DSO 562 Project Two

Identity Fraud in the Credit Card Dataset

Team 5:
Ruoxian Jia,
Keying Que,
Laura Qin,
Tianqi He,
Xu Duo

Table of Contents

Executive Summary

Nowadays, many people use credit cards in their everyday lives. However, due to exploitable loopholes in the credit card application process, many banks have become victims of fraudulent credit card application caused by identity fraud. In this project, our goal is to develop a fraud detection model that predicts if a credit card applicant uses falsified identity.

Our raw data consists of records of credit card applications in 2016. There are approximately 100,000 rows that contain name, birthday, phone number, address, SSN, application date, and whether the application is fraudulent.

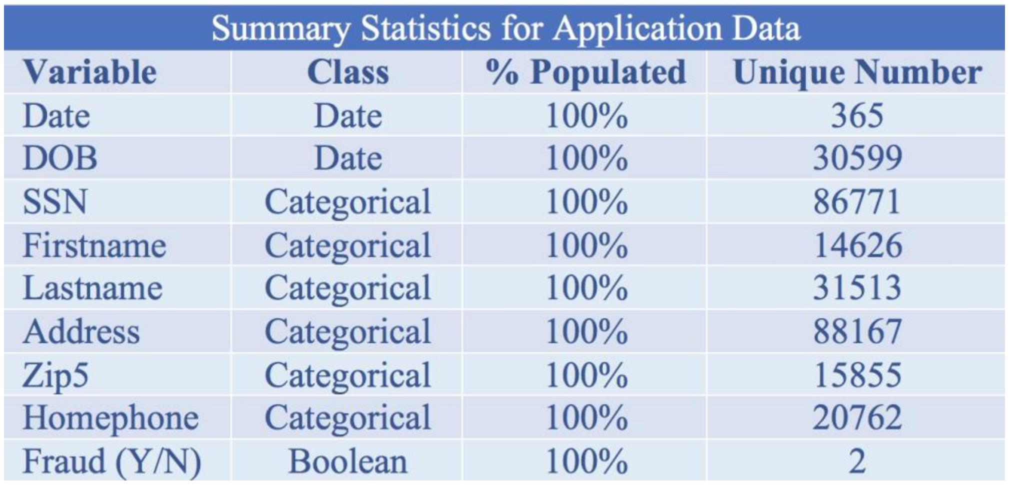
Initially, we created 80 expert variables using different time windows (1 day, 3 days, 6 days, 10 days, and 30 days) to see whether any particular field appears more than once in the time window.

Then we used Kolmogorov–Smirnov (KS) values to find the most significant variables and utilized backward selection to reduce the dimensions to 19 variables. After separating the data into training, testing, and out of time set, we built five different models and evaluated their performances by calculating the fraud detection rate in training, testing, and out-of-time data at 10% penetration. We found that the neural network model performs the best, and that its fraud detection rate at 10% is 11.78%.

Part I: Data Description

File Name: Applications
File description: The dataset contains the credit card application information of each customer
Number of Records: 94,866
Variables: 9 in total – 6 categorical variables, 2 date variables, 1 boolean variable

Summary statistics for numerical variables:



Summary Statistics for Application Data

Variable	Class	% Populated	Unique Number
Date	Date	100%	365
DOB	Date	100%	30599
SSN	Categorical	100%	86771
Firstname	Categorical	100%	14626
Lastname	Categorical	100%	31513
Address	Categorical	100%	88167
Zip5	Categorical	100%	15855
Homephone	Categorical	100%	20762
Fraud (Y/N)	Boolean	100%	2

Date: The date when the credit card application was filed. It ranges from 2016/01/01 to 2016/12/31

SSN: The SSN used for credit card application.

Firstname: Credit card applicant's first name

Lastname: Credit card applicant's last name

Address: Credit card applicant's address, including street number, street name and zip code

ZIP5: Zip code associated with the applicant's address

DOB: Applicant's date of birth. Format is "MM/DD/YYYY"

Homephone: Credit card applicant's home phone number

Fraud: It denotes if a credit card application is fraudulent. 1 for fraud, 0 for not fraud.

Part II: Variable Creation

By counting the occurences of different fields in 5 different time windows (1 day, 3 days, 6 days, 10 days and 30 days), we built our expert variables. Below is a table that shows the names of our expert variables and how we built them. Note that SSN “737610282” and phone number “6384782007” appear too frequently in the dataset, and are thus not included in all the counts related to SSN and phone number. Meanwhile, since our longest time window is 30 days, we filtered out all expert variable records from day 1 to day 30 to exclude any NA.

	“#” stands for the days of time window	
Category	**Variable Name**	**Description**
A	a_SSN_# a_phone_# a_address_#	How many same SSN, phone number, and address appeared in the past
B	b_SSN_# b_phone_# b_address_#	How many same SSN, phone number, and address were fraudulent in the past
C	c_name_# c_phone_# c_dob_#	How many First name & Last name combinations, birthdays, and home phones existed for one SSN
D	d_name_# d_snn_# d_dob_#	How many First name & Last name combinations, SSN, and birthdays existed for each home phone
E1	e_name_dob_#	How many different First name, Last name and birthday combinations existed
E2	e_homephone_# e_snn_#	How many different home phones, and SSNs existed for each First name, Last name and Birthday combination
F	f_homephone_#	How many different home phones existed for each First name, Last name and SSN combination

Part III: Feature Selection

During the feature selection stage, we calculated the Kolmogorov–Smirnov (KS) value for each variable and used backward stepwise methods to select 19 variables for our final models. The reason of conducting feature selection is to 1) make sure variables used for model building are the most critical ones to detect fraud; 2) reduce dimensions to avoid linearity and output a model that is closer to the actual model.

Univariate Feature Selection: Kolmogorov–Smirnov (KS)
KS value measures the maximum distances between two distributions. The larger the KS, the more separate the two distributions. In the context of detecting fraud, we used KS to measure the differences between fraud records and non-fraud records for each variable created. Specifically, for each variable, we gathered a list of fields corresponding to fraud records and the other list of fields relevant to non-fraud records. Then we applied *stats.ks.2samp* function under the *scipy* package in Python to compute KS and level of significance for all variables. Top 50% of the variables with the lowest level of significance and the highest KS values were selected. The filter feature selection method using KS left us 40 expert variables for the wrapped method.

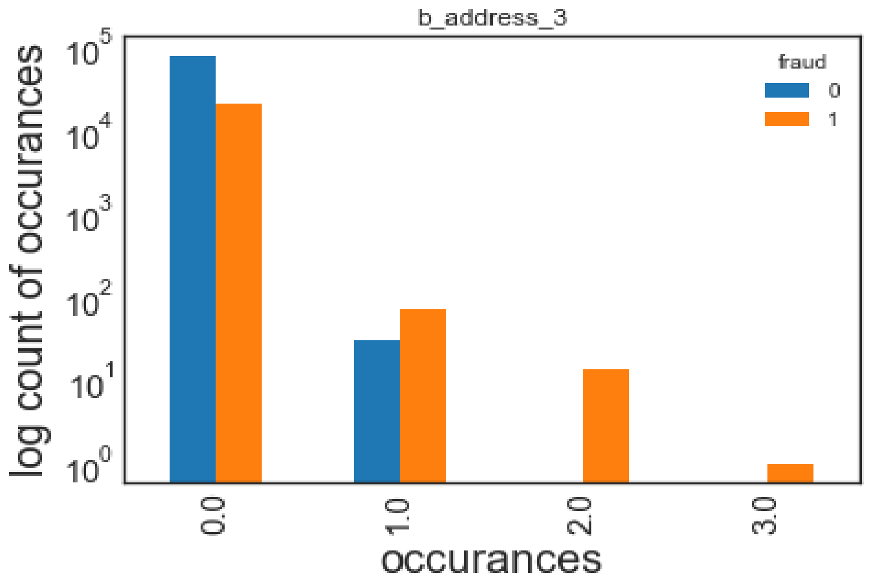
Wrapped Method: Backward Stepwise
Backward stepwise selection involves starting with all candidate variables, testing the deletion of each variable using a chosen model fit criterion, deleting the variable whose loss gives the most statistically insignificant deterioration of the model fit, and repeating this process until no further variables can be eliminated without a statistically significant loss of fit. In this project, we started with 40 variables left after filter feature selection and used the *step* function in R by setting the method argument equal to 'backward' to conduct backward stepwise selection on a logistic regression model.

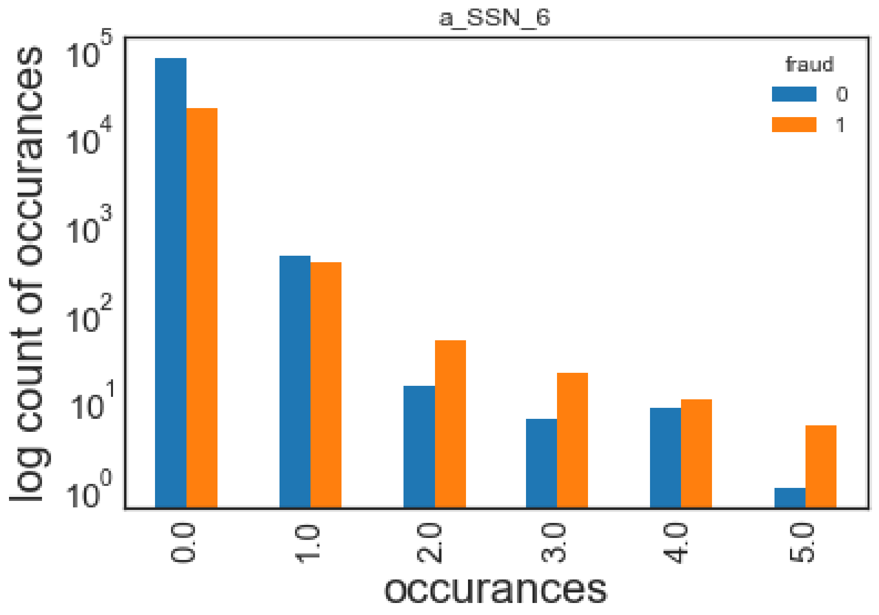
Below is a list of the 19 variables chosen by backward selection and are ones we used in our final models:

Variable Name	Description	Variable Name	Description
a_SSN_3	Number of same SSN seen in the past 3 days	**c_phone_30**	Number of different home phones seen in the past 30 days for a particular SSN

a_SSN_6	Number of same SSNs seen in the past 6 days	**c_dob_3**	Number of different dates of birth seen in the past 3 days for a particular SSN
a_SSN_10	Number of same SSNs seen in the past 10 days	**c_dob_6**	Number of different dates of birth seen in the past 6 days for a particular SSN
a_SSN_30	Number of same SSNs seen in the past 30 days	**c_dob_30**	Number of different dates of birth seen in the past 30 days for a particular SSN
a_address_3	Number of same addresses seen in the past 3 days	**e_homephone_10**	Number of different home phones for each First name, Last name and Birthday combination in the past 10 days
b_address_3	Number of same addresses that are fraud in the past 3 days	**e_homephone_30**	Number of different home phones for each First name, Last name and Birthday combination in the past 30 days
c_name_6	Number of different first and last name combinations seen in the past 6 days for a particular SSN	**e_name_dob_30**	Number of different First name, Last name and birthday combination for each home phone seen in the past 30 days
c_name_10	Number of different first and last name combinations seen in the past 10 days for a particular SSN	**e_ssn_3**	Number of different SSNs for each First name, Last name and Birthday combination in the past 3 days
c_name_30	Number of different first and last name combinations seen in the past 30 days for a particular SSN	**e_ssn_6**	Number of different SSNs for each First name, Last name and Birthday combination in the past 6 days
c_phone_10	Number of different home phones seen in the past 10 days for a particular SSN		

We checked whether the variables made sense by visualizing the fraud and non-fraud records distributions in a given variable. The graphs below are bar graphs of couple selected variable.





Part IV: Fraud Algorithm

After conducting feature selection, we started to build models and predicate the precision of our models by using false discovery rate.

First of all, we separated our dataset into three different portions: training data, testing data, and out-of-time data. Out of time data hold out all records in November and December. Training and testing set had equal number of records and were randomly selected from the January to October records. We used the training data to build models, and then used the models to predict the future values on those testing and out-of-time datasets separately. In each model, we used the fraud label as the dependent variable. The fraud label would be 1 if the record was detected as fraud. Otherwise, it would be 0. Also, we used all the variables chosen from feature selection as independent variables. After running each model, we calculated each algorithm's false discovery rate at top 10% by summing up the fraud column of our top 10% records and dividing it by the sum of the fraud column of all data.

We used 5 models to predict fraudulent records.

Logistic Regression:
Using the *glm* function in R, we fitted a logistic regression model on the training dataset and used the trained model to predict training, testing and out of time records based on their input expert variables. We used the predicted probability to sort the records, and calculated the fraud detection rate at 10% penetration utilizing the function mentioned earlier.

Support Vector Machine (SVM):
We used Python to build our SVM model. Specifically, we imported the SVM class from *sci-kit learn* module and utilized its SVC function to fit an SVM model based on the training dataset.

To get a different score for each record so that we can rank them, we set up the model to return a probability estimate for each record (i.e., the likelihood that the record is fraudulent) instead of merely predicting if a record is fraud.

We experimented with Gaussian, polynomial, linear and sigmoid kernel functions, and the result of each attempt showed that linear kernel yielded the highest fraud detection rate for training, testing and out of time data. Thus, our finalized SVM model used linear regression as its kernel function.

Random Forest:
We used the *randomForest* package in R. When we trained the random forest model, we tried generating 500, 1000 and 5000 decision trees and decided to go with 1000 trees since it gave us a reasonably high fraud detection at 10% penetration rate without being too computationally expensive.

Boosted Trees:
Our boosted tree model was built using the *gbm* package in R. We fine-tuned the model by setting the number of trees to 5000, learning rate to .01, minimum number of samples required in a terminal node to 20, and the maximum depth of variable interactions to 4. These adjustments were made to increase the precision of our model.

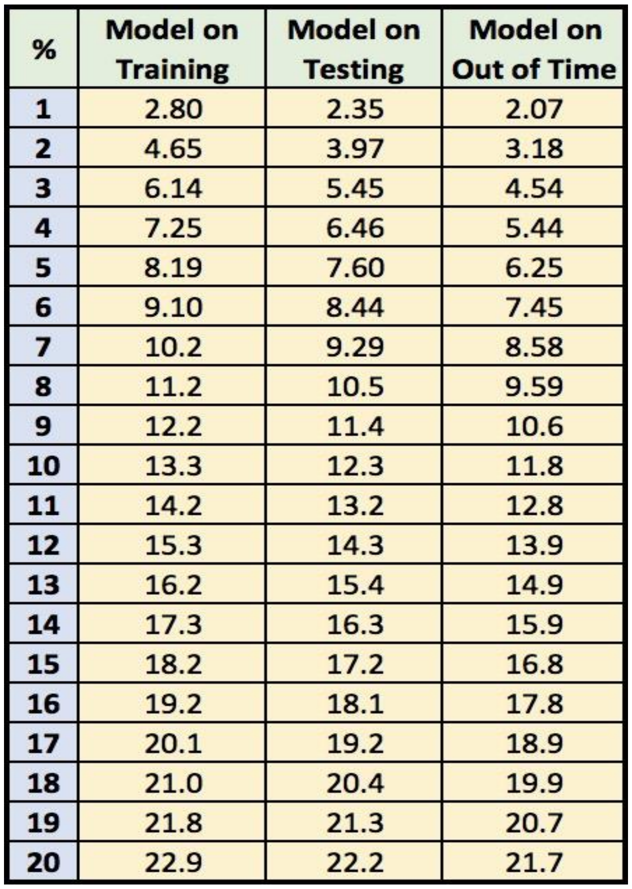
Neural Network:
We used the *keras* package in Python to build the neural network. More specifically, We used *Sequential()* function to create the model and use *model.add()* function to add layers with the activation type *'relu'*. We tried models of 3, 4, and 5 layers respectively and found that the three-layer model generated the best result. For the number of nodes, we added 19 nodes in the first layer and 30 nodes in the second layer.

The following table shows the fraud detection rate at top 10% for each model.

FDR @ 10%			
Model	**Training**	**Testing**	**Out of Time**
Logistic Regression	12.60%	12.32%	10.65%
SVM	12.57%	12.35%	10.93%
Random Forest	12.75%	11.87%	10.70%
Boosted Trees	12.73%	12.59%	10.90%
Neural Network	13.27%	12.31%	11.78%

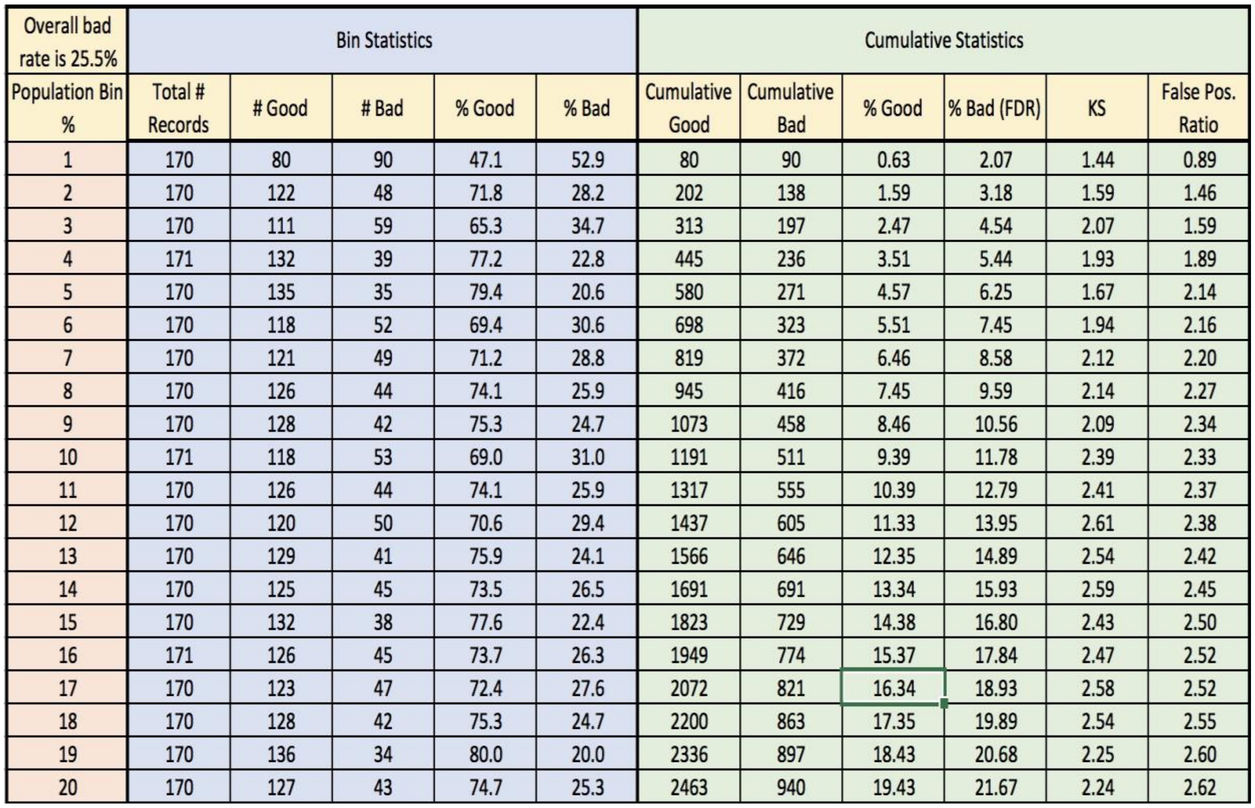
Part V: Results

By comparing the fraud detection rates in the table shown above, we found that neural network had the best performance at 10% penetration. Thus, we computed the fraud detection rate of the neural network model at different penetration levels, ranging from 1% to 20%. The following table shows the FDR of training data, testing data and out-of-time data at 1% to 20% penetration for the Neural Network model.

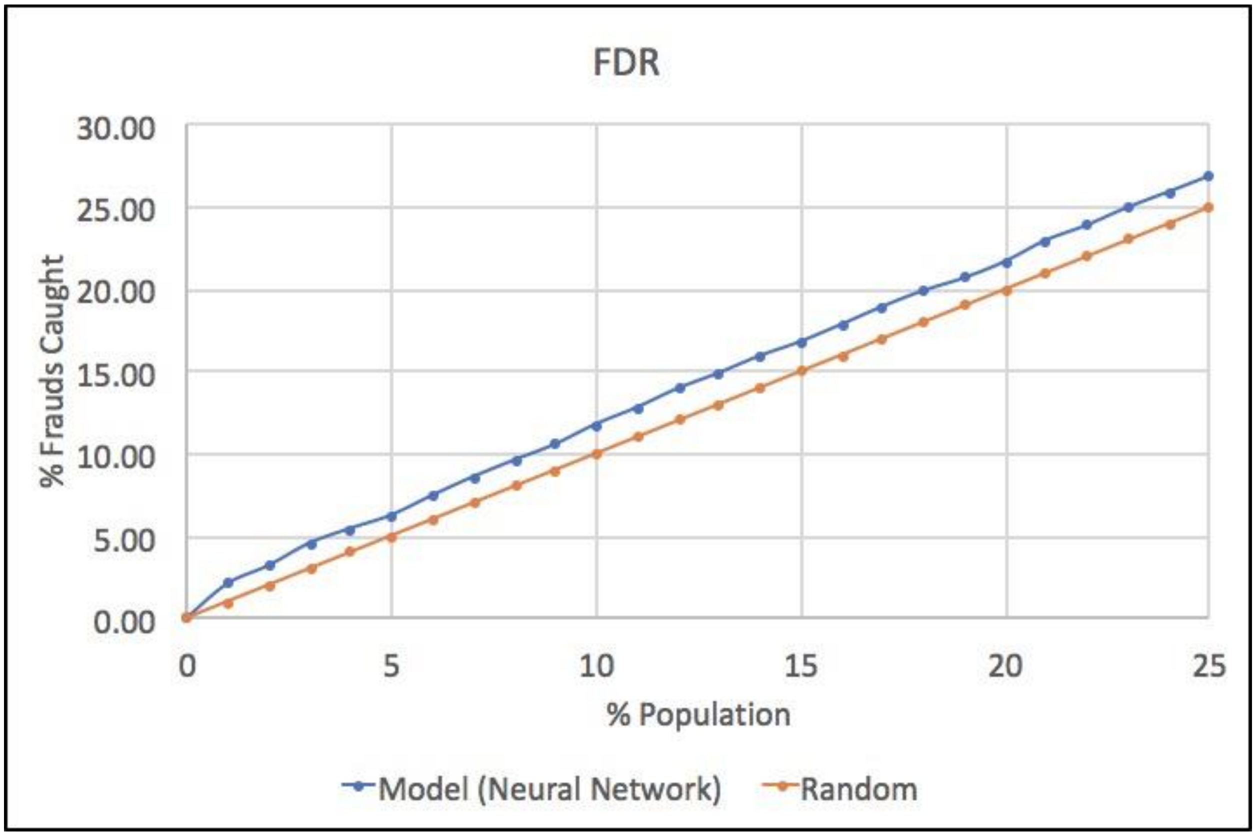


%	Model on Training	Model on Testing	Model on Out of Time
1	2.80	2.35	2.07
2	4.65	3.97	3.18
3	6.14	5.45	4.54
4	7.25	6.46	5.44
5	8.19	7.60	6.25
6	9.10	8.44	7.45
7	10.2	9.29	8.58
8	11.2	10.5	9.59
9	12.2	11.4	10.6
10	13.3	12.3	11.8
11	14.2	13.2	12.8
12	15.3	14.3	13.9
13	16.2	15.4	14.9
14	17.3	16.3	15.9
15	18.2	17.2	16.8
16	19.2	18.1	17.8
17	20.1	19.2	18.9
18	21.0	20.4	19.9
19	21.8	21.3	20.7
20	22.9	22.2	21.7

The next table shows some statistics that summarize the performances of neural network model on out-of-time data. The columns in blue on the left-hand side are summary statistics of data within each population bin. The columns in green on the right-hand side are cumulative statistics of each population bin. Note that KS is equal to percentage bad - percentage good, and the False Positive rate is equal to cumulative good divided by cumulative bad.



Overall bad rate is 25.5%	Bin Statistics					Cumulative Statistics					
Population Bin %	Total # Records	# Good	# Bad	% Good	% Bad	Cumulative Good	Cumulative Bad	% Good	% Bad (FDR)	KS	False Pos. Ratio
1	170	80	90	47.1	52.9	80	90	0.63	2.07	1.44	0.89
2	170	122	48	71.8	28.2	202	138	1.59	3.18	1.59	1.46
3	170	111	59	65.3	34.7	313	197	2.47	4.54	2.07	1.59
4	171	132	39	77.2	22.8	445	236	3.51	5.44	1.93	1.89
5	170	135	35	79.4	20.6	580	271	4.57	6.25	1.67	2.14
6	170	118	52	69.4	30.6	698	323	5.51	7.45	1.94	2.16
7	170	121	49	71.2	28.8	819	372	6.46	8.58	2.12	2.20
8	170	126	44	74.1	25.9	945	416	7.45	9.59	2.14	2.27
9	170	128	42	75.3	24.7	1073	458	8.46	10.56	2.09	2.34
10	171	118	53	69.0	31.0	1191	511	9.39	11.78	2.39	2.33
11	170	126	44	74.1	25.9	1317	555	10.39	12.79	2.41	2.37
12	170	120	50	70.6	29.4	1437	605	11.33	13.95	2.61	2.38
13	170	129	41	75.9	24.1	1566	646	12.35	14.89	2.54	2.42
14	170	125	45	73.5	26.5	1691	691	13.34	15.93	2.59	2.45
15	170	132	38	77.6	22.4	1823	729	14.38	16.80	2.43	2.50
16	171	126	45	73.7	26.3	1949	774	15.37	17.84	2.47	2.52
17	170	123	47	72.4	27.6	2072	821	16.34	18.93	2.58	2.52
18	170	128	42	75.3	24.7	2200	863	17.35	19.89	2.54	2.55
19	170	136	34	80.0	20.0	2336	897	18.43	20.68	2.25	2.60
20	170	127	43	74.7	25.3	2463	940	19.43	21.67	2.24	2.62



Part VI: Conclusions

In conclusion, we built a model to detect identity fraud in credit card applications using 94,866 past credit card application records with fraud labels. Our team first carefully examined all the raw variables, such as applicants' date of birth, Social Security Number, address, home phone number, etc. and completed a data quality report. The dataset is clean with no missing values. It also has a label column indicating whether a record is a fraud or not, which helps us to build a supervised fraud model.

We then built 80 expert variables using time windows. We chose five different time windows, namely 1 day, 3 days, 6 days, 10 days, and 30 days. In past number of days given by the time window, we examined 1) number of records that had the same SSN, address, and phone number; 2) number of fraud identified that had the same SSN, address, and phone number. Applying the same logic, we used the combination of first and last name, full name and date of birth, full name and SSN to create additional variables. The combinations we chose should indicate different individuals, so if these combinations appear in the application multiple times within a month, they would have higher chances to be fraudulent.

The feature selection process helped us to reduce expert variables from 80 to 19. We used the KS value to select 40 variables that had the largest distance between the distribution of fraud records and that of non-fraud records. Backward selection on logistic regression helped us to identify the final 19 variables that are most significant in indicating actual frauds.

For fraud detection models, we used logistic regression, SVM, random forest, boosted trees, and neural network. We fitted models on our training data and validated them on both testing data and the out-of-time data. The result showed that neural network model has the highest fraud detection rate. Thus, we dug into the model and calculated the fraud detection rate for every 1% increment up to top 20% of records with the highest risk of being fraud. After that, we drew an FDR graph to visualize the performance of the neural network model.

In the future, if we have more time and more data, we will optimize our analyses in several ways. We could build more expert variables, try different Filter Feature Selection methods such as the Fisher score, mutual information, or information value, and try more layers and nodes to tuning our best model even more, therefore, to increase overall fraud detection rate for the client.

Appendix:

Data Quality Report (DQR)

Summary Statistics for All Variables:

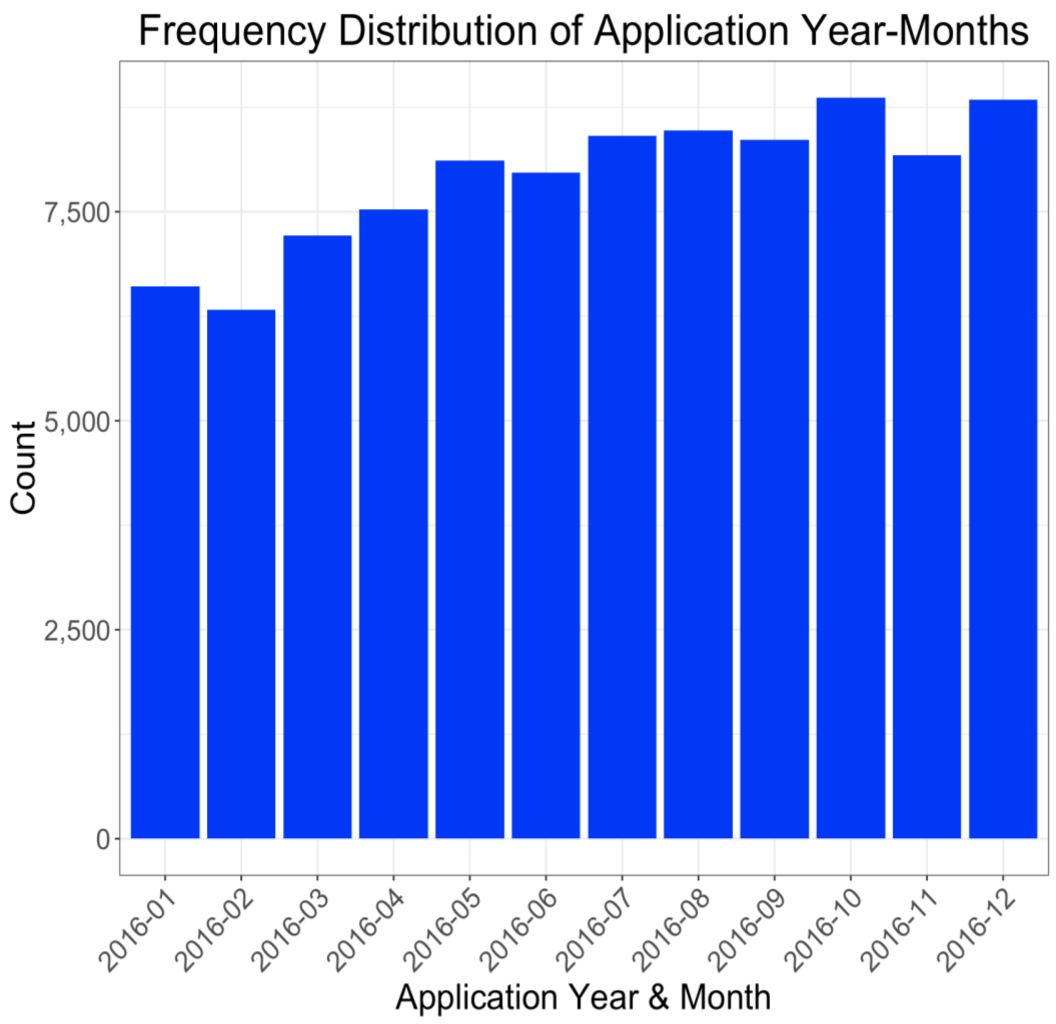
There are two date variables, six categorical variables and one Boolean variable in the dataset. Total number of records is 94866. The following is a summary table that shows the class, the percentage of populated records, and the number of unique records of each variable.

Summary Statistics for Application Data			
Variable	Class	% Populated	Unique Number
Date	Date	100%	365
DOB	Date	100%	30599
SSN	Categorical	100%	86771
Firstname	Categorical	100%	14626
Lastname	Categorical	100%	31513
Address	Categorical	100%	88167
Zip5	Categorical	100%	15855
Homephone	Categorical	100%	20762
Fraud (Y/N)	Boolean	100%	2

Distribution of Each Variable

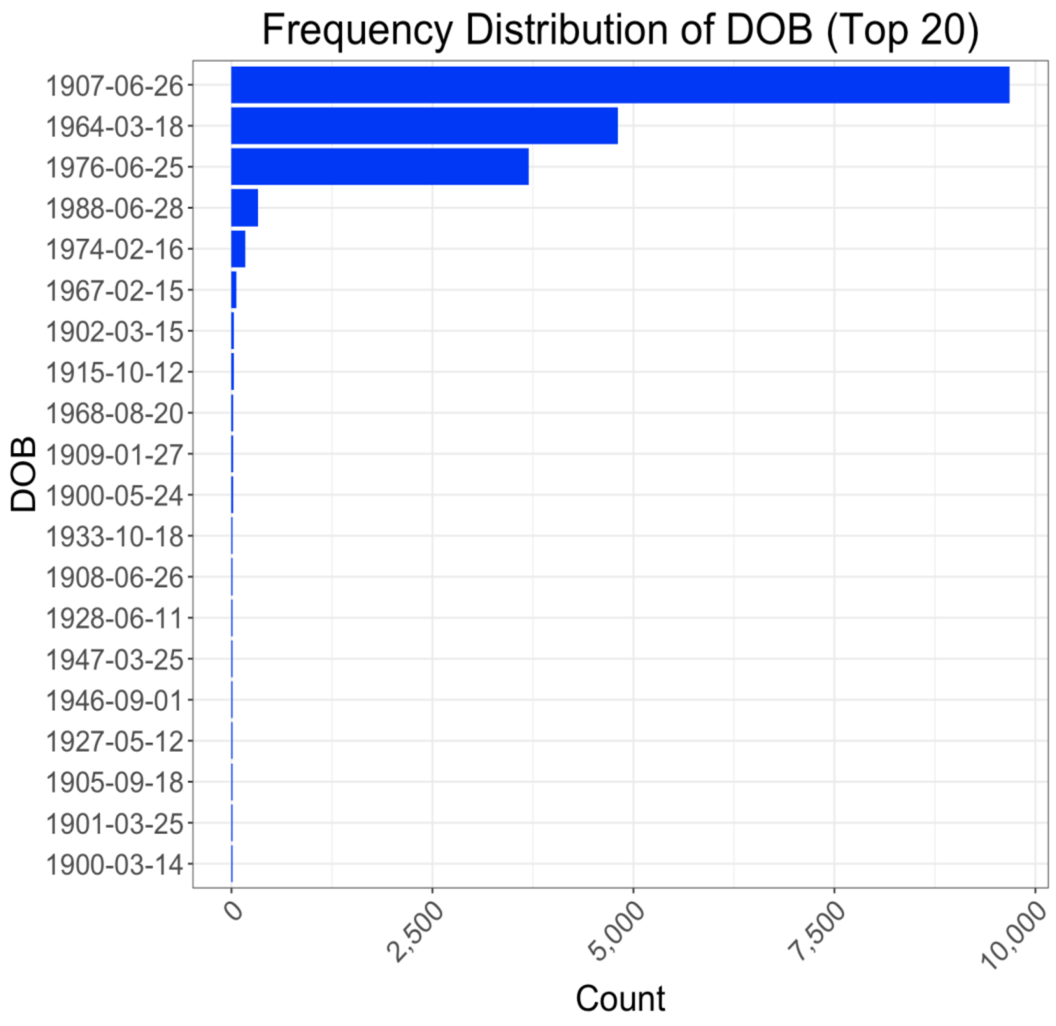
Date	
Description:	The date when credit card application is filed
Class:	Date
Unique Values:	365
Populated %:	100%

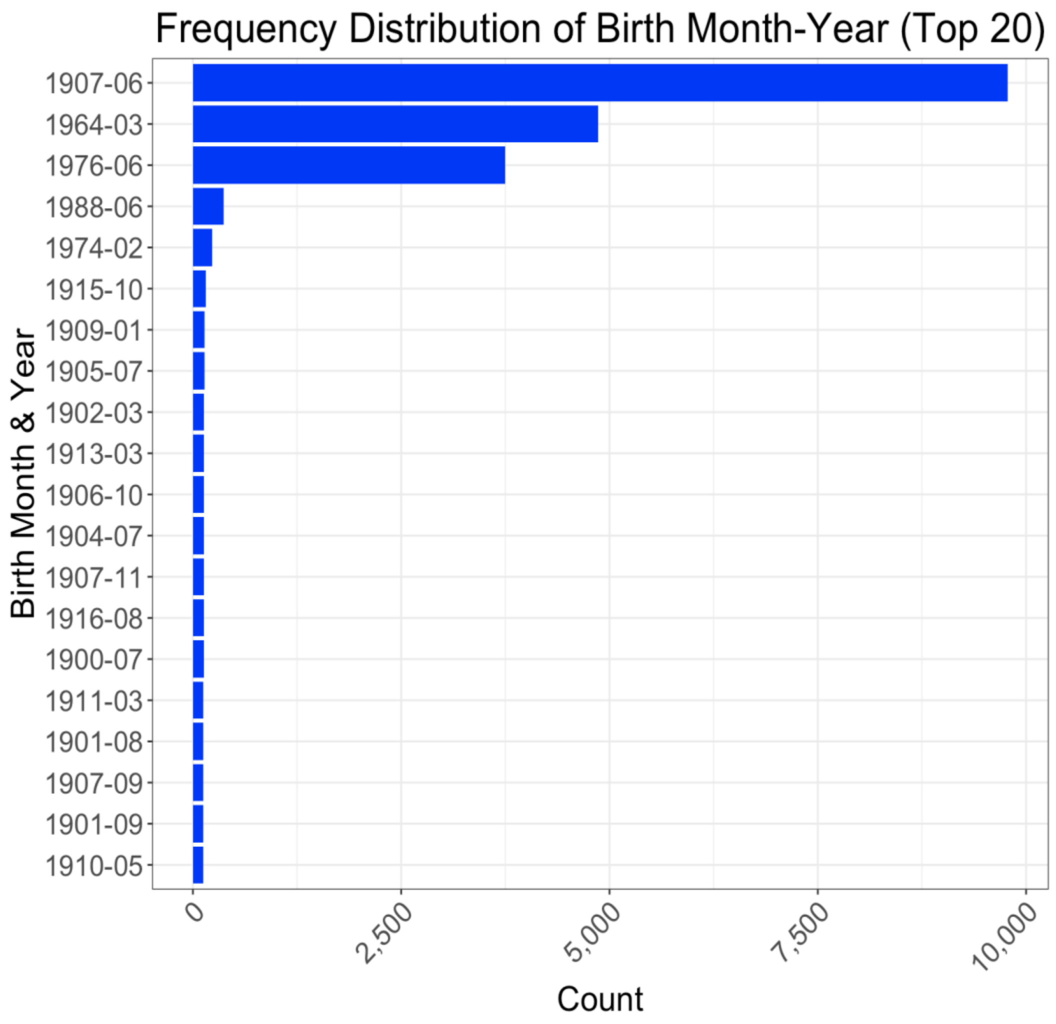
The number of applications increases over the year, but overall the distribution is consistent.

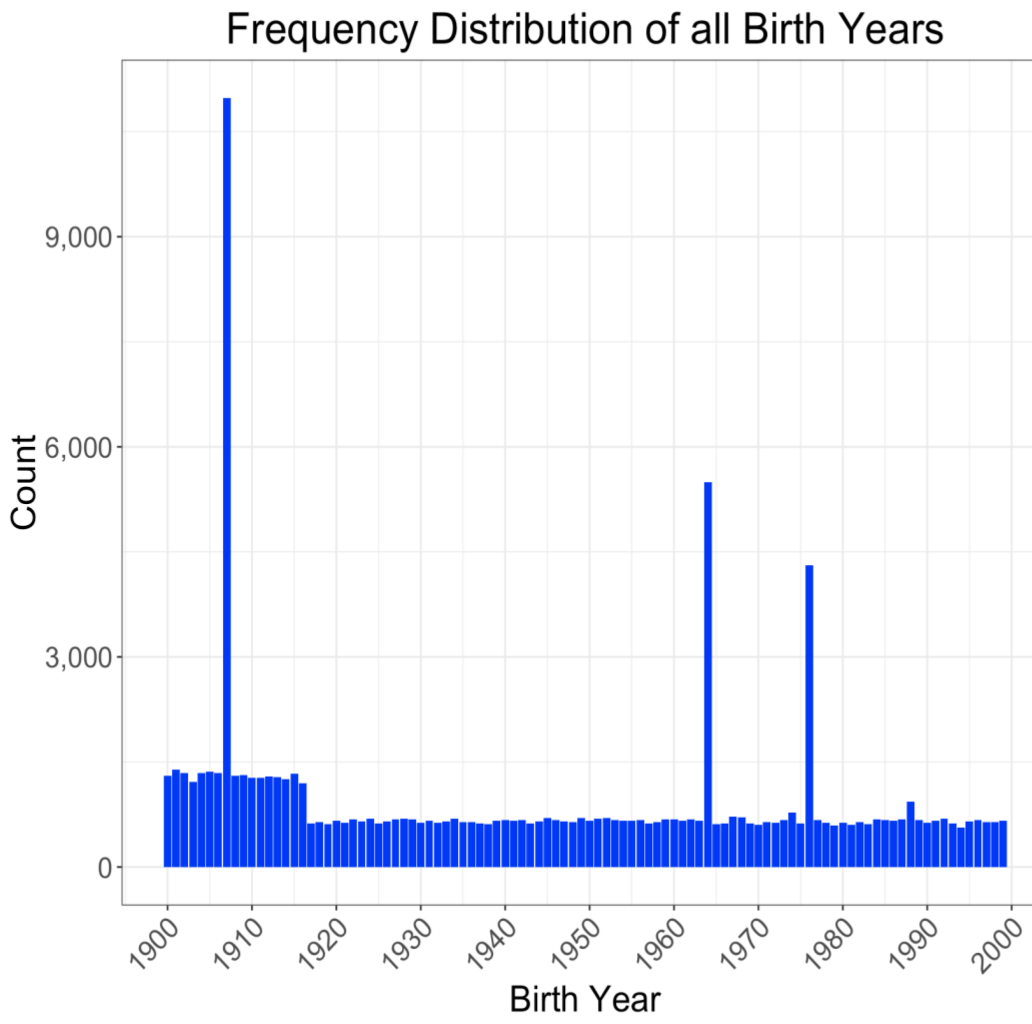


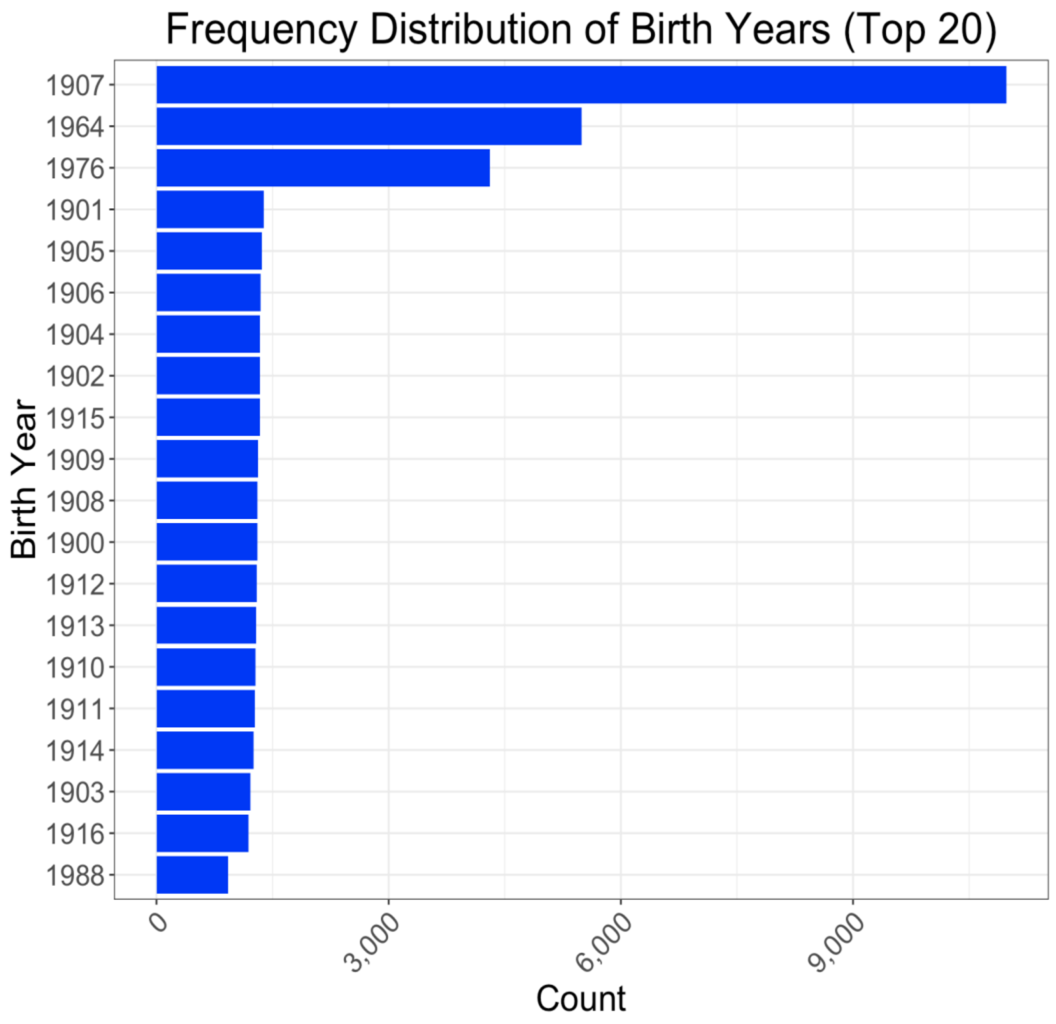
DOB	
Description:	Credit card applicant’s date of birth
Class:	Date
Unique Values:	30599
Populated %:	100%

The distribution of data of birth shows a few excessively used dates, which is very unusual. In later analysis, our team can look into the linkages between these specific dates of birth and records that have extreme last names, home phones, or addresses.




Frequency Distribution of Birth Month-Year (Top 20)
Birth Month & Year
1907-06
1964-03
1976-06
1988-06
1974-02
1915-10
1909-01
1905-07
1902-03
1913-03
1906-10
1904-07
1907-11
1916-08
1900-07
1911-03
1901-08
1907-09
1901-09
1910-05
0
2,500
5,000
7,500
10,000
Count

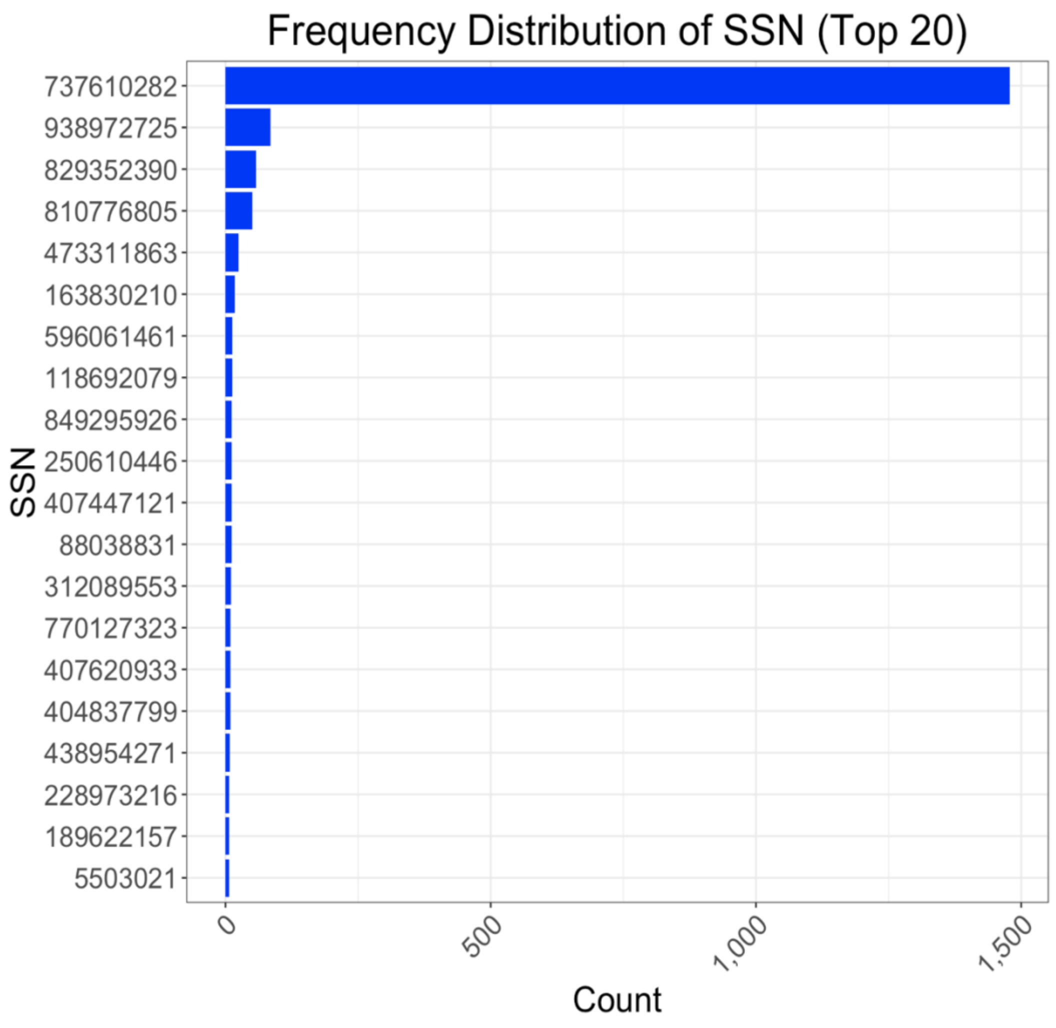

Frequency Distribution of all Birth Years
Count
9,000
6,000
3,000
0
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
Birth Year

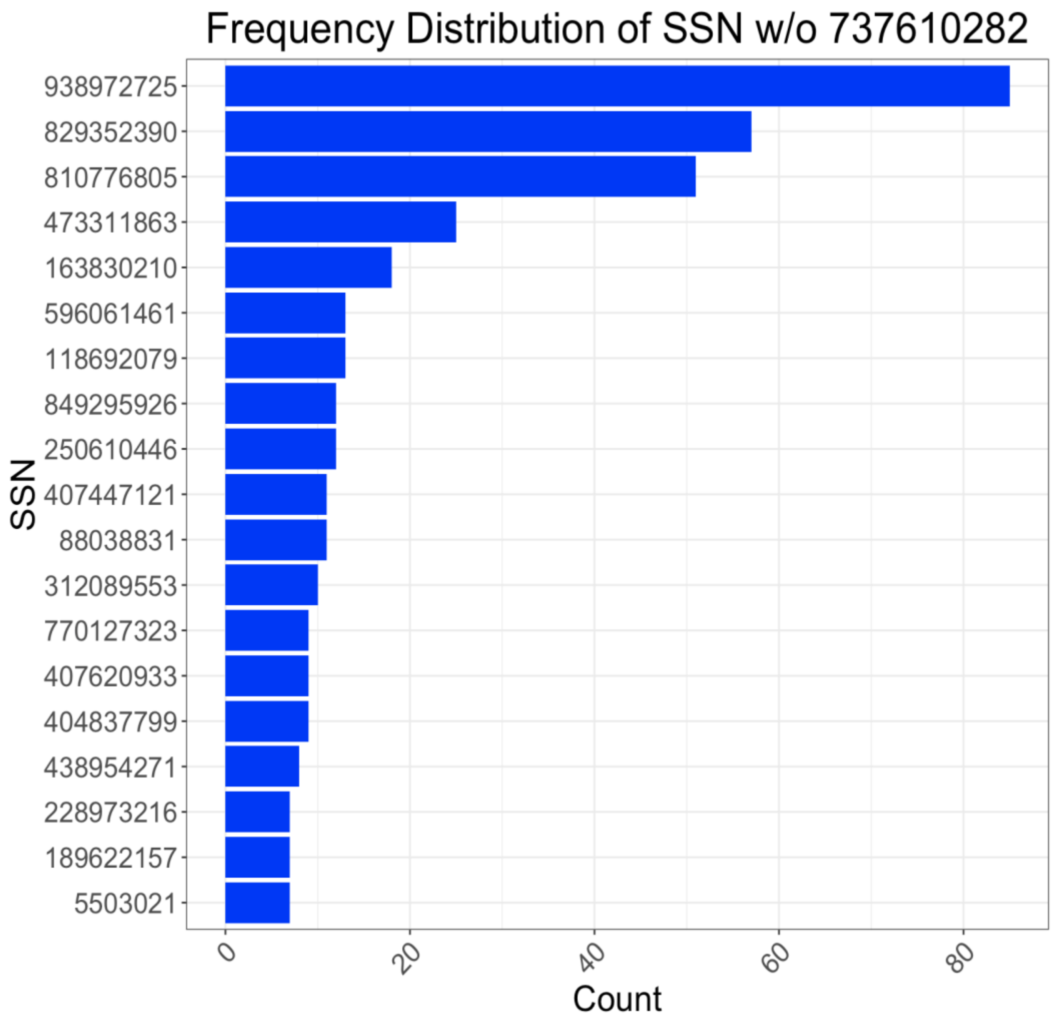

Frequency Distribution of Birth Years (Top 20)
Birth Year
1907
1964
1976
1901
1905
1906
1904
1902
1915
1909
1908
1900
1912
1913
1910
1911
1914
1903
1916
1988
0
3,000
6,000
9,000
Count

SSN	
Description:	Credit card applicant's social security number
Class:	Categorical
Unique Values:	86771
Populated %:	100%

The count of social security number (SSN) 737610282 is abnormally high. This could indicate missing values that are filled by the most frequent SSN number.

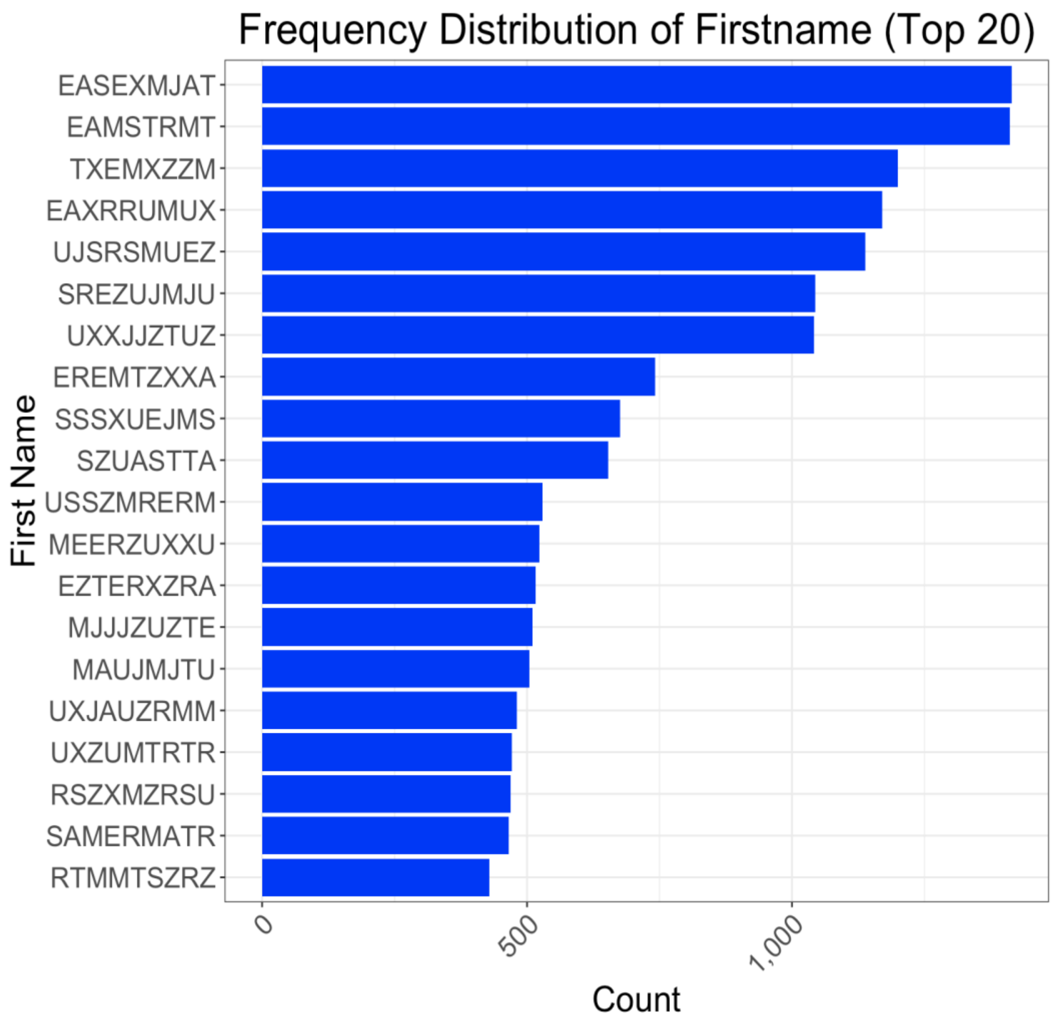
The following graph excludes this SSN, so it is a better representation of the SSN distribution overall. It is close to a polynomial distribution with three frequently used SSN.




Frequency Distribution of SSN w/o 737610282
SSN
938972725
829352390
810776805
473311863
163830210
596061461
118692079
849295926
250610446
407447121
88038831
312089553
770127323
407620933
404837799
438954271
228973216
189622157
5503021
0
20
40
60
80
Count

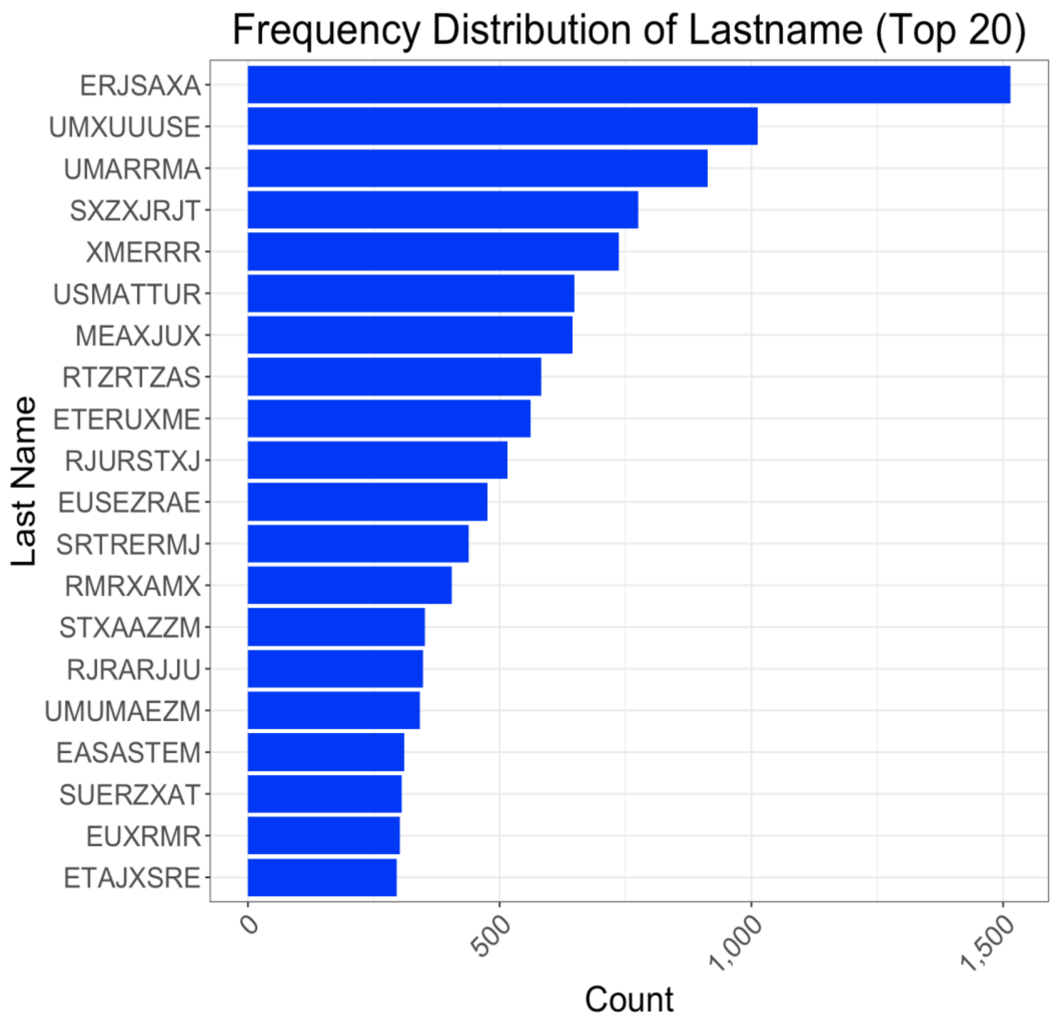
Firstname	
Description:	Credit card applicant’s first name
Class:	Categorical
Unique Values:	14626
Populated %:	100%

This graph indicates some frequently used first names. Further analysis is needed to determine whether these highly used first names are only representations of common names or indications of fraud.



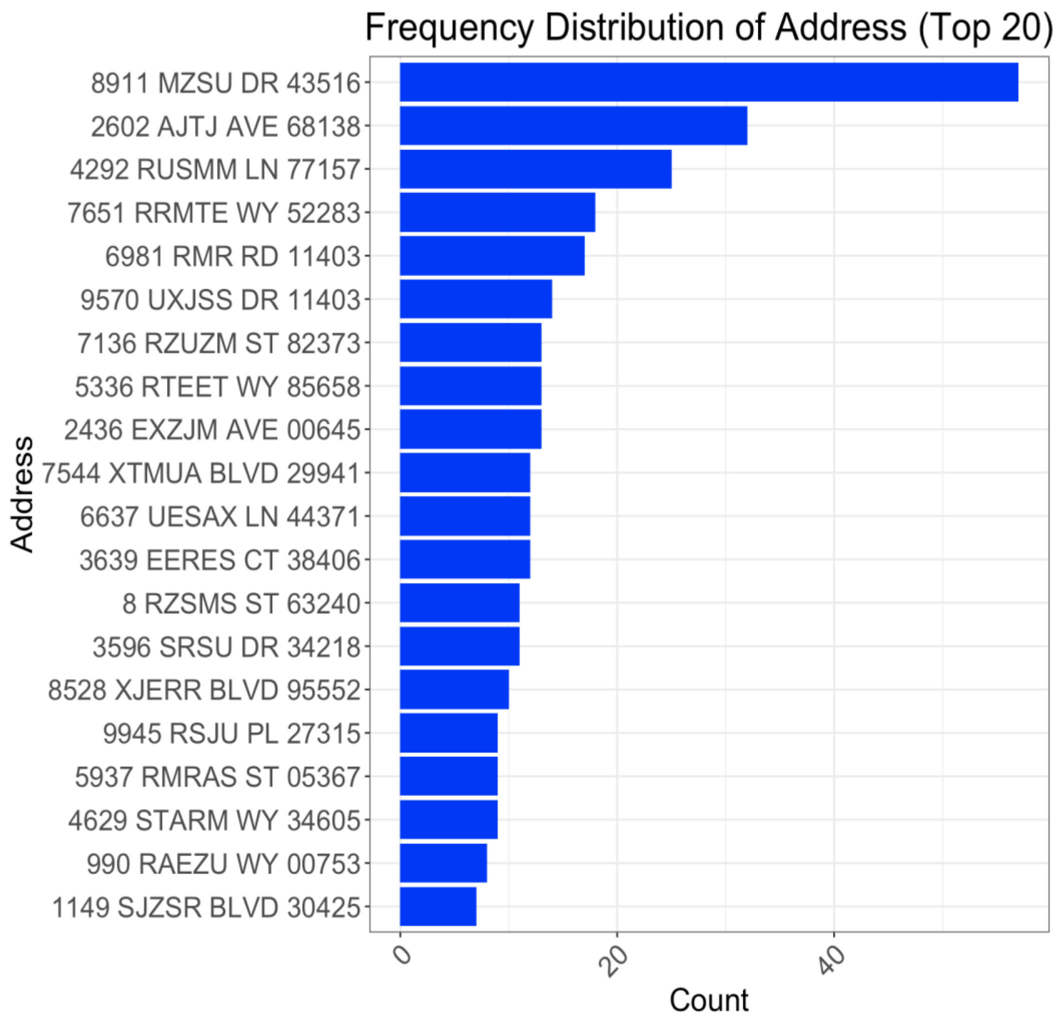
Lastname	
Description:	Credit card applicant’s last name
Class:	Categorical
Unique Values:	31513
Populated %:	100%

This graph indicates some excessively used last names. Because it is less common for people to have the same last name than first name, the highly-ranked last names are suspicious and could be signals of fraud.



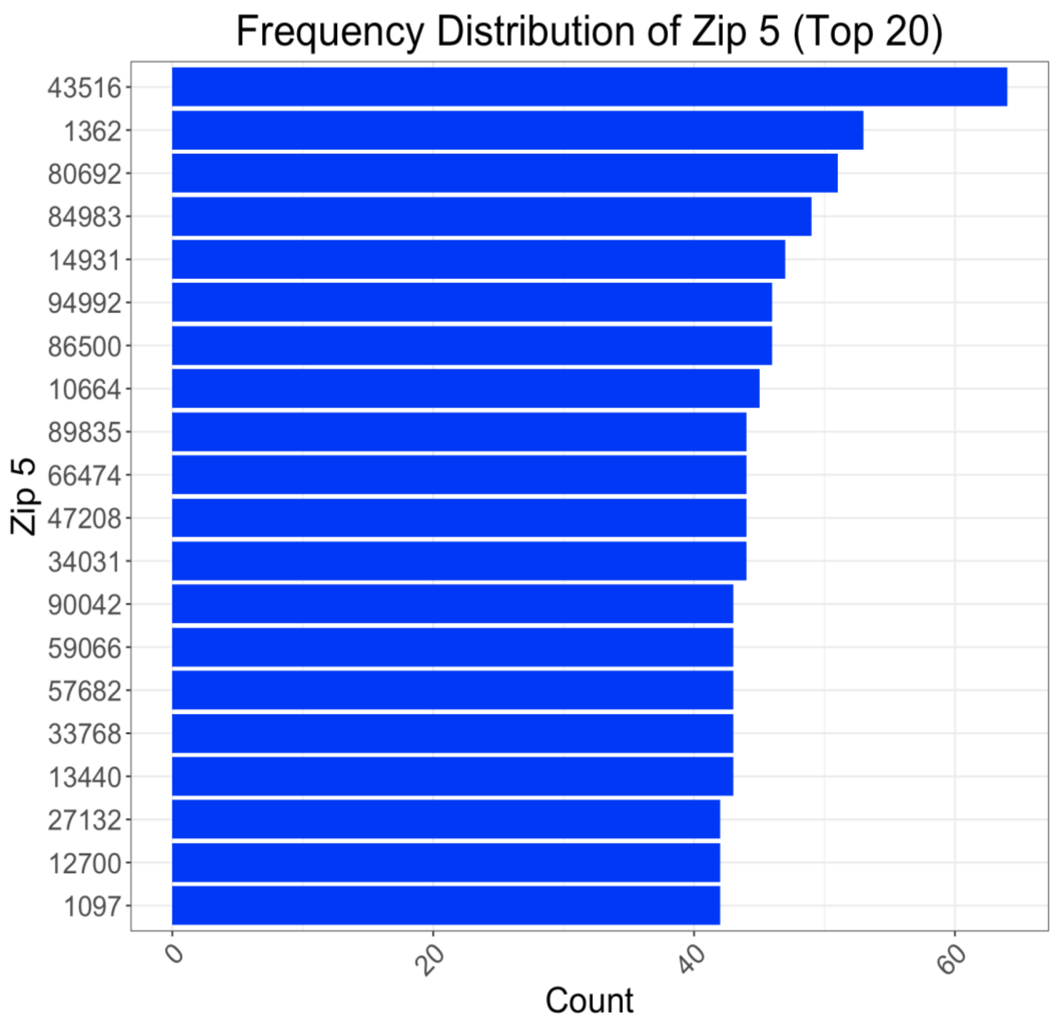
Address	
Description:	Credit card applicant’s home address
Class:	Categorical
Unique Values:	88167
Populated %:	100%

This graph shows the customers’ home addresses that appear most frequently. In particular, the address “8911 MZSU DR 43516” appears far more frequently than other addresses. Later, we’ll further examine details of this particular address and determine whether it’s fraudulent.



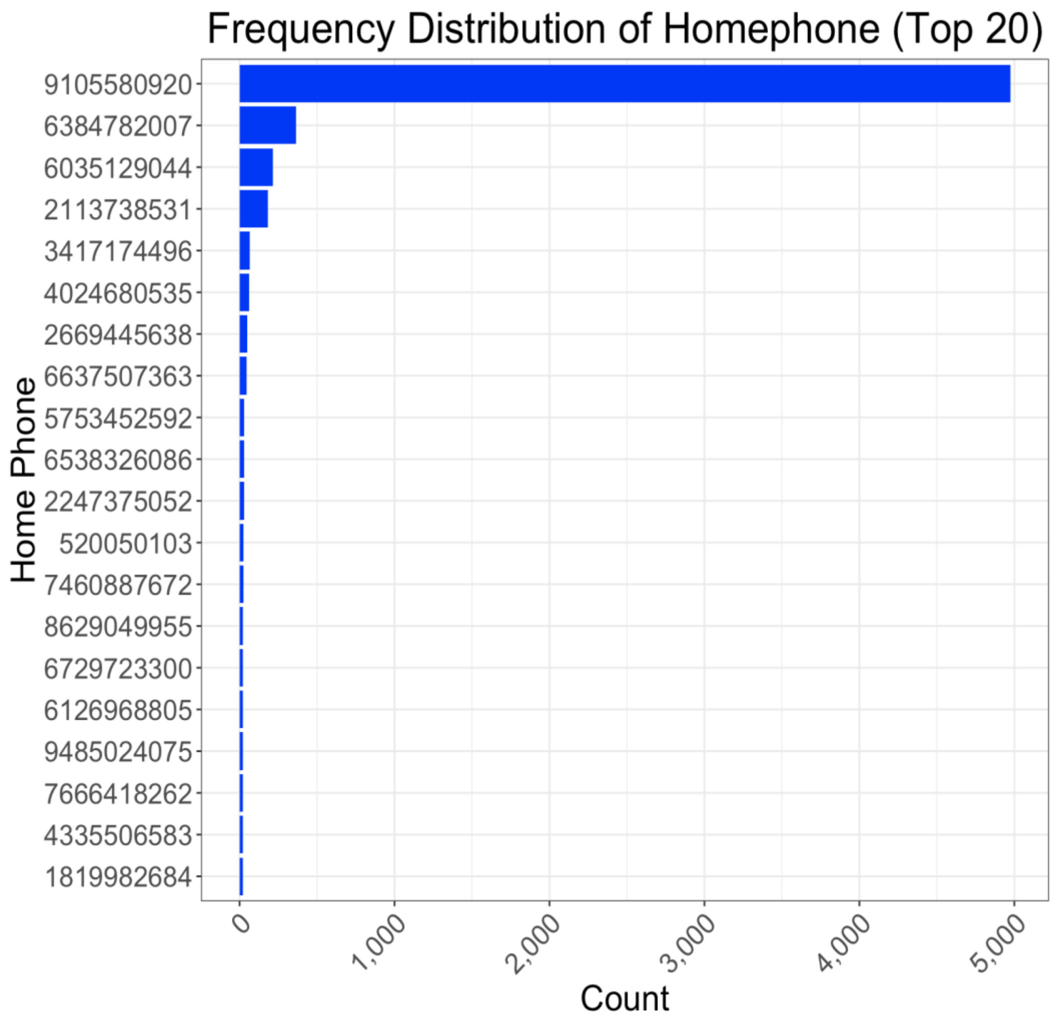
Zip5	
Description:	The zip code associated with credit card applicant's address (5 digits)
Class:	Categorical
Unique Values:	15855
Populated %:	100%

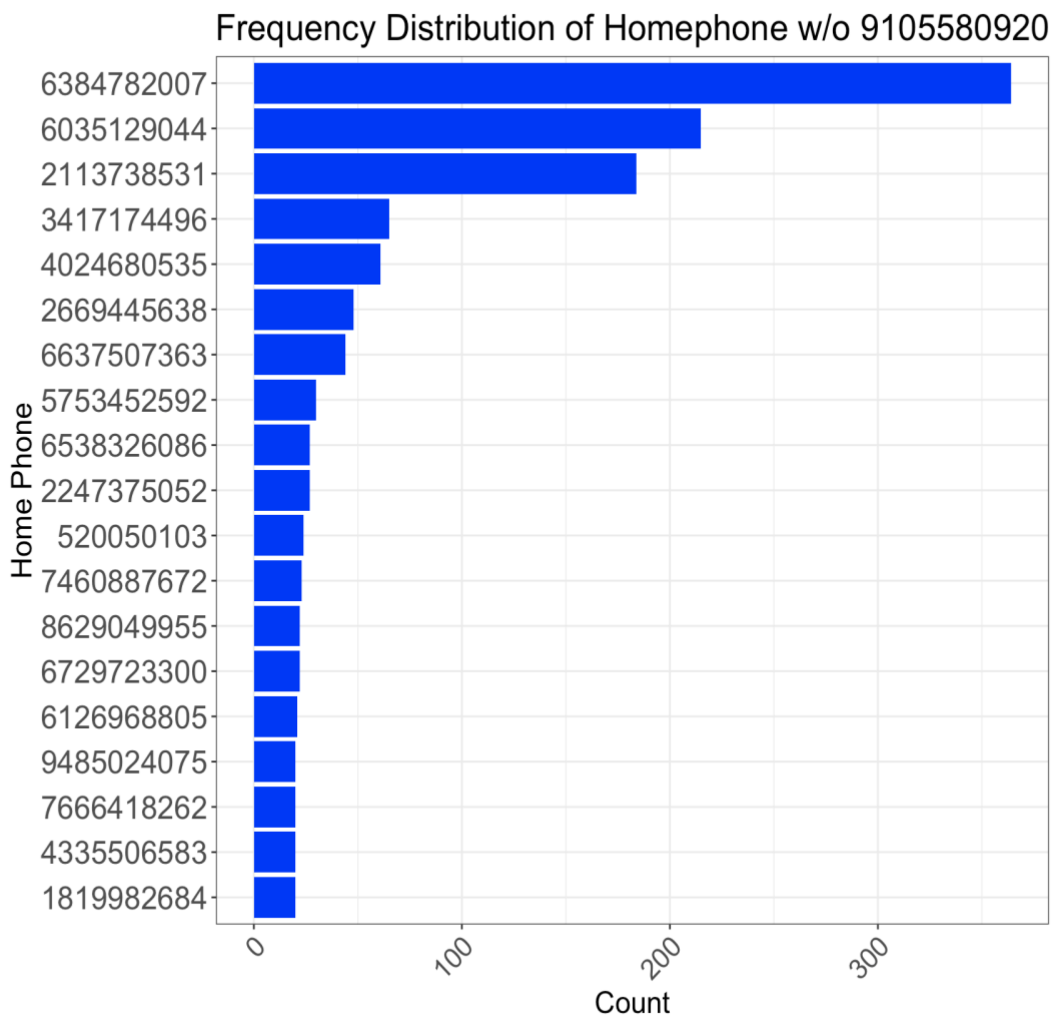
This graph plots the number of records associated with each Zip code, and the distribution looks quite normal with no extreme values.



Homephone	
Description:	Credit applicant’s home phone number
Class:	Categorical
Unique Values:	20762
Populated %:	100%

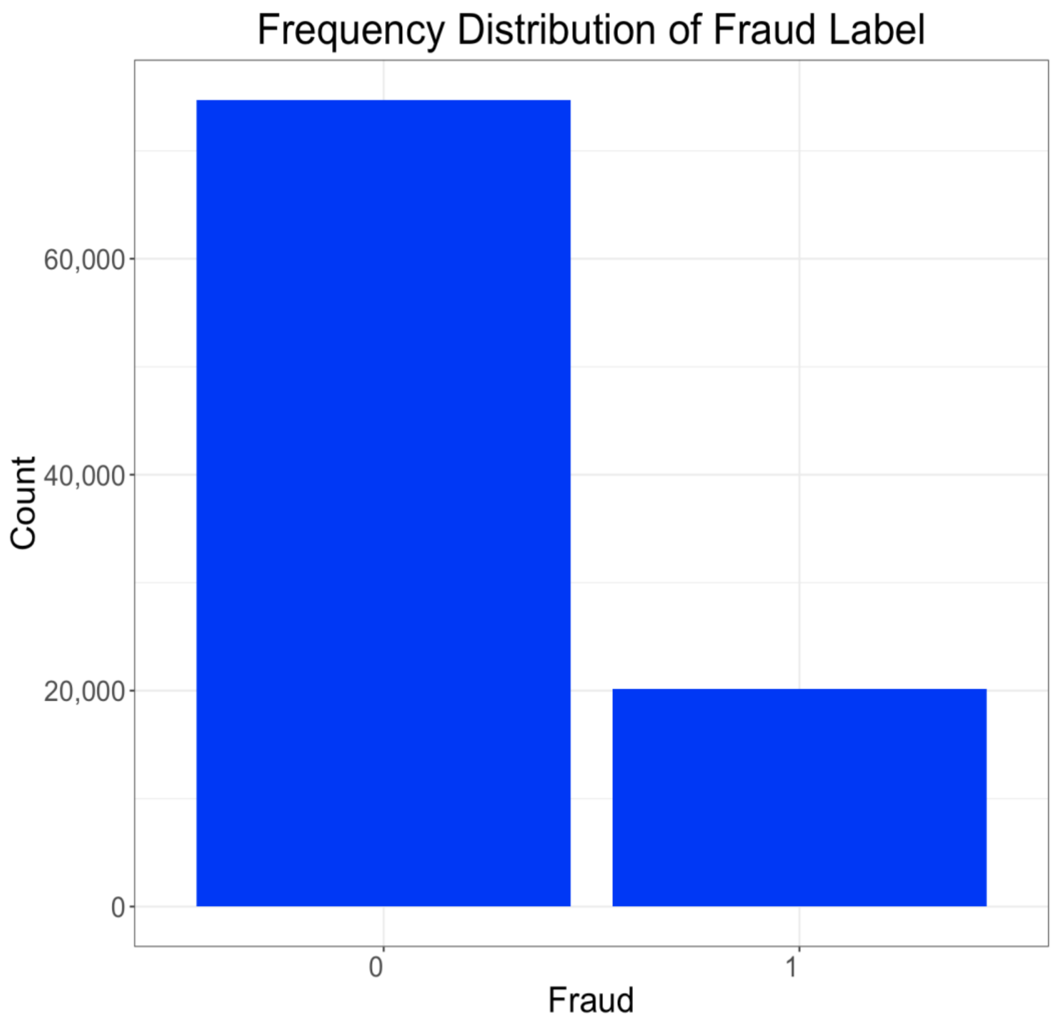
This graph shows the distribution of applicants’ home phone numbers. One particular number appeared far more frequently than other numbers. This is quite unusual because each applicant’s phone number is supposed to be unique, and having the same number appear ~5000 times could be indication of fraud, or missing values replaced by a default value.




Frequency Distribution of Homephone w/o 9105580920
Home Phone
6384782007
6035129044
2113738531
3417174496
4024680535
2669445638
6637507363
5753452592
6538326086
2247375052
520050103
7460887672
8629049955
6729723300
6126968805
9485024075
7666418262
4335506583
1819982684
0
100
200
300
Count

Fraud	
Description:	Whether the application is fraudulent (“1” for Yes, “0” for No)
Class:	Boolean
Unique Values:	2
Populated %:	100%

This graph shows the frequency distribution of fraud. The label 0 has over 60000 counts while 1 only has 2000 counts. This means that 22% of record are classified as fraud.


Frequency Distribution of Fraud Label
Count
Fraud