MGTA 463 Fraud Analytics Project 1

Credit Card Transaction Fraud Detection

Joshua Chen

May 9th, 2024

Table of Contents

Executive Summary

In recent years, fraudulent transactions have emerged as a significant threat to the financial sector. Fraudsters often use various sophisticated tactics to exploit vulnerabilities in payment systems, causing substantial losses to individuals and organizations. According to Time.com, The U.S. Federal Trade Commission (FTC) reported $8.8 billion lost by consumers due to fraud in 2022. In 2023, U.S. adults lost a record $10 billion to fraudsters. Card transaction fraud detection is a critical area of focus to safeguard against these illicit activities.

In response to this escalating threat, I conducted a rigorous analysis of credit card transaction data, leading to the development of a sophisticated fraud detection model. This model was trained and tested using an Out-of-Time (OOT) approach. This rigorous testing protocol ensures our model remains highly effective against new and evolving fraudulent patterns, not just under historical conditions.

To summarize this project, the best model was a Neural Network, specifically the Multi-Layer Perceptron classifier. The overall average accuracy is 0.79 for training data, 0.78 for testing data, and 0.57 for out of time data. From this, I selected FDR@5% as the cutoff, which I expect will capture 71% of the fraud transactions and save approximately $46.5 million per year if deployed.

Data Description

The dataset is **Card Transaction** data, which contains **real card transaction records** from a US government organization, likely from somewhere in Tennessee. The data came from an author's website. The author, Mark Nigrini, is an expert in applying Benford's law for forensic accounting. There are **10 fields** and contains **97852 records**. Note: The fraud labels were made up based on the Professor's professional expertise on how credit card fraud behaves.

Overview of Transaction Data

Data Size: 97,852 records
Variables: 10 fields (8 categorical; 2 numerical)
Time Frame: 2010-01-01 - 2010-12-31

Summary Tables

Numeric Fields Table

Field Name	Field Type	# Records w/ Values	% Populated	# Zeros	Min	Max	Mean	Standard Deviation	Most Common
Date	numeric	97,852	100%	0	2010-01-01	2010-12-31	2010-06-25	98.92 days	2010-02-28
Amount	numeric	97,852	100%	0	0.01	3,102,045.53	425.466	9,949.8	3.62

Categorical Fields Table

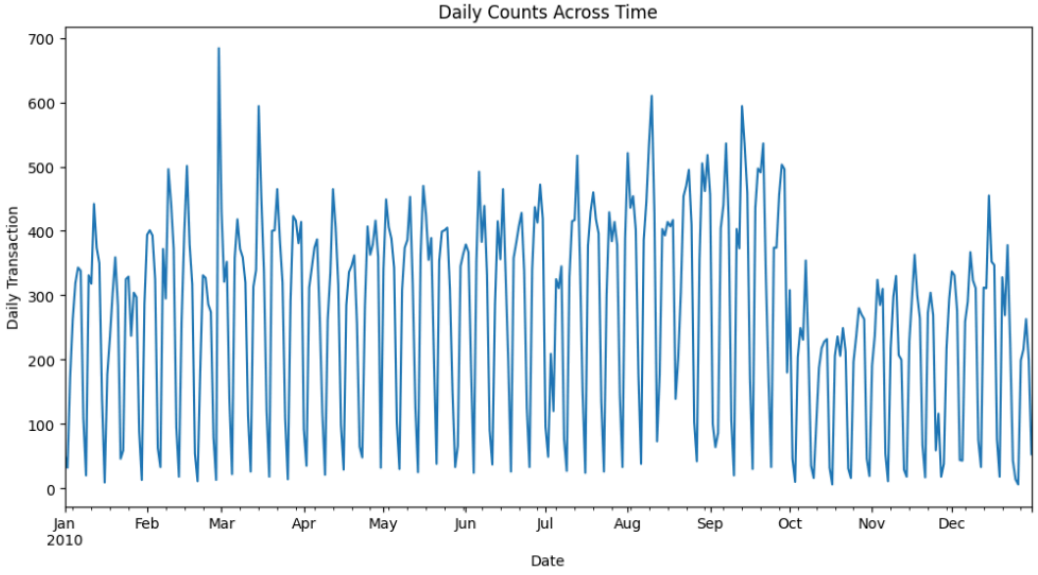
Field Name	Field Type	# Records w/ Values	% Populated	# Zeroes	# Unique Values	Most Common
Recnum	Categorical	97,852	100%	0	97,852	1
Cardnum	Categorical	97,852	100%	0	1,645	5142148452
Merchnum	Categorical	94,455	96.5%	0	13,091	930090121224
Merch description	Categorical	97,852	100%	0	13,126	GSA-FSS-ADV
Merch state	Categorical	96,649	98.8%	0	227	TN

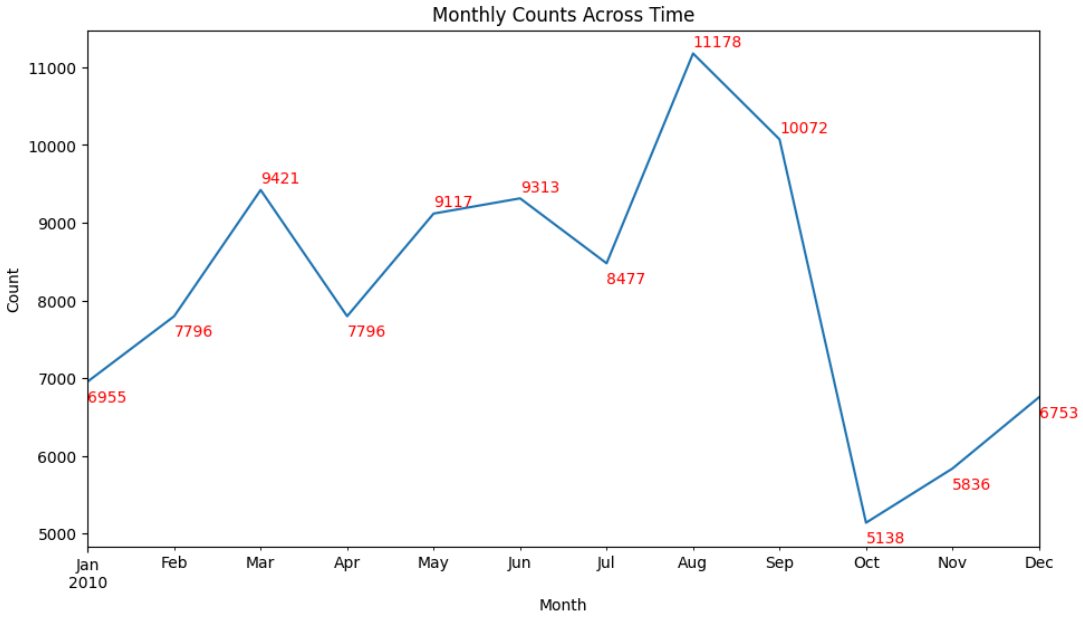
Merch zip	Categorical	93,149	95.2%	0	4,567	38,818.0
Transtype	Categorical	97,852	100%	0	4	P
Fraud	Categorical	97,852	100%	95,805	2	0

Important Field Distributions

Date

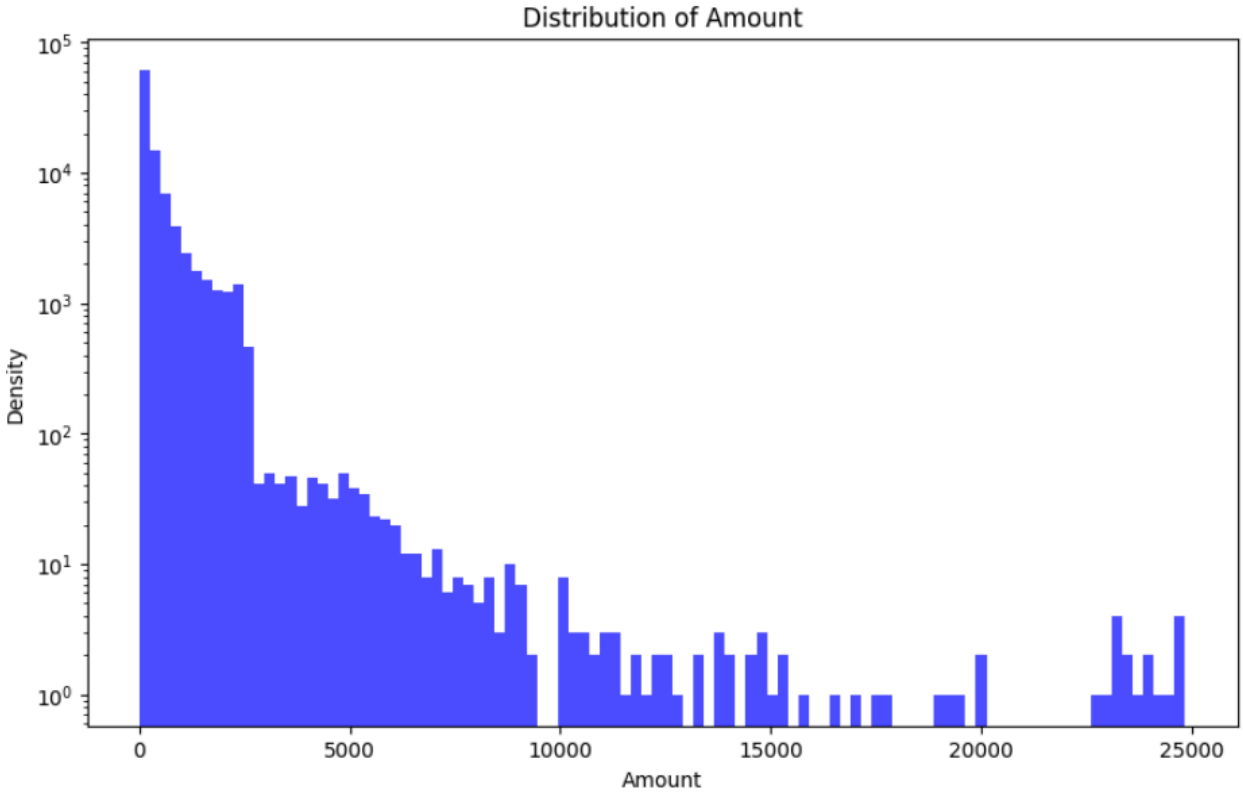
Description: Transaction date. The first distribution shows the number of daily transactions across time. The second distribution shows the number of monthly transactions across time.





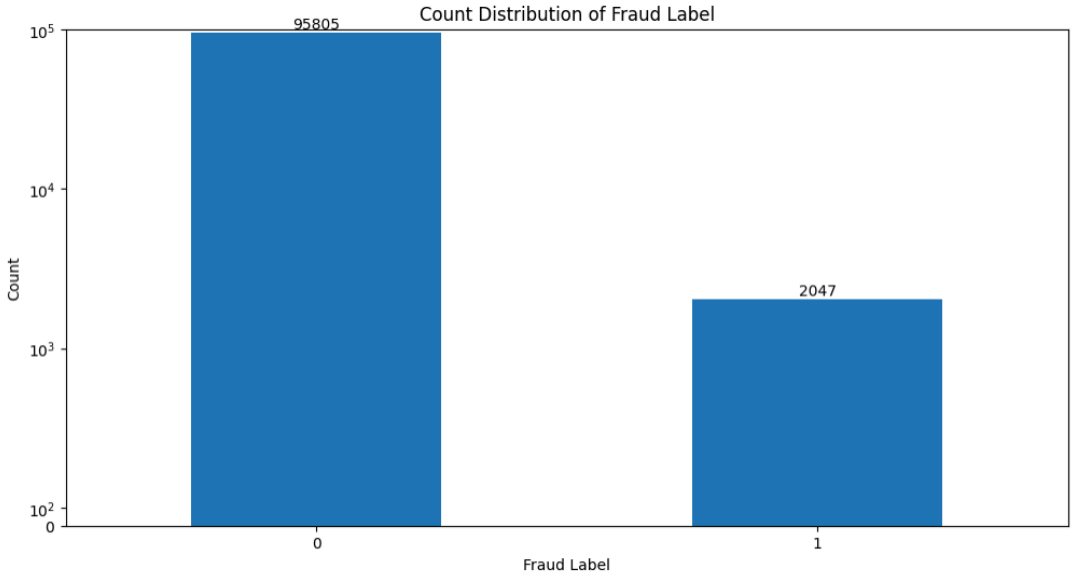
Amount

Description: Transaction amount. The first distribution shows a boxplot of all the transactions amount. As seen in the distribution, there is a **clear outlier at 3102045.53**. The second distribution shows a histogram of all transaction from 0 to 99th percentile. The third distributio shows a histogram excluding the outlier listed above.



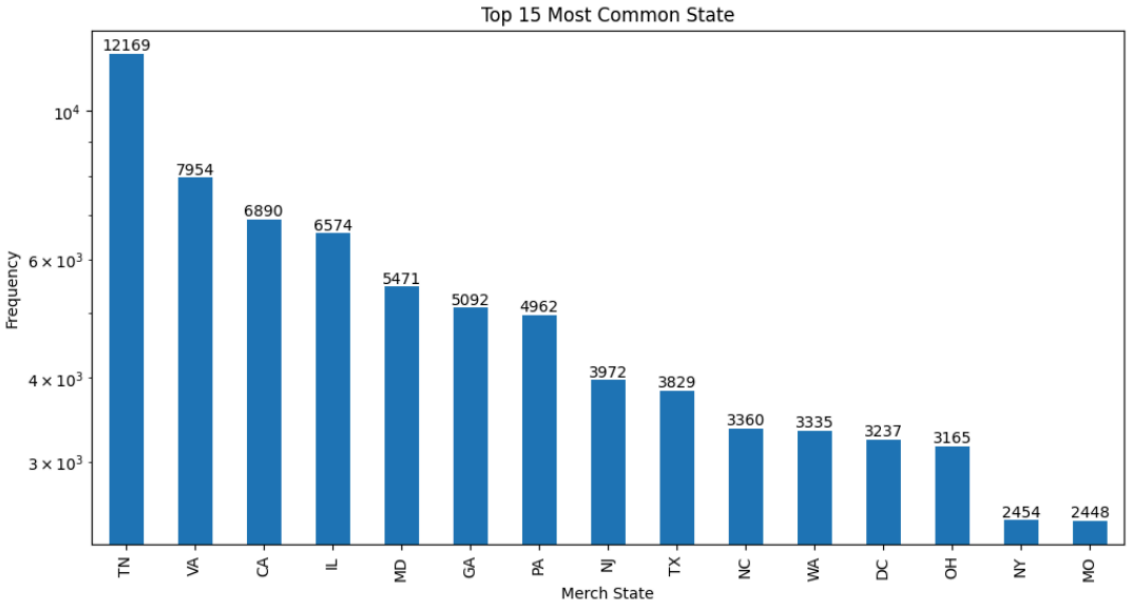
Fraud

Description: Fraud identification label. Fraud = 0 (not fraudulent), Fraud = 1 (Fraud identified). The total count of **Fraud = 0 is 95805** and total count of **Fraud = 1 is 2047**. Roughly **2%** of all transactions were labeled as frFraudulent.



Merch State

Description: State of the merchandise bought in the transaction. The first distribution shows the top 15 values of the merch state. The **most common is TN** (Tennessee), with count of **12169**.



Data Cleaning

First steps (Outlier removal and Data Exclusions)

To start the data cleaning process, I first **loaded the dataset** and **converted the 'Date' variable** into a standardized datatime format from pandas.

The original dataset contains 97852 rows and 10 columns, I then dropped columns that only contains null values and also dropped rows that only contains null values. The resulting dataset is still 97852 rows and 10 columns, meaning that no rows and columns only contains null values.

I then **excluded transactions types that are not P**. Because most (99.6%) of the data type is P, I inferred that it means purchase, and since I'm not sure what the other transaction type are, I have decided to exclude them.

I also **removed the outlier record where the transaction amount is 3M**, significantly higher than all other transactions (second highest being 47.9k). The outlier is likely recorded in a different currency. Looking at the transaction, it appears to be a purchase through a Mexican retailer. However, converting 3M pesos to USD still results in 180k, which is still significantly higher then the second highest amount (47.9k). Therefore, I will be removing the 3M entry.

By excluding the outlier and transactions types that are not P, the data now contains 97496 rows and 10 columns.

I then **checked for Null values** within the each columns and found that variables Merchnum, Merch state, and Merch zip all have Null values, meaning that I will have to impute those missing values. Merchnum has **3220** missing, Merch state has **1028** missing, Merch zip has **4347** missing.

Inputations - Merchnum

- All **Merchnum** IDs comprise of 9 digits, so an entry of **"0" was assumed to represent a null value**.
- **Changing entries of "0" to null** values resulted in **3279** total null values.
- A strong correlation between **Merchnum** and **Merch description** (with no missing **Merch description** values) enabled using known **Merch description** and **Merchnum** pairs to fill missing **Merchnum** values:
    - This approach resolved **1164** records, leaving **2115** still missing.

- Identified **Merch description** entries as adjustment transactions, which typically lack a **Merchnum**:
    - Set **Merchnum** to "unknown" for **Merch description** values containing "RETAIL CARD ADJUSTMENT" or "RETAIL DEBIT ADJUSTMENT."
    - Addressed an additional **694** missing values, leaving **1421** unresolved.
- For the remaining 1421 records lacking a **Merchnum**:
    - Identified 515 unique **Merch description** values.
    - Assigned a new and unique **Merchnum** to each distinct **Merch description** by adding 1 to the current maximum Merchnum value.

Imputations - Merch state

- Initially encountered **1028** missing Merch state values.
- Recognized the **1-to-1 mapping between state and zip code**:
    - Created a **dictionary** associating known **Merch zip with corresponding Merch state** values from the data.
    - **Manually added zip codes** from regions like **Puerto Rico, California, and Texas** to this dictionary.
- Created separate dictionaries to correlate:
    - Merch num with Merch zip.
    - Merch description with Merch zip.
- **Applied dictionary mappings** to reduce missing Merch state values:
    - Reduced to **954** using the zip code dictionary.
    - Reduced to **953** using the Merch num dictionary.
    - Reduced to **952** using the Merch description dictionary.
- Acknowledged that some transactions are adjustments and lack a specific state:
    - Set **Merch state to "unknown"** where Merch zip was missing and Merch description indicated an adjustment (e.g., "RETAIL CARD ADJUSTMENT," "RETAIL DEBIT ADJUSTMENT").
    - Reduced missing Merch state values to **297**.
- Final resolution of the remaining **297** missing states:
    - **Labeled them as "unknown."**
- Designated **all entries outside of U.S. states as "foreign,"** crucial for highlighting potential international transactions involving fraud.

Imputations - Zip

- Initially encountered **4347** missing Merch zip entries.
- Recognized the **1-to-1 mapping relationship** between Merch zip and both Merchnum and Merch description:

        - **Created separate dictionaries** to associate known **Merchnum and Merch description** with corresponding **Merch zip** values from the data.
    - Applied dictionary mappings:
        - Reduced missing values to **2670** using Merchnum - Merch zip pairs.
        - Further reduced missing values to **2625** using Merch description - Merch zip pairs.
    - Imputed missing zip codes based on the **correlation between known states and their most populous zip codes**:
        - Imputed the **most populous zip code** from the corresponding state for records where the Merch state is known but Merch zip is missing.
        - Reduced the number of missing zip codes to **1216**.
    - For the remaining 1216 missing zip codes:
        - **Labeled these entries as "unknown."**

Variable Creation

How Fraud Occurs

The financial fraud I'am discovering in this project is Card Not Present Fraud (CNP). CNP Fraud is mainly caused by skimming devices that are installed on top of credit card readers. The skimmer is capable of gaining enough information to make a counterfeit credit card and frauders will be able to make transactions with them. Hence, highly suspicious data can be a transaction with the same card but made from different locations.

Variables Created

I created as many fields as possible from the original data to observe possible pattern of behavior when CNP fraud occurred. The created fields follow 4 main principles:

1. Amount Variable: For each entity, the statistical measures within certain amount of days
2. Frequency Varialbe: For each entity, the occurrence within certain amount of days
3. Velocicy Change Varialbe: Ratio in between two entities' statistical measures
4. Day-since Varialbe: For each entity, the statistical measures within a certain time span

The following table contains all variables created and explanation for each creation

Description	# Variables Created	Cumulative Variables
Shipping Transaction or Not Whether or not the Transaction amount was the FedEx shipping charge	1	1
Day of Week Target Encoding Contains target encoded day of week, where the value shows risk of fraud for that particular day.	1	2
Linking Entities Contains either 2 or 3 variables (cardnum, Merchnum, Merch zip, Merch description, Dow, Merch state) grouped together. I.e. adding cardnum + merchnum to get card_merch.	21	23
Merch state Target Encoding Contains target encoded Merch state, where the value shows risk of fraud for that state	1	24
Merch zipTarget Encoding Contains target encoded Merch zip, where the value shows risk of fraud for that zip code	1	25
Days Since variable Amount of days since an application with that entity was seen **Velocity** Amount of records with the same entity attributes over the past [0,1,3,7,14,30,60] days, including calculations of frequency and various statistics related to the transaction amounts (average, maximum, median, total, actual/average, actual/max, actual/median, actual/total).	1472	1499
Relative Velocity by Count Number of applications with same entities seen in the past [0,1] day divided by the number of applications with those same entities seen in the last [7,14,30,60] days **Relative Velocity by Amount** Total transaction amount with same entities seen in the past [0,1] day divided by total transaction amount with those same entities seen in the last [7,14,30,60] days	368	1867
Relative Velocity Day Since Ratio For each entity, divide the number of applications with the same	184	2051

entities seen in the past [0,1] day by number of applications with those same entities seen in the last [7,14,30,60] days since the last transaction involving that entity plus 1 day.		
Amount Difference Variability Calculates the average, maximum, and median differences in transcrtiion amount for each entity over [0,1,3,7,14,30] days	414	2465
Unique Entity Count Number of unique unique values that one entity takes relative to another in [1,3,7,14,30,60] days.	880	3345
Transaction Amount Categorization Divides the transaction amounts into five quantile-based categories, ranging from 1 to 5.	1	3346
Foreign Identification Whether or not the zip code is in the U.S	1	3347
Month Target Encoding Contains target encoded Month of transaction, where the value shows risk of fraud for that particular month.	1	3348
Decay on Transaction Amounts Difference Variability First add a decay to transaction amounts, diminishing the importance of older data and emphasizing more recent transactions, then recalculate the Amount Difference Variability	414	3762
Entity-Day Average Calculates the ratio of each entity's daily transaction amount to the average transaction amount of all entities on the same day.	23	3785
Holiday Purchase Detection Determines whether the transaction was made on a holiday.	1	3786

Feature Selection

Feature Selection Process

The feature selection process follows the general pipeline below:

1. I first run a KS Filter, which has no stochasticity in nature, and filter down to only num_filter variables left
2. I then run a Wrapper of wrapper_size n, which returns the nbest variables ranked in order, this allows me to check for at which number of feature size the performance appears to saturate.

Setup:

In the filter step (step 1), I run a univariate KS filter, which returns the same variables across every iterations.

In the wrapper step (step 2), I can run a total of 4 combinations:

1. LBGM Classifier with SFS Forward Selection
2. LBGM Classifier with SFS Backward Selection
3. Random Forest classifier with SFS Forward Selection
4. Random Forest classfieir with SFS Backward Selection

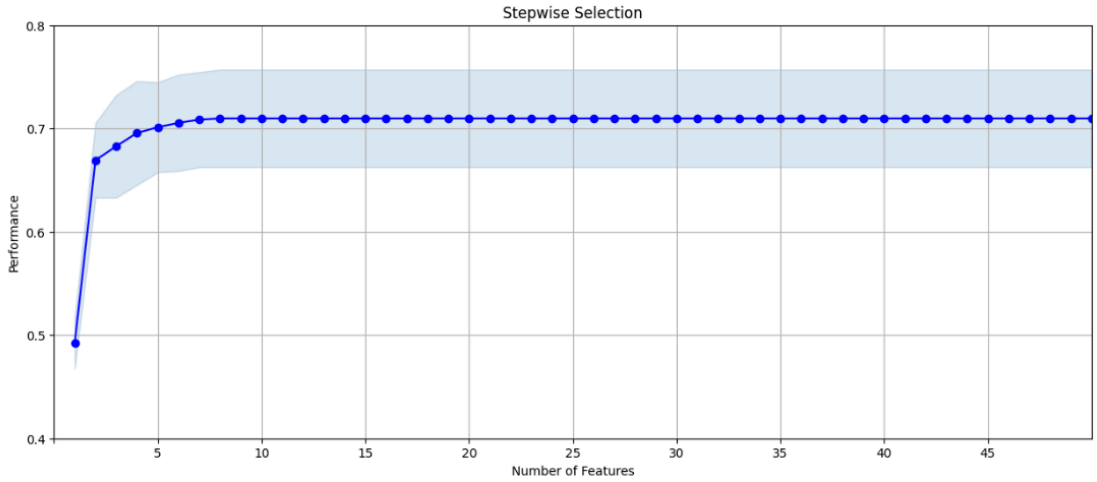
Note: LBGM has no stochasticitiy, so every iteration will return the same results. Random Forest, on the other hand, has stochasticitiy, so every iteration will likely return different results. The wrapper method is the SFS.

Round 1 Experimentations

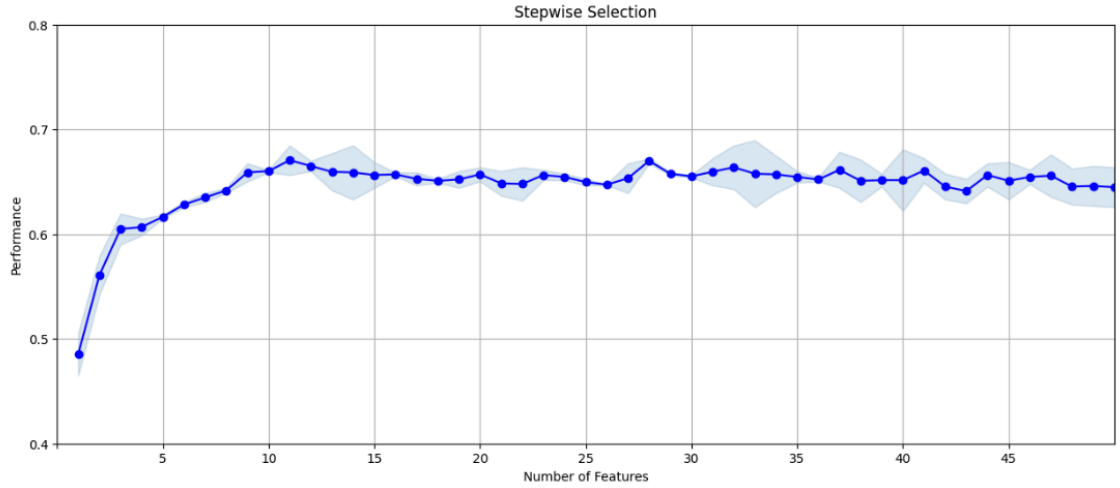
My initial trials used a **num_filter size of 150** and a **wrapper_size of 50** and tested the different wrappers. The results are shown below:

Trials	Performance (50 features)	Run Time
LGBMClassifier with SFS Forward	0.715217	13min26s
RFClassifier with SFS Forward	0.655928	47min 53s
RFClassifier with SFS Forward 2	0.670543	41min4s
RFClassifier with SFS Forward 3	0.667911	42min53s
LBGMClassifier with SFS Backward	0.719403	1h31min

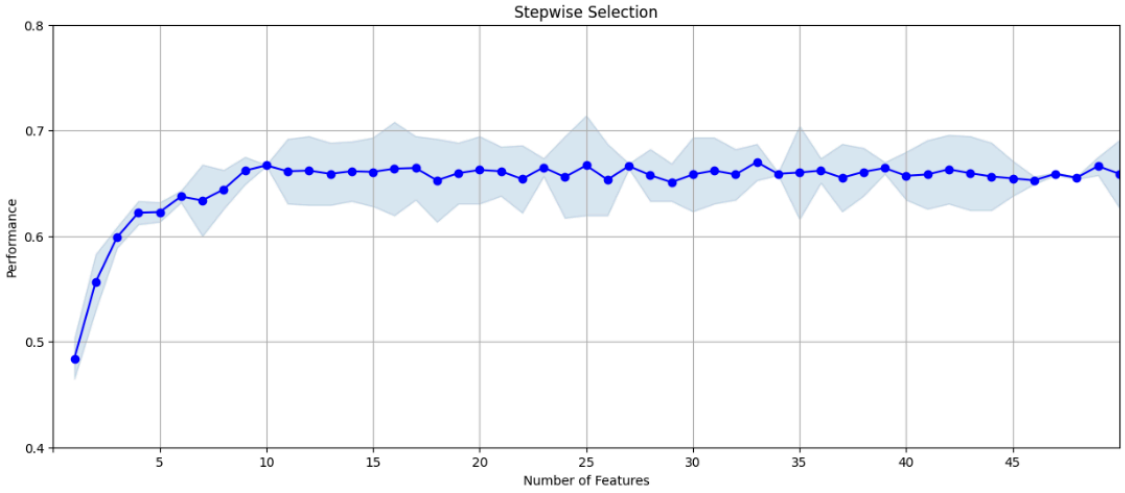
LGBMClassifier with SFS Forward Graph



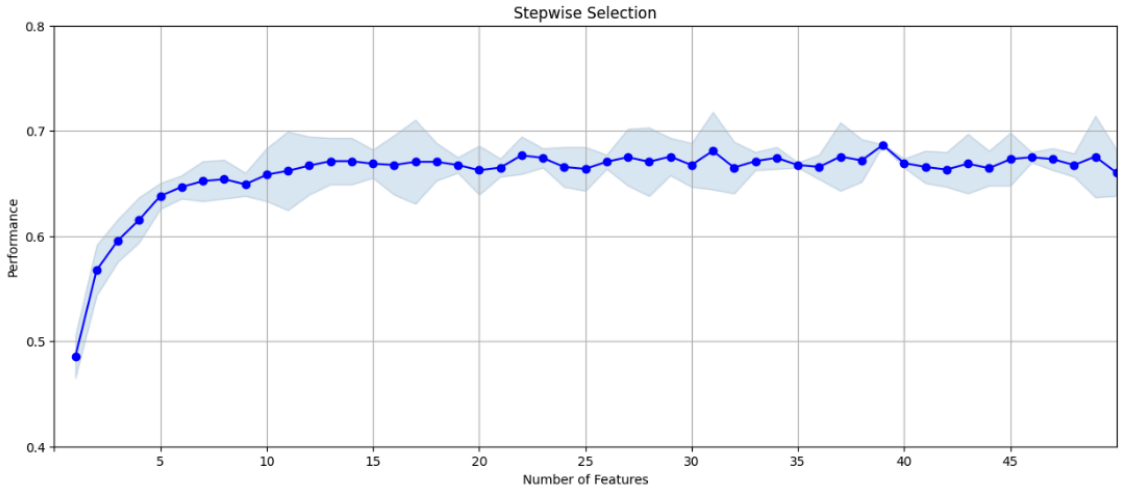
RFClassifier with SFS Forward Graph



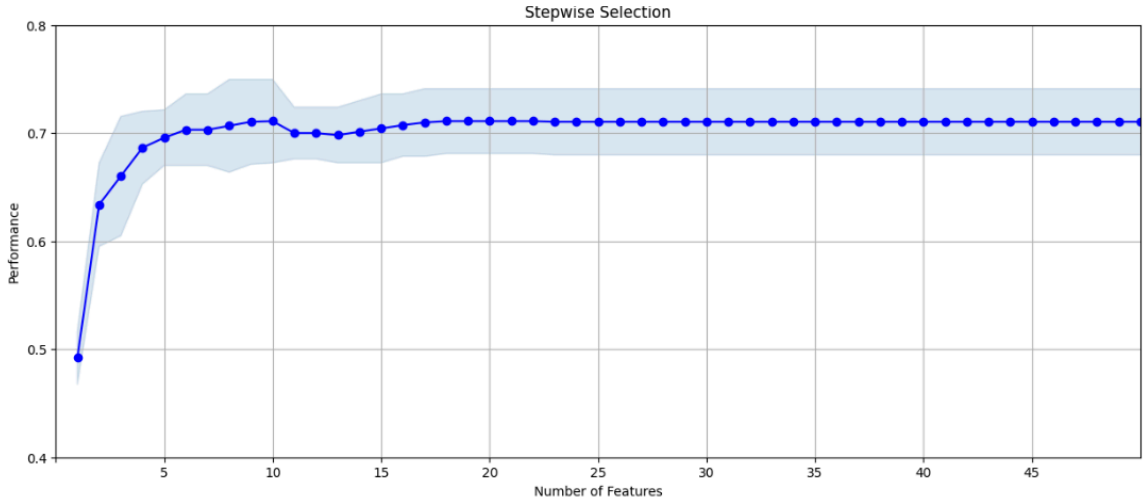
RFClassifier with SFS Forward 2 Graph



RFClassifier with SFS Forward 3 Graph



LBGMClassifier with SFS Backward Graph



With the round 1 experimentations, it appears that LBGM Classifier with SFS Forward Selection is the best, as it not only resulted in one the best performance (very similar to LBGM Classifier with backward) and it also took significantly less time (less than ¼ compared to the second shortest time). The performance cheats also seems to saturate around 10 features.

Note: I decided not to run the Random Forest Classifier with SFS Backward Selection for a couple of reasons. Firstly, the Random Forest Classifier already underperformed in SFS Forward Selection compared to the LGBM Classifier. Furthermore, the LGBM Classifier with SFS Backward, while similar in performance to its SFS Forward counterpart, required significantly more time. Therefore, it seemed unlikely that pairing the Random Forest Classifier with SFS Backward would surpass the efficiency and effectiveness of the LGBM Classifier with SFS Forward, justifying my decision to conserve resources and not proceed with that combination. However, it is possible that the RFClassifier with SFS Backward combination could outperform LBGMClassifier with SFS Forward.

Round 2 Experimentations

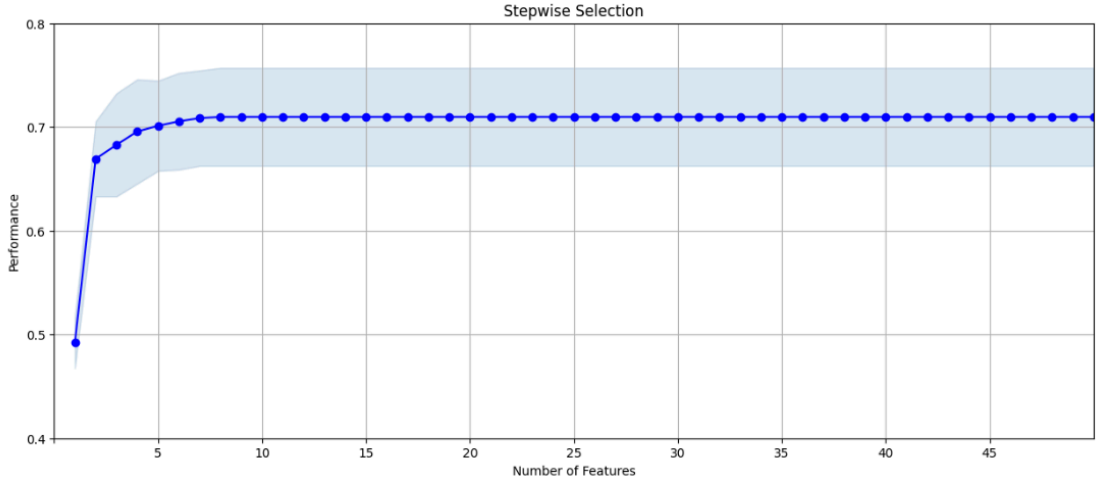
With LBGM Classifier and SFS Forward pair identified as the best, round 2 aimed to test the parameters of the forward function and LGBM Classifier. The filter_num is still set at 150 and wrapper_num is still set at 50.

The results are shown below:

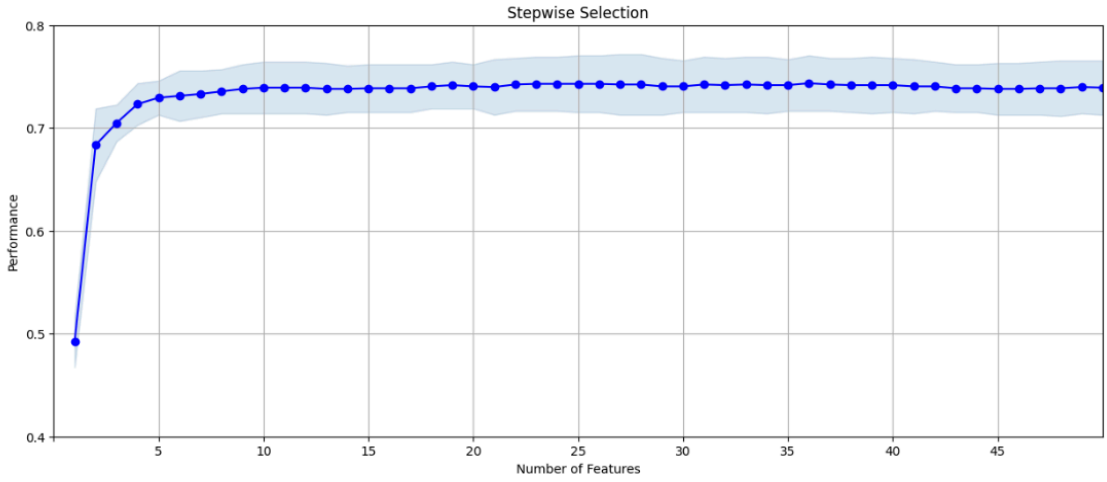
Trials	Performance (50 features)	Run Time
LBGM with 10 estimators, SFS Forward with 2 fold CV	0.715217	13min26s

LBGM with 100 estimators, SFS Forward with 2 fold CV	0.741014	54min
LBGM with 5 estimators, SFS Forward with 2 fold CV	0.676589	38min
LBGM with 5 estimators, SFS Forward with 4 fold CV	0.741952	1hr30min

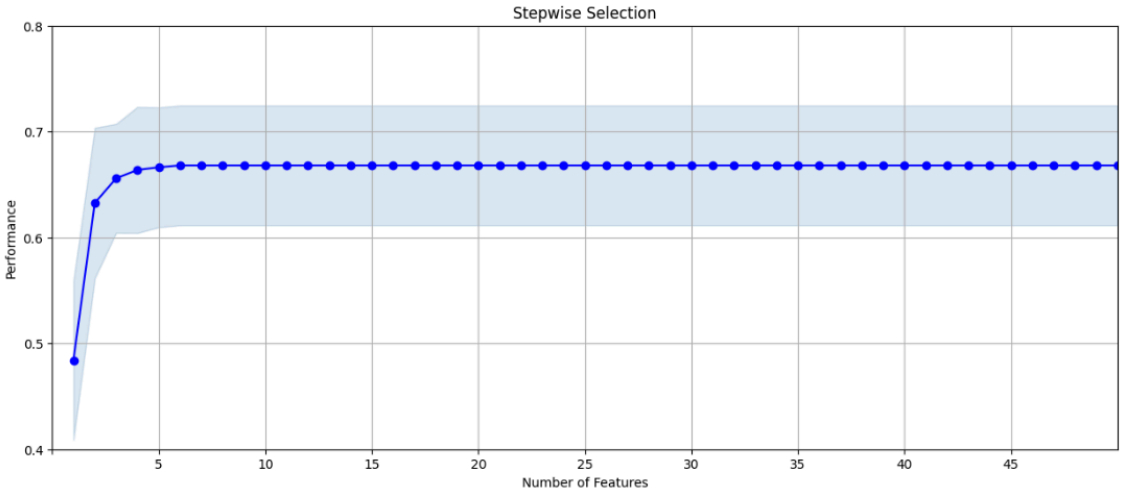
Baseline: LBGM with 10 estimators, SFS Forward with 2 fold CV Graph



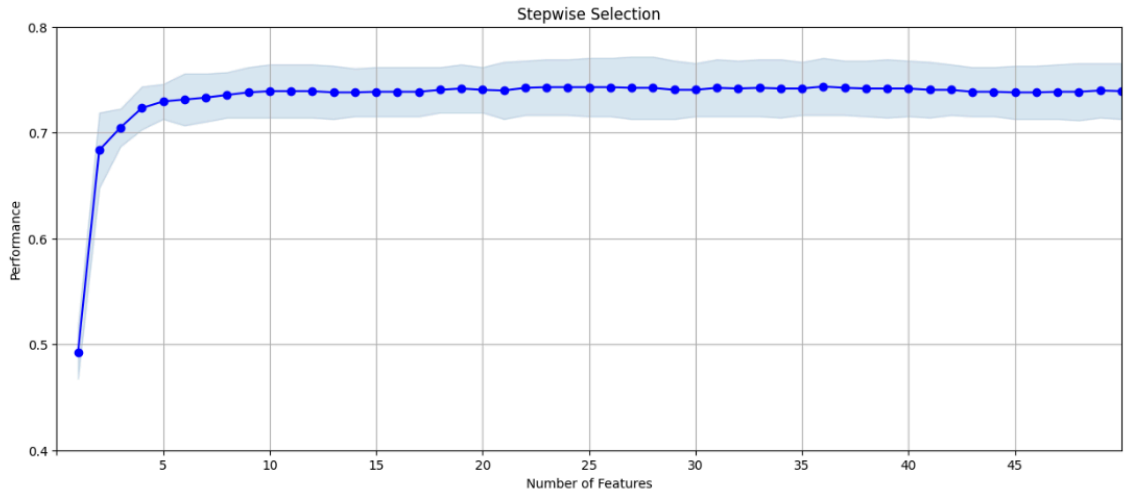
LBGM n_estimators increase to 100, SFS Forward with 2 fold CV Graph



LBGM n_estimators decrease to 5, SFS Forward with 2 fold CVGraph



LBGM with 100 estimators, SFS Forward with 4 fold CV Graph



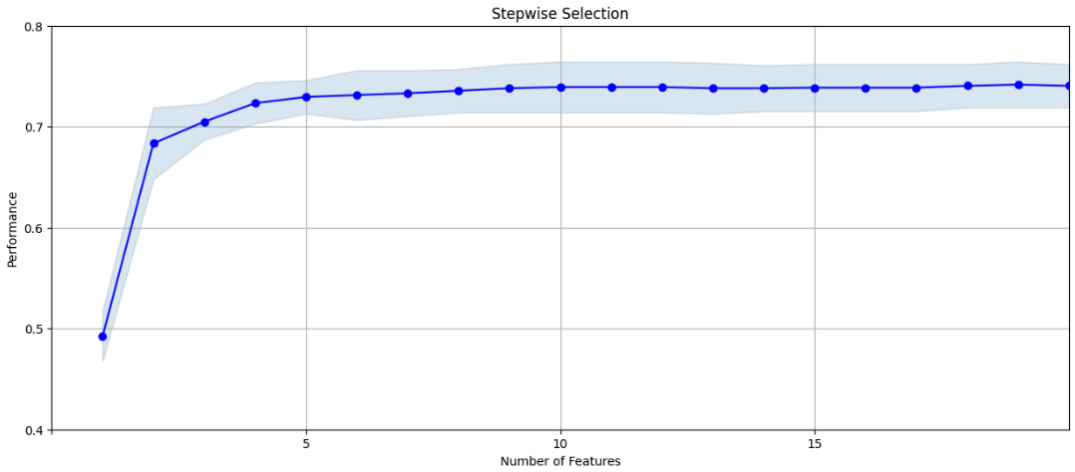
From round 2, it appears that increasing n_estimators to 100 increases the performance. Further increasing cross validation rounds in the SFS forward selection also seems to slightly increase the performance, however it is not worth the trade off as it takes significantly longer. Moving on, I will be keeping the LGBMClassfier with n_estimators as 100 and SFS forward selection with original parameters. In addition, the performance graphs from round 2 also seems to saturate around 10 features, similar to round 1.

Final Experimentations

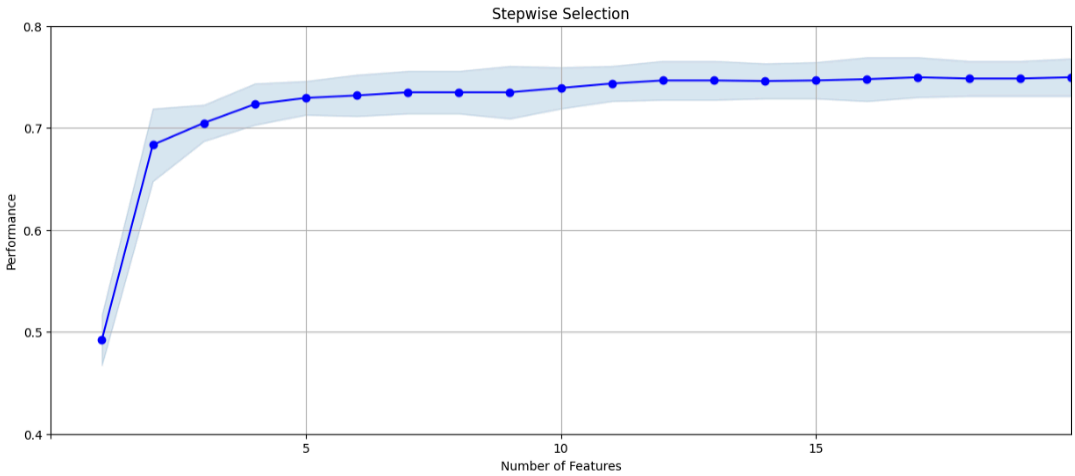
In the final experimentations, I will test the effect of increasing filter size (up to 20% of original data, so around 537), while keeping the LGBMClassifier with n_estimators as 100 and SFS forward selection with 2 fold CV. Since we have identified that it saturates around 10 number of features, we will also lower num_wrapper to 20 for a faster training time. The final model will be selected based on the best performance.

Trials	Performance (20 features)	Run Time
Filter_num = 150	0.740491	2min 56s
Filter_num = 300	0.749693	6min4s
Filter_num = 537	0.747853	10min 38s

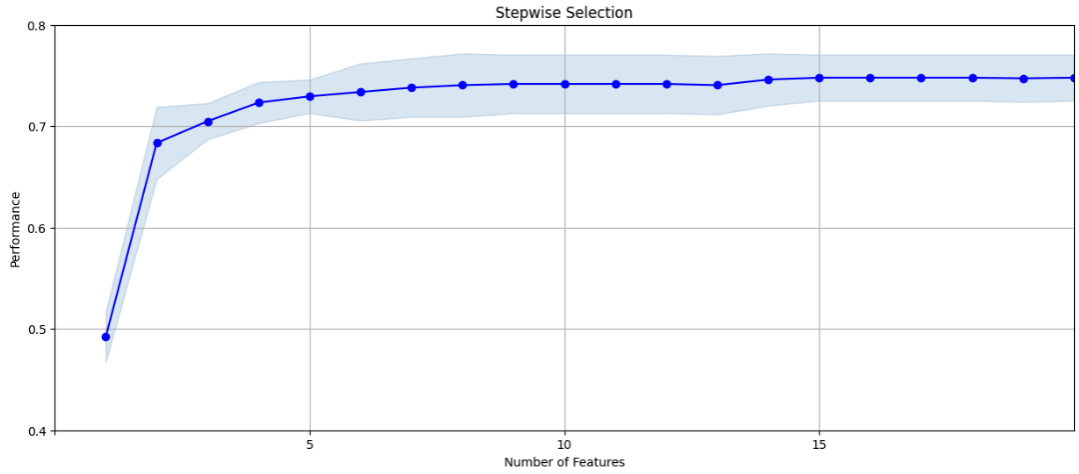
Baseline: **filter_num = 150 Graph**



Filter_num = 300 Graph



Filter_num = 537 Graph



It appears that with Filter_num = 300, the performance score is the best, therefore, we will take the 20 variables returned from the wrapper model with filter_num = 300.

Final Variables

Note: Across all 3 trials in the final experimentations, regardless of the filter_num size, the returned wrapper order 1-5 and the variables were the same.

Wrapper Order	Variable Name	Filter Score
1	Cardnum_unique_count_for_card_state_1	0.476067
2	card_merch_total_3	0.320614
3	card_state_max_14	0.305946
4	Card_dow_vdratio_0by14	0.479086
5	Card_dow_unique_count_for_Card_Merchdesc_1	0.447250
6	Merchnum_total_7	0.284703
7	state_des_total_1	0.311366
8	Card_dow_unique_count_for_state_des_1	0.447238
9	card_state_total_7	0.339462
10	Cardnum_total_60	0.340297
11	Merchnum_desc_total_1	0.304969
12	Cardnum_unique_count_for_card_zip_3	0.464311

13	merch_state_total_7	0.284715
14	Cardnum_day_since	0.432169
15	card_zip_max_7	0.310980
16	Cardnum_total_30	0.410799
17	Card_dow_unique_count_for_merch_zip_14	0.401402
18	Merchnum_desc_total_7	0.285040
19	Card_dow_day_since	0.432169
20	merch_state_total_0	0.292798

Preliminary Model Exploration

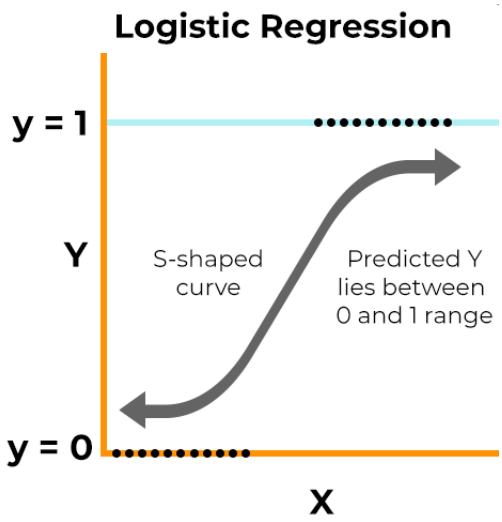
I explored various models in the preliminary phase, including Logistic Regression, Decision Tree, Random Forest, LightGBM, and Neural Network.

Overview of Models

Logistic Regression

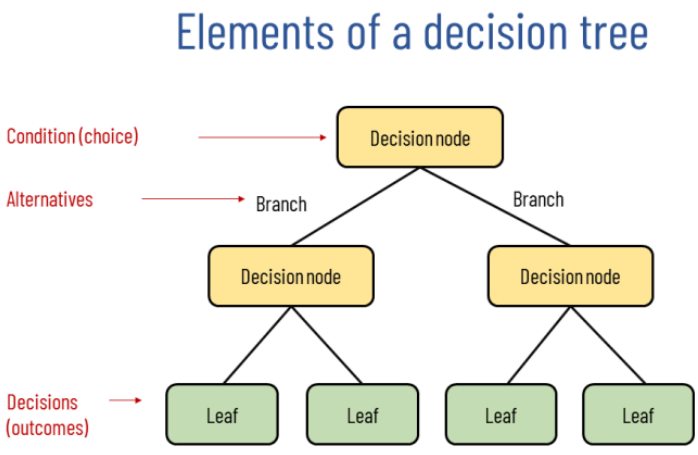
Logistic Regression is a fundamental statistical model used for binary classification problems, making it ideal for fraud detection. It estimates the probability of an outcome (e.g., fraud or non-fraud) by applying the logistic (sigmoid) function to a linear combination of input features. The result is an S-shaped curve that maps predicted probabilities to either class. While it's a

relatively simple and interpretable model, it can be powerful with well-chosen features and regularization methods like L1 (Lasso) or L2 (Ridge) to prevent overfitting.



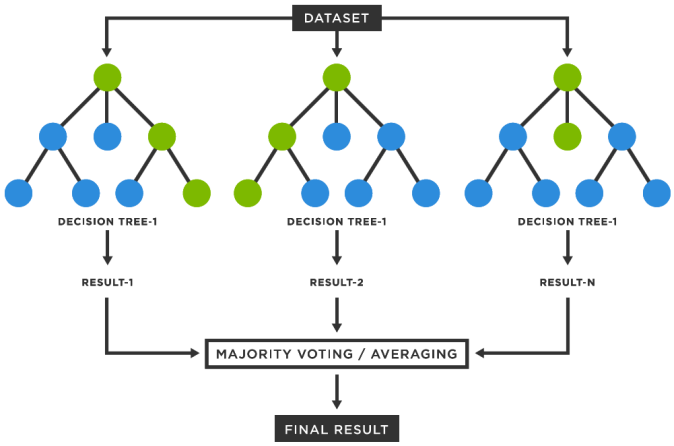
Decision Tree

Decision Tree model classifies transactions by recursively partitioning the data based on feature values. Each node in the tree represents a feature, and each branch represents a decision rule that leads to child nodes or terminal leaves, which contain the final classification (e.g., fraud or non-fraud). Decision Trees can capture complex, non-linear relationships between features and outcomes. They are intuitive and interpretable, though prone to overfitting without pruning or limiting tree depth.



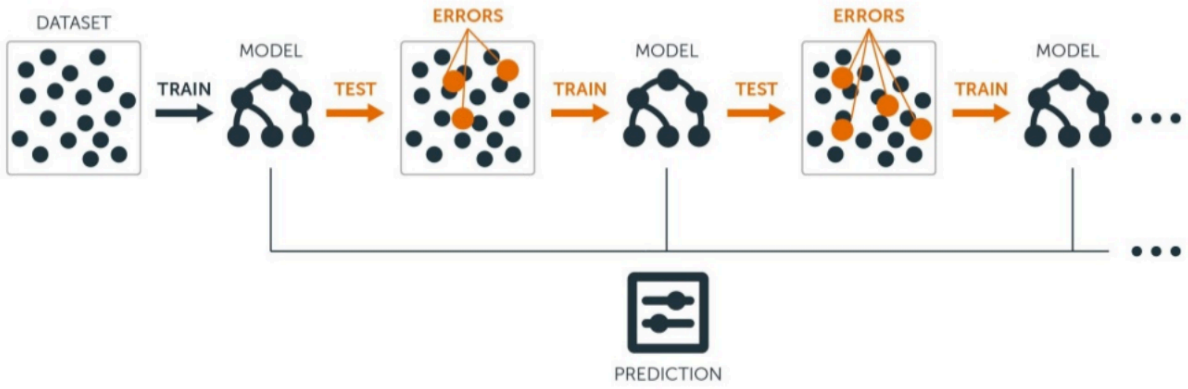
Random Forest

Random Forest is an ensemble learning method that mitigates the overfitting problem of Decision Trees by constructing multiple trees and aggregating their predictions. Each tree is trained on a random subset of the data and features, leading to diverse decision boundaries. The model classifies transactions by taking the majority vote across all trees. Random Forest improves prediction accuracy and stability, making it suitable for fraud detection tasks where data can be imbalanced and noisy.



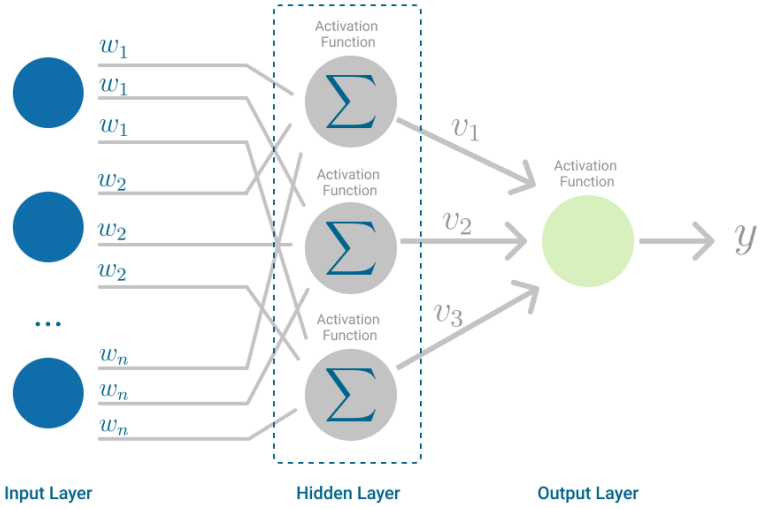
LightGBM

LightGBM employs a leaf-wise tree growth strategy. This approach allows the model to grow the decision tree with the highest loss reduction, leading to faster convergence and potentially better accuracy, especially in cases of complex patterns in data. LightGBM also uses histogram-based learning, which involves grouping continuous feature values into discrete bins to reduce the number of splits it must consider, significantly speeding up the training process and reducing memory usage. These features make LightGBM highly efficient for handling large datasets and high-dimensional features, which are common in fraud detection applications.



Neural Network (MLP Classifier)

Multilayer Perceptron (MLP) classifier, is a type of deep learning (Neural Network) model that consists of multiple layers of neurons. Each neuron applies a non-linear activation function to the weighted sum of its inputs. The network typically has an input layer, one or more hidden layers, and an output layer for binary classification (fraud or non-fraud). MLP classifiers excel at learning complex patterns in data, making them effective for detecting subtle fraud patterns. They require careful tuning of hyperparameters like the number of layers, neurons per layer, and learning rate for optimal performance.



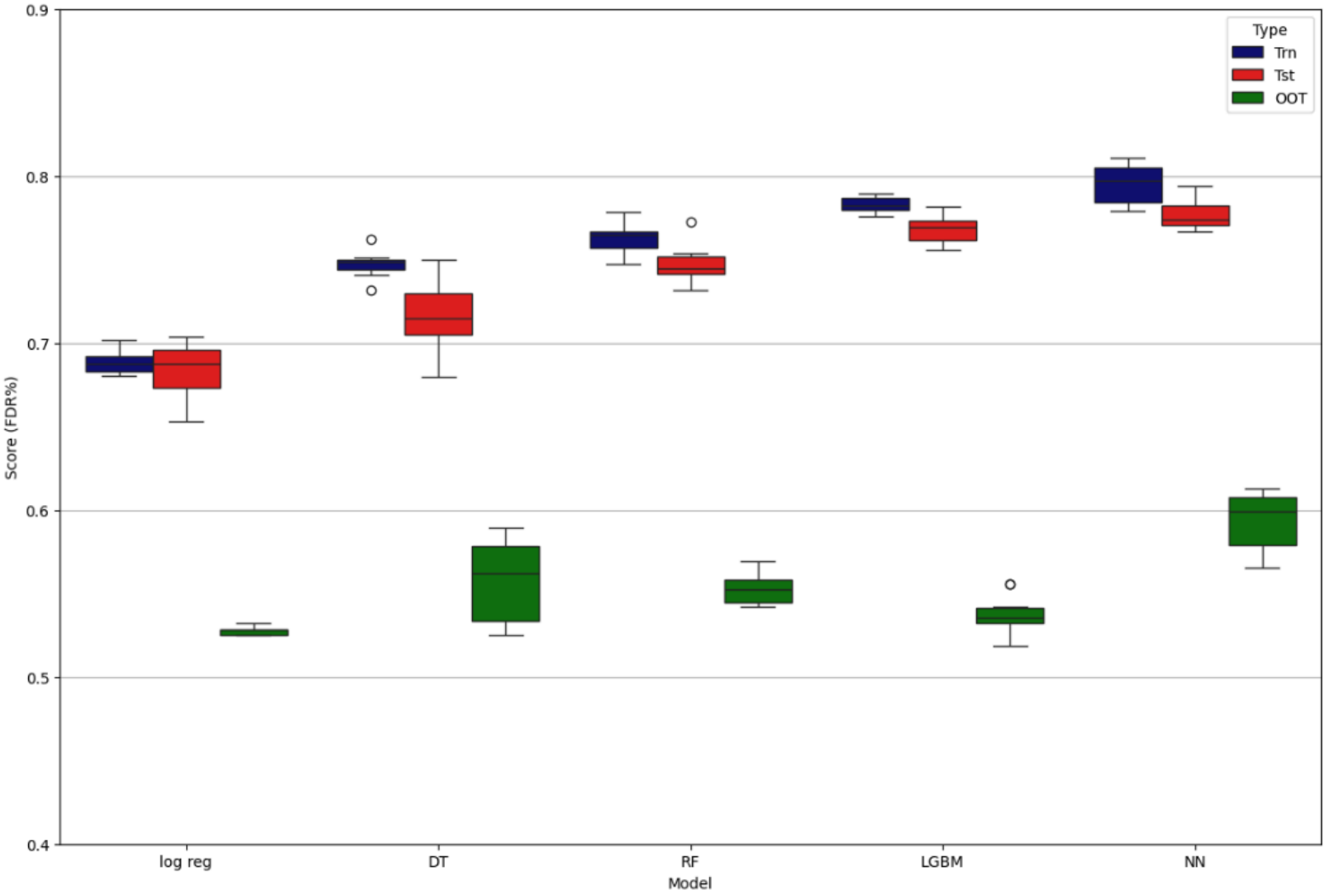
Hyperparamter Exploration

Baseline Model: Logistic Regression
Non-Linear Models: Decision Tree, Random Forest, LightGBM, Neural Net
Note: First row of each model are the default parameters, **highlighted rows** are the selected parameters

No	X
Slightly	O
Yes	V

	Hyperparameters					Train	Test	OOT	Overfitting
Logistic Regression	penalty	C	Solver						
	l2	1	lbfgs			0.685775	0.691152	0.525589	X
	l2	0.1	lbfgs			0.686332	0.688667	0.525926	X
	l2	0.01	lbfgs			0.686854	0.676043	0.526263	X
	l2	1	liblinear			0.689625	0.68244	0.528283	X
	l2	0.1	liblinear			0.686708	0.684202	0.525253	X
	l2	0.01	liblinear			0.682261	0.682389	0.525253	X
	l1	1	liblinear			0.687613	0.685926	0.526599	X
	l1	0.1	liblinear			0.688585	0.685336	0.525589	X
	l1	0.01	liblinear			0.679761	0.671423	0.508418	X
DT	criterion	max_depth	min_samples_spli	min_samples_leaf					
	gini	None	2	1		1	0.641527	0.36936	V
	gini	2	2	2		0.561862	0.542298	0.442761	X
	gini	6	2	2		0.762509	0.699204	0.479125	V
	gini	6	30	15		0.720807	0.715339	0.581818	X
	gini	7	30	15		0.73801	0.700638	0.5633	O
	gini	8	120	60		0.742152	0.714216	0.558923	X
	log_loss	2	2	2		0.668551	0.65579	0.478451	X
	log_loss	6	30	15		0.741041	0.714397	0.542088	O
	log_loss	5	30	15		0.706864	0.698208	0.539057	X
RF	criterion	n_estimators	max_depth	nin_samples_spli	min_samples_leaf				
	gini	100	None	2	1	1	0.820579	0.606734	V
	gini	50	3	2	1	0.707251	0.706279	0.561616	X
	gini	50	5	2	1	0.731591	0.722445	0.573737	X
	gini	50	5	120	60	0.719465	0.726687	0.562626	X
	gini	100	10	4	2	0.83057	0.773044	0.557912	V
	log_loss	50	5	2	1	0.746199	0.737026	0.539731	X
	log_loss	50	5	120	60	0.73716	0.727527	0.544108	X
	log_loss	50	10	2	1	0.899316	0.791088	0.569697	V
	log_loss	30	8	120	60	0.759368	0.732518	0.561953	X
LBGM	n_estimators	max_depth	num_leaves	learning_rate					
	100	None	31	0.1		0.978922	0.804936	0.554209	V
	10	2	2	0.1		0.662739	0.656106	0.458923	X
	100	5	2	0.1		0.716604	0.720488	0.485522	X
	100	5	2	0.1		0.714037	0.701381	0.490236	X
	100	3	5	0.1		0.787726	0.77119	0.541077	X
	200	3	5	0.1		0.815799	0.779127	0.551178	V
	50	3	8	0.1		0.764694	0.750241	0.526936	X
NN	activation	den_layer_siz	solver	alpha	learning_rate_init				
	relu	(1, 1)	adam	0.0001	0.001	0.426359	0.420132	0.340741	X
	relu	(15,)	adam	0.0001	0.001	0.755265	0.73702	0.517172	X
	relu	(15, 15)	adam	0.0001	0.001	0.789403	0.765124	0.564983	O
	relu	(15, 15)	adam	0.001	0.001	0.791055	0.767508	0.575084	O
	tanh	(15, 15)	adam	0.0001	0.001	0.792374	0.765212	0.574747	O
	tanh	(15, 15)	adam	0.001	0.01	0.810058	0.767456	0.578114	V
	tanh	(30, 30)	adam	0.0001	0.001	0.848838	0.767271	0.579291	V
	tanh	(30, 30)	adam	0.0005	0.001	0.830039	0.768479	0.573367	V
	tanh	(15,15)	sgd	0.0001	0.001	0.703621	0.71804	0.539057	X
	tanh	(30,30)	sgd	0.0001	0.001	0.70805	0.715152	0.543771	X
	tanh	(20,20)	adam	0.005	0.01	0.796224	0.782133	0.575084	X
	tanh	(65, 65)	sgd	0.00001	0.01	0.804667	0.776573	0.576768	O

Best Parameter Performance



The final model is selected as the Neural Network since it does not appear to suffer from the issue of overfitting yet has a higher FDR in both training and testing performance and also highest OOT performance compared to the other models.

Final Model Performance

Final Model Hyperparameters

My final model is a Multi-Layer Perceptron Classifier (a type of Neural Network) with the following architecture:

1. **2 hidden layers**, each with **20 neurons**
2. **'Adam'** optimizer
3. **Alpha (L2 penalty)** of 0.005
4. **Initial Learnig Rate** of 0.01

In Python code, it is:

```
MLPClassifier(
    hidden_layer_sizes=(20, 20),
    solver='adam',
    alpha=0.005,
    learning_rate_init=0.01
```

Where the MLPClassifier is from the Sci-Kit Learn library

Results Tables

Training

Training	#recs	#good	#bad	Fraud Rate											
	59684	58457	1227	0.02099											
bin	#recs	#good	#bad	%good	%bad	tot #recs	cum. g	cum. b	%cum. g	FDR	KS	FPR	Fraud Sav	FP Loss	Overall Savings
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	597	46	551	7.705193	92.29481	597	46	551	0.07869	44.90628	44.82759	0.083485	551000	1380	549620
2	597	274	323	45.89615	54.10385	1194	320	874	0.547411	71.23064	70.68323	0.366133	874000	9600	864400
3	597	498	99	83.41709	16.58291	1791	818	973	1.399319	79.2991	77.89978	0.840699	973000	24540	948460
4	596	554	42	92.95302	7.04698	2387	1372	1015	2.347024	82.72209	80.37506	1.351724	1015000	41160	973840
5	597	568	29	95.14238	4.857621	2984	1940	1044	3.318679	85.08557	81.7669	1.858238	1044000	58200	985800
6	597	582	15	97.48744	2.512563	3581	2522	1059	4.314282	86.30807	81.99379	2.381492	1059000	75660	983340
7	597	581	16	97.31993	2.680067	4178	3103	1075	5.308175	87.61206	82.30389	2.886512	1075000	93090	981910
8	597	582	15	97.48744	2.512563	4775	3685	1090	6.303779	88.83456	82.53078	3.380734	1090000	110550	979450
9	597	589	8	98.65997	1.340034	5372	4274	1098	7.311357	89.48655	82.1752	3.892532	1098000	128220	969780
10	596	588	8	98.65772	1.342282	5968	4862	1106	8.317225	90.13855	81.82132	4.396022	1106000	145860	960140
11	597	593	4	99.32998	0.670017	6565	5455	1110	9.331645	90.46455	81.1329	4.914414	1110000	163650	946350
12	597	588	9	98.49246	1.507538	7162	6043	1119	10.33751	91.19804	80.86053	5.400357	1119000	181290	937710
13	597	588	9	98.49246	1.507538	7759	6631	1128	11.34338	91.93154	80.58816	5.878546	1128000	198930	929070
14	597	595	2	99.66499	0.335008	8356	7226	1130	12.36122	92.09454	79.73332	6.39469	1130000	216780	913220
15	597	587	10	98.32496	1.675042	8953	7813	1140	13.36538	92.90954	79.54416	6.853509	1140000	234390	905610
16	596	590	6	98.99329	1.006711	9549	8403	1146	14.37467	93.39853	79.02386	7.332461	1146000	252090	893910
17	597	592	5	99.16248	0.837521	10146	8995	1151	15.38738	93.80603	78.41865	7.814944	1151000	269850	881150
18	597	591	6	98.99497	1.005025	10743	9586	1157	16.39838	94.29503	77.89665	8.28522	1157000	287580	869420
19	597	597	0	100	0	11340	10183	1157	17.41964	94.29503	76.87539	8.80121	1157000	305490	851510
20	597	591	6	98.99497	1.005025	11937	10774	1163	18.43064	94.78403	76.35338	9.263972	1163000	323220	839780

Testing

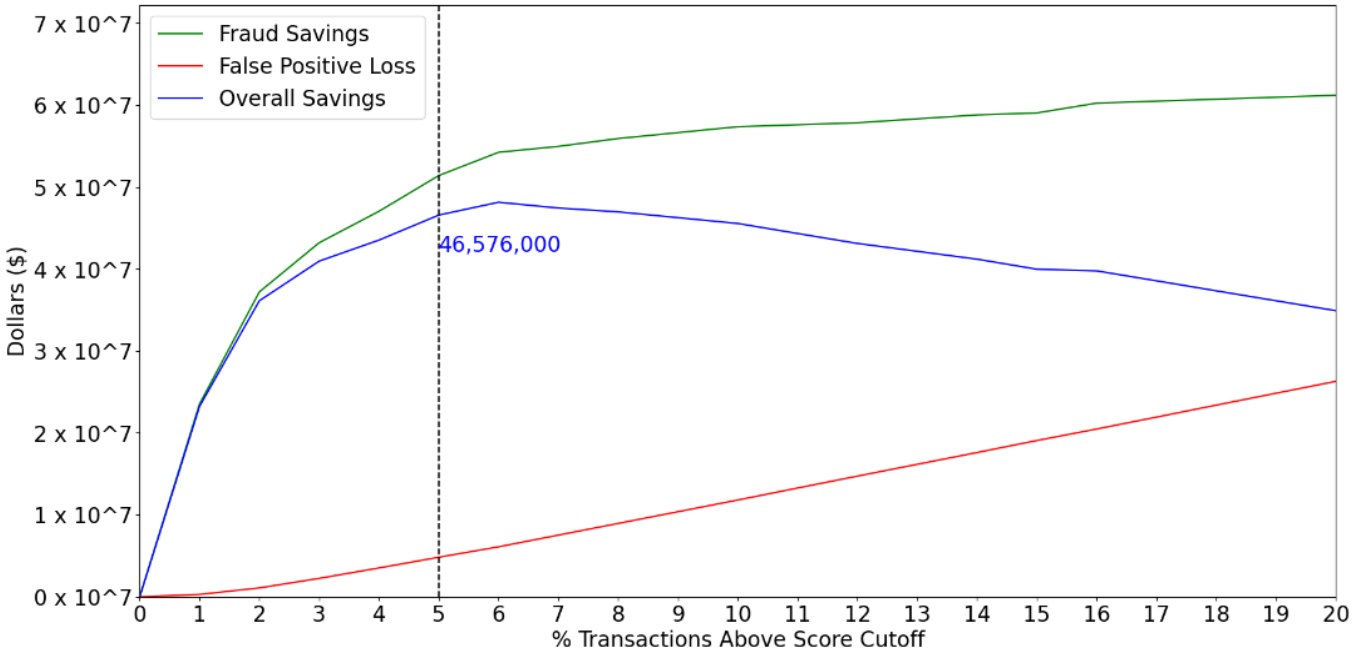
Training	#recs	#good	#bad	Fraud Rate											
	25580	25057	523	0.020872											
bin	#recs	#good	#bad	%good	%bad	tot #recs	cum. g	cum. b	%cum. g	FDR	KS	FPR	Fraud Savings	FP Loss	Overall Savings
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	256	26	230	10.15625	89.84375	256	26	230	0.103763	43.97706	43.87329	0.113043	230000	780	229220
2	256	119	137	46.48438	53.51563	512	145	367	0.578681	70.17208	69.5934	0.395095	367000	4350	362650
3	255	205	50	80.39216	19.60784	767	350	417	1.396815	79.73231	78.3355	0.839329	417000	10500	406500
4	256	238	18	92.96875	7.03125	1023	588	435	2.34665	83.174	80.82735	1.351724	435000	17640	417360
5	256	247	9	96.48438	3.515625	1279	835	444	3.332402	84.89484	81.56244	1.880631	444000	25050	418950
6	256	246	10	96.09375	3.90625	1535	1081	454	4.314164	86.80688	82.49272	2.381057	454000	32430	421570
7	256	251	5	98.04688	1.953125	1791	1332	459	5.31588	87.76291	82.44703	2.901961	459000	39960	419040
8	255	250	5	98.03922	1.960784	2046	1582	464	6.313605	88.71893	82.40532	3.409483	464000	47460	416540
9	256	250	6	97.65625	2.34375	2302	1832	470	7.31133	89.86616	82.55483	3.897872	470000	54960	415040
10	256	250	6	97.65625	2.34375	2558	2082	476	8.309055	91.01338	82.70433	4.37395	476000	62460	413540
11	256	256	0	100	0	2814	2338	476	9.330726	91.01338	81.68266	4.911765	476000	70140	405860
12	256	256	0	100	0	3070	2594	476	10.3524	91.01338	80.66099	5.44958	476000	77820	398180
13	255	252	3	98.82353	1.176471	3325	2846	479	11.3581	91.587	80.22889	5.941545	479000	85380	393620
14	256	256	0	100	0	3581	3102	479	12.37977	91.587	79.20722	6.475992	479000	93060	385940
15	256	253	3	98.82813	1.171875	3837	3355	482	13.38947	92.16061	78.77114	6.960581	482000	100650	381350
16	256	253	3	98.82813	1.171875	4093	3608	485	14.39917	92.73423	78.33506	7.439175	485000	108240	376760
17	256	255	1	99.60938	0.390625	4349	3863	486	15.41685	92.92543	77.50858	7.94856	486000	115890	370110
18	255	253	2	99.21569	0.784314	4604	4116	488	16.42655	93.30784	76.88129	8.434426	488000	123480	364520
19	256	256	0	100	0	4860	4372	488	17.44822	93.30784	75.85962	8.959016	488000	131160	356840
20	256	253	3	98.82813	1.171875	5116	4625	491	18.45792	93.88145	75.42354	9.419552	491000	138750	352250

OOT

Training	#recs	#good	#bad	Fraud Rate								
	12232	11935	297	0.024885								
bin	#recs	#good	#bad	%good	%bad	tot #recs	cum. g	cum. b	%cum. g	FDR	KS	FPR
0	0	0	0	0	0	0	0	0	0	0	0	0
1	122	26	96	21.31148	78.68852	122	26	96	0.217847	32.32323	32.10539	0.270833
2	123	64	59	52.03252	47.96748	245	90	155	0.754085	52.18855	51.43447	0.580645
3	122	88	34	72.13115	27.86885	367	178	189	1.491412	63.63636	62.14495	0.941799
4	122	107	15	87.70492	12.29508	489	285	204	2.387935	68.68687	66.29893	1.397059
5	123	117	6	95.12195	4.878049	612	402	210	3.368245	70.70707	67.33883	1.914286
6	122	115	7	94.2623	5.737705	734	517	217	4.331797	73.06397	68.73218	2.382488
7	122	120	2	98.36066	1.639344	856	637	219	5.337243	73.73737	68.40013	2.908676
8	123	120	3	97.56098	2.439024	979	757	222	6.34269	74.74747	68.40479	3.40991
9	122	113	9	92.62295	7.377049	1101	870	231	7.289485	77.77778	70.48829	3.766234
10	122	119	3	97.54098	2.459016	1223	989	234	8.286552	78.78788	70.50133	4.226496
11	123	120	3	97.56098	2.439024	1346	1109	237	9.291998	79.79798	70.50598	4.679325
12	122	121	1	99.18033	0.819672	1468	1230	238	10.30582	80.13468	69.82886	5.168067
13	122	119	3	97.54098	2.459016	1590	1349	241	11.30289	81.14478	69.84189	5.59751
14	122	118	4	96.72131	3.278689	1712	1467	245	12.29158	82.49158	70.2	5.987755
15	123	120	3	97.56098	2.439024	1835	1587	248	13.29703	83.50168	70.20466	6.399194
16	122	120	2	98.36066	1.639344	1957	1707	250	14.30247	84.17508	69.87261	6.828
17	122	121	1	99.18033	0.819672	2079	1828	251	15.3163	84.51178	69.19549	7.282869
18	123	122	1	99.18699	0.813008	2202	1950	252	16.3385	84.84848	68.50998	7.738095
19	122	121	1	99.18033	0.819672	2324	2071	253	17.35233	85.18519	67.83286	8.185771
20	122	119	3	97.54098	2.459016	2446	2190	256	18.34939	86.19529	67.84589	8.554688

Financial Curves and Recommended Cutoff

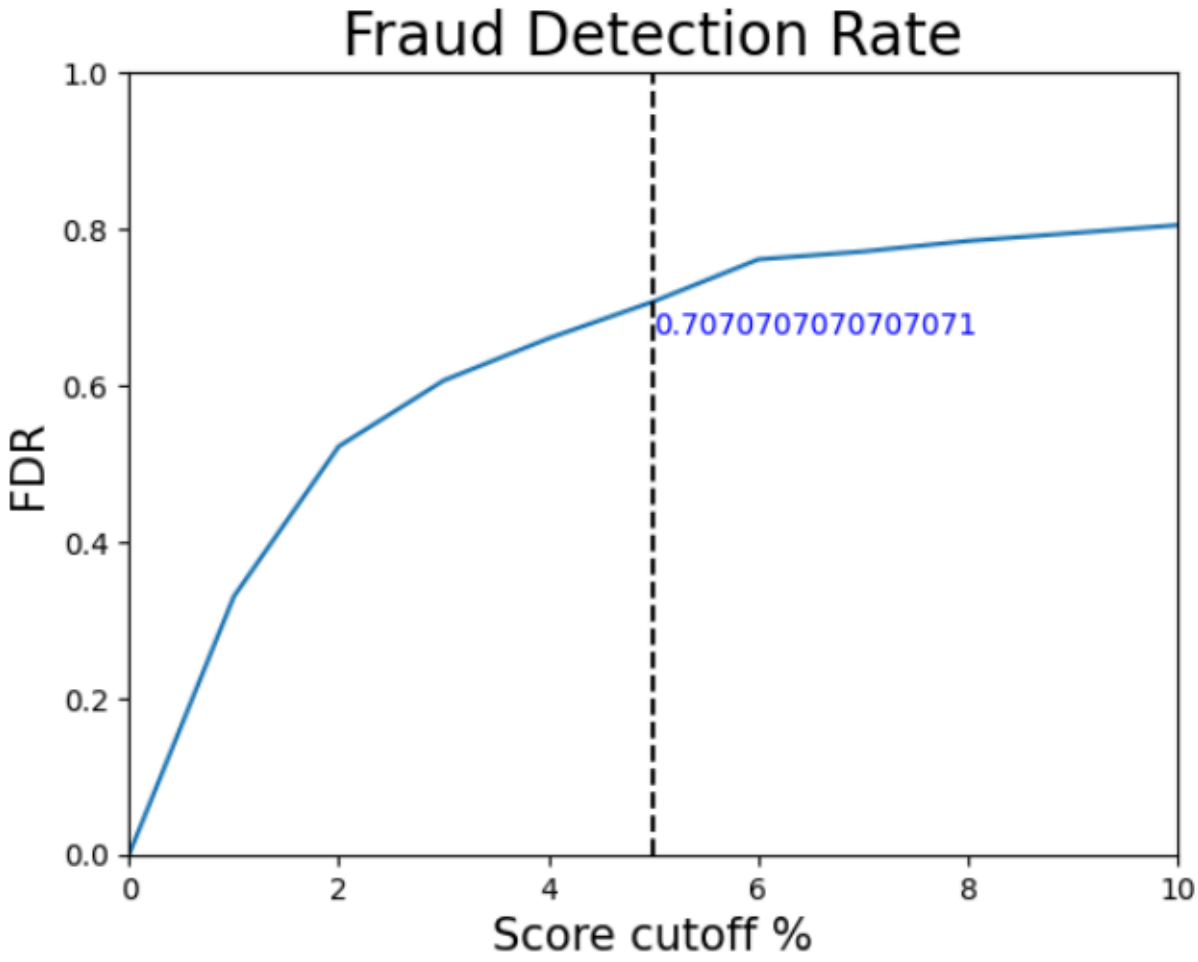
Cutoff Plot



Cutoff Recommendation

To put our model into use, I generated the above plot to provide a recommendation for the cutoff point (i.e. the percentage of transactions above the score of how likely a transaction is fraudulent). Assuming a $400 gain for every fraudulent transaction caught and a $20 loss for every false positive (non-fraudulent labeled as fraudulent) and a sample of 100,000 records from a portfolio of 10 million transations/year, I plotted the Fraud Savings (green curve), Loss from False Positives (red curve) and Overall Savings (blue curve). I then analyzed the savings and determined that a **cutoff point at 5%** is the best as it deny as few applicants as possible while still maintaining a high overall savings. Furthermore, In the Fraud Detection Rate plot shown below, the **fraud detection rate at the 5% cutoff stands at approximately 71%**. This point lies just beyond the steep initial increase, indicating a substantial improvement in detection efficacy early in the range. Although it does not reach the maximum rate, the difference is relatively minor, suggesting that the 5% cutoff offers a highly effective balance between performance and practicality. Therefore, my recommendation is to have a 5% cutoff and the **expected annual savings from the 5% cutoff is 46.576 Million**.

Fraud Detection Rate Plot



Summary

In this report, a comprehensitve analysize of credit card transaction fraud was performed. Key highlights of my approach include:

1. **Data Exploration**: Analyzed each of the 10 data fields in the dataset, providing histograms or tables to examine their distributions.
2. **Data Cleaning**: Cleaned the dataset of 96,753 records, which included removing an outlier and focusing only on records with a transaction type of “P” (purchase). Also handled missing values in the "Merchnum," "Merch state," and "Merch zip" fields through data imputation techniques.
3. **Variable Creation**: Developed a wide range of candidate variables for analysis, including amount variables, frequency variables, days-since variables, velocity change variables, Benford’s Law variables, and a day-of-the-week risk table variable.
4. **Feature Selection**: Employed both filter and wrapper methods for feature selection. Used Kolmogorov-Smirnov (KS) and Fraud Detection Rate (FDR) at 3% for filtering and averaged results from three tree-based methods as the wrapper method to determine the most effective variables.
5. **Model Development**: Segmented the data into training, testing, and out-of-time (OOT) sections to facilitate the development and analysis of machine learning models aimed at predicting fraudulent transactions. The models used were Logistic Regression, Random Forest, Decision Tree, Boosted Tree, and Neural Network.

The best-performing model was a Neural Network with two hidden layers (20 neurons each), using the 'Adam' optimizer, an L2 penalty (alpha) of 0.005, and an initial learning rate of 0.01. This model achieved an FDR of 70.7% at the 3% level on the out-of-time dataset.

The robustness of our Neural Network model was validated through out-of-time testing at an FDR of 3%, confirming its effectiveness against evolving fraudulent patterns. A recommended score cutoff of 5% optimizes the balance between detecting fraud and limiting the amount of denied applications. If deployed, this model is expected to generate approximately $46.5 million in annual savings due to its high fraud detection capability.

To further enhance the model's accuracy and adaptability, I could explore additional ensemble techniques, deeper neural networks, and the incorporation of more real-time transaction data into the model training process. These efforts would likely improve the model’s responsiveness to new fraudulent tactics, thereby increasing its preventive potential. Additionally, considering that the current dataset comprises transaction data likely from a government agency in Tennessee in 2010, incorporating more recent and diverse data sources could significantly refine the model’s efficacy. Updating the dataset would not only provide a broader view of contemporary fraud trends but also enhance the model's generalizability across different types and periods of transactional activities.

Appendix: Data Quality Report

1. Data Description

The dataset is **Card Transaction** data, which contains **real card transaction records** from a US government organization, likely from somewhere in Tennessee. The data came from an author's website. The author, Mark Nigrini, is an expert in applying Benford's law for forensic accounting. There are **10 fields** and contains **97852 records**. Note: The fraud labels were made up based on the Professor's professional expertise on how credit card fraud behaves.

2. Summary Tables

Numeric Fields Table

Field Name	Field Type	# Records w/ Values	% Populated	# Zeros	Min	Max	Mean	Standard Deviation	Most Common
Date	numeric	97,852	100%	0	2010-01-01	2010-12-31	2010-06-25	98.92 days	2010-02-28
Amount	numeric	97,852	100%	0	0.01	3,102,045.53	425.466	9,949.8	3.62

Categorical Fields Table

Field Name	Field Type	# Records w/ Values	% Populated	# Zeroes	# Unique Values	Most Common
Recnum	Categorical	97,852	100%	0	97,852	1
Cardnum	Categorical	97,852	100%	0	1,645	5142148452
Merchnum	Categorical	94,455	96.5%	0	13,091	930090121224
Merch description	Categorical	97,852	100%	0	13,126	GSA-FSS-ADV
Merch state	Categorical	96,649	98.8%	0	227	TN
Merch zip	Categorical	93,149	95.2%	0	4,567	38,818.0
Transtype	Categorical	97,852	100%	0	4	P
Fraud	Categorical	97,852	100%	95,805	2	0

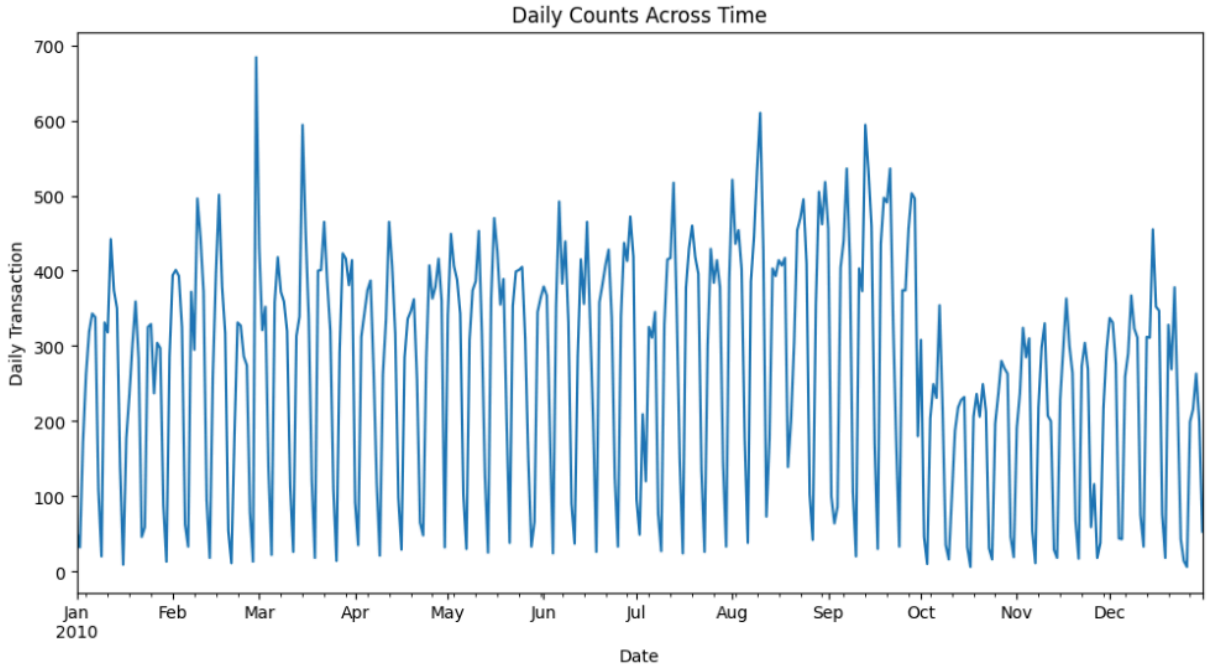
3. Visualization of Each Field

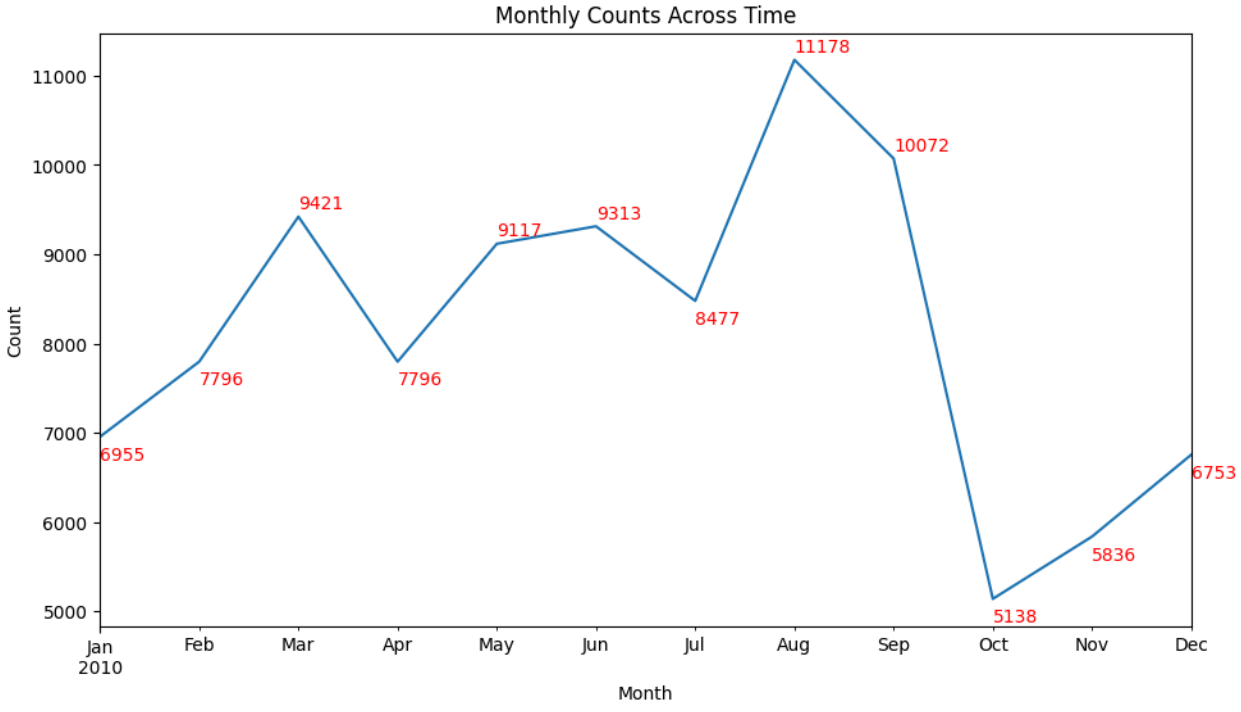
a. Field Name: Recnum

Description**:** Ordinal unique positive integer for each transaction record, from 1 to 97582

b. Field Name: Date

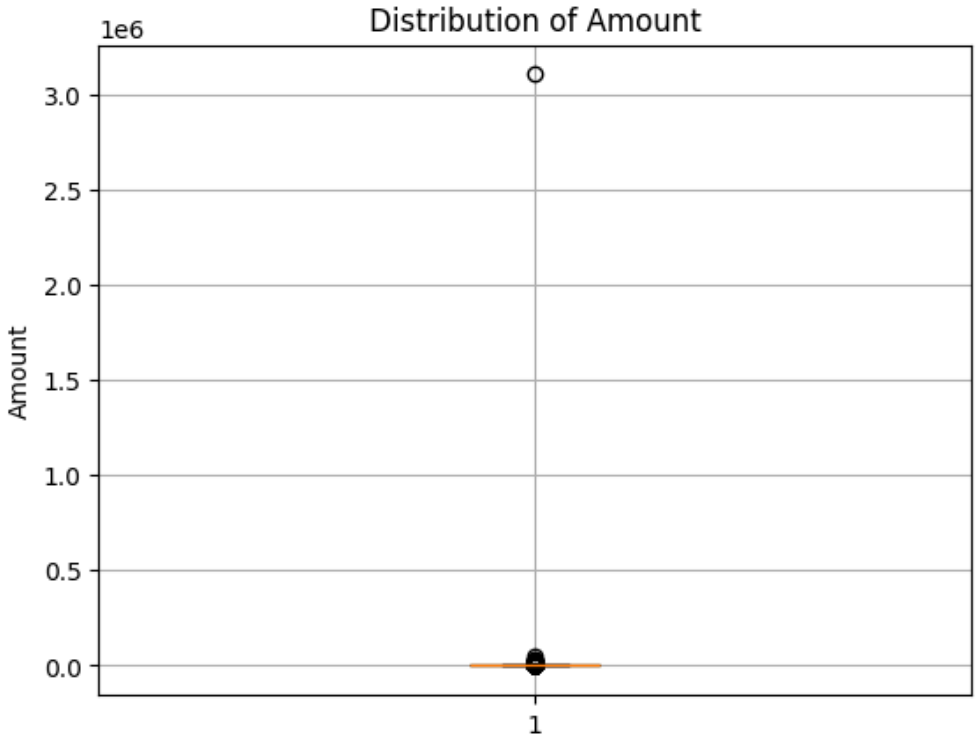
Description: Transaction date. The first distribution shows the number of daily transactions across time. The second distribution shows the number of monthly transactions across time.

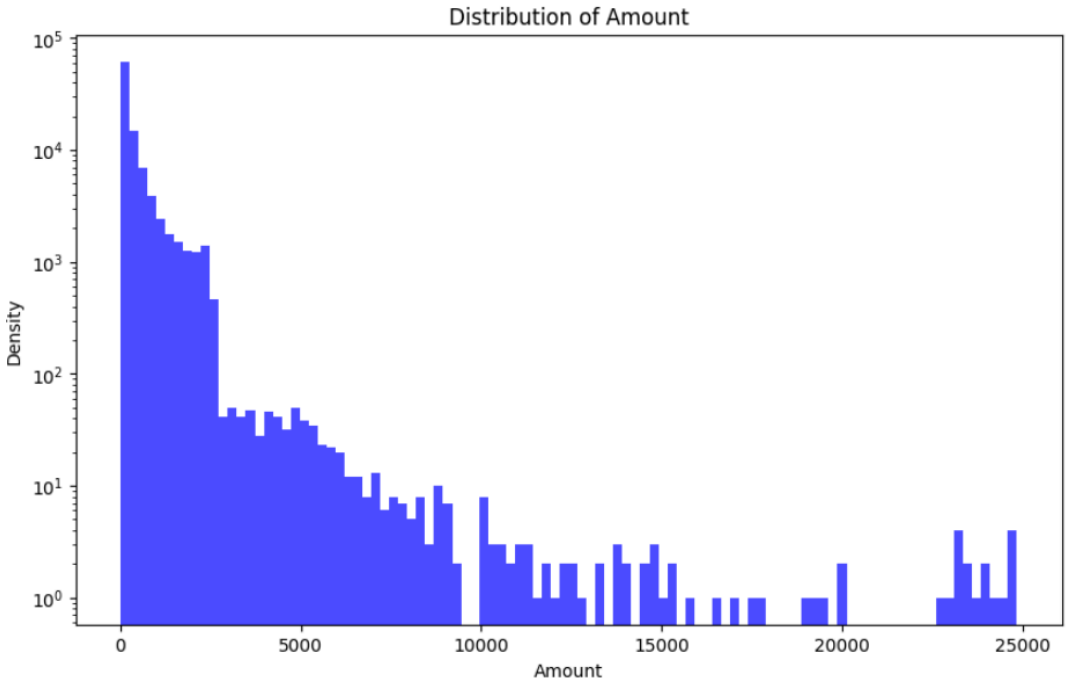




c. Field Name: Amount

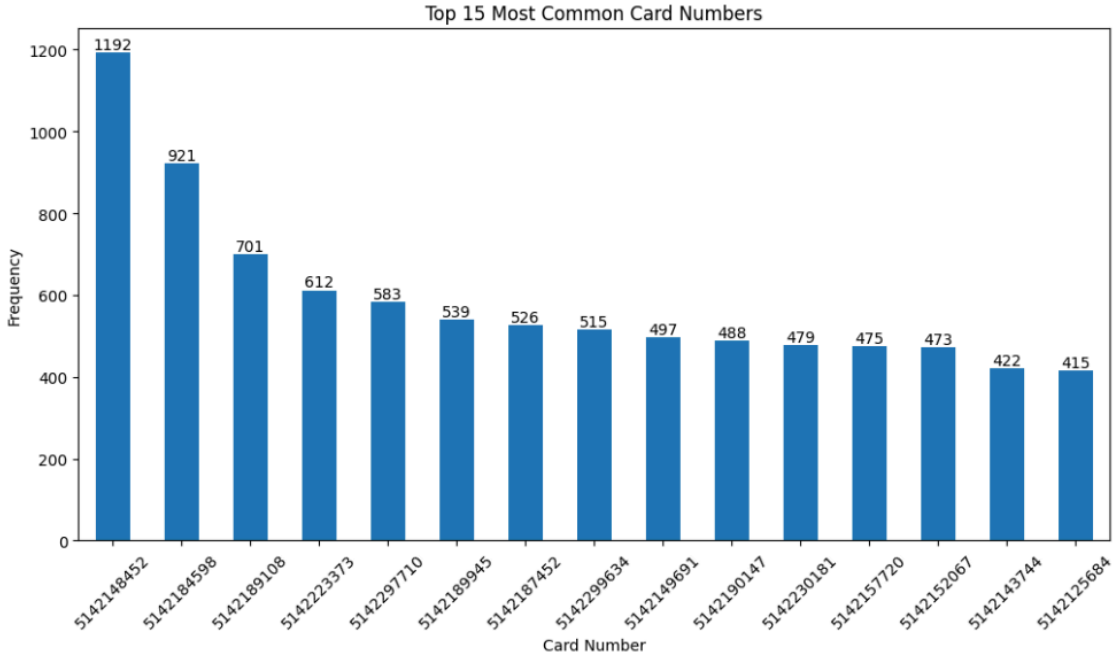
Description: Transaction amount. The first distribution shows a boxplot of all the transactions amount. As seen in the distribution, there is a **clear outlier at 3102045.53**. The second distribution shows a histogram of all transaction from 0 to 99th percentile. The third distributio shows a histogram of 99th percentile+ excluding the outlier listed above.





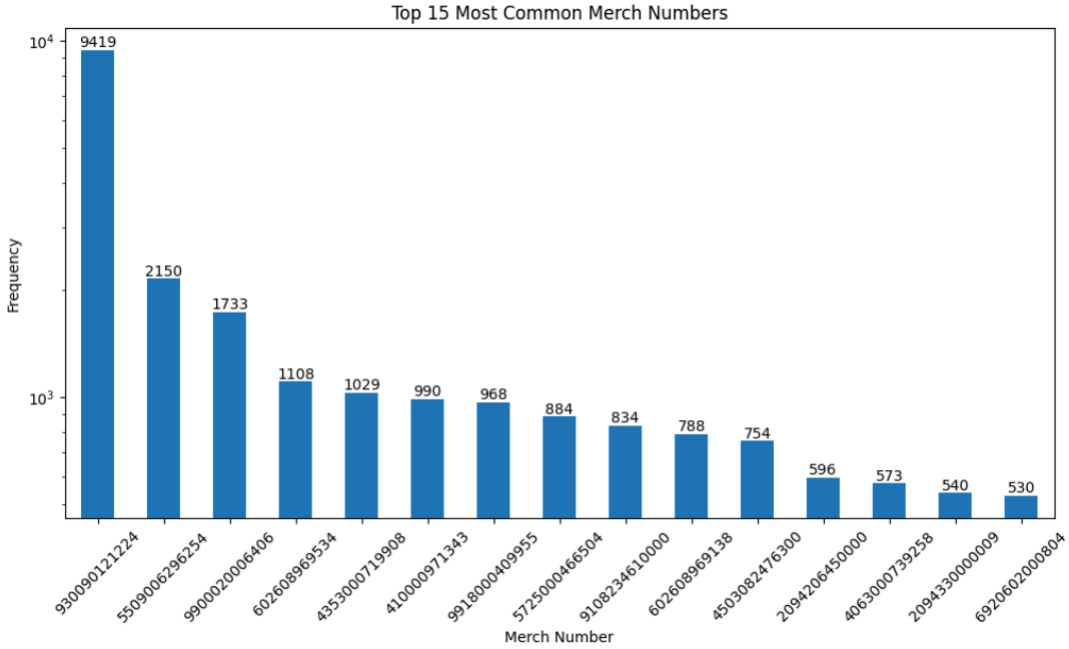
d. Field Name: Cardnum

Description: Card number used in the transaction. The first distribution shows the top 15 field values of the card number. The **most common is 5142148452**, with a total count of **1192**.



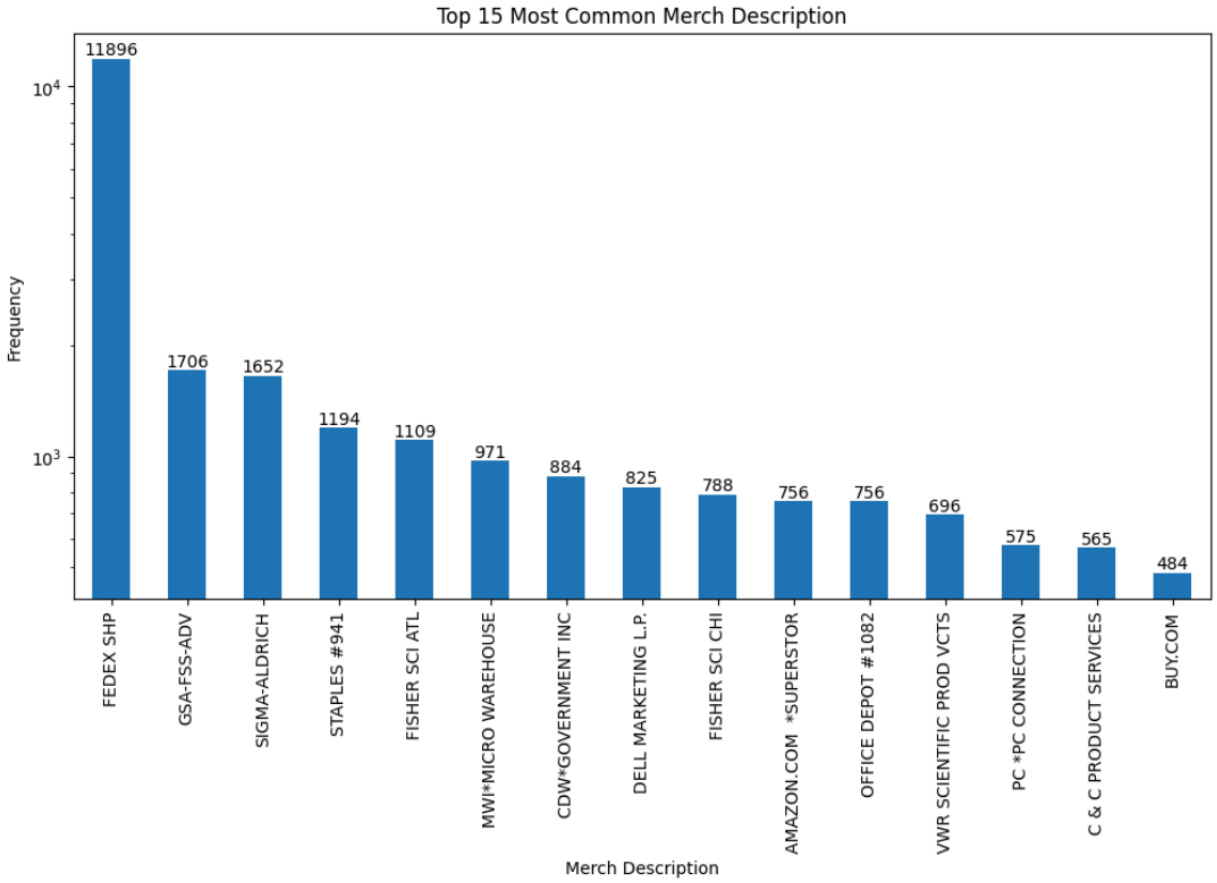
e. Field Name: Merchnum

Description: ID of the merchandise bought in the transaction. The first distribution shows the top 15 field values of the merch number. The **most common is 930090121224**, with a total count of **9419**.



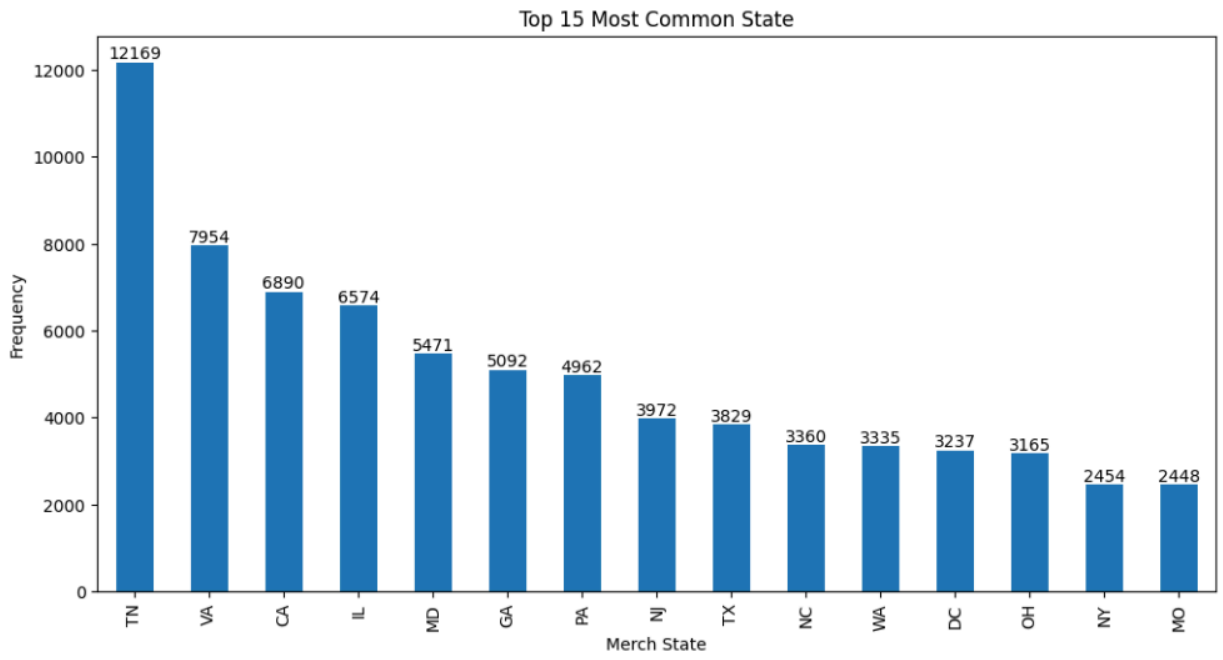
f. Field Name: Merch description

Description: Description of the merchandise bought in the transaction. The first distribution shows the top 15 field values of the merch number. The **most common is FEDEX SHP**, with a total count of **11896**.
Note: FEDEX SHP has 4 unique Merchnum, likely due to shipping location differences. The 4 Merchnum was {'5509006296254', '5569000000637', '930009906224', '930090121224'}



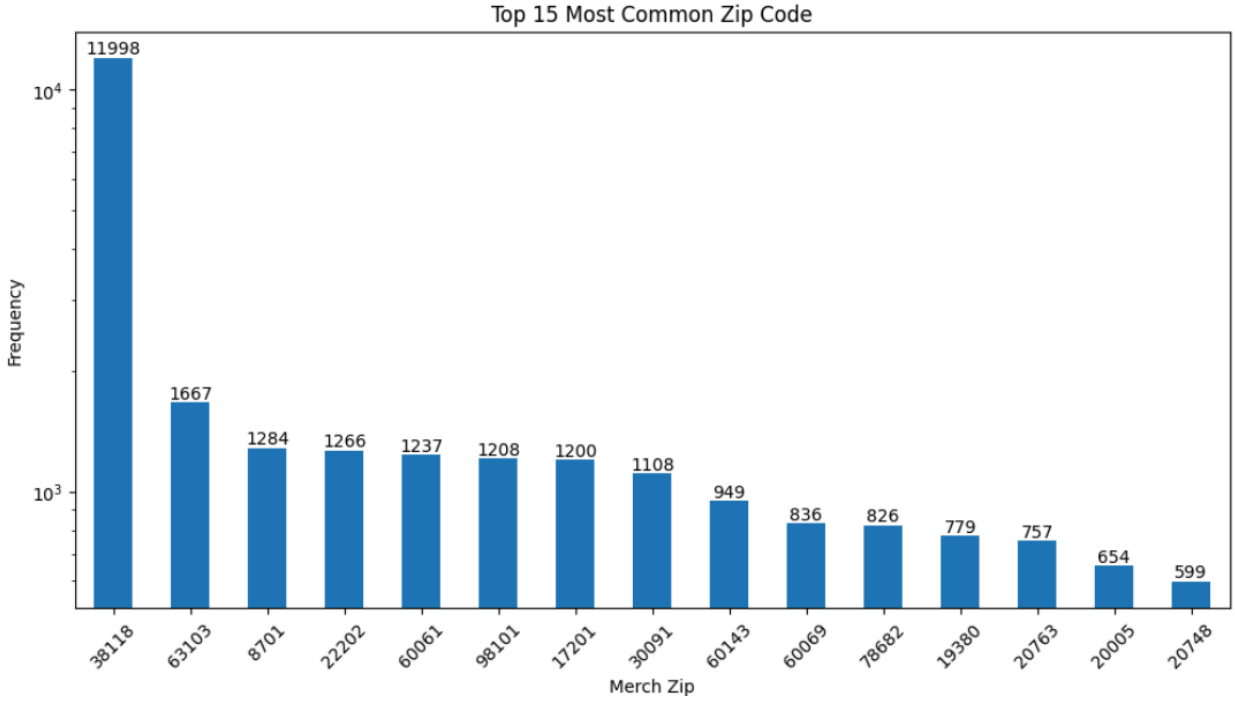
g. Field Name: Merch state

Description: State of the merchandise bought in the transaction. The first distribution shows the top 15 field values of the merch state. The **most common is TN** (Tennessee), with a total count of **12169**.



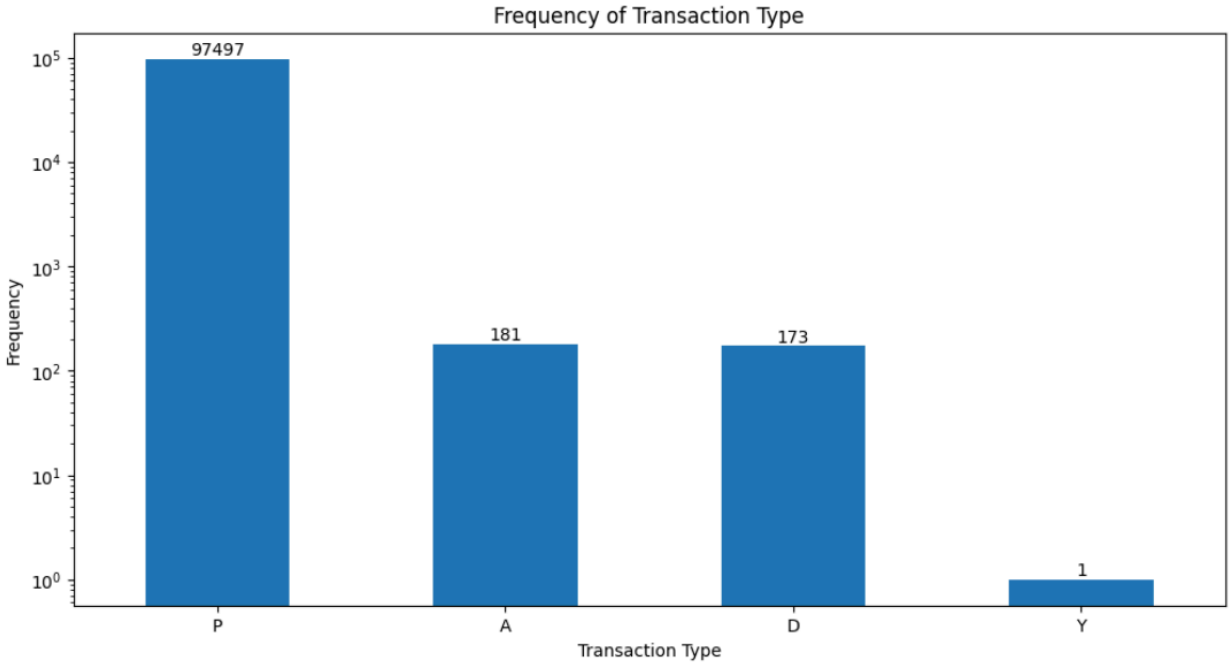
h. Field Name: Merch zip

Description: Zip code of the state of the merchandise bought in the transaction. The first distribution shows the top 15 field values of the merch state. The **most common is 38818** (Zip code for Memphis, Tennessee), with a total count of **11998**.



i. Field Name: Transtype

Description: Type of transction. The first distribution shows the distribution of the 4 transaction type recorded. The **most common is P,** with a total count of **97497**.



j. Field Name: Fraud

Description: Fraud identification label. Fraud = 0 (not fraudulent), Fraud = 1 (Fraud identified). The total count of **Fraud = 0 is 95805** and total count of **Fraud = 1 is 2047**. Roughly **2%** of all transactions were labeled as frFraudulent.