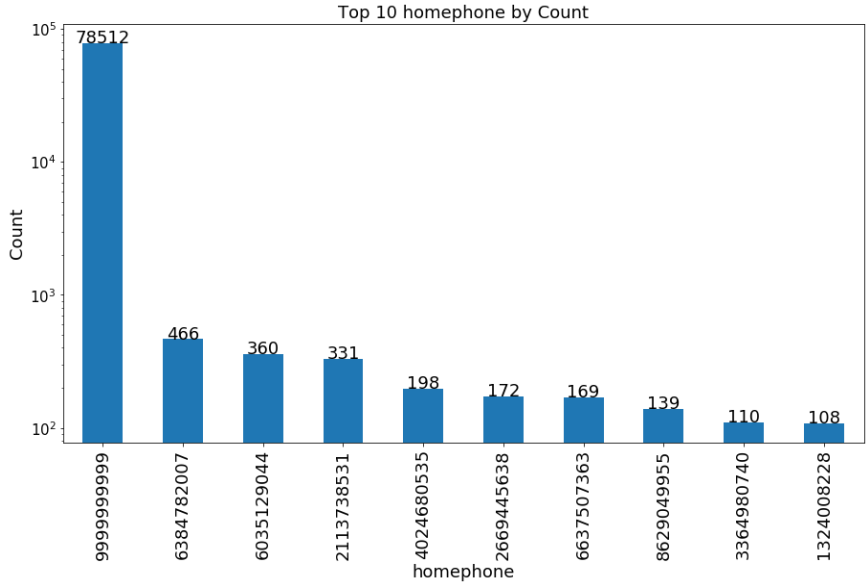
zip5: five-digit zip code in which the applicant's address is located, with 26,370 unique values.



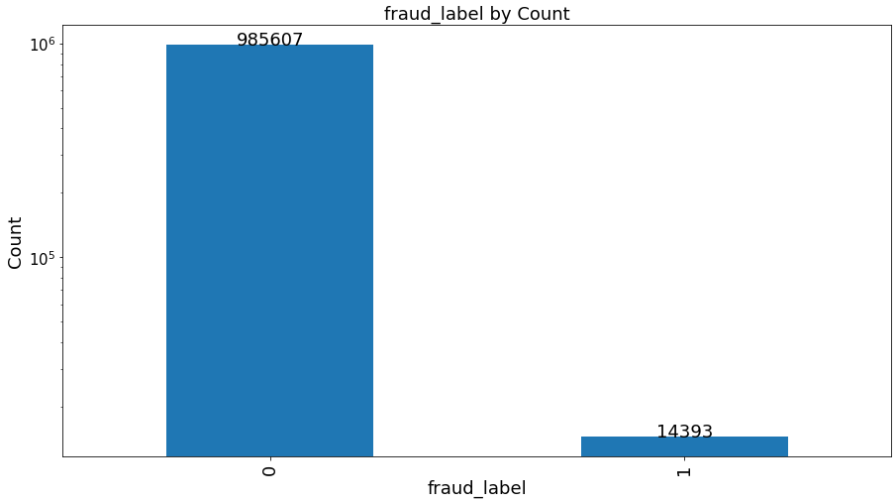
dob: Used to denote the date of birth of the applicant following the format of Year-Month-Day, with 42,673 distinct dates and 19070626 being the most common.



homephone: home phone number entered by the applicant when submitting the application, with 28,244 unique values and 999-999-9999 being the most common phone number.



fraud_label: A label created to indicate whether an application is fraudulent or not. It takes two values, where 1 = fraudulent, 0 = honest.



DATA CLEANING

Before we created our expert variables, preliminary data cleaning was conducted to prepare the dataset for proper analysis and modeling. The primary purpose of this process is to ensure that we:

1. Remove the effects of any frivolous values
2. Convert the fields to the correct format for modeling

FRIVOLOUS VALUES

While there were no null values in any of the fields, we found that certain frivolous values occurred in fields **ssn**, **dob**, **address**, and **homephone**, as seen in the table below:

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

It is important to treat these frivolous values so that they do not interfere with the accuracy of our models later. For example, the ssn 999999999 occurs around 17,000 times, which will certainly set off some velocity metrics in our fraud detection models, as it is unlikely that a person with that ssn applied for credit that many times within a year.

To eliminate the impacts posed by frivolous data in these fields, we simply replaced the values with its corresponding record number plus some constant (depending on the field). For example:

Field	Record	Constant	Updated Value
ssn	1234	1	1235
dob	22922	3	22923

DATETIME

We also converted the data type in the date field from integers to datetime. The motivation behind this step was to make the data easier to work with later when creating the variables, as the variable creation process is time sensitive.

VARIABLE CREATION

There are three main methods of identity fraud: identity theft, identity manipulation and synthetic identity. Each fraudulent application can include one or more of these methods, where the fraudster uses stolen information (i.e. ssn), altered identities (i.e. using the same address but changing the phone number) or entirely made up identities. Hence, it is important to create variables that will allow us to detect all of these types of identity fraud by linking different fields together.

ATTRIBUTES

Our first step is to define a list of attributes. Attributes are either a raw field, or a combination of raw fields. In essence, each attribute can be thought of as an identity, which we can later use in creating our expert variables for fraud detection.

Attribute	Description	Example
ssn	ssn	379070012
address	address	141 South St
dob	dob	19950101
homephone	homephone	9493921021
name	firstname + lastname	John Smith
address-zip5	address + zip5	141 South St 92001
name-dob	name + dob	John Smith 19950101
name-address-zip5	name + address + zip5	John Smith 141 South St 92001
name-homephone	name + homephone	John Smith 9493921021
address-zip5-dob	address + zip5 + dob	141 South St 92001 19950101
address-zip5-homephone	address + zip5 + homephone	141 South St 92001 9493921021
dob-homephone	dob + homephone	19950101 9493921021
homephone-name-dob	homephone + name + dob	9493921021 John Smith 19950101

We also created additional attributes by combining ssn to each attribute (other than the ssn itself) above, giving us a total of 25 attributes.

VARIABLES

Our main logic for fraud detection is to look at how frequently an identity (which is synonymous to attributes we defined above) is used to apply for credit. To do so, we create the following groups of variables which will allow our models to systematically detect fraudulent applications and identities.

VELOCITY VARIABLES

The number of times an identity (attribute) was seen over the previous n days. We chose the following days for n: 0, 1, 3, 7, 14 and 30. A high velocity value indicates that the identity may be fraudulent because the person has used that identity to apply for a large number of credit applications.

To calculate this variable for an attribute, we look back at the past n days, and count the number of records with the same attribute value during this timeframe. We calculated the velocity period for each of the 6 time periods mentioned above, for all 25 attributes. Below are a few examples illustrating these velocity variables:

Velocity Variable	Description
ssn//prev_d0_count	The number of times a particular ssn was used to apply for credit in the current day
name-dob//prev_d1_count	The number of times a particular name and date of birth combination was used to apply for credit over the past day
name-homephone//prev_d14_count	The number of times a particular name and homephone combination was used to apply for credit over the past 14 days

** For a full list of velocity variables, see the appendix B1*

We created 150 (25 attributes x 6 velocity periods) velocity variables in total.

RELATIVE VELOCITY VARIABLES

While it is important to look at the absolute velocity of each attribute, it is also important to consider the relative velocity as well, which is essentially the number of times an attribute has appeared in the day divided by the number of times that same attribute appeared in the past n days (where n = 3, 7, 14, 30). This will help us better identify whether a velocity value is truly high, or it simply appears high and all of the velocity for that grouping (attribute) is high.

To calculate the relative velocity for each attribute, we take the 1-day velocity of that attribute and divided by its corresponding 3, 7, 14 and 30-day velocity. Below are a few examples:

Relative Velocity Variable	Description	Mathematical Formula
ssn//prev_d1_d3_avg	The number of times a particular ssn appeared over the past day relative to the number of times that same ssn has appeared over the past 3 days.	$\frac{ssn//prev\_d1\_count}{ssn//prev\_d3\_count}$
name-dob//prev_d1_d14_avg	The number of times a particular name and date of birth combination appeared over the past day relative to the number of times that same combination has appeared over the past143 days.	$\frac{name-dob//prev\_d1\_count}{name-dob//prev\_d14\_count}$

** For a full list of relative velocity variables, see the appendix B2*

We created 100 (25 attributes * 4 relative velocity periods) relative velocity variables in total.

DAYS SINCE VARIABLES

The number of days since we last saw an attribute. A low day since value indicates that the identity may be fraudulent in nature, because the person is frequently applying for credit within a short period of time.

To calculate this variable, we go back in time and find the last day in which the identity (attribute) has appeared in our data, the difference between the current day and the previous day in which the same identity appeared is the days since value.

Days Since Variable	Description
ssn//days_since	The number of days since we last saw a particular ssn.
dob-homephone//days_since	The number of days we last saw a particular date of birth and homephone combination.

** For a full list of days since variables, see the appendix B3*

We created 25 (one for each attribute) days since variables in total.

RISK TABLE (DAY OF WEEK) VARIABLE

We want to incorporate the effect of the day of week in our models. However, since day of week is a categorical variable (i.e. Monday, Tuesday, Wednesday), we must encode it into a numerical value since our algorithms cannot directly take in categorical variables. To do so, we utilized the target encoding method and created a risk table for the day of week and used that table to assign a numerical value for each day of week.

To calculate our risk table, we simply calculated the average fraud rate for each day of week:

Day of Week	Risk Value
Monday	0.0135
Tuesday	0.0141
Wednesday	0.0151
Thursday	0.0150
Friday	0.0145
Saturday	0.0150
Sunday	0.0137

Afterwards, we will assign a value of 0.0135 for all records on Monday, 0.0141 for all records on Tuesday, and so on. After this process, we can use this Risk Table variable in our models.

STANDARDIZATION

The final step in our variable creation process is to standardize (using Z-score normalization) all of the variables we've created. This is important because the variables that we've created will vary in magnitudes, units and range. Since many of our machine learning models use Euclidean distance in its calculations, we want variables of different magnitudes to weigh evenly within the model. To do so, we essentially rescale the variables so that all of the values in for that variable have a mean of 0 and standard deviation of 1 using the following formula:

$$z = \frac{x - \mu}{\sigma}$$

Where x is the raw variable value, μ is the mean (average) of that variable, σ is the standard deviation of that variable, and z is the standardized variable value.

OUT-OF-TIME AND IN-TIME SPLIT

The data consists of time dependent records because all of the variables we created are time sensitive. Therefore, in order to estimate how our models would generalize, we divided the data into first 10 months as in-time and last 2 months as out-of-time data. The logic behind this is because we want to build our models from past data (first 10 months) to predict future cases (last 2 months) where our models have not seen before. Since most of our variables rely on what came before, and the first two weeks of data lack that information, we excluded the first two weeks of data for the in-time split. With the data properly split, we can proceed to feature selection.

FEATURE SELECTION

During the feature selection process, we first calculated the univariate Kolmogorov-Smirnov (KS) value and univariate Fraud Detection Rate (FDR) at 3% of the population for each variable via Python and sorted all of our candidate variables by both measures. For initial filtering, we selected the top 80 variables rank-ordered by the average rank. Then we used backward selection to further reduce them to 34 variables for our final models.

The main reasons for us conducting feature selection are 1) data could get sparse in high dimensions, especially considering the moderate volume of data (1 million records) we had, 2) as dimensionality increases, almost all data points could become outliers, 3) for nonlinear models we will be experimenting with, high dimensions could mean fitting noise rather than the true nonlinearities.

UNIVARIATE FEATURE SELECTION

KOLMOGOROV-SMIRNOV (KS)

KS value is a robust method to measure how well two distributions are separated by calculating the maximum distance between them:

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

The larger the KS, the more separated the two distributions. In the context of fraud detection, this means measuring the distance between the good records and fraudulent records for each of our candidate variables. Specifically, we used the ks.2samp function from the scipy.stats library to get a sorted list of our 276 variables with their corresponding ks scores.

FRAUD DETECTION RATE (FDR)

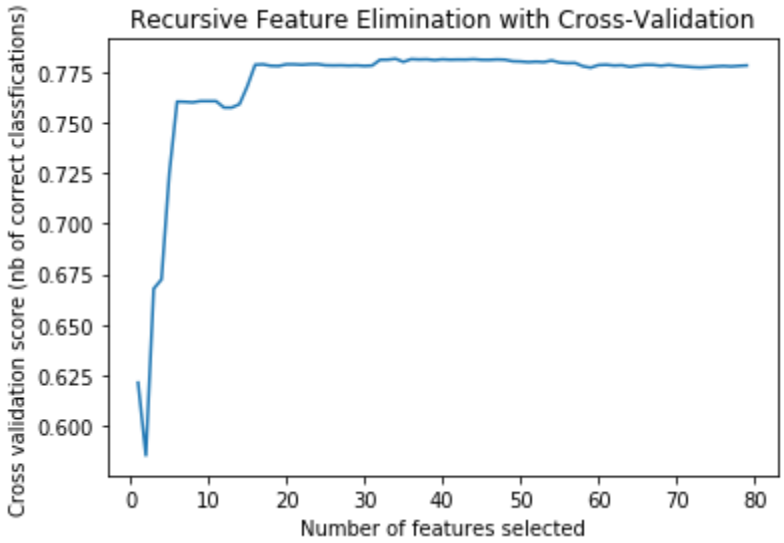
In the meantime, another measure of goodness we introduced was Fraud Detection Rate at a 3% cutoff. This means calculating the fraction of frauds caught / total number of frauds within 3% of the population for each candidate variable:

$$Fraud\ Detection\ Rate\ =\ \frac{Number\ of\ Frauds\ Caught}{Total\ Number\ of\ Frauds\ in\ Population}$$

Having obtained two ordered lists of the scores, the top 80 candidate variables ordered by the average rank of the two measures above were selected during the filtering stage.

WRAPPER METHOD: BACKWARD SELECTION

After narrowing down our candidate variables to 80, we introduced a wrapper method to further reduce the feature size for the sake of later modeling. Specifically, we conducted backward selection which starts with modeling with all the variables selected, testing the deletion of each variable based on a certain model fit criterion, removing the least statistically significant one at a step, and repeating the process until no more variables meet the criterion. The graph below shows how the cross-validation score changed as the number of variables fed into logistic regression changed.



At this stage, we started off with 80 variables and conducted backward selection on a logistic regression model as our baseline model in Python. The wrapper method in the end left us with 34 variables to proceed with for our final models. The chart below shows the final 34 variables (at the end of our feature selection process) out of the initial 276 variables.

Field
fraud_label
address-zip5//prev_d30_count
address//prev_d30_count
address-zip5//days_since
address//days_since
address//prev_d14_count
address-zip5//prev_d14_count
address//prev_d7_count
address-zip5//prev_d7_count
address//prev_d3_count
address-zip5//prev_d3_count
address//prev_d1_count
address-zip5//prev_d1_count
address-zip5-homephone//prev_d30_count
ssn-dob//prev_d30_count
name-dob//prev_d30_count
ssn//prev_d30_count
ssn-name-dob//prev_d30_count
ssn-name//prev_d30_count
address-zip5-homephone//days_since
address-zip5-homephone//prev_d14_count
ssn-dob//days_since
name-dob//days_since
ssn//days_since
name-dob//prev_d14_count
ssn-name-dob//days_since
ssn-name//days_since
ssn-dob//prev_d14_count
ssn-name-dob//prev_d14_count
ssn//prev_d14_count
name//prev_d30_count
ssn-name//prev_d14_count
address//prev_d1_d14_avg
address-zip5//prev_d1_d14_avg

TRAIN-TEST SPLIT

With the out-of-time data set aside as a final validation set, we took the in-time data, randomized the order and further divided it in a 7.5:2.5 ratio into train and test set. The reason we randomize here is to make sure the train set will have a good mixture of records from the in-time records.

ISSUE WITH UNBALANCED CLASS

After the previous splits, the count of fraud records is 9,189 but the count of non-fraud records is 626,807. The problem here is if we proceed to training the model, the model will probably try to minimize the overall error and have a poor performance predicting frauds which is essentially the sole purpose of the problem. To remedy this issue, the number of records per class should have a similar number of records. Additionally, since our training data is still relatively large, we downsampled the training records down to a 1:1 ratio with the fraud records by selecting randomly 9189 non-fraud records to match the same number of fraud records. After this process, we are ready to move on to model training phase with:

- 1:1 non-fraud vs fraud ratio downsampled data for training
- Train data with the original 75% of the in-time data
- Test data with 25% of the in-time data
- Out-of-time data consists of the last two months of dataset

FRAUD ALGORITHMS

For the purpose of this project, we performed 5 supervised machine learning algorithms to calculate the Fraud Detection Rate (FDR) using the variables created:

- Logistic Regression
- Decision Tree
- Gradient Boosted Trees
- Random Forest
- Neural Net

For the rest of the section, we will briefly discuss how each of these algorithms works and the detailed implementation in our project.

LOGISTIC REGRESSION

DEFINITION

Logistic regression is a supervised classification method to delineate the relationship between a dependent variable and numerous independent variables when the dependent variable is categorical. It models the conditional probability f (x) = P(Y=1|X=x). In the context of the project, the dependent variable is binary (either 0 or 1), therefore we believe it to be good and simple enough to begin with as our baseline.

APPLICATION

Through the previous feature selection, we finally kept 34 variables. By changing the number of variables, we tried to modify the logistic regression model. At first, we selected all the 34 variables. Then, we fitted the logistic model with the first 30 variables. We further decreased the number of variables from 30 to 15, with a step of 5. The reason to fit the model with different numbers of variables is that more variables mean more noise and thus would possibly negatively impact the model performance. The table below showcases the LR prediction accuracy for FDR at 3% for each of the three data sets.

Logistic Regression	**Parameters**		**Average FDR at 3%**		
Trial	Total Variables	# Variables Selected	Train	Test	OOT
1	34	34	0.5478	0.5384	0.513
2	34	30	0.5535	0.5434	0.5203
3	34	25	0.5616	0.5508	0.5253
4	34	20	0.5619	0.551	0.5275
5	34	15	0.3992	0.3802	0.353

DECISION TREE

DEFINITION

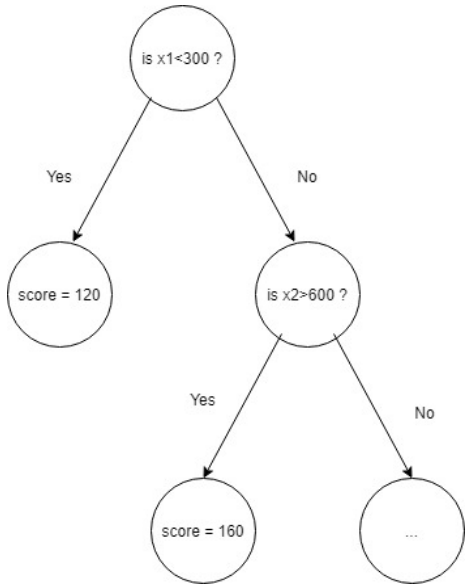
A decision tree splits the feature spaces into multiple high-dimensional boxes and decides the boundaries of each box and what the score in each box should be. It can be linearized into a set of decision rules, such as:

if x1 < 300, score = 120,

If x1>300 and x2 > 600, score = 160

...

Alternatively, it can be represented graphically by a tree-like structure:



APPLICATION

To build a single decision tree for our data, we loaded the DecisionTreeClassifier module from sklearn via Python with the following parameters for tuning:

max_features	= xx to specify number of columns to include in the tree
max_depth	= xx to specify maximun number of levels in a tree.
min_samples_leaf	= xx to specify the minimum number of samples for each split.
min_samples_split	= xx to specify the minimum number of samples required to be at a leaf node

As a result, the optimal model after tuning had 27 features, max depth of 15, and min_samples_leaf 156 and the min_samples_split 466, giving a FDR of 0.5394 for out-of-time data. The table below shows the decision tree prediction accuracy for FDR at 3% for each of the three data sets.

Model	Parameters			Average FDR at 3%		
Decision Tree	Total Variables	Min Samples Split	Min Samples Leaf	Train	Test	OOT
1	34	300	60	0.6144	0.5611	0.5356
2	34	466	154	0.5853	0.5618	0.5394

GRADIENT BOOSTED TREES

DEFINITION

Gradient boosted trees is a method producing a prediction model in the form of a series of weak learning decision trees. It builds the model in a stage-wise fashion and generalizes them through optimizing an arbitrary differentiable loss function.

APPLICATION

Similar as the implementation of the single decision tree classifier, we loaded the xgboost package with the following parameters for tuning:

n_estimators = xx to specify number of trees in the series.

max_depth = xx to specify maximun number of levels in a tree.

colsample_bytree = xx to specify subsample ratio of columns constructing each tree

gamma = xx to specify minimum loss required to make a partition

As a result, the optimal model after tuning had 100 trees, max depth of 5 for each individual tree, and colsample by tree was 0.8 and the gamma was 1, giving a FDR of 0.5511 for out-of-time data. The table below shows the xgboost prediction accuracy for FDR at 3% for each of the three data sets.

Model	Parameters				Average FDR at 3%		
XGBoost	Total Variables	Max Depth	Min Child Weight	Learning Rate	Train	Test	OOT
1	34	5	5	5	0.5855	0.5695	0.5511
2	34	5	2	0.02	0.5926	0.5719	0.5478
3	34	7	5	0.02	0.5979	0.5561	0.5461
4	34	12	3	0.3	0.6234	0.5545	0.531

RANDOM FOREST

DEFINITION

Random forest is a supervised classification algorithm built on many strong decision trees, each associated with a certain degree of randomness. Contrary to boosted trees, It acts as an ensemble of strong learners, each with the full ability to predict the output. The final output is an average or vote across all the strong models. Since each individual tree in the forest grows fully and overfits differently, random forest can limit the drawback of overfitting by averaging the mistakes and errors across the whole collection given a sufficient number of trees. Thus, we believe random forest tends to be more stable and robust.

APPLICATION

In our case of model creation, we used the ensemble.RandomForestClassifier function from sklearn with the following parameters for tuning:

n_estimators = xx to specify the number of trees in the random forest.

max_features = xx to specify the number of features to consider at every split.

max_depth = xx to specify maximum number of levels in a tree.

min_samples_split = xx to specify minimum number of samples required to split nodes.

min_samples_leaf = xx to specify minimum number of samples required at each leaf node.

As a result, the optimal model after tuning had 100 trees, max features of 10, max depth of 5 for each individual tree, and minimal sample size of 3 at each leaf node, giving a FDR of 0.5349 for out-of-time data. The table below shows the random forest prediction accuracy for FDR at 3% for each of the three data sets.

Model	**Parameters**					**Average FDR at 3%**		
Random Forest	Total Variables	# Estimators	Max Depth	Max Features	Min Node	Train	Test	OOT
1	34	50	20	10	3	0.6297	0.5327	0.4588
2	34	100	5	5	3	0.564	0.5512	0.5349
3	34	100	10	5	3	0.5826	0.5611	0.519
4	34	100	10	10	3	0.5833	0.559	0.5219
5	34	200	10	10	3	0.5828	0.5585	0.5215
6	34	100	20	10	3	0.6289	0.5334	0.5013

NEURAL NET

DEFINITION

Our last implemented method is a neural net classifier, which takes the independent variables to form the input layer with N number of nodes, and forwards feed through L number of hidden layer(s) where each layer can have its own number of nodes. The final output layer in our case consists of one node which outputs a binary class of 0 or 1.

The nodes all have an activation function which takes input from the previous layer and after the activation function will decide whether to 'fire' to the next node or not. Finally, the gradients with respect to the weights and biases of the model are back propagated and all of the weights and biases are updated with the negative gradient multiplied by a self-set learning rate. This whole process is repeated a preset number of epochs which is usually set to the right number of times when the testing set has the lowest loss. Neural networks generalize well in situations when training data is abundant which we believe is the case for this problem.

APPLICATION

To form our neural network, we used the neural_network.MLPClassifier function from sklearn with the following parameters for tuning:

solver = xx to specify the solver for weight optimization.

alpha = xx to specify the L2 penalty (regularization term).

hidden layers = xx to specify the hidden layers sizes.

learning_rate = xx to specify the learning rate for weight updates.

activation function = xx to specify the activation function for the hidden layer(s).

As a result, the optimal neural net after tuning takes an alpha of 0.000001, a lbfgs solver, (15, 20) hidden layers, a constant learning rate and the rectified linear unit activation function, giving a FDR of 0.5562 for out-of-time data, which is also by far the highest detection rate among all models. The table below shows the NN prediction accuracy for FDR at 3% for each of the three data sets.

Model	**Parameters**						**Average FDR at 3%**		
Neural Net	Total Variables	Solver	Alpha	Hidden Layers	Learning Rate	Activation Function	Train	Test	OOT
1	34	adam	1.00E-04	(15, 20)	Adaptive	tanh	0.5854	0.5657	0.5457
2	34	lbfgs	1.00E-05	(15, 20)	Constant	relu	0.576	0.5698	0.5528
3	34	lbfgs	1.00E-05	(6,)	Constant	relu	0.576	0.5698	0.5528

4	34	lbfgs	1.00E-04	(15, 20)	Constant	relu	0.5808	0.5688	0.5482
5	34	lbfgs	1.00E-05	(10,30)	Constant	relu	0.5784	0.5695	0.549
6	34	lbfgs	1.00E-06	(15, 20)	Constant	relu	0.5761	0.5677	0.5562

After implementing all five supervised learning methods on each segment of the dataset, we created a report comparing the optimal performance in predicting the FDR metrics at 3% across these models. Below are the overall results of the report:

Fraud Detection Rate at 3%			
Model	Training	Testing	OOT
Logistic Regression	0.5619	0.551	0.5275
Random Forest	0.564	0.5512	0.5349
Neural Net	0.5761	0.5677	0.5562
Decision Tree	0.5853	0.5618	0.5394
Boosted Trees	0.5855	0.5695	0.5511

We found overall at the 3% cutoff, the Neural Network had the best Fraud Detection rate of 55.62% and the Logistic Regression method had the poorest Fraud Detection rate of 52.75% on the out-of-time dataset.

RESULTS

Based on the overall table above, we concluded that the Neural Network model had the optimal overall fraud detection rate on the out-of-time segment. Below is our table of the top 20% population bins for the Neural Network model on the training, testing and out-of-time datasets, respectively. The columns in green on the left-hand side show statistics within each population bin, and the columns in purple on the right-hand side are cumulative statistics up to each population bin.

Training	# Records	# Goods	# Bads	Fraud Rate
	596247	587632	8615	0.01444871

Bin Statistics						Cumulative Statistics						
Population Bin%	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	5962	1335	4627	22.39%	77.61%	5962	1335	4627	0.23%	53.71%	53.48	0.29
2	5962	5713	249	95.82%	4.18%	11924	7048	4876	1.20%	56.60%	55.40	1.45
3	5963	5876	87	98.54%	1.46%	17887	12924	4963	2.20%	57.61%	55.41	2.60
4	5962	5890	72	98.79%	1.21%	23849	18814	5035	3.20%	58.44%	55.24	3.74
5	5963	5910	53	99.11%	0.89%	29812	24724	5088	4.21%	59.06%	54.85	4.86
6	5962	5920	42	99.30%	0.70%	35774	30644	5130	5.21%	59.55%	54.33	5.97
7	5963	5912	51	99.14%	0.86%	41737	36556	5181	6.22%	60.14%	53.92	7.06
8	5962	5917	45	99.25%	0.75%	47699	42473	5226	7.23%	60.66%	53.43	8.13
9	5963	5934	29	99.51%	0.49%	53662	48407	5255	8.24%	61.00%	52.76	9.21
10	5962	5923	39	99.35%	0.65%	59624	54330	5294	9.25%	61.45%	52.21	10.26
11	5963	5922	41	99.31%	0.69%	65587	60252	5335	10.25%	61.93%	51.67	11.29
12	5962	5915	47	99.21%	0.79%	71549	66167	5382	11.26%	62.47%	51.21	12.29
13	5963	5933	30	99.50%	0.50%	77512	72100	5412	12.27%	62.82%	50.55	13.32
14	5962	5918	44	99.26%	0.74%	83474	78018	5456	13.28%	63.33%	50.05	14.30
15	5963	5921	42	99.30%	0.70%	89437	83939	5498	14.28%	63.82%	49.53	15.27
16	5962	5914	48	99.19%	0.81%	95399	89853	5546	15.29%	64.38%	49.09	16.20
17	5962	5920	42	99.30%	0.70%	101361	95773	5588	16.30%	64.86%	48.57	17.14
18	5963	5918	45	99.25%	0.75%	107324	101691	5633	17.31%	65.39%	48.08	18.05
19	5962	5926	36	99.40%	0.60%	113286	107617	5669	18.31%	65.80%	47.49	18.98
20	5963	5919	44	99.26%	0.74%	119249	113536	5713	19.32%	66.31%	46.99	19.87

Training: Top 20% Population Bins for Neural Network Model

Testing	# Records	# Goods	# Bads	Fraud Rate
	198749	195878	2871	0.01444536

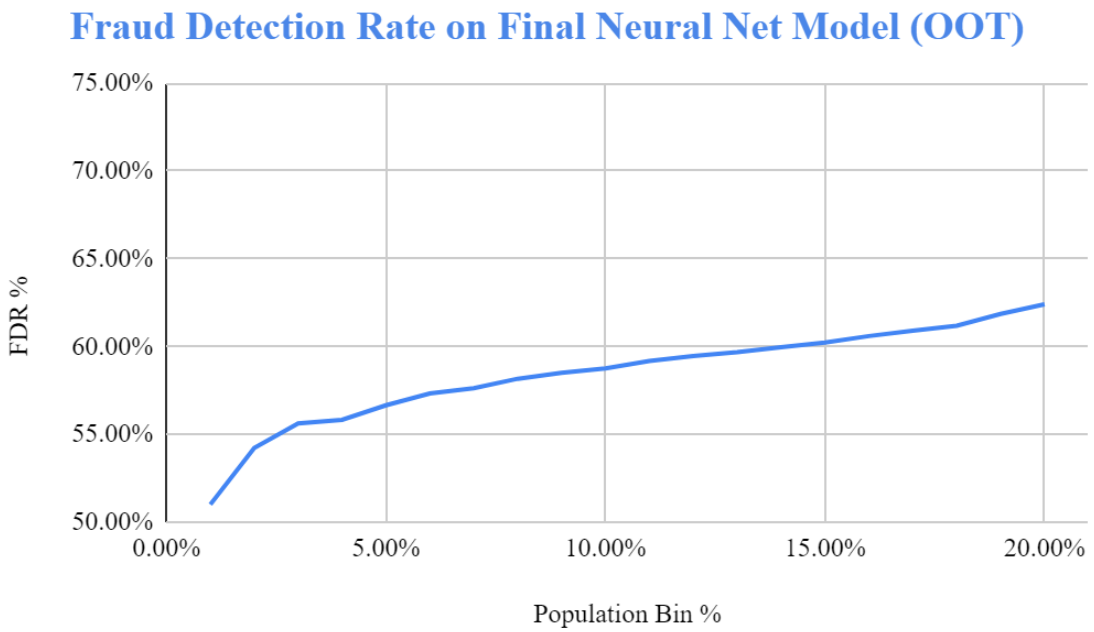
Bin Statistics						Cumulative Statistics						
Population Bin%	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	1987	456	1531	22.95%	77.05%	1987	456	1531	0.23%	53.33%	53.09	0.30
2	1987	1915	72	96.38%	3.62%	3974	2371	1603	1.21%	55.83%	54.62	1.48
3	1988	1961	27	98.64%	1.36%	5962	4332	1630	2.20%	56.77%	54.57	2.66
4	1987	1966	21	98.94%	1.06%	7949	6298	1651	3.20%	57.51%	54.31	3.81
5	1988	1972	16	99.20%	0.80%	9937	8270	1667	4.21%	58.06%	53.85	4.96
6	1987	1975	12	99.40%	0.60%	11924	10245	1679	5.21%	58.48%	53.27	6.10
7	1988	1978	10	99.50%	0.50%	13912	12223	1689	6.22%	58.83%	52.61	7.24
8	1987	1973	14	99.30%	0.70%	15899	14196	1703	7.23%	59.32%	52.09	8.34
9	1988	1976	12	99.40%	0.60%	17887	16172	1715	8.24%	59.74%	51.50	9.43
10	1987	1970	17	99.14%	0.86%	19874	18142	1732	9.25%	60.33%	51.08	10.47
11	1988	1980	8	99.60%	0.40%	21862	20122	1740	10.25%	60.61%	50.36	11.56
12	1987	1968	19	99.04%	0.96%	23849	22090	1759	11.26%	61.27%	50.01	12.56
13	1988	1980	8	99.60%	0.40%	25837	24070	1767	12.27%	61.55%	49.28	13.62
14	1987	1974	13	99.35%	0.65%	27824	26044	1780	13.28%	62.00%	48.72	14.63
15	1988	1976	12	99.40%	0.60%	29812	28020	1792	14.28%	62.42%	48.14	15.64
16	1987	1967	20	98.99%	1.01%	31799	29987	1812	15.29%	63.11%	47.82	16.55
17	1988	1972	16	99.20%	0.80%	33787	31959	1828	16.30%	63.67%	47.37	17.48
18	1987	1973	14	99.30%	0.70%	35774	33932	1842	17.31%	64.16%	46.85	18.42
19	1988	1973	15	99.25%	0.75%	37762	35905	1857	18.31%	64.68%	46.37	19.33
20	1987	1974	13	99.35%	0.65%	39749	37879	1870	19.32%	65.13%	45.81	20.26

Testing: Top 20% Population Bins for Neural Network Model

OOT	# Records	# Goods	# Bads	Fraud Rate
	166493	164107	2386	0.01433093

Bin Statistics						Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	1664	447	1217	26.86%	73.14%	1664	447	1217	0.27%	51.01%	50.73	0.37
2	1665	1588	77	95.38%	4.62%	3329	2035	1294	1.24%	54.23%	52.99	1.57
3	1665	1632	33	98.02%	1.98%	4994	3667	1327	2.20%	55.62%	53.42	2.76
4	1665	1660	5	99.70%	0.30%	6659	5327	1332	3.20%	55.83%	52.63	4.00
5	1665	1645	20	98.80%	1.20%	8324	6972	1352	4.21%	56.66%	52.45	5.16
6	1665	1649	16	99.04%	0.96%	9989	8621	1368	5.21%	57.33%	52.12	6.30
7	1665	1658	7	99.58%	0.42%	11654	10279	1375	6.22%	57.63%	51.41	7.48
8	1665	1652	13	99.22%	0.78%	13319	11931	1388	7.23%	58.17%	50.94	8.60
9	1665	1657	8	99.52%	0.48%	14984	13588	1396	8.24%	58.51%	50.27	9.73
10	1665	1659	6	99.64%	0.36%	16649	15247	1402	9.25%	58.76%	49.51	10.88
11	1665	1655	10	99.40%	0.60%	18314	16902	1412	10.25%	59.18%	48.93	11.97
12	1665	1658	7	99.58%	0.42%	19979	18560	1419	11.26%	59.47%	48.21	13.08
13	1665	1660	5	99.70%	0.30%	21644	20220	1424	12.27%	59.68%	47.41	14.20
14	1665	1658	7	99.58%	0.42%	23309	21878	1431	13.28%	59.97%	46.69	15.29
15	1664	1658	6	99.64%	0.36%	24973	23536	1437	14.28%	60.23%	45.95	16.38
16	1665	1656	9	99.46%	0.54%	26638	25192	1446	15.29%	60.60%	45.31	17.42
17	1665	1658	7	99.58%	0.42%	28303	26850	1453	16.30%	60.90%	44.60	18.48
18	1665	1658	7	99.58%	0.42%	29968	28508	1460	17.31%	61.19%	43.88	19.53
19	1665	1649	16	99.04%	0.96%	31633	30157	1476	18.31%	61.86%	43.55	20.43
20	1665	1652	13	99.22%	0.78%	33298	31809	1489	19.32%	62.41%	43.09	21.36

Out-of-time (OOT): Top 20% Population Bins for Neural Network Model



CONCLUSION

DATA PREPARATION

Our first step involved cleaning the data by taking care of frivolous values in the ssn, dob, address and homephone fields. We also formatted the date field into the proper datetime data type. In order to create our variables, we first defined and created a list of attributes, which are either raw fields (such as ssn) or a combination of raw fields (such as name, which is firstname + lastname) which can be used as identifiers in the fraud detection process. After this step, we were able to create expert variables for each attribute, such as the velocity, relative velocity and days since variables.

FEATURE SELECTION

Our feature selection process is twofold. First, we calculated the univariate Kolmogorov-Smirnov (KS) value and Fraud Detection Rate (FDR) at 3% for each expert variable we created in the previous step. This allowed us to crudely identify the effectiveness of each variable in detecting fraud. After rank ordering our variables using both KS and FDR, we selected the top 80 variables to proceed into the second step: a wrapper method. For this step, we performed a stepwise backward selection, where we systematically removed variables one at a time based on its statistical significance. At the end of this process, we were left with 34 expert variables.

MODELING

Before modeling, we created several sets of training, testing and out-of-time (OOT) validation data. This is important in helping us mitigate some of the effects of statistical variation in our dataset.

For our modeling process, we tried to fit our data using the following 5 classification algorithms:

- Logistic Regression
- Decision Tree
- Gradient Boosted Trees
- Random Forest
- Neural Net

After using the models to predict the fraud labels, we calculated the FDR (for population bins 1% through 20%) for the training, testing and OOT validation data sets.

EVALUATION

After evaluating all 5 models, we chose the Neural Net model as our final model, as it had the highest FDR (at 3%). To better assess the model's performance, we looked at the penetration rates of each population bin (ranging from 1% to 20%) for the training, testing and validation datasets. We calculated various bin and cumulative statistics for our client to assess our chosen model.

KEY TAKEAWAYS

- Statistical variation is very important to consider. We noticed that, depending on the randomization of our train/test data split, the FDR (at 3%) of the same model can differ by as much as 3%.
- Our baseline Logistic Regression model had a FDR (at 3%) of 52.75%.
- The best performing Decision Tree model had a FDR (at 3%) of 53.94%.
- The best performing Gradient Boosted Trees model had a FDR (at 3%) of 55.11%
- The best performing Random Forest model had a FDR (at 3%) of 53.49%
- The best performing Neural Net model had an average FDR (at 3%) of 55.62%

FUTURE STEPS

Overall, because of the limited resources and data, we believe that there is room for improvement in our modeling process:

1. Data Availability and Granularity
    a. With any modeling process, we can almost always benefit from the availability of more data, as we will be able to build more variables.
    b. The granularity of the data is also very important. While we had daily granularity for credit application data, we could potentially benefit from minute-level granularity, as there may be specific intraday patterns/seasonalities that are lost when data is grouped into days.
2. Feature Selection
    a. While we performed a stepwise selection method to arrive at our 34 final variables, there is a large amount of randomness involved in how the variables are selected, which could drastically impact our modeling step later on.
    b. With the availability of more time, we'd like to look into more advanced feature selection methods (as well as try other types of wrapper methods).
3. Training/Testing Data
    a. A large part of the variation in our model performance is a result of the randomization in our training/testing data splits. We want to explore other ways of sampling our data, such as cross-validation and bootstrapping,

which is more time intensive but will likely help us lower the statistical variation of our dataset.

4. Resampling Methods
    a. For simplicity, we chose to downsample the goods in our data set to be equal to bads (frauds). It would be interesting to try out other good to bad ratios as well as explore upsampling the bads instead.

APPENDIX

A - DATA QUALITY REPORT: PRODUCT APPLICATION DATA

PART I - DATA DESCRIPTION

This product application dataset includes information about the date of the application, social security number, name, address, five-digit zip code, date of birth, home phone number that each applicant reported as of the end of 2016, and a fraud label indicating whether each application is found fraudulent or not. The dataset consists of 1,000,000 records and 10 categorical fields. The detailed description of the dataset can be found as follows.

PART II - SUMMARY TABLE OF FIELDS

Below shows the summary table listing all the 10 fields for the product application dataset.

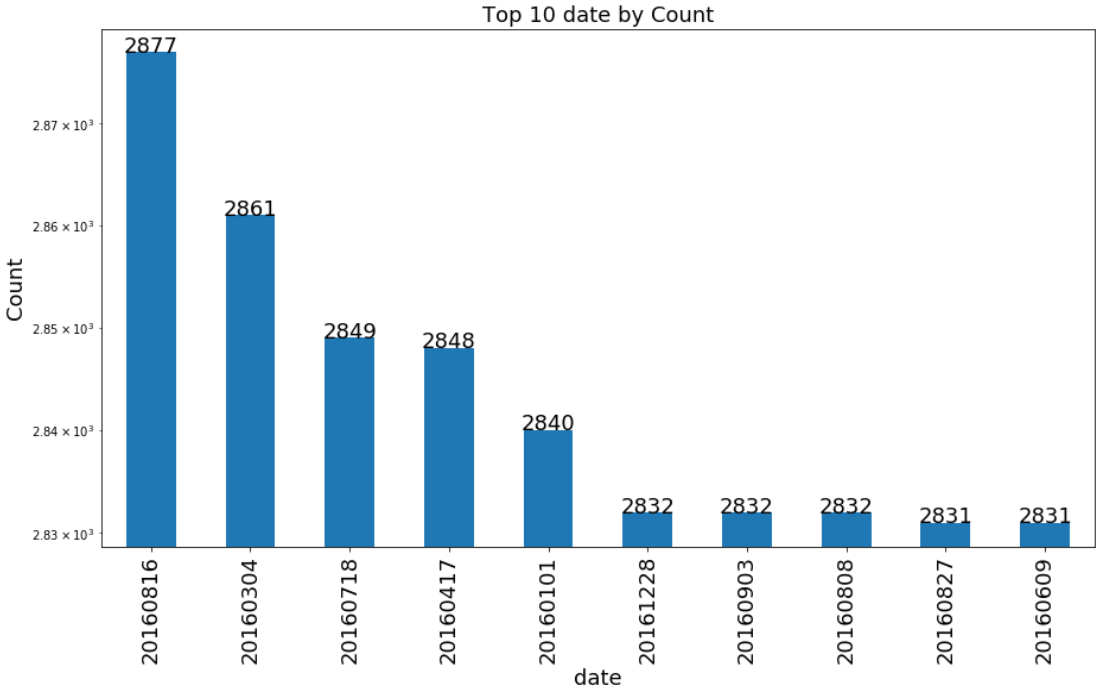
Field name	Field type	# of non-NA	% populated	# unique values	Most Common
record	Categorical	1,000,000	100%	1,000,000	NA
date	Categorical	1,000,000	100%	365	20160816
ssn	Categorical	1,000,000	100%	835,819	999999999
firstname	Categorical	1,000,000	100%	78,136	EAMSTRMT
lastname	Categorical	1,000,000	100%	177,001	ERJSAXA
address	Categorical	1,000,000	100%	828,774	123 MAIN ST
zip5	Categorical	1,000,000	100%	26,370	68138
dob	Categorical	1,000,000	100%	42,673	19070626
homephone	Categorical	1,000,000	100%	28,244	9999999999
fraud_label	Categorical	1,000,000	100%	2	0

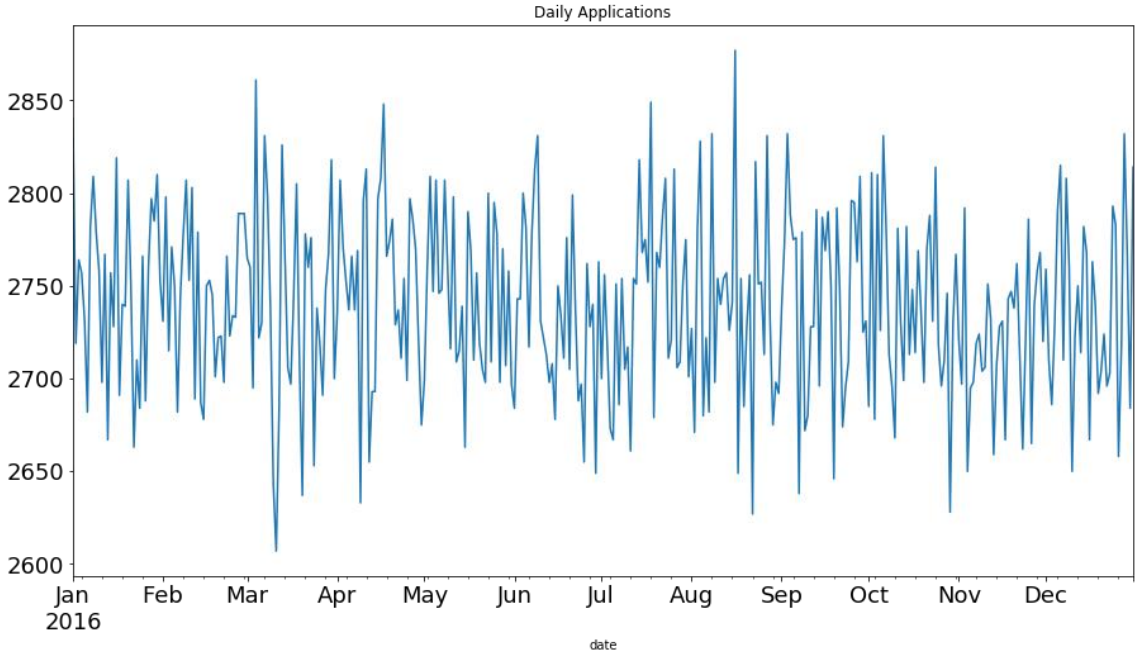
Table 1: Categorical Fields of Product Application Data

PART III - FIELD DESCRIPTION

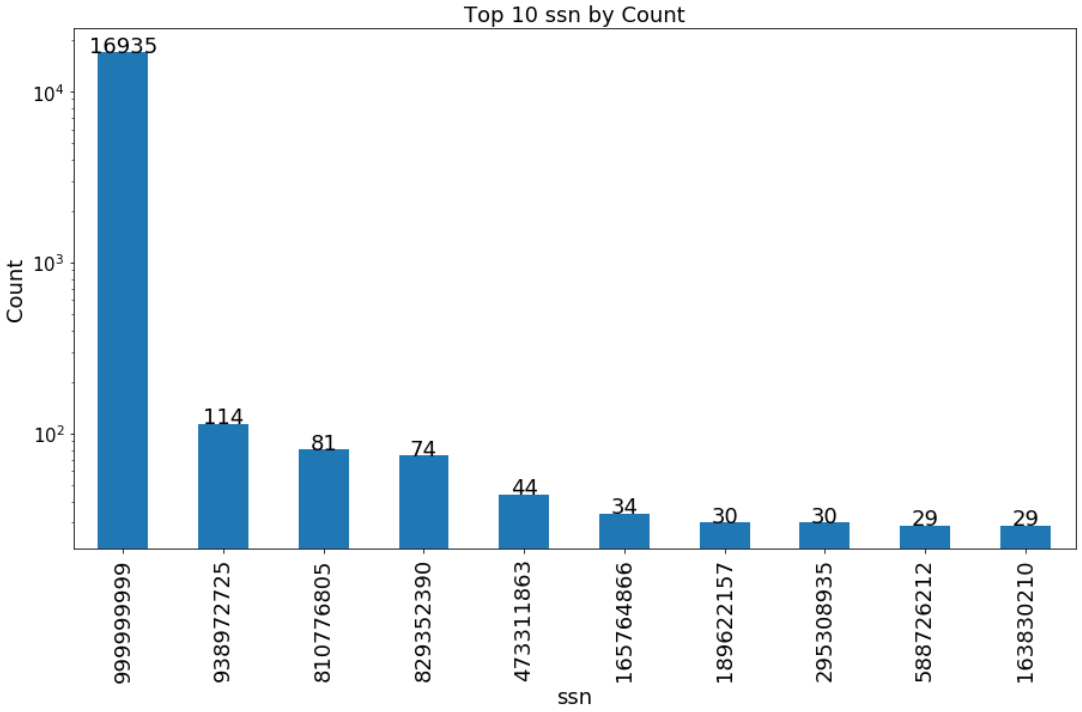
record: A unique record number for referencing each product application in the data, ranging from 1 to 1,000,000. No duplicates were found in the data.

date: Date of the application in the format of Year-Month-Day, with 365 unique values ranging from 2016-01-01 to 2016-12-31. Below are the distribution of the top 10 frequently-appearing dates and a graph showing how the number of applications changes over the course of the year:

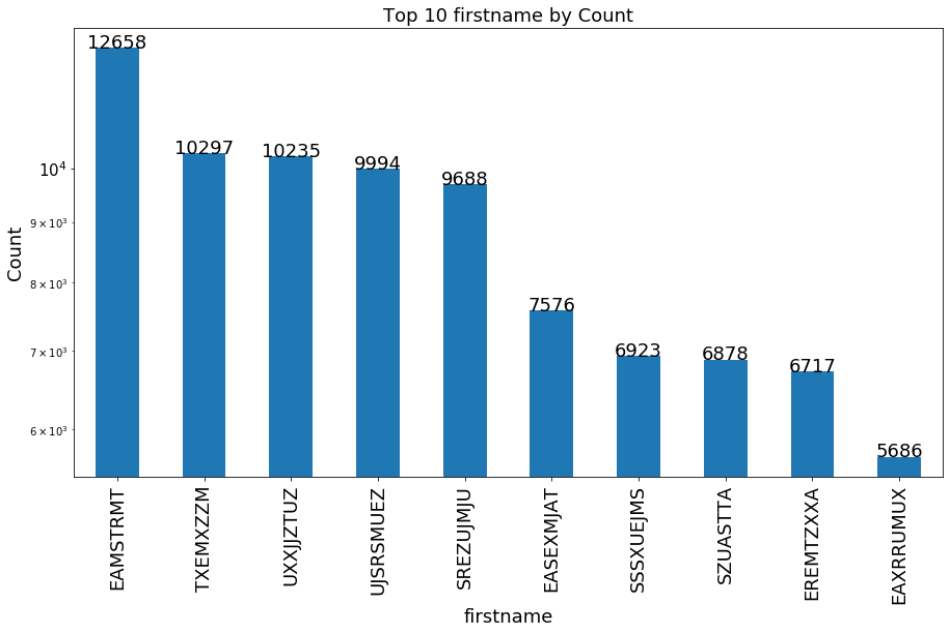




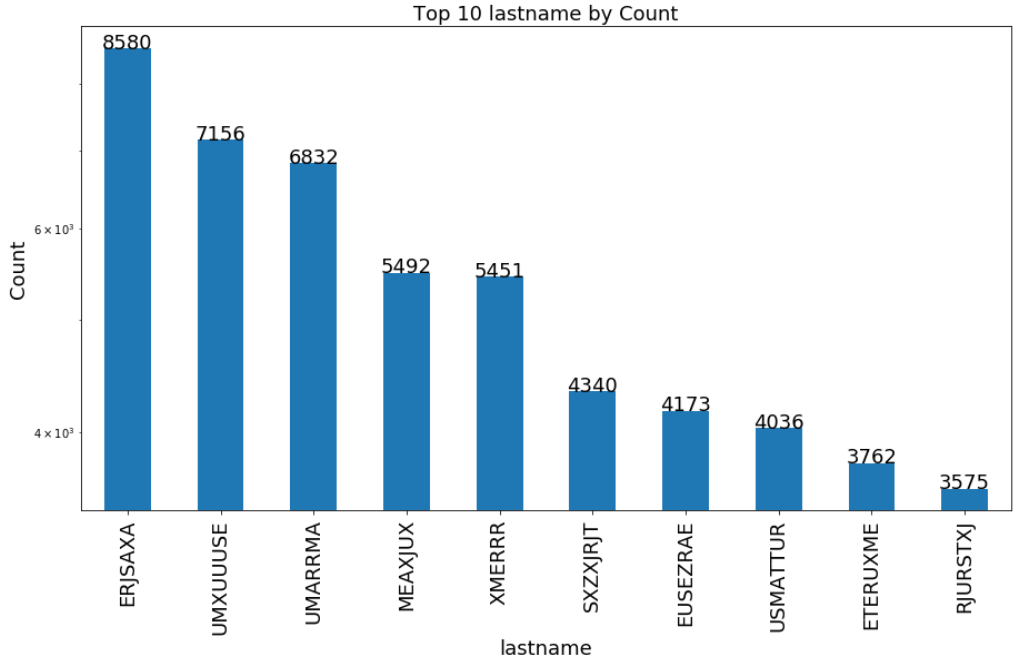
ssn: stands for the social security number of the applicant, with 835,819 distinct values ranging from 36 to 999999999 and 999999999 being the most common (likely to be frivolous data). Below are the top 10 SSN most commonly found in the dataset:



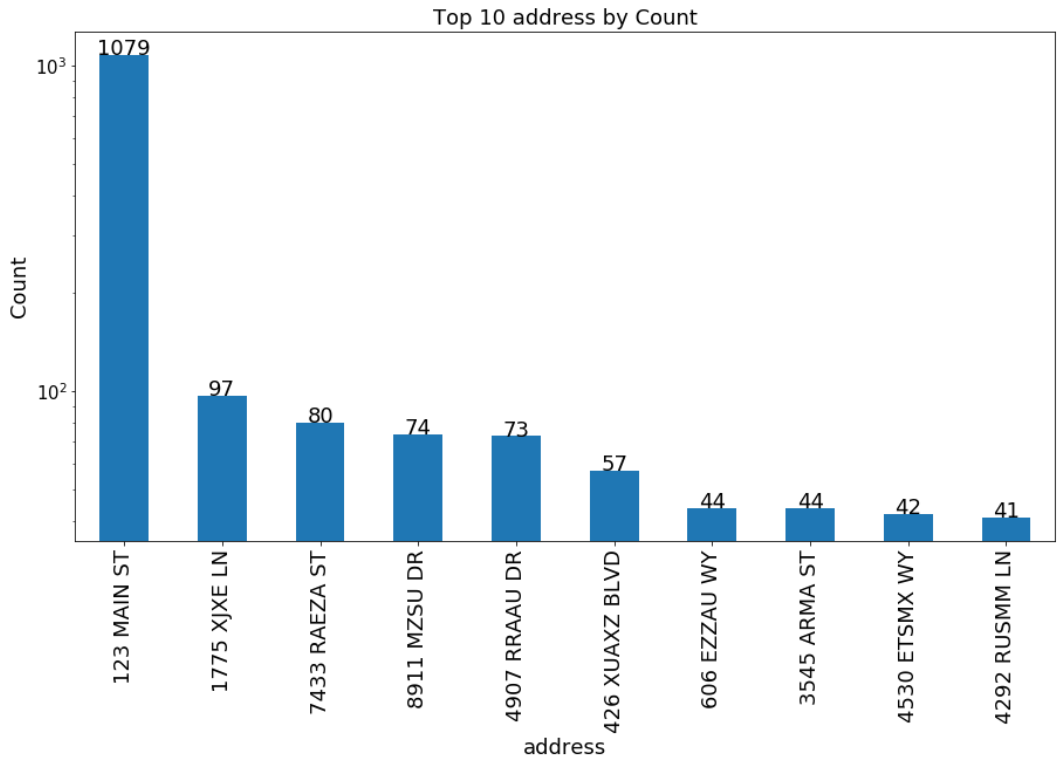
firstname: denotes the first name of the applicant, with 78,136 distinct values and EAMSTRMT being the most common. The top 10 most common first names are as follows:



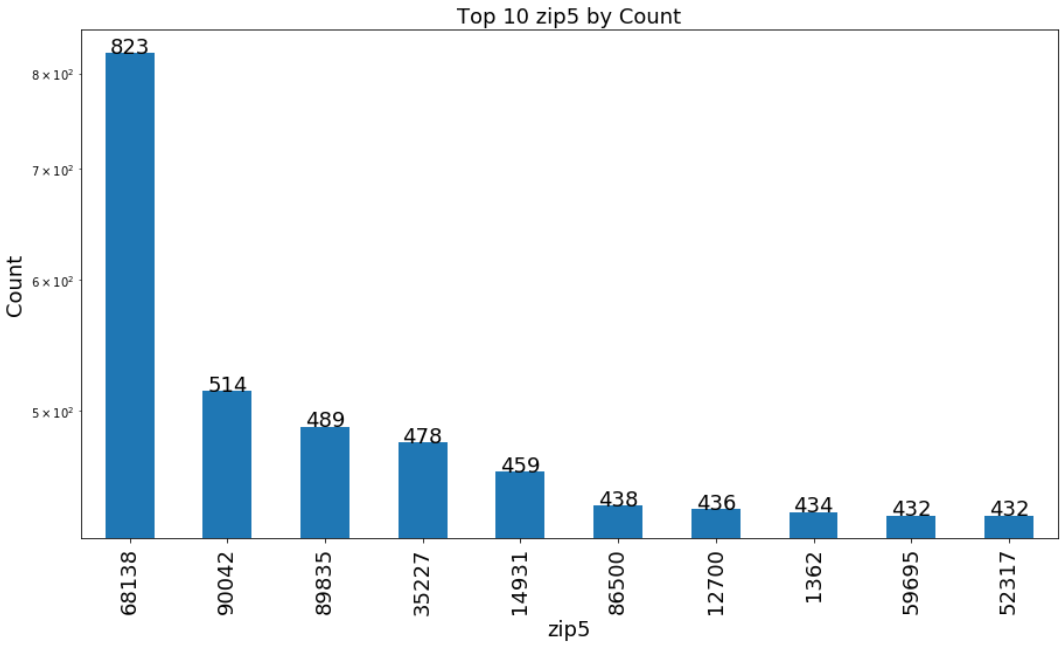
lastname: stands for the last name of the applicant, with 177,001 distinct values and ERJSAXA being the most common. The top 10 most common first names are as follows:



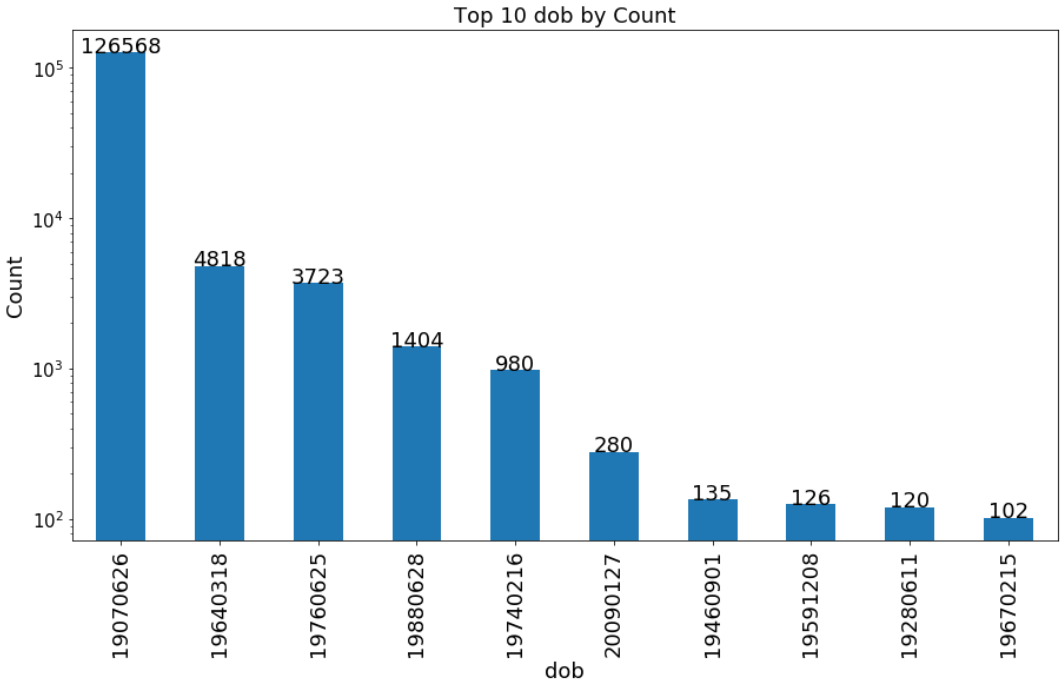
address: shows the home address of the applicant, with 828,774 unique addresses and 123 main street being the most common (likely to be frivolous data). Below are the top 10 most common addresses in the dataset:



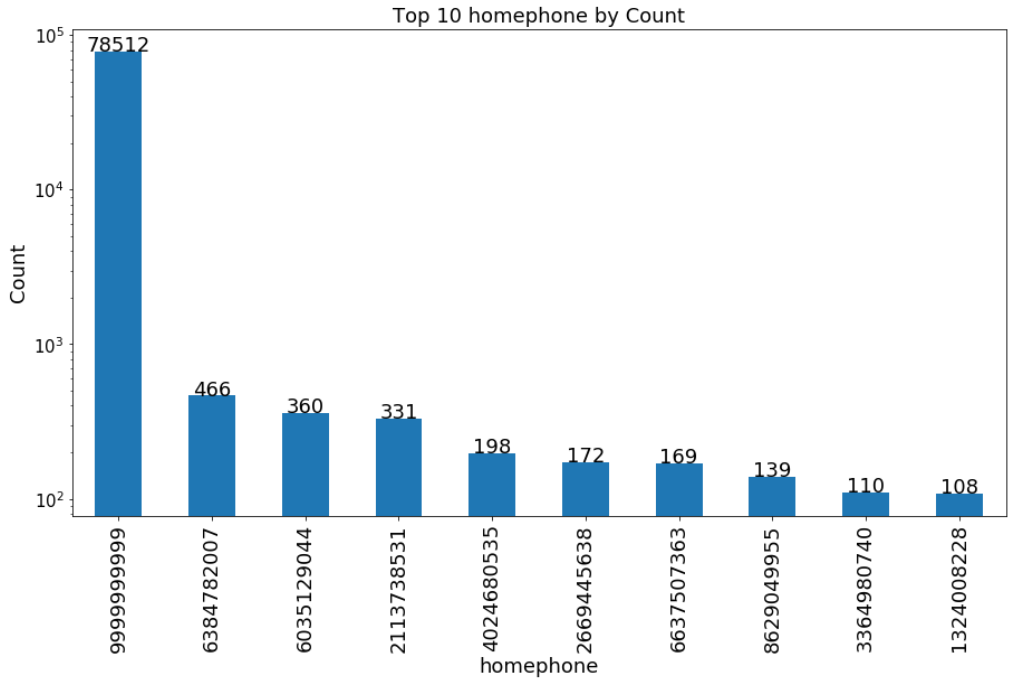
zip5: denotes the five-digit zip code in which the applicant's address is located, with 26,370 unique values. Below are the top 10 five-digit zip codes by frequency:



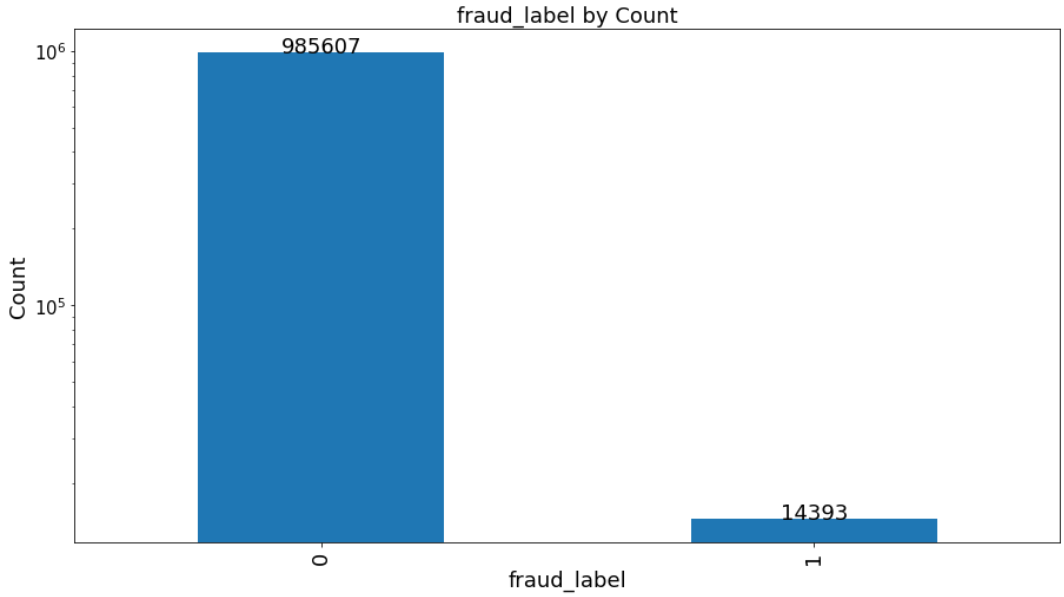
dob: Used to denote the date of birth of the applicant following the format of Year-Month-Day, with 42,673 distinct dates and 19070626 (likely to be frivolous data) being the most common. Below shows the top 10 dates of birth by frequency in the dataset:



homephone: home phone number entered by the applicant when submitting the application, with 28,244 unique values and 999-999-9999 being the most common phone number (likely to be frivolous data). Below shows the top 10 most common phone numbers in the dataset:



fraud_label: A label created to indicate whether an application is fraudulent or not. It takes two values, where 1 = fraudulent, 0 = honest. The distribution of fraud_label is shown below:



B - COMPLETE LIST OF VARIABLES

B1 - VELOCITY VARIABLES

#	Variable Name
1	ssn//prev_d0_count
2	address//prev_d0_count
3	dob//prev_d0_count
4	homephone//prev_d0_count
5	name//prev_d0_count
6	address-zip5//prev_d0_count
7	name-dob//prev_d0_count
8	name-address-zip5//prev_d0_count
9	name-homephone//prev_d0_count
10	address-zip5-dob//prev_d0_count
11	address-zip5-homephone//prev_d0_count
12	dob-homephone//prev_d0_count
13	homephone-name-dob//prev_d0_count
14	ssn-address//prev_d0_count
15	ssn-dob//prev_d0_count
16	ssn-homephone//prev_d0_count
17	ssn-name//prev_d0_count
18	ssn-address-zip5//prev_d0_count
19	ssn-name-dob//prev_d0_count
20	ssn-name-address-zip5//prev_d0_count
21	ssn-name-homephone//prev_d0_count
22	ssn-address-zip5-dob//prev_d0_count
23	ssn-address-zip5-homephone//prev_d0_count
24	ssn-dob-homephone//prev_d0_count
25	ssn-homephone-name-dob//prev_d0_count
26	ssn//prev_d1_count
27	address//prev_d1_count
28	dob//prev_d1_count
29	homephone//prev_d1_count
30	name//prev_d1_count
31	address-zip5//prev_d1_count
32	name-dob//prev_d1_count

33	name-address-zip5//prev_d1_count
34	name-homephone//prev_d1_count
35	address-zip5-dob//prev_d1_count
36	address-zip5-homephone//prev_d1_count
37	dob-homephone//prev_d1_count
38	homephone-name-dob//prev_d1_count
39	ssn-address//prev_d1_count
40	ssn-dob//prev_d1_count
41	ssn-homephone//prev_d1_count
42	ssn-name//prev_d1_count
43	ssn-address-zip5//prev_d1_count
44	ssn-name-dob//prev_d1_count
45	ssn-name-address-zip5//prev_d1_count
46	ssn-name-homephone//prev_d1_count
47	ssn-address-zip5-dob//prev_d1_count
48	ssn-address-zip5-homephone//prev_d1_count
49	ssn-dob-homephone//prev_d1_count
50	ssn-homephone-name-dob//prev_d1_count
51	ssn//prev_d3_count
52	address//prev_d3_count
53	dob//prev_d3_count
54	homephone//prev_d3_count
55	name//prev_d3_count
56	address-zip5//prev_d3_count
57	name-dob//prev_d3_count
58	name-address-zip5//prev_d3_count
59	name-homephone//prev_d3_count
60	address-zip5-dob//prev_d3_count
61	address-zip5-homephone//prev_d3_count
62	dob-homephone//prev_d3_count
63	homephone-name-dob//prev_d3_count
64	ssn-address//prev_d3_count
65	ssn-dob//prev_d3_count
66	ssn-homephone//prev_d3_count
67	ssn-name//prev_d3_count
68	ssn-address-zip5//prev_d3_count
69	ssn-name-dob//prev_d3_count
70	ssn-name-address-zip5//prev_d3_count

71	ssn-name-homephone//prev_d3_count
72	ssn-address-zip5-dob//prev_d3_count
73	ssn-address-zip5-homephone//prev_d3_count
74	ssn-dob-homephone//prev_d3_count
75	ssn-homephone-name-dob//prev_d3_count
76	ssn//prev_d7_count
77	address//prev_d7_count
78	dob//prev_d7_count
79	homephone//prev_d7_count
80	name//prev_d7_count
81	address-zip5//prev_d7_count
82	name-dob//prev_d7_count
83	name-address-zip5//prev_d7_count
84	name-homephone//prev_d7_count
85	address-zip5-dob//prev_d7_count
86	address-zip5-homephone//prev_d7_count
87	dob-homephone//prev_d7_count
88	homephone-name-dob//prev_d7_count
89	ssn-address//prev_d7_count
90	ssn-dob//prev_d7_count
91	ssn-homephone//prev_d7_count
92	ssn-name//prev_d7_count
93	ssn-address-zip5//prev_d7_count
94	ssn-name-dob//prev_d7_count
95	ssn-name-address-zip5//prev_d7_count
96	ssn-name-homephone//prev_d7_count
97	ssn-address-zip5-dob//prev_d7_count
98	ssn-address-zip5-homephone//prev_d7_count
99	ssn-dob-homephone//prev_d7_count
100	ssn-homephone-name-dob//prev_d7_count
101	ssn//prev_d14_count
102	address//prev_d14_count
103	dob//prev_d14_count
104	homephone//prev_d14_count
105	name//prev_d14_count
106	address-zip5//prev_d14_count
107	name-dob//prev_d14_count

108	name-address-zip5//prev_d14_count
109	name-homephone//prev_d14_count
110	address-zip5-dob//prev_d14_count
111	address-zip5-homephone//prev_d14_count
112	dob-homephone//prev_d14_count
113	homephone-name-dob//prev_d14_count
114	ssn-address//prev_d14_count
115	ssn-dob//prev_d14_count
116	ssn-homephone//prev_d14_count
117	ssn-name//prev_d14_count
118	ssn-address-zip5//prev_d14_count
119	ssn-name-dob//prev_d14_count
120	ssn-name-address-zip5//prev_d14_count
121	ssn-name-homephone//prev_d14_count
122	ssn-address-zip5-dob//prev_d14_count
123	ssn-address-zip5-homephone//prev_d14_count
124	ssn-dob-homephone//prev_d14_count
125	ssn-homephone-name-dob//prev_d14_count
126	ssn//prev_d30_count
127	address//prev_d30_count
128	dob//prev_d30_count
129	homephone//prev_d30_count
130	name//prev_d30_count
131	address-zip5//prev_d30_count
132	name-dob//prev_d30_count
133	name-address-zip5//prev_d30_count
134	name-homephone//prev_d30_count
135	address-zip5-dob//prev_d30_count
136	address-zip5-homephone//prev_d30_count
137	dob-homephone//prev_d30_count
138	homephone-name-dob//prev_d30_count
139	ssn-address//prev_d30_count
140	ssn-dob//prev_d30_count
141	ssn-homephone//prev_d30_count
142	ssn-name//prev_d30_count
143	ssn-address-zip5//prev_d30_count
144	ssn-name-dob//prev_d30_count
145	ssn-name-address-zip5//prev_d30_count

146	ssn-name-homephone//prev_d30_count
147	ssn-address-zip5-dob//prev_d30_count
148	ssn-address-zip5-homephone//prev_d30_count
149	ssn-dob-homephone//prev_d30_count
150	ssn-homephone-name-dob//prev_d30_count

B2 - RELATIVE VELOCITY VARIABLES

#	Variable Name
151	ssn//prev_d1_d3_avg
152	address//prev_d1_d3_avg
153	dob//prev_d1_d3_avg
154	homephone//prev_d1_d3_avg
155	name//prev_d1_d3_avg
156	address-zip5//prev_d1_d3_avg
157	name-dob//prev_d1_d3_avg
158	name-address-zip5//prev_d1_d3_avg
159	name-homephone//prev_d1_d3_avg
160	address-zip5-dob//prev_d1_d3_avg
161	address-zip5-homephone//prev_d1_d3_avg
162	dob-homephone//prev_d1_d3_avg
163	homephone-name-dob//prev_d1_d3_avg
164	ssn-address//prev_d1_d3_avg
165	ssn-dob//prev_d1_d3_avg
166	ssn-homephone//prev_d1_d3_avg
167	ssn-name//prev_d1_d3_avg
168	ssn-address-zip5//prev_d1_d3_avg
169	ssn-name-dob//prev_d1_d3_avg
170	ssn-name-address-zip5//prev_d1_d3_avg
171	ssn-name-homephone//prev_d1_d3_avg
172	ssn-address-zip5-dob//prev_d1_d3_avg
173	ssn-address-zip5-homephone//prev_d1_d3_avg
174	ssn-dob-homephone//prev_d1_d3_avg
175	ssn-homephone-name-dob//prev_d1_d3_avg
176	ssn//prev_d1_d7_avg
177	address//prev_d1_d7_avg

178	dob//prev_d1_d7_avg
179	homephone//prev_d1_d7_avg
180	name//prev_d1_d7_avg
181	address-zip5//prev_d1_d7_avg
182	name-dob//prev_d1_d7_avg
183	name-address-zip5//prev_d1_d7_avg
184	name-homephone//prev_d1_d7_avg
185	address-zip5-dob//prev_d1_d7_avg
186	address-zip5-homephone//prev_d1_d7_avg
187	dob-homephone//prev_d1_d7_avg
188	homephone-name-dob//prev_d1_d7_avg
189	ssn-address//prev_d1_d7_avg
190	ssn-dob//prev_d1_d7_avg
191	ssn-homephone//prev_d1_d7_avg
192	ssn-name//prev_d1_d7_avg
193	ssn-address-zip5//prev_d1_d7_avg
194	ssn-name-dob//prev_d1_d7_avg
195	ssn-name-address-zip5//prev_d1_d7_avg
196	ssn-name-homephone//prev_d1_d7_avg
197	ssn-address-zip5-dob//prev_d1_d7_avg
198	ssn-address-zip5-homephone//prev_d1_d7_avg
199	ssn-dob-homephone//prev_d1_d7_avg
200	ssn-homephone-name-dob//prev_d1_d7_avg
201	ssn//prev_d1_d14_avg
202	address//prev_d1_d14_avg
203	dob//prev_d1_d14_avg
204	homephone//prev_d1_d14_avg
205	name//prev_d1_d14_avg
206	address-zip5//prev_d1_d14_avg
207	name-dob//prev_d1_d14_avg
208	name-address-zip5//prev_d1_d14_avg
209	name-homephone//prev_d1_d14_avg
210	address-zip5-dob//prev_d1_d14_avg
211	address-zip5-homephone//prev_d1_d14_avg
212	dob-homephone//prev_d1_d14_avg
213	homephone-name-dob//prev_d1_d14_avg
214	ssn-address//prev_d1_d14_avg
215	ssn-dob//prev_d1_d14_avg

216	ssn-homephone//prev_d1_d14_avg
217	ssn-name//prev_d1_d14_avg
218	ssn-address-zip5//prev_d1_d14_avg
219	ssn-name-dob//prev_d1_d14_avg
220	ssn-name-address-zip5//prev_d1_d14_avg
221	ssn-name-homephone//prev_d1_d14_avg
222	ssn-address-zip5-dob//prev_d1_d14_avg
223	ssn-address-zip5-homephone//prev_d1_d14_avg
224	ssn-dob-homephone//prev_d1_d14_avg
225	ssn-homephone-name-dob//prev_d1_d14_avg
226	ssn//prev_d1_d30_avg
227	address//prev_d1_d30_avg
228	dob//prev_d1_d30_avg
229	homephone//prev_d1_d30_avg
230	name//prev_d1_d30_avg
231	address-zip5//prev_d1_d30_avg
232	name-dob//prev_d1_d30_avg
233	name-address-zip5//prev_d1_d30_avg
234	name-homephone//prev_d1_d30_avg
235	address-zip5-dob//prev_d1_d30_avg
236	address-zip5-homephone//prev_d1_d30_avg
237	dob-homephone//prev_d1_d30_avg
238	homephone-name-dob//prev_d1_d30_avg
239	ssn-address//prev_d1_d30_avg
240	ssn-dob//prev_d1_d30_avg
241	ssn-homephone//prev_d1_d30_avg
242	ssn-name//prev_d1_d30_avg
243	ssn-address-zip5//prev_d1_d30_avg
244	ssn-name-dob//prev_d1_d30_avg
245	ssn-name-address-zip5//prev_d1_d30_avg
246	ssn-name-homephone//prev_d1_d30_avg
247	ssn-address-zip5-dob//prev_d1_d30_avg
248	ssn-address-zip5-homephone//prev_d1_d30_avg
249	ssn-dob-homephone//prev_d1_d30_avg
250	ssn-homephone-name-dob//prev_d1_d30_avg

B3 - DAYS SINCE VARIABLES

#	Variable Name
251	ssn//days_since
252	address//days_since
253	dob//days_since
254	homephone//days_since
255	name//days_since
256	address-zip5//days_since
257	name-dob//days_since
258	name-address-zip5//days_since
259	name-homephone//days_since
260	address-zip5-dob//days_since
261	address-zip5-homephone//days_since
262	dob-homephone//days_since
263	homephone-name-dob//days_since
264	ssn-address//days_since
265	ssn-dob//days_since
266	ssn-homephone//days_since
267	ssn-name//days_since
268	ssn-address-zip5//days_since
269	ssn-name-dob//days_since
270	ssn-name-address-zip5//days_since
271	ssn-name-homephone//days_since
272	ssn-address-zip5-dob//days_since
273	ssn-address-zip5-homephone//days_since
274	ssn-dob-homephone//days_since
275	ssn-homephone-name-dob//days_since

B4 - RISK TABLE (DAY OF WEEK) VARIABLE

#	Variable Name
276	dayOfWeek_risk