Appendix

I. Executive Summary

II. Data Description

III. Data cleaning

IV. Variable creation

V. Feature selection

VI. Preliminary model explores

VII. Final model performance

VIII. Financial curves and recommended cutoff

IX. Summary

I. Executive Summary

The aim of this report is to provide an overview of the machine learning model developed to detect credit card fraud in credit card transactions. With the increasing use of credit cards for transactions, the risk of fraudulent activities is also rising, causing the financial loss and reputation damage of the company. In response, the use of machine learning algorithms has become an effective approach to detect and prevent fraud in real-time. The model developed in this report utilizes supervised learning techniques to classify fraudulent transactions. The model is trained on a large dataset of credit card transactions containing both fraudulent and non-fraudulent transactions, and it is validated in the latest out of time dataset. The dataset is preprocessed to remove any outliers, filter out non-purchase transactions, and fill in missing data, resulting in a more precise dataset for modeling. The final selected supervised learning algorithm used in this model is Random Forest, which is trained on a set of features chosen after feature selections. The model was evaluated by fraud detection rate @ 3%, which means the percentage of the fraud detected by declining top 3% of the applications sorted by fraud score. The 3% threshold is chosen by balancing the total saving and the degree of stableness. Overall, the developed machine learning model shows promise in detecting and preventing credit card fraud, which can potentially save businesses and consumers from financial losses.

II. Data Description

The data is about actual credit card transactions data during the account usage process in 2010 within 12 months. The data is about company card transactions records from a US government organization in Tennessee. There are **10 fields** and **96,753 records**. However, there were no labeled frauds on this dataset, so we invented about 1,059 typical fraud records.

1.Summary Tables

(1) Categorical Table

Field name	% Populated	# Unique Values	#zeros	#blanks	Most Common Value
Recnum	100.00%	96,753	0	0	1
Cardnum	100.00%	1,645	0	0	5142148452
Merchnum	96.51%	13,091	231	3375	930090121224
Merch description	100.00%	13,126	0	0	GSA-FSS-ADV
Merch state	98.76%	227	0	1195	TN
Merch zip	95.19%	4,567	0	4656	38118.0
Transtype	100.00%	4	0	0	P
Fraud	100.00%	2	0	0	0

(2) Numerical Table

Field name	% Populated	Min	Max	Mean	Stdev	% Zero
Date	100%	2010-01-01	2010-12-31	/	/	0.00
Amount	100%	0.01	3102045.53	427.89	10006.14	0.00

II. Data Cleaning

The first step in ensuring the predictive accuracy of our future machine learning models is to clean our data. Upon investigating the details of the data, we have found that there are numerous missing fields that require filling. As such, we will begin by filtering the necessary data and then filling in any gaps in our database to maintain the integrity of our future modeling.

1. **Drop Outliers**
   To achieve this, we have identified an extreme amount value of $3,102,045.53 when sorting the amount values in descending order. This particular transaction stands out as it would cause our dataset to be heavily skewed to the right, making it an obvious outlier. Therefore, we have decided to remove it from our dataset to prevent erroneous predictions in our future modeling.

```
52714    3102045.53
47339      47900.00
59516      30372.46
80886      28392.84
89673      27218.00
Name: Amount, dtype: float64
```

2. **Filter Data**
   To concentrate solely on purchase transactions for this specific project, we have decided to filter out any transactions that are not classified as type ‘P’. Prior to filtering, our dataset contained a total of 96,752 transactions. However, by restricting the transaction type to ‘P’, we were left with 96,397 records, indicating that 355 records were eliminated through this process.

3. **Fill in Missing Values**
   Continuing further, we have observed that there are three fields in our dataset, namely Merchnum, Merch state, and Merch zip, which contain missing values.

```
Recnum                  0
Cardnum                 0
Date                    0
Merchnum             3197
Merch description       0
Merch state          1019
Merch zip            4299
Transtype               0
Amount                  0
Fraud                   0
dtype: int64
```

Our approach to fill in these missing fields is to utilize the most common value, mapped from other non-missing fields, to fill in the gaps.

a. Merchnum:

Our initial focus was on the Merchnum field, which had a total of 3,198 missing values. We also observed that the field had some 0 values, which did not make much sense in our dataset. Therefore, we considered these 0 values as null values and waited to fill them.

Upon examining the information table of the entire dataset, we noticed that each record had a valid merchant description and contained no null values. We used this field as a reference to fill in the null values. We created a dictionary to store each merchant description with its corresponding Merchnum. We then found the mode of Merchnum for each merchant description. The dictionary we built is as following:

```
[('FEDEX SHP 12/23/09 AB#', '5509006296254'),
 ('SERVICE MERCHANDISE #81', '61003026333'),
 ('OFFICE DEPOT #191', '4503082993600'),
 ('FEDEX SHP 12/28/09 AB#', '5509006296254'),
 ('FEDEX SHP 12/22/09 AB#', '5509006296254')]
```

Based on the specific description of each row with missing merchant number, we filled in the null value by the most common merchant number matching that specific description. For example, if a row has merchant description ‘SERVICE MERCHANDISE #81’ and its merchant number is missing, that null value will be replaced by ‘61003026333. However, if the row had a unique description, there would be no mode to refer to, and we treated it as an unknown merchant. In such cases, we replaced the null value with the value 'unknown'.

b. Merch state:

The Merch state field contained 1,020 missing values. To fill in these missing values, we referred to the zip code field. For merchants who had a merchant zip code but not a merchant state, we looked up the zip code of that merchant and filled in the state with the state that the specific zip code belonged to. To achieve this, we transferred the zip code to a standard format and imported an external dataset that contained all states and their corresponding zip code ranges.

We created a dictionary with the merchant description corresponding to the state name and filled in each merchant description with the mode of the corresponding merchant description. We also created a dictionary with the Merchnum corresponding to the state name and filled in each Merchnum with the mode of the corresponding Merchnum. Then, we set all state names as 'foreign unknown' for all existing zip codes that were not in the US. Lastly, we filled in the rest of the Merch state field with the value 'unknown'.

c. Merch zip:

The Merch zip field contained 4,300 missing values. We followed a similar logic as with Merchnum and Merch state to fill in these missing values. Initially, we found the most common zip code for each specific merchant state.

Based on the specific merchant state of each row with a missing merchant zip code, we filled in the null value with the mode of the corresponding Merchnum, then with the mode of the corresponding merchant description, and finally with the corresponding merchant state. Lastly, we filled in the remaining missing values in the Merch zip field with the value 'unknown'.

We also standardize the format of zipcode by right alignment.

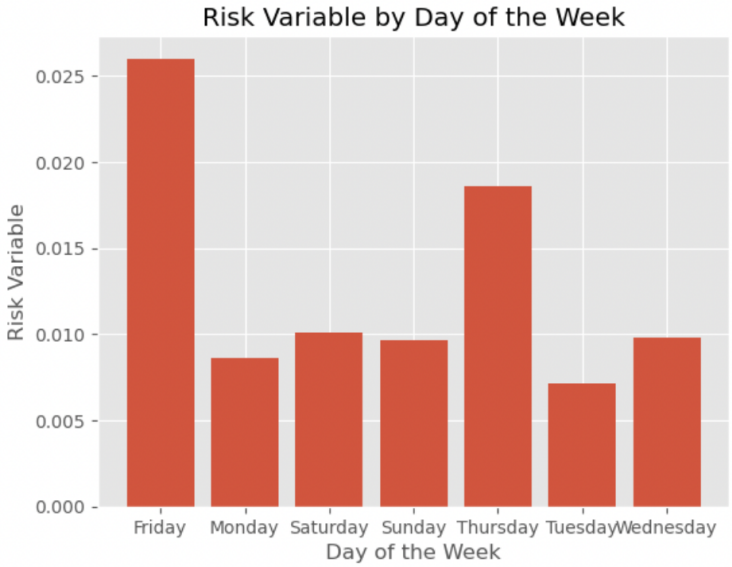
4. Target Encoding

The next step after data cleaning and outlier removal was to analyze the risk of fraud based on the day of week and state. However, using specific dates would result in too many categorical variables and limit our ability to extract meaningful insights. To address this, we extracted the day of the week from the date, resulting in only 7 categorical variables.

To further reduce dimensionality and ensure that each day and state is represented as a numerical value, we employed the target encoding method. This allowed us to create a risk table that assigned a numerical value to each day of the week and state based on their fraud risk.

To prevent overfitting, we performed statistical smoothing on the average fraud risk for each day of the week and state. The illustration of the risk table are as follows:

```
Dow
Friday       0.025994
Monday       0.008610
Saturday     0.010095
Sunday       0.009630
Thursday     0.018626
Tuesday      0.007127
Wednesday    0.009788
dtype: float64
```



The resulting risk table showed that fraud is more likely to occur on Fridays compared to other days of the week.

III. Variable Creation

1. New Entities

We created 5 new entities combining different original fields in the dataset. Below is a list of the 5 entities:

card_merch	add original field “Cardnum” and “Merchnum”
card_zip	add original field “Cardnum” and “Merch zip”
card_state	add original field “Cardnum” and “Merch state”
merch_zip	add original field “Merchnum” and “Merch zip”
merch_state	add original field “Merchnum” and “Merch state”

2. Candidate Variables

Varibale Name	Variable Description	# Variables Created
Risk("Dow_Risk")	the likelihood of fraud for any day of the week	1
Foreign tag	assign non-US states with value 1. Otherwise, assign US states with value 0	1
Benford's Law Variable	compute the unusaulness (U*) of the distribution of the first digit of purchase amount for "cardnum" and "merchnum" with smoothing and see if the distribution violates the Benford's Law	2
Day Since	number of days since a transaction with that entity has been seen. Calculated as the current date minus the date of the most recent transaction with same entity	9
Frequency	number of transactions with same entity over the past {0,1,3,7,14,30} days	172
Amount	The amount of the transaction with the same entity over the last {0,1,3,7,14,30} days. Amount is calculated in the following ways: average, maximum, median, total, actual/avergae, actual/maximum, actual/median, actual/total.	454
Average Relative Frequency Variable	The ratio of number of transactions with the same entity over the past {0,1} days to the number of transactions withat same group over the last {3,7,14, 30 days}	70
Velocity Days Since Variable	the ratio that measures how quickly an entity is changing over time, relative to the number of days since it was last observed. Calculated as the number of transactions with the entity over the past {0,1} days divided by the same entity over the past {3,7,14,30} days	54
Variability Variable	the amount difference of the transaction with the same entity over the last {0,1,3,7,14,30} days. Variability is calculated in the following ways: average, maximum, median.	162
Cross Entity Uniqueness Variable	number of unique entity cross the list of entities	72
Velocity Change Variable	number of transactions with same "Cardnum" and "Merchnum" over the past {0,1} days divided by the average daily number of transactions with same "Cardnum" and "Merchnum" over the past {3,7,14,30}days	16
Amount Bins	This splits Amount Variable into 5 equal bins	1
Unique Count Variable	number of unique entity cross the list of entities	792
Acceleration	the ratio that measures the change in the count of each entity over time, relative to the number of days between past {0,1} days and past {3,7,14,30} days. Calculated as dividing the count of the entity for the past {0,1} days by the count of the entity for the past {3,7,14,30} days, and then dividing the result by the square of the number of days between the past {0,1} days and past {3,7,14,30} days	54
Total Number of Variables	**1860**	

Entities: ‘Cardnum’,‘Merchnum’, ‘Merch description’, ‘Merch state’, ‘Merch zip’,’card_merch’,’card_zip’,’card_state’,’merch_zip’,’merch_state’

Change of variables after investigation:

- Dropped Benford’s law variable. The BL variables look at all the transactions for an entity (cardnum, merchnum). They look over all 12 months. So the BL variables on the early transactions of these entities are improperly seeing into the future, leading to over-fitting.

The table above summarizes the variables that were created to improve the detection and prevention of card transaction fraud in the dataset. For example, a new variable called 'Foreign Tag' was created, which represented if a transaction was made outside of the country. A variable called ‘Benford’s Law Variable’ represents the unusualness based on how likely the transaction amount’s first digit violates Benford’s Law. This allows us to identify unusual cases where the transaction is highly likely made up manually. Another variable called 'Frequency' was created to

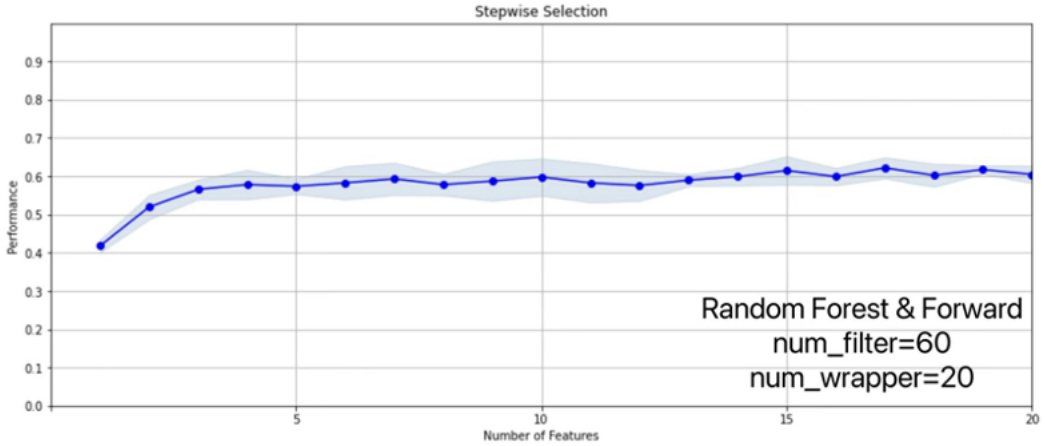
track the number of records with the same entity over a period of {0,1,3,7,14,30} days. A variable named 'Day Since' was also created, representing the number of days since an transaction with that entity was last seen. Both ‘Day Since’ and ‘Frequency’ can help our model identify unusual cases where multiple transactions are made within a short time frame. In addition, a 'Relative Velocity' variable was created to calculate the ratio of short-term velocity to longer-term averaged velocity. This also allows us to evaluate how unusual the transaction is based on its velocity. A ‘Unique Entities’ measures number of unique entities for a particular entity from {0,1,3,7,14,30} days. It can help us to detect situation like card details are stolen and fraudsters make small transactions across multiple vendors to evade large transaction alerts. A 'Max Indicators' variable was created to count the maximum or most common record for each entity in the last {0, 1,3,7,14,30,60} days. All these variables were instrumental in improving the accuracy of identifying and preventing card transaction fraud in the dataset.

IV. Feature Selection

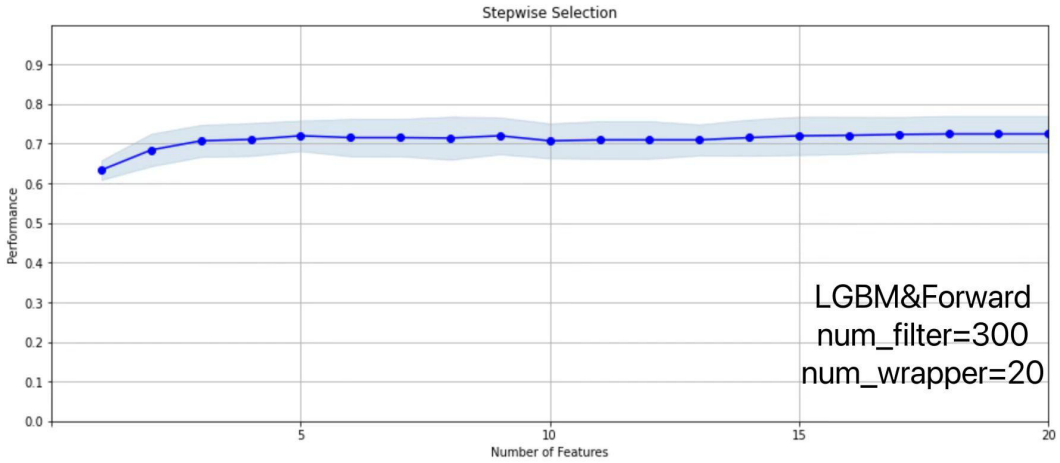
We included OOT validation set in feature selection process to decrease seasonality effect on the performance of model.

To select the best features for our model, we used two primary techniques: filtering and wrapper methods. The filtering method was used to clean the data and remove any irrelevant or redundant variables. After applying the filter, we used both forward and backward feature selection techniques to identify the top 20 variables that were most relevant to our model. Ultimately, we selected the LGBM forward feature selection technique as our final model, which yielded the best performance. The final LGBM model was configured with n_estimators=20, num_leaves=4, num_filter=300, and num_wrapper=20. The performance of the wrapper approach was approximately 0.73, indicating that the selected features were significant and contributed substantially to the model's accuracy. Our final model consisted of the top 20 features, which were ranked according to their multivariable importance.

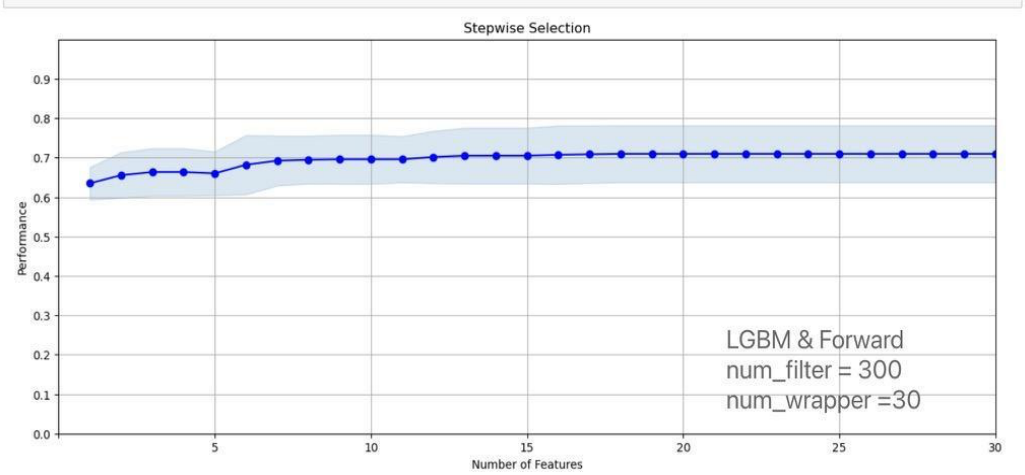
Forward feature selection:



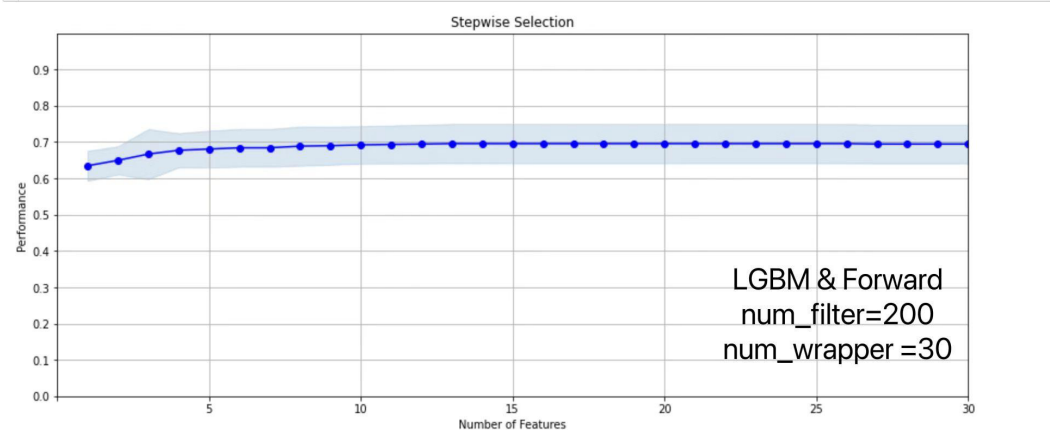
- Model: Random Forest with n_estimators=5
- num_filter = 60, num_wrapper = 20
- Runtime: ~15 minutes
- Wrapper performance: ~0.60 (not very good)



- Model: LGBM with n_estimators=20, num_leaves=4
- num_filter = 300, num_wrapper = 20
- Runtime: ~50 minutes
- Wrapper performance: ~0.73 (quite good, better than Random Forest)
- **Final choice of model**

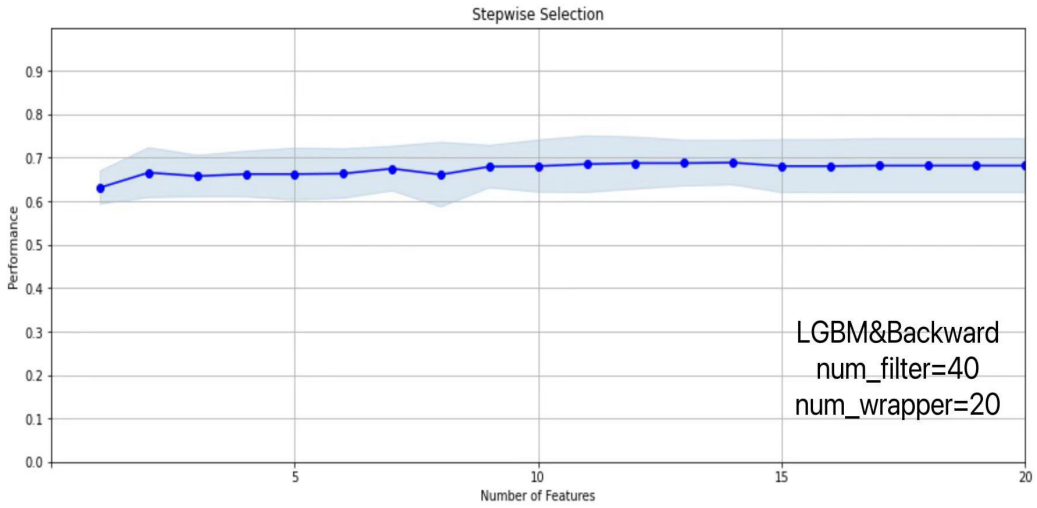


- Model: LGBM with n_estimators=20, num_leaves=3
- num_filter = 300, num_wrapper = 30
- Runtime: ~60 minutes
- Wrapper performance: ~0.72



- Model: LGBM with n_estimators=20, num_leaves=4
- num_filter = 200, num_wrapper = 30
- Runtime: ~45 minutes
- Wrapper performance: ~0.70

Backward feature selection:



- Model: LGBM with n_estimators=20, num_leaves=4
- num_filter = 40, num_wrapper = 30
- Runtime: ~20 minutes
- Wrapper performance: ~0.68

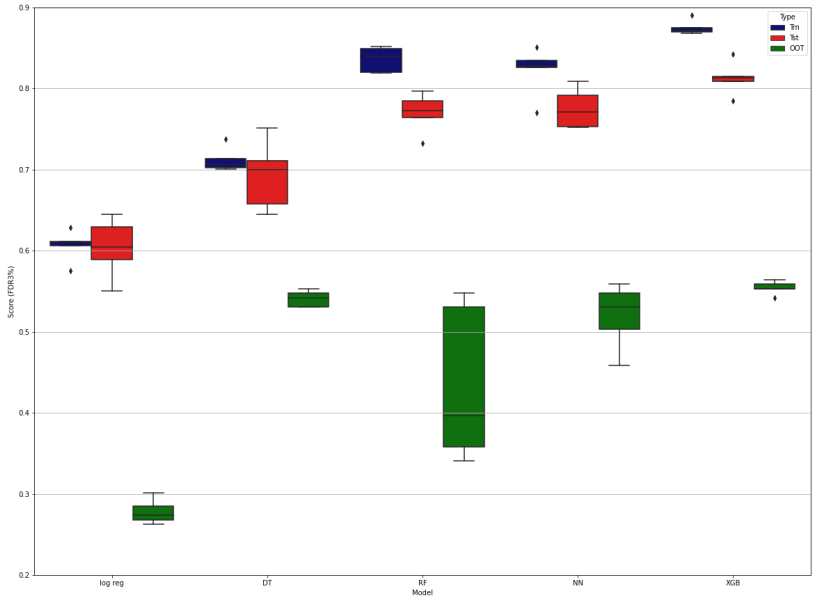
V. Preliminary Model Explores

Section I: Table of Hyperparameter Exploration

We explored different hypermeters for Logistic Regression, Decision Tree, Random Forest, Neural Net and XGBoost models. We ran each trial 5 times and averaged the results (FDR@3%) for the table(training, testing, out-of-time)

Model		Parameters						Average FDR at 3%			red color : best model performance
Logistic Regression	# variables	max_iteration	penalty	C	solver	l1_ratio		Train	Test	OOT	
1	10	20	l2	1	lbfgs	N/A		0.62	0.62	0.32	
2	10	30	l1	none	liblinear	N/A		0.63	0.63	0.31	
3	15	20	l2	1	liblinear	none		0.63	0.62	0.32	
4	15	50	l2	100	lbfgs	none		0.63	0.62	0.34	
5	20	20	l2	1	liblinear	N/A		0.61	0.59	0.28	
6	20	50	elasticnet	0.1	saga	0		0.63	0.61	0.29	
Decision Tree	# variables	max_depth	min_samples_split	min_samples_leaf	max_features	criterion		Train	Test	OOT	
1	10	5	1000	500	5	gini		0.65	0.67	0.44	underfitting
2	10	10	1000	500	10	gini		0.71	0.70	0.55	
3	15	20	1000	500	none			0.71	0.70	0.55	
4	15	10	100	30	10			0.82	0.77	0.48	
5	20	50	200	100	20	gini		0.82	0.76	0.52	
6	20	30	500	400	10	gini		0.69	0.66	0.52	
Random Forest	# variables	n_estimators	max_depth	min_samples_split	min_samples_leaf	max_features		Train	Test	OOT	
1	10	5	20	100	50	3		0.84	0.79	0.54	
2	10	50	20	2000	1000	8		0.62	0.64	0.51	
3	15	50	10	10	10	none		0.88	0.81	0.52	
4	15	20	10	30	20	5		0.53	0.52	0.51	
5	20	50	15	25	15	5		0.93	0.83	0.49	overfitting
6	20	40	10	1500	1000	10		0.62	0.63	0.51	
Neural Net	# variables	idden_layer_siz	activation	alpha	learning_rate	learning_rate_init	solver	Train	Test	OOT	
1	10	2	logistic	0.1	constant	0.1	adam	0.63	0.65	0.35	
2	10	(15,15)	logistic	0.01	constant	0.1	adam	0.70	0.71	0.44	
3	15	(20,20)	relu	0.0001	constant	0.001	lbfgs	0.83	0.77	0.48	
4	15	(5,10)	logistic	0.001	constant	0.01	adam	0.71	0.70	0.43	
5	20	(30,30)	relu	0.0001	adaptive	0.0001	lbfgs	0.87	0.80	0.43	
6	20	(25,25,25)	logistic	0.001	constant	0.001	adam	0.81	0.77	0.52	
XGBoost	# variables	max_depth	n_estimators	colsample_bytree	learning_rate	gamma	min_child_weight	Train	Test	OOT	
1	10	5	15	1	0.3	5	None	0.87	0.81	0.51	
2	10	10	30	1	0.3	3	None	0.96	0.86	0.36	overfitting
3	15	2	5	1	0.3	3	None	0.67	0.66	0.52	underfitting
4	15	100	50	0.8	0.3	20	None	0.89	0.85	0.53	
5	20	15	20	1	0.1	5	1	0.87	0.83	0.55	
6	20	80	60	0.5	0.3	25	5	0.88	0.80	0.55	

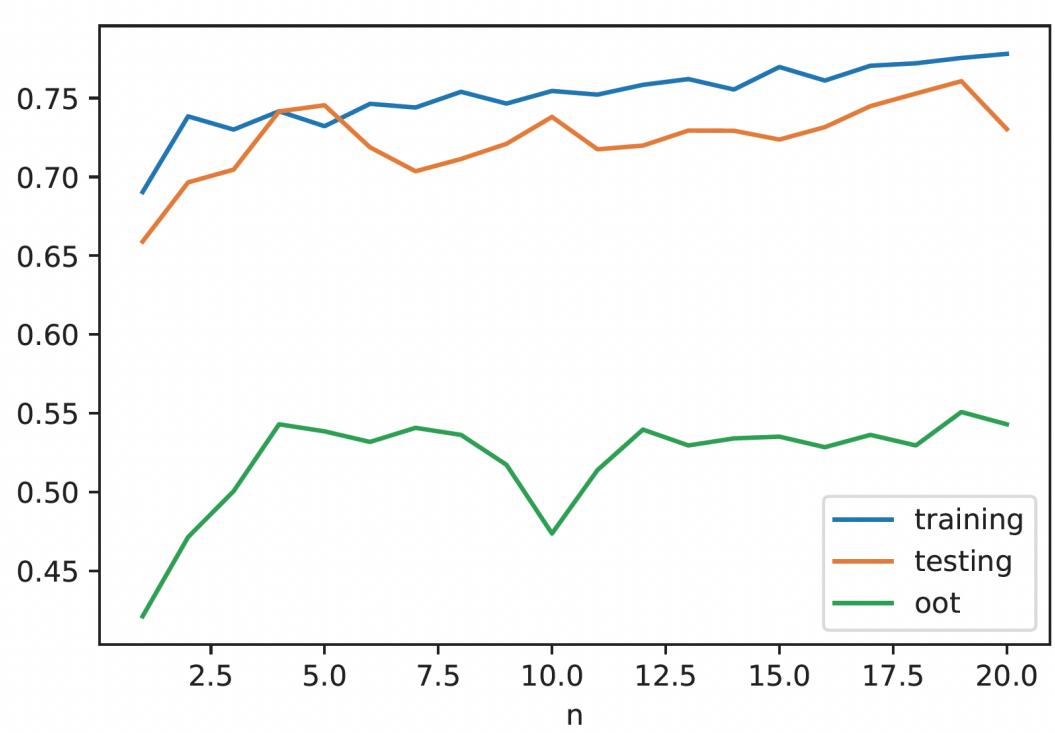
Section II: Best Parameters



Section III: Complexity

Decision Tree

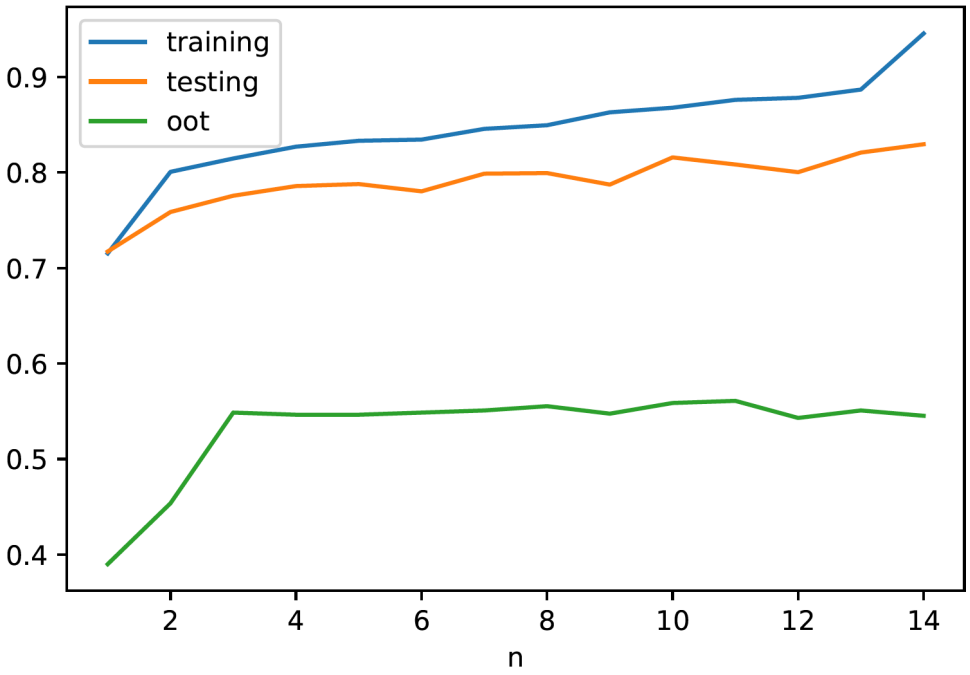
For the decision tree model, we used (criterion = 'gini',max_depth=5+i ,min_samples_split= 500-i,min_samples_leaf=200-i,max_features = 5+i) with i in range(1,100,5). We wanted to make i as large as possible to show the overfitting, and with the graph, we can see that as the model complexity increase, the training performance is getting better while the testing and oot have the tendency to decrease.



We increase the models' complexity for 20 time to observe the overfitting.

Random Forest

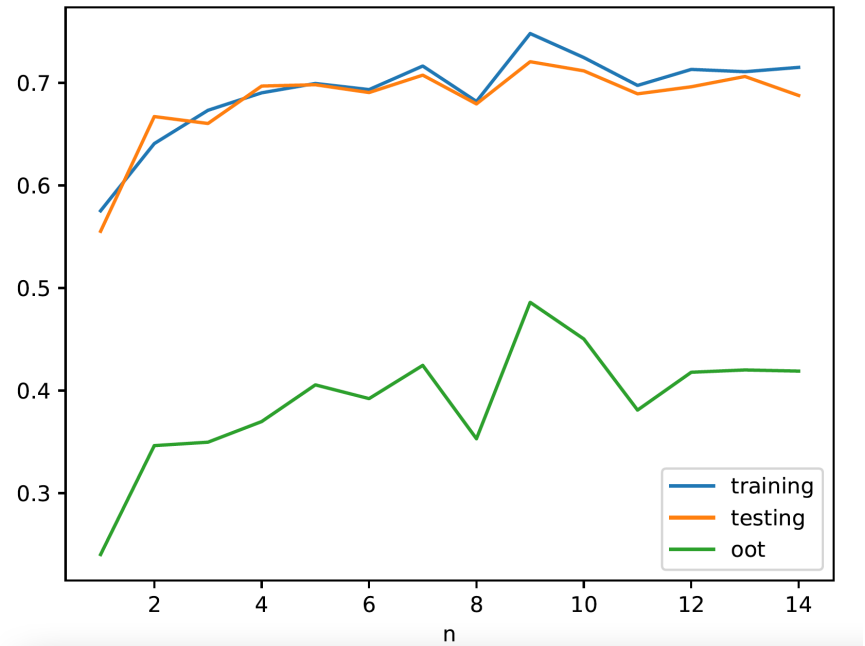
For random forest model, we used criterion = 'entropy',n_estimators=5*i, max_depth=5*i, max_features=3*i, min_samples_split = 300-i*10, min_samples_leaf = 150-i*10 while i within the range of(1,15,1). we believe with increase in [max-depth, n_estimators, max_features] and a decrease in ['min_sample_split',min_sample_leaf], the random forest model will increase complexity, resulting in overfitting.



We increase the models' complexity for 14 time to observe the overfitting.

Neural Net

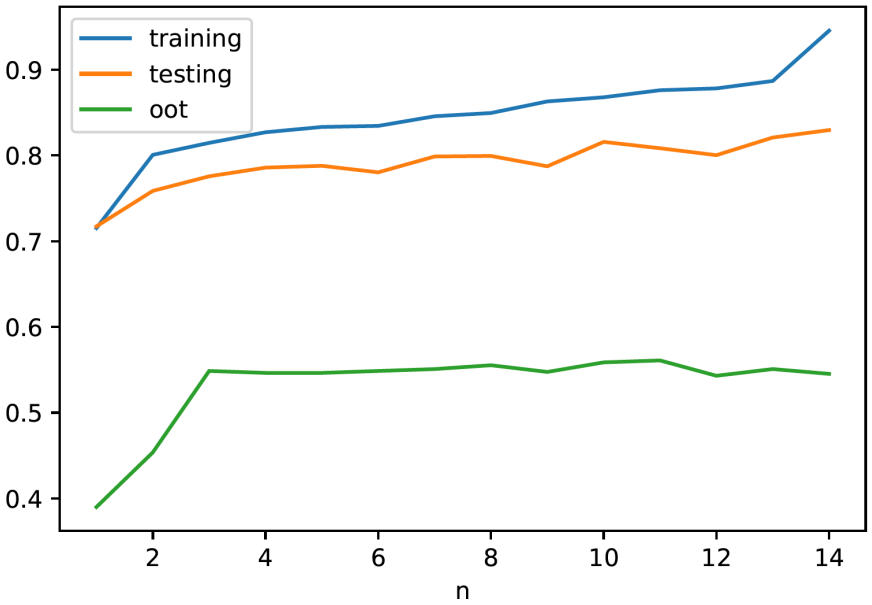
For the Neural Net model, we used MLPClassifier(hidden_layer_sizes=(20,20,),activation='logistic',alpha= math.exp(-i),solver='adam',learning_rate='constant') while i within the range of(1,15,1)



We increase the models' complexity for 14 time to observe the overfitting.

Random Forest

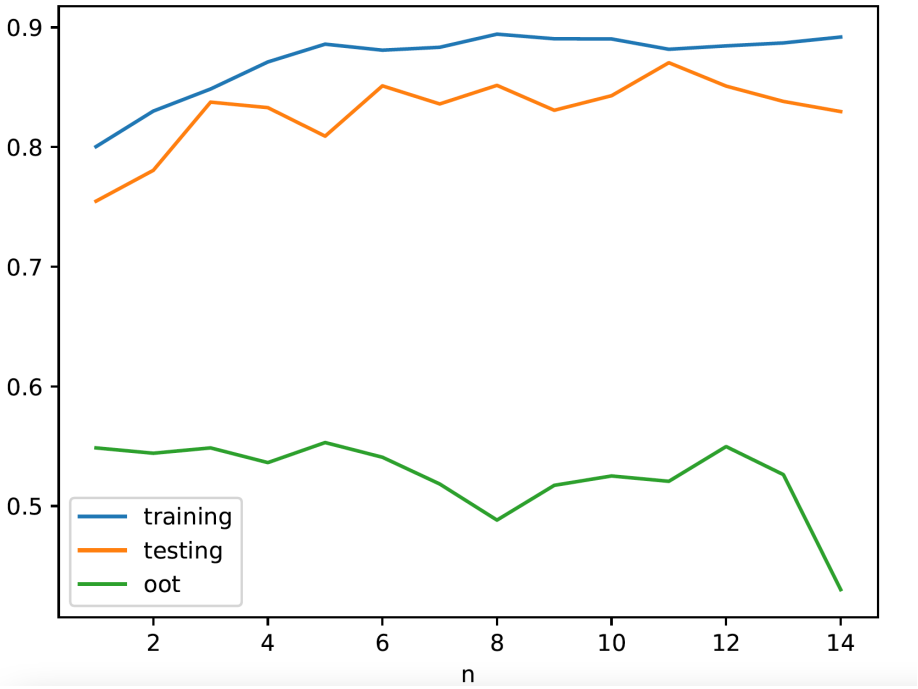
For the random forest model, we used model = RandomForestClassifier(criterion = 'entropy',n_estimators=5*i,max_depth=5*i,max_features=3*i,min_samples_split = 300-i*10, min_samples_leaf = 150-i*10 model = RandomForestClassifier(criterion = 'entropy',n_estimators=5*i,max_depth=5*i,max_features=3*i,min_samples_split = 300-i*10, min_samples_leaf = 150-i*10



We increase the models' complexity for 14 time to observe the overfitting.

XGBoost

For the XGBoost model, we used xgb.XGBClassifier(colsample_bytree = 0.8,max_depth= 50, n_estimators=5*i,learning_rate=0.3,gamma=20) while i within the range of(1,15,1)



We increase the models' complexity for 14 time to observe the overfitting.

VI. Final Model Performance

The final model is a random forest. The optimal hyperparameters that the model use is listed below:

model = RandomForestClassifier(criterion = 'entropy',n_estimators=5,max_depth=20,max_features=3,min_samples_split= 100,min_samples_leaf = 50)

Also, we use top 10 variables from feature selection for our final models.

The training, testing, and out-of-time result is listed as following:

Training:

Training	# Records	# Goods	# Bads	Fraud Rate
	59010	58393	617	0.01045585

Bin Statistics						Cumulative Statistics						
Population Bin%	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads(FDR)	KS	FPR
1	590	245	345	41.53%	58.47%	590	245	345	0.42%	55.92%	55.50	0.71
2	590	445	145	75.42%	24.58%	1180	690	490	1.18%	79.42%	78.23	1.41
3	590	536	54	90.85%	9.15%	1770	1226	544	2.10%	88.17%	86.07	2.25
4	590	572	18	96.95%	3.05%	2360	1798	562	3.08%	91.09%	88.01	3.20
5	590	578	12	97.97%	2.03%	2950	2376	574	4.07%	93.03%	88.96	4.14
6	591	579	12	97.97%	2.03%	3541	2955	586	5.06%	94.98%	89.92	5.04
7	590	585	5	99.15%	0.85%	4131	3540	591	6.06%	95.79%	89.72	5.99
8	590	584	6	98.98%	1.02%	4721	4124	597	7.06%	96.76%	89.70	6.91
9	590	585	5	99.15%	0.85%	5311	4709	602	8.06%	97.57%	89.50	7.82
10	590	585	5	99.15%	0.85%	5901	5294	607	9.07%	98.38%	89.31	8.72
11	590	589	1	99.83%	0.17%	6491	5883	608	10.07%	98.54%	88.47	9.68
12	590	588	2	99.66%	0.34%	7081	6471	610	11.08%	98.87%	87.78	10.61
13	590	588	2	99.66%	0.34%	7671	7059	612	12.09%	99.19%	87.10	11.53
14	590	588	2	99.66%	0.34%	8261	7647	614	13.10%	99.51%	86.42	12.45
15	591	590	1	99.83%	0.17%	8852	8237	615	14.11%	99.68%	85.57	13.39
16	590	590	0	100.00%	0.00%	9442	8827	615	15.12%	99.68%	84.56	14.35
17	590	588	2	99.66%	0.34%	10032	9415	617	16.12%	100.00%	83.88	15.26
18	590	590	0	100.00%	0.00%	10622	10005	617	17.13%	100.00%	82.87	16.22
19	590	590	0	100.00%	0.00%	11212	10595	617	18.14%	100.00%	81.86	17.17
20	590	590	0	100.00%	0.00%	11802	11185	617	19.15%	100.00%	80.85	18.13

Testing:

Testing	# Records	# Goods	# Bads	Fraud Rate
	25290	25027	263	0.01039937

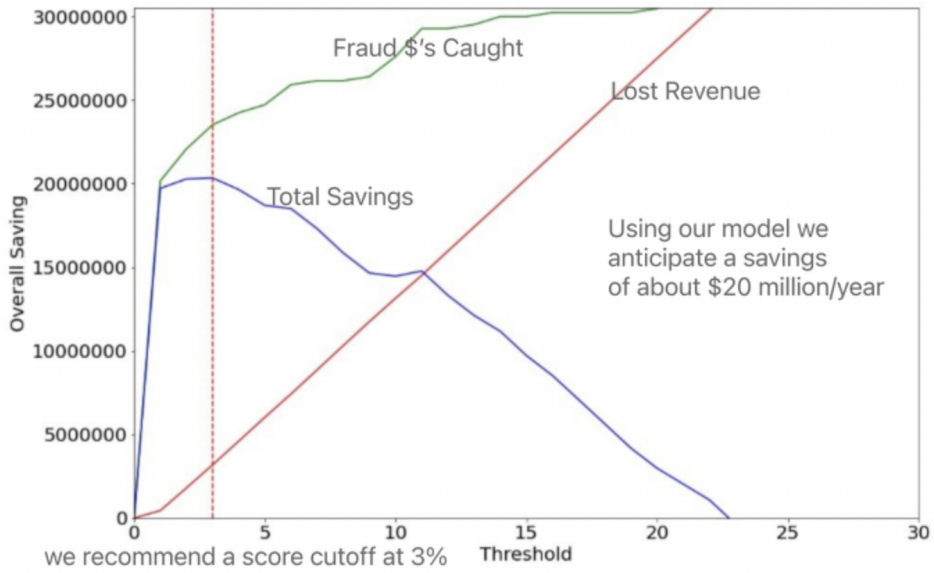
Bin Statistics						Cumulative Statistics						
Population Bin%	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads(FDR)	KS	FPR
1	253	117	136	46.25%	53.75%	253	117	136	0.47%	51.71%	51.24	0.86
2	253	202	51	79.84%	20.16%	506	319	187	1.27%	71.10%	69.83	1.71
3	253	232	21	91.70%	8.30%	759	551	208	2.20%	79.09%	76.89	2.65
4	253	248	5	98.02%	1.98%	1012	799	213	3.19%	80.99%	77.80	3.75
5	252	246	6	97.62%	2.38%	1264	1045	219	4.18%	83.27%	79.09	4.77
6	253	250	3	98.81%	1.19%	1517	1295	222	5.17%	84.41%	79.24	5.83
7	253	250	3	98.81%	1.19%	1770	1545	225	6.17%	85.55%	79.38	6.87
8	253	251	2	99.21%	0.79%	2023	1796	227	7.18%	86.31%	79.14	7.91
9	253	248	5	98.02%	1.98%	2276	2044	232	8.17%	88.21%	80.05	8.81
10	253	251	2	99.21%	0.79%	2529	2295	234	9.17%	88.97%	79.80	9.81
11	253	251	2	99.21%	0.79%	2782	2546	236	10.17%	89.73%	79.56	10.79
12	253	252	1	99.60%	0.40%	3035	2798	237	11.18%	90.11%	78.93	11.81
13	253	252	1	99.60%	0.40%	3288	3050	238	12.19%	90.49%	78.31	12.82
14	253	253	0	100.00%	0.00%	3541	3303	238	13.20%	90.49%	77.30	13.88
15	253	252	1	99.60%	0.40%	3794	3555	239	14.20%	90.87%	76.67	14.87
16	252	252	0	100.00%	0.00%	4046	3807	239	15.21%	90.87%	75.66	15.93
17	253	250	3	98.81%	1.19%	4299	4057	242	16.21%	92.02%	75.80	16.76
18	253	253	0	100.00%	0.00%	4552	4310	242	17.22%	92.02%	74.79	17.81
19	253	252	1	99.60%	0.40%	4805	4562	243	18.23%	92.40%	74.17	18.77
20	253	253	0	100.00%	0.00%	5058	4815	243	19.24%	92.40%	73.16	19.81

Out of Time:

OOT	# Records	# Goods	# Bads	Fraud Rate								
	12097	11918	179	0.01479706								
Bin Statistics						Cumulative Statistics						
Population Bin%	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads(FDR)	KS	FPR
1	121	37	84	30.58%	69.42%	121	37	84	0.31%	46.93%	46.62	0.44
2	121	113	8	93.39%	6.61%	242	150	92	1.26%	51.40%	50.14	1.63
3	121	115	6	95.04%	4.96%	363	265	98	2.22%	54.75%	52.53	2.70
4	121	118	3	97.52%	2.48%	484	383	101	3.21%	56.42%	53.21	3.79
5	121	119	2	98.35%	1.65%	605	502	103	4.21%	57.54%	53.33	4.87
6	121	116	5	95.87%	4.13%	726	618	108	5.19%	60.34%	55.15	5.72
7	121	120	1	99.17%	0.83%	847	738	109	6.19%	60.89%	54.70	6.77
8	121	121	0	100.00%	0.00%	968	859	109	7.21%	60.89%	53.69	7.88
9	121	120	1	99.17%	0.83%	1089	979	110	8.21%	61.45%	53.24	8.90
10	121	116	5	95.87%	4.13%	1210	1095	115	9.19%	64.25%	55.06	9.52
11	121	114	7	94.21%	5.79%	1331	1209	122	10.14%	68.16%	58.01	9.91
12	121	121	0	100.00%	0.00%	1452	1330	122	11.16%	68.16%	57.00	10.90
13	121	120	1	99.17%	0.83%	1573	1450	123	12.17%	68.72%	56.55	11.79
14	121	119	2	98.35%	1.65%	1694	1569	125	13.16%	69.83%	56.67	12.55
15	121	121	0	100.00%	0.00%	1815	1690	125	14.18%	69.83%	55.65	13.52
16	121	120	1	99.17%	0.83%	1936	1810	126	15.19%	70.39%	55.20	14.37
17	120	120	0	100.00%	0.00%	2056	1930	126	16.19%	70.39%	54.20	15.32
18	121	121	0	100.00%	0.00%	2177	2051	126	17.21%	70.39%	53.18	16.28
19	121	121	0	100.00%	0.00%	2298	2172	126	18.22%	70.39%	52.17	17.24
20	121	120	1	99.17%	0.83%	2419	2292	127	19.23%	70.95%	51.72	18.05

Based on the above statistics, we can see that the model is not overfitting, and the result is pretty good. According to OOT FDR@3%, we know that our model will eliminate about 54.7% of the fraud by rejecting only 3% of the transactions.

VII. Financial Curves and recommended cutoff



Green line is the Fraud amount caught (dollar), the red line is the lost revenue, the blue line is the overall savings. We recommend using 3% as the threshold, where we achieve 0.547 FDR and

max saving at $20,340,000. Also, the point at 3% lies on a flat line, meaning that the point is stable and a small change would not influence our saving.

VIII. Summary

Initial data analysis was performed on company card transaction records from a US government organization in Tennessee, consisting of 10 fields and 96,753 records. To address the lack of labeled frauds, 1,059 typical fraud records were added. Outliers with extreme amount value of $3,102,045.53 were dropped, and transaction type was restricted to 'P', resulting in 96,397 records. Missing values were imputed with the most common value mapped from non-missing fields. Target encoding was used to create a risk table assigning a numerical value to each day of the week and state based on their fraud risk, revealing a higher likelihood of fraud on Fridays.

Five new entities were created by combining different original fields, and variables were dropped from the relative frequency and variability classes to avoid data leakage and overfitting. Feature selection was performed using filtering and wrapper methods, with LGBM forward feature selection ultimately selected as the final model. The preliminary model exploration involved evaluating different hyperparameters for Logistic Regression, Decision Tree, Random Forest, Neural Net, and XGBoost models.

The final model selected was a Random Forest, with optimal hyperparameters including n_estimators=5, max_depth=20, max_features=3, min_samples_split=100, and min_samples_leaf=50. Using a 3% threshold, an FDR of 0.547 was achieved, resulting in maximum savings of $20,340,000. The stability of the point at 3% indicates that small changes would not significantly affect the results. That would being said, if the company classified the top 3% of population sorting by fraud score as frauds, it can catch 54.7% of frauds and save 20,340,000 dollar annually by using our algorithm.

Appendix

Data Quality Report

The data is about actual credit card transactions data during the account usage process in 2010 within 12 months. The data is about company card transactions records from a US government organization in Tennessee. There are **10 fields** and **96,753 records**. However, there were no labeled frauds on this dataset, so we invented about 1,059 typical fraud records.

1.Summary Tables

Categorical Table

Field name	% Populated	# Unique Values	#zeros	#blanks	Most Common Value
Recnum	100.00%	96,753	0	0	1
Cardnum	100.00%	1,645	0	0	5142148452
Merchnum	96.51%	13,091	231	3375	930090121224
Merch description	100.00%	13,126	0	0	GSA-FSS-ADV
Merch state	98.76%	227	0	1195	TN
Merch zip	95.19%	4,567	0	4656	38118.0
Transtype	100.00%	4	0	0	P
Fraud	100.00%	2	0	0	0

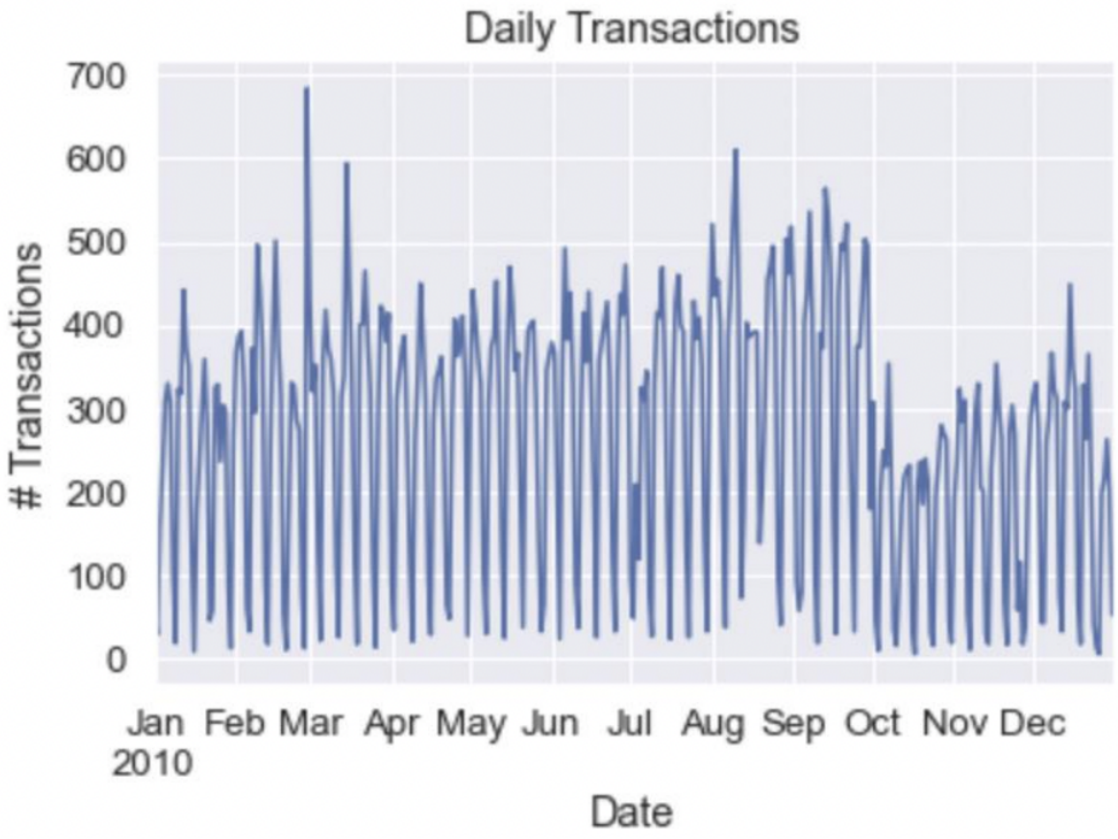
Numerical Table

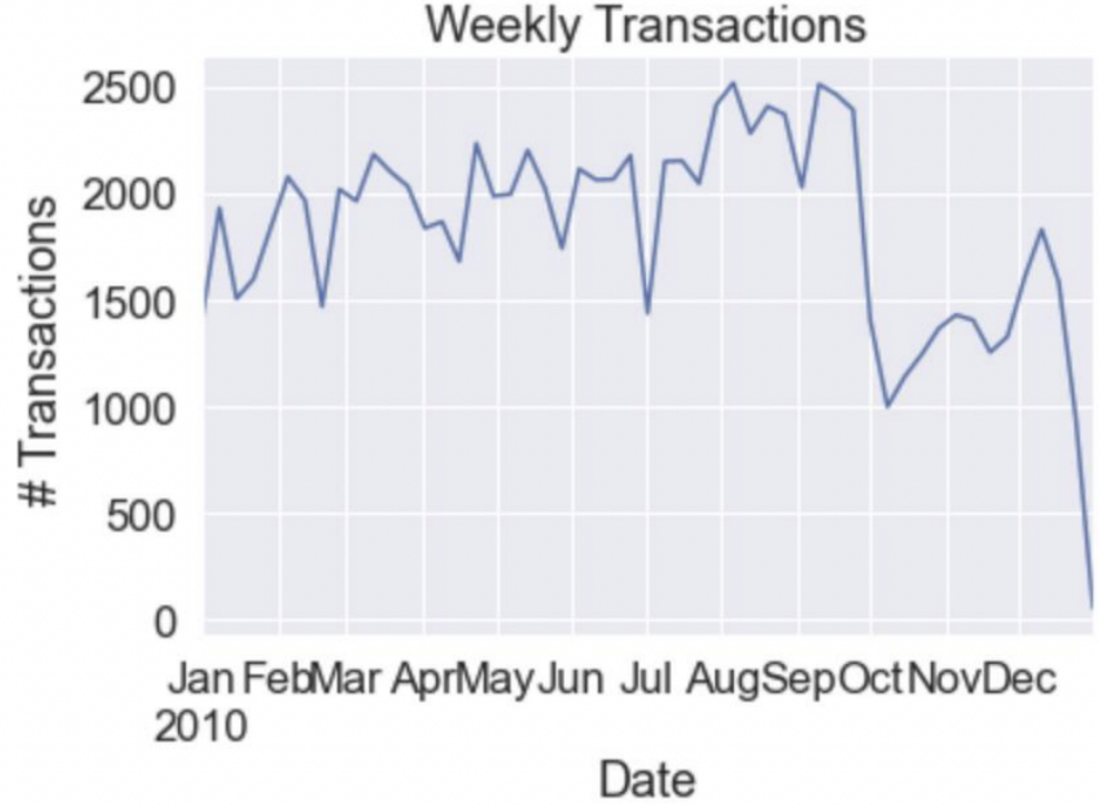
Field name	% Populated	Min	Max	Mean	Stdev	% Zero
Date	100%	2010-01-01	2010-12-31	/	/	0.00
Amount	100%	0.01	3102045.53	427.89	10006.14	0.00

2. Visualization of Each Field

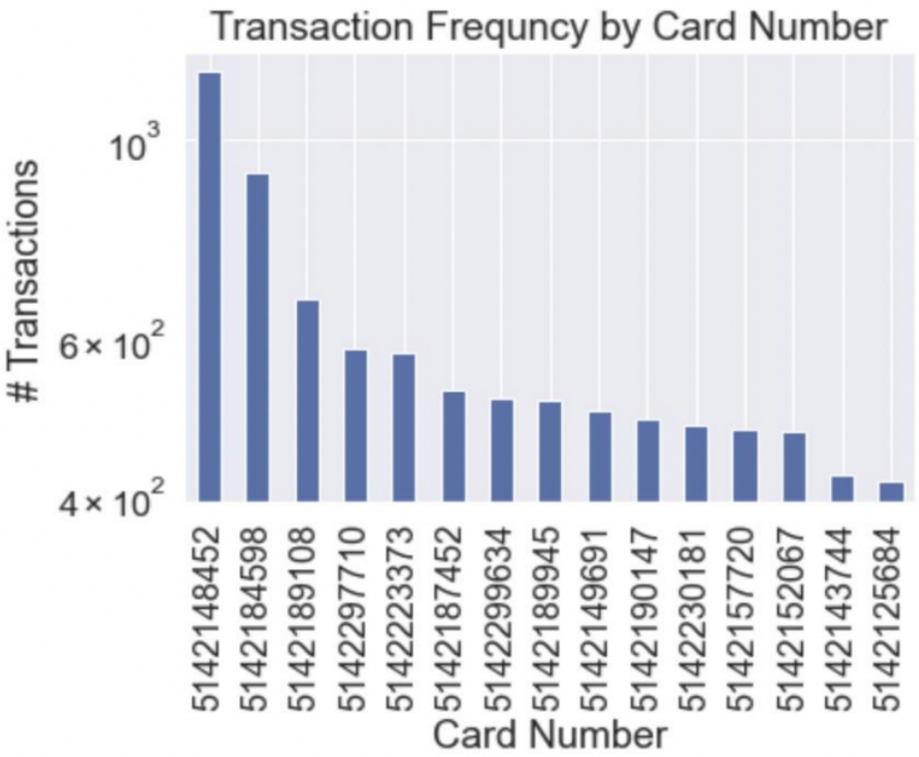
(1) Field Name: Recnum
Description: Record Number. Ordinal unique positive integer for each record, from 1 to 96,753.

(2) Field Name: Date
Description: a numerical field, the date of the transaction. It starts from 2010-01-01 to 2010-12-31. It can be grouped by daily applications or weekly applications, as shown below; The first distribution shows the daily transaction distribution across time from January 2010 to December 2010. The greatest amount of daily transaction is 684, which occurs on 2010-02-28. The second distribution shows the weekly application distribution. The greatest amount of weekly application is 2516, which occurs on the first week of August. There're 365 unique values in this field.

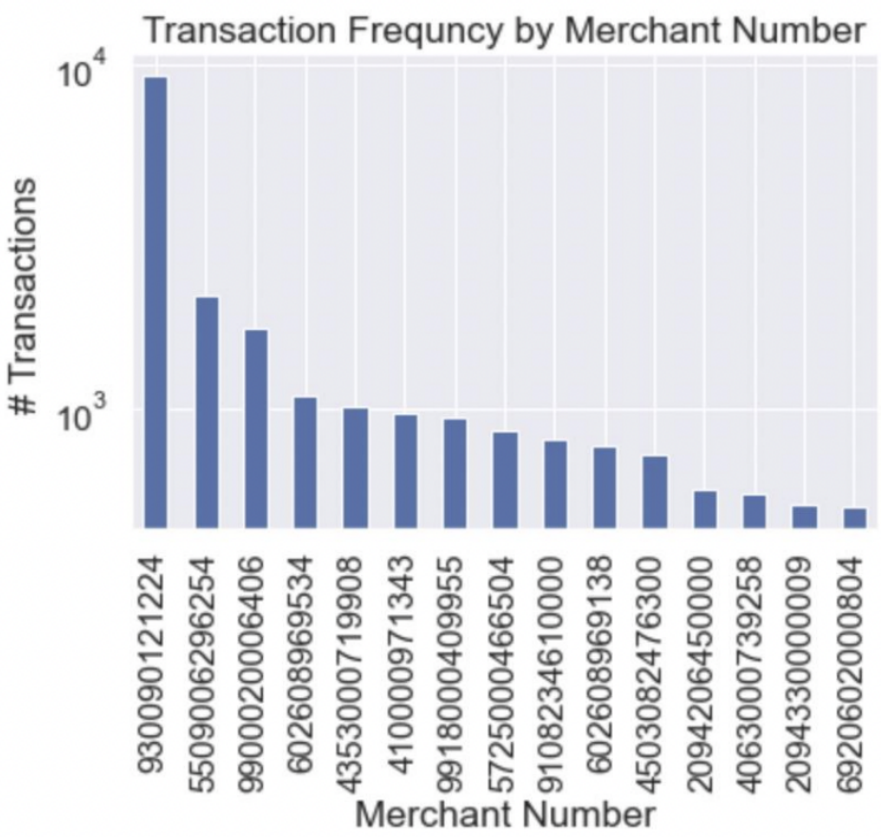

Daily Transactions
Transactions
700
600
500
400
300
200
100
0
Jan
2010
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Date


Weekly Transactions
Transactions
2500
2000
1500
1000
500
0
Jan
2010
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Date

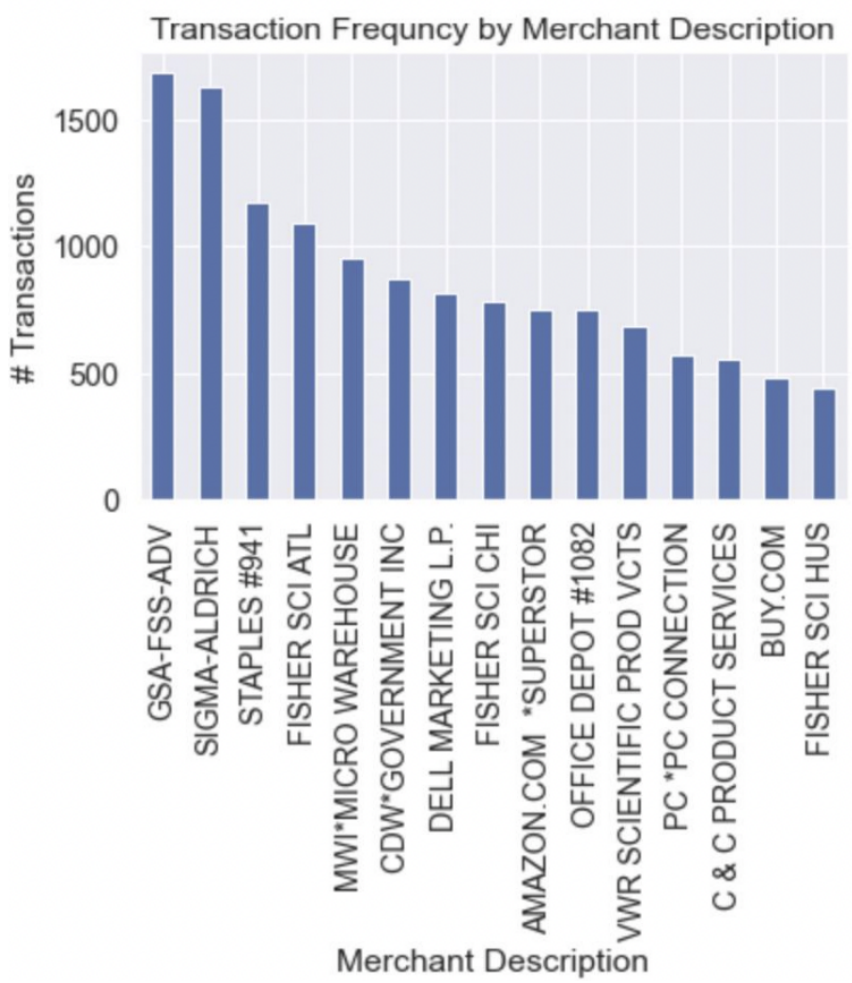
(3) Field Name: Cardnum
Description: a categorical field, the card number used for each transaction in the dataset. The distribution is the top 15 field values of card numbers. The most frequently appearing card number is 5148148425, count is 1,192. There're 1,645 unique values among the 96,753 transactions in this field, no missing value.



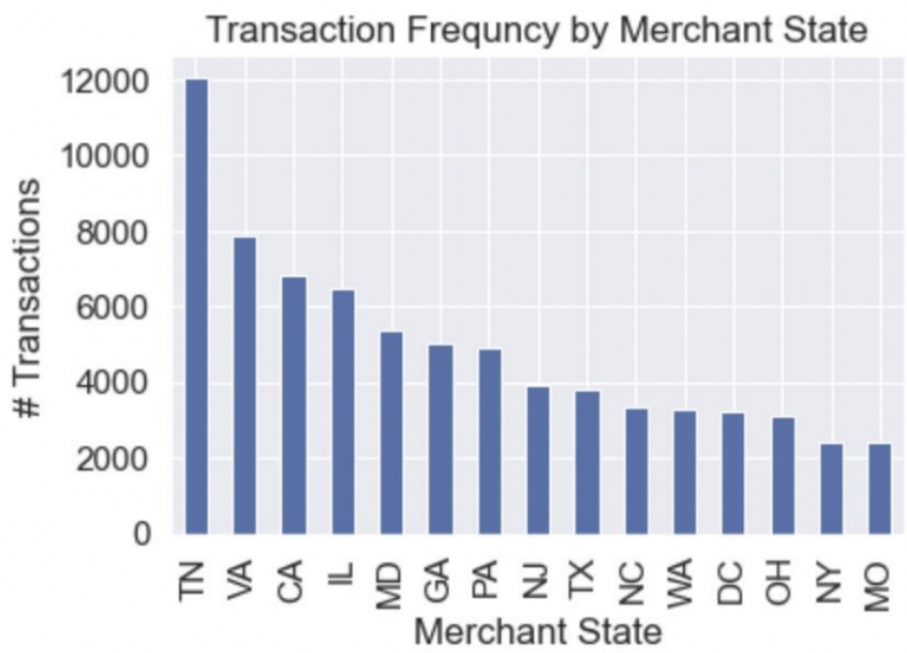
(4) Field Name: Merchnum
Description: a categorical field, the merchant number associated with each card transaction. Categorical field. Combination of positive integers from 1 to 9. The distribution is the top 15 field values of merchant number. The most common value is 930090121224, the count of which is 9,310.



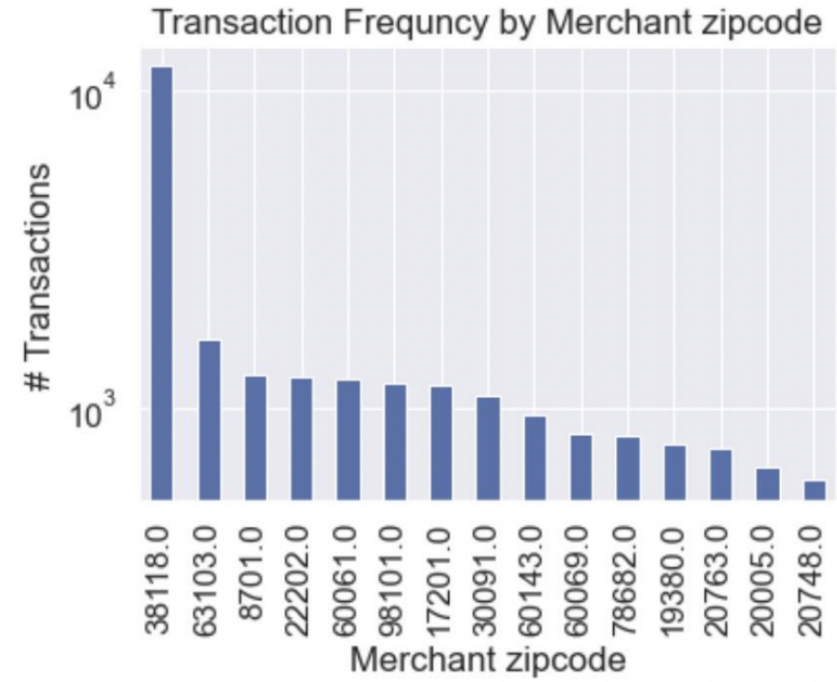
(5) Field Name: Merch description
Description: a categorical field, describes the merchant. The distribution is the top 15 field values of Merch description. The most common value is "GSA-FSS-ADV", count is 1,688. There're 13,126 unique values in this field, no missing value.



(6) Field Name: Merch state
Description: a categorical field, the state merchants from. The distribution is the top 15 field values of Merch state. The most common value is TN (stands for Tennessee),, the count is 12035. There're 228 unique values in this field, there are some missing values.

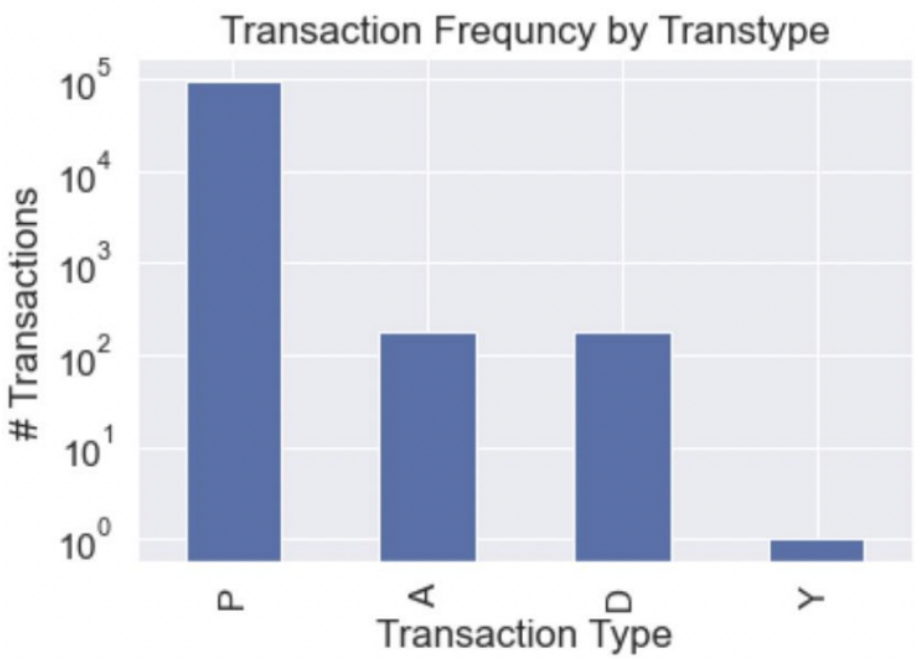


(7) Field Name: Merch zip
Description: a categorical field, the zip code of the merchant. The distribution is the top 15 field values of zip codes. The most common value of the zip code is 38118, the count is 11,868. There're 4,568 unique values in this field, no missing value.

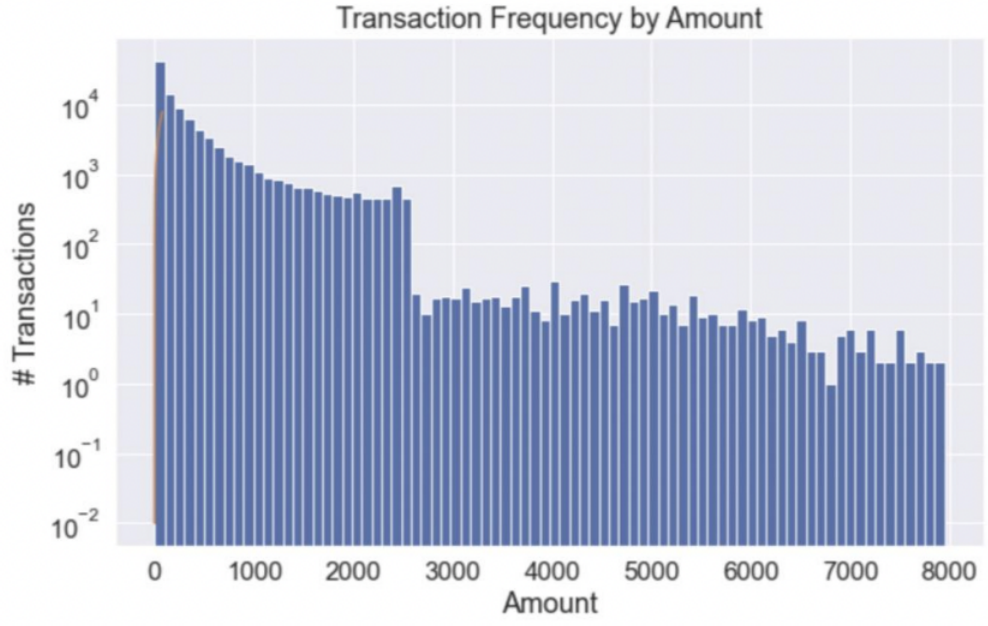


(8) Field Name: Transtype
Description: a categorical field, the type of transaction that took place. The available transaction types comprise of four distinct categories: purchase (P), authorization (A),

debit (D), and year-end (Y). Among these, the purchase type (P) is the most frequently observed with a count of 96398. This prevalence suggests a significant imbalance in transaction types. It is noteworthy that the variable under consideration is complete, with no missing values.



(9) Field Name: Amount
Description: a numeric field, the monetary value of each transaction. The histogram shows the distribution of amount with the range between 0 to 8000. As the amount increases, the number of transactions decreases. The most common value of the amount is 3.62, count 4,283. The mean amount for all the transactions is 427.89. No missing value.



(10) Field Name: Fraud

Description: a dummy variable. Represents the Boolean value of the application, 0 means not labeled as fraud, 1 as labeled as fraud. The non-fraudulent transactions count is 95694, while that of fraudulent transactions is 1059. There're 2 unique values in this field, no missing values.