Documentation of Project 2 (DSO 562)

Jiayi (Lily) Hu

INDEX

Section 1. Executive Summary 2
Section 2. Description of the data 2
Section 3. Data cleaning 3
(1) Filtering the outliers and transaction types 3
(2) Filling in the missing values 3
Section 4. Variable creation 4
Section 5. Feature selection 5
Section 6. Preliminary model explores 6
Section 7. Final model performance 12
Section 8. Financial curves and recommended cutoff 14
Section 9. Summary 14
Section 10. Appendix 16

Section 1. Executive Summary

This project aims to use a machine learning model to solve credit card transaction fraud detection problems. The whole project involved data quality exploration, feature engineering, feature selection, preliminary model testing, final model running, and financial curves and cutoff settings based on the final model outcomes. The final model of this project can detect 55.87% of all frauds in 3% of the population. Based on the result from the model, we recommend setting the score cutoff at 3%, at which our estimated overall saving is $20,472,000 per year. The cutoff is not too sensitive to get a high denial rate, which will help reduce the chance to misjudge a normal transaction to be fraudulent.

Section 2. Description of the data

This data is about credit card transaction data. This dataset contains the basic transaction data including card number, transaction date, amount & type, and merchant information. The data covers the time of the **year 2010** (from 2010-01-01 to 2010-12-31). There are **10 fields** (2 are numerical and 8 are categorical) and **96,753 records**. Among all those records, there are 1,059 records labeled to be fraudulent and 95,694 labeled to be normal. Therefore, the overall fraudulent rate is 1.095%.

Table 1. Summary of fields

Numerical Fields						
Field Name	**% Populated**	**Min**	**Max**	**Mean**	**Stdev**	**% Zero**
Date	100	2010-01-01 (1/1/10)	2010-12-31 (12/31/10)	/	/	0
Amount	100	0.01	3,102,045.53	427.89	10,006.14	0
Categorical Fields						
Field Name	**% Populated**	**# Blanks**	**# Zeros**	**# Unique Values**	**Most Common Value**	
Recnum	100	0	0	96,753	/	
Cardnum	100	0	0	1,645	5142148452	
Merchnum	96.51	3,375	231	13,091	9.3009E+11	
Merch description	100	0	0	13,126	GSA-FSS-ADV	
Merch state	98.76	1,195	0	227	TN	
Merch zip	95.19	4,656	0	4,567	38118	
Transtype	100	0	0	4	P	
Fraud	100	0	0	2	0	

Section 3. Data cleaning

The data cleaning part for this project has two parts:

- filtering the outliers and transaction types;
- filling in the missing values.

(1) Filtering the outliers and transaction types

There are 4 types of transaction types in the dataset, the majority of all the transaction records are type ‘P’, which means ‘Purchase’. However, there is no clear clue that can tell what are the other 3 types mean. Therefore, the project will only pay attention to the subset with all transactions for purchase.

According to the DQR, there is an outlier in the field ‘Amount’. Therefore, the outlier is also dropped by the filtering process.

(2) Filling in the missing values

There are 3 fields with missing values: ‘Merchnum’, ‘Merch state’, and ‘Merch zip’.

- Merch state:
  There are 3 different types of missing ‘Merch state’: with ‘Merch zip’ in the US, with ‘Merch zip’ outside the US, and without ‘Merch zip’. For those missing values in this field with ‘Merch zip’, SearchEngine function from the uszipcode package is used to generate the abbreviation of the states, this function will help detect those ‘Merch zip’ outside the US as well. Missing values with ‘Merch zip’ outside the US are filled in with ‘Unknown’. Missing values without ‘Merch zip’ are filled in with the mode of ‘Merch description’, which is ‘VA’.

- Merchnum:
  ‘Merchnum’ with value of ‘0’ is defined as a frivolous value because Merchant numbers shouldn’t be one digit. All ‘0’s in field ‘Merchnum’ are replaced with ‘NaN’, and all the ‘NaN’ (3251 records after filtering and replace ‘0’ with NaN) are filled in with the mode of ‘Merch description’, which is '9900020006406'.

- Merch zip:
  All the missing values in ‘Merch zip’ are filled in with the mode of ‘Merch description’, which is ‘22202’.

Section 4. Variable creation

Transaction fraud basically means a kind of fraud involving the unauthorized use of financial accounts. There are several types of transaction frauds, those basic types including: (1) Account lost or stolen; (2) Skimming and counterfeit: crimes that use skimmers to steal credit or debit information from victims and store stolen information in a counterfeit card to make a purchase or withdraw cash; (3) Phishing: tricking people to reveal their personal and financial information to get access to their accounts and make transactions.

The purpose of the feature engineering (variable creation) process is to create as many potential variables as possible for further selection. The original fields are combined to form a list of 32 entities (see Entities List below), and those entities are used to build variables. The following groups of variables are created to help catch transaction fraud:

Table 2. Variable Creation Summary

Description of variables	# Variables created	# Variables after dedup
Risk Variable (dow risk): the likelihood of fraud for any day of the week	1	1
Benford's Law Variables: Benford's Law measure with smoothing on 'Cardnum' (Card Number) and 'Merchnum' (Merch number)	2	2
Days Since Variables: # days since a transaction with that entity has been seen. Entities list is attached below	32	207
Velocity: # records with the same entity over the past {0,1,3,7,14, 30} days	192	
Relative Velocity (velocity change variables): the ratio of # records with the same entity over the past {0,1} day out of the average # records with the same entity over the past {7,14,30} days	128	252
Velocity Days Since Ratio (velocity change variables): the ratio of # records with the same entity over the past {0,1} day out of relative velocity over # days since a transaction with that entity has been seen (days since)	128	
Amount Variables: {avg, max, median, total} of amount with the same entity over the past {0,1,3,7,14,30} days; and the actual amount over the {avg, max, median, total} of amount with the same entity over the past {0,1,3,7, 14,30} days	1538	1454
Variability: {avg, max, median} of difference in amount with the same entity between the recent amount and previous amount over the past {0,1,3,7,14,30} days	576	544

Cross Entity Uniqueness Variables: For the same entity, # unique records of another entity over the past {0,1,3,7,14,30} days *PS: Due to my computational power, I can't run all the entities I built before, so I used a subset of my entities that my laptop is able to run and also contained those I believe most make sense (see Entities List 2).*	2394	521
Total # of variables	4991	2981

Entities List:
['Cardnum','Merchnum','Merch_description','Merch_state','Merch_zip','Merch_num_des','Merch_num_state','Merch_num_zip','Merch_des_state','Merch_des_zip','Merch_state_zip','Merch_num_des_state','Merch_ num_des_zip','Merch_num_state_zip','Merch_des_state_zip','Merch_all_info','Card_Merchnum','Card_Merch_description','Card_Merch_state','Card_Merch_zip','Card_Merch_num_des','Card_Merch_num_state','Card_ Merch_num_zip','Card_Merch_des_state','Card_Merch_des_zip','Card_Merch_state_zip','Card_Merch_num_ des_state','Card_Merch_num_des_zip','Card_Merch_num_state_zip','Card_Merch_des_state_zip','Card_Merch_all_ info', 'zip3']

Entities List 2 (subset of entities, for generating 'Cross Entity Uniqueness Variables' only):
['Cardnum','Merchnum','Merch_description','Merch_num_des','Merch_all_info','Card_Merchnum','Card_Merch_description','Card_Merch_state','Card_Merch_zip','Card_Merch_num_des','Card_Merch_num_state','Card_Merch_ num_zip','Card_Merch_des_state','Card_Merch_des_zip','Card_Merch_state_zip','Card_Merch_num_des_state','Card_Merch_num_des_zip','Card_Merch_num_state_zip','Card_Merch_des_state_zip','Card_Merch_all_info','zip3']

PS: Days Since Variables & Velocity, Relative Velocity & Velocity Days Since Ratio are generated in the same cell and saved in the same CSV files, so my code for deduplication only generates the total number of variables in those CSV files for me.

Section 5. Feature selection

The feature engineering (variable creation) process generated 4,991 variables. After deduplication, there are 2,981 variables left for further usage. To get rid of the curse of dimensionality, we ran feature selection before building supervised learning models.

In the feature selection process, we first ran a filter, using a univariate KS algorithm to calculate the filter score for each variable, higher score means the independent variable is more important for predicting the dependent variable. We chose the top 600 variables with the highest KS values and ran a wrapper, by setting the number of variables in the wrapper to 25, we finally detect the set of variables that can best predict whether an application is fraudulent or not. Here is the list of variables in the wrapper:

Table 3. Final variables in the wrapper

wrapper order	variable	filter score
1	card_merch_total_14	0.63018599
2	card_zip3_max_14	0.62963229
3	Card_Merchdesc_count_7	0.3671098
4	Merchnum_desc_max_0	0.53083798
5	zip3_avg_1	0.40477391
6	Card_Merchnum_desc_avg_30	0.51958114
7	Card_Merchdesc_avg_14	0.51755581
8	card_merch_avg_7	0.52448126
9	card_zip3_med_14	0.47833039
10	zip3_med_1	0.36449386
11	Merchnum_avg_3	0.48775621
12	Merchnum_avg_7	0.45707857
13	card_merch_avg_30	0.52114358
14	card_merch_avg_60	0.5151611
15	Merchnum_desc_avg_1	0.50655124
16	card_zip3_med_7	0.48994143
17	merch_zip_avg_3	0.48693151
18	merch_zip_med_0	0.47147531
19	Merchnum_med_0	0.47143183
20	Merchnum_desc_med_0	0.47130142
21	merch_zip_avg_7	0.45631909
22	Merchnum_desc_avg_7	0.45362765
23	merch_zip_med_1	0.45118048
24	Merchnum_med_1	0.4502471
25	Merchnum_desc_med_1	0.44891161

Section 6. Preliminary model explores

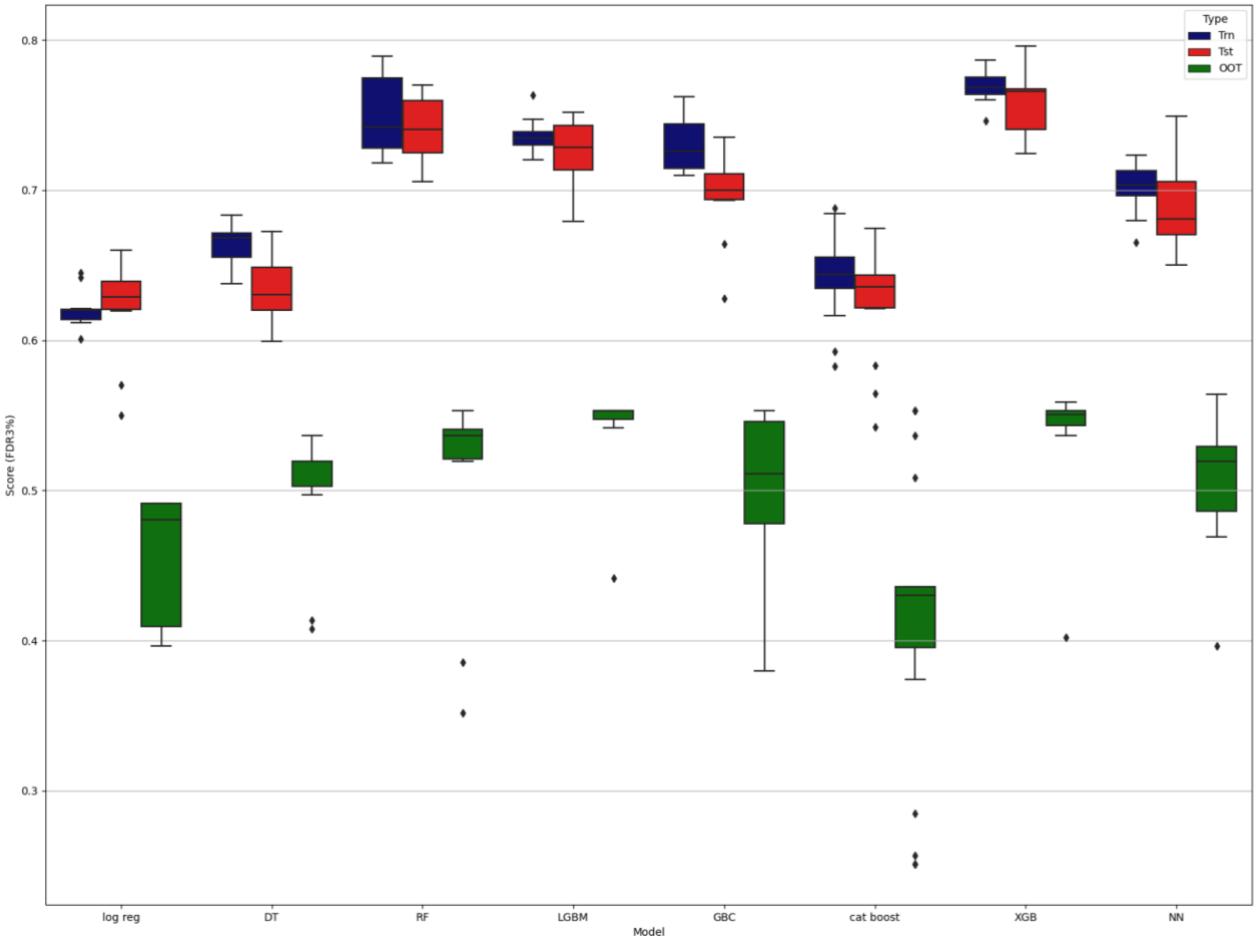
Using the 25 selected final variables (See Table 3) to run several supervised models and find the one that works best. Each model was run 10 times and the average of the 10 results was taken as the final performance. A logistic regression model is used to test the baseline, all other non-linear models should get prediction scores higher than logistic regression, or they shouldn't be chosen as the final model. T.he following table shows the outcomes of the preliminary model exploration.

Table 4. Preliminary models exploration

Model		Parameters								Average FDR at 3%		
Logistic Regression	Iteration	Num of Variables	penalty	solver	C	l1_ratio				Train	Test	OOT
	1	10	l2	lbfgs	1	N/A				0.6106	0.6195	0.4570
	2	10	l1	saga	1	N/A				0.6130	0.6079	0.4352
	3	10	l2	saga	0.5	N/A				0.6128	0.6038	0.4285
	4	10	elasticnet	saga	0.5	0.5				0.6120	0.6254	0.4676
	5	15	l2	lbfgs	1	N/A				0.5996	0.6168	0.4587
	6	15	elasticnet	saga	0.5	0.5				0.6066	0.6114	0.4648
Decision Tree	Iteration	Num of Variables	criterion	splitter	max_depth	min_sample_split	min_samples_leaf			Train	Test	OOT
	1	10	gini	best	5	10	5			0.6887	0.6736	0.4631
	2	10	gini	best	3	20	10			0.6289	0.6037	0.4458
	3	10	gini	random	10	5	5			0.7435	0.6766	0.4341
	4	10	entropy	best	3	5	5			0.6566	0.6486	0.5089
	5	15	gini	best	5	10	5			0.6969	0.6658	0.4860
	6	15	entropy	best	3	5	5			0.6595	0.6486	0.4994
Random Forest	Iteration	Num of Variables	criterion	n_estimators	max_depth	min_sample_split	min_samples_leaf			Train	Test	OOT
	1	10	gini	50	10	50	10			0.8525	0.7874	0.4296
	2	10	gini	10	5	30	30			0.7474	0.7411	0.5140
	3	10	entropy	8	5	50	10			0.7810	0.7493	0.5034
	4	10	entropy	8	5	20	5			0.7729	0.7636	0.5402
	5	15	gini	10	5	30	30			0.7533	0.7399	0.5084
	6	15	entropy	8	5	20	5			0.7721	0.7848	0.4475
LightGBM	Iteration	Num of Variables	boosting_type	n_estimators	num_leaves	max_depth	learning_rate	colsample_bytree	subsample	Train	Test	OOT
	1	10	GOSS	10	10	5	0.03	1	1	0.7562	0.7383	0.5369
	2	10	GOSS	10	5	5	0.03	0.8	0.8	0.7221	0.7106	0.5391
	3	10	gbdt	10	5	8	0.01	1	1	0.6822	0.6805	0.5413
	4	**10**	**gbdt**	**20**	**5**	**8**	**0.01**	**0.8**	**0.8**	**0.7130**	**0.6985**	**0.5430**
	5	15	GOSS	10	5	5	0.03	0.8	0.8	0.7304	0.7239	0.5285
	6	15	gbdt	20	5	8	0.01	1	1	0.7022	0.6825	0.5391

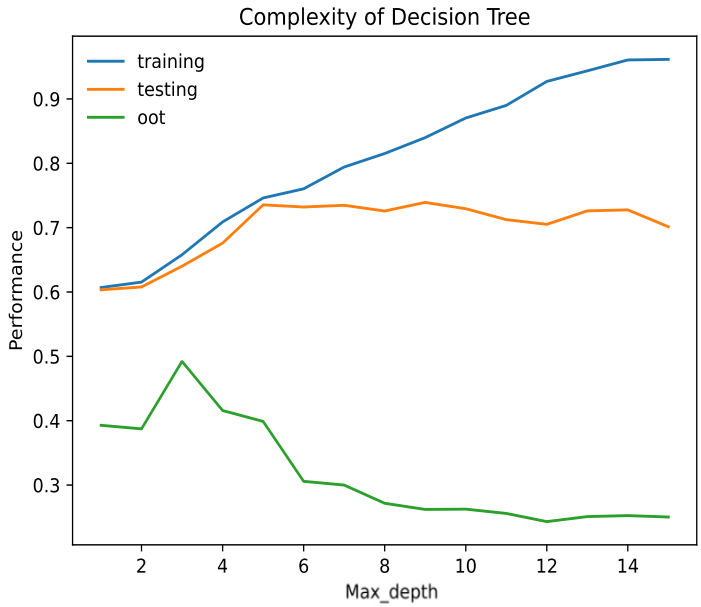
Model		Parameters							Average FDR at 3%		
Neural Network	Iteration	Num of Variables	hidden_layer_sizes	activation	alpha	learning_rate	solver	learning_rate_init	Train	Test	OOT
	1	10	(5,5)	logistic	0.0001	adaptive	lbfgs	0.0001	0.7127	0.6991	0.4592
	2	10	(10,10)	relu	0.0001	adaptive	lbfgs	0.0001	0.7460	0.7314	0.4525
	3	10	(20,20,20)	relu	0.001	constant	adam	0.0001	0.7079	0.6933	0.5145
	4	10	(20,20,20)	logistic	0.01	constant	adam	0.001	0.6599	0.6607	0.5408
	5	15	(10,10)	relu	0.001	constant	adam	0.001	0.7569	0.7447	0.4564
	6	15	(20,20,20)	logistic	0.01	constant	adam	0.001	0.6814	0.6745	0.5011
Gradient Boosting Classifier	Iteration	Num of Variables	max_depth	n_estimators	learning_rate		min_sample_split	min_samples_leaf	Train	Test	OOT
	1	10	6	50	0.001		10	5	0.7239	0.6781	0.4989
	2	10	6	20	0.01		10	5	0.7228	0.7082	0.5179
	3	10	4	50	0.001		20	10	0.6663	0.6507	0.4810
	4	10	4	20	0.01		20	10	0.6841	0.6769	0.5101
	5	15	6	20	0.01		10	5	0.7246	0.6883	0.4922
	6	15	4	20	0.01		20	10	0.6715	0.6736	0.5045
Catboost	Iteration	Num of Variables	max_depth	iterations	learning_rate		bootstrap_type	l2_leaf_reg	Train	Test	OOT
	1	10	5	10	0.01		Bayesian	5	0.6404	0.6113	0.4704
	2	10	10	5	0.01		Bayesian	5	0.6210	0.6119	0.4179
	3	10	10	50	0.001		Bayesian	8	0.6675	0.6772	0.4179
	4	10	10	10	0.001		Bernoulli	8	0.6708	0.6483	0.4067
	5	15	5	5	0.1		Bayesian	8	0.6162	0.6112	0.4453
	6	15	5	5	0.2		Bayesian	10	0.6445	0.6432	0.4173
XGBoost	Iteration	Num of Variables	max_depth	n_estimators	booster	tree_method	min_child_weight	subsample	eta	colsample_bytree	Train	Test	OOT
	1	10	10	50	gbtree	exact	10	0.8	0.2	0.8	0.9057	0.8294	0.4123
	2	10	6	20	gbtree	exact	5	1	0.1	1	0.7727	0.7467	0.5480
	3	10	6	20	gbtree	auto	5	0.8	0.2	0.8	0.8333	0.7807	0.4916
	4	15	6	20	gbtree	exact	5	0.8	0.2	0.8	0.8341	0.7781	0.4531
	5	15	6	20	gbtree	auto	10	1	0.1	1	0.7640	0.7509	0.5453
	6	15	4	10	gbtree	atuo	10	1	0.1	1	0.6997	0.7032	0.5441

There are several models whose performances on out of time set (OOT) are over 0.54, which is a relatively high performance. To choose the final model, the best model hyperparameters for each model type are chosen and being rerun to generate the model comparison box plot:



From the box plot, we can see that both XGBoost and LGBM work well on the OOT set, with high performance and low dispersion. Considering interpretability, I will choose LGBM as my final model. (The row highlighted in Table 4).

For each kind of model (decision tree, random forest, boosted tree, and neural network), a complexity–performance plot is generated to show how varying a specific hyperparameter can affect model performance. This can help us better understand how to tune the hyperparameters of models to avoid overfitting and underfitting.

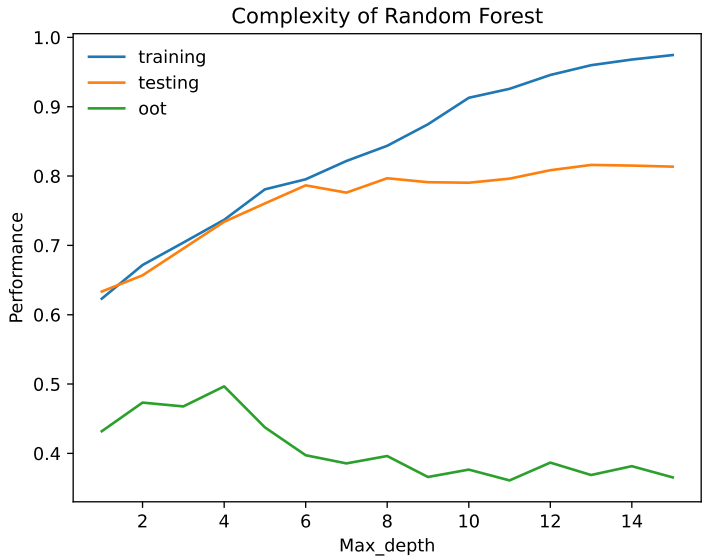


Complexity of Decision Tree:

Number of Variable: 10

Hyperparameters: criterion = 'entropy', splitter = 'best',
min_samples_split = 5, min_samples_leaf = 5

What is varying: max_depth, ranges from 1 to 15.

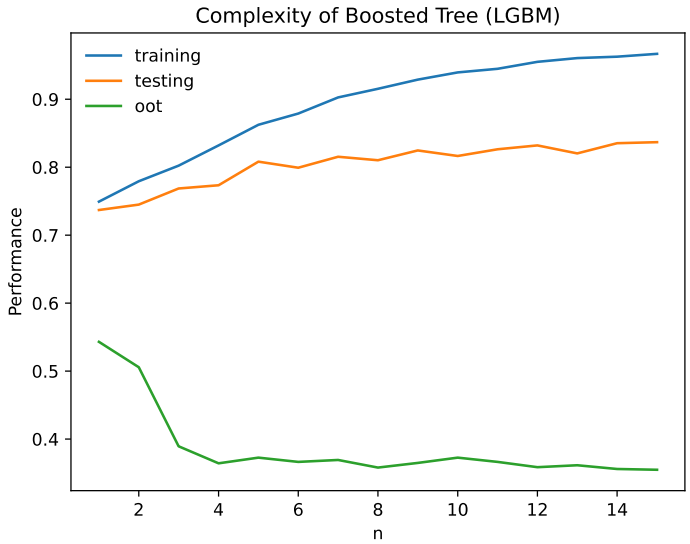


Complexity of Random Forest:

Number of Variable: 10

Hyperparameters: criterion = 'entropy', n_estimators= 8,
min_samples_split = 20, min_samples_leaf = 5

What is varying: max_depth, ranges from 1 to 15.

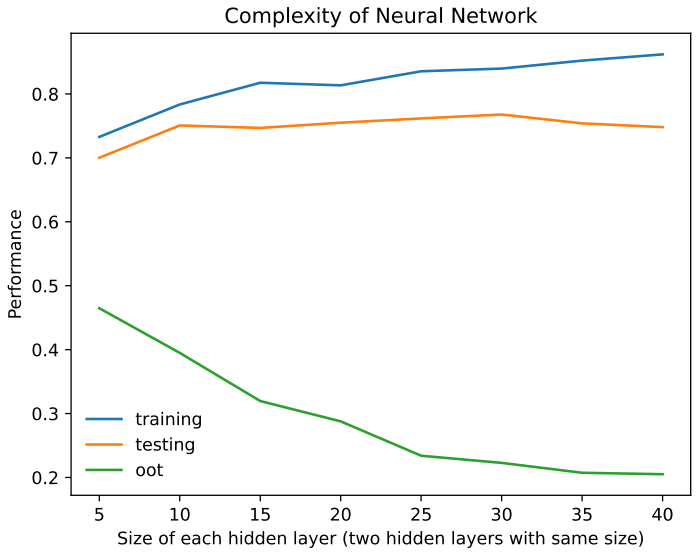


Complexity of Boosted Tree (LGBM):

Number of Variable: 10

Hyperparameters: boosting_type = 'gbdt', colsample_bytree =1,
max_depth=8, subsample=1, learning_rate = 0.01

What are varying: n_estimators = 20*n, num_leaves= 5+5*n, the range of n is from 1 to 15. So the range of n_estimators is range from 20 to 300 with a step of 20, and num_leaves is range from 10 to 80 with a step of 5.



Complexity of Neural Network:

Number of Variable: 10

Hyperparameters: activation = 'relu', alpha = 0.1, learning_rate = 'constant',
solver = 'lbfgs', learning_rate_init = 0.0001, max_iter = 500

What is varying: the size of hidden layers. I set this neural network with hidden_layer_sizes= (5*n,5*n) (where n ranges from 1 to 8), so the neural network has two hidden layers with the same size, and the size of each layer ranges from 5 to 40 with a step of 5.

Section 7. Final model performance

The final model I choose is the LightGBM model (a tree-based gradient-boosting machine learning model) with the following hyperparameters:

- number of variables = 10: choosing 10 variables out of the 25 final ones to run the model;
- boosting_type = 'gbdt': Gradient-boosted decision trees method;
- n_estimators = 20: number of boosted trees to fit;
- num_leaves= 5: maximum tree leaves for base learners;
- max_depth = 8: maximum tree depth for base learners;
- learning_rate = 0.01: boosting learning rate, governing the pace for the algorithm to learn the values of a parameter estimate;
- colsample_bytree = 0.8: subsample ratio of columns when constructing each tree;
- subsample = 0.8: subsample ratio of the training instance.

Here are the final variables used in this model:

1	card_merch_total_14
2	card_zip3_max_14
3	Card_Merchdesc_count_7
4	Merchnum_desc_max_0
5	zip3_avg_1
6	Card_Merchnum_desc_avg_30
7	Card_Merchdesc_avg_14
8	card_merch_avg_7
9	card_zip3_med_14
10	zip3_med_1

The following are the three result tables for the final model (See Table 5). For all three subsets, the top 2% of the population bin can already detect over 50% of the total number of frauds. As we set the bins to be top 3% for running the model, the best FDR we can get for the training set should be about 71.02%, the test set about 69.39%, and the out-of-time set about 55.87%.

Table 5. Result tables of the final model
(TRN: training set, TST: test set, OOT: out-of-time set)

TRN	# Records	# Goods	# Bads	Fraud Rate
	58779	58144	635	1.08%

	Bin Statistics					Cumulative Statistics						
Population Bin%	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Cumulative Goods	% Cumulative Bads (FDR)	KS	FPR
1	588	233	355	39.63%	60.37%	588	233	355	0.40%	55.91%	55.50	0.66
2	588	511	77	86.90%	13.10%	1176	744	432	1.28%	68.03%	66.75	1.72
3	587	568	19	96.76%	3.24%	1763	1312	451	2.26%	71.02%	68.77	2.91
4	588	574	14	97.62%	2.38%	2351	1886	465	3.24%	73.23%	69.98	4.06
5	588	572	16	97.28%	2.72%	2939	2458	481	4.23%	75.75%	71.52	5.11
6	588	565	23	96.09%	3.91%	3527	3023	504	5.20%	79.37%	74.17	6.00
7	588	587	1	99.83%	0.17%	4115	3610	505	6.21%	79.53%	73.32	7.15
8	587	580	7	98.81%	1.19%	4702	4190	512	7.21%	80.63%	73.42	8.18
9	588	585	3	99.49%	0.51%	5290	4775	515	8.21%	81.10%	72.89	9.27
10	588	584	4	99.32%	0.68%	5878	5359	519	9.22%	81.73%	72.52	10.33
11	588	583	5	99.15%	0.85%	6466	5942	524	10.22%	82.52%	72.30	11.34
12	587	585	2	99.66%	0.34%	7053	6527	526	11.23%	82.83%	71.61	12.41
13	588	581	7	98.81%	1.19%	7641	7108	533	12.22%	83.94%	71.71	13.34
14	588	582	6	98.98%	1.02%	8229	7690	539	13.23%	84.88%	71.66	14.27
15	588	582	6	98.98%	1.02%	8817	8272	545	14.23%	85.83%	71.60	15.18
16	588	581	7	98.81%	1.19%	9405	8853	552	15.23%	86.93%	71.70	16.04
17	587	583	4	99.32%	0.68%	9992	9436	556	16.23%	87.56%	71.33	16.97
18	588	583	5	99.15%	0.85%	10580	10019	561	17.23%	88.35%	71.12	17.86
19	588	583	5	99.15%	0.85%	11168	10602	566	18.23%	89.13%	70.90	18.73
20	588	582	6	98.98%	1.02%	11756	11184	572	19.24%	90.08%	70.84	19.55

TST	# Records	# Goods	# Bads	Fraud Rate
	25191	24946	245	0.97%

	Bin Statistics					Cumulative Statistics						
Population Bin%	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Cumulative Goods	% Cumulative Bads (FDR)	KS	FPR
1	252	120	132	47.62%	52.38%	252	120	132	0.48%	53.88%	53.40	0.91
2	252	223	29	88.49%	11.51%	504	343	161	1.37%	65.71%	64.34	2.13
3	252	243	9	96.43%	3.57%	756	586	170	2.35%	69.39%	67.04	3.45
4	252	252	0	100.00%	0.00%	1008	838	170	3.36%	69.39%	66.03	4.93
5	252	248	4	98.41%	1.59%	1260	1086	174	4.35%	71.02%	66.67	6.24
6	251	241	10	96.02%	3.98%	1511	1327	184	5.32%	75.10%	69.78	7.21
7	252	250	2	99.21%	0.79%	1763	1577	186	6.32%	75.92%	69.60	8.48
8	252	249	3	98.81%	1.19%	2015	1826	189	7.32%	77.14%	69.82	9.66
9	252	248	4	98.41%	1.59%	2267	2074	193	8.31%	78.78%	70.46	10.75
10	252	250	2	99.21%	0.79%	2519	2324	195	9.32%	79.59%	70.28	11.92
11	252	249	3	98.81%	1.19%	2771	2573	198	10.31%	80.82%	70.50	12.99
12	252	252	0	100.00%	0.00%	3023	2825	198	11.32%	80.82%	69.49	14.27
13	252	251	1	99.60%	0.40%	3275	3076	199	12.33%	81.22%	68.89	15.46
14	252	250	2	99.21%	0.79%	3527	3326	201	13.33%	82.04%	68.71	16.55
15	252	251	1	99.60%	0.40%	3779	3577	202	14.34%	82.45%	68.11	17.71
16	252	250	2	99.21%	0.79%	4031	3827	204	15.34%	83.27%	67.92	18.76
17	251	245	6	97.61%	2.39%	4282	4072	210	16.32%	85.71%	69.39	19.39
18	252	248	4	98.41%	1.59%	4534	4320	214	17.32%	87.35%	70.03	20.19
19	252	251	1	99.60%	0.40%	4786	4571	215	18.32%	87.76%	69.43	21.26
20	252	252	0	100.00%	0.00%	5038	4823	215	19.33%	87.76%	68.42	22.43

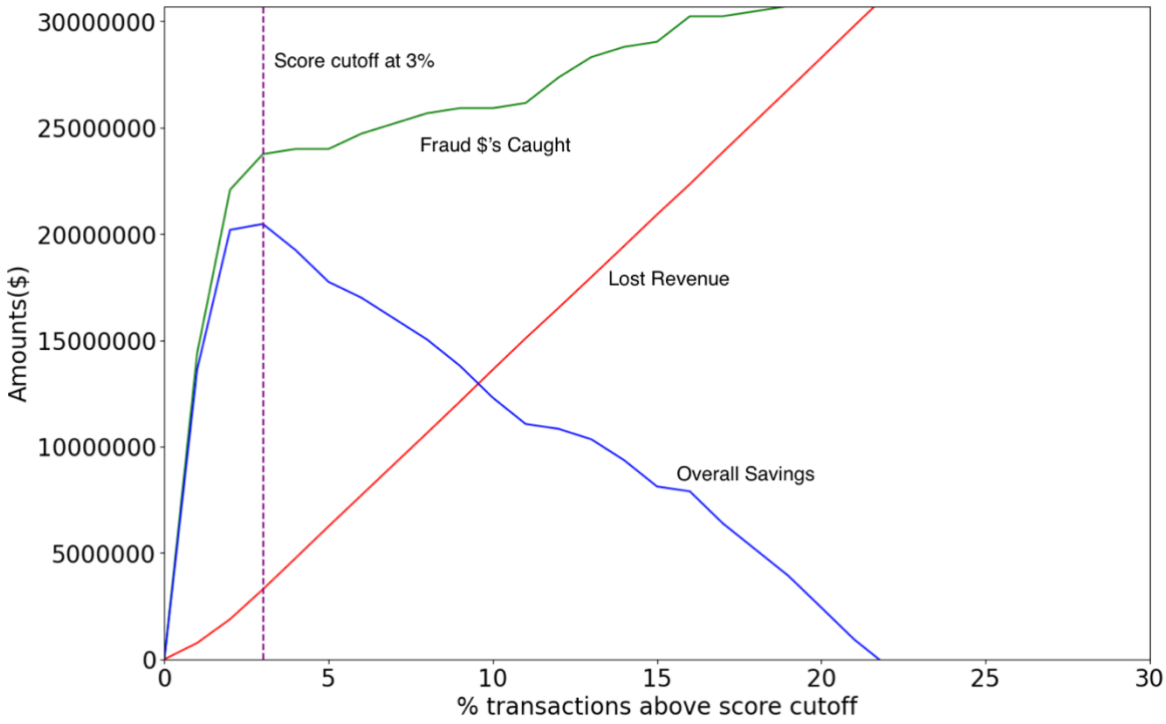
OOT	# Records	# Goods	# Bads	Fraud Rate
	12427	12248	179	1.44%

	Bin Statistics					Cumulative Statistics						
Population Bin%	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Cumulative Goods	% Cumulative Bads (FDR)	KS	FPR
1	124	49	75	39.52%	60.48%	124	49	75	0.40%	41.90%	41.50	0.65
2	125	105	20	84.00%	16.00%	249	154	95	1.26%	53.07%	51.82	1.62
3	124	119	5	95.97%	4.03%	373	273	100	2.23%	55.87%	53.64	2.73
4	124	124	0	100.00%	0.00%	497	397	100	3.24%	55.87%	52.62	3.97
5	124	124	0	100.00%	0.00%	621	521	100	4.25%	55.87%	51.61	5.21
6	125	125	0	100.00%	0.00%	746	646	100	5.27%	55.87%	50.59	6.46
7	124	124	0	100.00%	0.00%	870	770	100	6.29%	55.87%	49.58	7.70
8	124	120	4	96.77%	3.23%	994	890	104	7.27%	58.10%	50.83	8.56
9	124	120	4	96.77%	3.23%	1118	1010	108	8.25%	60.34%	52.09	9.35
10	125	122	3	97.60%	2.40%	1243	1132	111	9.24%	62.01%	52.77	10.20
11	124	123	1	99.19%	0.81%	1367	1255	112	10.25%	62.57%	52.32	11.21
12	124	123	1	99.19%	0.81%	1491	1378	113	11.25%	63.13%	51.88	12.19
13	125	125	0	100.00%	0.00%	1616	1503	113	12.27%	63.13%	50.86	13.30
14	124	123	1	99.19%	0.81%	1740	1626	114	13.28%	63.69%	50.41	14.26
15	124	123	1	99.19%	0.81%	1864	1749	115	14.28%	64.25%	49.97	15.21
16	124	122	2	98.39%	1.61%	1988	1871	117	15.28%	65.36%	50.09	15.99
17	125	121	4	96.80%	3.20%	2113	1992	121	16.26%	67.60%	51.33	16.46
18	124	120	4	96.77%	3.23%	2237	2112	125	17.24%	69.83%	52.59	16.90
19	124	123	1	99.19%	0.81%	2361	2235	126	18.25%	70.39%	52.14	17.74
20	124	119	5	95.97%	4.03%	2485	2354	131	19.22%	73.18%	53.96	17.97

Section 8. Financial curves and recommended cutoff

Assume that $400 gain from every fraud that is caught, and $20 is lost when a normal transaction is labeled to be fraudulent (false positive). Based on the prediction results from running the final model, I calculated the Fraud $'s Caught (green curve), Lost Revenue (red curve), and Overall Savings (blue curve) and showed those curves in the following plot to illustrate expected annual savings based on the percentage of transactions above score cutoff.

The recommended cutoff is 3% because the detection system will not be too sensitive to get a high false positive rate, and we can also get a high overall saving ($20,472,000) at this score cutoff.



Section 9. Summary

The project is designed for building a supervised machine learning model to detect credit card transaction frauds. The dataset used in this project has 10 fields (2 are numerical and 8 are categorical) and 96,753 records, with 1,059 records labeled to be fraudulent and 95,694 labeled to be normal. The overall fraudulent rate is 1.095%.

The dataset is filtered on the 'Transtype' field (only keep Transtype = 'P') and the missing values in 'Merch state', 'Merchnum', and 'Merch zip' are filled by the corresponding values of the mode, the outlier in 'Amount' is removed. After data cleaning, there are 96,397 records left in the cleaned dataset. By using the 8 original fields (exclude 'Recnum' and 'Fraud'), 32 entities are generated, and 2,981 variables (after deduplication) are created based on those entities. There are 9 types of all variables, including risk on day of week, Benford's Law variable, day since, velocity, relative velocity, velocity day since ratio, amount, variability, and cross-entity uniqueness. To reduce dimensionality and capture the best indicators, we conducted the feature selection process. First, we calculated the filter score for each variable using the univariate KS algorithm. Variables with the top 600 highest KS values are chosen for running the wrapper. We set the number of variables in the wrapper to 25 so that we finally got a set of 25 variables that can best predict fraud.

The final set of 25 variables is used to test several supervised learning models and help make the final decision on model selection. The last two month's data is separated as the out-of-time set, which is the main subset to test the model performance. For the rest 10 months' data, cross-validation is used to generate different train and test sets for each run, and each model is run 10 times, performance is measured based on the average value of those 10 runs. Logistic regression is used to test the baseline of predictability, if a model's performance is poorer than logistic regression, then it can't be chosen as the final model. The models we used in the exploration include a decision tree, random forest, boosted tree (including LGBM, GBC, Catboost, XGBoost), and neural network. Hyperparameters of each model have been tuned in 6 ways for getting the best performance and avoiding overfitting and underfitting. The best hyperparameter sets of each model are chosen for the final run before the final model decision. A box plot is generated to show the performance of the best model in each model type. According to interpretability, performance, and stability of the outcomes, LGBM with the following hyperparameters (boosting_type = 'gbdt', n_estimators = 20, num_leaves= 5, max_depth = 8, learning_rate = 0.01, colsample_bytree = 0.8, subsample = 0.8) is chosen to be the final model. (number of variables = 10). The final model can catch over 50% of the total number of frauds with the top 2% of the population.

Assume that $400 gain from every fraud that is caught, and $20 is lost when a normal transaction is labeled to be fraudulent (false positive). Based on the final model outcomes, we recommended setting the score cutoff to 3%, where the estimated yearly overall saving can reach $20,472,000. Setting the score cutoff at 3% can avoid the fraud detection system to be too sensitive and also get a high savings. At a 3% score cutoff, the best FDR we can get for the training set should be about 71.02%, the test set about 69.39%, and the out-of-time set about 55.87%.

There are some limitations and potential improvements we can do to the process of the project. First, because of the computational power, even though there are about 3,000 variables, we can still get more different types of variables (especially variables based on locations and zip codes) to help improve the metrics used to test the models. Second, the pattern of the last two months' data is abnormal compared to the previous 10 months, so whether it is suitable to use this subset of data as an out-of-time set (OOT) is questionable. It might be a better idea that we don't use OOT in this case. Third, the assumptions of the financial curve are quite simple, the reality can be much more complicated, so we need to consider more before setting the score cutoff.

Section 10. Appendix

Data Quality Report

1. Data Description

This data is about credit card transaction data. This dataset contains the basic transaction data including card number, transaction date, amount & type, and merchant information. The data covers the time of the **year 2010** (from 2010-01-01 to 2010-12-31). There are **10 fields** (2 are numerical and 8 are categorical) and **96,753 records**.

2. Summary Tables

(1) Numerical Table

Field Name	% Populated	Min	Max	Mean	Stdev	% Zero
Date	100.00	2010-01-01 (1/1/10)	2010-12-31 (12/31/10)	/	/	0.00
Amount	100.00	0.01	3,102,045.53	427.89	10,006.14	0.00

(2) Categorical Table

Field Name	% Populated	# Blanks	# Zeros	# Unique Values	Most Common Value
Recnum	100.00	0	0	96,753	/
Cardnum	100.00	0	0	1,645	5142148452
Merchnum	96.51	3,375	231	13,091	930090121224
Merch description	100.00	0	0	13,126	GSA-FSS-ADV
Merch state	98.76	1,195	0	227	TN
Merch zip	95.19	4,656	0	4,567	38118
Transtype	100.00	0	0	4	P
Fraud	100.00	0	0	2	0

3. Visualization of Each Field

(1) Field Name: Recnum
Description: Number of Records. Ordinal unique positive integer for each record, from 1 to 96,753.

(2) Field Name: Date
Description: The date when one transaction has happened and has been recorded. Fig 1-1 shows daily transactions. The number of transactions per day fluctuated so it's hard to detect useful information. However, a decrease in the number of transactions can be noticed from October onwards. Fig 1-2 shows the number of transactions per week. Noticed that the last week only contains one day (2010-12-31), so we don't include that week in the plot. The pattern becomes much more obvious that the number of transactions slowly increased from January to early September, and sharply decreased starting from

late September. Fig 1-3 shows the number of transactions per month[1], it's even clearer that number of transactions sharply went down by September and reached its lowest in October.

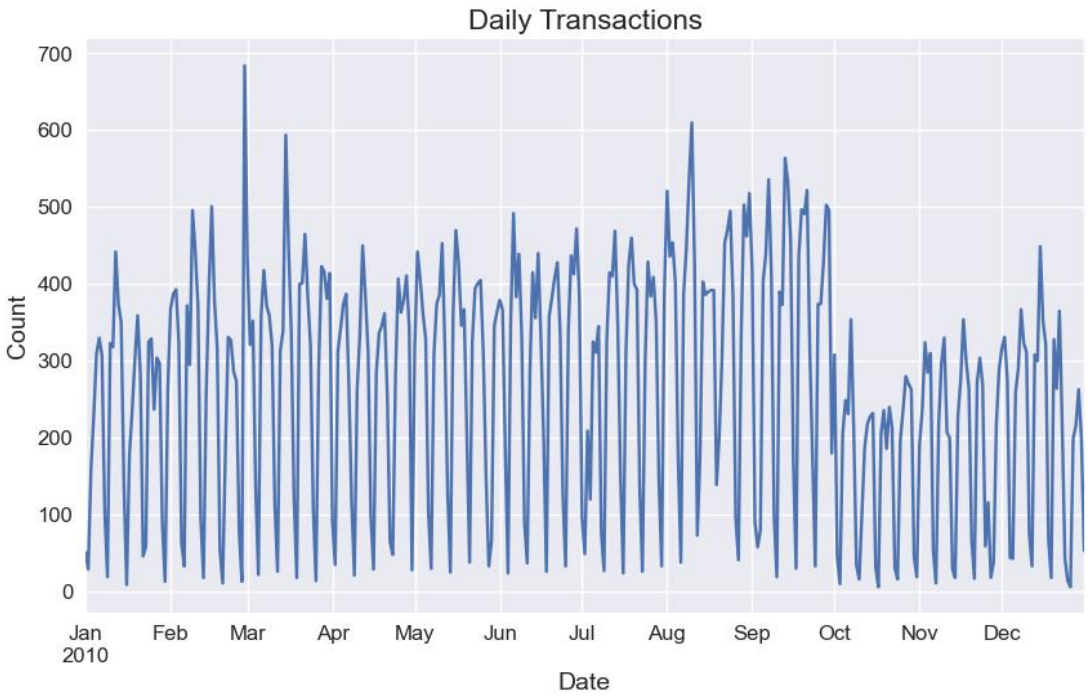


Fig. 1-1

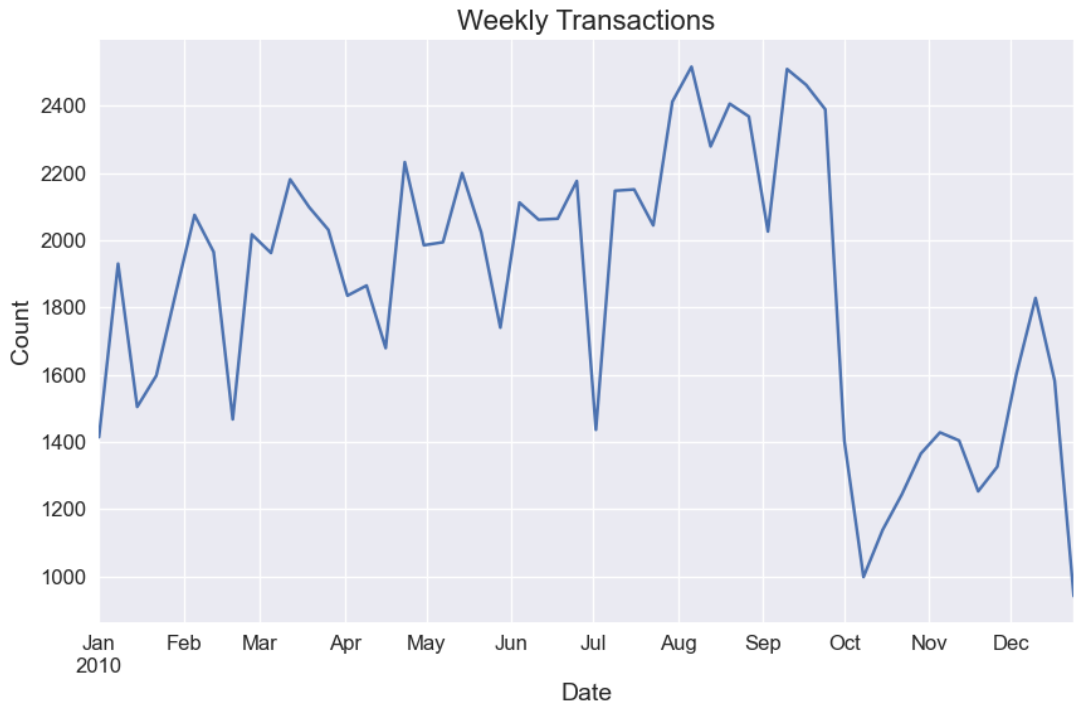


Fig. 1-2

[1] I used dt.month to get the month of each transaction, so the method is different from weekly transaction counts. Since every week has 7 days, but the number of days in different months varies, I think this method is more reasonable. You can see the code in Appendix.

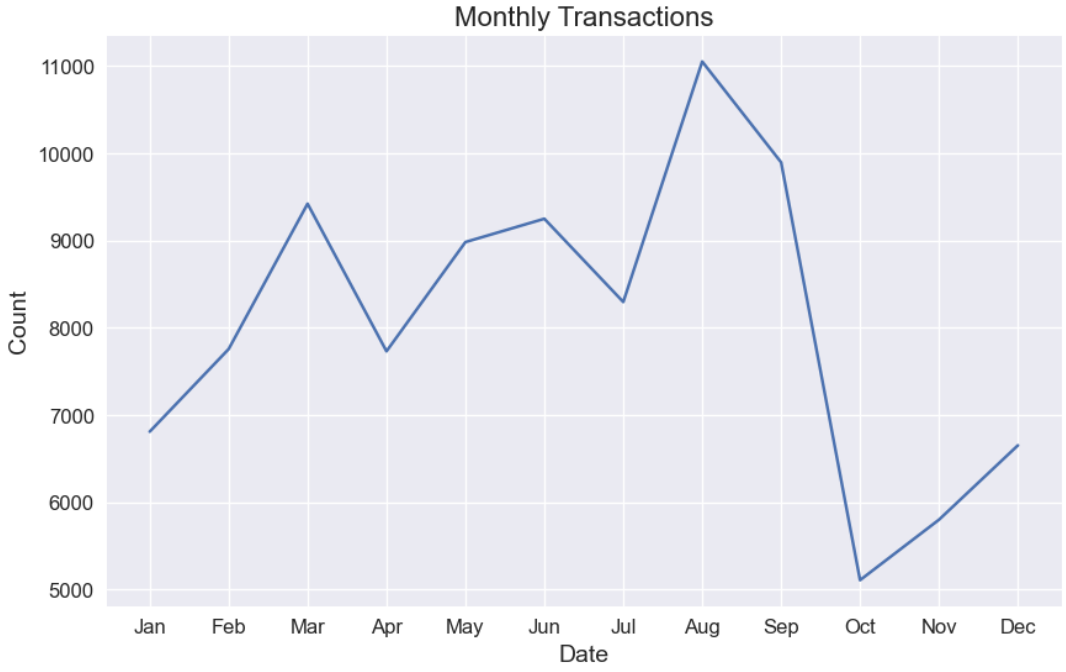
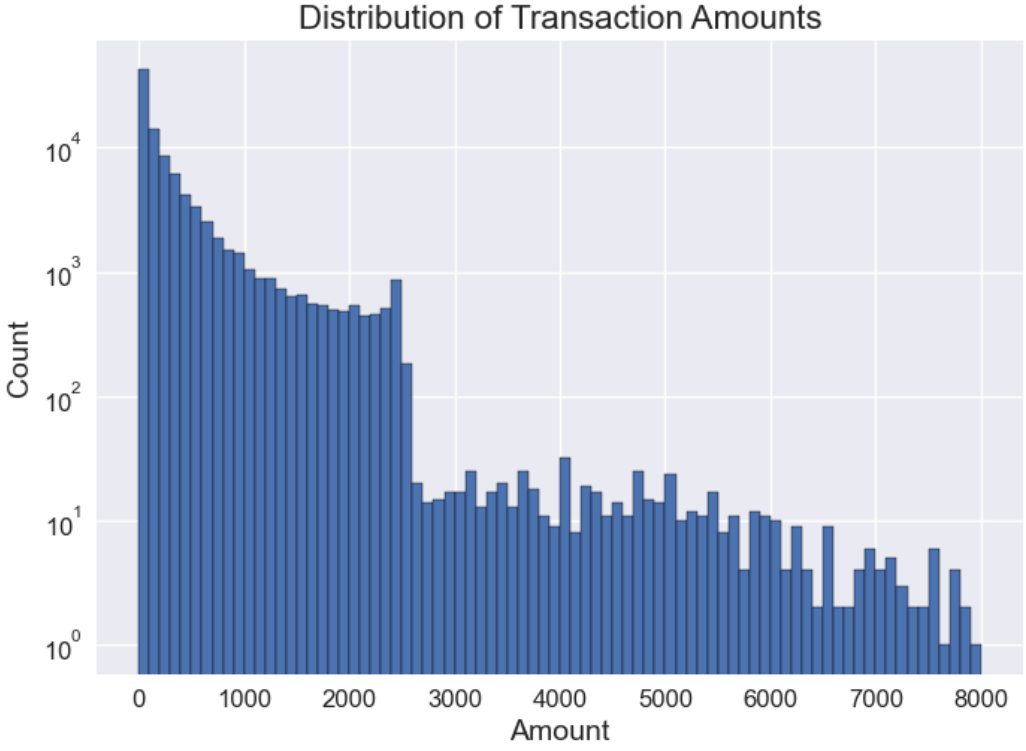


Fig. 1-3

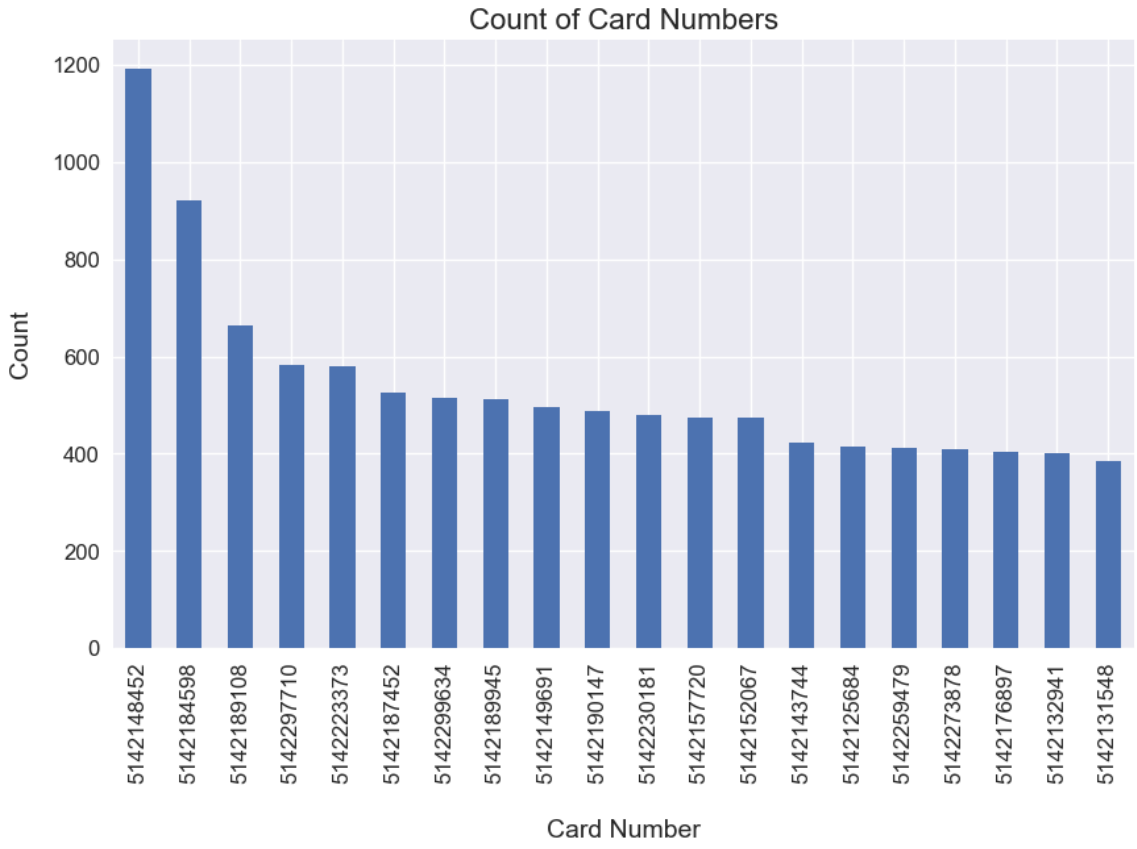
(3) Field Name: Amount

Description: Number of amounts involved in each transaction. The histogram shows the frequency distribution of the amount per transaction. Since 99.88% of all the transactions have amounts no more than $8,000, the range of the x-axis is set to be [0,8000]. According to the figure, most of the transactions were in amounts less than $2,500, and the frequency of transaction amounts showed a significant drop of around $2,500.



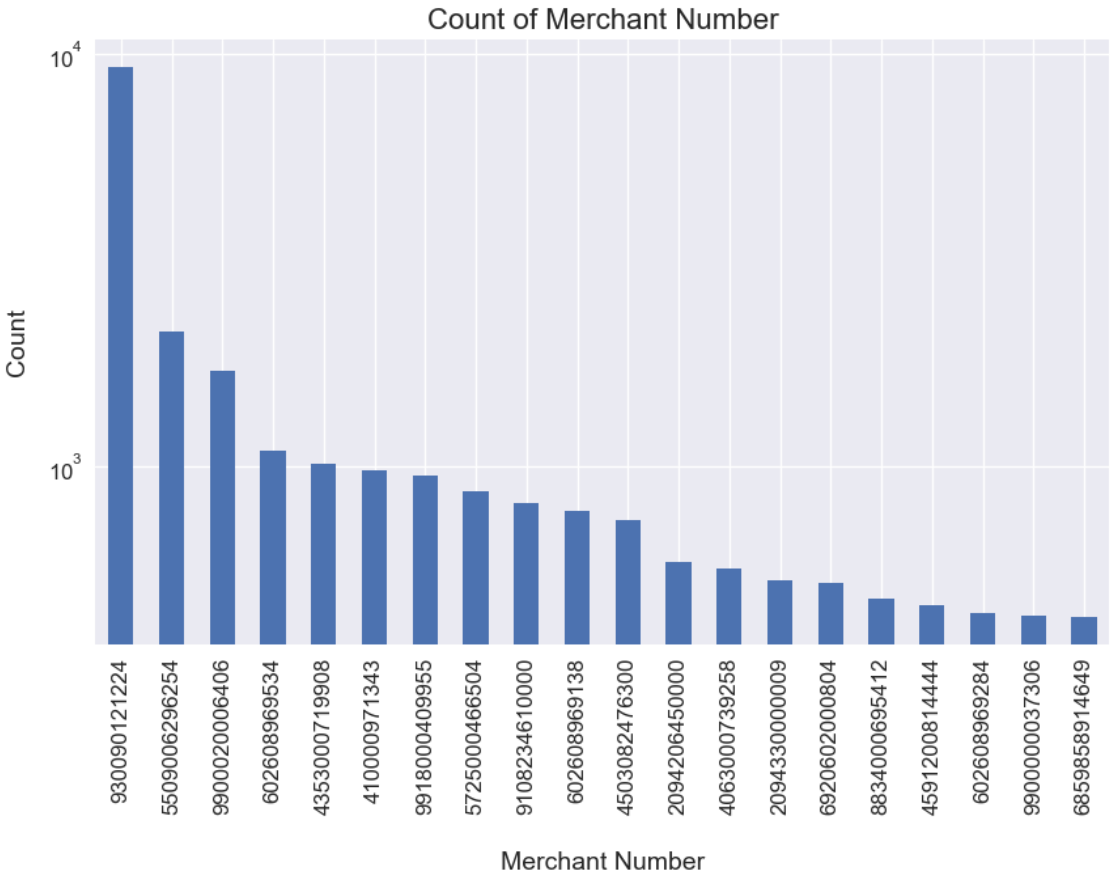
(4) Field Name: Cardnum

Description: The card numbers used for the transactions. This distribution shows the top 20 values of this field. ‘5142148452’ is the card number that has been used in most transactions in this dataset, the count of which is about 1,200.



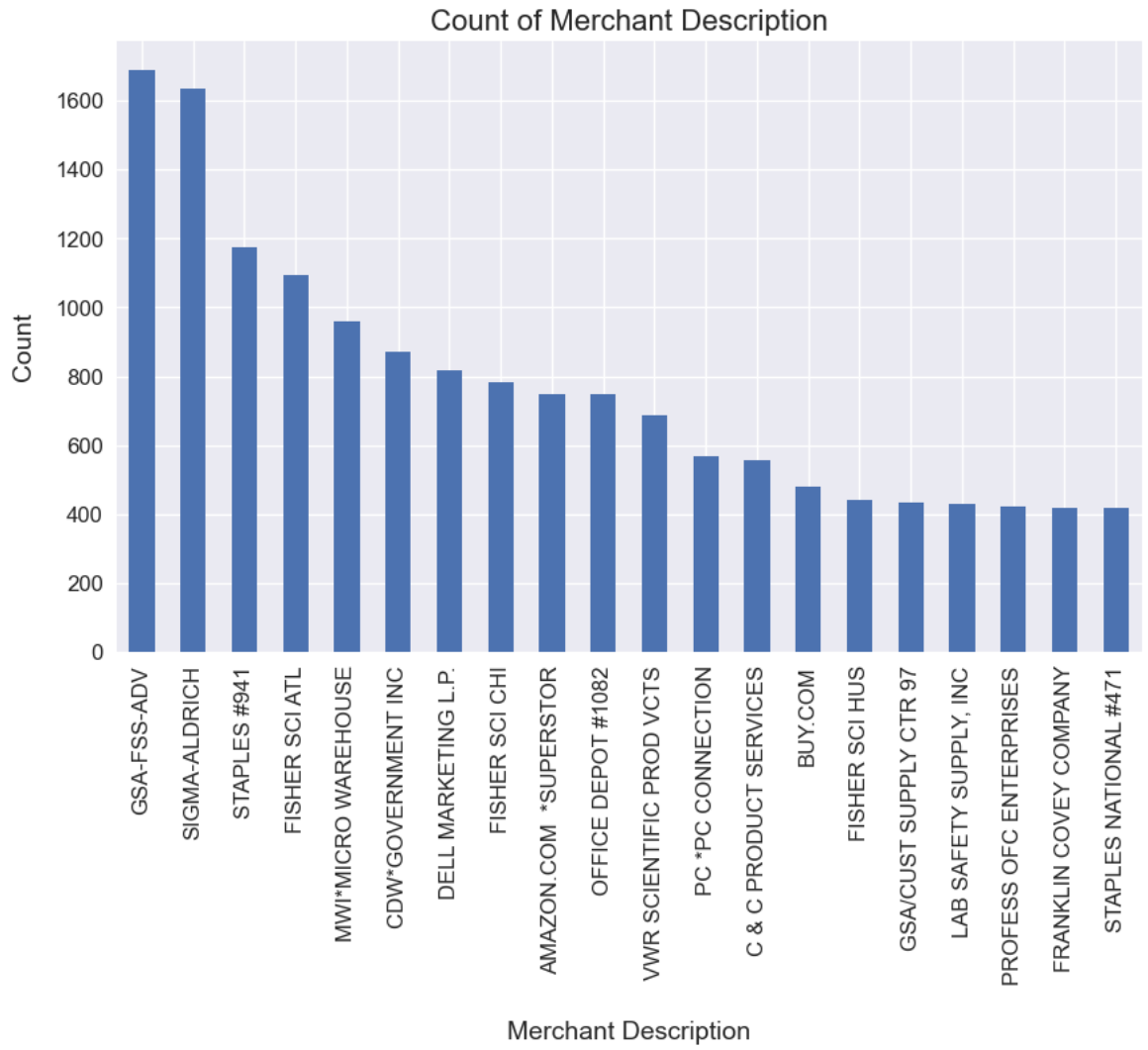
(5) Field Name: Merchnum

Description: The merchant numbers involved in the transactions, merchant number is issued by credit card processors when opening a merchant account. Notice that the 0 at the beginning of the merchant number is ignored. This distribution shows the top 20 values of this field. '930090121224' is the merchant number involved in most transactions, the count of which is close to 10,000.



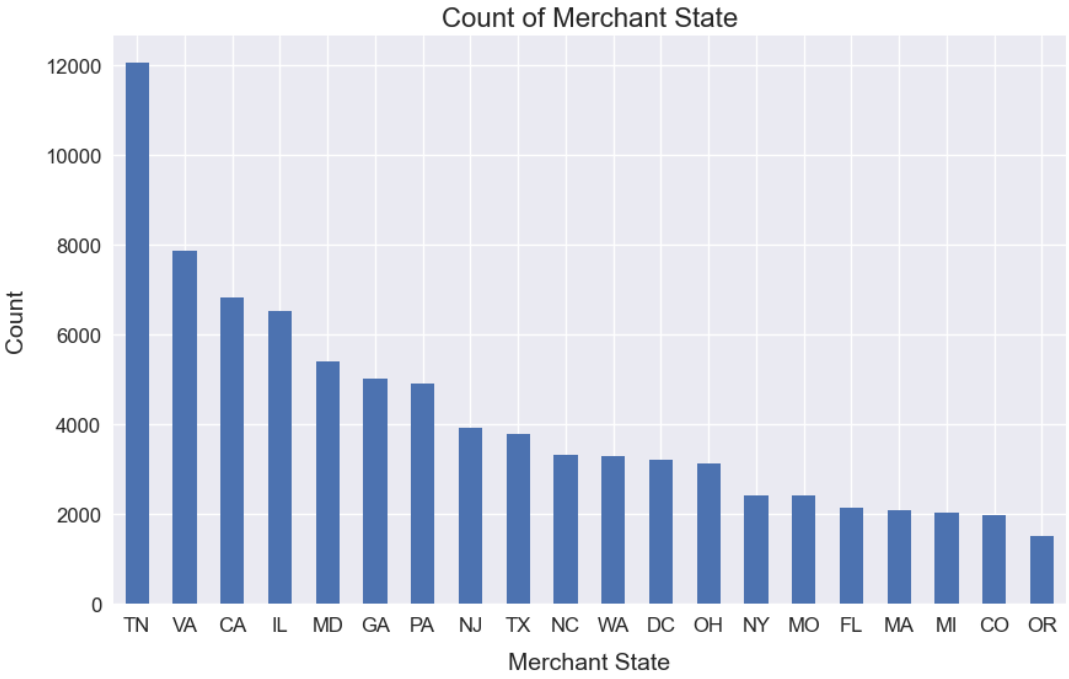
(6) Field Name: Merch description

Description: The short description of each merchant involved in the transactions. This distribution shows the top 20 values in this field. The most commonly used merch description is GSA-FSS-ADV, while the count of the second place (SIGMA-ALDRICH) has roughly the same as the first place, the counts of which are a bit over 1,600.



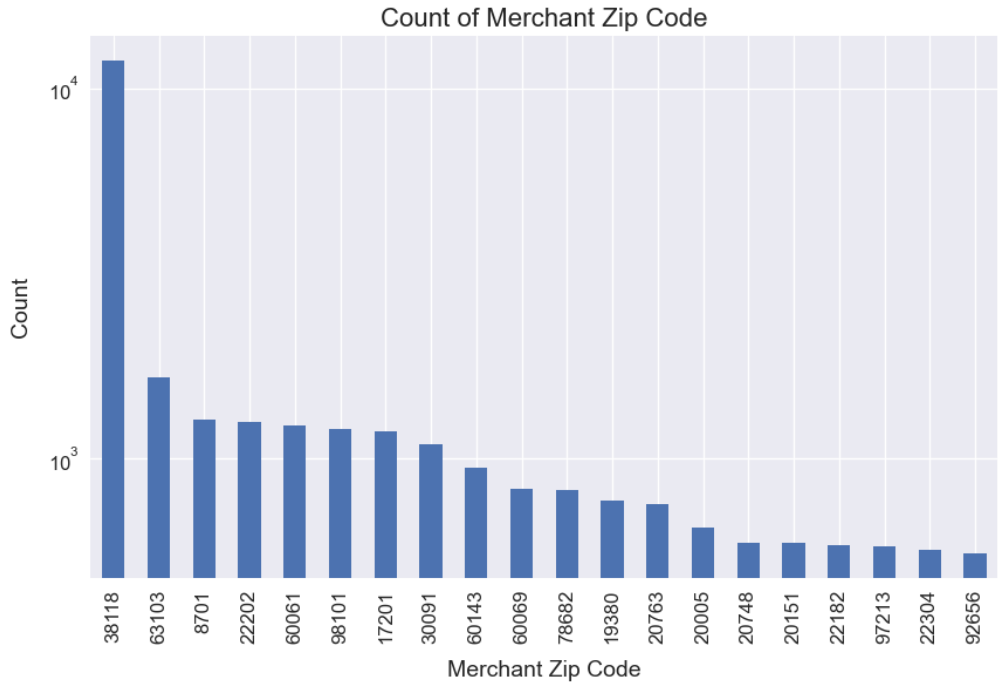
(7) Field Name: Merch state

Description: State where the merchant is located. This distribution shows the top 20 filed values. Notice that we have both abbreviations and numbers in this field to represent states, abbreviations are for states in the US and numbers are for states in Canada. TN (Tennessee) is the state that has the largest amount of merchant records in this dataset, the count of which is around 12,000.



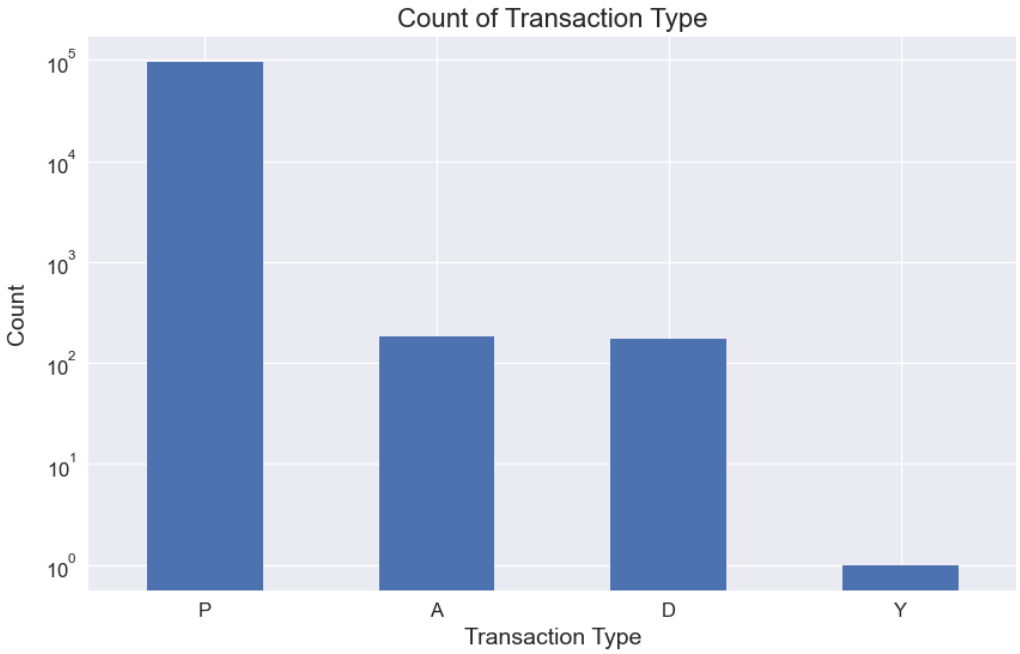
(8) Field Name: Merch zip

Description: The zip code in which a merchant is located. Notice that the 0 at the beginning of the merchant zip code is ignored. The distribution shows the top 20 values in this field. The most commonly used merchant zip code is 38118, the count of which is over 10,000.



(9) Field Name: Transtype

Description: Showing the type of a specific transaction. There are 4 different transaction types: P (Purchase), A (Authorization/ Approval), D (Debit), and Y (Year-end). This figure shows the total number of each transaction type in this dataset. The majority amount of transactions is type P (Purchase), the count of which is near 1,000,000. The numbers of type A and D transactions are roughly about 200, and there is only 1 record of type Y transaction.



(10) Field Name: Fraud

Description: The label used to tell whether a specific transaction is a fraud or not, 0 means normal transaction and 1 means fraudulent transaction. There are 1059 records labeled to be fraudulent and 95694 labeled to be normal. Therefore 1.095% of all the transaction records are fraudulent.