DETECT FRAUD IN CREDIT CARD TRANSACTION DATA

MGTA 495 FRAUD ANALYTICS PROJECT 2

Jie Chen, Shiyi Hua, Kexin Liu, Qiuyi Lu, Zhanyi Xu, Jingwen Yan, Ziyuan Yan

Date: 06/14/2020

Table of Contents

I. Executive Summary

The problem of credit card transaction fraud is very serious in America. According to ACI Worldwide, 47% of Americans have fallen victim to credit card fraud in the past five years. The Nilson Report estimates that losses from credit card fraud exceeded $27 billion in 2018. Therefore, the bank needs to spend a lot of energy and money to detect this kind of fraud.

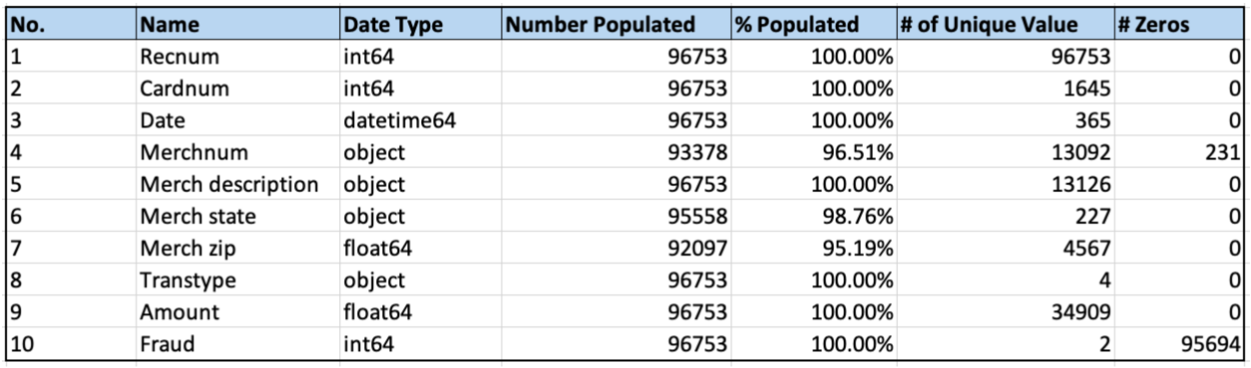
The purpose of this project, then, is to build a robust supervised fraud model that could properly detect potential credit card transaction fraud. There are mainly three ways that could cause credit card transaction fraud, which are lost/stolen cards, common points of compromise (merchants "steal" many cards, counterfeit cards, skimmers at gas stations, POSs, restaurants, or stores, etc.), and online account hacking. As such, we could conclude several signals for transaction fraud: the burst of activity at different merchants, larger than normal purchase amounts at same or different merchants, used at merchants not used before for that card, used at very different geography, used at a high-risk merchant, increased usage in card-not-present, employee or merchant invents transactions, infrequent recurring charges, the same amount or same merchant, employee or merchant invents a fictitious merchant.

We conducted this project based on over 96,000 historical credit card transaction records with labeled fraud. After cleaning the dataset, we would create candidate variables that could capture the features of transaction fraud and conduct feature selection to find expert variables. We would separate the out of time data and use the rest of the data to build different supervised models, such as logistics, neural network, random forest, and bootstrap. Finally, we could find the best model for the OOT data and calculate the overall saving by detecting frauds. Based on the result, we would make a recommendation for where the client should set a score cutoff (OOT population % bins).

II. Description of Data

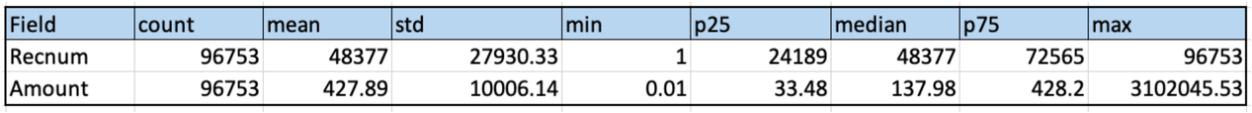
The dataset used for this project shows real credit card transaction information in various merchants in a chronological order, which is from January 2010 to December 2010. The dataset has 96,753 rows and 10 fields. It not only includes the basic transaction information like merchants' names, locations, transaction types, and amount but also labels whether a record is a fraud or not. Fields Merchnum, Merch State, Merch zip have null values, and fields Merchnum and Fraud have zero values. The following tables are summary tables of field statistics.

Summary of all the fields in the dataset:



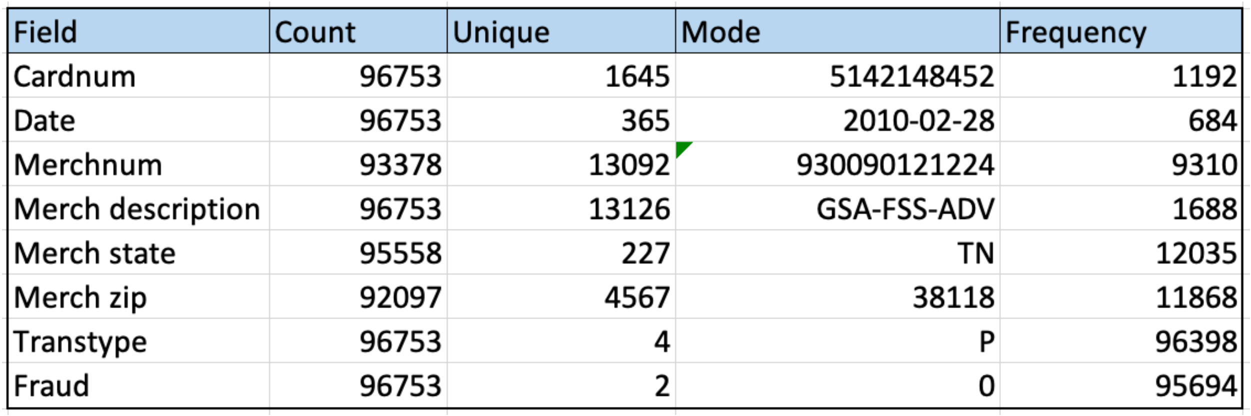
No.	Name	Date Type	Number Populated	% Populated	# of Unique Value	# Zeros
1	Recnum	int64	96753	100.00%	96753	0
2	Cardnum	int64	96753	100.00%	1645	0
3	Date	datetime64	96753	100.00%	365	0
4	Merchnum	object	93378	96.51%	13092	231
5	Merch description	object	96753	100.00%	13126	0
6	Merch state	object	95558	98.76%	227	0
7	Merch zip	float64	92097	95.19%	4567	0
8	Transtype	object	96753	100.00%	4	0
9	Amount	float64	96753	100.00%	34909	0
10	Fraud	int64	96753	100.00%	2	95694

For numeric fields:



Field	count	mean	std	min	p25	median	p75	max
Recnum	96753	48377	27930.33	1	24189	48377	72565	96753
Amount	96753	427.89	10006.14	0.01	33.48	137.98	428.2	3102045.53

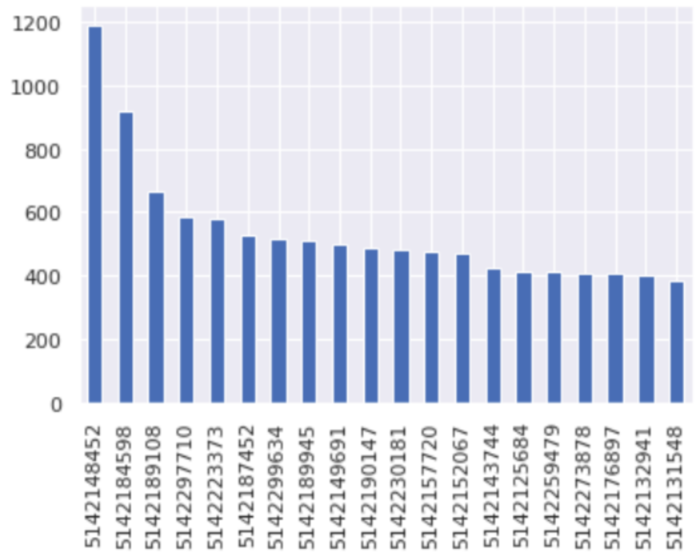
For categorical fields:



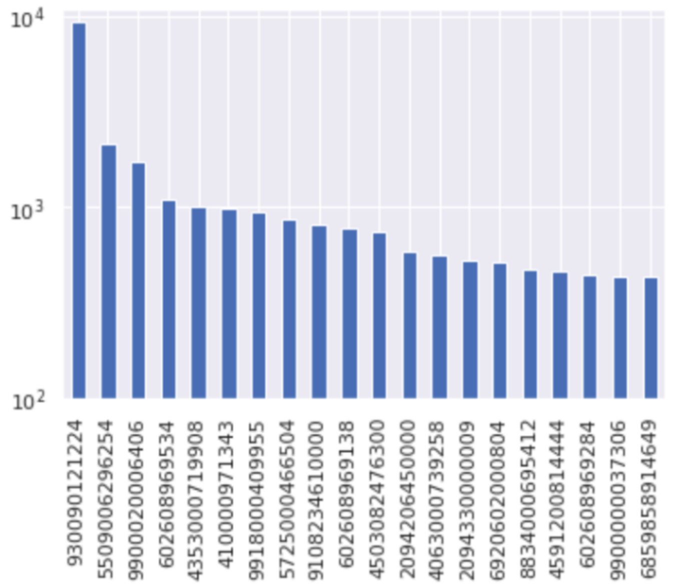
Field	Count	Unique	Mode	Frequency
Cardnum	96753	1645	5142148452	1192
Date	96753	365	2010-02-28	684
Merchnum	93378	13092	930090121224	9310
Merch description	96753	13126	GSA-FSS-ADV	1688
Merch state	95558	227	TN	12035
Merch zip	92097	4567	38118	11868
Transtype	96753	4	P	96398
Fraud	96753	2	0	95694

In credit card transaction fraud issues, we consider card number, Merchant number, Merchant description, Amount, and Fraud as important fields. The following are the distribution/histograms of those fields. Other fields' distributions and histograms can be found in Appendix A.

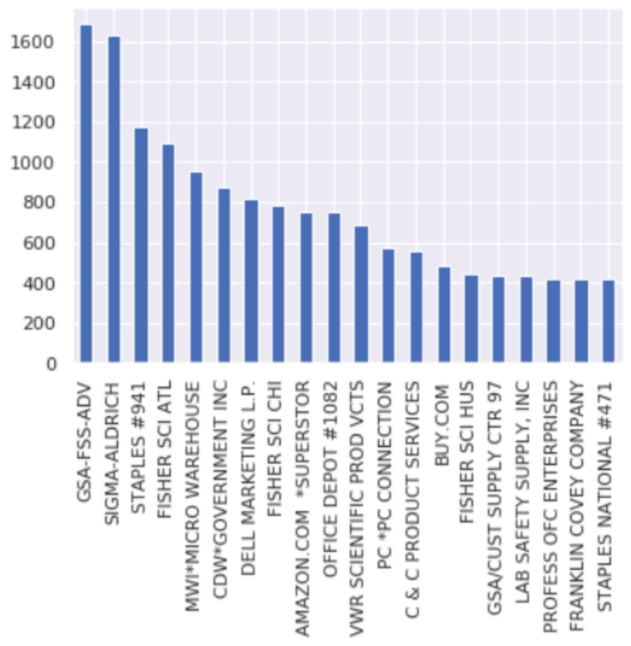
1. **Cardnum**: Card number of the credit card that made transactions.



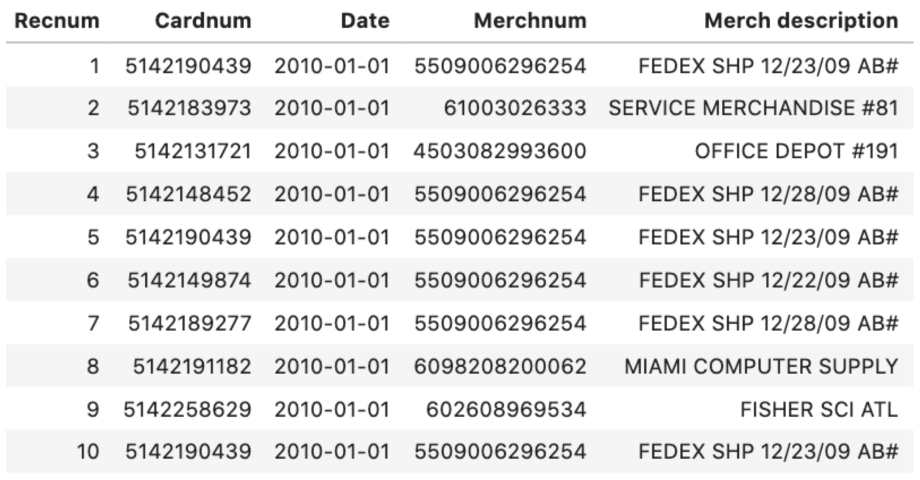
2. **Merchnum:** Number to identify different merchants. (Y-axis was log-transformed)



3. **Merch description:** The description of each merchant.



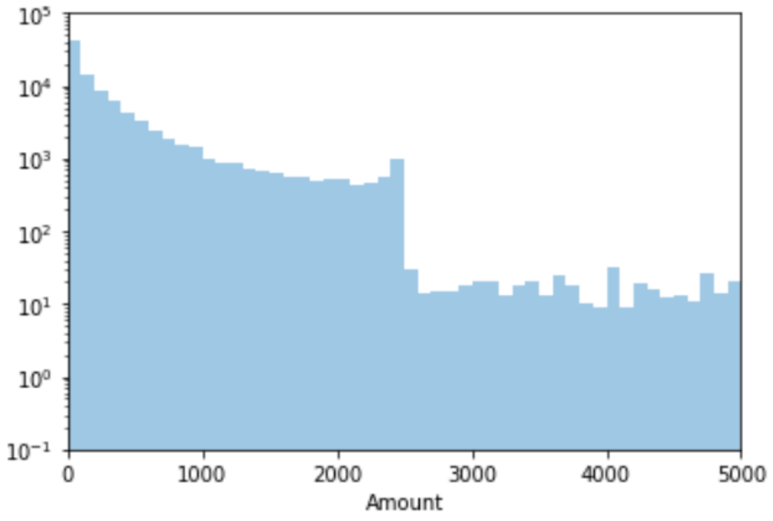
If we look at the original dataset, in the Merchant description column, many merchants are described as "FEDEX SHP" with different dates attached.



Recnum	Cardnum	Date	Merchnum	Merch description
1	5142190439	2010-01-01	5509006296254	FEDEX SHP 12/23/09 AB#
2	5142183973	2010-01-01	61003026333	SERVICE MERCHANDISE #81
3	5142131721	2010-01-01	4503082993600	OFFICE DEPOT #191
4	5142148452	2010-01-01	5509006296254	FEDEX SHP 12/28/09 AB#
5	5142190439	2010-01-01	5509006296254	FEDEX SHP 12/23/09 AB#
6	5142149874	2010-01-01	5509006296254	FEDEX SHP 12/22/09 AB#
7	5142189277	2010-01-01	5509006296254	FEDEX SHP 12/28/09 AB#
8	5142191182	2010-01-01	6098208200062	MIAMI COMPUTER SUPPLY
9	5142258629	2010-01-01	602608969534	FISHER SCI ATL
10	5142190439	2010-01-01	5509006296254	FEDEX SHP 12/23/09 AB#

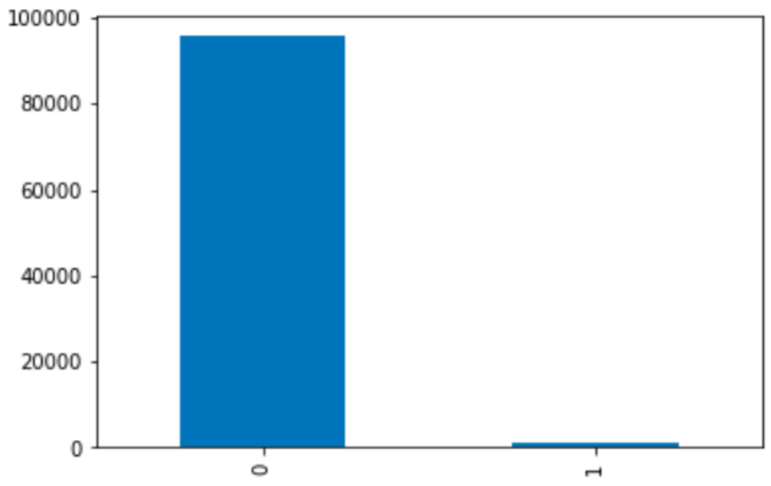
4. **Amount:** The amount of money each transaction spent. (Y axis was log transformed)

The "Amount" distribution plot shows a special non-smooth pattern. Generally speaking, as the "amount" value increases, the number of "amount" decreases. However, after the amount reaches 2500, there is a cliff-like drop-off from 1000 to 100 in the number of "amount" (Y-axis).



5. **Fraud:** Labels of each record to show whether it is a fraud (1) or not a fraud (0).

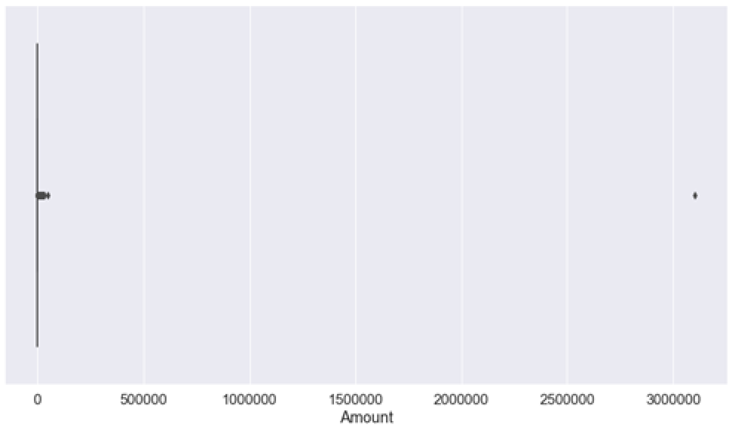
From this plot, we can see that our data is highly unbalanced. The number of records labeled "0" is 95694 (98.91% of total data) and the number of records labeled "1" is 1059 (1.09% of total data).



III. Data Cleaning

By cleaning data, we should consider exclusions, outliers, missing fields, and frivolous field values. In considering exclusions, outliers, and frivolous field values, we should not only browse the graph and value counts in a data quality report, but also understand the meaning of each term.

By graphing the boxplot below, it is obvious that there is an extreme value in the 'Amount' field. Most of the amount value is less than 250,000, but only one of the values is larger than 3,000,000. This big difference between the values may have a critical influence on model prediction, so we should remove this outlier.



In the field of "Transtype", there are four kinds of transaction types, including "P", "A", "D" and "Y". "P" is a prenote, which Refers to a Prenotification payment (with no monetary amount transferred) that you can use to verify bank account validity. "D" is delayed capture. Delayed capture marks the funds set aside by an approved Authorization transaction for capture during the next settlement period. "A" represents authorization, which requests to charge a cardholder. Since it is the problem to test fraud in credit card transactions, we should focus on specific transaction types that the money has not transferred, and the type used to verify bank account validity. Prenote is the only type we should include in our data, thus we should remove other data except "P" type transactions.

We then should fill the missing field and refill the zero value with more meaningful value except for the dependent variable. There are only three fields that have missing values, including Merchnum, Merch state, and Merch zip. All the fields in our dataset do not have zero value.

1. Filling Merchnum

We first aggregate merchant numbers by merchant description and fill in the most common merchant number for each merchant description, because merchant description is the field most correlated with merchant numbers. Then we aggregate it by merchant zip and fill in the mode of merchant number for each merchant zip code. Lastly, we then aggregate it by merchant state and fill in the most common merchant number for each merchant state. After doing these, we have filled in all the missing merchant numbers.

2. Filling Merch state

First, we aggregate merchant state by merchant zip code and fill in the most common merchant state for each zip code. Second, we then aggregate by merchant number and fill in the most common merchant state for each merchant number. Lastly, we aggregate it by merchant description and fill in the most common merchant state for each merchant description. After doing these, we have filled in most of the missing values. However, there are still a few records containing missing merchant states. Consequently, we decided to fill "TN", which is the most common value of the merchant state across the whole dataset.

3. Filling Merch zip

We first group merchant zip by merchant number and merchant state at the same time and fill in the most common merchant zip for each merchant number and merchant state. We then group it by merchant description and merchant state, respectively, and fill in the mode of merchant zip for each merchant description and merchant state. After that, we still notice that there are some records with missing merchant zip code within the dataset. We decided to fill in 38118, which is the most common merchant zip code across the whole dataset.

IV. Candidate Variables

To determine whether a credit transaction is fraudulent or not, complicated factors about transaction details need to be considered and new variables need to be created for further feature selection and model building process. For this reason, the following 5 categories of variables are created: Amount Variables, Frequency Variables, Days-since Variables, Velocity Change Variables, and Risk Table Variables. 7 entities are created for Amount Variables (4.1), Frequency Variables (4.2) and Days-since Variables (4.3): card, merchant, card at this merchant (card_merch), card in this zip code (card_zip), card in this state (card_state), merchant in this zip code (merch_zip), and merchant in this state (merch_state).

4.1 Amount Variables

A total of 8 (amount information) * 7 (entities) * 6 (day) = 336 Amount Variables are created at this step. Each variable represents the transaction amount information at each entity over one of the 6 time periods.

1. Amount information on day 0 (today) as well as over the past 1 (yesterday), 3, 7 (1 week), 14 (approximately 1/2 month), and 30 days (approximately 1 month) are created for further feature selection process. The choice of days is based on domain knowledge.
2. Transaction amounts for each credit card and each merchant are included. Besides these 2 entities which are in the original data set, 5 other entities are created for a more detailed amount of information on cards and merchants. 3 additional entities that are related to credit cards are: Card at this merchant (card_merch), Card in this zip code (card_merch) and Card in this state (card_state). 2 additional entities that are related to merchants are: Merchant in this zip code (merch_zip) and Merchant in this state (merch_state). Entities are created by concatenating the strings together. For instance, card_merch is created by concatenating the unique card number with the merchant number.
3. Average/Maximum/Median/Total transaction amount is created over the past days at all 7 entities. In addition, the actual transaction amount is divided by the above four values so that a clear comparison of how the actual amount value is compared to these values can be seen.

We created the Amount Variables because possible credit card frauds occur when the transaction amounts of an account have a significant difference from its past amount, from the average/max/median amount or from the amount in a certain group. The Amount of Variables allows us to find anomalies in transactions.

4.2 Frequency Variables

A total of 7 (entities) * 6 (day) = 42 Frequency Variables are created at this step. For the same reason as above, 7 entities and 6 different time periods are used at this step. Each variable represents the number of transactions with each entity over one of the 6 time periods. Similarly, credit card frauds occur when the transaction frequencies of an account have a significant difference from its past frequencies, or from the frequencies in a certain group.

4.3 Days-since Variables

A total of 7 variables are created at this step. Each variable represents the current date minus date of the most recent transactions for each entity. Similarly, credit card frauds occur when the time difference is unusual compared to the value within the same group.

4.4 Velocity Change Variables

A total of 2*2*2*3 = 24 variables are created at this step.

1. The number of transactions and the transaction amount with the same card/merchant over the past 0 day (today) and 1 day (yesterday) is created.
2. The average daily value of transactions and the transaction amount with the same card/merchant over the past 7, 14, and 30 days is created.
3. Divide Step 1 by Step 2 to get an idea of how today and yesterday's values are compared to values over the past 7, 14, and 30 days.

Similarly, credit card frauds occur when the ratio is too big or too small compared to the value within the same group.

4.5 Risk Table Variables

2 Risk Table Variables are created at this step, one for Weekdays and one for Merchant State, bringing the total number of variables to 411. These two risk table variables are created for the purpose of comparison among the average fraud rate in each group. Data from January to October are treated as the train_test data set, and data from November to December are treated as the validation data set. The average fraud rate is first calculated on the train_test data set by different Weekdays/Merch State, then mapped to the validation data set. A smoothing formula was applied so that even if the number of samples in a certain group is not statistically sufficient, the average fraud rate is still meaningful given that it's been smoothed by the average fraud rate of the whole data set.

V. Feature Selection Process

Feature selection is the process of selecting a subset of relevant features for use in model construction. It serves the purpose of avoiding the curse of dimensionality, enhancing generalization by reducing overfitting, and reducing training times. The central premise when using feature selection is that the data contains some features that are either redundant or irrelevant and can thus be removed without incurring much loss of information. There are a couple of methods that have been utilized to perform feature selection.

The first feature selection method we used is filter. This method is often univariate and considers the feature independently, or with regard to the dependent variables. We used both Kolmogorov-Smirnov (KS) and Fraud Detection Rate, (FDR) for the filter-feature selection. KS is based on calculating the highest difference between cumulative distribution between good and bad records. The more different they are the better the variables for separating, and thus the more important the variables. We calculated the KS score for all the variables and labeled them by their ranking number. FDR is what % of all the frauds are caught at a particular examination cutoff location. In this case, we take 3% as our cutoff. We first sort the data by a certain variable in ascending order, and calculate FDR for both top 3% and bottom 3% of the data, then label them by the ranking number of the maximum score of the two. In the end, we calculate the average ranking by (rank_ks + rank_FDR)/2 for each variable, sort them again, then keep the top 80 most valuable variables.

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

The Second feature selection method we used is wrapper. This method evaluates a specific machine-learning algorithm to find optimal features. It follows a greedy search approach by evaluating all the possible combinations of features against the evaluation criteria. We did recursive feature elimination and cross-validation selection for the best number of features. It begins by building a logistic regression model on the entire set of predictors and computing an importance score for each predictor. The least important predictors are then removed, the model is re-built, and importance scores are computed again. We first reduced these 80 variables to about 50 variables, then ran the wrapper again, and reduced them to about 30 variables.

Below are the final 30 variables :

cardnum_count_past_1, cardnum_total_14, cardnum_total_30, card_merch_med_7, card_state_count_past_3, card_state_total_3, card_zip_avg_7, state_risk, card_zip_med_14, card_merch_avg_30, card_merch_max_30, card_zip_total_30, card_state_med_0, card_merch_total_7, card_zip_max_1, card_merch_total_7, merch_state_total_1, merchnum_total_14, merch_state_total_7, card_zip_total_14, merchnum_med_3, merch_zip_avg_1, merch_zip_max_3, card_merch_max_3, merch_zip_actual/med_30, merchnum_actual/med_30, card_merch_total_3, card_merch_actual/total_3, cardnum_amt1/30, card_zip_max_30.

VI. Model Algorithms

Basically, we used 10-fold cross-validation on the train data set by trying different parameters on all of our models. After training the model, we applied the model on the train, test, and OOT data set to predict the probability of fraud. Then, we sorted the result by the predicted probability. Our metric to evaluate the model performance is the fraud detection rate (FDR) at a 3% population, the same as the metric we used for feature selection. So, we sum the real frauds caught at the top 3% population and divided by the total number of frauds in that data set (training, testing, OOT). We took the average of the 10 times running as the result of the model's performance.

The models we run are Logistic Regression, Random Forest, XGBoosted Tree, and Deep Learning.

First, we built a logistic regression model as our baseline model using the 30 variables. The algorithm behind it is that it uses the linear regression of the features to predict the logit.

$$log\frac{P(Y|X_1,X_2,X_3)}{P(\bar{Y}|X_1,X_2,X_3)} = w_1x_1 + w_2x_2 + w_3x_3 + \ldots + b$$

We tried the defaulted L2 regularization which would help avoid overfitting by average the effect of the variables, parameters of c=1,10,100,1000, and with/without sample balance. The smaller c means a larger lambda, which means stronger regularization on features.

Second, we built a random forest model to improve predictive power. The algorithm behind the random forest model is that it tries to decrease the impurity in the classified groups step by step by using different features. It randomly picks k features from the dataset, repeats the process, and thus builds N trees. Each tree makes predictions on the final probability individually and then averages the result. So, it is a kind of embed model which usually has better predictive power than logistic regression.

Third, we built the xgboost tree model. The algorithm is similar to the random forest (picking features to decrease the impurity) but the difference is that the xgboost tree doesn't build individual trees but a lot of trees on the prior classified samples. So it builds trees step by step and makes predictions of probability on each leaf. Finally, it sums up all the probabilities on the leaves in the same branch and that is the final probability. Thus it builds weaker trees than the random forest.

We also built deep learning models to capture non-linear relations. Deep Learning is a method that uses a set of connected nodes and mimics the human brain function, which is also known as Deep Neural Learning or Deep Neural Network. A Deep Neural Network consists of three layers of input, hidden, and output. Each of the input nodes (neurons) is connected to the nodes in the hidden layer by a communication link with a weight, and the output nodes indicate the classifications.

We used MLP classifiers to implement the Deep Learning model. By tuning different hyperparameters including the number of layers, the number of nodes in each layer, and activation function, we gained the following sets of performance metrics (FDRs) on training, test, and OOT sets respectively.

Table of high-level results for each algorithm (FDR for training, testing, oot):

model	parameter	class_weight	train	test	oot
logistic regression	c=1000	balanced	0.752219	0.747155	0.52067
	c=100	balanced	0.753316	0.739951	0.52905
	c=10	balanced	0.750796	0.757539	0.512849
	c=1	balanced	0.7595	0.750854	0.498324
	c=10	not balanced	0.735211	0.724009	0.349162

model	train	test	oot	n_estimators	max_depth	max_features	class_weight
random forest	0.928211	0.869092	0.6201119	100	10	20	not balanced
	0.988457	0.889856	0.5860336	60	15	30	not balanced
	1	0.920194	0.6189945	100	20	20	not balanced
	1.00000	0.89185	0.54357	150	30	20	not balanced
	0.86030	0.81784	0.55083	200	8	5	not balanced
	0.95197	0.81579	0.30055	200	8	5	balanced

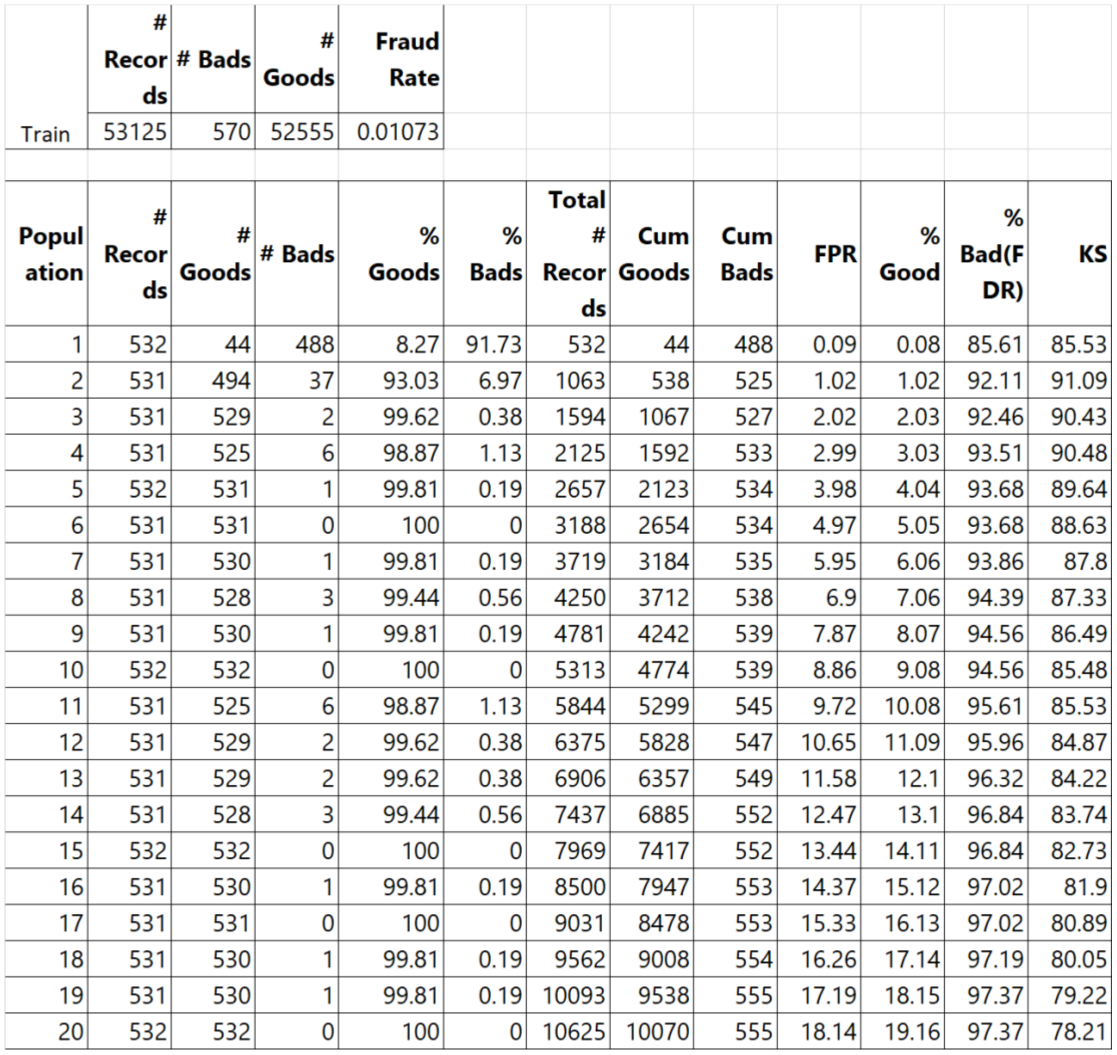
model	train	test	oot	n_estimators	max_depth	learning_rate
boosted tree	1	0.9454014	0.5664805	600	5	0.1
	1	0.9299308	0.4547486	500	3	1
	0.9944229	0.9207777	0.6089386	1000	6	0.01
	0.76427	0.71066	0.54860	800	4	0.001
	0.99884	0.89176	0.51508	800	3	0.1

model	train	test	oot	number of layers	nodes on layer1	nodes on layer2	activation
deep learning	0.67485	0.67639	0.38547	1	4	0	default
	0.70872	0.71778	0.51731	1	8	0	default
	0.72554	0.70814	0.51061	1	12	0	default
	0.76697	0.75434	0.54581	2	8	8	default
	0.68725	0.67655	0.41731	2	8	8	logistic

VII. Results

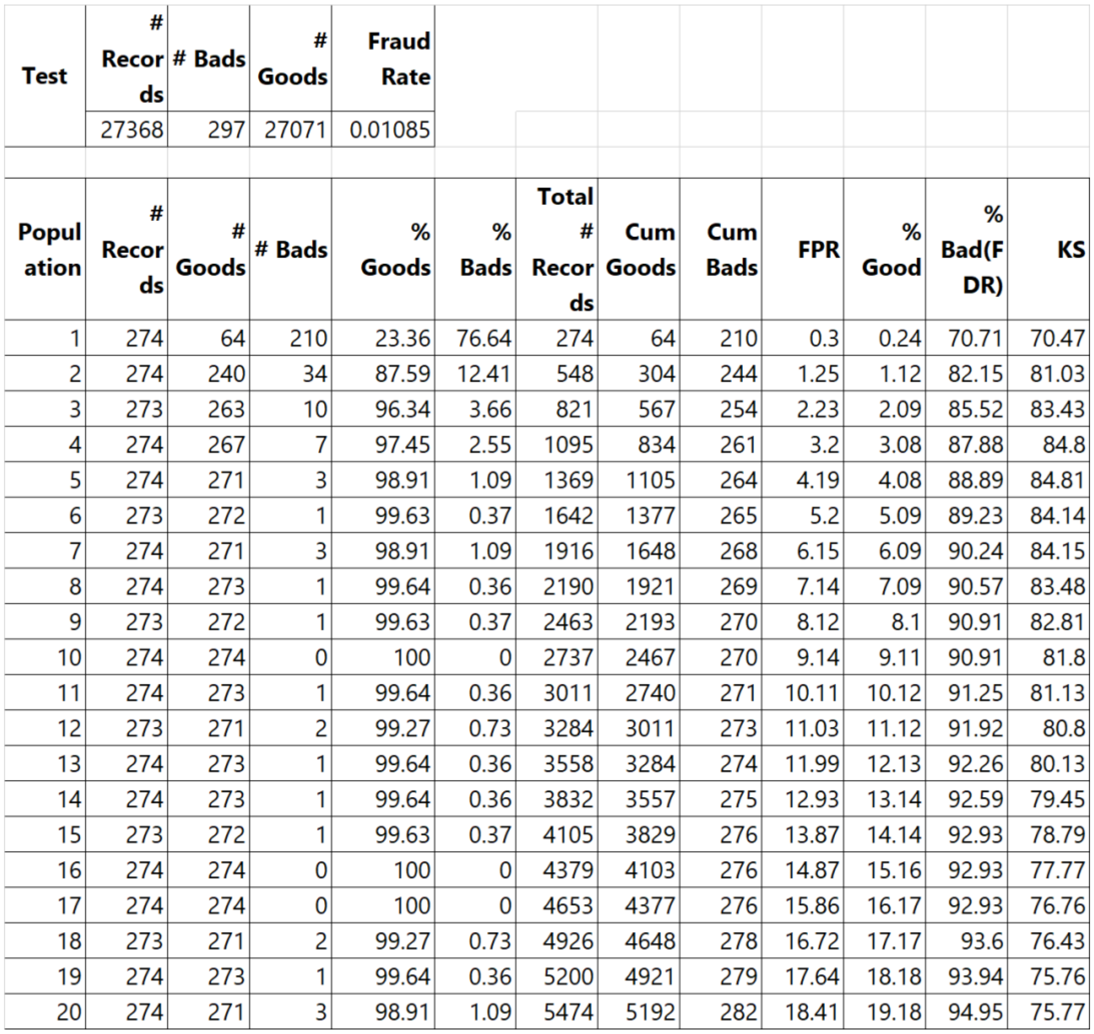
So, our final algorithm is the random forest model because it has the highest FDR on the OOT dataset and didn't show overfitting. The result showed that our model is good. Most of the bad ones are concentrated at the top either in train, test, or OOT. At a population of 3%, the fraud detection rate is 93% in the training set, 87% in the testing set, and 62% in OOT.

Three tables for training, testing, oot populations:



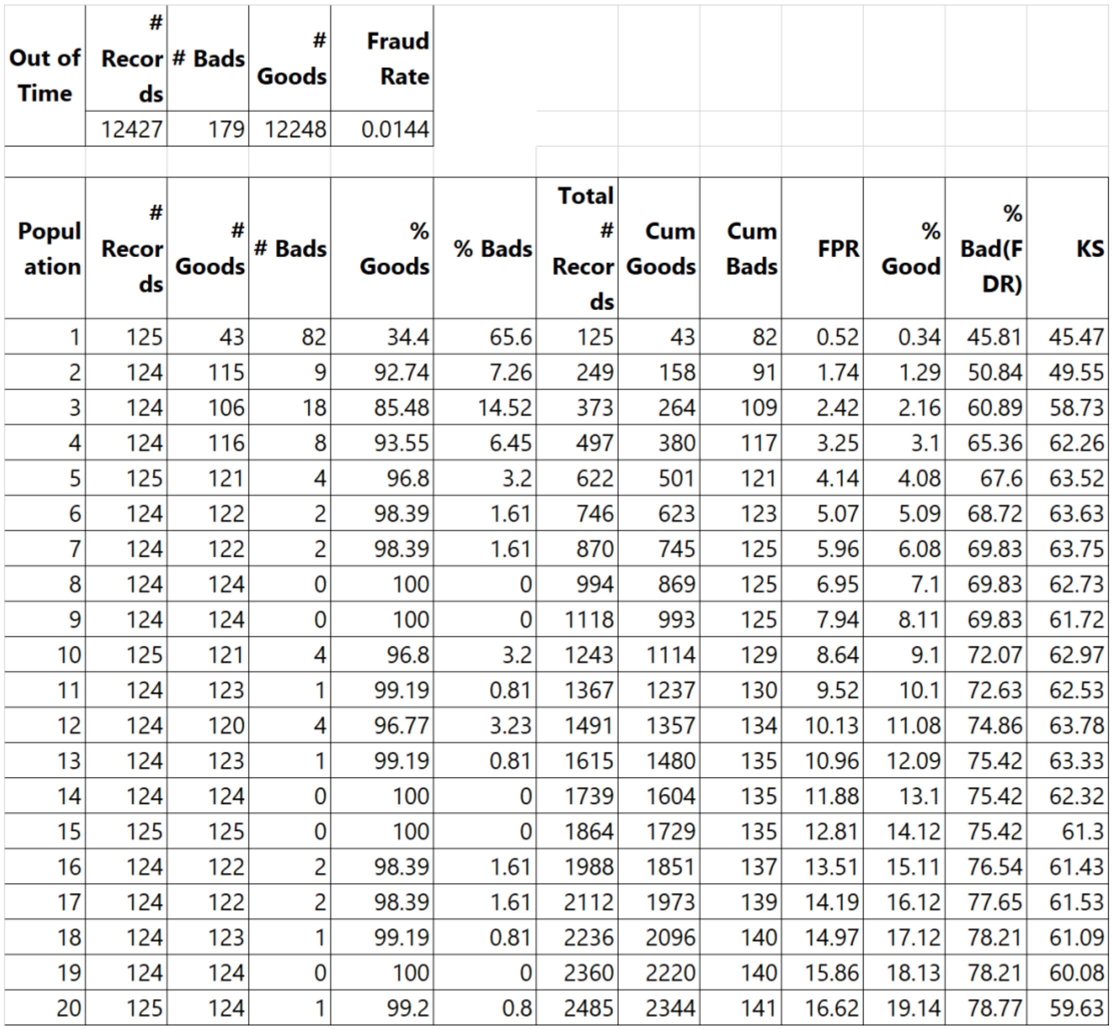
	# Records	# Bads	# Goods	Fraud Rate
Train	53125	570	52555	0.01073

Population	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cum Goods	Cum Bads	FPR	% Good	% Bad(FDR)	KS
1	532	44	488	8.27	91.73	532	44	488	0.09	0.08	85.61	85.53
2	531	494	37	93.03	6.97	1063	538	525	1.02	1.02	92.11	91.09
3	531	529	2	99.62	0.38	1594	1067	527	2.02	2.03	92.46	90.43
4	531	525	6	98.87	1.13	2125	1592	533	2.99	3.03	93.51	90.48
5	532	531	1	99.81	0.19	2657	2123	534	3.98	4.04	93.68	89.64
6	531	531	0	100	0	3188	2654	534	4.97	5.05	93.68	88.63
7	531	530	1	99.81	0.19	3719	3184	535	5.95	6.06	93.86	87.8
8	531	528	3	99.44	0.56	4250	3712	538	6.9	7.06	94.39	87.33
9	531	530	1	99.81	0.19	4781	4242	539	7.87	8.07	94.56	86.49
10	532	532	0	100	0	5313	4774	539	8.86	9.08	94.56	85.48
11	531	525	6	98.87	1.13	5844	5299	545	9.72	10.08	95.61	85.53
12	531	529	2	99.62	0.38	6375	5828	547	10.65	11.09	95.96	84.87
13	531	529	2	99.62	0.38	6906	6357	549	11.58	12.1	96.32	84.22
14	531	528	3	99.44	0.56	7437	6885	552	12.47	13.1	96.84	83.74
15	532	532	0	100	0	7969	7417	552	13.44	14.11	96.84	82.73
16	531	530	1	99.81	0.19	8500	7947	553	14.37	15.12	97.02	81.9
17	531	531	0	100	0	9031	8478	553	15.33	16.13	97.02	80.89
18	531	530	1	99.81	0.19	9562	9008	554	16.26	17.14	97.19	80.05
19	531	530	1	99.81	0.19	10093	9538	555	17.19	18.15	97.37	79.22
20	532	532	0	100	0	10625	10070	555	18.14	19.16	97.37	78.21



Test	# Records	# Bads	# Goods	Fraud Rate
	27368	297	27071	0.01085

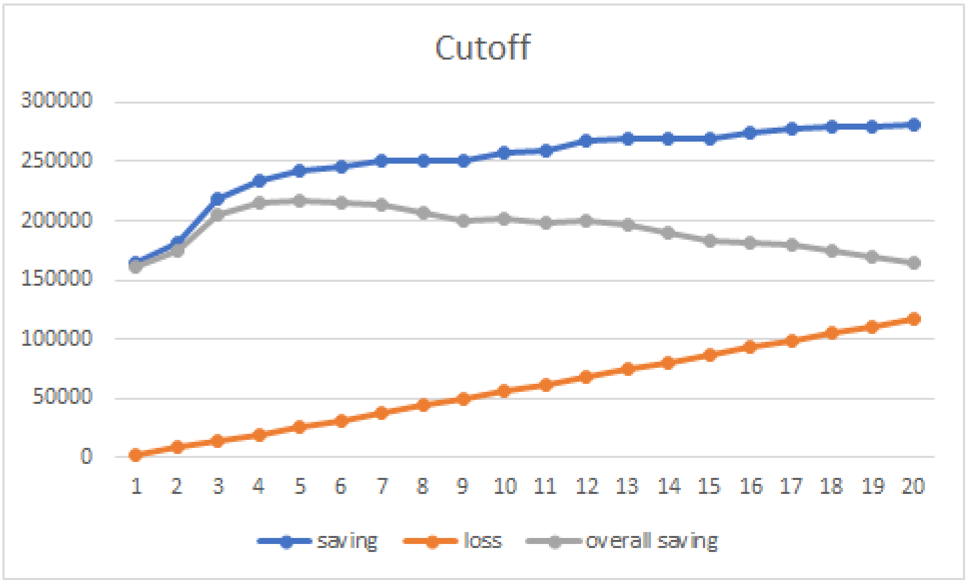
Population	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cum Goods	Cum Bads	FPR	% Good	% Bad(FDR)	KS
1	274	64	210	23.36	76.64	274	64	210	0.3	0.24	70.71	70.47
2	274	240	34	87.59	12.41	548	304	244	1.25	1.12	82.15	81.03
3	273	263	10	96.34	3.66	821	567	254	2.23	2.09	85.52	83.43
4	274	267	7	97.45	2.55	1095	834	261	3.2	3.08	87.88	84.8
5	274	271	3	98.91	1.09	1369	1105	264	4.19	4.08	88.89	84.81
6	273	272	1	99.63	0.37	1642	1377	265	5.2	5.09	89.23	84.14
7	274	271	3	98.91	1.09	1916	1648	268	6.15	6.09	90.24	84.15
8	274	273	1	99.64	0.36	2190	1921	269	7.14	7.09	90.57	83.48
9	273	272	1	99.63	0.37	2463	2193	270	8.12	8.1	90.91	82.81
10	274	274	0	100	0	2737	2467	270	9.14	9.11	90.91	81.8
11	274	273	1	99.64	0.36	3011	2740	271	10.11	10.12	91.25	81.13
12	273	271	2	99.27	0.73	3284	3011	273	11.03	11.12	91.92	80.8
13	274	273	1	99.64	0.36	3558	3284	274	11.99	12.13	92.26	80.13
14	274	273	1	99.64	0.36	3832	3557	275	12.93	13.14	92.59	79.45
15	273	272	1	99.63	0.37	4105	3829	276	13.87	14.14	92.93	78.79
16	274	274	0	100	0	4379	4103	276	14.87	15.16	92.93	77.77
17	274	274	0	100	0	4653	4377	276	15.86	16.17	92.93	76.76
18	273	271	2	99.27	0.73	4926	4648	278	16.72	17.17	93.6	76.43
19	274	273	1	99.64	0.36	5200	4921	279	17.64	18.18	93.94	75.76
20	274	271	3	98.91	1.09	5474	5192	282	18.41	19.18	94.95	75.77



Out of Time	# Records	# Bads	# Goods	Fraud Rate
	12427	179	12248	0.0144

Population	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cum Goods	Cum Bads	FPR	% Good	% Bad(FDR)	KS
1	125	43	82	34.4	65.6	125	43	82	0.52	0.34	45.81	45.47
2	124	115	9	92.74	7.26	249	158	91	1.74	1.29	50.84	49.55
3	124	106	18	85.48	14.52	373	264	109	2.42	2.16	60.89	58.73
4	124	116	8	93.55	6.45	497	380	117	3.25	3.1	65.36	62.26
5	125	121	4	96.8	3.2	622	501	121	4.14	4.08	67.6	63.52
6	124	122	2	98.39	1.61	746	623	123	5.07	5.09	68.72	63.63
7	124	122	2	98.39	1.61	870	745	125	5.96	6.08	69.83	63.75
8	124	124	0	100	0	994	869	125	6.95	7.1	69.83	62.73
9	124	124	0	100	0	1118	993	125	7.94	8.11	69.83	61.72
10	125	121	4	96.8	3.2	1243	1114	129	8.64	9.1	72.07	62.97
11	124	123	1	99.19	0.81	1367	1237	130	9.52	10.1	72.63	62.53
12	124	120	4	96.77	3.23	1491	1357	134	10.13	11.08	74.86	63.78
13	124	123	1	99.19	0.81	1615	1480	135	10.96	12.09	75.42	63.33
14	124	124	0	100	0	1739	1604	135	11.88	13.1	75.42	62.32
15	125	125	0	100	0	1864	1729	135	12.81	14.12	75.42	61.3
16	124	122	2	98.39	1.61	1988	1851	137	13.51	15.11	76.54	61.43
17	124	122	2	98.39	1.61	2112	1973	139	14.19	16.12	77.65	61.53
18	124	123	1	99.19	0.81	2236	2096	140	14.97	17.12	78.21	61.09
19	124	124	0	100	0	2360	2220	140	15.86	18.13	78.21	60.08
20	125	124	1	99.2	0.8	2485	2344	141	16.62	19.14	78.77	59.63

Assuming $2000 gain for every fraud that's caught and $50 loss for every false positive (good that's flagged as a bad), we calculated the fraud savings (blue curve), lost sales(orange), and overall savings (grey).



By calculating the overall saving, we could make a recommendation for where the client should set a score cutoff

We could see that 4% and 5% don't make a large difference in the overall saving. To satisfy, we choose 4% as the cutoff, which will generate an overall saving of $215000.

VIII. Conclusions

In this project, we used the historical credit card transaction records from 2010-01-01 to 2010-12-31 to build supervised models and detect transaction fraud. We first briefly analyzed the data and its distribution. Based on those basic analyses, we removed outliers with their amounts over 3,000,000 and transaction types other than “P” (prenote). Then, we filled the missing fields and zero values with the most common values by different groups. After the dataset was ready, we created 410 candidate variables that could describe the volume, frequency, velocity change, and time-span for each card and merchant. In order to avoid the curse of dimensionality, we conducted a feature selection process. Before this step, we separated the whole population into modeling and out-of-time (OOT) data sets and only used the modeling dataset to do the feature selection. In this step, we applied both Kolmogorov-Smirnov (KS) and Fraud Detection Rate (FDR) for the filter-feature selection that reduces variable numbers to 80. Then, we used the wrapper method that finally reduced variables to 30 that are important to build models. After getting the expert variables, we built logistic regression as the baseline. We applied 10-fold cross-validation on the training set on several different logistic models and tuned hyperparameters. Then, in order to avoid overfitting, we applied the predicted model on the training, testing, and OOT data to predict the probability of fraud in each step. We tried the same thing for the random forest, bootstrap, and deep learning (neural network). To evaluate each model’s performance, we calculated the fraud detection rate (FDR) at a 3% population. Based on our tuning process, we found that a random forest model with 100 estimators, 10 maximum depth, no balanced weight, and maximum feature equals 20 had the largest FDR on the OOT data set but a very small difference between the training and testing set. We Assume $2000 gain for every fraud that’s caught and $50 loss for every false positive to plot the relationship between the percentage of the OOT population and overall savings (fraud saving – lost sales). We found that it is most profitable to see all the transactions belonging to the top 4% score as fraudulent and it could generate a net saving for $215,000.

Admittedly, there are still several parts that could be improved. First, after discussing with experts in credit card transactions, we could create more candidate variables that might be more robust to explain the differences between fraudulent and normal transactions. Second, even though we already conduct many machine-learning models above, it might still have more robust models that we did not discover. We could return our recent best to our customers/users. Based on the users’ feedback, we could furtherly tune the parameters to improve our best model.

Appendix

Appendix A

Section I: General description of data

Dataset Name: Credit Card Transactions Data

Data Description: The dataset shows real credit card transactions in various merchants in the chronological order, which from January 2010 to December 2010. It not only includes the basic transaction information like merchants' names, locations, transaction types, and amount but also labels whether a record is a fraud or not.

Time period: 2010

Number of fields: 10

Number of records: 96,753

Section II: Summary Table

1. **Summary of all the fields in the dataset**

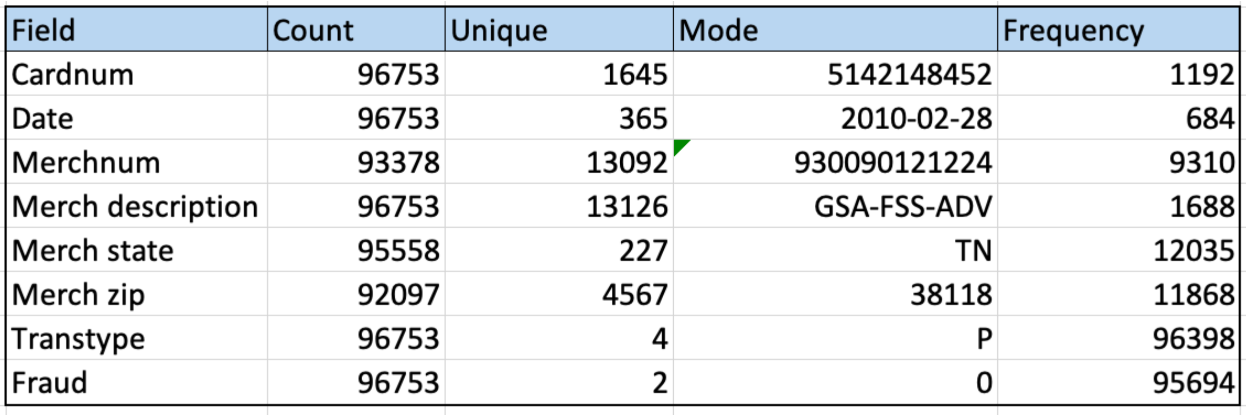
(Noted: "NA" was not counted to unique value)

No.	Name	Date Type	Number Populated	% Populated	# of Unique Value	# Zeros
1	Recnum	int64	96753	100.00%	96753	0
2	Cardnum	int64	96753	100.00%	1645	0
3	Date	datetime64	96753	100.00%	365	0
4	Merchnum	object	93378	96.51%	13092	231
5	Merch description	object	96753	100.00%	13126	0
6	Merch state	object	95558	98.76%	227	0
7	Merch zip	float64	92097	95.19%	4567	0
8	Transtype	object	96753	100.00%	4	0
9	Amount	float64	96753	100.00%	34909	0
10	Fraud	int64	96753	100.00%	2	95694

2. **Summary of numeric fields**

Field	count	mean	std	min	p25	median	p75	max
Recnum	96753	48377	27930.33	1	24189	48377	72565	96753
Amount	96753	427.89	10006.14	0.01	33.48	137.98	428.2	3102045.53

3. **Summary of categorical fields**



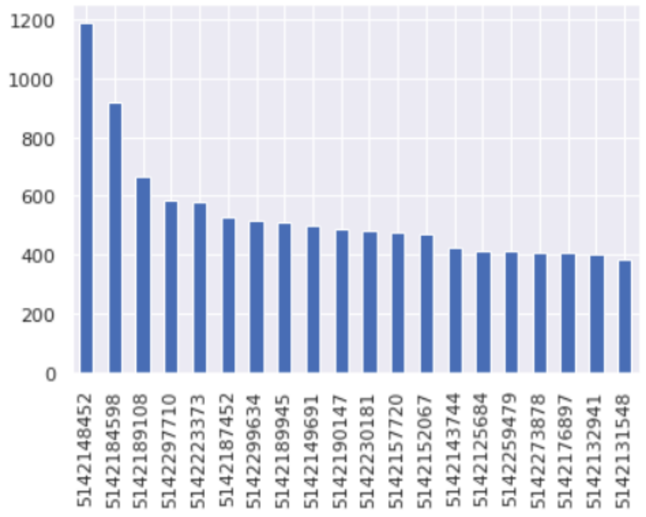
Field	Count	Unique	Mode	Frequency
Cardnum	96753	1645	5142148452	1192
Date	96753	365	2010-02-28	684
Merchnum	93378	13092	930090121224	9310
Merch description	96753	13126	GSA-FSS-ADV	1688
Merch state	95558	227	TN	12035
Merch zip	92097	4567	38118	11868
Transtype	96753	4	P	96398
Fraud	96753	2	0	95694

Section III: Detailed description of all the fields

In this section, every field will be explained in detail. For all the numeric fields, the section will show their distributions. In order to present a good pattern of distribution, the scales may be changed by logging the y-axis, removing outliers, and limiting ranges. For all the categorical fields,
the section will show histograms to count the frequency for each value.

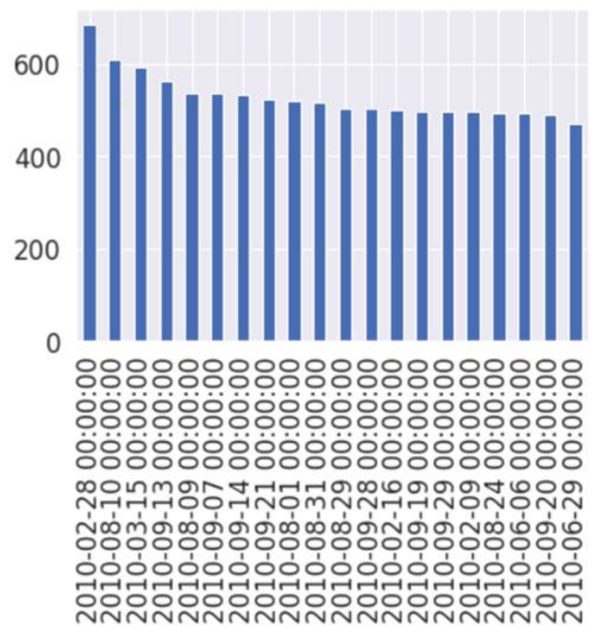
1. Recnum: Index for each record in the dataset, start from 1 to 96753.

2. Cardnum: The card number of the credit card that made transactions.

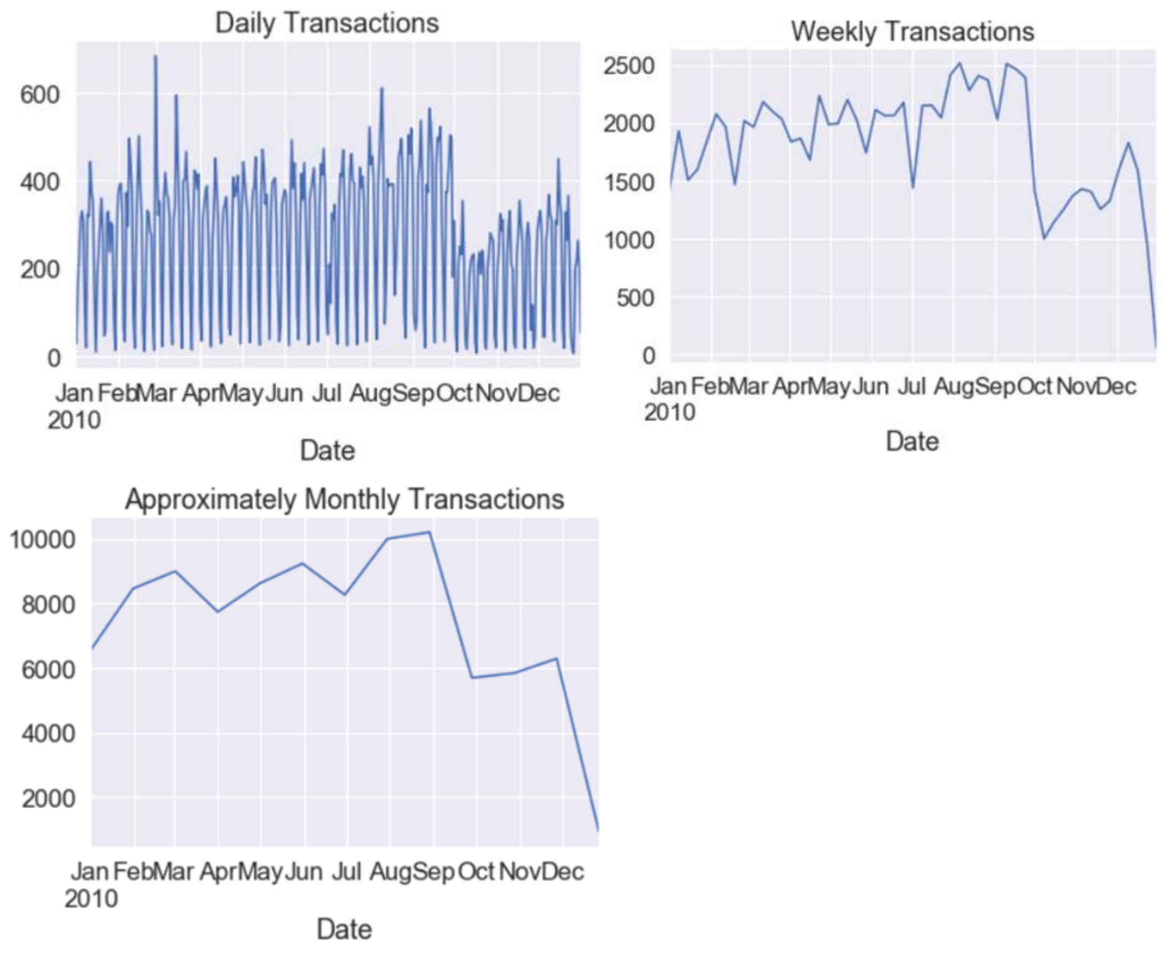


3. Date: The time when the transaction was made.

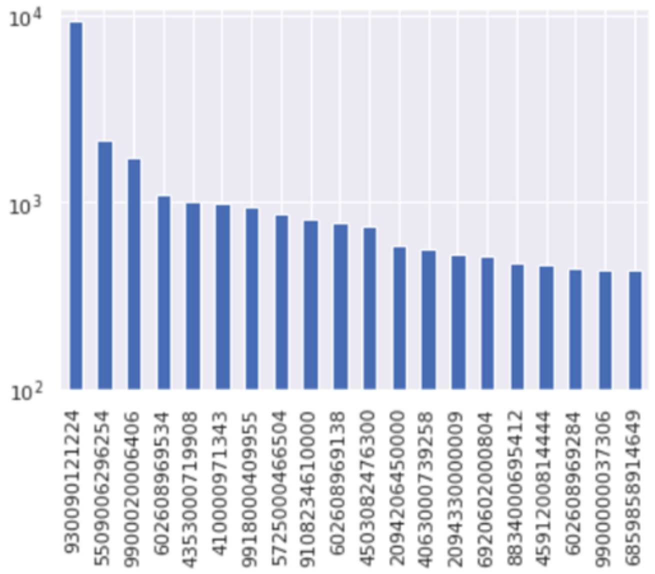
(1): Histogram of date frequency



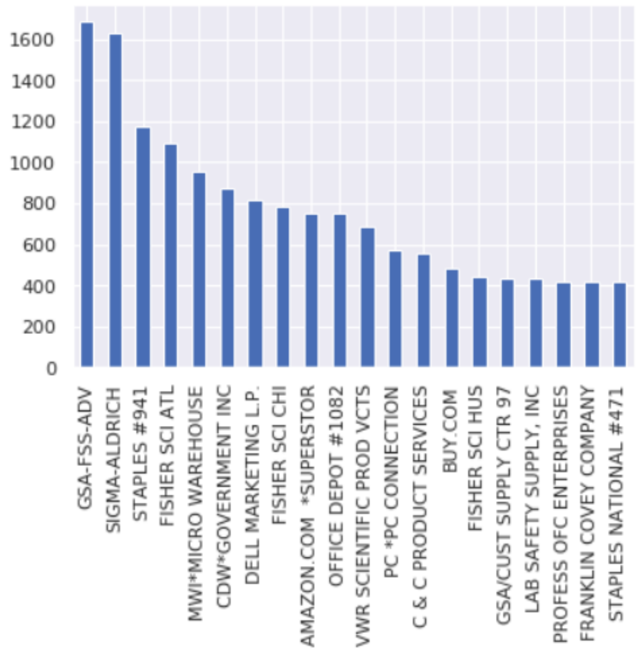
(2): Transactions count by chronological order (Daily, Weekly, Monthly)



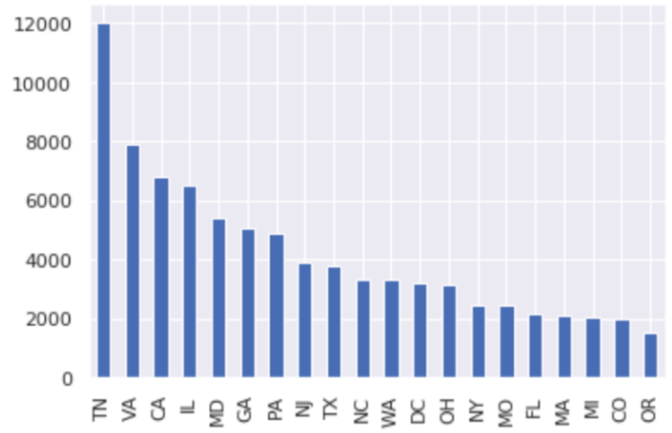
4. Merchnum: The number to identify different merchants. (Y axis was log transformed)



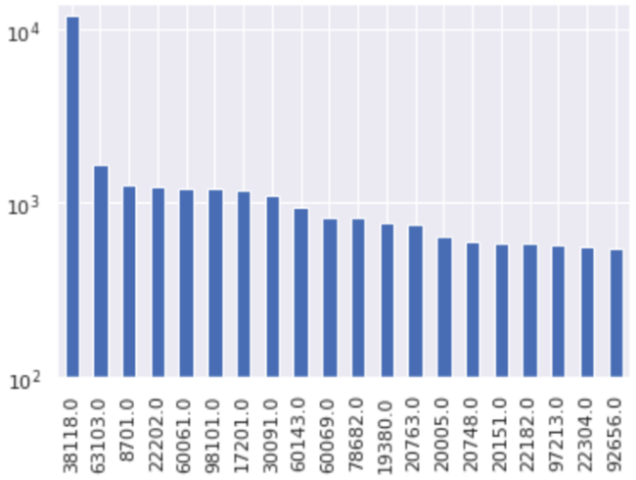
5. Merch description: The description of each merchant.



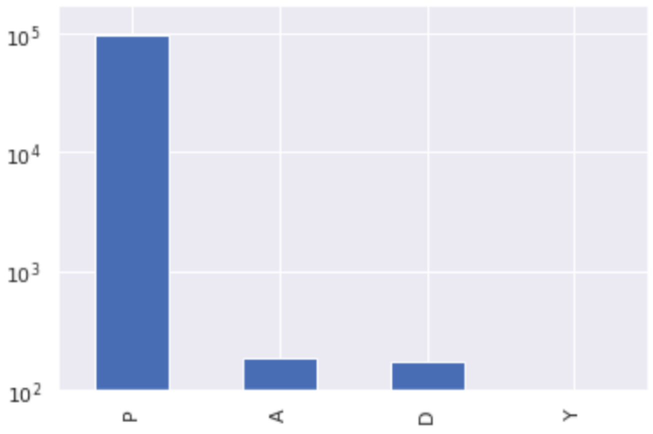
6. Merch State: The state that transactions happened.



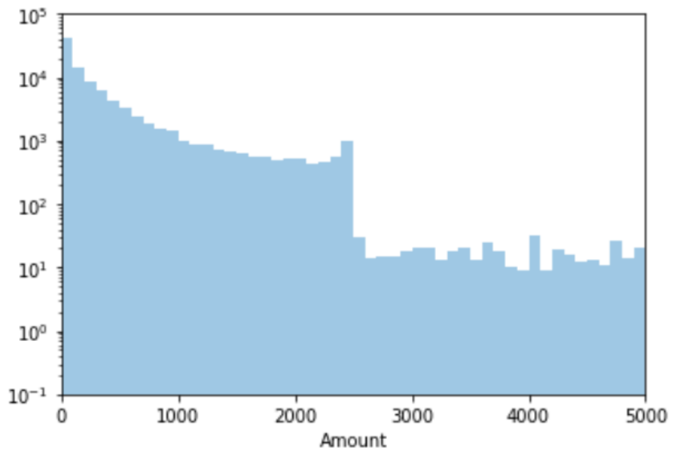
7. Merch zip: ZIP code information of each transaction. (Y axis was log transformed)



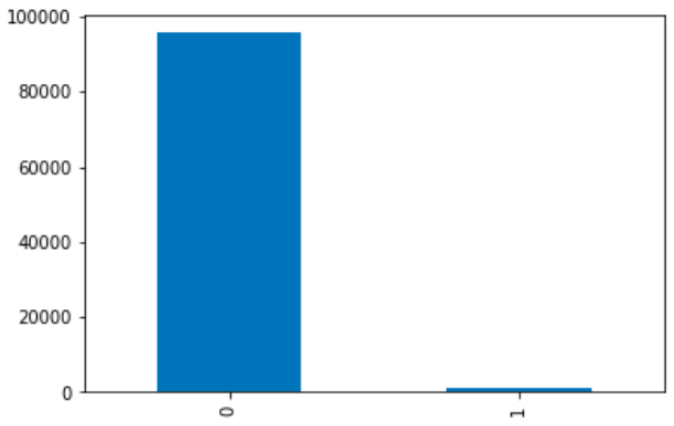
8. Transtype: Types of transactions. (Y axis was log transformed)



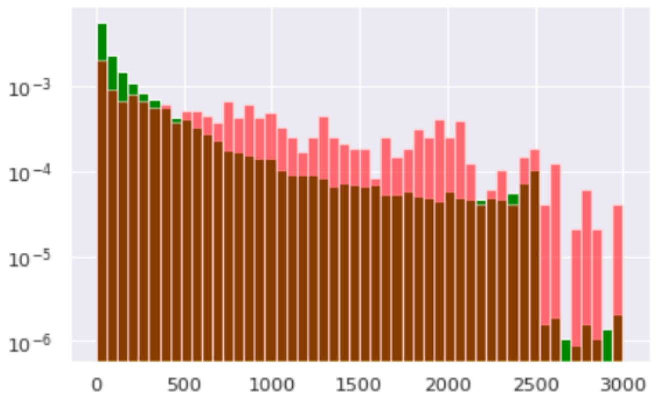
9. Amount: The amount of money each transaction spent. (Y axis was log transformed)



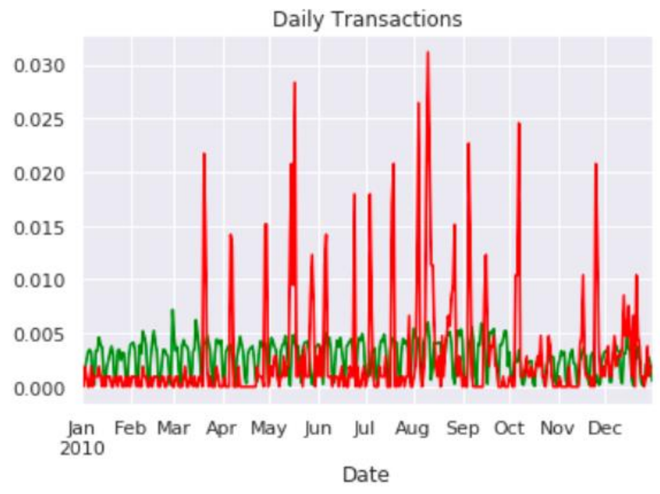
10. Fraud: Label each record whether it is a fraud (1) or not fraud (0).
(1) Fraud label count

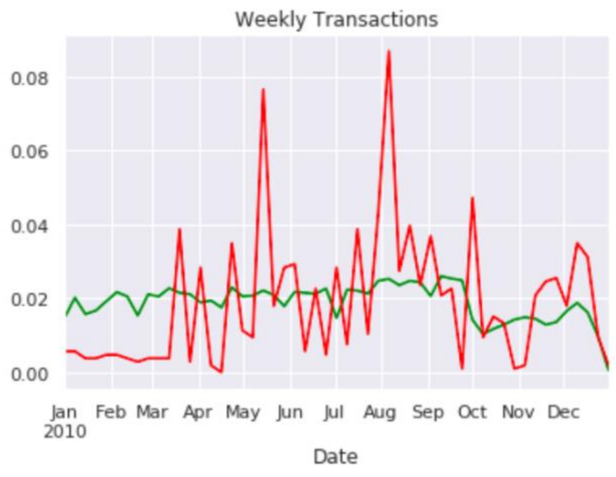


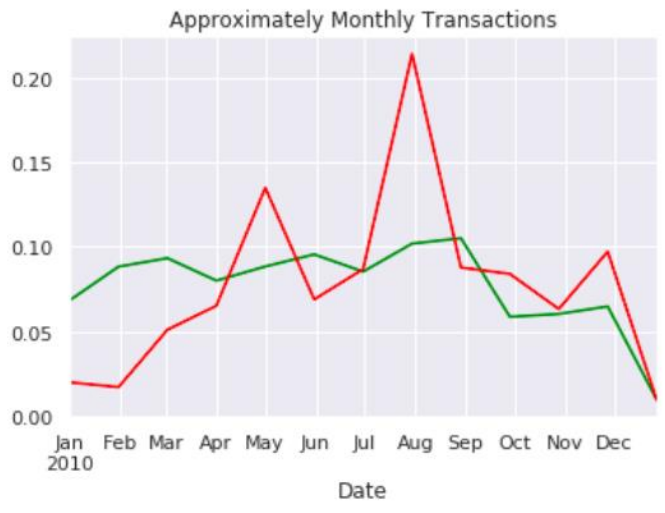
(2) Fraud label with "Amount" distribution (Green color represents no fraud, and red color represents fraud)



(3) Fraud label with chronological order transactions. (Daily, Weekly, and Monthly)







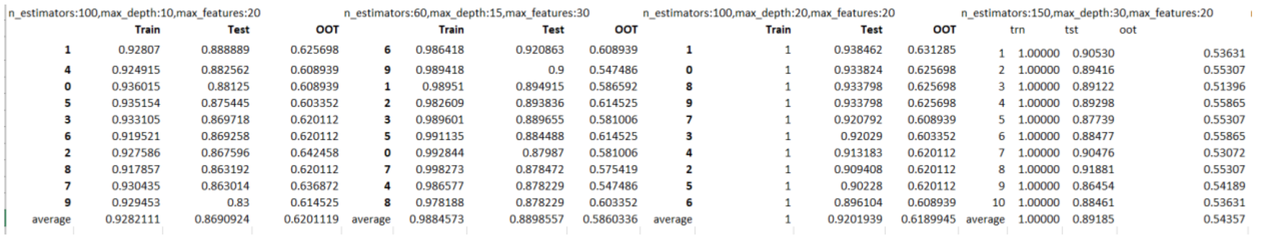
Appendix B

Part of the model results:

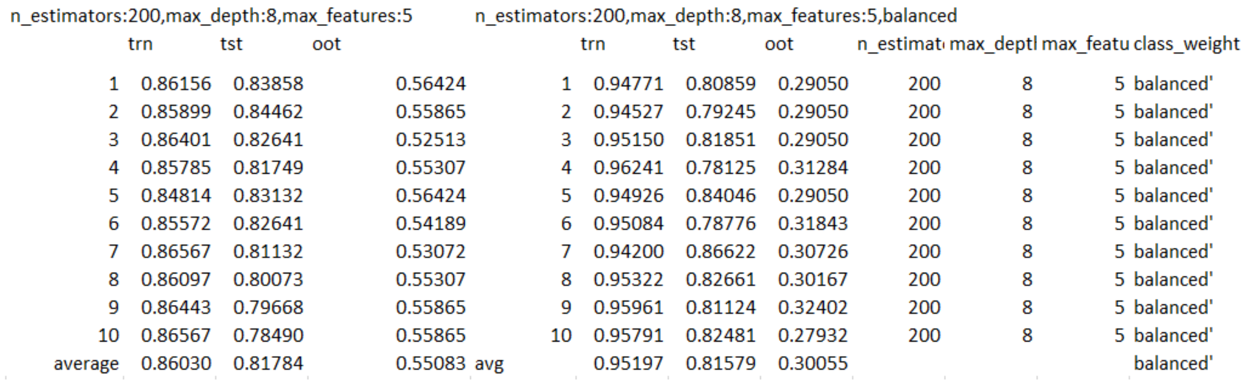
Logistic Regression:

c=1000					c=100				c=10				c=1				c=10,not balanced		
	Train	Test	OOT		Train	Test	OOT		Train	Test	OOT		Train	Test	OOT		Train	Test	OOT
9	0.73768	0.77926	0.49162	9	0.7487	0.762069	0.553073	6	0.73402	0.80903	0.53073	7	0.75086	0.77895	0.49162	5	0.72759	0.77181	0.34078
7	0.73274	0.77815	0.49721	0	0.75132	0.75	0.497207	8	0.72727	0.78983	0.51955	1	0.75267	0.77705	0.50279	0	0.71712	0.76603	0.35196
4	0.7465	0.76949	0.53631	2	0.76083	0.748276	0.541899	1	0.74242	0.77289	0.53073	0	0.74536	0.77372	0.46369	2	0.72931	0.75958	0.36313
3	0.74112	0.76087	0.53073	5	0.74414	0.746795	0.480447	4	0.74825	0.7661	0.54749	5	0.74955	0.76266	0.48045	6	0.74068	0.73026	0.36872
1	0.75478	0.75	0.5419	8	0.75231	0.745399	0.564246	3	0.74341	0.7651	0.49721	3	0.7401	0.75525	0.49162	7	0.72679	0.71336	0.30726
6	0.76357	0.73649	0.55307	3	0.74364	0.744479	0.530726	7	0.74271	0.75704	0.48603	2	0.75916	0.7551	0.50838	4	0.7425	0.71	0.36313
2	0.76329	0.72887	0.56425	6	0.76125	0.733564	0.547486	9	0.76191	0.74552	0.49162	8	0.76036	0.74039	0.54749	8	0.73554	0.70992	0.34637
5	0.75393	0.72789	0.48603	7	0.75685	0.731449	0.530726	5	0.7585	0.73377	0.4581	6	0.77005	0.72876	0.48603	3	0.73604	0.70652	0.34637
8	0.75704	0.72241	0.50279	1	0.7568	0.731183	0.519553	2	0.77304	0.71886	0.50279	9	0.78058	0.72026	0.49721	9	0.7452	0.69728	0.35196
0	0.77153	0.71812	0.50279	4	0.75732	0.706294	0.52514	0	0.77643	0.71724	0.56425	4	0.78631	0.71642	0.51397	1	0.75134	0.67533	0.35196
average	0.75222	0.74716	0.52067	average	0.75332	0.739951	0.52905	average	0.7508	0.75754	0.51285	average	0.7595	0.75085	0.49832	average	0.73521	0.72401	0.34916

Random Forest:

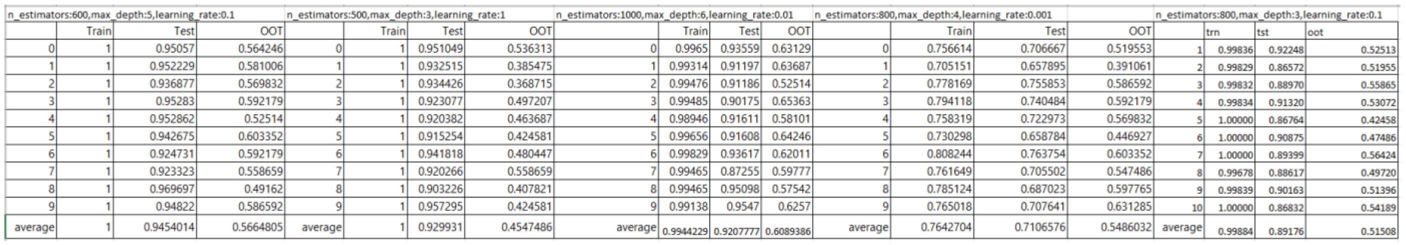


n_estimators:100,max_depth:10,max_features:20				n_estimators:60,max_depth:15,max_features:30				n_estimators:100,max_depth:20,max_features:20				n_estimators:150,max_depth:30,max_features:20			
	Train	Test	OOT		Train	Test	OOT		Train	Test	OOT		trn	tst	oot
1	0.92807	0.888889	0.625698	6	0.986418	0.920863	0.608939	1	1	0.938462	0.631285	1	1.00000	0.90530	0.53631
4	0.924915	0.882562	0.608939	9	0.989418	0.9	0.547486	0	1	0.933824	0.625698	2	1.00000	0.89416	0.55307
0	0.936015	0.88125	0.608939	1	0.98951	0.894915	0.586592	8	1	0.933798	0.625698	3	1.00000	0.89122	0.51396
5	0.935154	0.875445	0.603352	2	0.982609	0.893836	0.614525	9	1	0.933798	0.625698	4	1.00000	0.89298	0.55865
3	0.933105	0.869718	0.620112	3	0.989601	0.889655	0.581006	7	1	0.920792	0.608939	5	1.00000	0.87739	0.55307
6	0.919521	0.869258	0.620112	5	0.991135	0.884488	0.614525	3	1	0.92029	0.603352	6	1.00000	0.88477	0.55865
2	0.927586	0.867596	0.642458	0	0.992844	0.87987	0.581006	4	1	0.913183	0.620112	7	1.00000	0.90476	0.53072
8	0.917857	0.863192	0.620112	7	0.998273	0.878472	0.575419	2	1	0.909408	0.620112	8	1.00000	0.91881	0.55307
7	0.930435	0.863014	0.636872	4	0.986577	0.878229	0.547486	5	1	0.90228	0.620112	9	1.00000	0.86454	0.54189
9	0.929453	0.83	0.614525	8	0.978188	0.878229	0.603352	6	1	0.896104	0.608939	10	1.00000	0.88461	0.53631
average	0.9282111	0.8690924	0.6201119	average	0.9884573	0.8898557	0.5860336	average	1	0.9201939	0.6189945	average	1.00000	0.89185	0.54357



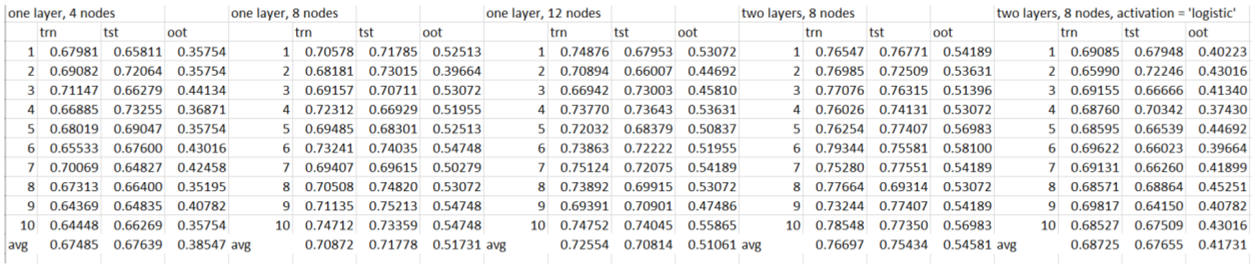
n_estimators:200,max_depth:8,max_features:5				n_estimators:200,max_depth:8,max_features:5,balanced							
	trn	tst	oot		trn	tst	oot	n_estimat	max_deptl	max_featu	class_weight
1	0.86156	0.83858	0.56424	1	0.94771	0.80859	0.29050	200	8	5	balanced'
2	0.85899	0.84462	0.55865	2	0.94527	0.79245	0.29050	200	8	5	balanced'
3	0.86401	0.82641	0.52513	3	0.95150	0.81851	0.29050	200	8	5	balanced'
4	0.85785	0.81749	0.55307	4	0.96241	0.78125	0.31284	200	8	5	balanced'
5	0.84814	0.83132	0.56424	5	0.94926	0.84046	0.29050	200	8	5	balanced'
6	0.85572	0.82641	0.54189	6	0.95084	0.78776	0.31843	200	8	5	balanced'
7	0.86567	0.81132	0.53072	7	0.94200	0.86622	0.30726	200	8	5	balanced'
8	0.86097	0.80073	0.55307	8	0.95322	0.82661	0.30167	200	8	5	balanced'
9	0.86443	0.79668	0.55865	9	0.95961	0.81124	0.32402	200	8	5	balanced'
10	0.86567	0.78490	0.55865	10	0.95791	0.82481	0.27932	200	8	5	balanced'
average	0.86030	0.81784	0.55083	avg	0.95197	0.81579	0.30055				balanced'

XGBoosted Tree:



n_estimators:600,max_depth:5,learning_rate:0.1				n_estimators:500,max_depth:3,learning_rate:1				n_estimators:1000,max_depth:6,learning_rate:0.01				n_estimators:800,max_depth:4,learning_rate:0.001				n_estimators:800,max_depth:3,learning_rate:0.1			
	Train	Test	OOT		Train	Test	OOT		Train	Test	OOT		Train	Test	OOT		trn	tst	oot
0	1	0.95057	0.564246	0	1	0.951049	0.536313	0	0.9965	0.93559	0.63129	0	0.756614	0.706667	0.519553	1	0.99836	0.92248	0.52513
1	1	0.952229	0.581006	1	1	0.932515	0.385475	1	0.99314	0.91197	0.63687	1	0.705151	0.657895	0.391061	2	0.99829	0.86572	0.51955
2	1	0.936877	0.569832	2	1	0.934426	0.368715	2	0.99476	0.91186	0.52514	2	0.778169	0.755853	0.586592	3	0.99832	0.88970	0.55865
3	1	0.95283	0.592179	3	1	0.923077	0.497207	3	0.99485	0.90175	0.65363	3	0.794118	0.740484	0.592179	4	0.99834	0.91320	0.53072
4	1	0.952862	0.52514	4	1	0.920382	0.463687	4	0.98946	0.91611	0.58101	4	0.758319	0.722973	0.569832	5	1.00000	0.86764	0.42458
5	1	0.942675	0.603352	5	1	0.915254	0.424581	5	0.99656	0.91608	0.64246	5	0.730298	0.658784	0.446927	6	1.00000	0.90875	0.47486
6	1	0.924731	0.592179	6	1	0.941818	0.480447	6	0.99829	0.93617	0.62011	6	0.808244	0.763754	0.603352	7	1.00000	0.89399	0.56424
7	1	0.923323	0.558659	7	1	0.920266	0.558659	7	0.99465	0.87255	0.59777	7	0.761649	0.705502	0.547486	8	0.99678	0.88617	0.49720
8	1	0.969697	0.49162	8	1	0.903226	0.407821	8	0.99465	0.95098	0.57542	8	0.785124	0.687023	0.597765	9	0.99839	0.90163	0.51396
9	1	0.94822	0.586592	9	1	0.957295	0.424581	9	0.99138	0.9547	0.6257	9	0.765018	0.707641	0.631285	10	1.00000	0.86832	0.54189
average	1	0.9454014	0.5664805	average	1	0.929931	0.4547486	average	0.9944229	0.9207777	0.6089386	average	0.7642704	0.7106576	0.5486032	average	0.99884	0.89176	0.51508

Deep learning:



one layer, 4 nodes				one layer, 8 nodes				one layer, 12 nodes				two layers, 8 nodes				two layers, 8 nodes, activation = 'logistic'			
	trn	tst	oot		trn	tst	oot		trn	tst	oot		trn	tst	oot		trn	tst	oot
1	0.67981	0.65811	0.35754	1	0.70578	0.71785	0.52513	1	0.74876	0.67953	0.53072	1	0.76547	0.76771	0.54189	1	0.69085	0.67948	0.40223
2	0.69082	0.72064	0.35754	2	0.68181	0.73015	0.39664	2	0.70894	0.66007	0.44692	2	0.76985	0.72509	0.53631	2	0.65990	0.72246	0.43016
3	0.71147	0.66279	0.44134	3	0.69157	0.70711	0.53072	3	0.66942	0.73003	0.45810	3	0.77076	0.76315	0.51396	3	0.69155	0.66666	0.41340
4	0.66885	0.73255	0.36871	4	0.72312	0.66929	0.51955	4	0.73770	0.73643	0.53631	4	0.76026	0.74131	0.53072	4	0.68760	0.70342	0.37430
5	0.68019	0.69047	0.35754	5	0.69485	0.68301	0.52513	5	0.72032	0.68379	0.50837	5	0.76254	0.77407	0.56983	5	0.68595	0.66539	0.44692
6	0.65533	0.67600	0.43016	6	0.73241	0.74035	0.54748	6	0.73863	0.72222	0.51955	6	0.79344	0.75581	0.58100	6	0.69622	0.66023	0.39664
7	0.70069	0.64827	0.42458	7	0.69407	0.69615	0.50279	7	0.75124	0.72075	0.54189	7	0.75280	0.77551	0.54189	7	0.69131	0.66260	0.41899
8	0.67313	0.66400	0.35195	8	0.70508	0.74820	0.53072	8	0.73892	0.69915	0.53072	8	0.77664	0.69314	0.53072	8	0.68571	0.68864	0.45251
9	0.64369	0.64835	0.40782	9	0.71135	0.75213	0.54748	9	0.69391	0.70901	0.47486	9	0.73244	0.77407	0.54189	9	0.69817	0.64150	0.40782
10	0.64448	0.66269	0.35754	10	0.74712	0.73359	0.54748	10	0.74752	0.74045	0.55865	10	0.78548	0.77350	0.56983	10	0.68527	0.67509	0.43016
avg	0.67485	0.67639	0.38547	avg	0.70872	0.71778	0.51731	avg	0.72554	0.70814	0.51061	avg	0.76697	0.75434	0.54581	avg	0.68725	0.67655	0.41731