Fraud Detection in Product Applications

Project 2 Report

DSO 562 - Fraud Analytics
April 2018

TEAM 2

Louis Yansaud | Olivia Huang | Jamie Lee | Tong Xie | Conglin Xu | Ravi Gangumalla

TABLE OF CONTENTS

EXECUTIVE SUMMARY

PROJECT DESCRIPTION

This report provides a detailed analysis of the ‘applications’ dataset using supervised fraud algorithms and machine learning statistical techniques.

The programming tools used for Data Cleaning, Modeling, and Evaluation were Microsoft Excel, Tableau, R, along with mySQL through R. The original dataset contains unique records of more 90,000 product applications over the year 2016.

Necessary feature analysis include:

- Data Understanding and Data Cleaning (dealing with frivolous values)
- Feature Engineering (creating expert Variables, performing feature selection)
- Data Modeling (running fraud detection models)
- Data Evaluation (comparing model results)

PROJECT GOAL

Our objective for this project is to use a supervised fraud algorithm to build a fraud detection model that predicts product application fraud with high predictive accuracy. More precisely, we aim to find the top performing machine learning model that is able to predict the highest number of fraudulent cases with a low false positive rate.

The report will present several classification machine learning techniques for fraud detection and their performances on our dataset, in complete detail such as Logistic Regression, Random Forests, Support Vector Machine, and Gradient Boosting Trees.

KEY FINDINGS

Here are the main key findings from building a Fraud Detection Model on the Applications dataset:

- The Gradient Boosting Trees method yielded the best Fraud Detection model at 10% penetration on the OOT (Out of Time) dataset with a FDR (Fraud Detection Rate) of 13.02% including a FDR of 12.50% on the testing dataset
- The Support Vector Machine Method yielded the lowest Fraud Detection Rate (FDR) at 10% penetration on the OOT (Out of Time) dataset with a FDR (Fraud Detection Rate) of 11.81% including a FDR of 11.01% on the testing dataset
- Overall, Gradient Boosting Trees and Random Forest Models both demonstrated relatively high predictive performance, with the highest predictive power in their respective segments

DATA UNDERSTANDING

DESCRIPTION OF THE DATA

The file dataset is called “applications.csv”. This dataset contains records of all product applications filed in the year 2016 across the United States. The dataset consists of 984,866 product applications data with 10 different fields. It contains the information about a unique id, the date the application was created, a social security number, a first name, a last name, an address, a zip code, a date of birth, a home phone number, and a Fraud indicator variable.

Below is a list of variables, categorized by categorical, date, text, and indicator/dummy, along with short descriptions and percent populated values.

CATEGORICAL VARIABLES

Table 1. Categorical Variables

Categorical Variable	Description	Percent Populated (%)
record	Record (row) number of application	100.0
ssn	Applicant social security number	100.0
firstname	Applicant first name	100.0
lastname	Applicant last name	100.0
zip5	Applicant zip code	100.0
homephone	Assessed value for the land	100.0

DATE VARIABLES

Table 2. Date Variables

Date Variable	Description	Percent Populated (%)
date	Product application filing date	100.0
dob	Applicant dob	100.0

TEXT VARIABLES

Table 3. Text Variables

Text Variable	Description	Percent Populated (%)
Address	Applicant home address	100.0

INDICATOR/DUMMY VARIABLES

Table 4. Indicator/Dummy Variables

Indicator/ Dummy Variable	Description	Percent Populated (%)
Fraud	Fraudulent or not fraudulent	100.0

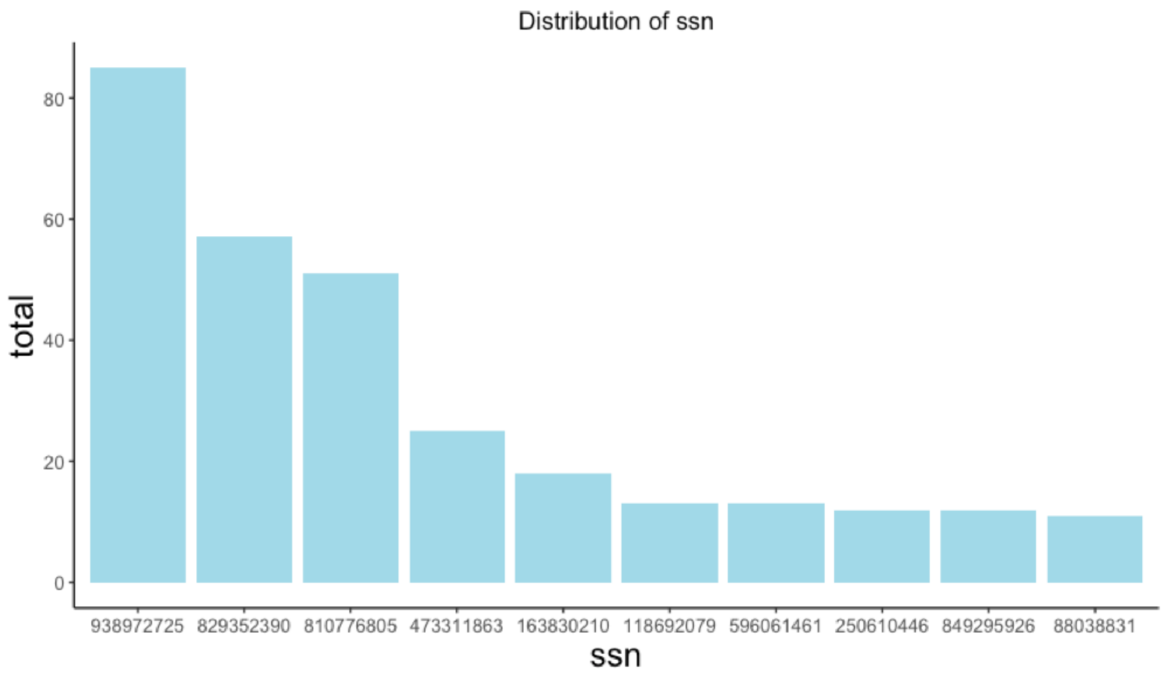
IMPORTANT VARIABLES IN DETAIL

Bellow is a description of the most important variables in complete details along with its distribution.

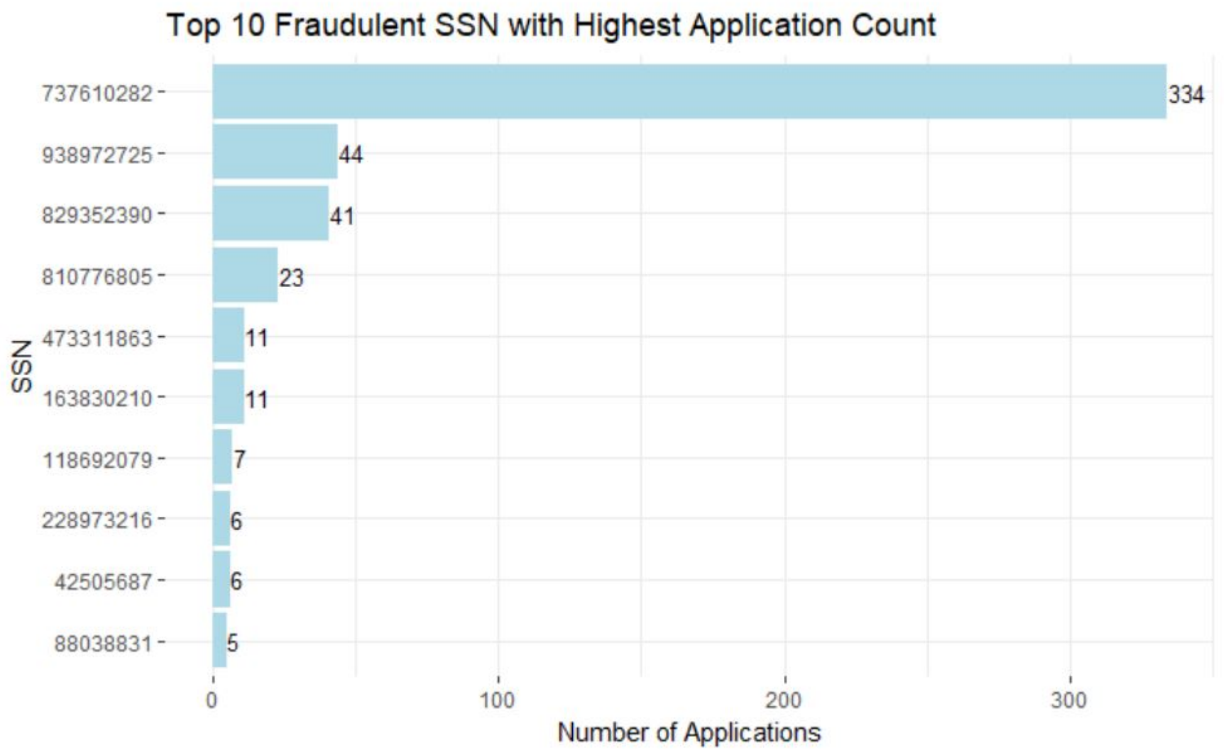
ssn

ssn is a categorical variable with 9 digits total showing the social security number of the person that was reported on the application. It is 100% populated with 86,771 unique values. The range of *ssn* goes from two-digit social security number to 9-digit social security number. Any social security number shorter than 9 digits means that they have some leadings 0 in the value, i.e. “99” is actually “000000099.” Something worth noticing is the value “737610282”, there is a count of 1478 total which could indicate a potential fraudulent record.

Here is a distribution of the Top 10 ssn excluding “737610282”.



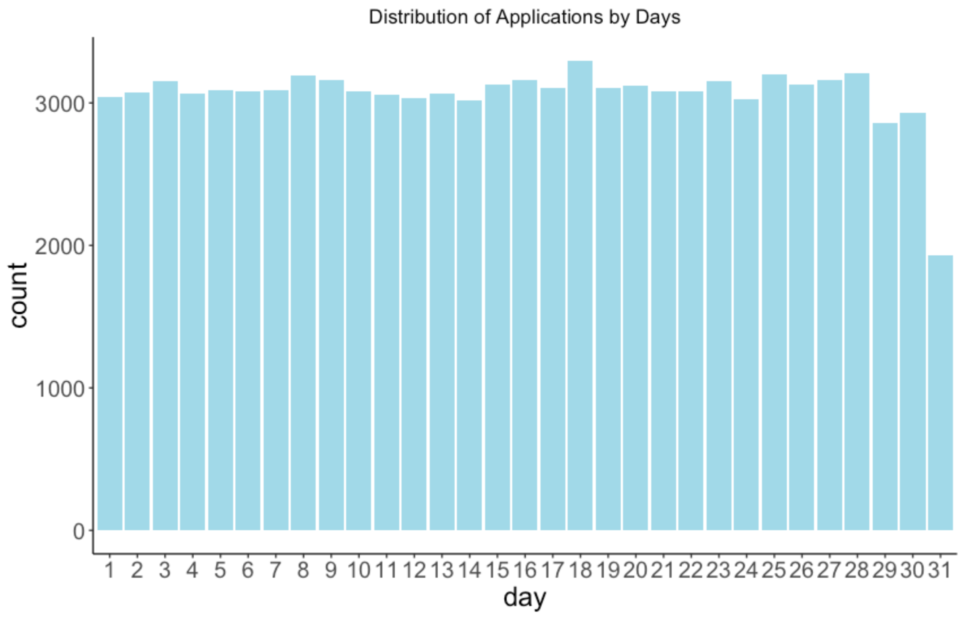
.

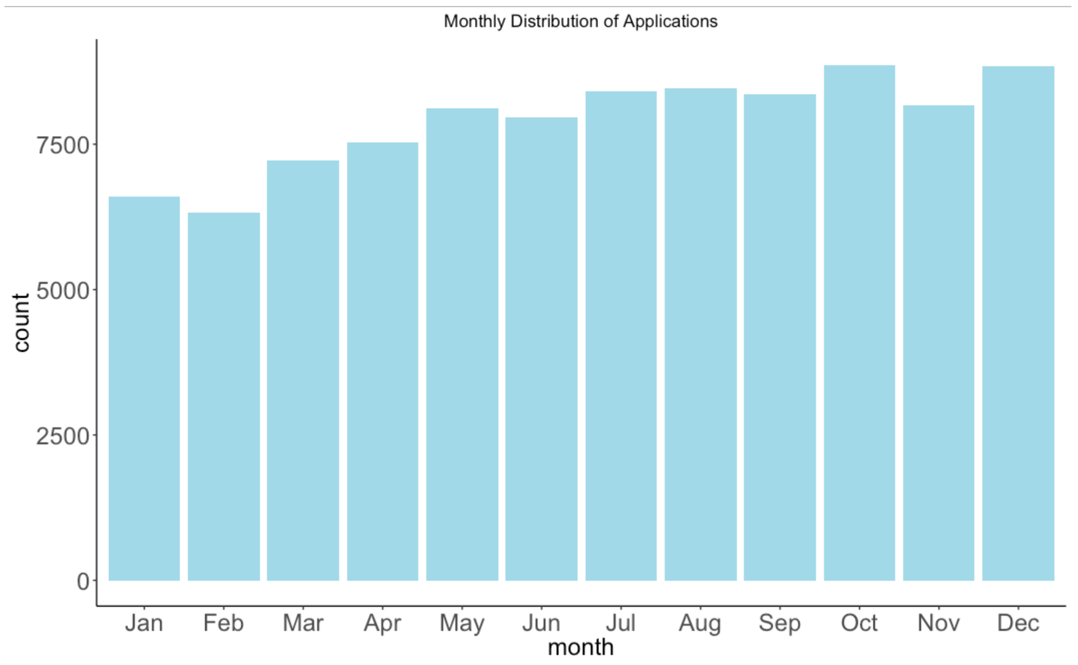

Top 10 Fraudulent SSN with Highest Application Count
SSN
737610282
938972725
829352390
810776805
473311863
163830210
118692079
228973216
42505687
88038831
334
44
41
23
11
11
7
6
6
5
0
100
200
300
Number of Applications

date

date is a date type variable showing the date in which the application was created. It goes in the format of "%m/%d/%y".

It is 100% populated with a total of 365 unique values. The total unique values actually show that each observation is populated by a day of the month and year of 2016.

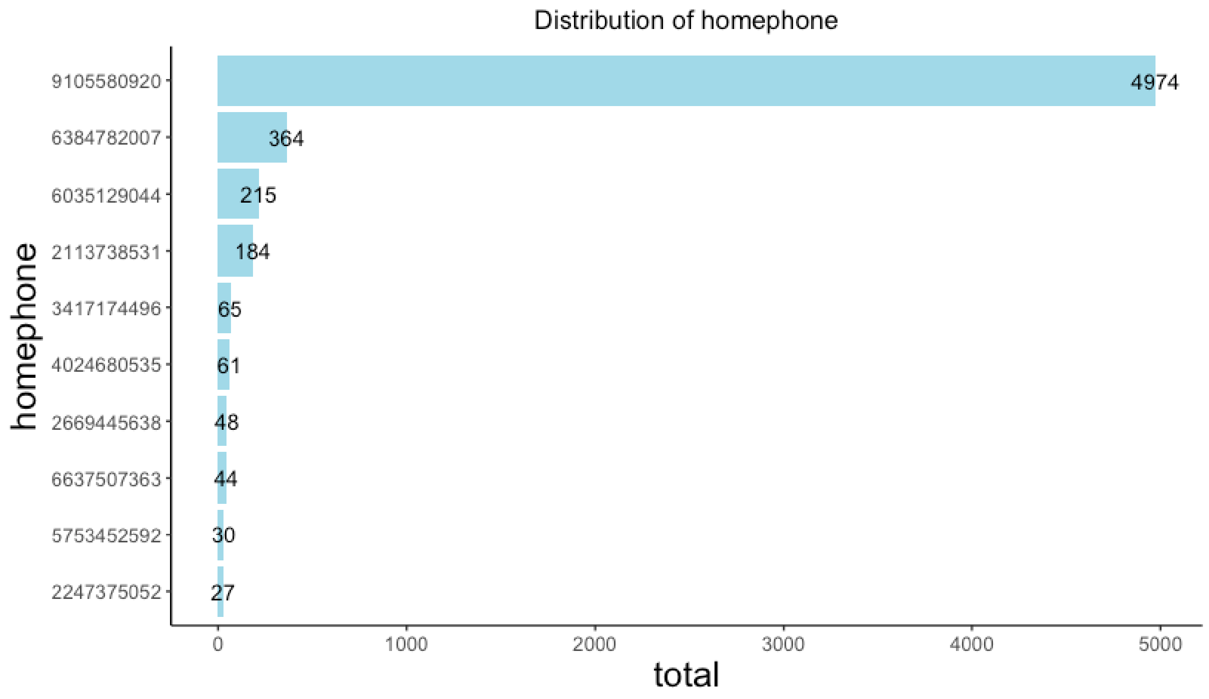


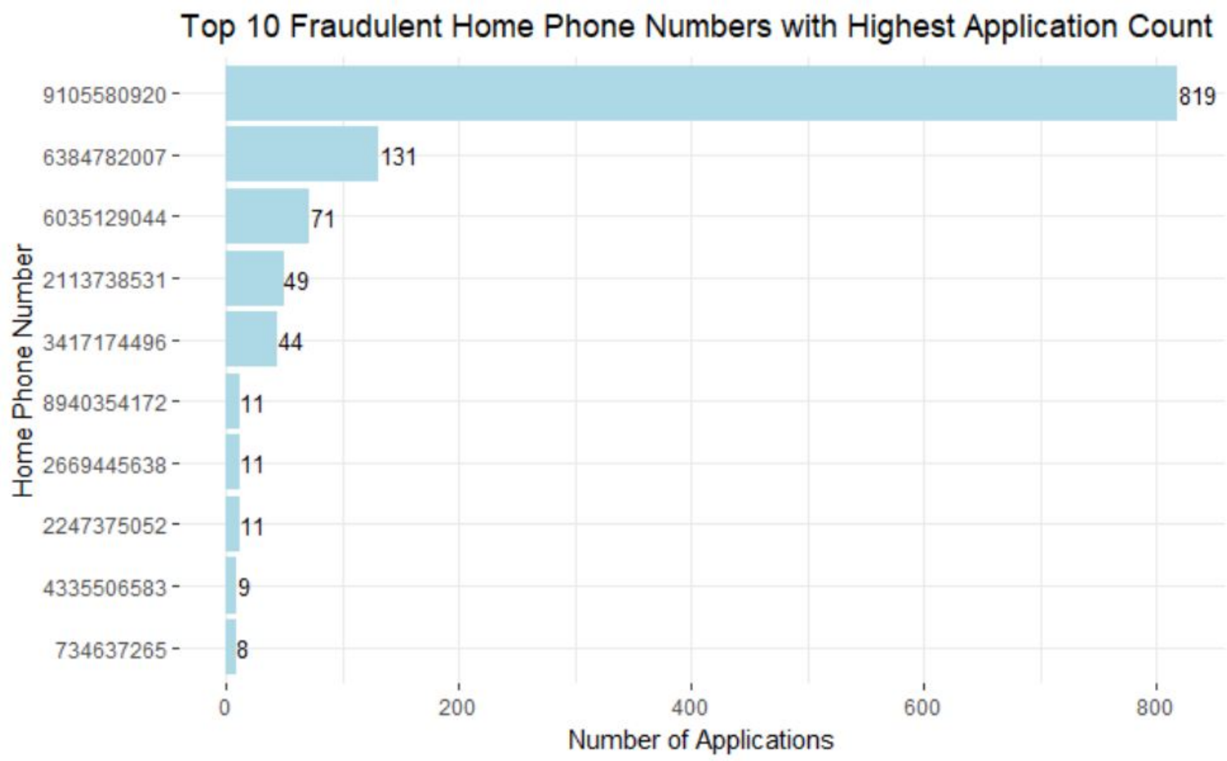

Monthly Distribution of Applications
count
0
2500
5000
7500
month
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

homephone

homephone is a categorical variable indicating the phone number of a person that he/she reported on the application.

It is 100% populated and it contains 20,762 unique values. The most frequent homephone is "9105580920" with a total count of 4974 and it accounts for 5.24% of all records. This could indicate a high fraudulent record because of the high frequency it populates.

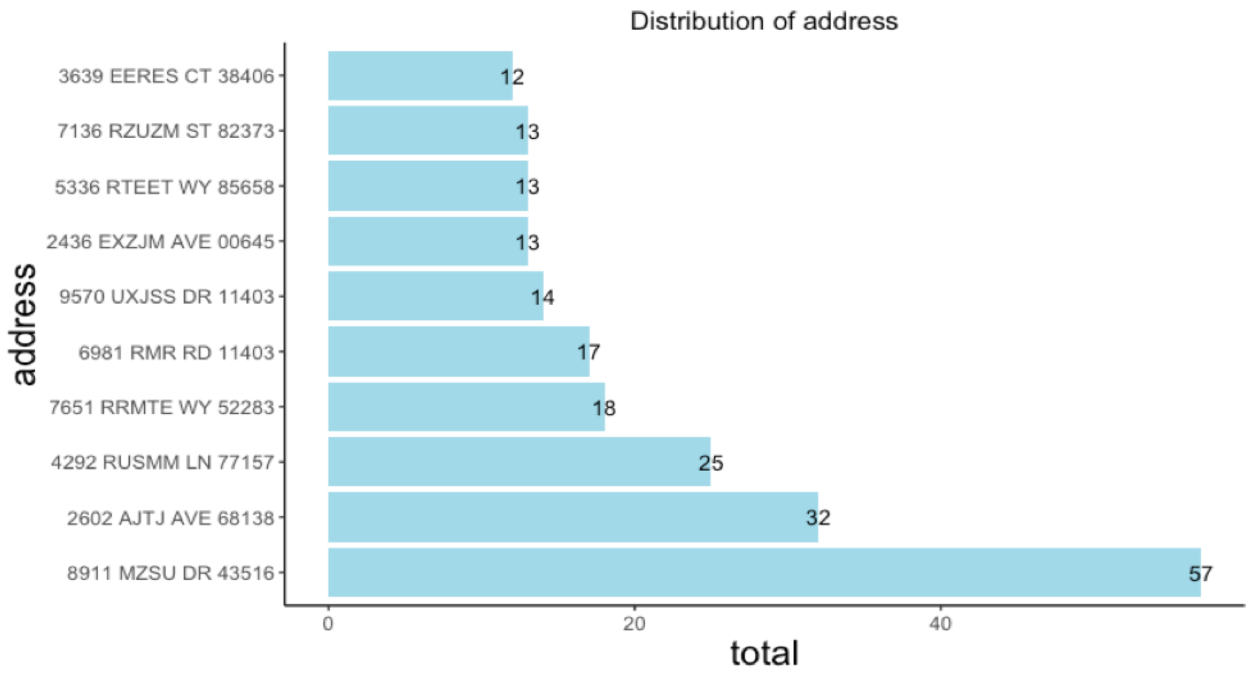


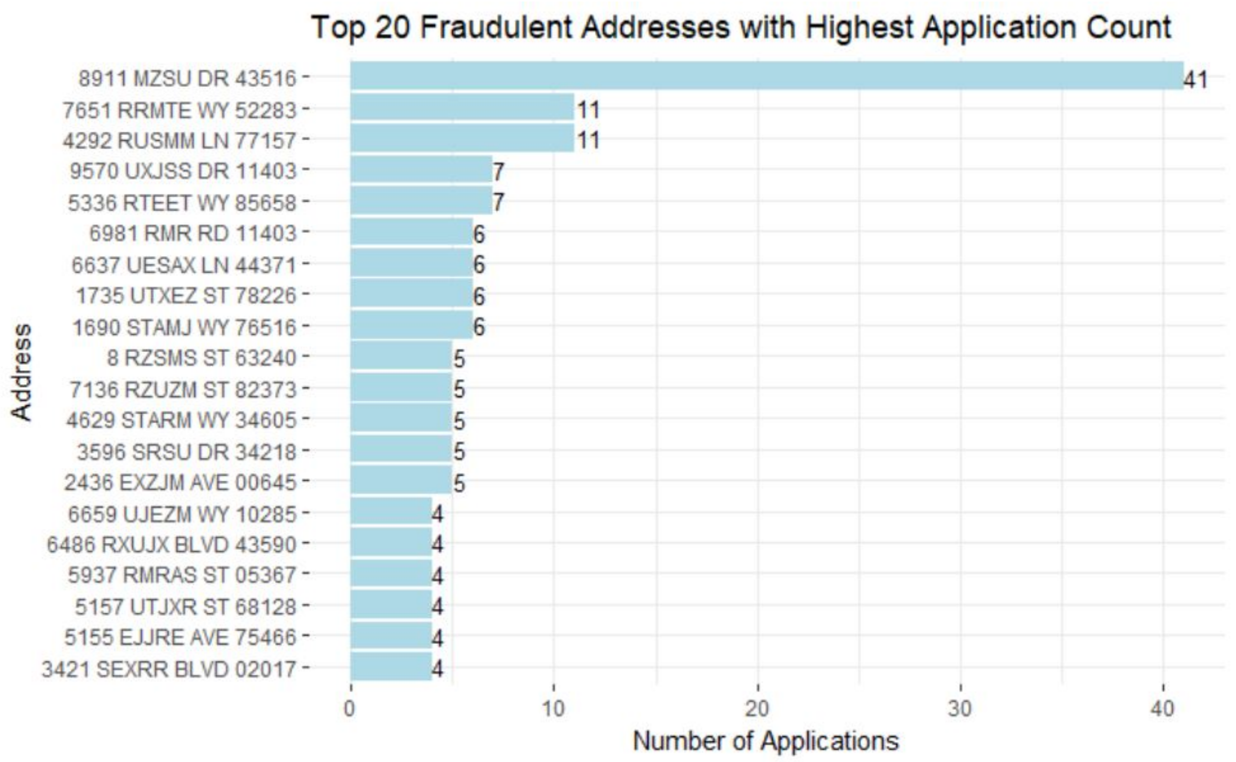

Top 10 Fraudulent Home Phone Numbers with Highest Application Count
Home Phone Number
9105580920
6384782007
6035129044
2113738531
3417174496
8940354172
2669445638
2247375052
4335506583
734637265
819
131
71
49
44
11
11
11
9
8
0
200
400
600
800
Number of Applications

address

address is a text variable indicating the address of a person that he/she reported on the application.

It is 100% populated with 88,167 unique values totals. The most frequent address is “8911 MZSU DR 43516” which appeared 57 times inside the Applications dataset and it indicates 0.06% of all records.

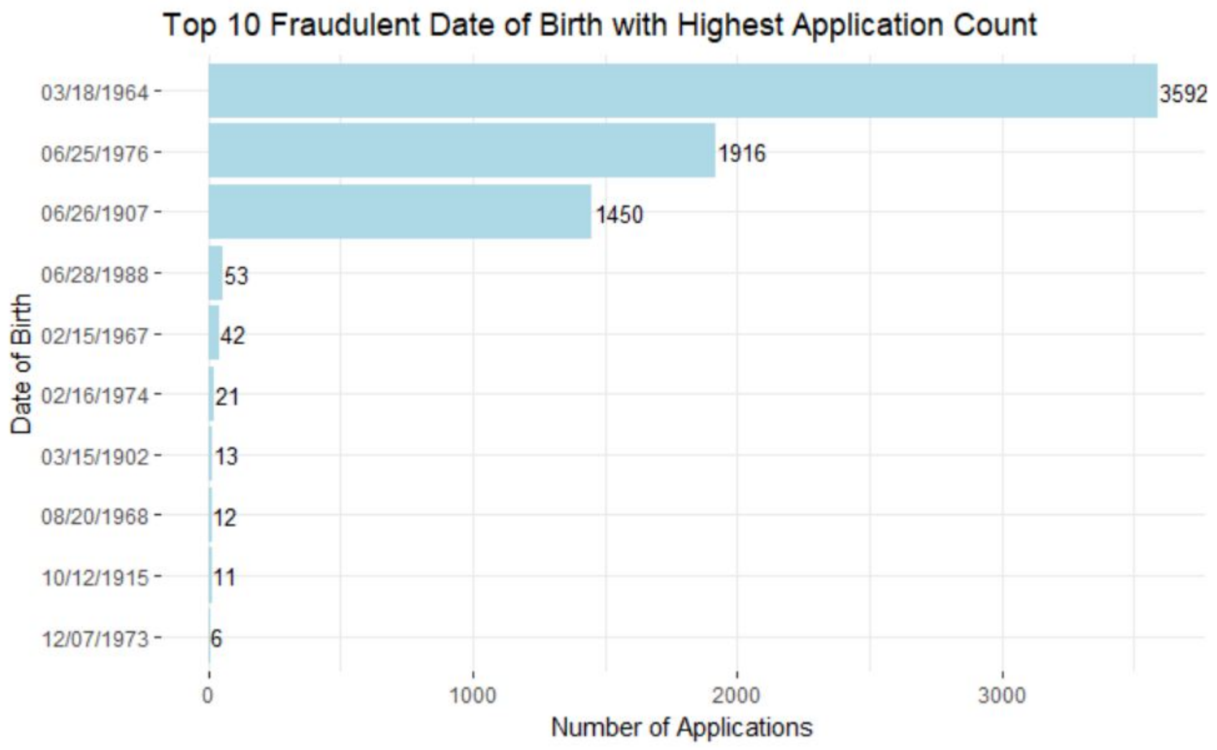



Top 20 Fraudulent Addresses with Highest Application Count
8911 MZSU DR 43516
7651 RRMTE WY 52283
4292 RUSMM LN 77157
9570 UXJSS DR 11403
5336 RTEET WY 85658
6981 RMR RD 11403
6637 UESAX LN 44371
1735 UTXEZ ST 78226
1690 STAMJ WY 76516
8 RZSMS ST 63240
7136 RZUZM ST 82373
4629 STARM WY 34605
3596 SRSU DR 34218
2436 EXZJM AVE 00645
6659 UJEZM WY 10285
6486 RXUJX BLVD 43590
5937 RMRAS ST 05367
5157 UTJXR ST 68128
5155 EJJRE AVE 75466
3421 SEXRR BLVD 02017
41
11
11
7
7
6
6
6
6
5
5
5
5
5
4
4
4
4
4
4
Address
0
10
20
30
40
Number of Applications

dob

dob is a date variable indicating the date of birth of a person that he/she reported on the application.

It is 100% populated and it contains 30,599 unique values. The most frequent *dob* is “6/27/07”, and June 26, 2007 is a Tuesday with a total count of 9681 and a populated frequency of 10.2% of all records.



DATA CLEANING

Our primary goals in the data cleaning process were to:

1. Delete frivolous values for necessary fields (SSN and home phone)
2. Perform necessary data format conversions as needed for variable creation
3. Consolidate the data to relevant and useful fields

FRIVOLOUS VALUES

We discovered two fields, ssn and home phone, that had frivolous values occurring at relatively high frequency within their respective fields.

- Frivolous ssn: 737610282
  (total count of 1478, frequency populated 1.56%)

- Frivolous home phone: 9105580920
  (total count of 4974, frequency populated 5.24%)

Since these values occur at a high magnitude compared to other values, we labelled them as frivolous because they are not meaningful for data analysis. For the purposes of our analysis, we chose to ignore these values by deleting all records containing either of the frivolous values listed above.

DATE CONVERSION

We took the date fields date & dob and converted them to *yyyy-mm-dd* format. Since our analysis involves time as a key characteristic, both in product application filing date and applicant’s date of birth, we converted the date fields to prepare for special variable creation and feature engineering.

EXTRACTING RELEVANT AND USEFUL FIELDS

In the process of consolidating our data to contain only useful fields, we created several special variables that would allow for more efficient data analysis. These special variables, along with the reasoning behind their creation, are described in the following section.

VARIABLES & FEATURE ENGINEERING

SPECIAL VARIABLES

Before building expert variables, we built the following special variables, listed below.

Table 5. List of Special Variables

Special Variable	Variables Combined	Description
person	firstname, lastname, dob, ssn	1 Entity concatenating 3 variables to represent 1 person. Used to create more specific and unique identifier to represent 1 person.
dayofyear	none	Integer value representing day of the year, from 1,...,366 (366 total days in year 2016). Used to split data into training, testing, and out of time validation sets.
fullname_birthday	firstname, lastname, dob	Concatenation of 3 variables to represent firstname, lastname and dob combination for single record. Used to create another specific identifier linking an individual's name with his/her birthday.

In order to create expert variables, we selected the features we believed would have the strongest relationship to the likelihood that an application is fraudulent. The goal was to create a variable that, when performing feature selection algorithms, would produce a high score of importance based on the measure.

CREATING EXPERT VARIABLES BASED ON TIME WINDOW

Considering that product application information incorporates time as a key factor, we based our expert variable creation on time period windows of 3, 7, 14, and 30 days. We wanted to capture as many fraudulent applications in both shorter and longer time periods. By utilizing four different time windows, we were able to capture a range of time periods, from several days to a week and at most, a month.

Within each of these time windows, we constructed variables to count the number of persons with the same values for:

- ssn (20 expert variables)
- homephone (20 expert variables)
- person (concatenation of firstname, lastname, ssn, dob) (12 expert variables)
- address (20 expert variables)
- zip5 (20 expert variables)
- dob (16 expert variables)

CREATING EXPERT VARIABLES BY COMBINING DIFFERENT FIELDS

We also created expert variables by combining various fields that we believed were strongly related and would potentially be of high importance:

- Variables surrounding ssn: A ssn associated with a high number of address, homephone, zip and fullname_birthday respectively may imply a fraudulent behavior.
    - ssn and distinct address

      Number of persons within certain time period (3, 7, 14, 30) with particular ssn and different addresses.

- ssn and homephone

  Number of persons within certain time period (3, 7, 14, 30) with particular ssn and different homephone.

- ssn and zip

  Number of persons within certain time period (3, 7, 14, 30) with particular ssn and different zips

- ssn and fullname_birthday

  Number of persons within certain time period (3, 7, 14, 30) with particular ssn and different fullname_birthdays

- Variables surrounding homephone: An ssn associated with a high number of address, homephone, zip and fullname_birthday respectively may imply a fraudulent behavior.
  - homephone and address

    Number of persons within certain time period (3, 7, 14, 30) with particular homephone and different addresses

  - homephone and ssn

    Number of persons within certain time period (3, 7, 14, 30) with particular homephone and different ssns

  - homephone and zip

    Number of persons within certain time period (3, 7, 14, 30) with particular homephone and different zips

  - homephone and fullname_birthday

    Number of persons within certain time period (3, 7, 14, 30) with particular homephone and different fullname_birthdays

- Variables surrounding person: A ssn associated with a high number of address and zip respectively may imply a fraudulent behavior.
    - person and address

      Number of persons within certain time period (3, 7, 14, 30) with particular homephone and different addresses

    - person and zip

      Number of persons within certain time period (3, 7, 14, 30) with particular homephone and different zips

- Variables surrounding address: A ssn associated with a high number of person, homephone, ssn and fullname_birthday respectively may imply a fraudulent behavior.
    - address and person

      Number of person within certain time period (3, 7, 14, 30) with particular address and different person

    - address and homephone

      Number of person within certain time period (3, 7, 14, 30) with particular address and different homephone

    - address and ssn

      Number of person within certain time period (3, 7, 14, 30) with particular address and different ssns

    - address and fullname_birthday

      Number of person within certain time period (3, 7, 14, 30) with particular address and different fullname_birthday

- Variables surrounding address: A ssn associated with a high number of person, homephone, ssn, address and fullname_birthday respectively may imply a fraudulent behavior.

- zip and person

  Number of person within certain time period (3, 7, 14, 30) with particular zip and different person

- zip and homephone

  Number of person within certain time period (3, 7, 14, 30) with particular zip and different homephone

- zip and ssn

  Number of person within certain time period (3, 7, 14, 30) with particular zip and different ssn

- zip and address

  Number of person within certain time period (3, 7, 14, 30) with particular zip and different address

- zip and fullname_birthday

  Number of person within certain time period (3, 7, 14, 30) with particular zip and different fullname_birthday

- Variables surrounding dob: A ssn associated with a high number of person, homephone, ssn and address respectively may imply a fraudulent behavior.
- dob and person
  - Number of person within certain time period (3, 7, 14, 30) with particular dob and different person
- dob and homephone
  - Number of person within certain time period (3, 7, 14, 30) with particular dob and different homephone
- dob and ssn
  - Number of person within certain time period (3, 7, 14, 30) with particular dob and different ssn
- dob and address
  - Number of person within certain time period (3, 7, 14, 30) with particular dob and different address

Following is a list of expert variables we created. In total, we created 113 expert variables.

Table 6. List of Expert Variables

	Expert Variable Name	Description
1	person_address3	the number of person associated with one particular address over a 3 day period
2	person_address7	the number of person associated with one particular address over a 7 day period
3	person_address14	the number of person associated with one particular address over a 14 day period
4	person_address30	the number of person associated with one particular address over a 30 day period
5	address_distinct_homephone3	the number of distinct homephone associated with one particular address over a 3 day period
6	address_distinct_homephone7	the number of distinct homephone associated with one particular address over a 7 day period
7	address_distinct_homephone14	the number of distinct homephone associated with one particular address over a 14 day period
8	address_distinct_homephone30	the number of distinct homephone associated with one particular address

		over a 30 day period
9	address_distinct_ssn3	the number of distinct ssn associated with one particular address over a 3 day period
10	address_distinct_ssn7	the number of distinct ssn associated with one particular address over a 7 day period
11	address_distinct_ssn14	the number of distinct ssn associated with one particular address over a 14 day period
12	address_distinct_ssn30	the number of distinct ssn associated with one particular address over a 30 day period
13	address_distinct_zip3	the number of distinct zip5 associated with one particular address over a 3 day period
14	address_distinct_zip7	the number of distinct zip5 associated with one particular address over a 7 day period
15	address_distinct_zip14	the number of distinct zip5 associated with one particular address over a 14 day period
16	address_distinct_zip30	the number of distinct zip5 associated with one particular address over a 30 day

		period
17	address_distinct_fullname_birthday3	the number of distinct fullname_birthday (concatenation of firstname, lastname, and dob) associated with one particular address over a 3 day period
18	address_distinct_fullname_birthday7	the number of distinct fullname_birthday (concatenation of firstname, lastname, and dob) associated with one particular address over a 7 day period
19	address_distinct_fullname_birthday14	the number of distinct fullname_birthday (concatenation of firstname, lastname, and dob) associated with one particular address over a 14 day period
20	address_distinct_fullname_birthday30	the number of distinct fullname_birthday (concatenation of firstname, lastname, and dob) associated with one particular address over a 30 day period
21	person_dob3	the number of person associated with one particular dob (date of birth) code over a 3 day period
22	person_dob7	the number of person associated with one particular dob (date of birth) code over a 7 day period

23	person_dob14	the number of person associated with one particular dob (date of birth) code over a 14 day period
24	person_dob30	the number of person associated with one particular dob (date of birth) code over a 30 day period
25	dob_distinct_homephone3	the number of distinct homephone associated with one particular date of birth over a 3 day period
26	dob_distinct_homephone7	the number of distinct homephone associated with one particular date of birth over a 7 day period
27	dob_distinct_homephone14	the number of distinct homephone associated with one particular date of birth over a 14 day period
28	dob_distinct_homephone30	the number of distinct homephone associated with one particular date of birth over a 30 day period
29	dob_distinct_ssn3	the number of distinct ssn associated with one particular date of birth over a 3 day period
30	dob_distinct_ssn7	the number of distinct ssn associated with

		one particular date of birth over a 7 day period
31	dob_distinct_ssn14	the number of distinct ssn associated with one particular date of birth over a 14 day period
32	dob_distinct_ssn30	the number of distinct ssn associated with one particular date of birth over a 30 day period
33	dob_distinct_address3	the number of distinct address associated with one particular date of birth over a 3 day period
34	dob_distinct_address7	the number of distinct address associated with one particular date of birth over a 7 day period
35	dob_distinct_address14	the number of distinct address associated with one particular date of birth over a 14 day period
36	dob_distinct_address30	the number of distinct address associated with one particular date of birth over a 30 day period
37	person_homephone3	the number of person associated with one particular homephone over a 3 day period

38	person_homephone7	the number of person associated with one particular homephone over a 7 day period
39	person_homephone14	the number of person associated with one particular homephone over a 14 day period
40	person_homephone30	the number of person associated with one particular homephone over a 30 day period
41	homephone_distinct_address3	the number of distinct address associated with one particular homephone over a 3 day period
42	homephone_distinct_address7	the number of distinct address associated with one particular homephone over a 7 day period
43	homephone_distinct_address14	the number of distinct address associated with one particular homephone over a 14 day period
44	homephone_distinct_address30	the number of distinct address associated with one particular homephone over a 30 day period
45	homephone_distinct_ssn3	the number of distinct ssn associated with one particular homephone over a 3 day

		period
46	homephone_distinct_ssn7	the number of distinct ssn associated with one particular homephone over a 7 day period
47	homephone_distinct_ssn14	the number of distinct ssn associated with one particular homephone over a 14 day period
48	homephone_distinct_ssn30	the number of distinct ssn associated with one particular homephone over a 30 day period
49	homephone_distinct_zip3	the number of distinct zip5 associated with one particular homephone over a 3 day period
50	homephone_distinct_zip7	the number of distinct zip5 associated with one particular homephone over a 7 day period
51	homephone_distinct_zip14	the number of distinct zip5 associated with one particular homephone over a 14 day period
52	homephone_distinct_zip30	the number of distinct zip5 associated with one particular homephone over a 30 day period

53	homephone_distinct_fullname_birthday3	the number of distinct fullname_birthday associated with one particular homephone over a 3 day period
54	homephone_distinct_fullname_birthday7	the number of distinct fullname_birthday associated with one particular homephone over a 7 day period
55	homephone_distinct_fullname_birthday14	the number of distinct fullname_birthday associated with one particular homephone over a 14 day period
56	homephone_distinct_fullname_birthday30	the number of distinct fullname_birthday associated with one particular homephone over a 30 day period
57	person_person3	the number of people associated with the same person (person being a concatenation of firstname, lastname, ssn, and date of birth) over a 3 day period
58	person_person7	the number of people associated with the same person (person being a concatenation of firstname, lastname, ssn, and date of birth) over a 7 day period
59	person_person14	the number of people associated with the same person (person being a concatenation of firstname, lastname, ssn,

		and date of birth) over a 14 day period
60	person_person30	the number of people associated with the same person (person being a concatenation of firstname, lastname, ssn, and date of birth) over a 30 day period
61	person_distinct_address3	the number of distinct address associated with the same person (person being a concatenation of firstname, lastname, ssn, and dob) over a 3 day period
62	person_distinct_address7	the number of distinct address associated with the same person (person being a concatenation of firstname, lastname, ssn, and dob) over a 7 day period
63	person_distinct_address14	the number of distinct address associated with the same person (person being a concatenation of firstname, lastname, ssn, and dob) over a 14 day period
64	person_distinct_address30	the number of distinct address associated with the same person (person being a concatenation of firstname, lastname, ssn, and dob) over a 30 day period
65	person_distinct_zip3	the number of distinct zip5 associated with the same person (person being a

		concatenation of firstname, lastname, ssn, and dob) over a 3 day period
66	person_distinct_zip7	the number of distinct zip5 associated with the same person (person being a concatenation of firstname, lastname, ssn, and dob) over a 7 day period
67	person_distinct_zip14	the number of distinct zip5 associated with the same person (person being a concatenation of firstname, lastname, ssn, and dob) over a 14 day period
68	person_distinct_zip30	the number of distinct zip5 associated with the same person (person being a concatenation of firstname, lastname, ssn, and dob) over a 30 day period
69	person_ssn3	the number of people with the same ssn in a 3-day time window
70	person_ssn7	the number of people with the same ssn in a 7-day time window
71	person_ssn14	the number of people with the same ssn in a 14-day time window
72	person_ssn30	the number of people with the same ssn in a 30-day time window

73	ssn_distinct_address3	the number of people with the same ssn but different addresses over a 3 day period
74	ssn_distinct_address7	the number of people with the same ssn but different addresses over a 7 day period
75	ssn_distinct_address14	the number of people with the same ssn but different addresses over a 14 day period
76	ssn_distinct_address30	the number of people with the same ssn but different addresses over a 30 day period
77	ssn_distinct_phone3	the number of distinct phone number associated with one particular ssn over a 3 day period
78	ssn_distinct_phone7	the number of distinct phone number associated with one particular ssn over a 7 day period
79	ssn_distinct_phone14	the number of distinct phone number associated with one particular ssn over a 14 day period
80	ssn_distinct_phone30	the number of distinct phone number

		associated with one particular ssn over a 30 day period
81	ssn_distinct_zip3	the number of distinct zip5 associated with one particular ssn over a 3 day period
82	ssn_distinct_zip7	the number of distinct zip5 associated with one particular ssn over a 7 day period
83	ssn_distinct_zip14	the number of distinct zip5 associated with one particular ssn over a 14 day period
84	ssn_distinct_zip30	the number of distinct zip5 associated with one particular ssn over a 30 day period
85	ssn_distinct_fullname_birthday3	the number of distinct fullname_birthday (concatenation of firstname, lastname, and dob) associated with one particular ssn over a 3 day period
86	ssn_distinct_fullname_birthday7	the number of distinct fullname_birthday (concatenation of firstname, lastname, and dob) associated with one particular ssn over a 7 day period

87	ssn_distinct_fullname_birthday14	the number of distinct fullname_birthday (concatenation of firstname, lastname, and dob) associated with one particular ssn over a 14 day period
88	ssn_distinct_fullname_birthday30	the number of distinct fullname_birthday (concatenation of firstname, lastname, and dob) associated with one particular ssn over a 30 day period
89	person_zip3	the number of person associated with one particular zip code over a 3 day period
90	person_zip7	the number of person associated with one particular zip code over a 7 day period
91	person_zip14	the number of person associated with one particular zip code over a 14 day period
92	person_zip30	the number of person associated with one particular zip code over a 30 day period
93	zip_distinct_homephone3	the number of distinct homephone associated with one particular zip code over a 3 day period
94	zip_distinct_homephone7	the number of distinct homephone associated with one particular zip code over a 7 day period

95	zip_distinct_homephone14	the number of distinct homephone associated with one particular zip code over a 14 day period
96	zip_distinct_homephone30	the number of distinct homephone associated with one particular zip code over a 30 day period
97	zip_distinct_ssn3	the number of distinct ssn associated with one particular zip code over a 3 day period
98	zip_distinct_ssn7	the number of distinct ssn associated with one particular zip code over a 7 day period
99	zip_distinct_ssn14	the number of distinct ssn associated with one particular zip code over a 14 day period
100	zip_distinct_ssn30	the number of distinct ssn associated with one particular zip code over a 30 day period
101	zip_distinct_address3	the number of distinct address associated with one particular zip code over a 3 day period
102	zip_distinct_address7	the number of distinct address associated

		with one particular zip code over a 7 day period
103	zip_distinct_address14	the number of distinct address associated with one particular zip code over a 14 day period
104	zip_distinct_address30	the number of distinct address associated with one particular zip code over a 30 day period
105	zip_distinct_fullname_birthday3	the number of distinct fullname_birthday (concatenation of firstname, lastname, and dob) associated with one particular zip code over a 3 day period
106	zip_distinct_fullname_birthday7	the number of distinct fullname_birthday (concatenation of firstname, lastname, and dob) associated with one particular zip code over a 7 day period
107	zip_distinct_fullname_birthday14	the number of distinct fullname_birthday (concatenation of firstname, lastname, and dob) associated with one particular zip code over a 14 day period
108	zip_distinct_fullname_birthday30	the number of distinct fullname_birthday (concatenation of firstname, lastname, and dob) associated with one particular

		zip code over a 30 day period
109	ssn_previous_fraud	whether the applicant's ssn associated with a previous identified fraud(1-Yes,0-No)
110	address_previous_fraud	whether the applicant's address associated with a previous identified fraud(1-Yes,0-No)
111	homephone_previous_fraud	whether the applicant's home phone associated with a previous identified fraud(1-Yes,0-No)
112	standard_ssn	whether the applicant's ssn is standard format(9 digit) (not standard-1,standard-0)
113	standard_homephone	whether the applicant's home phone is standard format(10 digit) (not standard-1,standard-0)

As shown in the table, the new features included various combinations of categorical, date, and text variables. We created a new dataset containing all of the above listed new features selected specifically from the dataset containing our special variables. We used this new dataset of expert variables for our performance tests and feature subset selection process.

After creating the expert variables, we performed feature selection algorithms on each one to select the most important variables. For each of the expert variables, we plotted separate distributions for the two cases: fraudulent and non-fraudulent. The goal is to filter the most important expert variables by choosing those that have a high amount of separation between the distributions.

By using these feature selection algorithms, we were able to compare whether the expert variables capturing shorter or longer periods were more important in our fraud detection analysis.

KOLMOGOROV-SMIRNOV TEST

DEFINITION

The Kolmogorov-Smirnov test (KS test) is a nonparametric test that compares the cumulative distributions of two data sets. Since it is nonparametric, it does not assume that data are sampled from any defined distribution. The null hypothesis is that both groups were sampled from populations with identical distributions. It tests for any difference in medians, variances or distributions. The P value represents the probability that the two cumulative frequency distributions would be as far as observed, if the two samples were randomly sampled from identical populations. KS test is used frequently as it is a simple distribution independent measure of distance between two distributions.

CALCULATION

The following formula calculates the Kolmogorov-Smirnov statistic, KS, by finding the maximum difference between the cumulative true positive and cumulative false positive rate.

$$KS = max \sum_{x_{min}}^{x} [P_{good} - P_{bad}]$$

To calculate the KS statistic for all of our expert variables, we created a function in R built on the ROCR library. Our code calculates the KS using the ROC curve and the parameters from the logistic regression. We fit a generalized linear model by setting up a function called myks.

In this function, we first ran the glm() command to perform the fitting with the following parameters:

- family=binomial, to describe the error distribution and link function to use in the model
- data=train, to specify the function to use the training data

We also ran the predict() function to calculate the predicted probability and included the following argument:

- object=glm.fit, to specify the model to be fitted as the generalized linear model we fitted previously
- type = “response”, to get our prediction

To calculate the KS statistic, we took the maximum difference between cumulative bad and good rates being plotted by ROCR using the following code on R:

$$max(attr(perf, "y.values")[[1]] - attr(perf, "x.values")[[1]])$$

APPLICATION

We first took the first 10 months of the dataset and divided it into training (80%) and testing (20%) by random sampling. The data from the last two months of the year 2016 (consisting of day of year from 306 to 366) were set aside for out-of-time validation.

We split our data into 3 time segments as follows:

1. Training (80%, Randomly chosen from Jan - Oct 2016)
2. Testing (20%, Randomly chosen from Jan - Oct 2016)
3. Out of Time (Nov - Dec 2016, specifically, day 306 - 366)

We then calculated the KS statistic on the training data set using the formula shown above. This value represents the degree of separation between the two distributions (fraud = 1 and fraud = 0) under each of the 113 expert variables created. The higher the separation, the higher the measure of goodness and importance of the variable. The KS statistic acts as a measure importance for each of the variables. Using this measure, we performed filter feature selection to sort in descending order and select the top 40 most important expert variables with the highest KS values.

Below is a table of the top 40 expert variables along with their KS values.

Table 7. List of Top 40 Expert Variables and KS Values

Expert Variable	KS Value
person_ssn30	0.04388
person_zip30	0.03997
person_address30	0.03891
person_ssn14	0.03852
person_zip14	0.03551
person_ssn7	0.03545
person_address_14	0.03368
person_zip_7	0.03168
person_address_7	0.03066
person_ssn_3	0.02940
person_zip_3	0.02913
homephone_distinct_zip7	0.02759
homephone_distinct_address7	0.02757

homephone_distinct_ssn7	0.02753
homephone_distinct_bdname7	0.02742
person_address_3	0.02518
homephone_distinct_zip3	0.02479
homephone_distinct_address3	0.02475
homephone_distinct_ssn3	0.02471
homephone_distinct_bdname3	0.02456
zip_distinct_homephone30	0.02239
zip_distinct_ssn30	0.02218
zip_distinct_bdname30	0.02204
zip_distinct_address30	0.02199
homephone_distinct_zip14	0.02188
homephone_distinct_address14	0.02186
homephone_distinct_ssn14	0.02176
homephone_distinct_bdname14	0.02169
homephone_distinct_zip1	0.02092

homephone_distinct_zip1.1	0.02092
homephone_distinct_address1	0.02088
homephone_distinct_bdname1	0.02076
person_ssn_1	0.01981
person_zip_1	0.01870
person_homephone_3	0.01768
person_homephone_7	0.01762
person_homephone_14	0.01755
homephone_distinct_address30	0.01753
homephone_distinct_bdname30	0.01753
homephone_distinct_zip30	0.01753

BACKWARD SELECTION

DEFINITION

After running KS and reducing our expert variables down to 40, we performed Stepwise Backward Selection to filter the set of features/variables that produce optimal model performance. This is a wrapper method that takes out features that reduce model performance. At each iteration, different feature combinations are selected as input and evaluated using the predictive model.

Stepwise Backward Selection begins with all features/variables and builds a single model. The model removes one feature at each iteration and checks model performance. After completing backward selection, the final set of features that results in the highest model performance is selected.

CALCULATION

The Stepwise Backward Selection methodology is as follows:

1. Begin with all n features/variables and build a single model
2. Build n models for each variable, independently, each removing one variable
3. Among n models, select best model that degrades the least (consisting of n-1 variables)
4. Build n-1 separate models, each removing one variable (such that each model consists of n-2 variables)
5. Among n-1 models, select the best model that degrades the least (consisting of n-2 variables)
6. Build n-2 separate models, each removing one variable (such that each model consists of n-3 variables)

7. Among n-2 models, select the best model that degrades the least(consisting of n-3 variables)
8. Continue process until model degradation below certain degree

APPLICATION

First, we used the glm function in R to fit a generalized linear model called fullmod using all 40 expert variables. We set the following arguments: family=binomial and data=train.

$$fullmod = glm(fraud\tilde{}\ ., data = train, family = binomial)$$

After running the full model with all variables, we ran the step function in R to choose a model by AIC in a stepwise algorithm. The step function takes the model as the object argument. We set this value object = fullmod so that our full model would be used as the initial model in the stepwise search.

$$backwards = step(fullmod)$$

For the full model containing all 40 expert variables, the initial AIC = 62820.4.

After performing Stepwise Backward Selection and dropping variables that reduce model performance, the best model had an improved AIC = 62807.29. According to the step function we ran, this is the best model with the lowest AIC possible (13.11 lower than the initial value). The final set of variables producing the model of top performance had a total of 10 variables, meaning that after performing all iterations of the Stepwise Backward Selection process, a total of 20 variables that reduced model performance were dropped.

Below is the list of the top 10 expert variables we kept for building our models.

Table 8. List of Top 10 Expert Variables

Expert Variable	Description
person_address_3	the number of person associated with one particular address over a 3 day period
person_address_7	the number of person associated with one particular address over a 7 day period
person_address_30	the number of person associated with one particular address over a 30 day period
person_homephone14	the number of person associated with one particular homephone over a 14 day period
homephone_distinct_address3	the number of distinct address associated with one particular homephone over a 3 day period
homephone_distinct_address7	the number of distinct address associated with one particular homephone over a 7 day period
homephone_distinct_address30	the number of distinct address associated with one particular homephone over a 30 day period
homephone_distinct_zip3	the number of distinct zip5 associated with one particular homephone over a 3 day period

homephone_distinct_zip7	the number of distinct zip5 associated with one particular homephone over a 7 day period
person_ssn7	the number of people with the same ssn in a 7-day time window

DATA MODELING

ALGORITHMS

For the purpose of this project, we performed 4 supervised machine learning algorithms to calculate the Fraud Detection Rate (FDR):

- Logistic Regression
- Gradient Boosted Trees
- Support Vector Machine (SVM)
- Random Forests

In the following sections, we will briefly describe each algorithm, the associated mathematical formulas, along with the application to our project.

LOGISTIC REGRESSION

DEFINITION

Logistic regression is a supervised classification algorithm to find the best fitting model to describe the relationship between a dependent variable and a set of independent variables. It is a commonly used model in replacement with linear regression when the dependent variable is categorical. It models the conditional probability f(x) = P(Y=1|X). In our case, the dependent variable (fraud) is either 0 or 1, which is binary, therefore logistic regression is a simple and good model to start with.

A primary issue that needs consideration surrounding logistic regression is overfitting. This is often caused by adding too many variables to the model. To avoid overfitting, we can increase size of the training data, stop iterations before the model begins to overfit and through regularization.

Major assumptions when applying binary logistic regression are:

1. The dependent variable should be binary in nature;
2. There should be no outliers in the data, which have already been handled in the data cleaning step;
3. There should be no high correlations among the predictors.

CALCULATION

Instead of minimizing the sum of squared errors like in linear regression, logistic regression aims to maximize the likelihood of observing the sample values. It generates the coefficients of a formula to predict a logit transformation of the probability presence of the outcome.

$$logit\ (p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \ldots + \beta_k X_k$$

where p is the probability of presence of the outcome, P(Y=1|X). The logit transformation is defined as the logged odds:

$$odds = \frac{p}{1-p} = \frac{probability\ of\ presence\ of\ the\ outcome}{probability\ of\ absence\ of\ the\ outcome}$$

and

$$logit(p) = ln(\frac{p}{1-p})$$

The logistic function is therefore as follows:

$$f(x) = p(x) = \frac{e^{\beta_0+\beta_1x_1+\beta_2x_2+\beta_3x_3+\ ...\ +\beta_kx_k}}{1 + e^{\beta_0+\beta_1x_1+\beta_2x_2+\beta_3x_3+\ ...\ +\beta_kx_k}}$$

APPLICATION

To run the Logistic Regression on the Applications dataset, we used the package stats in R to build our Fraud Detection Model. In R, there are mainly two important functions in order to perform a Logistic Regression:

- glm () function to perform generalized linear models (regressions) on binary outcome data, count data, probability data, proportion data and many other data types.
- predict () function to calculate the predicted probability. We include the argument type="response" in order to get our prediction.

Using the training data set as split above when calculating KS score and performing feature selection, we ran the logistic regression model with the glm( ) function on R with fraud as the dependent categorical variable:

glm.fit=glm(fraud ~ person_address_3 + person_address_7 + person_address_30 + person_homephone14 + homephone_distinct_address3 + homephone_distinct_address7 + homephone_distinct_address3 + homephone_distinct_zip3 + homephone_distinct_zip7 + person_ssn7, data = train, family = 'binomial')

We then apply the predict function on the test data from the Logistic Regression model:

$$glm.probs = predict(glm.fit, test,\ type = "response")$$

This function predicts the response variable, giving us the number of frauds caught for each record from the Logistic Regression model.

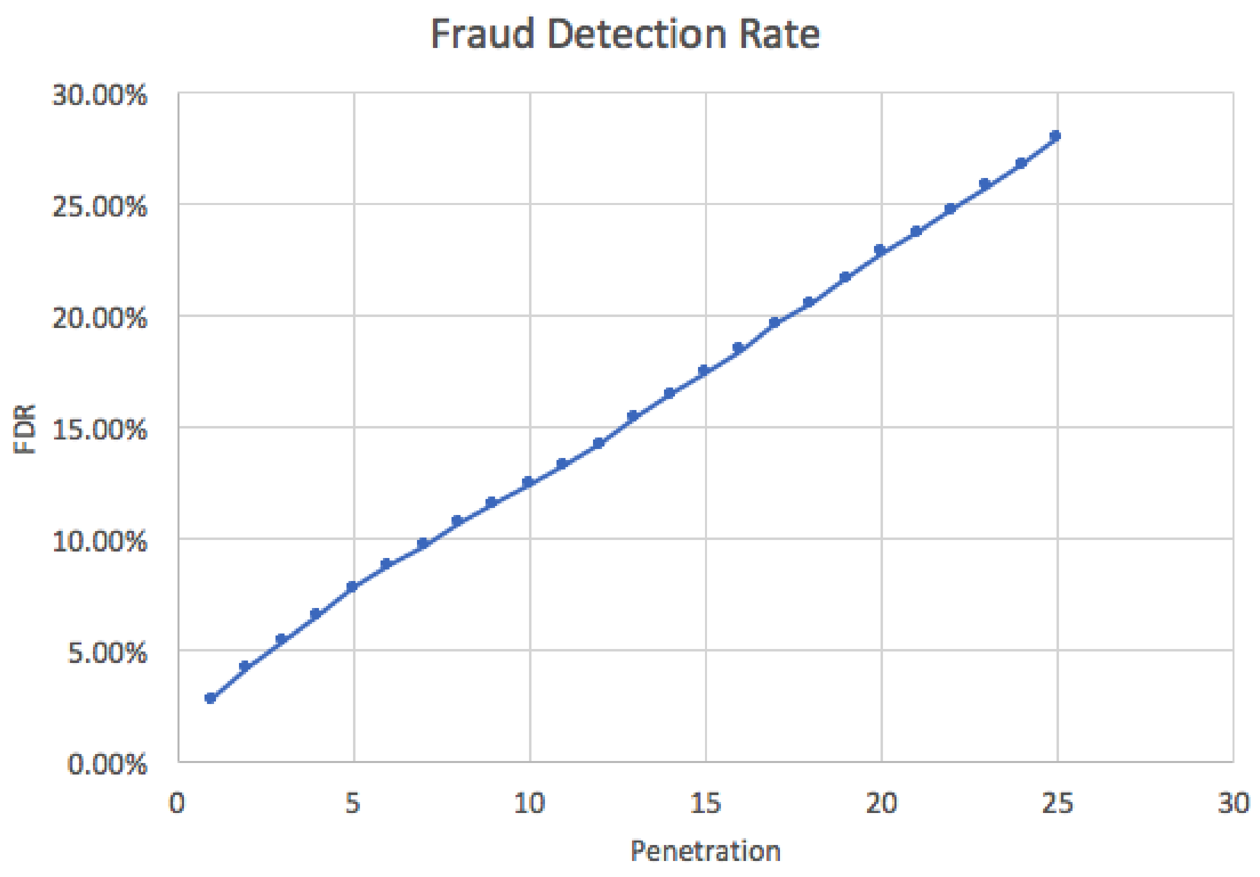
To calculate the fraud detection rate, we first sorted all records by the highest score to the lowest score in order to examine each population bin.

For each population bin, we summed the total number of frauds caught from the logistic regression model and then divided by the total number of frauds model in that specific population. Here we are calculating the fraud detection rate at 10% penetration rate.

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

We then applied the same model to the testing and out of time dataset to build the fraud detection rate at 10% penetration.Here is a graph plotting the fraud detection rates for the top 20% populations bins for the Logistic Regression model on the OOT dataset.

Graph 1. Fraud Detection Rate on the OOT Dataset for the Logistic Regression Model



GRADIENT BOOSTED TREES

DEFINITION

Gradient boosting is a machine learning technique that produces a prediction model in the form of an ensemble of weak decision trees. It builds the model in a stage-wise fashion like other boosting methods and generalizes them by allowing optimization of an arbitrary differentiable loss function.

Like other boosting methods, gradient boosting combines weak "learners" into a single strong learner in an iterative fashion.

At each stage m, $1 \leq m \leq M$, of gradient boosting, it may be assumed that there is some imperfect model F_m (at the outset, a very weak model that just predicts the mean y in the training set could be used). The gradient boosting algorithm improves on F_m by constructing a new model that adds an estimator h to provide a better model: $F_{m+1}(x) = F_m(x) + h(x)$ To find h, the gradient boosting solution starts with the observation that a perfect h would imply

$$F_{m+1}(x) = F_m(x) + h(x) = y$$

or, equivalently,

$$h(x) = y - F_m(x)$$

Therefore, gradient boosting will fit h to the residual $y - F_m(x)$ Like in other boosting variants, each F_{m+1} learns to correct its predecessor F_m. A generalization of this idea to loss functions other than squared error — and to classification and ranking problems — follows from the observation that residuals $y - F(x)$ for a given model are the negative gradients (with respect to $F(x)$) of the squared error loss function $\frac{1}{2}(y - h(x))^2$. So, gradient boosting is a gradient descent algorithm and generalizing it entails "plugging in" a different loss and its gradient.

CALCULATION

Gradient boosting involves three elements:

1. A loss function to be optimized.
2. A weak learner to make predictions.
3. An additive model to add weak learners to minimize the loss function.

1. Loss Function

The loss function used depends on the type of problem being solved. It is generally differentiable. A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used.

2. Weak Learner

Generally, Decision trees are used as the weak learner in gradient boosting. Specifically, regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and “correct” the residuals in the predictions. Trees are constructed in a greedy manner, choosing the best split points based on purity scores to minimize the loss.

It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes. This is to ensure that the learners remain weak but can still be constructed in a greedy manner.

3. Additive Model

Trees are added one at a time, and existing trees in the model are not changed. A gradient descent procedure is used to minimize the loss when adding trees. Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error. After calculating the loss, to perform the gradient descent procedure, we add a tree to the model that reduces the loss. We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by reducing the residual loss. The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model. A fixed number of trees are added, or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset.

Algorithm:

Input: training set $\{(x_{i,} \, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

1. Initialize model with a constant value:

$$F_0(x) = arg_\gamma \, min \sum_{i=1}^{n} L(y_i \, , \gamma) \, .$$

2. For $m = 1 \; to \; M$:
    a. Compute pseudo-residuals:

$$r_{im} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad for \; i = 1, \ldots., n.$$

    b. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
    c. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = arg_\gamma \, min \sum_{i=1}^{n} L(y_i \, , F_{m-1}(x_i) \; + \; \gamma h_m(x_i)) \, .$$

3. Update the model: $F_m(x) = \; F_{m-1}(x) + \gamma h_m(x_i)$
4. Output $F_m(x)$

APPLICATION

To run the Gradient Boosted Trees on the Applications dataset, we loaded the gbm package on R. The gbm package contains mainly two important function:

- gbm() the function fits generalized boosted regression models. The formula includes one important argument within the gbm () function called "distribution." Distribution can either be a character string specifying the name of the distribution to use or a list with a component name specifying the distribution and any additional parameters needed. If not specified will try to guess: if the response has only 2 unique values, Bernoulli is assumed; otherwise, if the response is a factor, multinomial is assumed; otherwise Gaussian is assumed.

- predict () function produces predicted values for each observation in newdata using the the first n.trees iterations of the boosting sequence. If n.trees is a vector than the result is a matrix with each column representing the predictions from gbm models with n.trees[1] iterations, n.trees[2] iterations, and so on.

Using the training data set as split above when calculating KS score and performing feature selection, we ran the gradient boosted tree with the gbm( ) function on R with fraud as the dependent categorical variable:

> *gbm.fit = gbm(fraud ~ person_address_3 + person_address_7 + person_address_30 + person_homephone14 + homephone_distinct_address3 + homephone_distinct_address7 + homephone_distinct_address3 + homephone_distinct_zip3 + homephone_distinct_zip7 + person_ssn7, data = train, distribution = "bernoulli", n.trees = 1000, shrinkage = 0.01, interaction.depth = 4)*

We then used the distribution method as "bernoulli" since the response variable has only 2 unique values, and we set the number of trees to 1000 and the number of nodes per tree to be 4.

We then apply the predict function on the test data from the gbm model:

$$gbm.probs = predict(gbm.fit,\ test,\ n.trees = 1000)$$

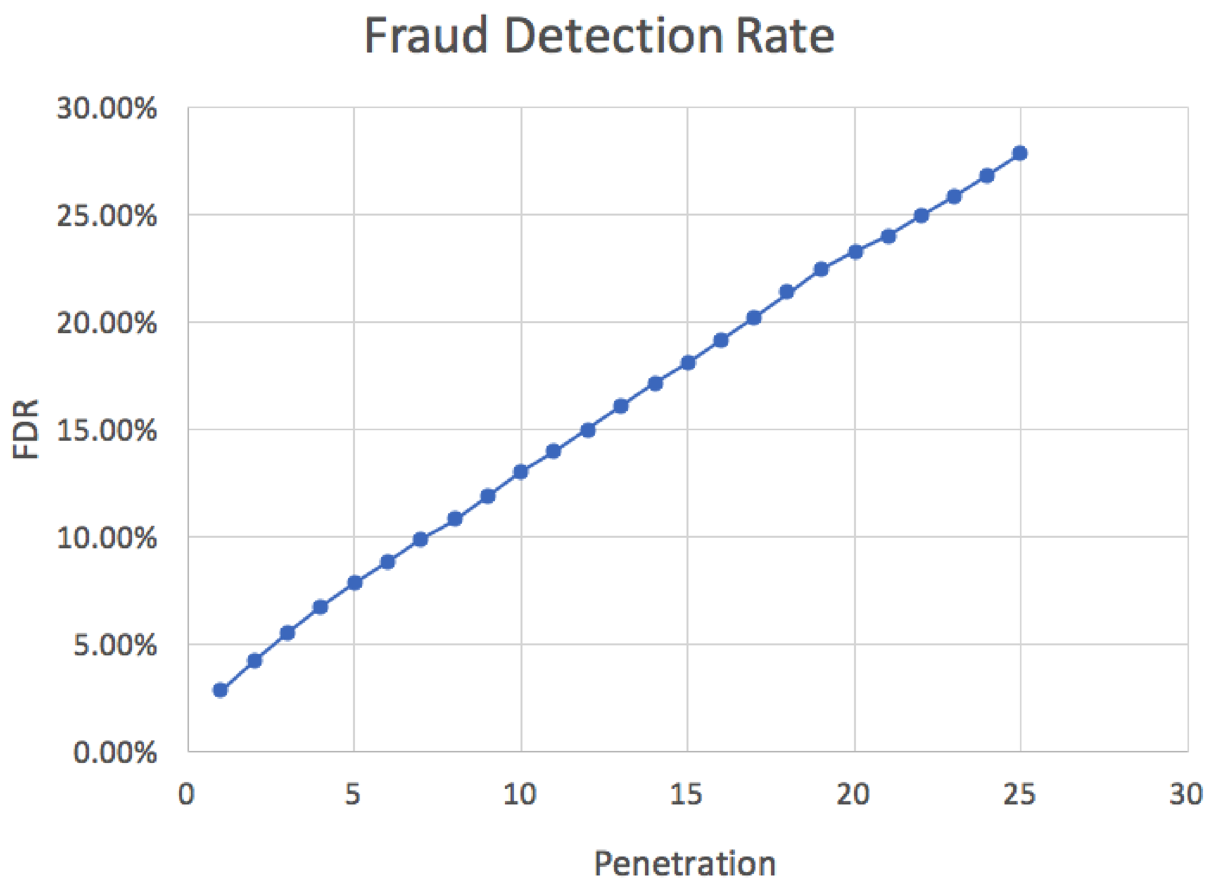
This function predicts the response variable, giving us the number of frauds caught for each record from the gradient boosted tree.

To calculate the fraud detection rate, we first sorted all records by the highest score to the lowest score in order to examine each population bin. For each population bin, we summed the total number of frauds caught from the gradient boosted tree model and then divided by the total number of frauds model in that specific population. Here we are calculating the fraud detection rate at 10% penetration rate.

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

We then applied the same model to the testing and out of time dataset to build the fraud detection rate at 10% penetration. Here is a graph plotting the fraud detection rates for the top 20% populations bins for the Gradient Boosted Trees model on the OOT dataset.

Graph 2. Fraud Detection Rate on the OOT Dataset for the Gradient Boosted Trees Model



SUPPORT VECTOR MACHINE (SVM)

DEFINITION

The support vector machines (SVMs) are a set of supervised learning models that constructs a hyperplane or a set of hyperplanes in a high- or infinite-dimensional space used for classification, regression and outliers detection.

Given a set of labeled training data, the SVM algorithm outputs an optimal hyperplane which categorizes new examples. The algorithm is performed by finding the hyperplane that gives the largest minimum distance to the training examples. In other words, a good separation of examples is defined by the hyperplane that has the largest distance to the nearest training-data point of any class, since the larger the margin the lower the generalization error of the classifier. Thus, the optimal separating hyperplane *maximizes* the margin of the training data. The vectors (cases) that define the hyperplane are the *support vectors*.

CALCULATION

Algorithm:

1. Define an optimal hyperplane that maximizes the margin between classes.
2. Extend the definition above for non-linearly separable problems - have a penalty term for misclassifications.
3. Map data to a high-dimensional space where it is easier to classify with linear decision surfaces. Reformulate the problem so that data is mapped implicitly to this space.

Linear SVM

If the training data is linearly separable, two parallel hyperplanes are selected to separate the two classes of data points, thus maximizing the distance between them. The region bounded by these hyperplanes is the “margin”, and the maximum-margin hyperplane is the hyperplane that lies halfway between them.

Given a training dataset of n points of the form

$$(\vec{x}_1, y_1), \ \dots \ , (\vec{x}_n, y_n)$$

where the y_i are either 1 or -1, each indicating the class to which point $\vec{x}_i$ belongs. Each $\vec{x}_i$ is a *p*-dimensional real vector. Maximum-margin hyperplane divides the group of points $\vec{x}_i$ for which $y_i = 1$ from the group of points for which $y_i = -1$. In other words, the optimal hyperplane can be defined by maximizing the width of the margin (w).

Geometrically, the distance between the two hyperplanes is $\frac{2}{||\vec{w}||}$. To maximize the distance, we need to minimize $||\vec{w}||$. To prevent the data points from falling into the margin, we need to add the following constraints:

$$\vec{w} \cdot \vec{x}_i - b \geq 1, \ if \ y_i = 1$$

or

$$\vec{w} \cdot \vec{x}_i - b \leq -1, \ if \ y_i = -1 .$$

This can be rewritten as:

$$y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \ for \ all \ 1 \leq i \leq n .$$

Non-linear SVM

The simplest way to separate two groups of data is with a straight line, flat plane, or an N-dimensional hyperplane. However, there are situations where the data sets to discriminate are not linearly separable in that space. In this case, the original finite-dimensional space is mapped into a higher-dimensional space, making the separation easier in that space.

The SVM is used to enlarge the feature space using a *kernels* in order to accommodate a non-linear boundary between classes. To make it possible to perform the linear separation, *kernel trick* approach is used to transform the data into a higher dimensional, implicit feature space without computing the coordinates of the data in that space, but rather by computing the *inner products* between the images of all pairs of data in the feature space. In other words, every dot product is replaced by a nonlinear kernel function, thus fitting the maximum-margin hyperplane in a transformed feature space.

A nonlinear classification rule corresponds to a linear classification rule for the transformed data points $\phi(\vec{x}_i)$. Moreover, we are given a kernel function k which satisfies

$$k(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j).$$

The classification vector, $\vec{w}$, in the transformed space also satisfies

$$\vec{w} = \sum_{i=1}^{n} c_i y_i \phi(\vec{x}_i),$$

where the c_i are obtained by solving the optimization problem.

And, the dot products with w can be computed by the kernel trick:

$$\vec{w} \cdot \phi(\vec{x}) = \sum_i c_i y_i \phi(\vec{x}_i, \vec{x}).$$

The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant. If $k(x,y)$ becomes small as y grows further away from x, each term in the sum measures the degree of closeness of the test point x to the corresponding data base point x_i. Therefore, the sum of kernels can be used to measure the relative nearness of each test point to the training data points.

Some common kernels include:

- Polynomial (homogeneous): $k(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j)^d$
- Polynomial (non-homogeneous): $k(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^d$

- Gaussian radial basis function: $k(\vec{x}_i, \vec{x}_j) = exp\,(-\gamma\|\vec{x}_i - \vec{x}_j\|^2)$, for $\gamma > 0$.

APPLICATION

In order to run the Support Vector Machine (SVM) Algorithm on the Applications dataset, we loaded a library package called "parallelSVM" on R.

This package consists of two main functions:

- parallelSVM: a function which allows you to create multiple Support Vector Machine models: one for each core you provide. It returns a list of Support-Vector-Machine models.
- predict: An extension of the predict function, which uses the prediction of each Support-Vector-Machine model. When probability is TRUE, it returns the average of all predictions, otherwise it returns the class most models agree upon.

Using the training data set as split above when calculating KS score and performing feature selection, we ran the Support Vector Machine model with the parallelSVM( ) function on R with fraud as the dependent categorical variable:

svm_model = parallelSVM (fraud ~ person_address_3 + person_address_7 + person_address_30 + person_homephone14 + homephone_distinct_address3 + homephone_distinct_address7 + homephone_distinct_address3 + homephone_distinct_zip3 + homephone_distinct_zip7 + person_ssn7, data = train, family = 'binomial')

We then apply the predict() function on the test data from the Support Vector Machine Model:

$$svm.probs = predict(svm\_model, test)$$

This function predicts the response variable, giving us the number of frauds caught for each record from the Support Vector Machine model.

To calculate the fraud detection rate, we first sorted all records by the highest score to the lowest score in order to examine each population bin.

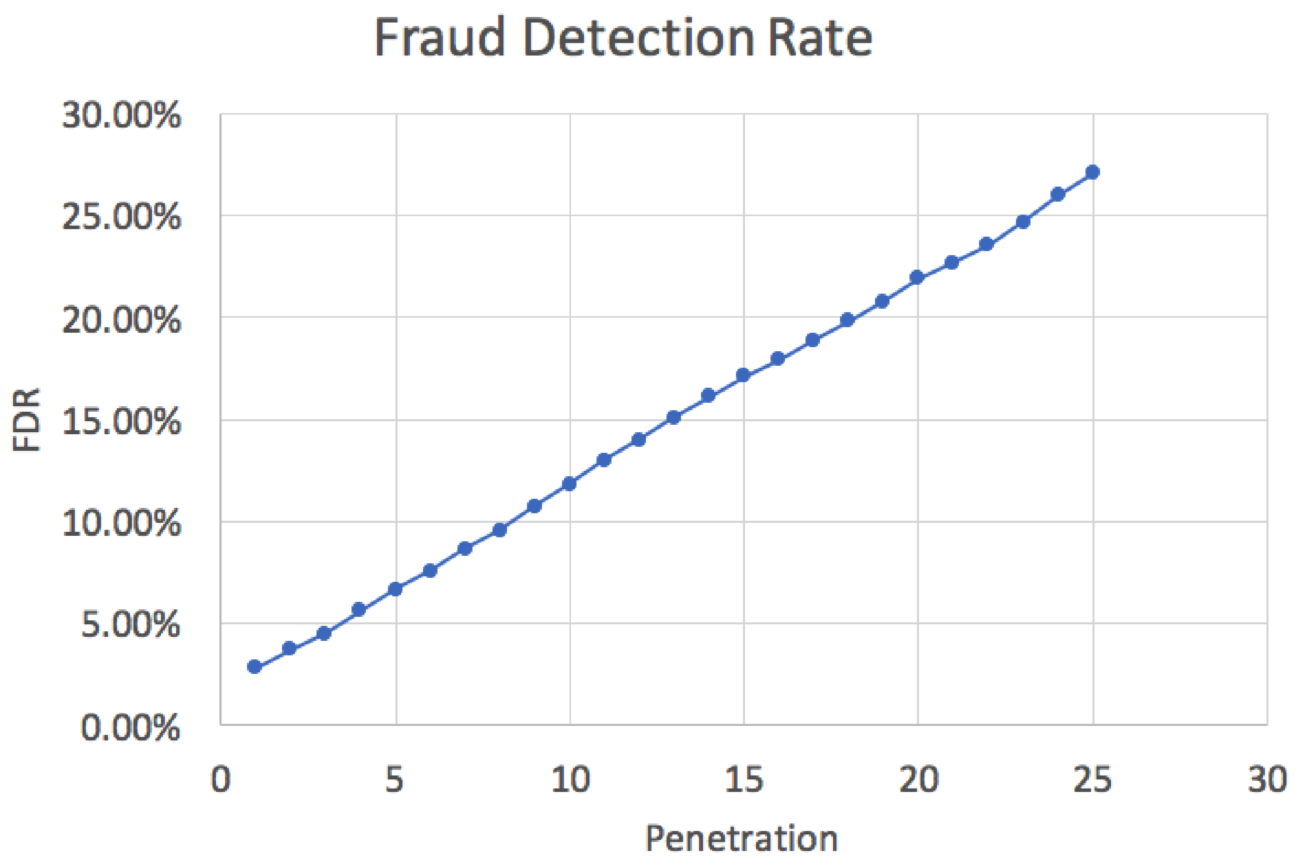
For each population bin, we summed the total number of frauds caught from the Support Vector Machine model and then divided by the total number of frauds model in that specific population. Here we are focusing on calculating the fraud detection rate at 10% penetration.

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

We then applied the same model to the testing and out of time dataset to build the fraud detection rate at 10% penetration.

Here is a graph plotting the fraud detection rates for the top 20% populations bins for the Support Vector Machine model on the OOT dataset.

Graph 3. Fraud Detection Rate on the OOT Dataset for the Support Vector Machine Model



RANDOM FORESTS

DEFINITION

Random forest is a supervised classification algorithm that is built on a random subset of decision trees. It acts as an ensemble model because it takes a combination of decision trees to create a superior model. One of the greatest pitfalls of the decision tree is overfitting. However, by growing a lot of trees and voting on the outcome across all of them, we can bypass this limitation. In other words, with enough trees in the model, the random forest will not overfit the model in our classification problem.

Among the collection of trees in the random forest, each tree is built independently on a random subset of features. To create completely unique individual trees within the collection of trees, random forests incorporate two critical aspects: the first is by performing bagging, for bootstrap aggregating, and the second is by looking at only a subset of variables. Bagging involves taking a randomized sample of rows from the training set, with replacement. With every iteration in bagging, rows are repeated and omitted, and each tree evolves slightly. Random forests take into account only a subset of the entire pool of available variables, and this selection is changed for every single node of the decision tree. Every tree in the forest is grown out fully and overfits in different ways, so the individual mistakes among each of them are averaged across the entire collection. The higher the number of trees, the more robust the model and the higher the accuracy of the results.

CALCULATION

There are two phases of the Random Forest algorithm, described briefly below:

1) **Random Forest Creation**
    a) Randomly select K features from total m features (k << m)
    b) Among K features, use best split point to calculate node d
    c) Use best split to split node into daughter nodes
    d) Repeat a-c until desired number of nodes has been reached

e) Repeat a-d, n number of times to create n trees, to create a forest

2) Random Forest Prediction

a) Take test features. Then use the rules of each randomly created decision tree to predict outcome and store the predicted outcome as the target
b) For each predicted target, calculate number of votes
c) Set final prediction as the predicted target with the highest number of votes

APPLICATION

To grow a Random Forest using the splitted training, testing, and OOT datasets, we loaded the randomForest package in R.

We used the following parameters in randomForest function:

- Set ntree = 1000 to specify model to grow 1000 trees
- Predicting fraud using 10 expert variables chosen through filter feature selection (listed previously) as the predictors
- data as training, testing, or OOT (depending on the case)

We ran the randomForest function three times for each of the separate datasets, training, testing, and OOT datasets, to implement the randomForest algorithm. For example, on the training dataset we ran the following:

rf.fit =randomForest (fraud ~ person_address_3 + person_address_7 + person_address_30 + person_homephone14 + homephone_distinct_address3 + homephone_distinct_address7 + homephone_distinct_address3 + homephone_distinct_zip3 + homephone_distinct_zip7 + person_ssn7, data = train, ntree = 1000)

We then implemented the prediction function using predict(). By running this function, all the trees make their classifications and vote on the outcome.

$$rf.probs = predict(rf.fit, test)$$

This function predicts the response variable, giving us the number of frauds caught for each record. We set this random forest probability value, the number of frauds caught, as the score for each record. By using this methodology, we were able to score all records and add these scores as a new field in the corresponding dataset. We then ordered the records by descending order of the scores.

To calculate the Fraud Detection Rate at different penetration rates for the top 20% population bins, we used the following methodology at all population windows, for 1%,...,20%:

1. We summed the random forest probabilities (the scores) across all records to calculate the total number of frauds caught by the Random Forest Model
2. We divided this value by the total number of frauds in the population, given to us in the dataset

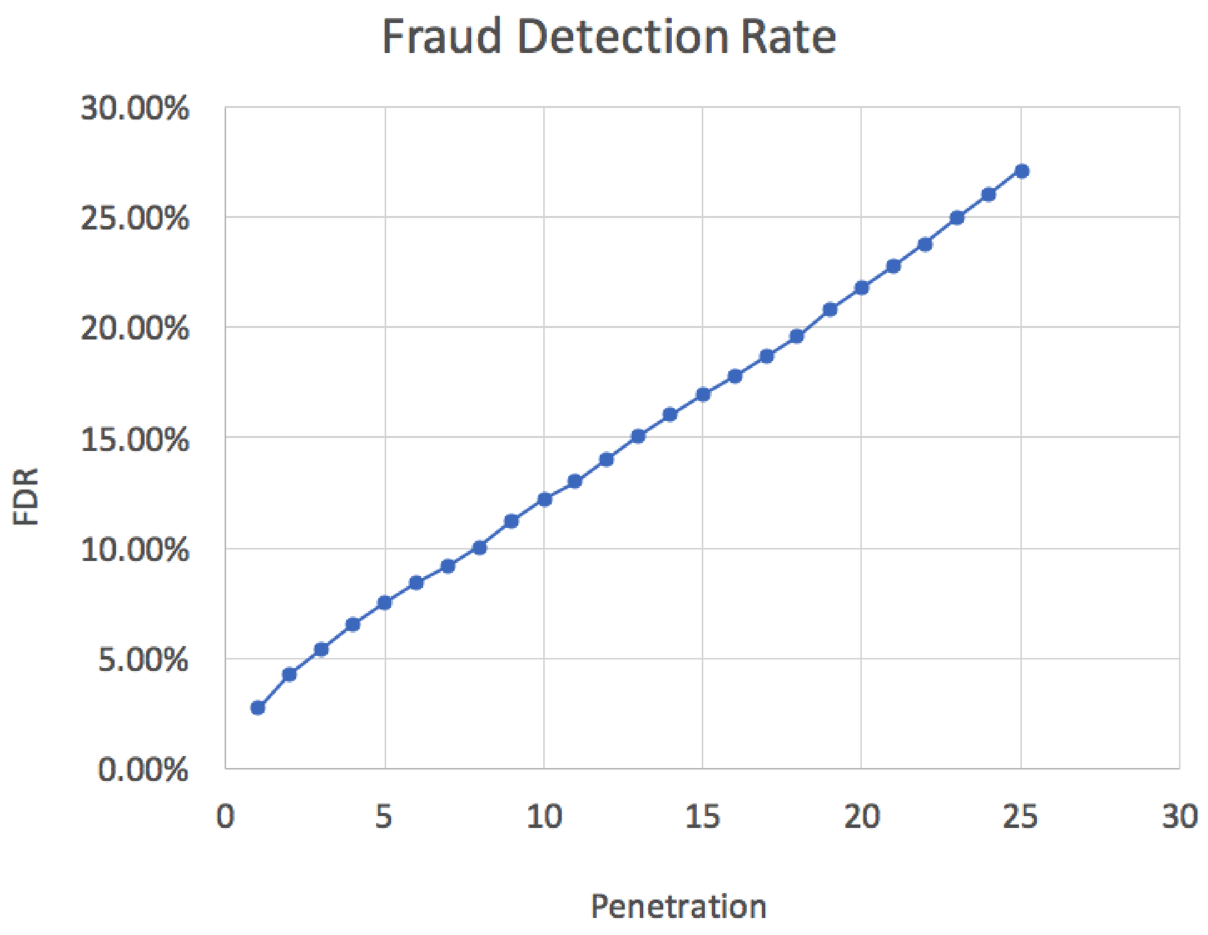
The following is the formula we used for all population bins:

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

For example, we calculated at 10% penetration rate, the FDR for the training, testing, and OOT datasets was 17.76%, 13.55%, and 12.34%, respectively.

Here is a graph plotting the fraud detection rates for the top 20% populations bins for the Random Forest model on the OOT dataset.

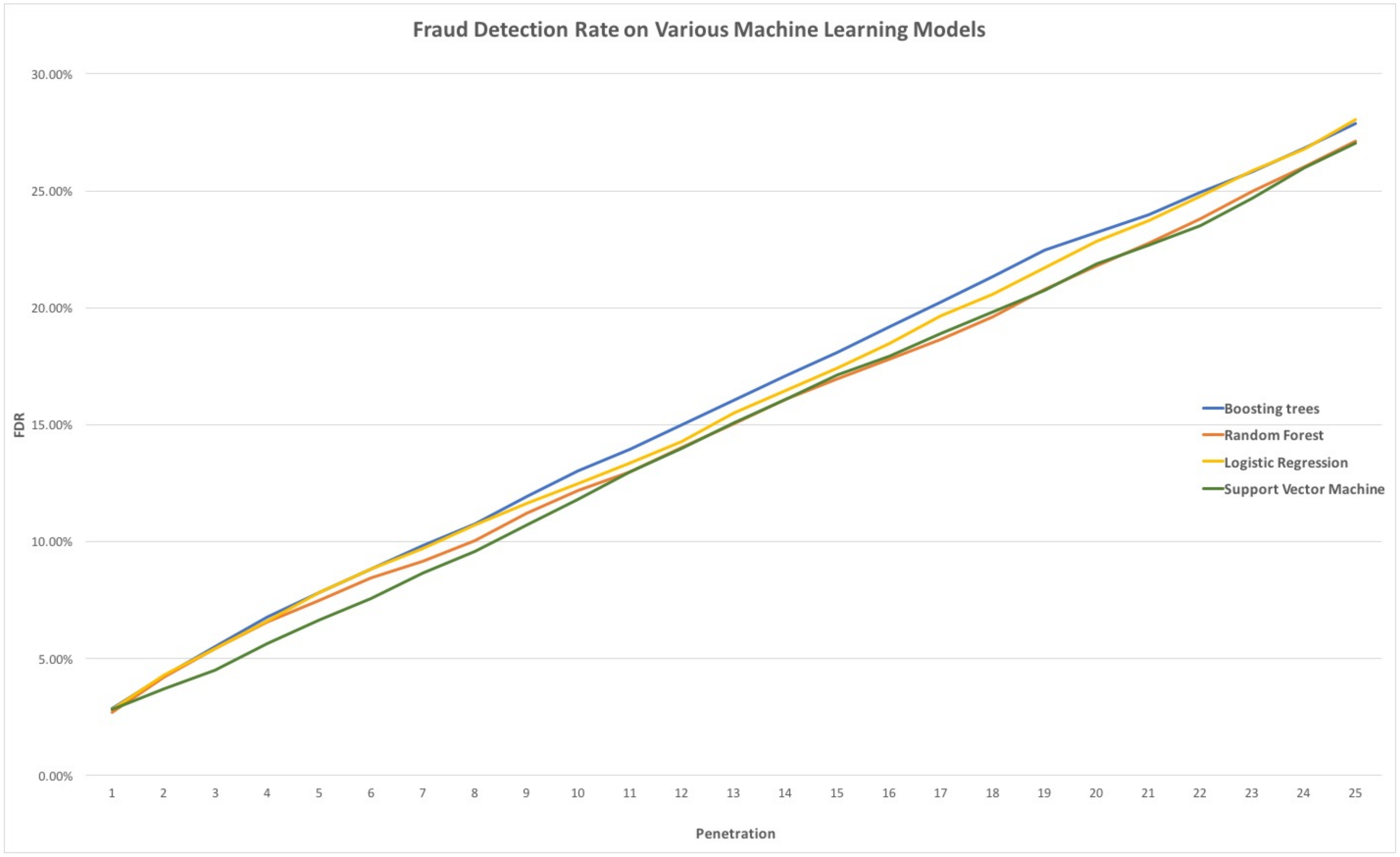
Graph 4. Fraud Detection Rate on the OOT Dataset for the Random Forest Model



DATA EVALUATION

RESULT

Graph 5. Fraud Detection Rate on Various Machine Learning Models to the OOT Applications dataset



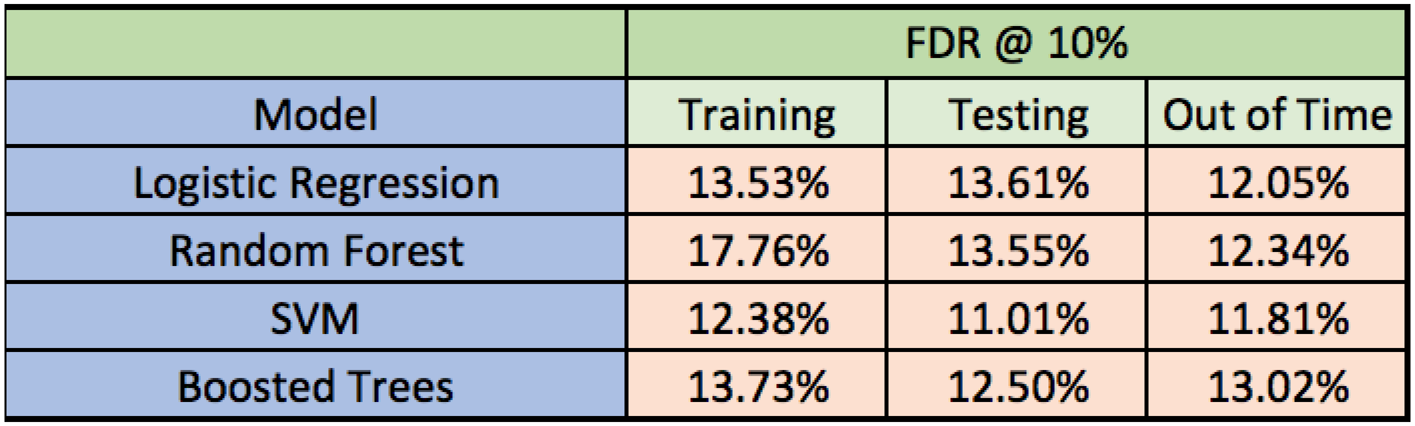
From Graph 5, the Gradient Boosting Trees machine learning method (Blue line) produced a higher Fraud Detection Rate (FDR) from population bin 8% to population bin 22%. After population bin 22%, the Logistic Regression method (Yellow line) produced a Fraud Detection Rate as high as the Gradient Boosting Trees. The Random Forest method (Orange line) outperformed the Support Vector Machine method (Green line) up to 10% penetration and beyond that population bin, the Random Forest method slowly fluctuated.

Overall at 10% penetration, the Support Vector Machine method (Green line) had the worst Fraud Detection rate and the Gradient Boosting Tree method (Blue line) had the best Fraud Detection rate on the OOT Applications dataset.

COMPARISON OF THE ALGORITHMIC METHODS

After performing 4 supervised machine learning models on each segment of the dataset, we created a report containing the Fraud Detection Rate metrics at 10% penetration rate. Below are the overall results of the report:

Table 9. Fraud Detection Rate at 10% Penetration for Models



	FDR @ 10%		
Model	Training	Testing	Out of Time
Logistic Regression	13.53%	13.61%	12.05%
Random Forest	17.76%	13.55%	12.34%
SVM	12.38%	11.01%	11.81%
Boosted Trees	13.73%	12.50%	13.02%

Based on the overall table above, we concluded that the Gradient Boosting Trees machine learning model had the best overall Fraud Detection Rate on the OOT (Out of Time) segment of the Product Applications Dataset, with FDR of 13.02%.

Below is our Table of top 20% bin population for the Gradient Boosting Trees Model on the OOT segment.

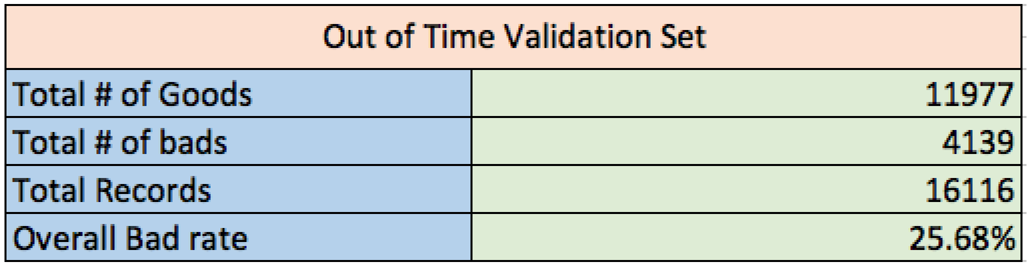
Table 10. Top 20% Bin Population for Gradient Boosting Trees Model on OOT

Overall Bad Rate is 25.68 %	Bin Statistics					Cumulative Statistics					
Population Bin %	Total # Records	# Good	# Bad	% Good	% Bad	Cumulative Good	Cumulative Bad	% Good	% Bad (FDR)	KS	False Positive Ratio
1	161	44	117	27.33%	72.67%	44	117	0.37%	2.85%	2.48%	0.38
2	161	103	58	63.98%	36.02%	147	175	1.23%	4.25%	3.02%	0.84
3	161	109	52	67.70%	32.30%	256	227	2.14%	5.50%	3.36%	1.13
4	161	109	52	67.70%	32.30%	365	279	3.05%	6.76%	3.71%	1.31
5	161	117	44	72.67%	27.33%	482	323	4.02%	7.82%	3.80%	1.49
6	161	120	41	74.53%	25.47%	602	364	5.03%	8.81%	3.78%	1.65
7	161	119	42	73.91%	26.09%	721	406	6.02%	9.83%	3.81%	1.78
8	161	122	39	75.78%	24.22%	843	445	7.04%	10.77%	3.73%	1.89
9	161	114	47	70.81%	29.19%	957	492	7.99%	11.91%	3.92%	1.95
10	161	115	46	71.43%	28.57%	1072	538	8.95%	13.02%	4.07%	1.99
11	161	122	39	75.78%	24.22%	1194	577	9.97%	13.96%	3.99%	2.07
12	161	118	43	73.29%	26.71%	1312	620	10.95%	15.00%	4.05%	2.12
13	161	117	44	72.67%	27.33%	1429	664	11.93%	16.06%	4.13%	2.15
14	161	118	43	73.29%	26.71%	1547	707	12.92%	17.10%	4.18%	2.19
15	161	120	41	74.53%	25.47%	1667	748	13.92%	18.09%	4.17%	2.23
16	161	116	45	72.05%	27.95%	1783	793	14.89%	19.18%	4.29%	2.25
17	161	118	43	73.29%	26.71%	1901	836	15.87%	20.22%	4.35%	2.27
18	161	114	47	70.81%	29.19%	2015	883	16.82%	21.35%	4.53%	2.28
19	161	115	46	71.43%	28.57%	2130	929	17.78%	22.46%	4.68%	2.29
20	161	129	32	80.12%	19.88%	2259	961	18.86%	23.24%	4.38%	2.35
21	161	130	31	80.75%	19.25%	2389	992	19.95%	23.99%	4.04%	2.41
22	161	121	40	75.16%	24.84%	2510	1032	20.96%	24.95%	3.99%	2.43
23	161	125	36	77.64%	22.36%	2635	1068	22.00%	25.82%	3.82%	2.47
24	161	120	41	74.53%	25.47%	2755	1109	23.00%	26.81%	3.81%	2.48
25	161	117	44	72.67%	27.33%	2872	1153	23.98%	27.88%	3.90%	2.49

From table 10, each population bin consisted of 161 records. There was a big increase of the FDR resulting from the increase of the total count of Good and the big decrease of the total count of Bad from population bin 1% to population bin 2%. Population bin 21% had the highest % Good whereas population bin 1% had the highest % Bad.
Relative to the overall incremental increases in the false positive ratio across all population bins, the false positive ratio dramatically increased from 0.38 to 0.84 from population bin 1% to population bin 2%. This indicates that the number of frauds caught incorrectly increased dramatically after adding another 1% window of the population. After adding another 1% population, for population bin 3% the false positive ratio increased by 0.29 from population bin 2%. All subsequent increases are relatively smaller in comparison.

Here is table summarizing our total number of records including the total count of Good and the total count of Bad

Table 11. Summary of OOT Validation Set



Out of Time Validation Set	
Total # of Goods	11977
Total # of bads	4139
Total Records	16116
Overall Bad rate	25.68%

CONCLUSION

Data Cleaning

We first cleaned the data by handling frivolous ssn and homephone values and converting date fields, dob and date, to a specific date format (YYYY-MM-DD). In preparation for expert variable creation, we created several special variables by consolidating fields to represent various entities, such as person. Our feature selection process consisted of selecting features from the dataset that we believed would have the strongest relationship to the likelihood that an application is fraudulent. We selected features that we believed were strongly related and constructed 113 special variables in total.

Feature Engineering

Our filter feature selection process consisted of two parts. We first filtered our collection of special variables down to the top 40 using the Kolmogorov-Smirnov statistic as a measure of goodness for all the variables. We ranked ordered them in descending order of the KS statistic and filtered out the top 40 special variables. We then performed Stepwise Backward Selection to select the variables that produced the model of top performance. After running all iterations of the Stepwise Backward Selection method, we filtered our top 40 special variables down to 10 total special variables that produced the best performing model. We kept these 10 special variables to run our four machine learning models, Logistic Regression, Random Forest, Support Vector Machine, and Gradient Boosting Trees Model.

Data Modeling

For each model, we computed the response value as the number of frauds caught for each record and set this as the score. We then ordered the records in descending order of score. By summing up the number of frauds caught by each model and dividing that by the total number of frauds for that population bin, we were able to compute the Fraud Detection Rate for population bins 1% through 25%. The various machine learning models we applied on the Training, Testing, and OOT datasets were able to detect fraudulent cases at varying rates, depending on the population penetration rate.

Data Evaluation

To perform further analysis and comparison, we plotted the FDR at the different population penetration rates. Finally, we focused our analysis on the OOT dataset and calculated both bin and cumulative statistics and found the top performing model in this category to be the Gradient Boosting Trees Model.

KEY TAKEAWAYS

1. Worst performance overall by Support Vector Machine Model. Across all datasets, SVM had the lowest FDR. We suspect this may be due to the high complexity of the model.
2. Best performance by Gradient Boosting Trees Model on OOT at 10% penetration (FDR of 13.02%)
3. Best performance by Random Forest Model on Training data at 10% penetration (FDR of 17.76%)
4. Best performance by Logistic Regression Model on Testing data at 10% penetration (FDR of 13.61%)
5. We saw outperformance by our ensemble machine learning models, Gradient Boosting Trees and Random Forest Models. Both models demonstrated relatively high predictive performance, with the highest predictive power in their respective segments; OOT for Gradient Boosting Trees Model and Training for Random Forest Model. Further, the Gradient Boosting Trees Model ranked second on the Training data with an FDR of 13.73%. Similarly, the Random Forest Model had the second highest FDR on OOT, with an FDR of 12.34%.
6. The overall bad rate on the OOT dataset is 25.68%

POTENTIAL IMPROVEMENT

Building fraud detection models on the Product Applications dataset consisted of multiple phases, including understanding, cleaning, modeling, and evaluating the Product Applications dataset.

Throughout the overall process, the following areas could have significantly improved our analysis on the Product Applications dataset:

1. **Access to More Data:**
   Data is critical for modeling and building good machine learning algorithms. The more data we have, the more effective our model is when it comes to fraud detection.
2. **Algorithm Tuning:**
   Machine learning algorithms are parameterized and modification of those parameters can influence the outcome of the learning process.
3. **Extreme Feature Engineering:**
   Performing better feature decomposition and aggregation in order to better normalize the data for machine learning algorithms. Extreme Feature Engineering applies only when the attribute decomposition and aggregation seen in data preparation is pushed to the limits.
4. **Performing More Supervised Learning Algorithm Methods:**
   Building more Fraud Detection models would have provided further insights on potentially better fraud detection rates. For further investigation we could have used methods such as:
    a. Discriminant Analysis (classification)
    b. Naive Bayes (classification)

APPENDIX

DATA QUALITY REPORT

DATA BASIC INFORMATION

DATASET DESCRIPTION

The file dataset is called “applications.csv.” Applications is a dataset containing the records of 984,866 product applications data along with 10 different fields. It contains the information about a unique id, the date the application was created, a social security number, a first name, a last name, an address, a zip code, a date of birth, a home phone number, and a Fraud indicator variable.

NUMBER OF RECORDS

984,866 records

NUMBER OF FIELDS

10 variables in total – 6 categorical variables, 1 text variable and 1 indicator variable or dummy variable.

VARIABLE INFORMATION

VARIABLE/FIELD NAME	DATA TYPE
record	Categorical variable
date	Date variable
ssn	Categorical variable
firstname	Categorical variable
lastname	Categorical variable
address	Text variable
zip5	Categorical variable
dob	Date variable
homephone	Categorical variable
fraud	Indicator variable / Dummy variable

DATA UNDERSTANDING

VARIABLE 1: record

Description

record is a categorical variable that is considered a primary key because it is the unique identifier of each records within the Applications dataset.

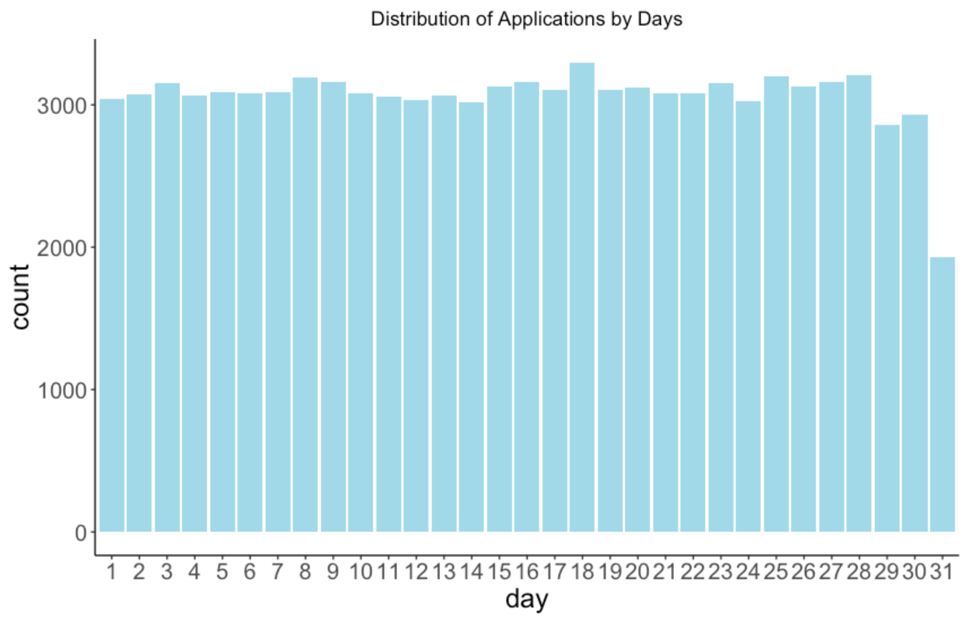
Record is 100% populated with 984,866 unique values and 0 missing values.

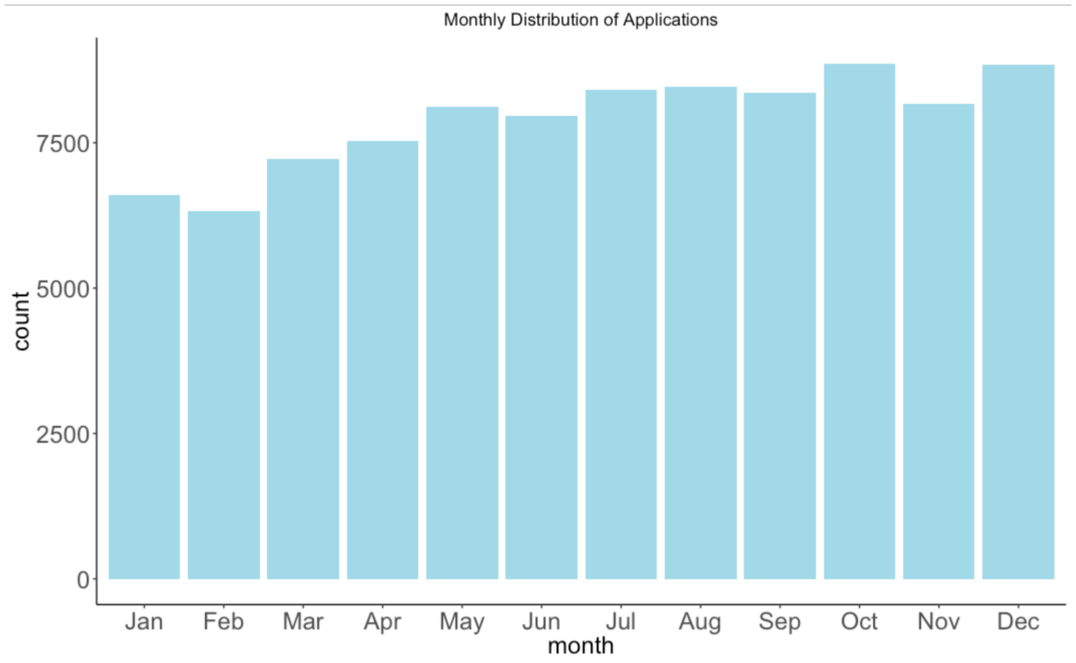
VARIABLE 2: date

Description

date is a date type variable showing the date in which the application was created. It goes in the format of "%m/%d/%y".

It is 100% populated with a total of 365 unique values. The total unique values actually show that each observation is populated by a day of the month and year of 2016.




Monthly Distribution of Applications
count
7500
5000
2500
0
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
month

VARIABLE 3: ssn

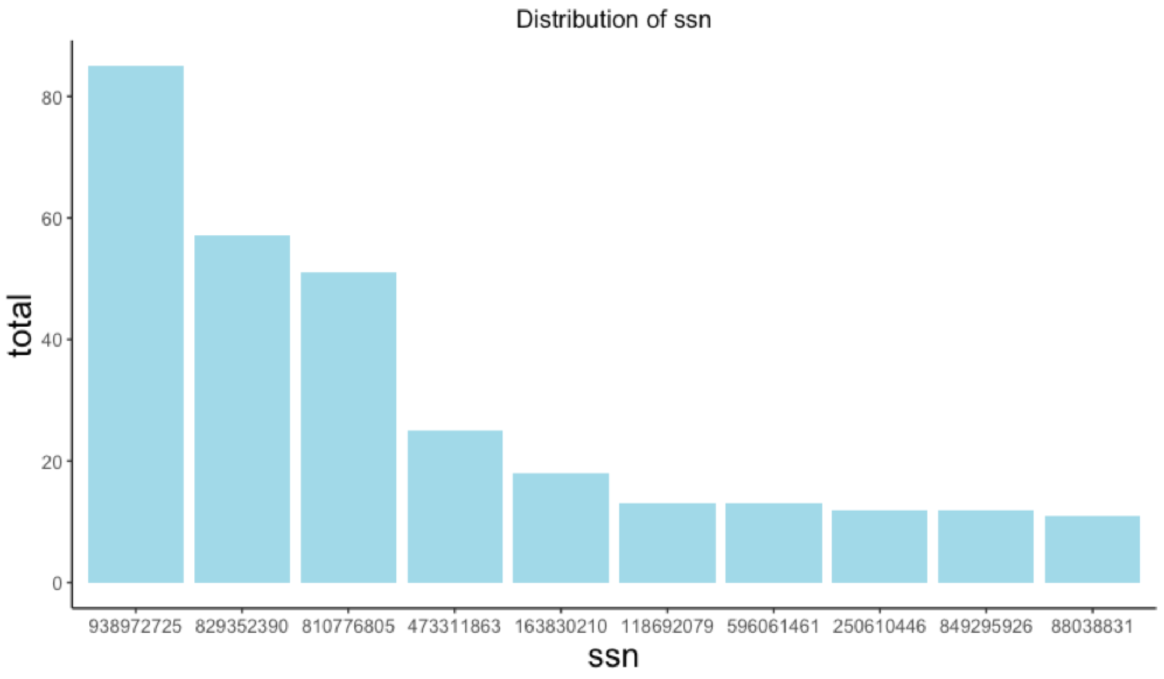
<u>Description</u>

ssn is a categorical variable with 9 digits total showing the social security number of the person that he/she reported on the application.

It is 100% populated with 86,771 unique values. The range of *ssn* goes from two-digit social security number to 9-digit social security number. Any social security number shorter than 9 digits means that they have some leadings 0 in the value, i.e. "99" is actually "000000099." Something worth noticing is the value "737610282", there is a count of 1478 total which could indicate a potential fraudulent record.

Rank (Top 10)	SSN	Total Count	Frequency Populated
1	737610282	1478	1.56%
2	938972725	85	0.09%
3	829352390	57	0.06%
4	810776805	51	0.05%
5	473311863	25	0.03%
6	163830210	18	0.02%
7	118692079	13	0.01%
8	596061461	13	0.01%
9	250610446	12	0.01%
10	849295926	12	0.01%

Here is a distribution of the Top 10 ssn excluding “737610282”.



VARIABLE 4: firstname

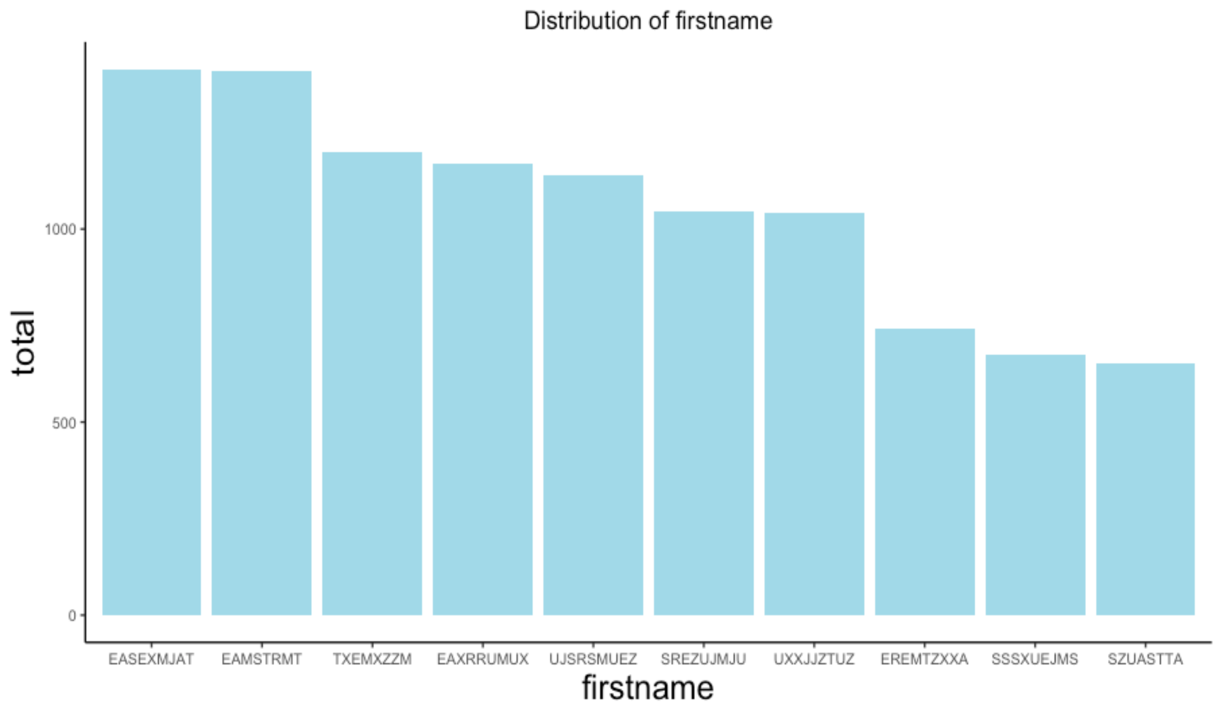
Description

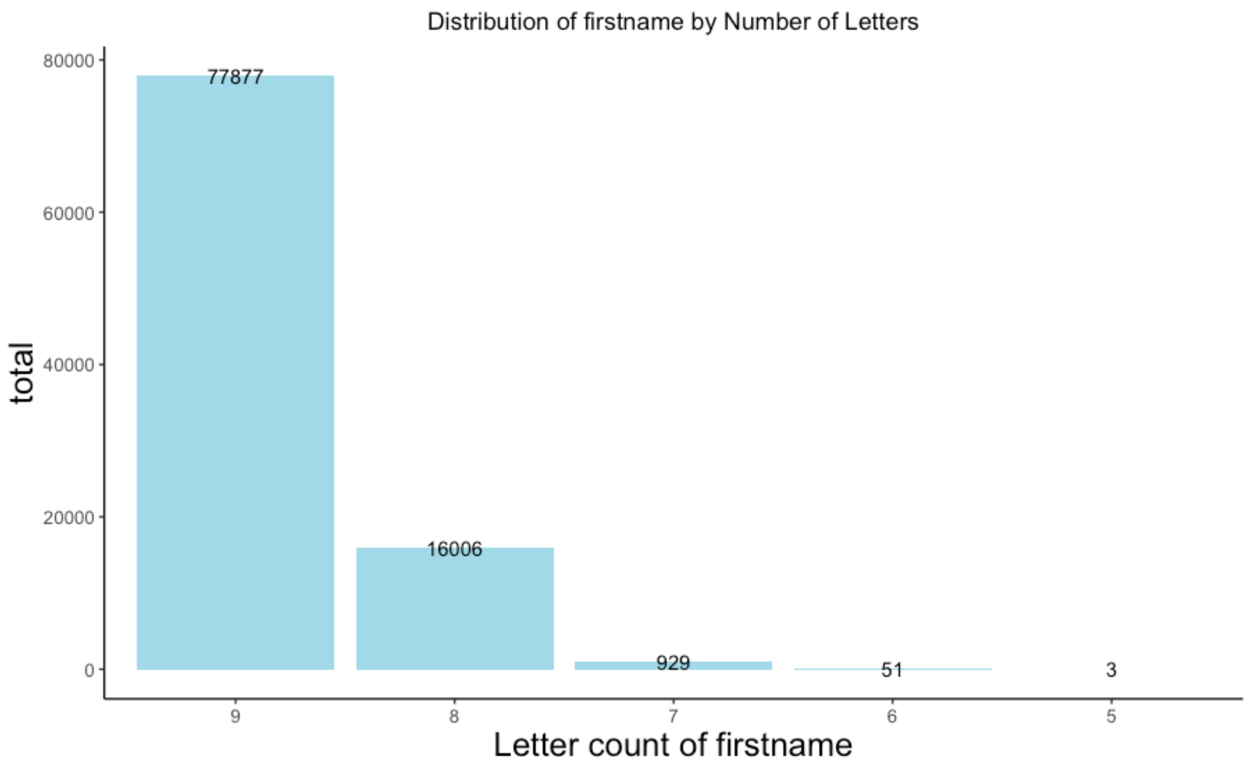
firstname is a categorical variable indicating the first name of the person that he/she reported on the application.

It is 100% populated with 14,626 unique values. They all seem to have alphabet letters as values within the *firstname* field. Here is a descriptive summary about the number of letters within the values of the field *firstname*. The minimum number of letters within the field *firstname* is 5 and the maximum number of letters within the field *firstname* is 9.

Min	Q1	Median	Mean	Q3	Max
5	9	9	8.81	9	9

Rank (Top 10)	firstname	Total Count	Frequency Populated
1	EASEXMJAT	1414	1.49%
2	EAMSTRMT	1411	1.49%
3	TXEMXZZM	1200	1.26%
4	EAXRRUMUX	1170	1.23%
5	UJSRSMUEZ	1138	1.20%
6	SREZUJMJU	1044	1.10%
7	UXXJJZTUZ	1042	1.10%
8	EREMTZXXA	742	0.78%
9	SSSXUEJMS	675	0.71%
10	SZUASTTA	653	0.69%


Distribution of firstname
total
1000
500
0
EASEXMJAT
EAMSTRMT
TXEMXZZM
EAXRRUMUX
UJSRSMUEZ
SREZUJMJU
UXXJJZTUZ
EREMTZXXA
SSSXUEJMS
SZUASTTA
firstname


Distribution of firstname by Number of Letters
total
80000
60000
40000
20000
0
77877
16006
929
51
3
9
8
7
6
5
Letter count of firstname

VARIABLE 5: lastname

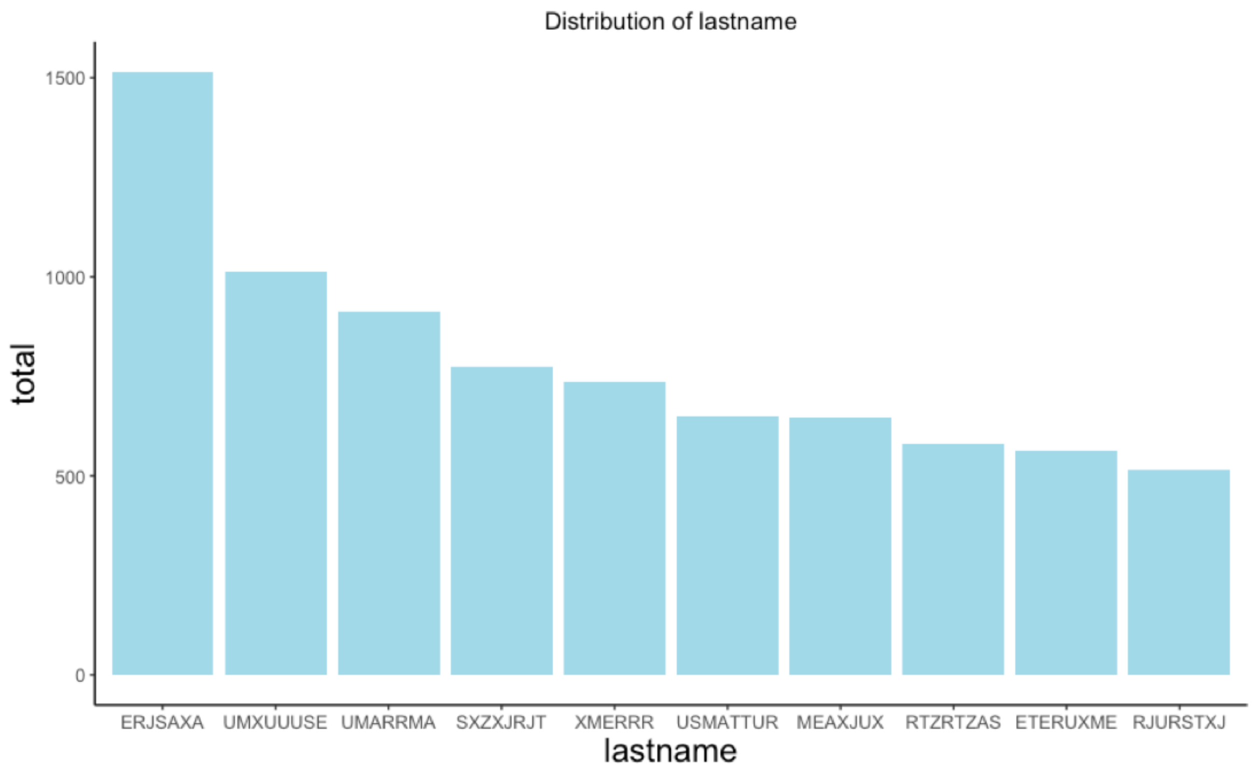
<u>Description</u>

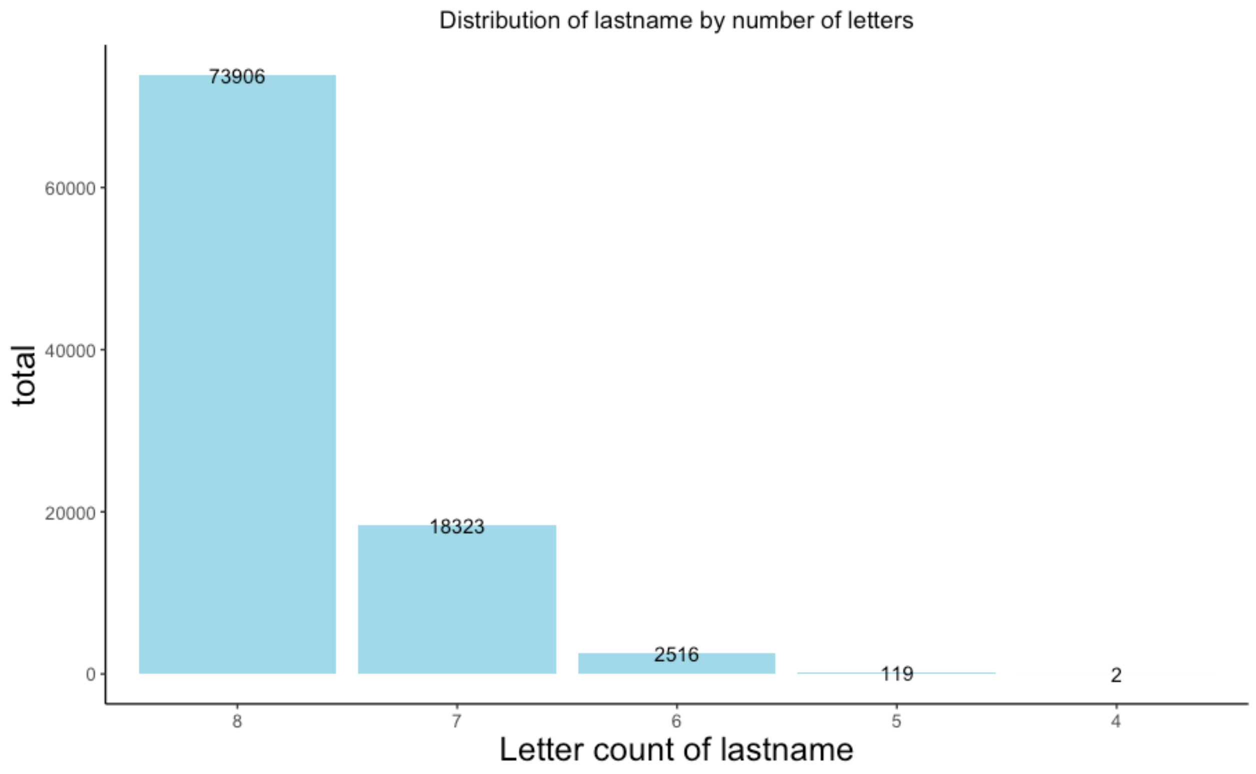
lastname is a categorical variable indicating the last name of the person that he/she reported on the application.

It is 100% populated with 31,513 unique values. Same characteristics as the field *firstname*, all values seem to contain only alphabet letters within each values of the field *lastname*. Here is a descriptive summary of the number of letters within each value of *lastname*. The minimum number of letters from values within the field *lastname* is 4 and the maximum number of letters from values within the field *lastname* is 8.

Min	Q1	Median	Mean	Q3	Max
4	8	8	7.75	8	8

Rank (Top 10)	lastname	Total Count	Frequency Populated
1	ERJSAXA	1515	1.60%
2	UMXUUUSE	1013	1.07%
3	UMARRMA	913	0.96%
4	SXZXJRJT	775	0.82%
5	XMERRR	737	0.78%
6	USMATTUR	649	0.68%
7	MEAXJUX	645	0.68%
8	RTZRTZAS	582	0.61%
9	ETERUXME	562	0.59%
10	RJURSTXJ	515	0.54%


Distribution of lastname
total
1500
1000
500
0
ERJSAXA
UMXUUUSE
UMARRMA
SXZXJRJT
XMERRR
USMATTUR
MEAXJUX
RTZRTZAS
ETERUXME
RJURSTXJ
lastname


Distribution of lastname by number of letters
total
Letter count of lastname
73906
18323
2516
119
2
8
7
6
5
4
0
20000
40000
60000

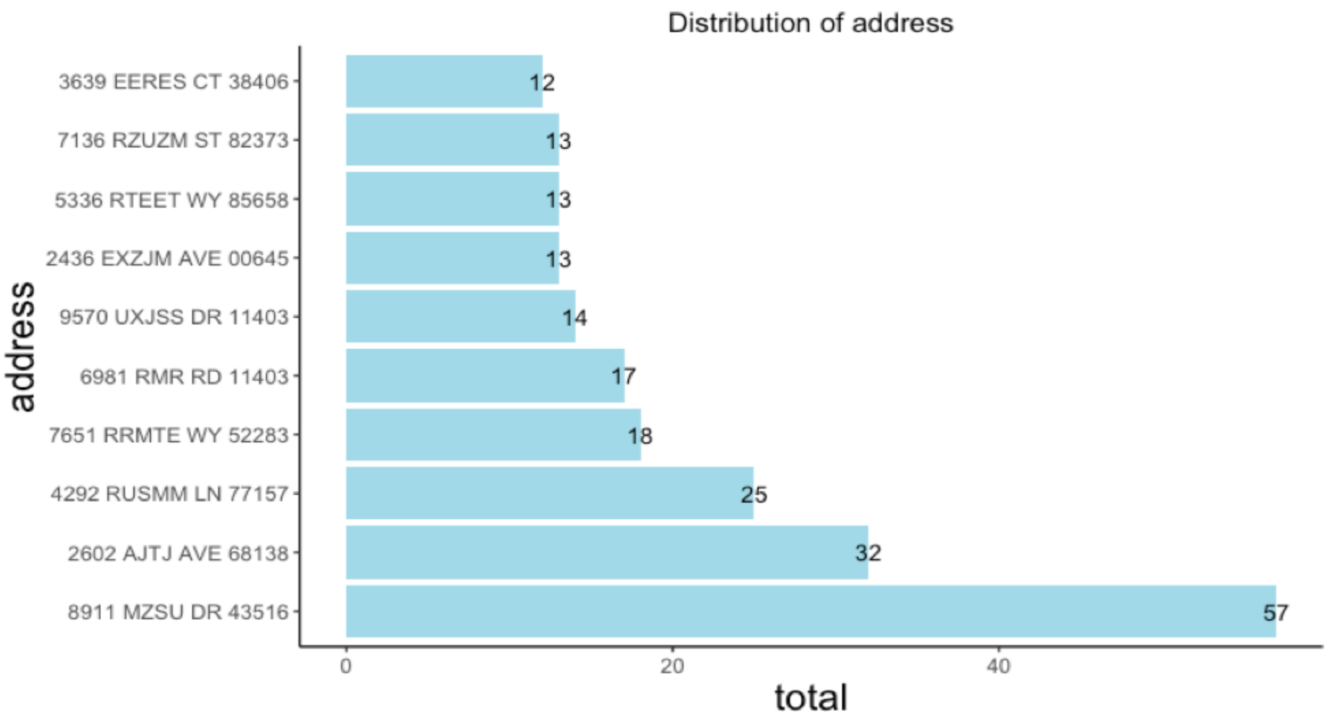
VARIABLE 6: address

Description

address is a text variable indicating the address of a person that he/she reported on the application.

It is 100% populated with 88,167 unique values totals. The most frequent address is “8911 MZSU DR 43516” which appeared 57 times inside the Applications dataset and it indicates 0.06% of all records.

Rank (Top 10)	address	Total Count	Frequency Populated
1	8911 MZSU DR 43516	57	0.06%
2	2602 AJTJ AVE 68138	32	0.03%
3	4292 RUSMM LN 77157	25	0.03%
4	7651 RRMTE WY 52283	18	0.02%
5	6981 RMR RD 11403	17	0.02%
6	9570 UXJSS DR 11403	14	0.01%
7	2436 EXZJM AVE 645	13	0.01%
8	5336 RTEET WY 85658	13	0.01%
9	7136 RZUZM ST 82373	13	0.01%
10	3639 EERES CT 38406	12	0.01%



Distribution of address
3639 EERES CT 38406
7136 RZUZM ST 82373
5336 RTEET WY 85658
2436 EXZJM AVE 00645
9570 UXJSS DR 11403
6981 RMR RD 11403
7651 RRMTE WY 52283
4292 RUSMM LN 77157
2602 AJTJ AVE 68138
8911 MZSU DR 43516
12
13
13
13
14
17
18
25
32
57
0
20
40
address
total

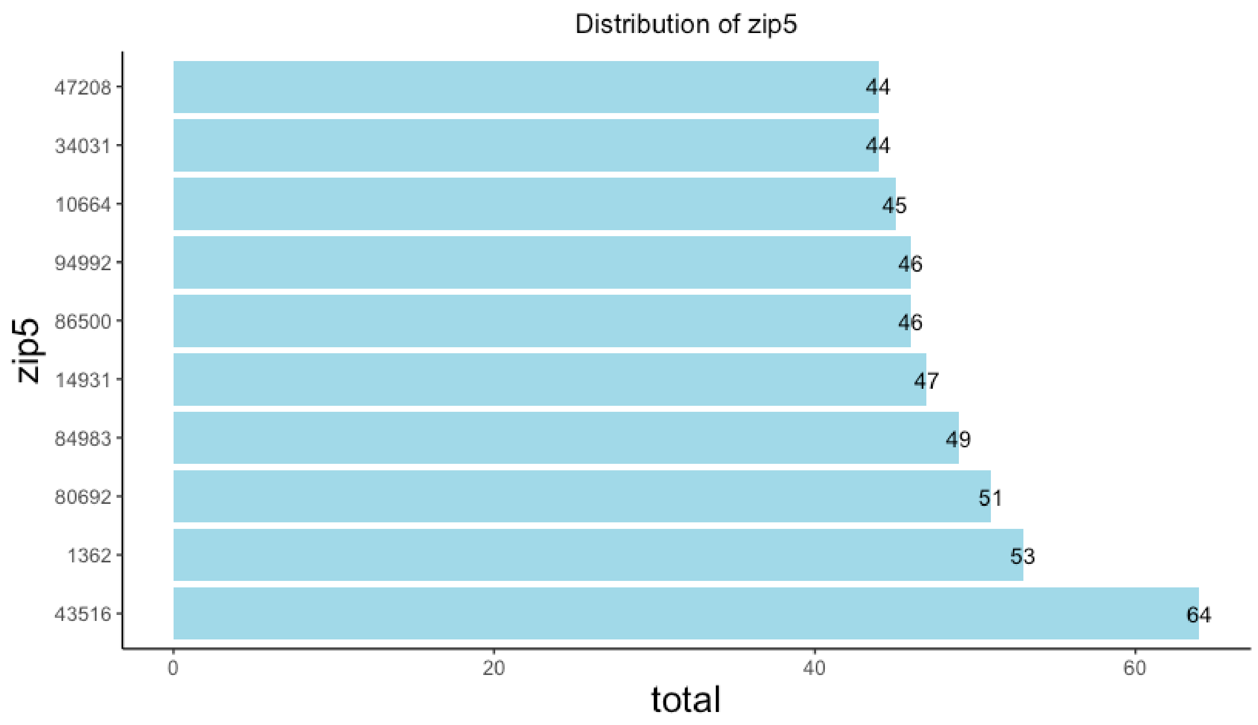
VARIABLE 7: zip5

Description

zip5 is a categorical variable indicating the zip code of an address that he/she reported on the application.

It is 100% populated and it contains 15,855 unique values. The most frequent zip5 is “43516” with a total count of 64 within the Applications dataset and it accounts for 0.07% of all records.

Rank (Top 10)	Zip5	Total Count	Frequency Populated
1	43516	64	0.07%
2	1362	53	0.06%
3	80692	51	0.05%
4	84983	49	0.05%
5	14931	47	0.05%
6	86500	46	0.05%
7	94992	46	0.05%
8	10664	45	0.05%
9	34031	44	0.05%
10	47208	44	0.05%


Distribution of zip5
zip5
47208
34031
10664
94992
86500
14931
84983
80692
1362
43516
44
44
45
46
46
47
49
51
53
64
0
20
40
60
total

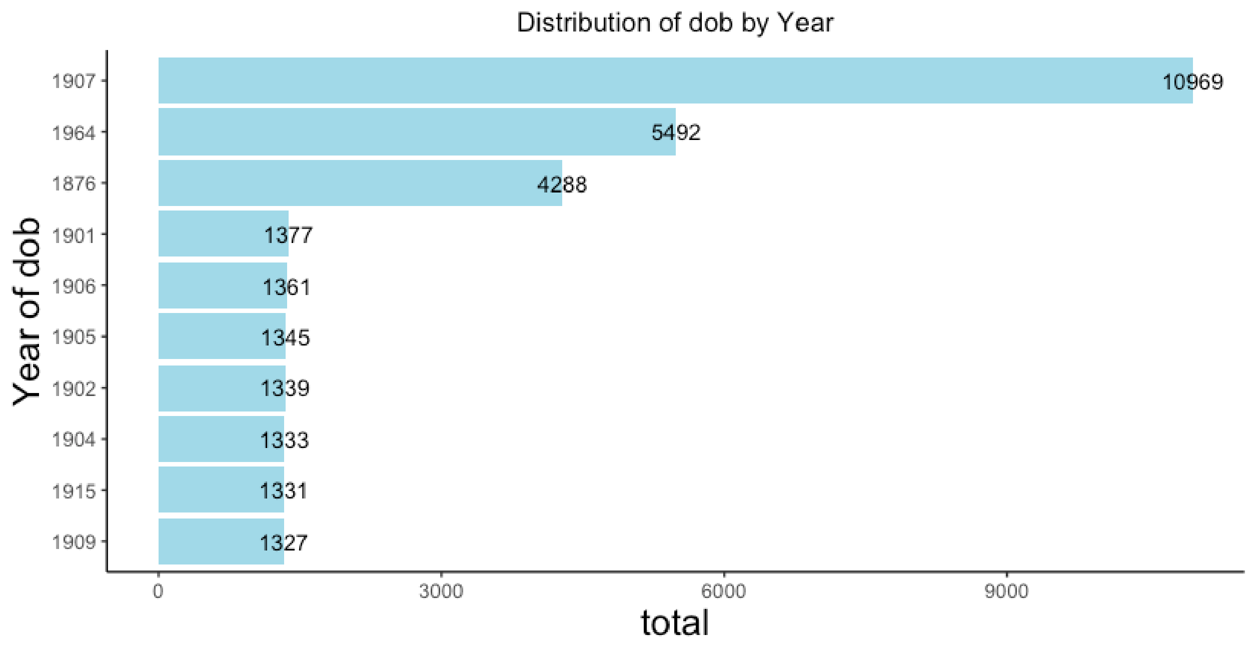
VARIABLE 8: dob

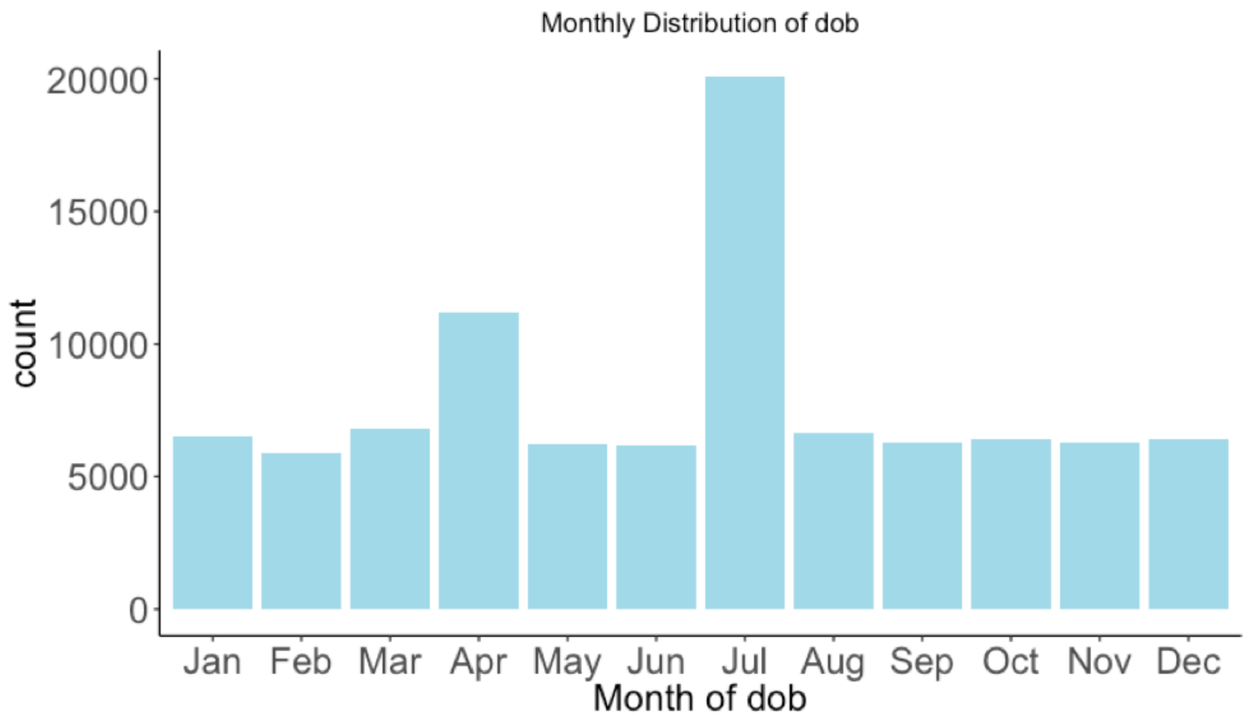
Description

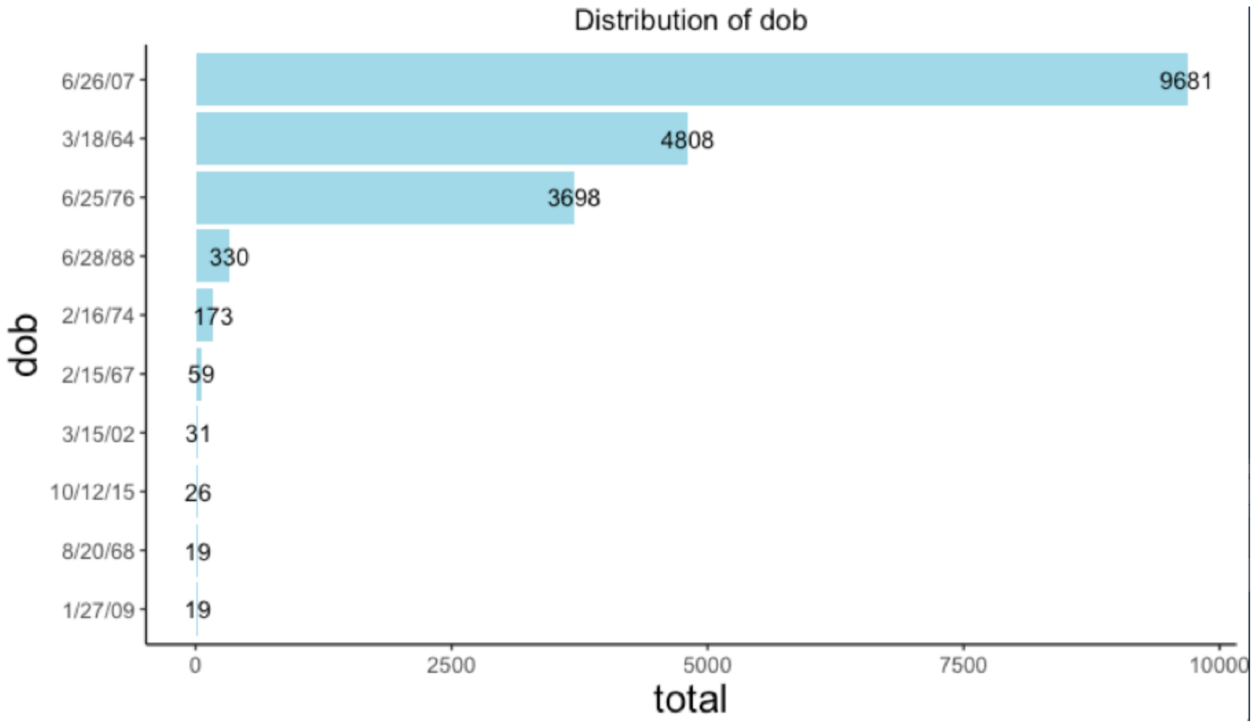
dob is a date variable indicating the date of birth of a person that he/she reported on the application.

It is 100% populated and it contains 30,599 unique values. The most frequent *dob* is "6/27/07", and June 26, 2007 is a Tuesday with a total count of 9681 and a populated frequency of 10.2% of all records.

Rank (Top 10)	dob	Total Count	Frequency Populated
1	6/26/07	9681	10.2%
2	3/18/64	4808	5.1%
3	6/25/76	3698	3.9%
4	6/28/88	330	0.3%
5	2/16/74	173	0.2%
6	2/15/67	59	0.1%
7	3/15/02	31	0.0%
8	10/12/15	26	0.0%
9	1/27/09	19	0.0%
10	8/20/68	19	0.0%


Distribution of dob by Year
Year of dob
1907
1964
1876
1901
1906
1905
1902
1904
1915
1909
10969
5492
4288
1377
1361
1345
1339
1333
1331
1327
0
3000
6000
9000
total


Monthly Distribution of dob
count
20000
15000
10000
5000
0
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month of dob


Distribution of dob
dob
total
6/26/07
3/18/64
6/25/76
6/28/88
2/16/74
2/15/67
3/15/02
10/12/15
8/20/68
1/27/09
9681
4808
3698
330
173
59
31
26
19
19
0
2500
5000
7500
10000

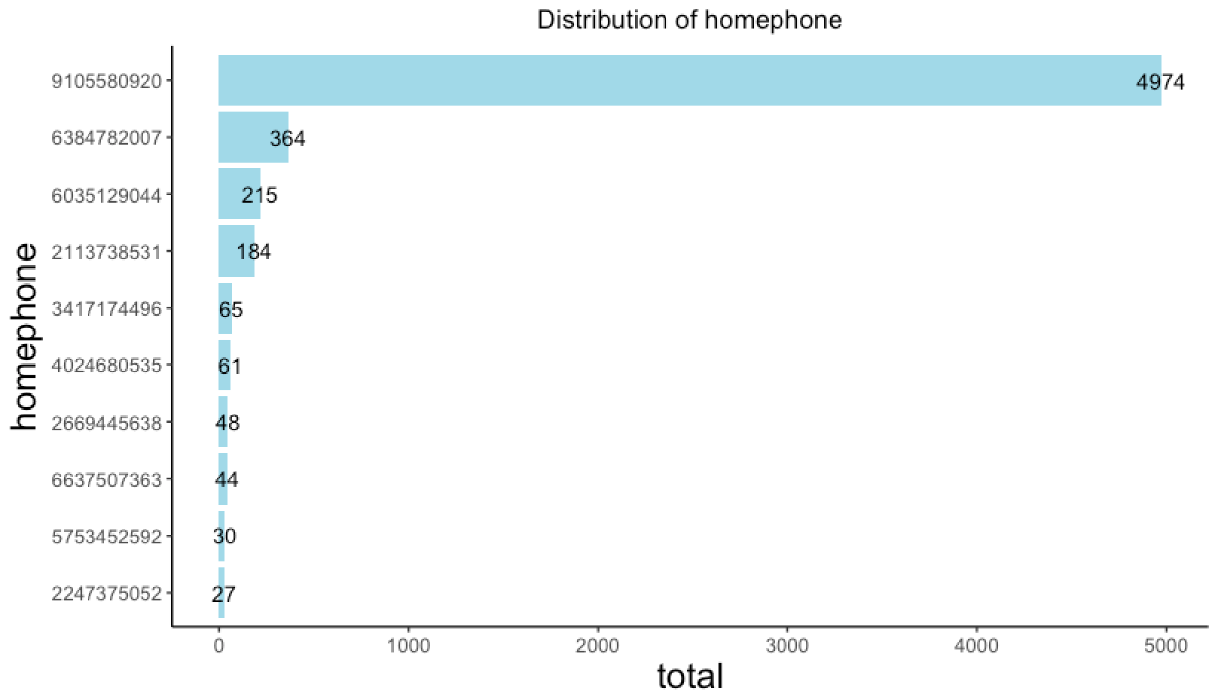
VARIABLE 9: homephone

Description

homephone is a categorical variable indicating the phone number of a person that he/she reported on the application.

It is 100% populated and it contains 20,762 unique values. The most frequent homephone is “9105580920” with a total count of 4974 and it accounts for 5.24% of all records. This could indicate a high fraudulent record because of the high frequency it populates.

Rank (Top 10)	homephone	Total Count	Frequency Populated
1	9105580920	4974	5.24%
2	6384782007	364	0.38%
3	6035129044	215	0.23%
4	2113738531	184	0.19%
5	3417174496	65	0.07%
6	4024680535	61	0.06%
7	2669445638	48	0.05%
8	6637507363	44	0.05%
9	5753452592	30	0.03%
10	2247375052	27	0.03%

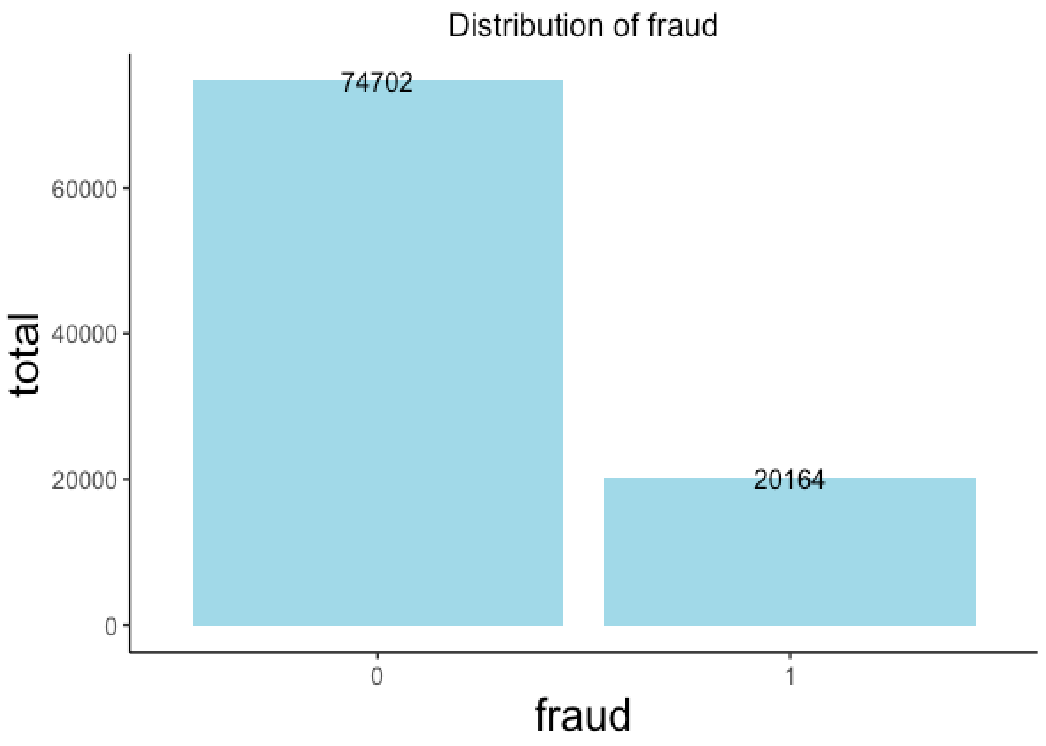

Distribution of homephone
homephone
total
9105580920
6384782007
6035129044
2113738531
3417174496
4024680535
2669445638
6637507363
5753452592
2247375052
4974
364
215
184
65
61
48
44
30
27
0
1000
2000
3000
4000
5000

VARIABLE 10: fraud

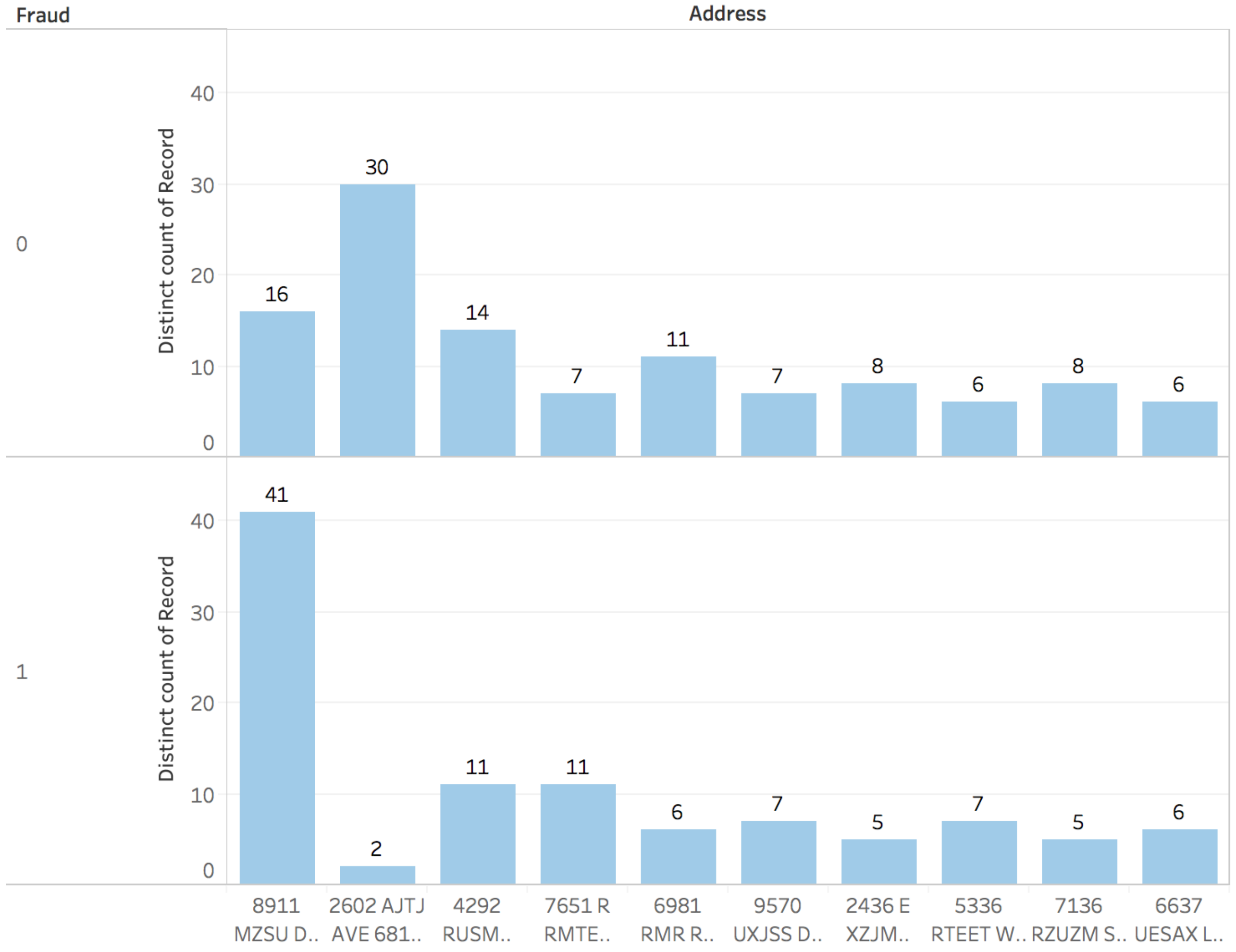
Description

fraud is an indicator variable or dummy that represent a subgroup called fraud. As implied by the name, fraud is a subgroup attribute, and it is used with two levels: 1 as being “fraudulent” and 0 as being “not fraudulent.”

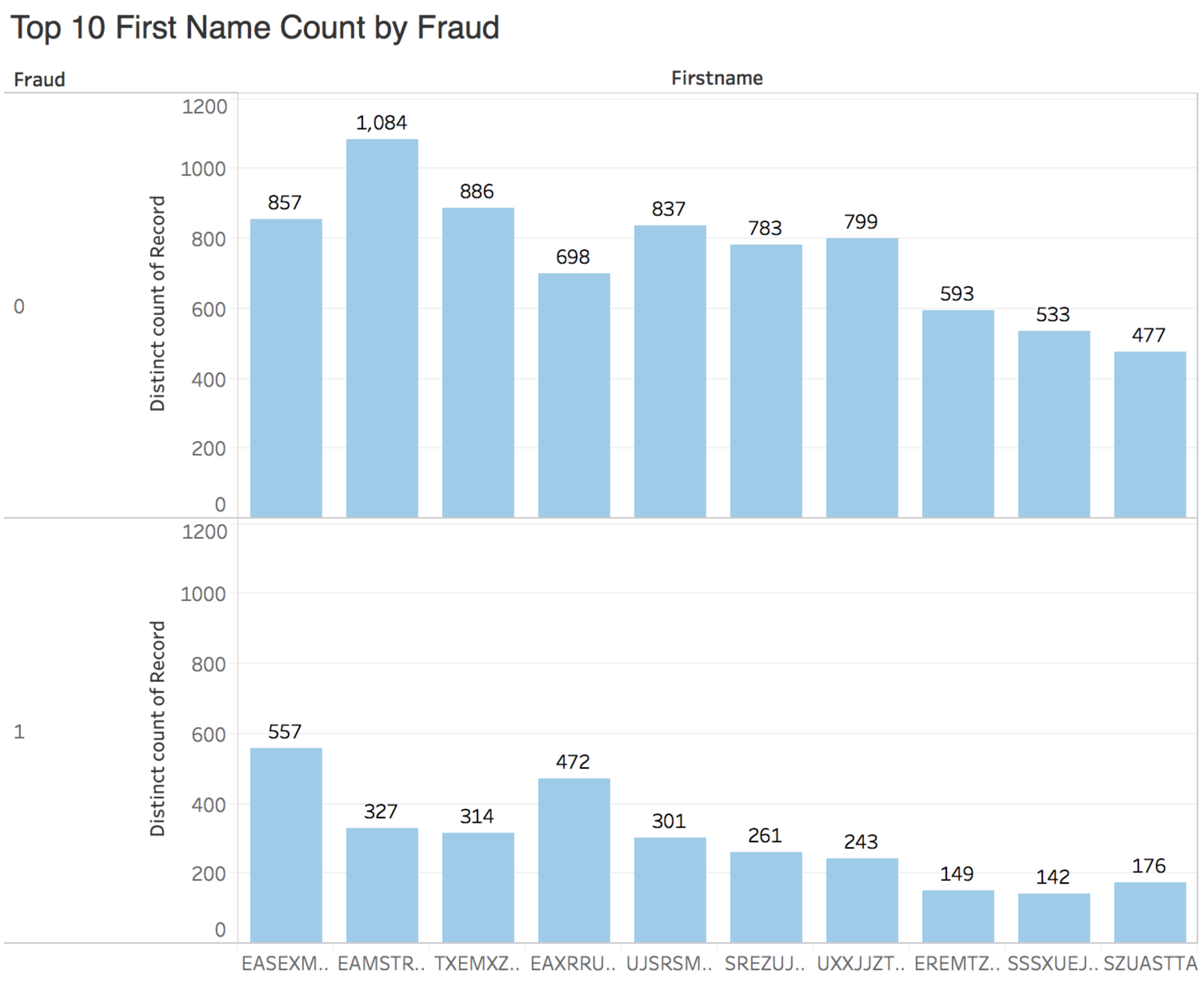
It is 100% populated and as a dummy variable, it contains only 2 unique values. 0 and 1. The most frequent values of the dummy variable fraud is “0” with a total count of 74,702.



Top 10 Address Count by Fraud

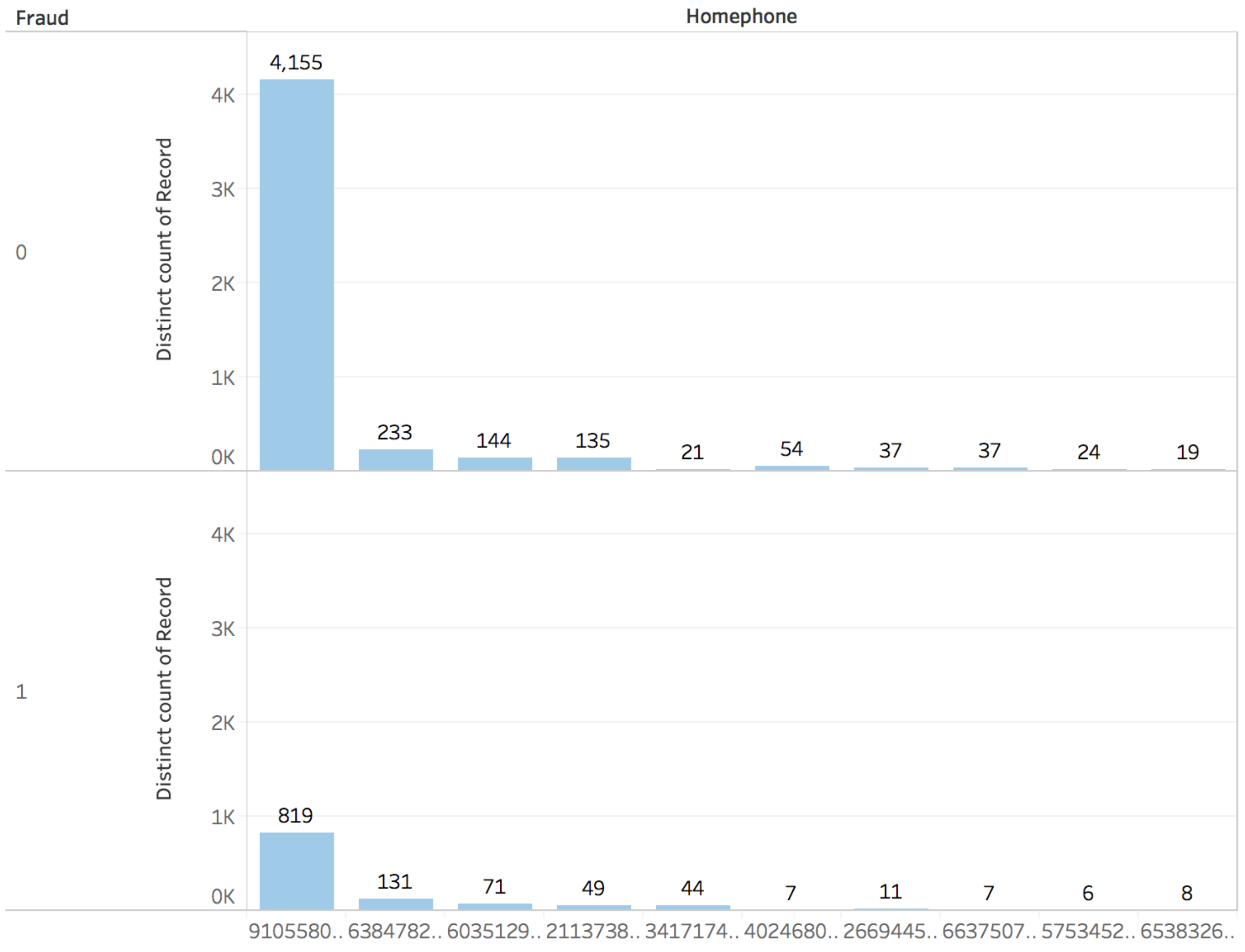


Distinct count of Record for each Address broken down by Fraud. The view is filtered on Address, which keeps 10 of 88,167 members.

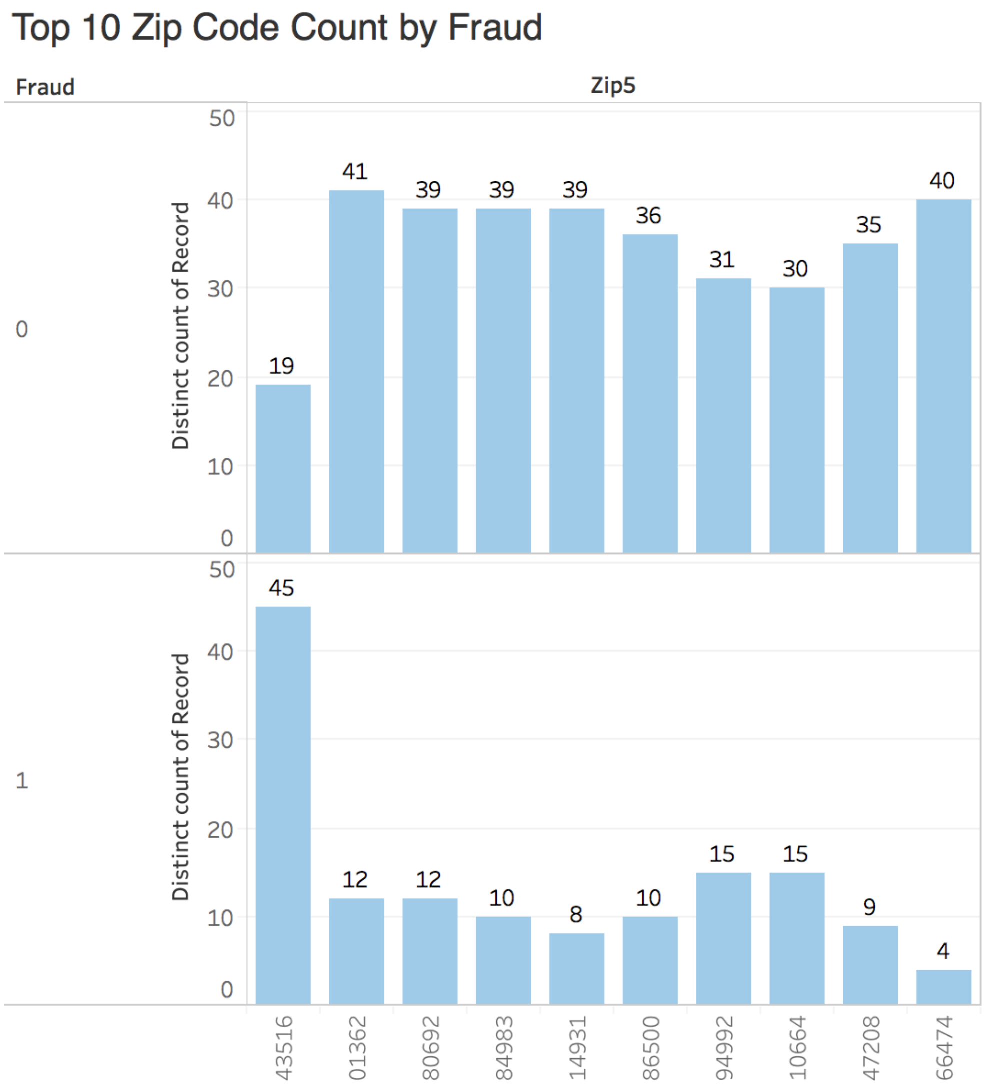


Distinct count of Record for each Firstname broken down by Fraud. The view is filtered on Firstname, which has multiple members selected.

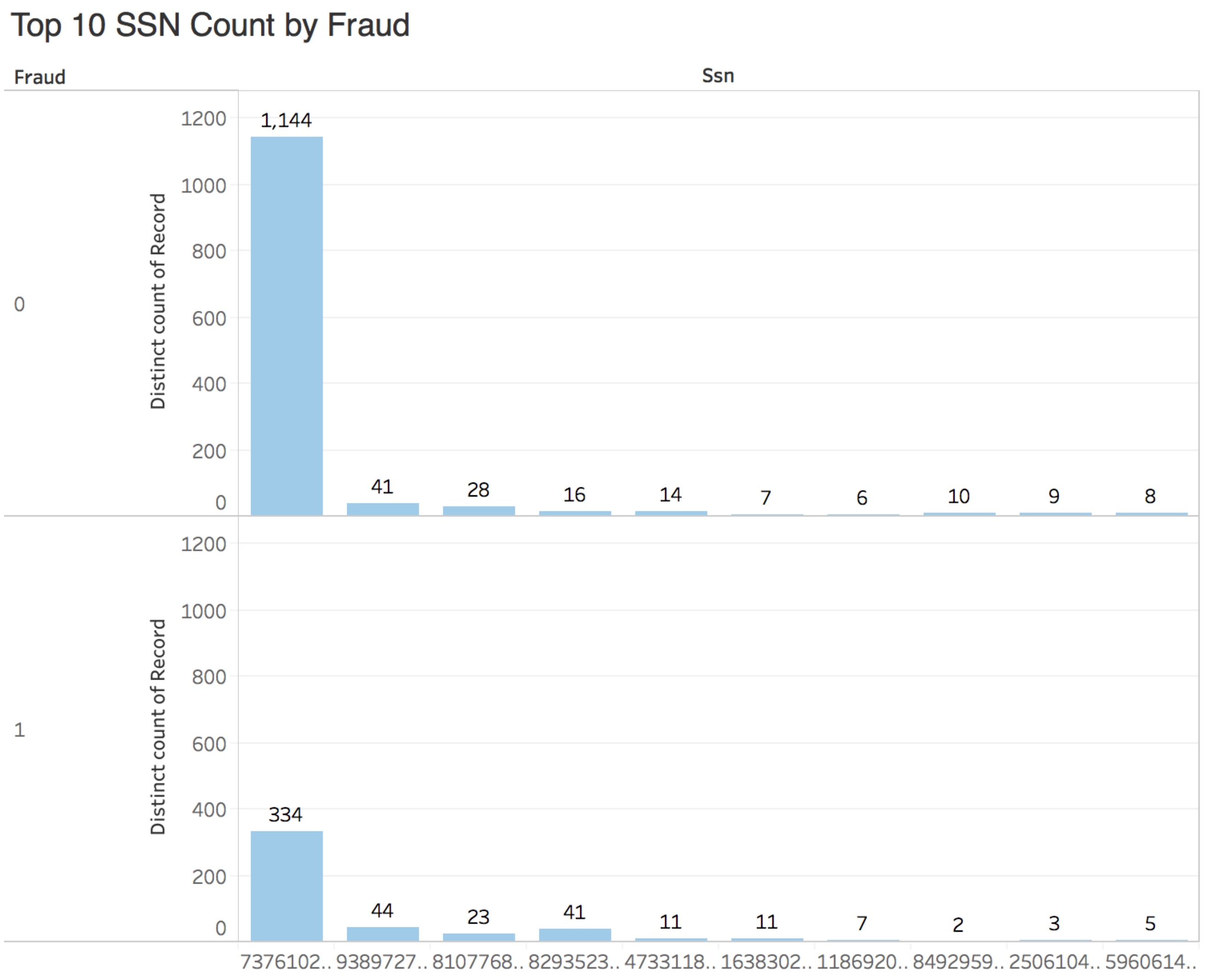
Top 10 Home Phone Number Count by Fraud



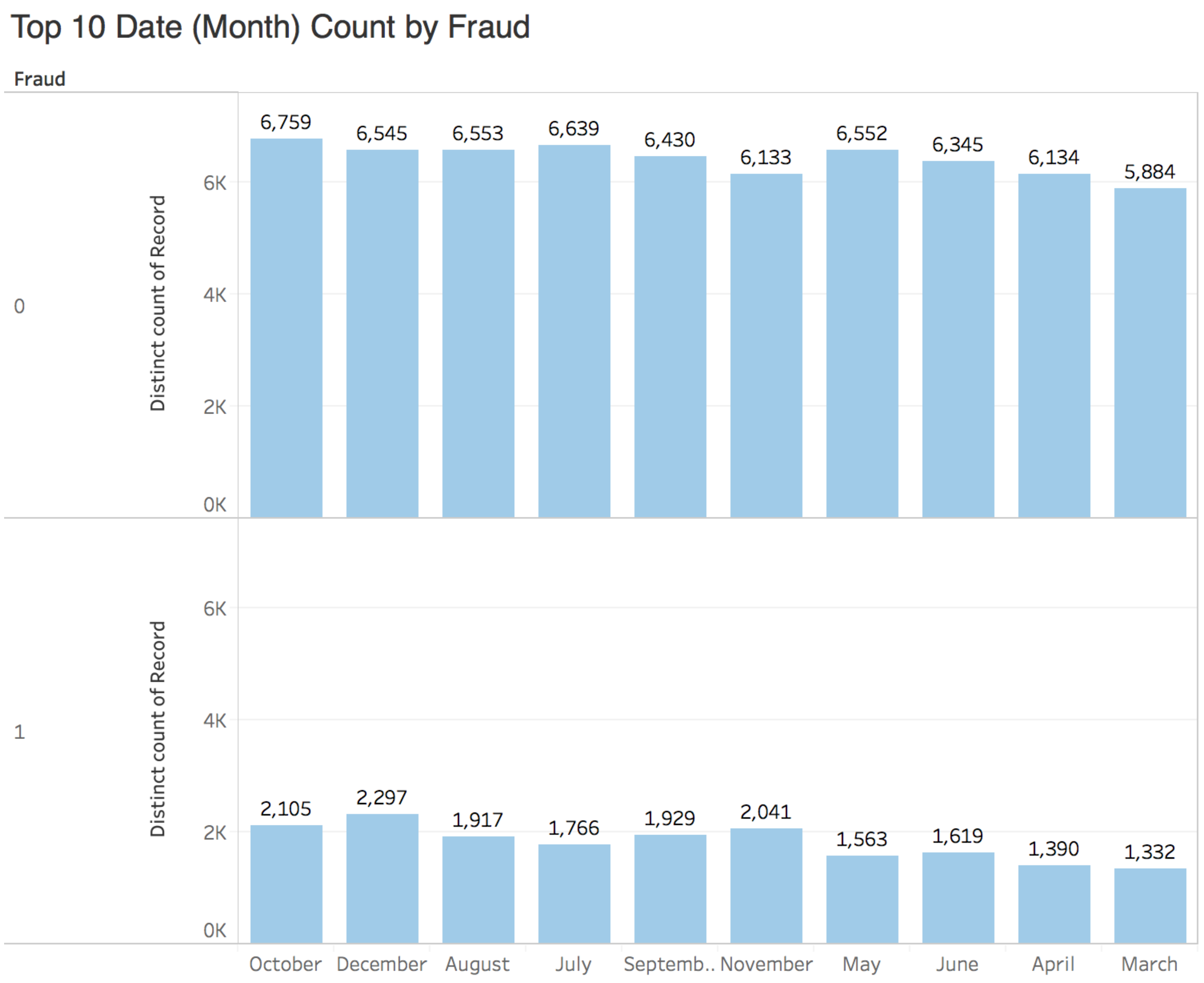
Distinct count of Record for each Homephone broken down by Fraud. The view is filtered on Homephone, which keeps 10 of 20,762 members.



Distinct count of Record for each Zip5 broken down by Fraud. The view is filtered on Zip5, which has multiple members selected.



Distinct count of Record for each Ssn broken down by Fraud. The view is filtered on Ssn, which has multiple members selected.



Distinct count of Record for each Date Month broken down by Fraud. The view is filtered on Date Month, which excludes January and February.