Fraud Detection for Credit Card Transactions

Project Report

Reuben Chatterjee
PID - A59026323

Table of Contents

1. Executive Summary

In this project, I develop a robust fraud detection model for credit card transactions using advanced machine learning techniques. My primary goal is to enhance the detection of fraudulent activities while minimizing false positives, ultimately protecting the financial interests of the customers and the company.

By leveraging a comprehensive dataset of 97,852 transactions, I meticulously cleaned the data and engineered over 3,200 new features to enrich my analysis. After evaluating various models, I selected the LightGBM (LGBM) model for its superior performance and consistency. The model achieved a good Fraud Detection Rate (FDR) on out-of-time data, ensuring reliable detection of fraudulent transactions beyond the immediate training period.

Financially, the implementation of this model is projected to yield significant savings. By setting the optimal cutoff at 4%, we can maximize fraud detection while minimizing losses. For every correctly identified fraudulent transaction, we gain $400, while each false positive incurs a $20 loss. This strategic approach is anticipated to bolster the overall financial health, safeguarding both customer assets and company revenue.

2. Description of Data

The Dataset is **Credit Card Transaction Data,** comprising transaction details (*amount, card number, merchant number and description, state and zip, and date of transaction*) between **1645 different credit card users** with **13091 different merchants** across **227 states** in the **year 2010**. There are a total of **97,852 recorded transactions** (records) and **10 different fields**.

Broadly, the records can be categorized into **Numeric** and **Categorical** Variables.

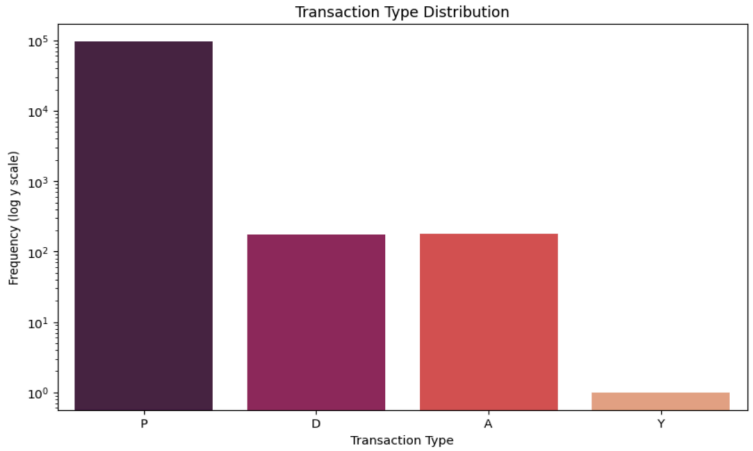
Numeric Fields Table:

	Field Name	Field Type	# Records Have Values	% Populated	# Zeros	Min	Max	Mean	Standard Deviation	Most Common
0	Date	Numeric	97,852.00	100.00	0.00	2010-01-01 00:00:00	2010-12-31 00:00:00	NaN	NaN	2010-02-28 00:00:00
1	Amount	Numeric	97,852.00	100.00	0.00	0.01	3102045.53	425.47	9,949.85	3.62

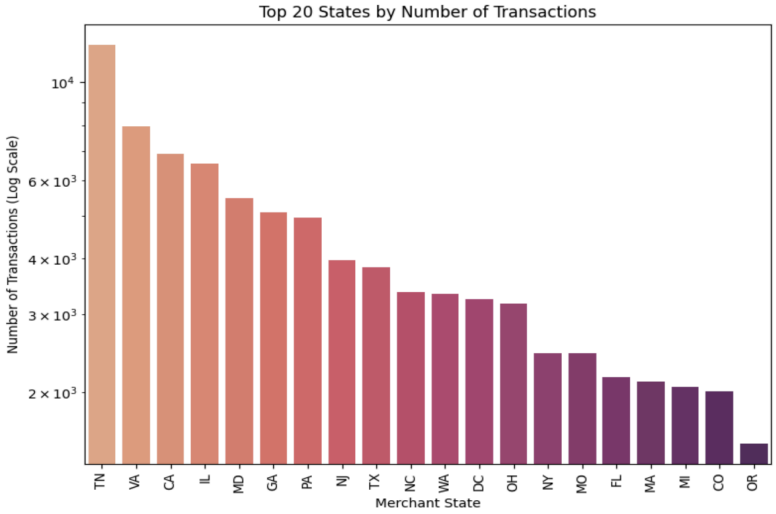
Categorical Fields Table:

	Field Name	Field Type	# Records Have Values	% Populated	# Zeros	# Unique Values	Most Common
0	Recnum	Categorical	97,852.00	100.00	0.00	97,852.00	1
1	Cardnum	Categorical	97,852.00	100.00	0.00	1,645.00	5142148452
2	Merchnum	Categorical	94,455.00	96.53	0.00	13,091.00	930090121224
3	Merch description	Categorical	97,852.00	100.00	0.00	13,126.00	GSA-FSS-ADV
4	Merch state	Categorical	96,649.00	98.77	0.00	227.00	TN
5	Merch zip	Categorical	93,149.00	95.19	0.00	4,567.00	38118.0
6	Transtype	Categorical	97,852.00	100.00	0.00	4.00	P
7	Fraud	Categorical	97,852.00	100.00	95,805.00	2.00	0

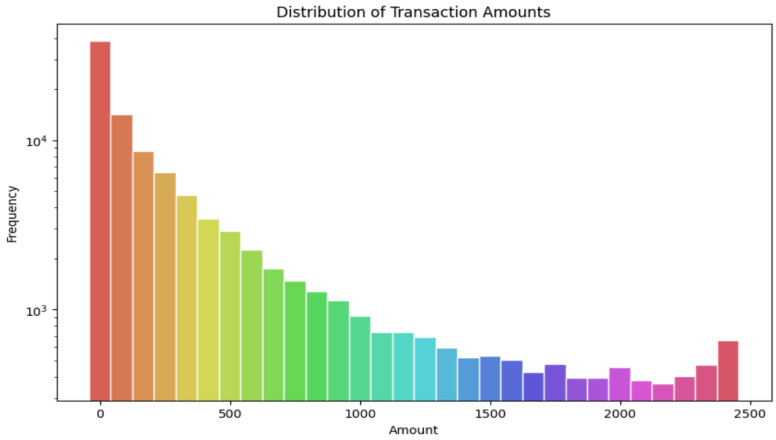
Distribution of the field “Transtype” -



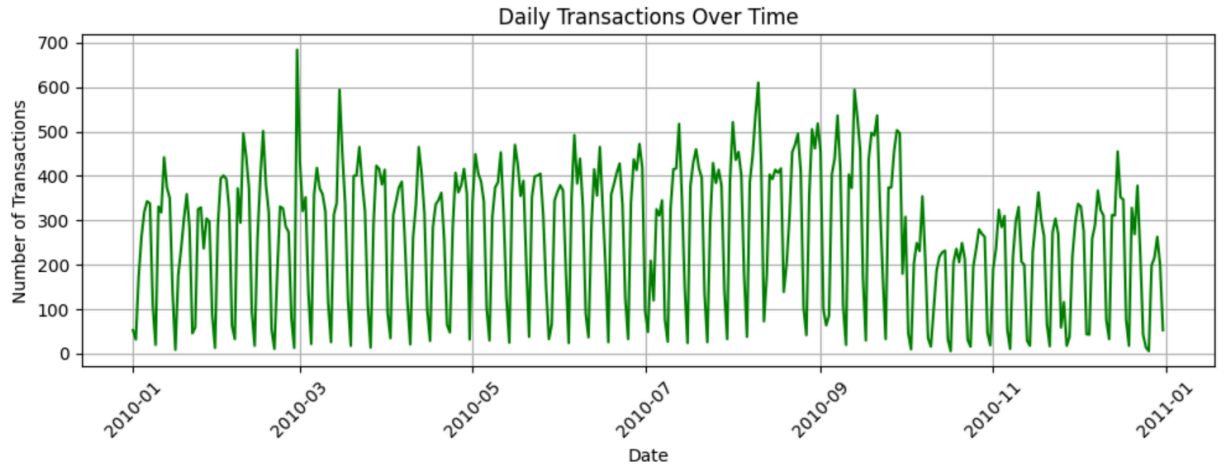
Distribution of the field “Merch state” -

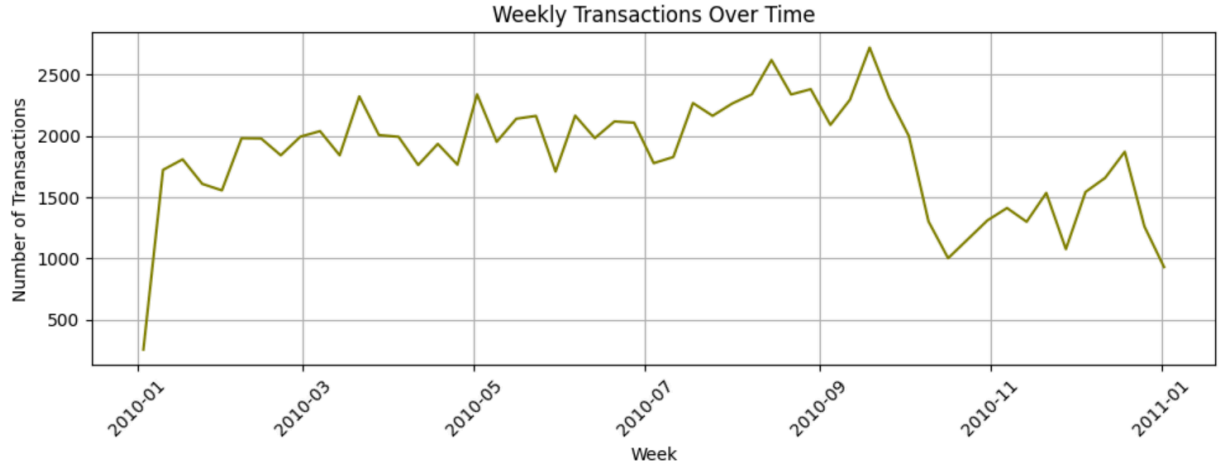


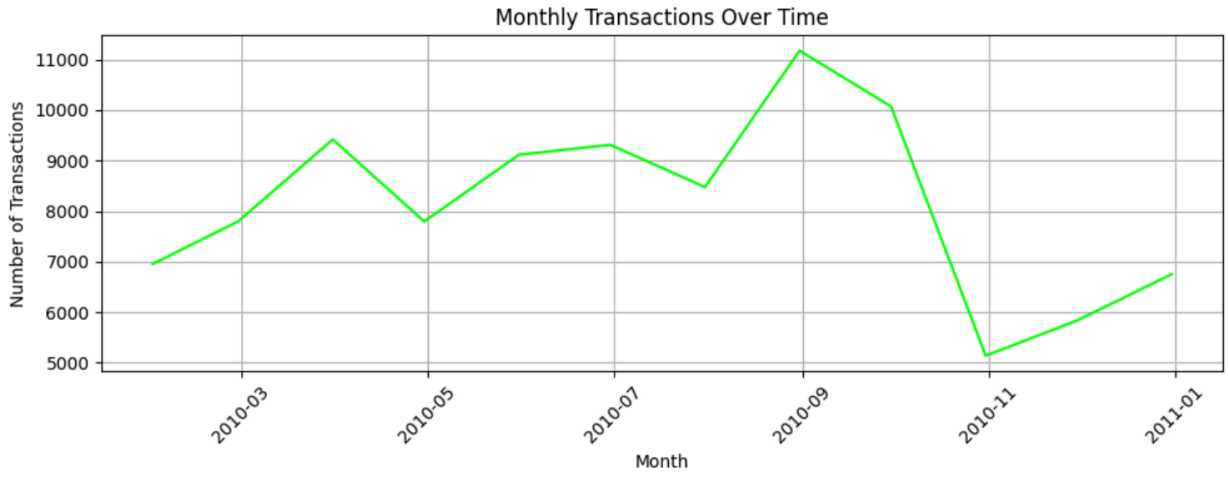
Distribution of the field “Amount” -



Distribution of the field “Date” (varying from Daily, Weekly and Monthly transactions) -







3. Data Cleaning

1. Merch Number Field:

a) Description of Exclusions:
No specific exclusions were applied directly to the Merch Number field outside of initial data cleaning steps, which filtered transactions by type and amount.

b) Imputation Logic:
The imputation process for the Merch Number field involved several steps:

- Placeholder entries ('0') were replaced with NaN to signify missing data, resulting in 3,279 missing values initially.
- A dictionary was created to map existing Merch Descriptions to their corresponding Merch Numbers.
- NaN values were filled by mapping Merch Descriptions to their corresponding Merch Numbers using the dictionary, reducing missing values to 2,115.
- Entries with descriptions indicating adjustments were set to 'unknown', further reducing missing values.
- For remaining missing entries, unique Merch Numbers were generated for each unique Merch Description, ensuring all entries had non-blank values.
- At the end of the process, there were no remaining missing values in the Merch Number field.

2. Merch State Field:

a) Description of Exclusions:
No specific exclusions were made for the Merch State field during the data cleaning process.

b) Imputation Logic:
The imputation process for the Merch State field involved several steps:

- Initial identification of missing entries.
- Mapping attempts using non-null Merch Zip codes to fill missing Merch State values.
- Further mappings using Merch Number and Merch Description as references.
- Entries indicating adjustments were labeled as 'unknown'.
- Non-U.S. state values were manually reviewed and labeled as 'foreign'.
- Remaining missing values were labeled as 'unknown'.
- By the conclusion of the process, all Merch State entries were accounted for.

3. Merch Zip Field:

a) Description of Exclusions:
No specific exclusions were made for the Merch Zip field during the data cleaning process.

b) Imputation Logic:
The imputation process for the Merch Zip field involved several steps:

- Construction of dictionaries to map Merch Numbers and Merch Descriptions to corresponding Merch Zip codes.
- Filling missing Merch Zip codes using these mappings.
- Entries related to adjustments were labeled as 'unknown'.
- The most populous zip code for each state was used as an estimate for missing zip codes.
- Remaining missing values were labeled as 'unknown'.
- Ultimately, no missing entries remained in the Merch Zip field.

4. Variable Creation

How Fraud Occurs -

Fraud in credit card transactions can occur through various methods, often exploiting vulnerabilities in the payment system or taking advantage of stolen or compromised card information.
Fraudulent activities in credit card transactions typically stem from stolen card information or compromised account credentials. Methods include:

1. *Stolen Card Information:* Obtained through data breaches, skimming devices, phishing scams, or physical theft, fraudsters use this data for unauthorized purchases or transactions, both online and in-person.
2. *Account Takeover:* Through phishing or hacking, fraudsters gain control of a victim's account, enabling them to make purchases, transfer funds, or engage in other fraudulent activities using the victim's credentials.
3. *Card Not Present (CNP) Transactions*: Fraudsters utilize stolen card details for online purchases where the physical card isn't required. This method is prevalent in online shopping fraud.
4. *Identity Theft:* Stolen personal information is used to open new credit card accounts or take over existing ones, allowing fraudsters to make purchases or obtain cash advances, leaving victims responsible for the charges.

Identifying Fraudulent Transactions:

1. *Anomaly Detection:* Automated systems can flag transactions that deviate from the cardholder's typical spending behavior, such as unusually large purchases, transactions in unfamiliar locations, or purchases from high-risk merchants.
2. *Geolocation and Device Verification:* Analyzing the geographic location of the transaction and cross-referencing it with the cardholder's usual location can help detect suspicious activity. Similarly, verifying the device used for the transaction against known devices associated with the cardholder can identify potential fraud.

3. *Behavioral Analysis:* Monitoring for patterns of behavior that are indicative of fraud, such as multiple failed authentication attempts, frequent changes to account details, or sudden spikes in transaction volume, can help detect fraudulent activity.

Variable Creation:

Description	No. of Variables created
Day of Week: Categorizes the transaction date into weekdays.	**1**
Risk for Day of Week: Assigns a risk score to transactions based on the weekday.	**1**
Target Encoded Features: Numeric representation of categorical features using mean fraud values.	**3**
Days Since Last Transaction: Tracks days since the last transaction for each entity.	**23**
Transaction Count: Number of transactions per entity within specified time frames. Most entities follow the time frames - 0, 1, 3, 7, 14, 30, 60 days	**161**
Average Transaction Amount: Average amount per transaction across different time windows.	**161**
Maximum Transaction Amount: Highest transaction amount within specified time windows.	**161**
Median Transaction Amount: Median transaction amount across various time windows.	**161**
Total Transaction Amount: Total transactions amount over specified time windows.	**161**
Transaction Amount Ratios: Normalized transaction amounts compared to average, maximum, median, and total.	**644**
Transaction Count Ratios: Compares transaction counts in shorter versus longer time windows.	**184**

Total Transaction Amount Ratios: Normalized total transaction amounts in short versus long windows.	**184**
Transaction Velocity Ratios: Frequency comparison of transactions in shorter versus longer windows.	**184**
Average Transaction Variability: Variability in transaction amounts over selected time windows.	**138**
Maximum Transaction Variability: Largest transaction amount change within selected windows.	**138**
Median Transaction Variability: Median transaction amount difference across time windows.	**138**
Unique Interaction Counts: Number of unique interactions between entity pairs within selected windows.	**696**

Additional Variables Created:

Time Weighted Transaction Frequency: Calculates the frequency of transactions for each entity, adjusted by a decay factor that weighs more recent transactions higher	**23**
High-Value Transaction Rate: Identifies the proportion of transactions that are in the top 90th percentile of amounts for each entity	**23**
Most Common Transaction Hour: Determines the hour of the day when most transactions occur for each entity	**23**
New vs. Returning Customer Analysis: Flags transactions as either from new or returning customers	**46**
High-Value Transaction Rate: Identifies the proportion of transactions that are in the top 90th percentile of amounts for each entity	**1**

5. Feature Selection

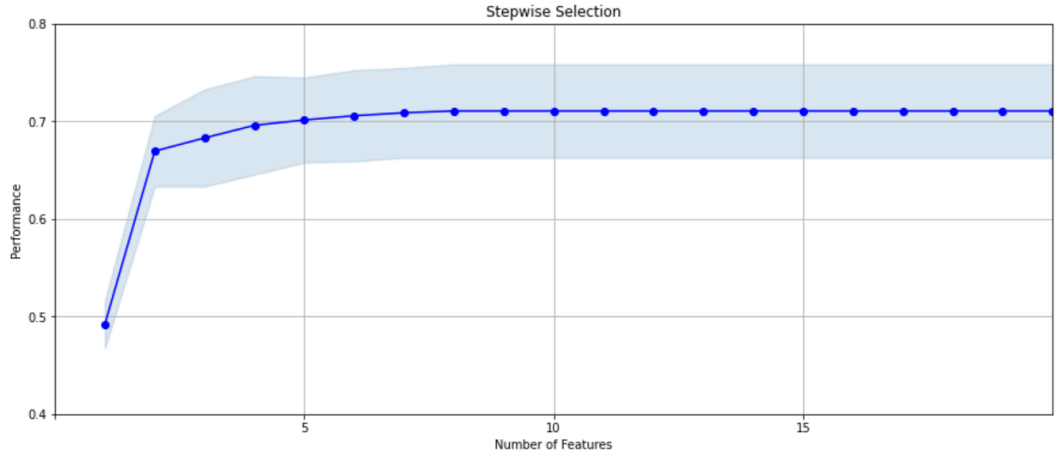
For Feature Selection, I select the best or top 20 variables from the ~3200 that were created during the variable creation task. Here, I run 4 instances of the wrapper, 2 with forward selection of variables and 2 with backward selection; each being run on the LGBM and Random Forest Models.

Ideally, we keep the num_filter value to be 10-20% of the total variables created, but after running into extremely long computation times, I decided to keep the values to 200 for Forward selection runs, and 80 for Backward selection runs.

Plots of Performance vs No. of Variables -

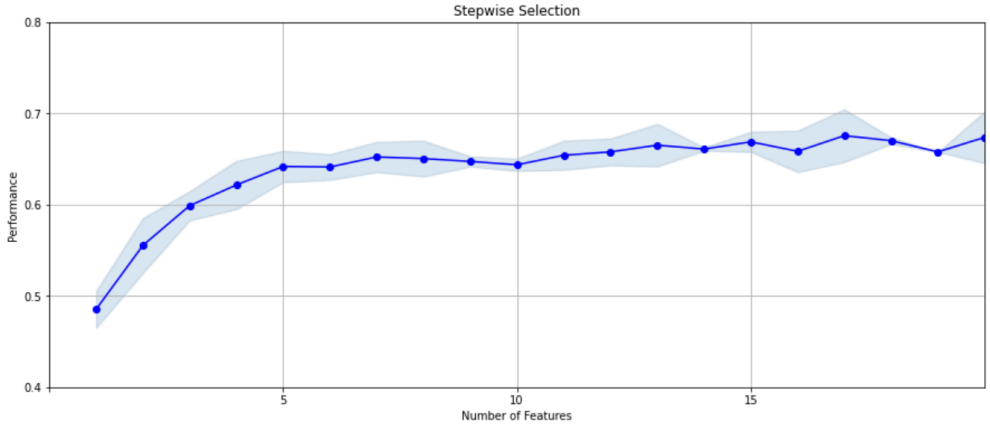
Forward Selection: Running with num_filter = 200

1. LGBM (time taken - 19 minutes 09 seconds)



I initially ran the model with num_filter = 80, which took 1 minute and 39 seconds, but the graph only barely crossed the 0.7 mark. Hence I ran it with 200.

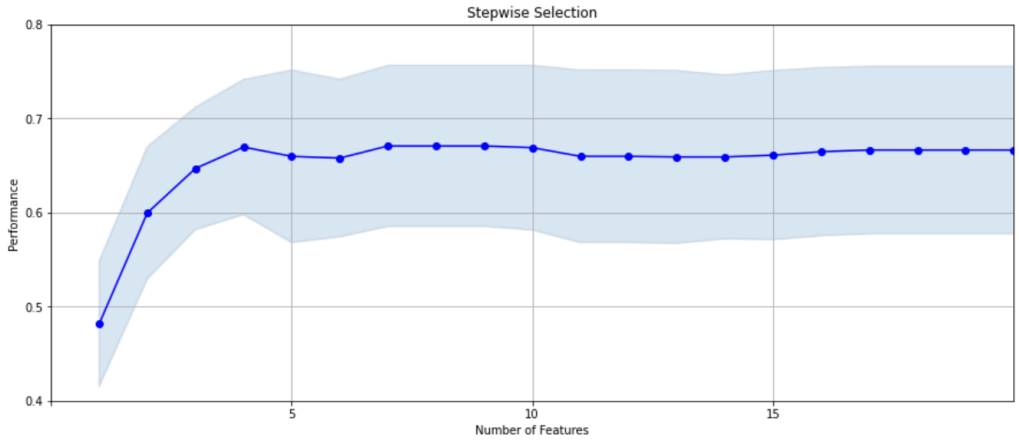
2. Random Forest (time taken - 9 minutes 14 seconds)



I initially ran the model with num_filter = 80, which took 3 minutes and 46 seconds, but the graph only barely crossed the 0.6 mark. Hence I ran it with 200, getting a better plot that is closer to 0.7.

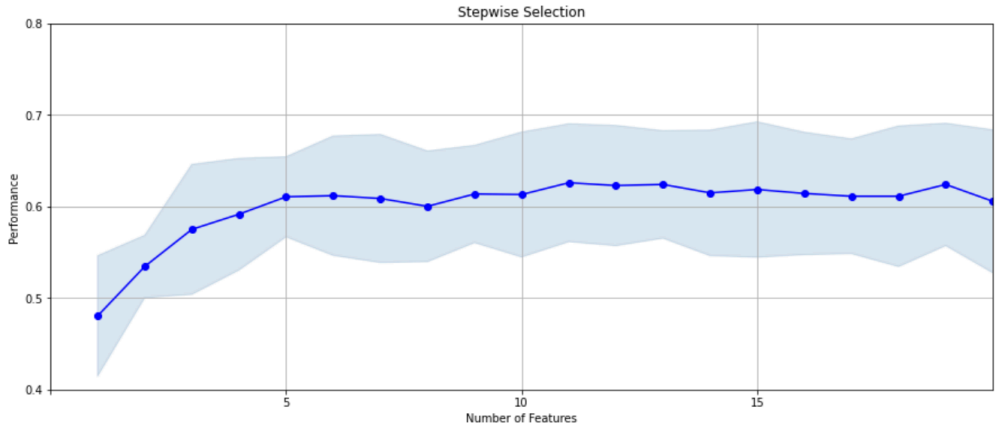
Backward Selection: Running with num_filter 120

1. LGBM (time taken - 2 hours 9 minutes 40 seconds)



I initially ran with num_filter = 200, but each instance of backward selection took over 5 hours, and the plot did not manage to cross 0.7. Hence I decided to run both instances with num_filter as 120, to reduce the runtimes.

2. Random Forest (time taken - 1 hour 23 minutes 16 seconds)



From my plots, I see that the forward selection with LGBM gives the best result (i.e > 0.7), Hence I chose to go ahead with the top 20 variables from that run.

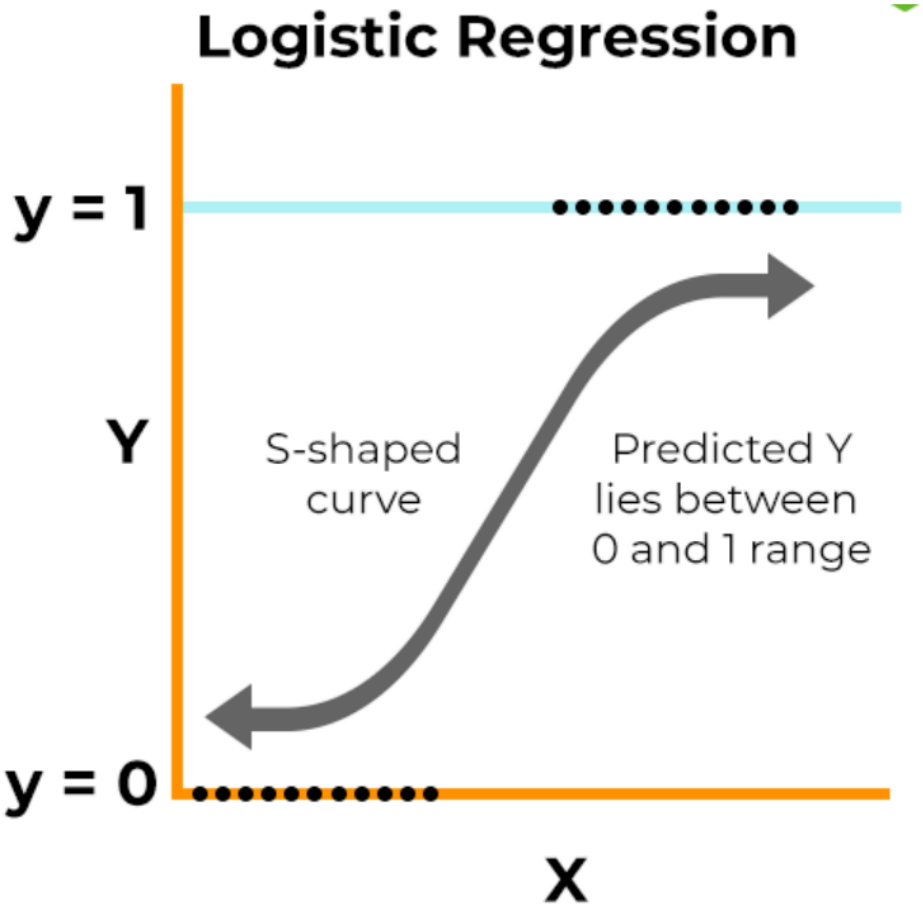
The Final list of Variables I decide to keep is:

	wrapper order	variable	filter score
0	1	Cardnum_unique_count_for_card_state_1	0.476067
1	2	card_state_total_3	0.368079
2	3	Cardnum_max_7	0.410589
3	4	Cardnum_vdratio_1by14	0.485431
4	5	Card_dow_unique_count_for_merch_state_1	0.447357
5	6	Card_dow_vdratio_0by14	0.479086
6	7	Cardnum_count_7	0.526897
7	8	Card_dow_total_30	0.474759
8	9	Cardnum_total_14	0.494375
9	10	Card_dow_count_7	0.482384
10	11	Cardnum_actual/toal_0	0.479550
11	12	Cardnum_variability_max_1	0.477836
12	13	Card_dow_vdratio_0by7	0.467961
13	14	Cardnum_vdratio_1by7	0.466766
14	15	Cardnum_unique_count_for_card_state_3	0.466410
15	16	Cardnum_unique_count_for_card_zip_3	0.464311
16	17	Cardnum_unique_count_for_Merchnum_3	0.460748
17	18	Cardnum_actual/toal_1	0.459715
18	19	Card_dow_unique_count_for_Card_Merchdesc_1	0.447250
19	20	Card_dow_unique_count_for_state_des_1	0.447238

6. Preliminary Model Explores

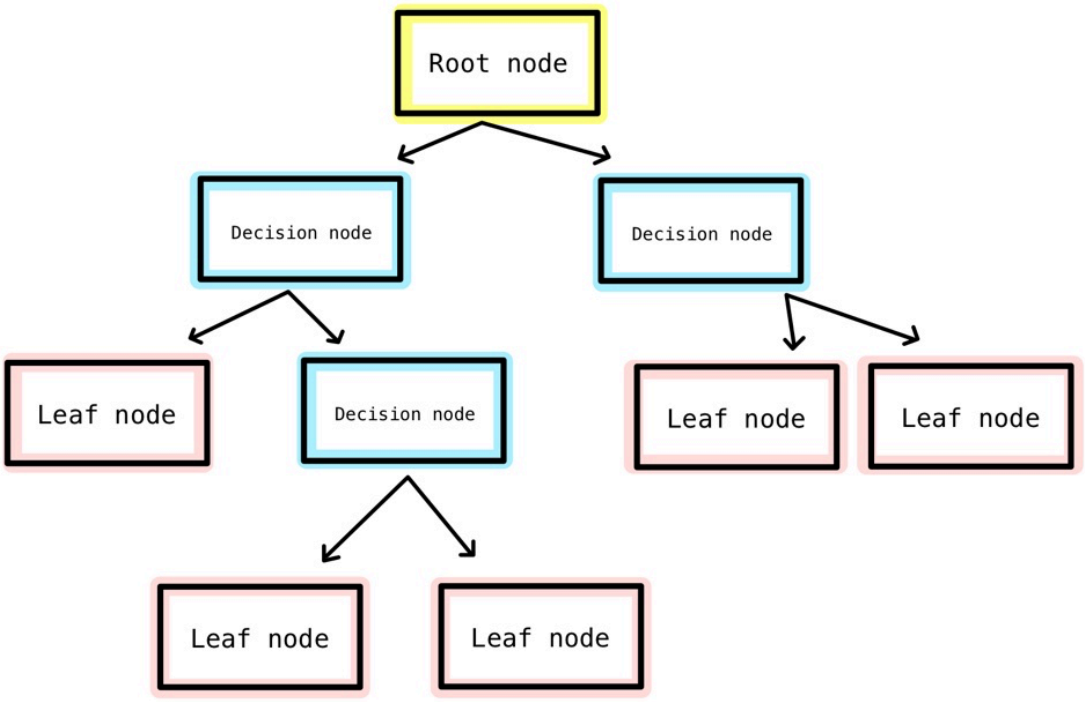
I explored a couple of different Models for this project. Initially, I tested out trial runs using a simple linear model, i.e. Logistic Regression.

Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on.



Next, I moved on to running the trials with Non-Linear Machine Learning Models. These were -

1. Decision Tree - Decision tree classifiers provide a readable classification model that is potentially accurate in many different application contexts, including energy-based applications. The decision tree classifier (Pang-Ning et al., 2006) creates the classification model by building a decision tree. Each node in the tree specifies a test on an attribute, each branch descending from that node corresponds to one of the possible values for that attribute. Each leaf represents class labels associated with the instance. Instances in the training set are classified by navigating them from the root of the tree down to a leaf, according to the outcome of the tests along the path. Starting from the root node of the tree, each node splits the instance space into two or more sub-spaces according to an attribute test condition. Then moving down the tree branch corresponding to the value of the attribute, a new node is created. This process is then repeated for the subtree rooted at the new node, until all records in the training set have been classified.

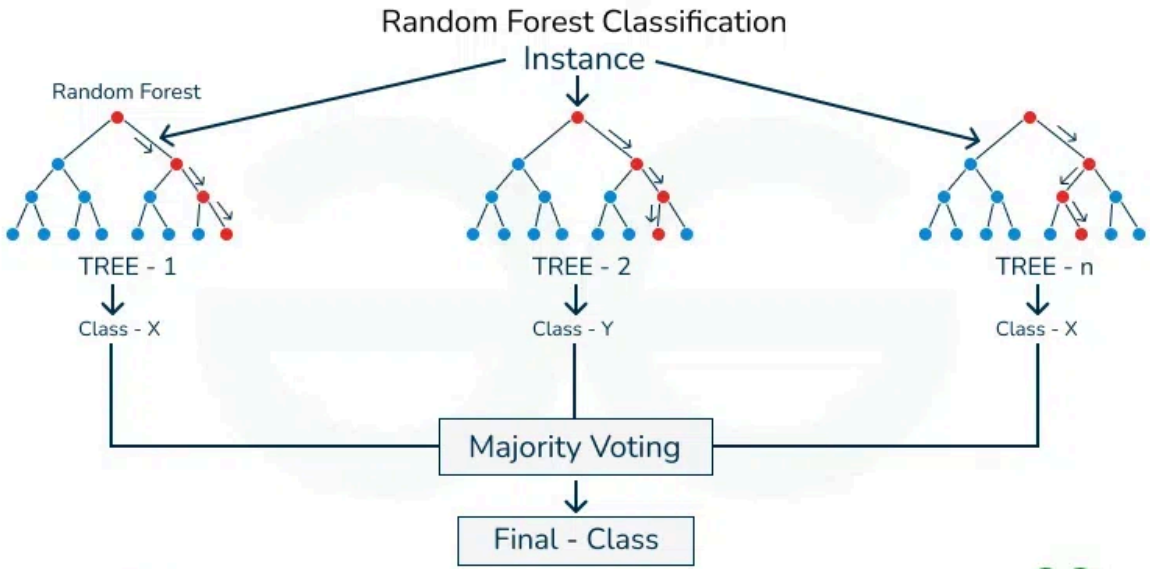


2. Random Forest - The Random forest or Random Decision Forest is a supervised Machine learning algorithm used for classification, regression, and other tasks using decision trees.

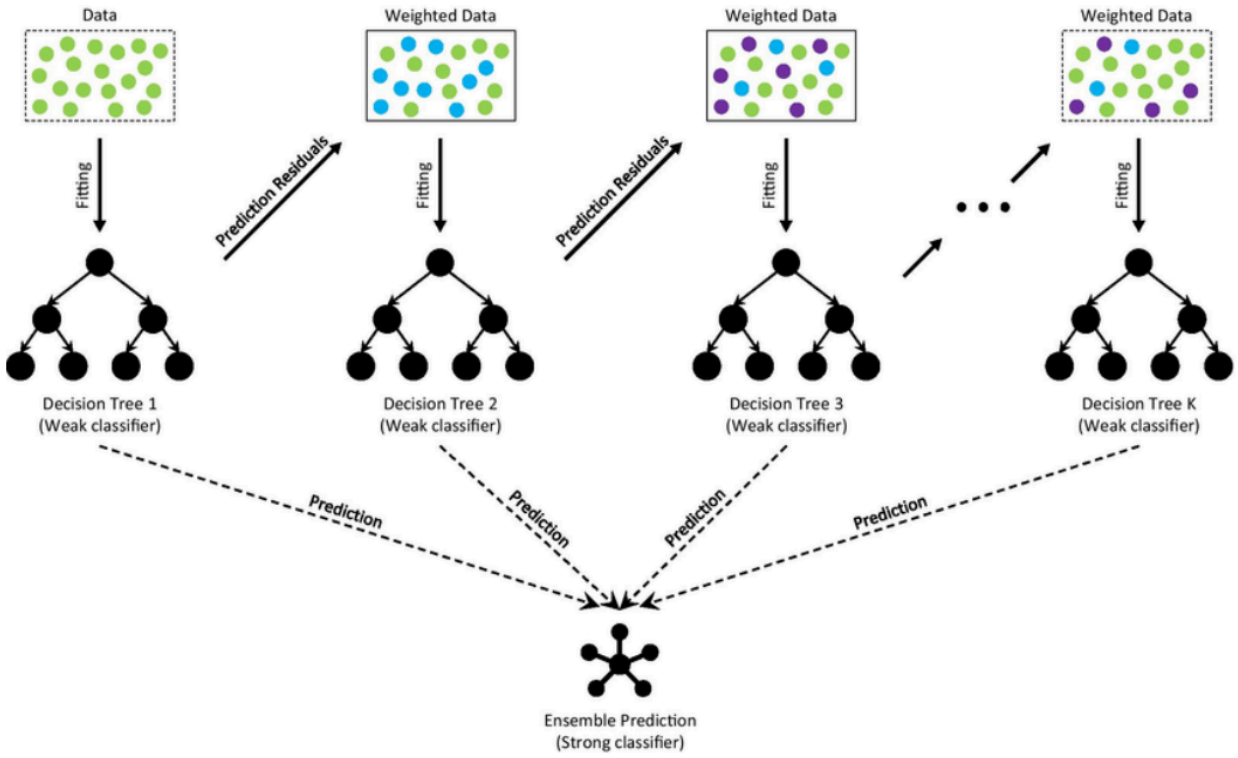
Random Forest Classification is an ensemble learning technique designed to enhance the accuracy and robustness of classification tasks. The algorithm builds a multitude of decision trees during training and outputs the class that is the mode of the classification classes. Each decision tree in the random forest is constructed using a subset of the training data and a random subset of features introducing diversity among the trees, making the model more robust and less prone to overfitting.

The random forest algorithm employs a technique called bagging (Bootstrap Aggregating) to create these diverse subsets. During the training phase, each tree is built by recursively partitioning the data based on the features. At each split, the algorithm selects the best feature from the random subset, optimizing for information gain or Gini impurity. The process continues until a predefined stopping criterion is met, such as reaching a maximum depth or having a minimum number of samples in each leaf node.

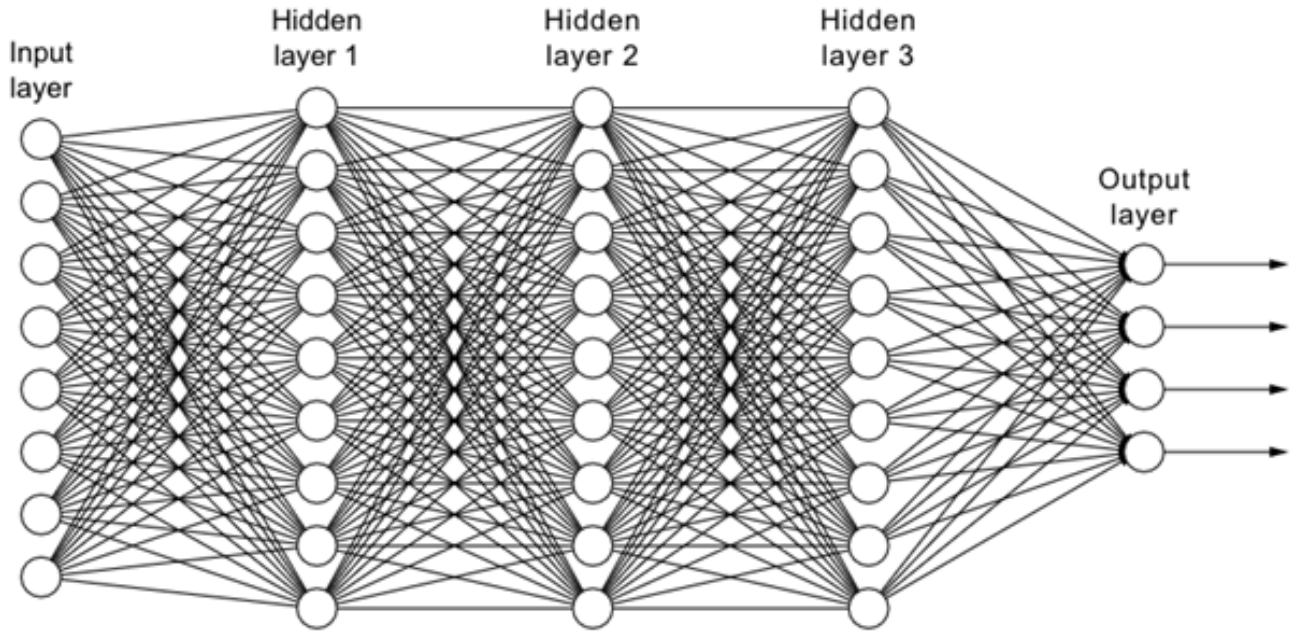
Once the random forest is trained, it can make predictions, using each tree “votes” for a class, and the class with the most votes becomes the predicted class for the input data.



3. Boosted Tree - Gradient boosting works by building simpler (weak) prediction models sequentially where each model tries to predict the error left over by the previous model. Because of this, the algorithm tends to overfit rather quickly. Boosting transforms weak decision trees (called weak learners) into strong learners. Each new tree is built considering the errors of previous trees. In both bagging and boosting, the algorithms use a group (ensemble) of decision trees.



4. Neural Nets - Neural Networks are computational models that mimic the complex functions of the human brain. The neural networks consist of interconnected nodes or neurons that process and learn from data, enabling tasks such as pattern recognition and decision making in machine learning.
Neural networks extract identifying features from data, lacking pre-programmed understanding. Network components include neurons, connections, weights, biases, propagation functions, and a learning rule. Neurons receive inputs, governed by thresholds and activation functions. Connections involve weights and biases regulating information transfer. Learning, adjusting weights and biases, occurs in three stages: input computation, output generation, and iterative refinement enhancing the network's proficiency in diverse tasks.



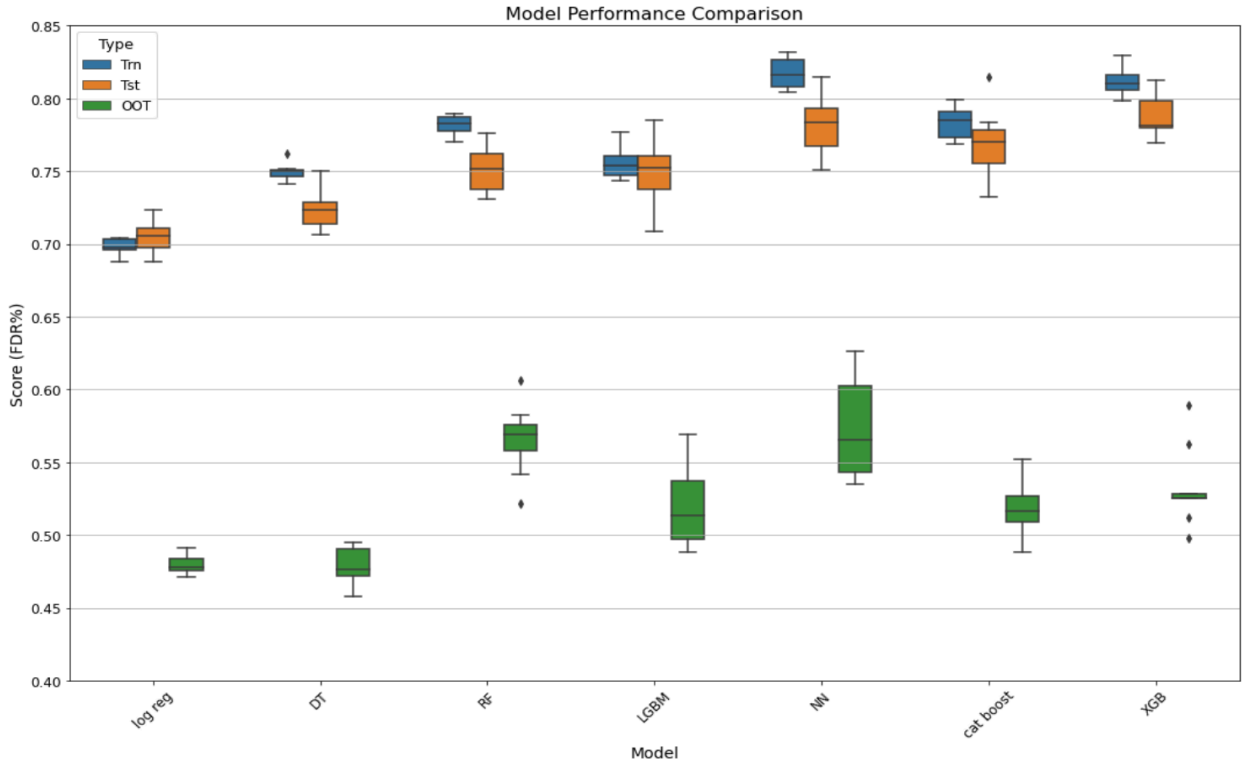
I started out by running the logistic regression model with 5 variables, gradually increasing the number of variables and tuning the hyperparameters. Since this model is not very useful for the sake of this project, I did not wait to get OOT > 0.05, and moved on to the Non-linear models.

For the Non-linear models, I tweaked one parameter at a time and on noticing slight improvements in OOT, kept the particular parameter constant and moved on to tweak other parameters, eventually choosing the iteration with the best OOT value. I followed this approach for all the models I trained.

The trials I conducted gave the following results -

Model	Parameter					Average FDR at 3%				
Logistic Regression	No. of Variables	penalty	C	solver	l1_ratio	train	test	oot		
1	5	l2	1	lbfgs	1	0.674	0.658	0.473		
2	10	l2	1	lbfgs	1	0.6817	0.6816	0.464		
3	5	elasticnet	1	saga	1	0.665	0.674	0.473		
4	10	elasticnet	1	saga	1	0.68	0.683	0.465		
5	10	elasticnet	0.5	saga	0.8	0.681	0.678	0.464		
Decision Tree	No. of Variables	criterion	splitter	max_depth	min_samples_split	min_samples_leaf	train	test	oot	
1	20	gini	random	20	2	1	0.93	0.643	0.415	
2	20	gini	best	None	2	1	0.946	0.678	0.401	
3	20	gini	best	20	2	20	0.75	0.72	0.49	
4	20	gini	random	20	2	40	0.701	0.693	0.481	
5 (chosen)	20	gini	best	20	60	200	0.724	0.713	0.507	
Random Forest	No. of Variables	n_estimators	criterion	max_depth	min_samples_split	min_samples_leaf	train	test	oot	
1	20	50	entropy	None	2	1	0.989	0.816	0.502	
2	20	50	gini	None	2	1	0.998	0.794	0.512	
3	20	100	gini	20	300	50	0.974	0.817	0.542	
4	20	100	gini	None	1000	100	0.961	0.829	0.554	
5 (chosen)	20	30	-	20	120	60	0.789	0.785	0.551	
Boosted Tree	No. of Variables	n_estimators	learning_rate	max_depth	num_leaves	train	test	oot		
1	20	50	0.01	1	None	0.661	0.666	0.464		
2	20	50	0.1	1	None	0.705	0.693	0.464		
3	20	100	0.01	5	None	0.724	0.716	0.471		
4	20	200	0.01	5	None	0.735	0.73	0.477		
5 (chosen)	20	500	0.01	5	None	0.768	750	0.485		
NN (MLP Classifier)	No. of Variables	hidden_layer_sizes	activation	alpha	learning_rate	learning_rate_init	max_iter	train	test	oot
1	20	(10,)	relu	0.0001	constant	0.001	50	0.739	0.729	0.544
2	20	(10,)	relu	0.0001	adaptive	0.001	50	0.73	0.729	0.531
3	20	(10,)	relu	0.001	constant	0.001	50	0.741	0.716	0.547
4	20	(10,)	relu	0.001	constant	0.01	50	0.766	0.736	0.551
5 (chosen)	20	(20,20)	-	0.005	-	0.01	-	0.815	0.785	0.578
NN (Catboost)	No. of Variables	Verbose	depth	l2_leaf_reg	learning_rate	min_data_in_leaf	train	test	oot	
1	20	0	3	6	0.02	60	0.786	0.761	0.509	
2	20	0	3	6	0.02	100	0.781	0.763	0.513	
NN (XGBoost)	No. of Variables	booster	max_depth	n_estimators	learning_rate	train	test	oot		
1	20	gbtree	3	70	0.1	0.793	0.765	0.519		
2	20	gbtree	3	100	0.1	0.809	0.784	0.546		

Boxplots to compare how each model fared -



The boxplot comparison provides a clear picture of each model's performance, highlighting their strengths and weaknesses in terms of training, testing, and generalization over time.

Observations -

Logistic Regression (log reg):

- Training (Tm) and Testing (Tst) scores are consistent and close, with scores around 0.70.
- Out-of-time (OOT) score is slightly lower, indicating a slight decrease in performance over time.

Decision Tree (DT):

- Shows a moderate performance with scores around 0.55 for Tm and Tst.
- OOT performance drops significantly, indicating poor generalization over time.

Random Forest (RF):

- High variability in OOT scores, with the median around 0.55, suggesting inconsistent performance over time.
- Training and testing scores are higher and closer to each other.

LightGBM (LGBM):

- Consistently high performance in Tm and Tst, with scores around 0.80.
- OOT performance is slightly lower but still relatively high.

Neural Network (NN):

- Shows high variability in OOT scores, similar to RF.
- Training and testing performances are high and consistent.

CatBoost:

- High performance across Tm, Tst, and OOT, with median scores around 0.75 to 0.80.
- Slight decrease in OOT but still maintains high performance.

XGBoost (XGB):

- High and consistent performance across Tm, Tst, and OOT, with median scores around 0.80.
- Very few outliers and a tight interquartile range, indicating stable performance.

Based on the Trials, I choose the ***LGBM model*** as my ***Final Model***.

7. Final Model Performance

I chose to go ahead with the **LGBM model**, as it gives a decent result, not too overfit, and with a good OOT value.

LightGBM is a gradient boosting ensemble method that is used by the Train Using AutoML tool and is based on decision trees. As with other decision tree-based methods, LightGBM can be used for both classification and regression. LightGBM is optimized for high performance with distributed systems.
LightGBM creates decision trees that grow leaf wise, which means that given a condition, only a single leaf is split, depending on the gain. Leaf-wise trees can sometimes overfit especially with smaller datasets. Limiting the tree depth can help to avoid overfitting.

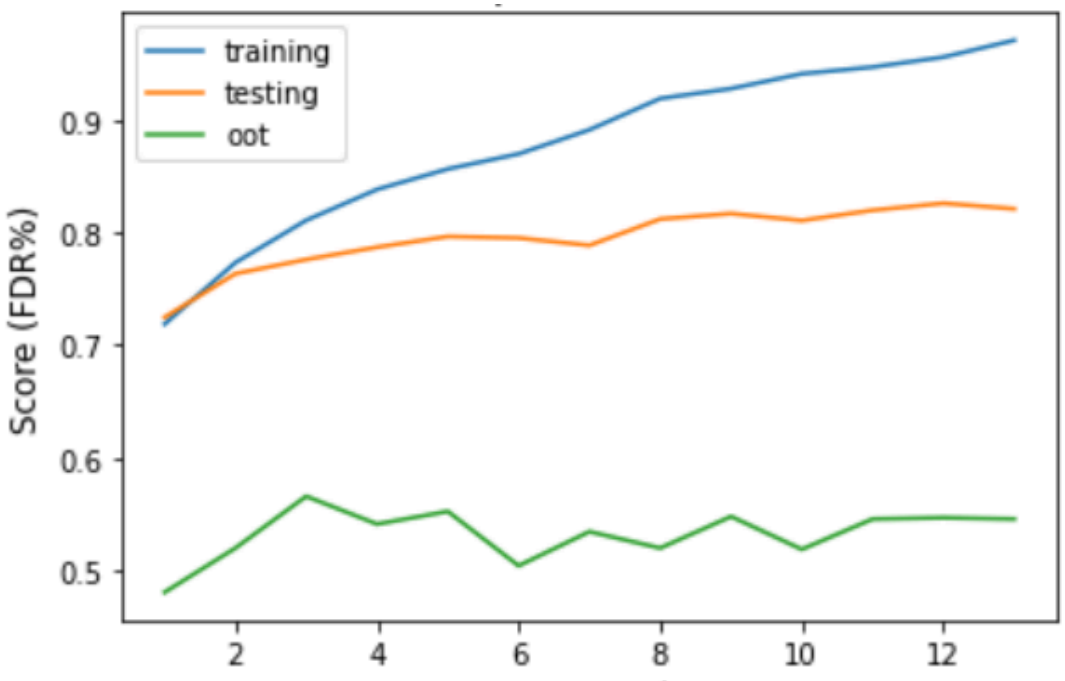
Some *parameters* of the lgb.LGBMClassifier are:

- **boosting_type** (str, optional (default='gbdt')) – 'gbdt', traditional Gradient Boosting Decision Tree. 'dart', Dropouts meet Multiple Additive Regression Trees. 'rf', Random Forest.
- **num_leaves** (int, optional (default=31)) – Maximum tree leaves for base learners.
- **max_depth** (int, optional (default=-1)) – Maximum tree depth for base learners, <=0 means no limit.
- **learning_rate** (float, optional (default=0.1)) – Boosting learning rate. You can use callbacks parameter of fit method to shrink/adapt learning rate in training using reset_parameter callback. Note, that this will ignore the learning_rate argument in training.
- **n_estimators** (int, optional (default=100)) – Number of boosted trees to fit.

I used the following parameters for my final model:

```
model = lgb.LGBMClassifier(
        n_estimators=200,
        num_leaves = 4 )
```

Additionally, I checked to see the effect of increasing the “num_leaves” parameter on the model, and it turns out that the model shows significant overfitting as seen in the plot below.



Results:

Best Model TRN: 0.793
Best Model TST: 0.785
Best Model OOT: 0.593

Training Dataset-

Total Records: 59684
Number of Goods: 58463
Number of Bads: 1221
Fraud Detection Rate: 0.0204

Training	Records		Goods		Bads		Fraud Rate								
	59684		58463		1221		0.0204								
	Bin Stats						Cumulative Stats								
bin	#recs	#g	#b	%g	%b	tot	cg	cb	%cg	FDR	KS	FPR	Fraud Savings	FP Loss	Overall Savings
1	597	36	561	6.030150754	93.96984925	597	36	561	0.06157740793	45.94594595	45.88436854	0.06417112299	561000	1080	559920
2	597	271	326	45.39363484	54.60636516	1194	307	887	0.525118451	72.64537265	72.12025419	0.3461104848	887000	9210	877790
3	597	493	104	82.57956449	17.42043551	1791	800	991	1.368386843	81.16298116	79.79459432	0.8072653885	991000	24000	967000
4	596	551	45	92.44966443	7.55033557	2387	1351	1036	2.310863281	84.84848485	82.53762157	1.304054054	1036000	40530	995470
5	597	562	35	94.13735343	5.862646566	2984	1913	1071	3.272155038	87.71498771	84.44283268	1.786181139	1071000	57390	1013610
6	597	575	22	96.31490787	3.685092127	3581	2488	1093	4.255683082	89.51678952	85.26110644	2.276303751	1093000	74640	1018360
7	597	582	15	97.48743719	2.512562814	4178	3070	1108	5.25118451	90.74529075	85.49410624	2.770758123	1108000	92100	1015900
8	597	585	12	97.98994975	2.010050251	4775	3655	1120	6.251817389	91.72809173	85.47627434	3.263392857	1120000	109650	1010350
9	597	587	10	98.32495812	1.675041876	5372	4242	1130	7.255871235	92.54709255	85.29122131	3.753982301	1130000	127260	1002740
10	596	589	7	98.82550336	1.174496644	5968	4831	1137	8.263346048	93.12039312	84.85704707	4.248900616	1137000	144930	992070
11	597	586	11	98.15745394	1.842546064	6565	5417	1148	9.26568941	94.02129402	84.75560461	4.718641115	1148000	162510	985490
12	597	591	6	98.99497487	1.005025126	7162	6008	1154	10.27658519	94.51269451	84.23610932	5.206239168	1154000	180240	973760
13	597	592	5	99.16247906	0.837520938	7759	6600	1159	11.28919145	94.92219492	83.63300347	5.69456428	1159000	198000	961000
14	597	594	3	99.49748744	0.5025125628	8356	7194	1162	12.30521869	95.16789517	82.86267648	6.191049914	1162000	215820	946180
15	597	593	4	99.32998325	0.6700167504	8953	7787	1166	13.31953543	95.4954955	82.17596006	6.67838765	1166000	233610	932390
16	596	593	3	99.4966443	0.5033557047	9549	8380	1169	14.33385218	95.74119574	81.40734356	7.168520103	1169000	251400	917600
17	597	594	3	99.49748744	0.5025125628	10146	8974	1172	15.34987941	95.98689599	80.63701658	7.656996587	1172000	269220	902780
18	597	594	3	99.49748744	0.5025125628	10743	9568	1175	16.36590664	96.23259623	79.86668959	8.142978723	1175000	287040	887960
19	597	596	1	99.83249581	0.1675041876	11340	10164	1176	17.38535484	96.31449631	78.92914147	8.642857143	1176000	304920	871080
20	597	596	1	99.83249581	0.1675041876	11937	10760	1177	18.40480304	96.3963964	77.99159336	9.141886151	1177000	322800	854200

Testing-

Total Records: 25580
Number of Goods: 25051
Number of Bads: 529
Fraud Detection Rate: 0.0206

Test	Records		Goods		Bads		Fraud Rate								
	25580		25051		529		0.0206								
	Bin Stats						Cumulative Stats								
bin	#recs	#g	#b	%g	%b	tot	cg	cb	%cg	FDR	KS	FPR	Fraud Savings	FP Loss	Overall Savings
1	256	34	222	13.28125	86.71875	256	34	222	0.1357231248	41.96597353	41.83025041	0.1531531532	222000	1020	220980
2	256	121	135	47.265625	52.734375	512	155	357	0.6187377749	67.48582231	66.86708453	0.4341736695	357000	4650	352350
3	255	194	61	76.07843137	23.92156863	767	349	418	1.393157958	79.01701323	77.62385527	0.8349282297	418000	10470	407530
4	256	234	22	91.40625	8.59375	1023	583	440	2.327252405	83.1758034	80.848551	1.325	440000	17490	422510
5	256	234	22	91.40625	8.59375	1279	817	462	3.261346852	87.33459357	84.07324672	1.768398268	462000	24510	437490
6	256	250	6	97.65625	2.34375	1535	1067	468	4.259311006	88.46880907	84.20949807	2.27991453	468000	32010	435990
7	256	250	6	97.65625	2.34375	1791	1317	474	5.257275159	89.60302457	84.34574942	2.778481013	474000	39510	434490
8	255	248	7	97.25490196	2.745098039	2046	1565	481	6.247255599	90.92627599	84.67902039	3.253638254	481000	46950	434050
9	256	253	3	98.828125	1.171875	2302	1818	484	7.257195322	91.49338374	84.23618842	3.756198347	484000	54540	429460
10	256	251	5	98.046875	1.953125	2558	2069	489	8.259151331	92.43856333	84.179412	4.231083845	489000	62070	426930
11	256	254	2	99.21875	0.78125	2814	2323	491	9.273082911	92.81663516	83.54355225	4.731160896	491000	69690	421310
12	256	253	3	98.828125	1.171875	3070	2576	494	10.28302263	93.38374291	83.10072028	5.214574899	494000	77280	416720
13	255	253	2	99.21568627	0.7843137255	3325	2829	496	11.29296236	93.76181474	82.46885239	5.703629032	496000	84870	411130
14	256	255	1	99.609375	0.390625	3581	3084	497	12.31088579	93.95085066	81.63996487	6.205231388	497000	92520	404480
15	256	255	1	99.609375	0.390625	3837	3339	498	13.32880923	94.13988658	80.81107735	6.704819277	498000	100170	397830
16	256	253	3	98.828125	1.171875	4093	3592	501	14.33874895	94.70699433	80.36824538	7.169660679	501000	107760	393240
17	256	254	2	99.21875	0.78125	4349	3846	503	15.35268053	95.08506616	79.73238563	7.64612326	503000	115380	387620
18	255	254	1	99.60784314	0.3921568627	4604	4100	504	16.36661211	95.27410208	78.90748997	8.134920635	504000	123000	381000
19	256	254	2	99.21875	0.78125	4860	4354	506	17.38054369	95.65217391	78.27163022	8.604743083	506000	130620	375380
20	256	256	0	100	0	5116	4610	506	18.40245898	95.65217391	77.24971493	9.110671937	506000	138300	367700

OOT-

Total Records: 12232

Number of Goods: 11935

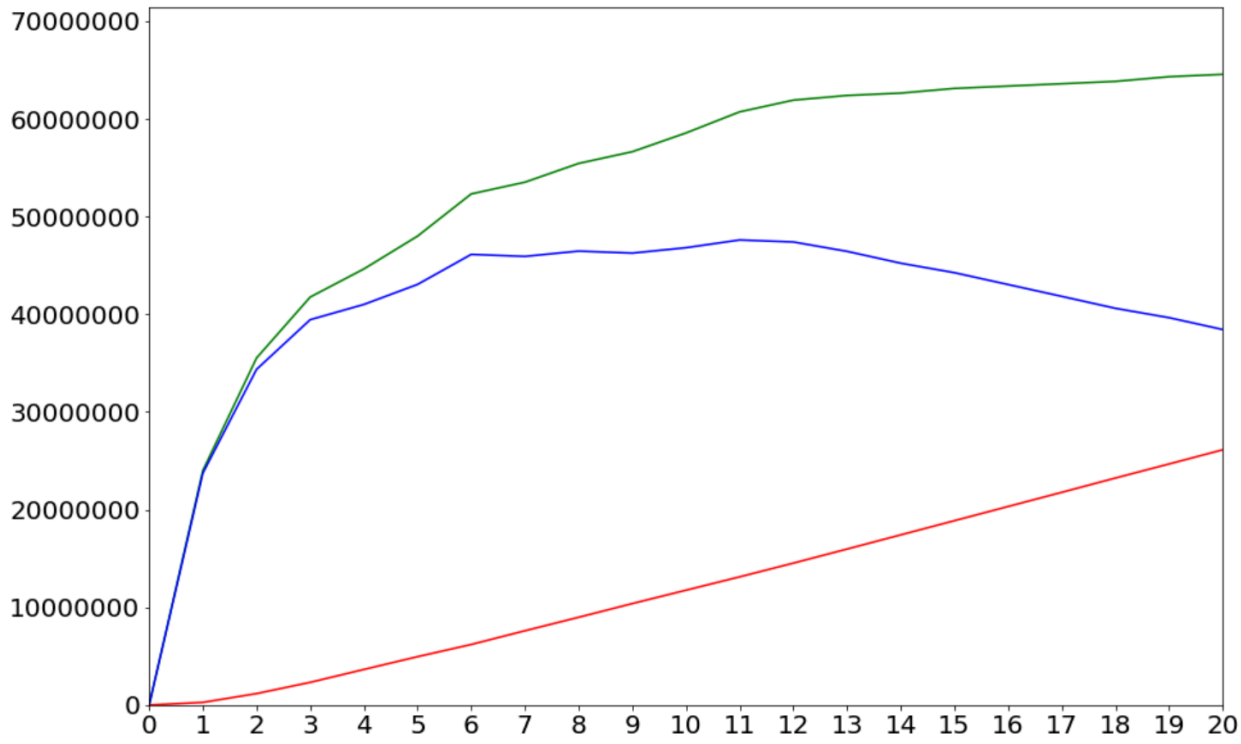
Number of Bads: 297

Fraud Detection Rate: 0.0242

OOT	Records		Goods		Bads		Fraud Rate					
	12232		11935		297		0.0242					
	Bin Stats						Cumulative Stats					
bin	#recs	#g	#b	%g	%b	tot	cg	cb	%cg	FDR	KS	FPR
0	0	0	0	0	0	0	0	0	0	0	0	0
1	122	22	100	18.03278689	81.96721311	122	22	100	0.1843317972	33.67003367	33.48570187	0.22
2	123	75	48	60.97560976	39.02439024	245	97	148	0.8127356514	49.83164983	49.01891418	0.6554054054
3	122	96	26	78.68852459	21.31147541	367	193	174	1.617092585	58.58585859	56.968766	1.109195402
4	122	110	12	90.16393443	9.836065574	489	303	186	2.538751571	62.62626263	60.08751106	1.629032258
5	123	109	14	88.61788618	11.38211382	612	412	200	3.452031839	67.34006734	63.8880355	2.06
6	122	104	18	85.24590164	14.75409836	734	516	218	4.323418517	73.4006734	69.07725488	2.366972477
7	122	117	5	95.90163934	4.098360656	856	633	223	5.30372853	75.08417508	69.78044655	2.838565022
8	123	115	8	93.49593496	6.504065041	979	748	231	6.267281106	77.77777778	71.51049667	3.238095238
9	122	117	5	95.90163934	4.098360656	1101	865	236	7.247591119	79.46127946	72.21368834	3.665254237
10	122	114	8	93.44262295	6.557377049	1223	979	244	8.202764977	82.15488215	73.95211718	4.012295082
11	123	114	9	92.68292683	7.317073171	1346	1093	253	9.157938835	85.18518519	76.02724635	4.320158103
12	122	117	5	95.90163934	4.098360656	1468	1210	258	10.13824885	86.86868687	76.73043802	4.689922481
13	122	120	2	98.36065574	1.639344262	1590	1330	260	11.14369501	87.54208754	76.39839253	5.115384615
14	122	121	1	99.18032787	0.8196721311	1712	1451	261	12.1575199	87.87878788	75.72126798	5.559386973
15	123	121	2	98.37398374	1.62601626	1835	1572	263	13.17134478	88.55218855	75.38084377	5.977186312
16	122	121	1	99.18032787	0.8196721311	1957	1693	264	14.18516967	88.88888889	74.70371922	6.412878788
17	122	121	1	99.18032787	0.8196721311	2079	1814	265	15.19899455	89.22558923	74.02659467	6.845283019
18	123	122	1	99.18699187	0.8130081301	2202	1936	266	16.22119816	89.56228956	73.34109141	7.278195489
19	122	120	2	98.36065574	1.639344262	2324	2056	268	17.22664432	90.23569024	73.00904591	7.671641791
20	122	121	1	99.18032787	0.8196721311	2446	2177	269	18.24046921	90.57239057	72.33192136	8.092936803

8. Financial Curves and Recommended Cutoff

Assuming we lose 20$ for every wrongly classified fraud transaction, and gain 400$ for every correctly classified transaction. Also, since we are using a fraction of the data (100,000) here out of what could possibly be millions of records, and the OOT data only takes into account 2 months of the year as its time period, we use a multiplier of **(12/2)*(10,000,000/100,000)** to forecast the savings my model will help the company make. We can plot a financial curve as shown below:



The Green line depicts every Fraud caught, the Red line is the Revenue Lost, and the Blue line depicts our overall savings.

From my plot, I can make a recommendation for the cut-off to be set at 4%.

This ensures that the company can maximize on savings, expect customer retention and detect fraudulent transactions at the same time.

9. Summary

This report outlines the comprehensive process undertaken to analyze and model credit card transaction data for fraud detection. The project involved several critical steps, starting with data cleaning and culminating in the selection and evaluation of a final model. Here is a summary of each major phase:

Data Description
The dataset consisted of credit card transaction details, including the amount, card number, merchant information, state and zip code, and transaction date. The data covered 97,852 transactions from 1,645 credit card users interacting with 13,091 merchants across 227 states in 2010. The fields were categorized into numeric and categorical variables.

Data Cleaning
Data cleaning was a meticulous process involving several fields such as Merchant Number, Merchant State, and Merchant Zip. Missing values were handled using various imputation techniques, including mapping descriptions to their corresponding numbers and states, and labeling unresolvable entries as 'unknown' or 'foreign'.

Variable Creation
Numerous variables were engineered to capture different aspects of the transaction data, such as transaction frequency, amount, and variability over different time windows. This included creating features like day of the week, risk scores, transaction counts, and various statistical measures (mean, median, maximum). In total, over 3,200 variables were created to enrich the dataset.

Feature Selection
Given the high dimensionality of the dataset, feature selection was crucial. The process involved using both forward and backward selection methods on LightGBM (LGBM) and Random Forest models. The top 20 variables were selected based on their performance in the models, ensuring a balance between computational efficiency and model accuracy.

Preliminary Model Exploration
Several models were explored, including Logistic Regression, Decision Tree, Random Forest, Boosted Tree (such as Gradient Boosting), and Neural Networks. Initial trials were conducted to understand their performance and to tweak hyperparameters for optimal results.

Final Model Selection

Based on the preliminary explorations, the LGBM model was chosen as the final model due to its high and consistent performance across training, testing, and out-of-time datasets. The model was optimized with specific parameters (e.g., number of estimators, number of leaves) to balance between accuracy and overfitting.

Model Performance

The LGBM model demonstrated robust performance with scores of 0.793 on the training set, 0.785 on the testing set, and 0.593 on the out-of-time set. These results indicate a good balance between model accuracy and generalization.

Financial Curves and Recommended Cutoff

A financial analysis was performed assuming a loss of $20 for each wrongly classified fraud transaction and a gain of $400 for each correctly classified transaction. The analysis recommended setting the cutoff at 4% to maximize savings while effectively detecting fraud.

Conclusion

The project successfully identified a reliable and effective model for fraud detection in credit card transactions. The process demonstrated the importance of thorough data cleaning, feature engineering, and model selection in building a robust predictive system. The final model provides a solid foundation for deploying a real-world fraud detection system, balancing accuracy, generalization, and financial impact.