Credit Card Transactions Analysis and Fraud Detection

Table of Contents

I. Executive Summary

The goal of this project is to identify patterns and behaviors associated with fraudulent credit card transactions, develop a fraud detection algorithm, and maximize overall savings. Each real-world transaction data was analyzed and processed to uncover insights into potential fraudulent activity. Several advanced analytical models were built and tested to ensure high accuracy in identifying fraudulent transactions. In the final stage, a strategic score cutoff was applied to balance the effectiveness of fraud detection, the cost of investigating flagged transactions, and minimizing losses from incorrectly flagging legitimate transactions.

At a score cutoff of 6%, the fraud detection rate (FDR) for out-of-time (OOT) data is 60%, indicating that the model captures 60% of the fraudulent transactions within the top 6% of high-risk transactions. This approach is projected to save the company approximately $47.18 million annually by preventing fraudulent transactions while maintaining a manageable level of false positives, ensuring both revenue protection and customer trust.

II. Dataset Description

1. Purpose

This section conducts a basic exploratory analysis on a dataset of real credit card transactions from a U.S. government organization in Tennessee. The data includes transactions recorded from January to December 2010, comprising 97,852 records, with 2,000 manually generated fraud cases for analysis. The variables encompass transaction amounts, dates, merchant information, cardholder IDs, and fraud labels.

By examining the statistical distributions of these variables (e.g., mean, median, standard deviation), it is possible to identify anomalies, trends, and relationships that may indicate potential fraud.

This step is crucial for understanding the structure of the data and uncovering hidden patterns and behaviors associated with fraudulent activity, which is essential for subsequent steps like data cleaning, variable creation, and feature selection.

2. Field Summary Tables

1) Numeric fields

Field Name	Field Type	# Records Have Values	% Populated	# Zeros	Min	Max	Mean	Standard Deviation	Most Common
Amount	Numeric	97,852	100.0%	0	0.01	3,102,045	425	9,949	3.62

- The "Amount" field represents the transaction amount for each credit card record. Values range from as low as $0.01 to as high as $3,102,045.53, with a mean of $425.47. The distribution is highly skewed, as indicated by the large standard deviation (9,949.8), suggesting there are a few very high-value transactions. Most transactions are relatively small, with the most common amount being $3.62. No records have a value of zero.

2) Categorical fields

Field Name	Field Type	# Records Have Values	% Populated	# Zeros	# Unique Values	Most Common
Date	Categorical	97,852	100.%	0	365	2/28/10
Merchnum	Categorical	94,455	96.5%	0	13,091	930090121224
Merch description	Categorical	97,852	100.0%	0	13,126	GSA-FSS-ADV
Merch state	Categorical	96,649	98.8%	0	227	TN
Transtype	Categorical	97,852	100.0%	0	4	P
Recnum	Categorical	97,852	100.0%	0	97,852	1
Fraud	Categorical	97,852	100.0%	95,805	2	0
Cardnum	Categorical	97,852	100.0%	0	1,645	5142148452
Merch zip	Categorical	93,149	95.2%	0	4,567	38118

- The "Date" field represents the date of each transaction and is fully populated. It contains 365 unique values, covering all days in the year 2010. The most common date is February 28, 2010.

- The "Merchnum" field represents the merchant number associated with the transaction and is 96.5% populated. It contains 13,091 unique values, indicating a large variety of merchants. The most frequent merchant number is 930090121224.

- The "Merch Description" field describes the merchant and is fully populated with 13,126 unique descriptions. The most common value is "GSA-FSS-ADV," indicating government-related transactions.

- The "Merch State" field represents the U.S. state where the merchant is located. The field is 98.8% populated and contains 227 unique state codes. The most common state is Tennessee (TN), which makes sense given the dataset's origin from a government organization in Tennessee.

- The "Transtype" field indicates the type of transaction. It is fully populated with only 4 unique values, and the most common type is represented by the letter "P."

- The "Recnum" field contains unique record numbers for each transaction. Since each record is unique, the field is fully populated with no duplicates, and each value is distinct.

- The "Fraud" field identifies whether the transaction is fraudulent (1) or not (0). This field is fully populated, with the vast majority of records (95,805) being non-fraudulent (0). Only 2,000 records are labeled as fraudulent, which were manually generated for the analysis.

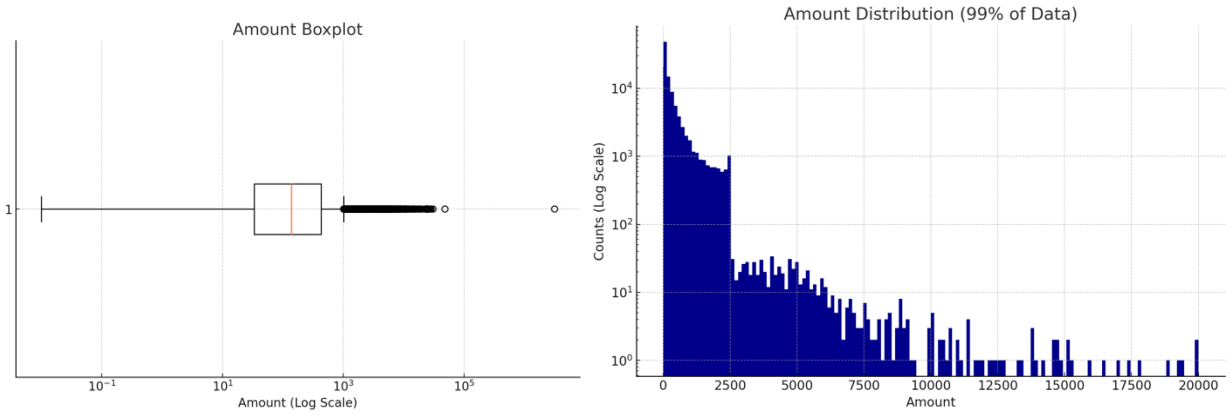
- The "Cardnum" field represents the credit card number associated with each transaction. It is fully populated, containing 1,645 unique credit card numbers. The most common card number is 5142148452, which may have made frequent transactions.

- The "Merch Zip" field represents the zip code of the merchant. It is 95.2% populated and contains 4,567 unique values. The most common zip code is 38118.0, which may correspond to a location with a high concentration of transactions.

3. Visualizations

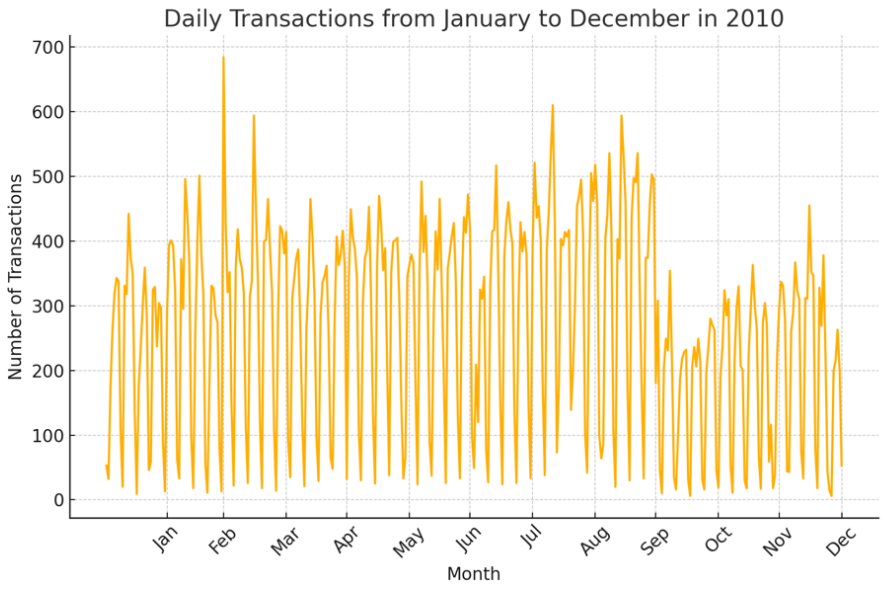
1) Amount

- The boxplot reveals that most transaction amounts are small, with a significant number of outliers representing much larger values. These extreme outliers justified their exclusion, leading to the removal of the top 1% of values in the distribution visualization, which provides a clearer focus on the primary range of data without distortion. The distribution shows that the majority of transactions are under $2,500, while larger transactions are increasingly rare.



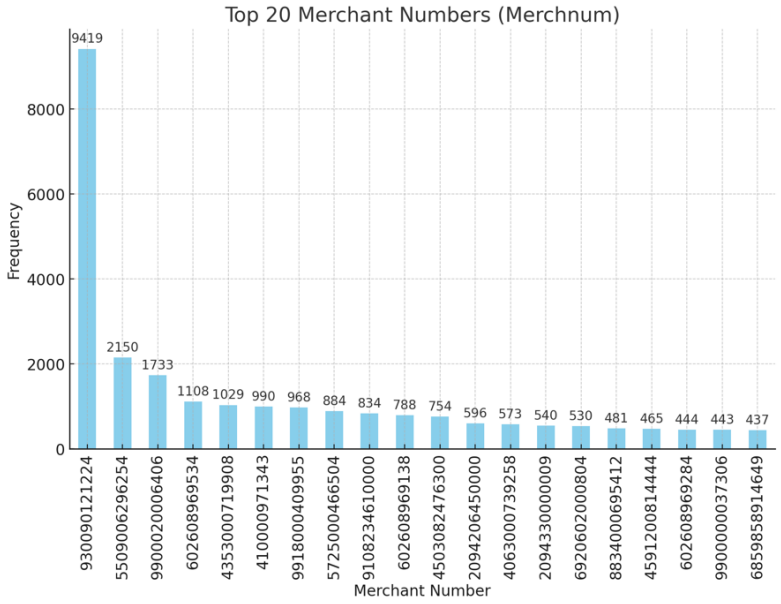
2) Date

- There's a noticeable increase in transactions in February, likely due to tax refunds, performance bonuses, and post-holiday sales. July and August also see higher activity, likely due to holiday seasons. From October 1, the start of the U.S. government fiscal year, transaction volumes stabilize but remain lower. However, volumes spike again between November and December, likely due to year-end spirit and holiday shopping.



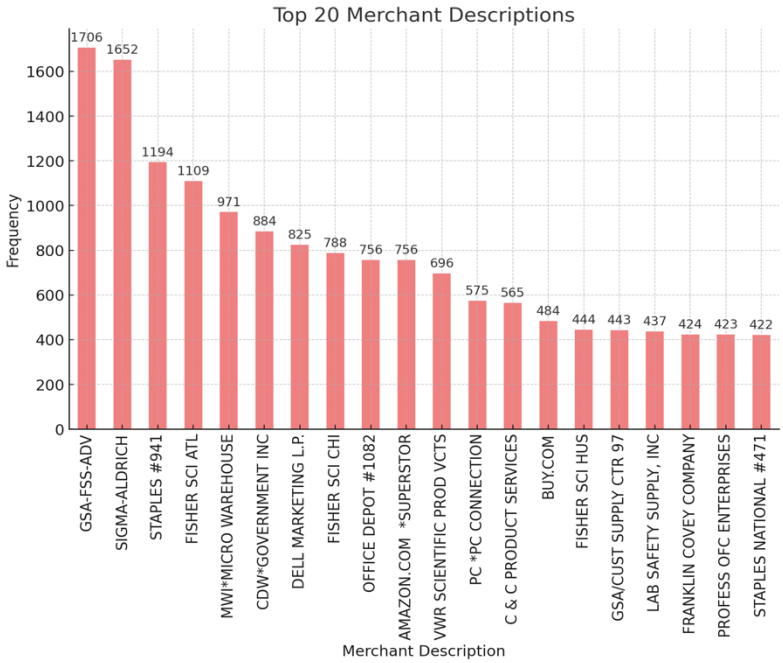
3) Merchant Number

- The highest frequency is observed for merchant number 93009012124, with over 8,000 occurrences, while the other merchant numbers have lower frequencies, ranging from around 500 to 2,500. This visualization highlights the significant dominance of a single merchant within the transaction dataset.



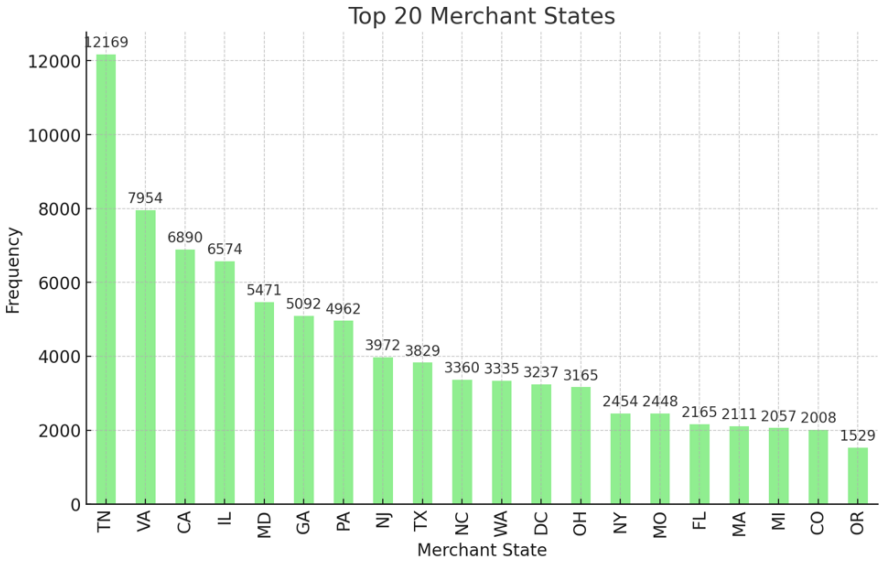
4) Merchant Description

- The description GSA-FSS-ADV has the highest frequency with 1,706 occurrences, followed closely by SIGMA-ALDRICH with 1,652. The remaining merchant descriptions range from 422 to 1,194 occurrences.



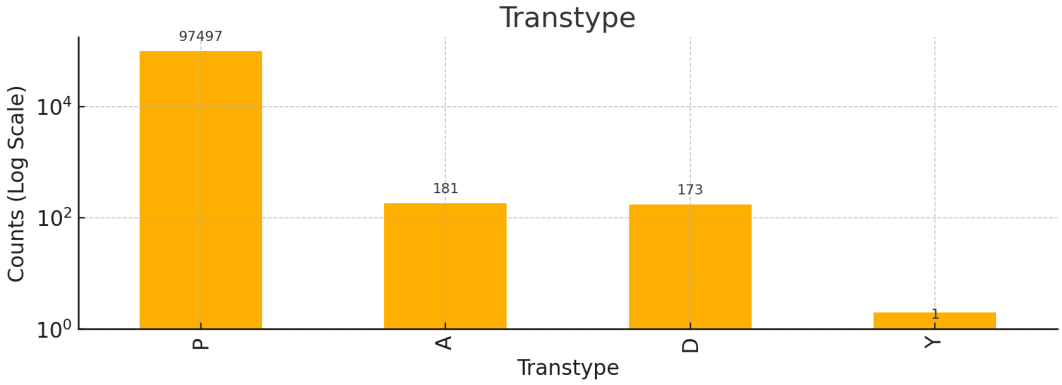
5) Merchant State

- Tennessee (TN) has the highest frequency with 12,169 occurrences, followed by Virginia (VA) with 7,954 occurrences, and California (CA) with 6,890. The remaining top states range between 1,529 and 6,574 occurrences.



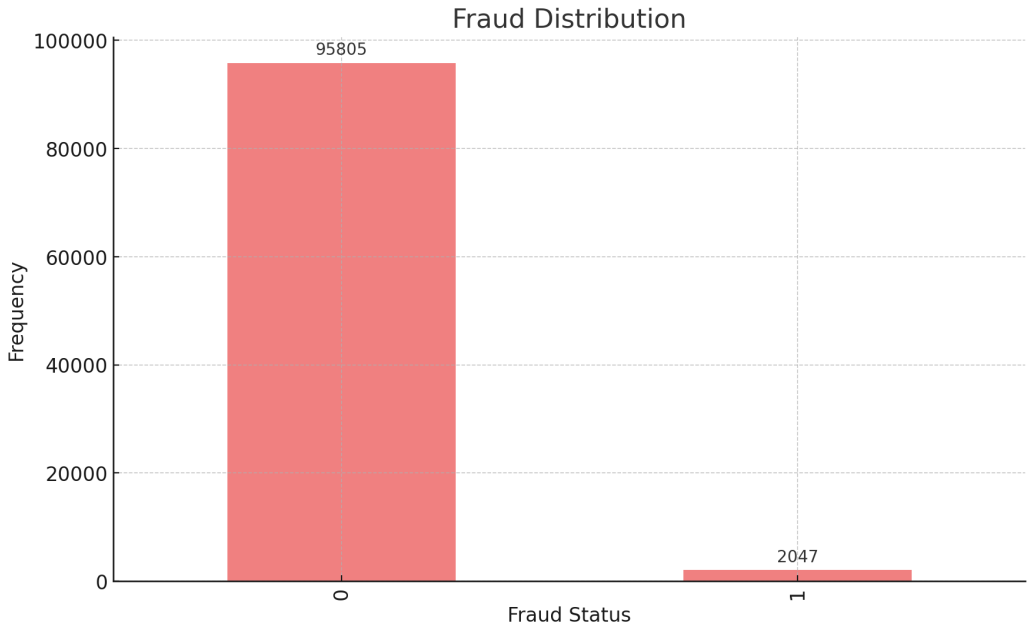
6) Transaction Type

- The most frequent transaction type is P (Purchase) with 97,497 occurrences, followed by A (Authorization) with 181, D (Declined) with 173, and Y (System Error) with 1 occurrence.



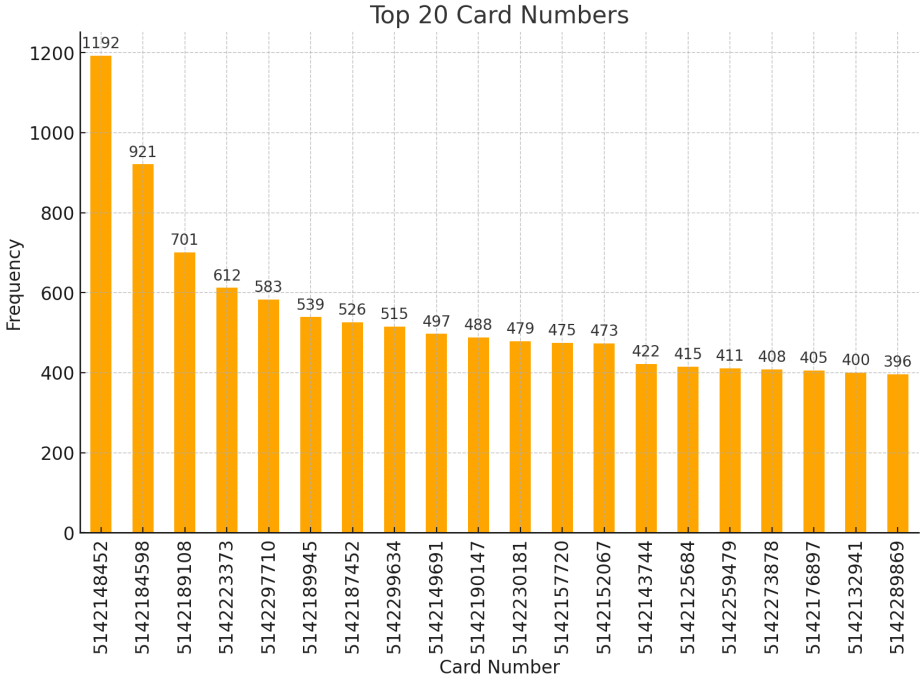
7) Fraud Distribution

- The majority of transactions (95,805) are labeled as non-fraudulent (Fraud = 0), while a much smaller portion (2,047) are labeled as fraudulent (Fraud = 1). This indicates that fraudulent transactions make up a small percentage of the dataset.



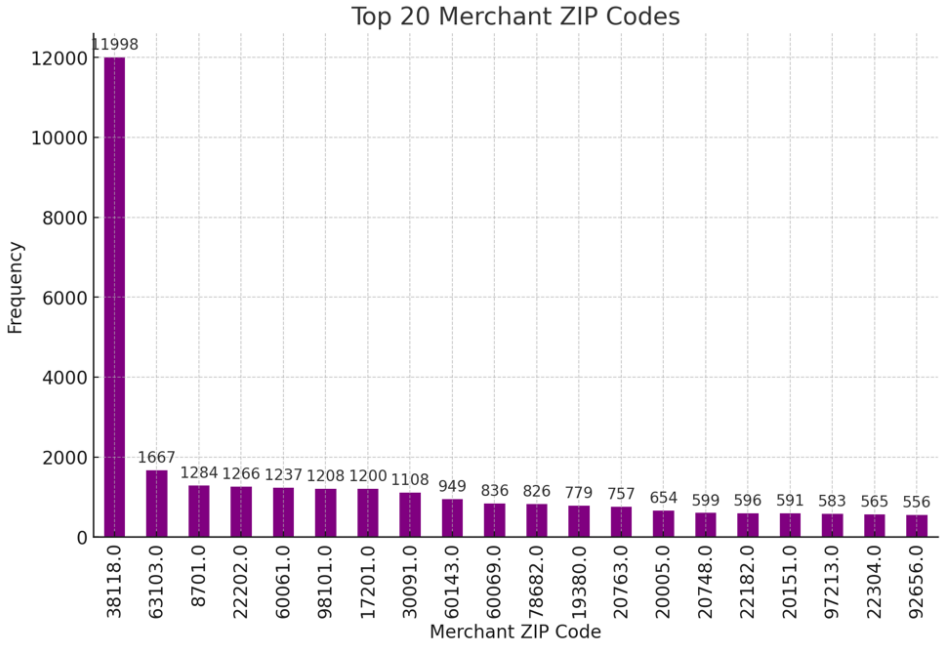
8) Card Numbers

- The most frequent card number appears over 1,300 times, with the other top card numbers ranging between approximately 650 and 1,200 occurrences.



9) Merchant Zipcodes

- The highest frequency ZIP code appears over 2,300 times, while the remaining top ZIP codes range between approximately 500 and 2,000 occurrences.



III. Data Cleaning

1. Purpose

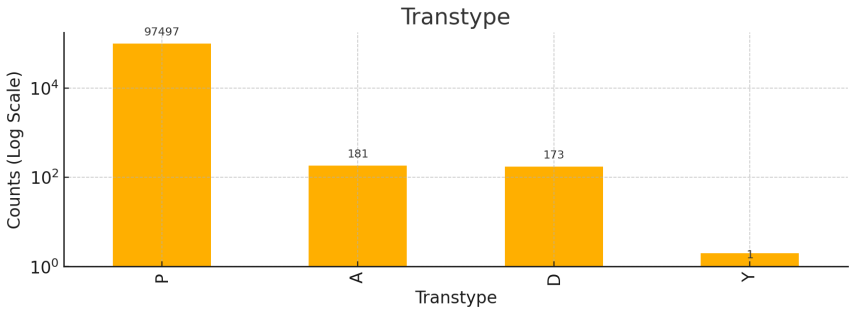
In this section, data cleaning is performed by excluding irrelevant data, addressing outliers, and handling approximately 8,600 missing values. For exclusion, records not relevant to fraud detection, such as non-purchase transaction types, are removed. Outlier treatment involves identifying and removing extreme outliers that could distort the model's results. Different imputation methods are applied based on relationships between related fields such as Merchant Number, Merchant State, and Merchant Zip. These steps ensure that the model is not misled by noisy data and can focus on valid and consistent patterns, ultimately enhancing its ability to detect fraudulent transactions.

2. Exclusion

In data cleaning, exclusions refer to data records that are irrelevant or inappropriate for the model's learning process. These records should be removed to ensure the model focuses on valid and relevant data for accurate predictions.

In the credit card transactions dataset, there are four transaction types: P (97,497), A (181), D (173), and Y (1). P represents 'Purchase,' the most common type. After further review, A is assumed to be 'Authorization,' D is 'Declined,' and Y is 'System Error.'

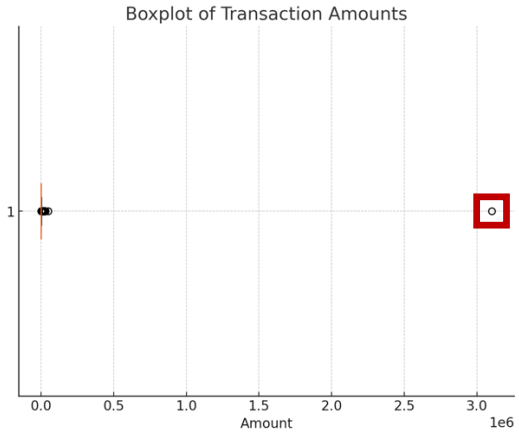
For fraud detection, only Purchase (P) transactions were included. The other types were excluded due to their different nature from regular purchases and lower relevance to the model.



3. Outlier

Outliers can significantly affect machine learning algorithms like Neural Networks (NN), Support Vector Machines (SVM), and clustering models, potentially leading to inaccurate results.

As seen in the boxplot of transaction amounts, there is one clear outlier—a transaction exceeding $3,000,000, while the next highest amount is only $47,900. This transaction was made through a Mexican retailer (INTERMEXICO) and could be the result of a large international wire transfer or a currency exchange issue. Given its extreme nature, this outlier should be removed to prevent it from skewing the model.



4. Imputation Process

After filtering the dataset to include only 'Purchase' transactions and excluding the outlier, three fields still contained missing values: Merchnum, Merch state, and Merch zip.

The most common approach was to fill missing values with the most frequent value. Alternatively, grouping by a related field and imputing using the mode or average within that group also provides consistency.

1) Merchant Number (Merchnum)

Originally, there were 3,279 missing records. The first step was to map missing records to Merch description where existing merchant numbers were available, successfully filling 1,164 values and leaving 2,115 missing records.

Next, for "retail credit/debit adjustments," which were not tied to actual merchants, the value 'unknown' was assigned, filling 297 missing records and leaving 1,818 missing records.

Then the remaining records were reviewed, which included 515 unique merch descriptions like 'montgomery college-phone,' 'compressor parts & servic,' etc. For the 1,303 records with missing merchant numbers, each description was assigned a new, unique merchant number. These numbers were guaranteed to be unique and sequentially higher than the current highest merchant number in the dataset.

2) Merch state

There were 1,028 missing values in the Merch state. Most of these could be identified using a Zip code database, which matched the Zip code to the corresponding state abbreviation. For records where the Zip code did not have a match in the database, the state should be manually entered.

Merchant number is a 15-digit business identifier issued by credit card processors when a merchant account is opened. In most cases, each merchant number is associated with specific locations where

the merchant operates. Similarly, merch descriptions, which are either the business name or a descriptive phrase related to transaction, can serve as a proxy to identify the business location.

Based on this, the remaining missing values in Merch state were imputed by leveraging the relationship between Merch Num, Merch description, and Merch state. Mappings were created from records where both Merchnum or Merch description and Merch state were present. These mappings were then used to fill in missing Merch state values by assigning the state associated with the same Merchnum or Merch description elsewhere in the dataset.

After the process, 952 missing values remained. The merchant descriptions 'RETAIL CREDIT ADJUSTMENT' and 'RETAIL DEBIT ADJUSTMENT' were assigned the value 'unknown' since these represented adjustment transactions, not actual merchants. (Adjustment transactions referred to entries made by financial institutions or merchants to correct or modify previously processed transactions such as refunds, reversals, charge corrections, and fee adjustments)

When there were 297 remaining blank values, a final decision was made to label non-U.S. states as ‘foreign’. This was useful because foreign transactions may indicate potential fraud.

3) Merch zip

There were initially 4,347 missing values in the Merch zip field. The missing values were first mapped to their corresponding Merchnum. If the Merchnum was also missing, the ZIP code was filled by mapping it to the Merch description, reducing the missing values to 2,625.

For transactions classified as retail adjustments, the ZIP code was replaced with 'unknown,' as these transactions do not have relevant ZIP codes. This step reduced the missing values to 2,251.

Next, for records where the Merch state was known but the ZIP code was missing, the most populated ZIP code for that state was used as a proxy. A dictionary of the most populous ZIP codes for each U.S. state was created using publicly available data. This step further reduced the missing values to 842.

Finally, if a ZIP code was still missing, it was replaced with 'unknown,' ensuring no missing values remained in the Merch zip field.

5. Target Encoding

Target encoding is applied to categorical features to convert them into numerical values with the average target value for that category. The features chosen for target encoding are Merch state, Merch zip, and Day of the Week (Dow).

1) Fitting Target Encoder

The TargetEncoder from the category_encoders library is used to encode these categorical variables based on the target variable, Fraud. Each category value is replaced with the average fraud rate for that category.

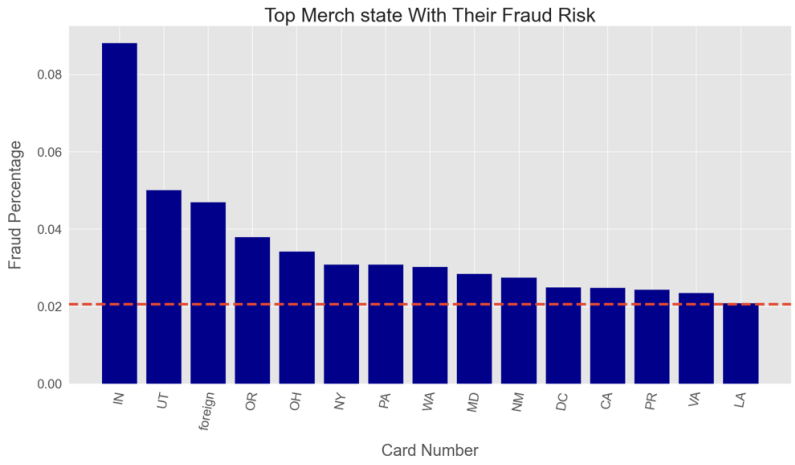
2) Adding Encoded Features

After fitting the encoder, the transformed columns are renamed with a _TE suffix (e.g., Merch state_TE, Merch zip_TE, Dow_TE) and are concatenated with the original dataset. This expands the dataset by adding new target-encoded columns.

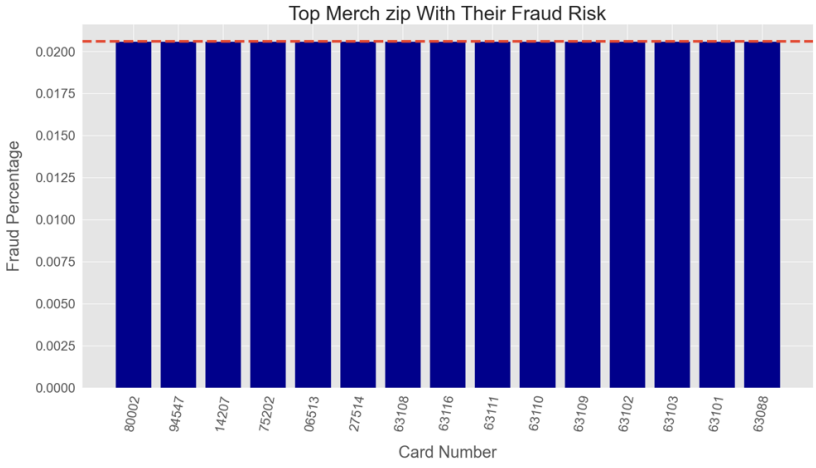
3) Visualization of Fraud Risk

The encoded values are grouped by their categorical feature, and the top 15 categories with the highest average fraud risk are visualized using bar charts. This helps identify which categories (e.g., states, ZIP codes, days) have the highest fraud risk.

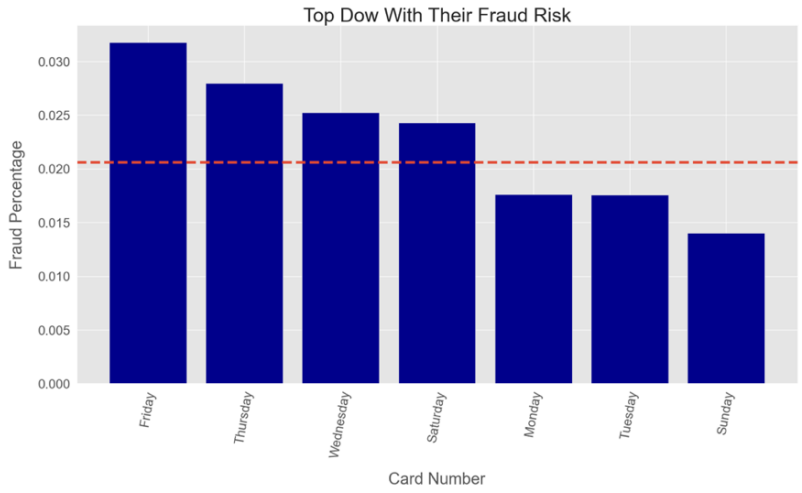
- States like IN and UT should be flagged for closer monitoring, as they exhibit a higher-than-average likelihood of fraudulent transactions. The only state that was about the same as the average fraud risk was LA.



- There are no prominent ZIP codes that significantly stand out in terms of higher or lower fraud risk. The fraud risk is relatively uniform across the listed ZIP codes.



- Fraud risk is generally higher during the workweek, particularly on Friday, Thursday and Wednesday. Friday stands out as the day with the highest fraud risk. This suggests that businesses may need to be particularly vigilant during the latter half of the workweek to prevent fraudulent transactions.



IV. Variable Creation

1. Purpose

This section describes the creation of several variables based on different aspects of transaction behavior, including time-based features, transaction frequency, amount variability, and entity relationships. Newly created variables include Day of Week (DoW), Days-Since, Frequency and Amount, Velocity Ratios, and Amount Variability, among others (detailed in the summary table below).

The rationale behind creating these variables is that credit card fraud is often characterized by patterns like rapid transaction frequency, sudden spikes in amounts, or irregular transaction behavior compared to historical norms. By generating these enhanced variables, the model becomes better equipped to detect subtle irregularities that might go unnoticed with basic transaction data.

2. Variable Creation Summary Table

Variable	Description	# Variables Created
Day of Week (DoW) *Column: DoW, DoW_Risk*	• Average fraud percentage by day • Use target encoding with smoothing to assign a numerical fraud likelihood to each day of the week	1
Days-Since *Column: entity_day_since*	• Number of days between the current and previous transaction • Abnormally large or small gaps between transactions might indicate fraudulent behavior	23
Frequency and Amount *Column: entity_count_time, avg_time, max_time, med_time, total_time*	• Frequency: How many transactions occurred within specified time windows (0, 1, 3, 7, 14, 30, 60 days) • Amount: Average, maximum, median and total transaction amounts for the entity during each time window • Ratios: Comparison of the current transaction amount to the historical average, maximum, median, and total amounts within the same time window	1,449
Velocity Amount *Column: entity_count_d_by_dd, entity_total_amount_d_by_dd*	• Number of transactions and the total transaction amounts change over different time periods • Comparison of short-term periods (0, 1 day) with long-term (7, 14, 30, 60 days) periods, normalized by the length of the long-term period • Sudden spikes in transaction frequency or amounts may indicate potential fraudulent behavior	368
Velocity Ratio *Column: entity_vdratio_d_by_dd*	• Comparison of the short-term transaction count (0, 1 day) to the long-term transaction count (7, 14, 30, 60 days), adjusted by the number of days since the last transaction • A higher velocity ratio, a quick transaction frequency changes relative to recent activity, can signal abnormal fraud behavior	184
Amount Variability *Column: entity_variability_avg_time, max_time, med_time*	• Amount variability (average, maximum, and media) over different time windows (0, 1, 3, 7, 14, 30, 60 days) • Large fluctuations or irregularities in transaction amounts, especially over short time periods, can be indicative of anomalous behaivor	414

Entity's Unique Counts *Column: i_unique_count_for_v_t*	• For each pair of entities, number of unique occurrences of entity '*I'* relative to entity '*v*' within specified time windows '*t*' (1, 3, 7, 14, 30, 60 days) • Tracking unique entity interactions over certain time periods can be useful for identifying patterns and anomalies in fraud	696
Normalized Count Ratio *Column: ent_count_d_by_dd_sq*	• Short-term transaction count (0, 1 day) to long-term transaction count (7, 14, 30, 60 days) • Ratio normalized by dividing by the square of the long-term period to adjust for the difference in time frames • Spikes in transaction frequency can highlight potential frauds	184
Amount Bins *Column: amount_cat*	• Transaction amounts divided into 5 quantile-based categories where each category represents 20% of the data • Categories labeled numerically from 1 (smallest 20%) to 5 (largest 20%)	1
Foreign Zip Code *Column: foreign*	• Assignment of a binary value (1: foreign, 0: domestic) based on the merchant's zip code (reference: list of U.S. zip codes) • Identifying foreign transactions is useful for flagging and detecting fraud cases	1

V. Feature Selection

1. Purpose

After the previous step of variable creation, this section outlines the process of selecting the most relevant features for model training. Two feature selection methods were employed: filter and wrapper. The filter method ranked features based on their filter scores, while the wrapper method used the False Discovery Rate (FDR) to evaluate performance.

Various models, including Random Forest, LGBM Classifier, and CatBoost, were tested, each applying either forward or backward stepwise selection methods, as well as different values for num_filter and other model parameters. The final model was chosen based on factors such as wrapper performance, filter score, and execution time, leading to the selection of the top 20 features for training the LGBM model.

Feature selection is critical for reducing the dimensionality of the dataset, improving model training time, and ensuring that only the most relevant variables for fraud detection are included.

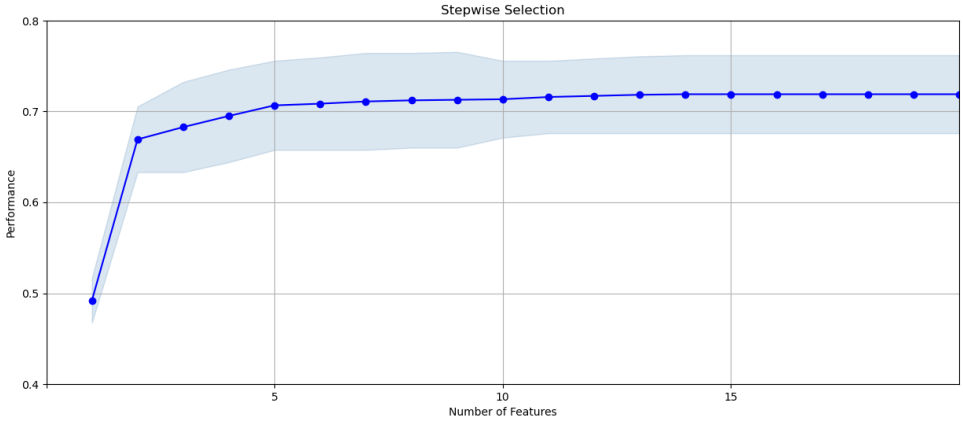
2. LGBM Classifier Model Choice

1) Methods and Parameters

- **Filter Analysis**: Out of 2635 variables, 90% were removed, and the top 10% (263 variables) were passed to the wrapper selection process based on the filter score
- **Wrapper Analysis**: After being evaluated by FDR, the top 20 features were selected
- **Model Parameter**: 10 trees and 3 layers
- **Selection Strategy**: Stepwise Forward Selection
- **Duration Time**: 7 minutes, 4 seconds

2) Saturation Plot

The final wrapper selection resulted in a performance score close to 0.70, indicating that the selected features significantly contributed to identifying fraud cases. The performance stabilized after around 10 features and remained consistent through the final 20, suggesting that these features were the most important for the model.

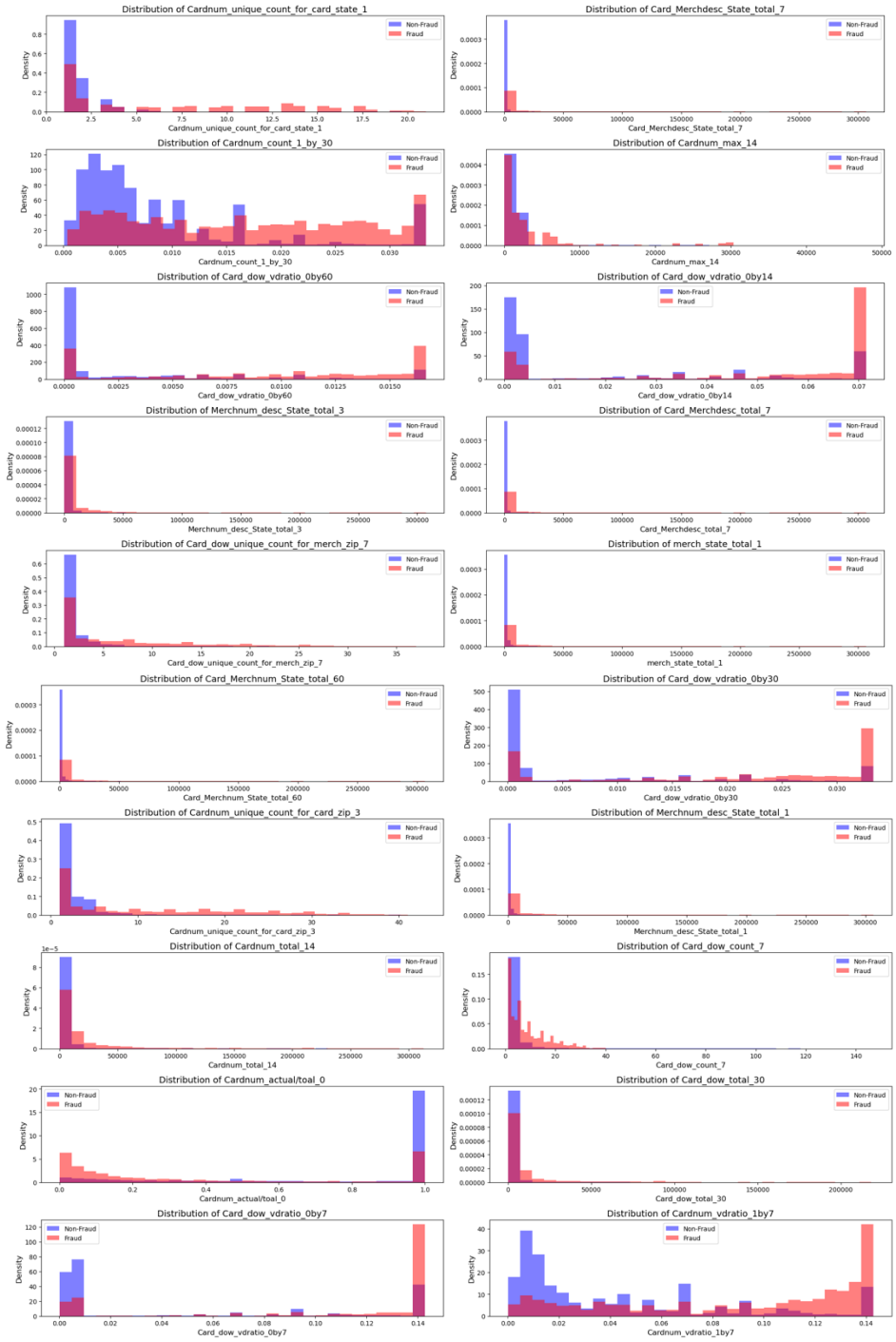


3) Final Top 20 Features

- **List**

wrapper order	variable	filter score
1	Cardnum_unique_count_for_card_state_1	0.4760666122777140
2	Card_Merchdesc_State_total_7	0.32466842229574100
3	Cardnum_count_1_by_30	0.42822889583921400
4	Cardnum_max_14	0.31882556436477600
5	Card_dow_vdratio_0by60	0.48648015528051000
6	Card_dow_vdratio_0by14	0.47908633628938200
7	Merchnum_desc_State_total_3	0.3085859758562840
8	Card_Merchdesc_total_7	0.32463084996901200
9	Card_dow_unique_count_for_merch_zip_7	0.4189430451164500
10	merch_state_total_1	0.3048932077685130
11	Card_Merchnum_State_total_60	0.30538533609714000
12	Card_dow_vdratio_0by30	0.48922717642504800
13	Cardnum_unique_count_for_card_zip_3	0.4643230396427230
14	Merchnum_desc_State_total_1	0.3049692744422580
15	Cardnum_total_14	0.4943749232610200
16	Card_dow_count_7	0.482384034050824
17	Cardnum_actual/toal_0	0.47955008176014900
18	Card_dow_total_30	0.4747594487486570
19	Card_dow_vdratio_0by7	0.4679610397253980
20	Cardnum_vdratio_1by7	0.4667663012034360

- **Plot Distributions**



VI. Preliminary Model Exploration

1. Purpose

In this section, various machine learning algorithms are explored to assess their effectiveness in detecting fraudulent credit card transactions. The models considered include Logistic Regression, Decision Tree, Random Forest, LightGBM, CatBoost, and Neural Networks.

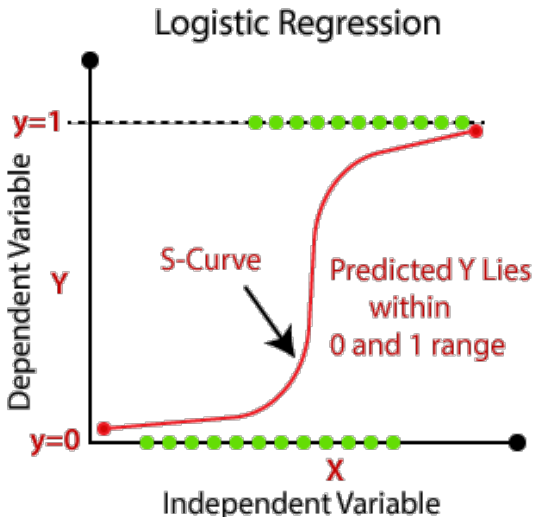
These models are tuned and evaluated using their own set of hyperparameters to achieve an optimal balance between training accuracy, testing accuracy, and out-of-time (OOT) performance. The selection criteria for the models are based on two key factors: (1) Minimal Train-Test Difference, where the difference between training and testing performance is kept below 0.040 to avoid overfitting, and (2) OOT Performance, where the model's ability to generalize to unseen data is prioritized.

By testing different models and fine-tuning their hyperparameters, the objective is to develop a model that performs well on training data, generalizes effectively to real-world data, and avoids overfitting.

2. ML Algorithms Description

1) Logistic Regression

- Logistic Regression is used for binary classification tasks, where the outcome is a binary variable (taking one of two values). The model estimates the probability that a given input belongs to a particular class by applying the logistic function, which transforms the linear combination of input features into a probability value between 0 and 1.

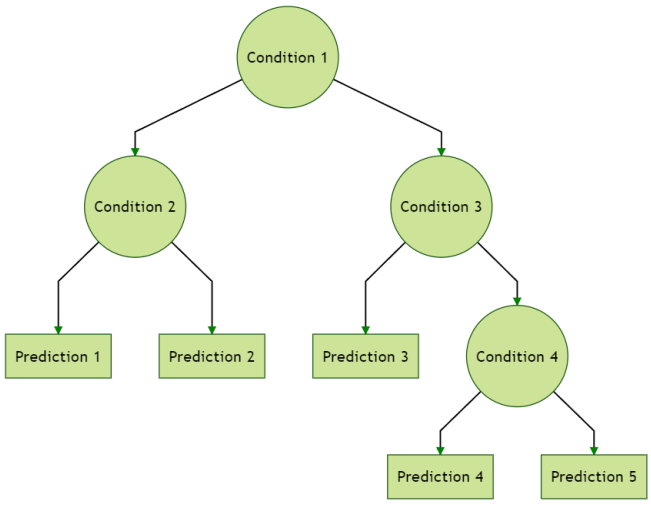


- **Key parameters**
  - Penalty: Regularization technique that adds a constraint to prevent overfitting

        - L1 (Lasso regularization): shrinks less important feature coefficients to zero, essentially performing feature selection
        - L2 (Ridge regularization): shrinks coefficients but doesn't set any of them to zero, helps reduce overfitting while keeping all features
        - ElasticNet: combines both L1 and L2
    - C: Regularization strength, smaller values mean stronger regularization
    - Solver: Finds the optimal coefficients for minimizing the loss function
        - Lbfgs: L2 regularization, large datasets
        - Liblinear: L1 regularization, small datasets
        - Sag: L2 regularization, large datasets
        - Saga: ElasticNet
    - L1 ratio: Mixed ratio btw L1 and L2 for ElasticNet
        - l1_ratio=0: purely L2 Ridge
        - l1_ratio=1: purely L1 Lasso

2) Decision Tree

- A Decision Tree is a supervised learning algorithm used for classification and regression tasks. It splits data into subsets based on feature values, creating a tree-like structure where each internal node represents a decision rule, each branch represents an outcome, and each leaf node represents a class label or continuous value. It makes decisions by recursively partitioning the data until a specific criterion is met.

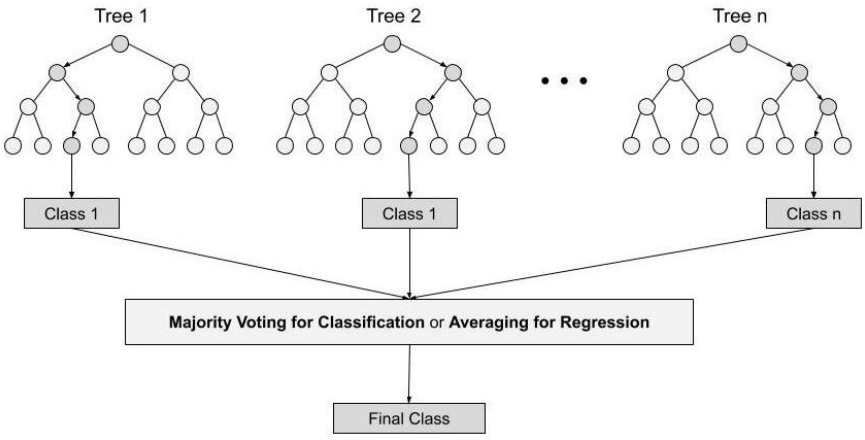


- **Key Parameters**
    - Criterion: Defines how to evaluate the quality of a split
        - gini: faster to compute and is the default criterion
        - entropy: more computationally expensive but can sometimes result in a more accurate tree
    - Max depth: Limits how deep the tree can grow
        - Smaller depth (e.g., 3-5): results in a simpler model with fewer decision rules, which can generalize better but may underfit the data

        - Larger depth (e.g., 10-20): allows the tree to capture more complexity, but it may overfit by learning noise in the training data
    - Min samples leaf: Defines the minimum number of samples required at a leaf
        - Higher values (e.g., 5, 20): result in more generalized trees, as the leaf nodes will need more samples, which reduces the chance of overfitting
        - Lower values (e.g., 1): allow the model to split until there's only one sample per leaf, which can lead to overfitting
    - Splitter: Determines how the algorithm chooses the feature to split
        - best: generally gives better performance, as it optimally splits based on the selected criterion (Gini or Entropy)
        - random: might make the model faster but could lead to less accurate results since the splits aren't chosen optimally

3) Random Forest

- A Random Forest is an ensemble learning algorithm used for classification and regression tasks. It builds multiple decision trees during training and aggregates their predictions (by voting for classification or averaging for regression). Each tree is trained on a random subset of data and features, making the model more robust, reducing overfitting, and improving accuracy compared to individual decision trees.

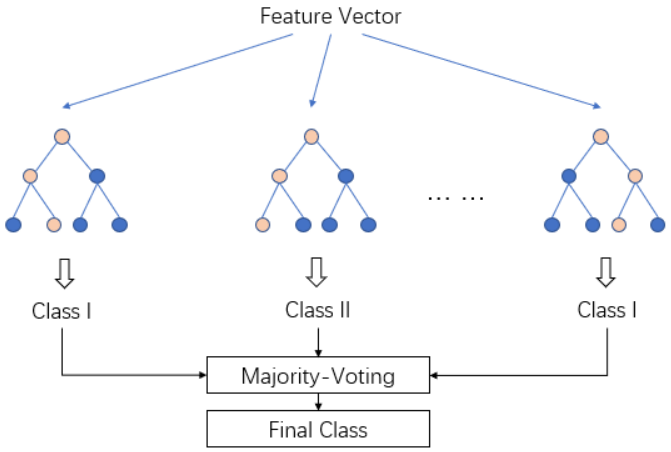


- **Key Parameters**
    - Criterion: Defines how to evaluate the quality of a split
        - gini: faster to compute and is the default criterion
        - entropy: more computationally expensive but can sometimes result in a more accurate tree
    - Max depth: Limits how deep the tree can grow
        - Smaller depth (e.g., 3-5): results in a simpler model with fewer decision rules, which can generalize better but may underfit the data
        - Larger depth (e.g., 10-20): allows the tree to capture more complexity, but it may overfit by learning noise in the training data
    - Min samples leaf: Defines the minimum number of samples required at a leaf
        - Higher values (e.g., 5, 20): result in more generalized trees, as the leaf nodes will need more samples, which reduces the chance of overfitting

        - Lower values (e.g., 1): allow the model to split until there's only one sample per leaf, which can lead to overfitting
    - N_estimators: Defines the number of decision trees that are built
        - n_estimators=100: default
        - For larger datasets or when more accuracy is needed, can go up to n_estimators=200 or n_estimators=500
        - Very large values like n_estimators=1000 can further improve accuracy but take significantly longer to run
    - Bootstrap: Defines whether or not bootstrap samples are used
        - bootstrap=True: each tree is trained on a bootstrapped subset of the data (a random sample with replacement)
        - bootstrap=False: each tree is trained on the entire training dataset without any bootstrapping

4) Light LGBM

- LightGBM (Light Gradient Boosting Machine) is a fast, high-performance gradient boosting framework for classification and regression tasks. It uses decision tree-based learning algorithms and is optimized for speed and efficiency. LightGBM works by growing trees leaf-wise, rather than level-wise like traditional boosting algorithms, and uses techniques like histogram-based binning to handle large datasets and high-dimensional features efficiently. It is well-suited for large-scale datasets, offers faster training, and has lower memory usage compared to other boosting methods.

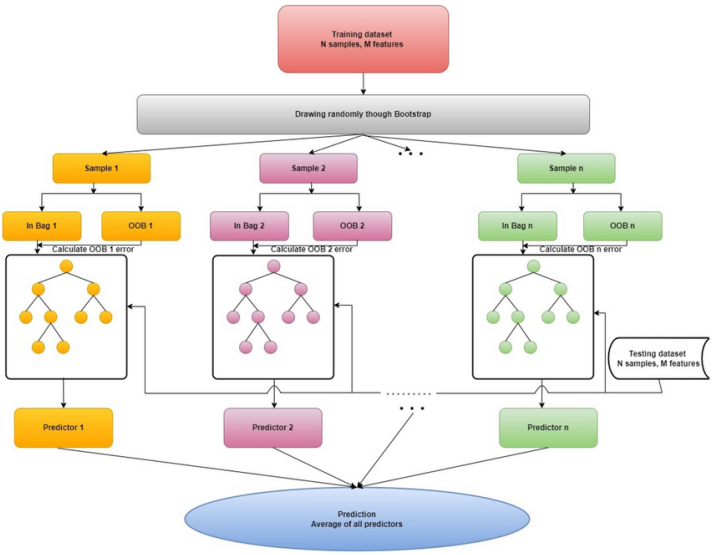


- **Key Parameters**
    - Subsample: The process of using a fraction of the data to train each tree
        - Typical Values: values range between 0.5 to 1.0
        - Effect: a lower value may reduce overfitting but could also lead to underfitting if it's too small
        - Common: 0.7 - 0.9 is often used to reduce overfitting
    - Max depth: Limits how deep the tree can grow
        - Smaller depth (e.g., 3-5): results in a simpler model with fewer decision rules, which can generalize better but may underfit the data

        - Larger depth (e.g., 10-20): allows the tree to capture more complexity, but it may overfit by learning noise in the training data
    - Learning rate: Controls the contribution of each tree in the boosted ensemble
        - Effect: a smaller learning rate slows down the learning process but allows more trees to be built, typically resulting in better generalization, A larger learning rate can make the model fit faster but can cause overfitting or make the model unstable
        - Common: default is 0.1, for better generalization, reduce it to 0.01 or 0.05 but more trees (higher n_estimators) will be needed to compensate
    - N_estimators: Defines the number of decision trees that are built
        - n_estimators=100: default
        - For larger datasets or when more accuracy is needed, can go up to n_estimators=200 or n_estimators=500
        - Very large values like n_estimators=1000 can further improve accuracy but take significantly longer to run
    - Metric: Evaluation criterion used to measure the performance of the model
        - logloss, binary_logloss: classification tasks (switch to auc to focus on imbalanced dataset or ranking problems)
        - rmse (mae: mean absolute error): regression tasks

5) CatBoost

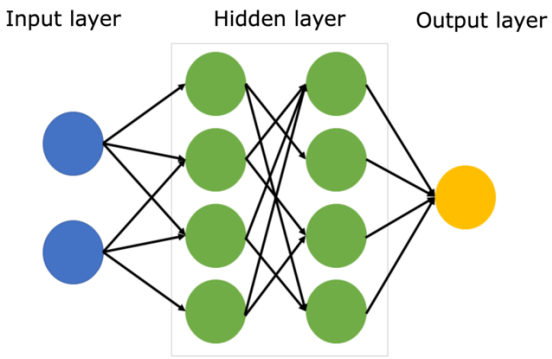
- CatBoost is a powerful gradient boosting algorithm designed to efficiently handle both categorical and numerical data. It stands out for its ability to process categorical features without requiring complex preprocessing like one-hot encoding, which simplifies the workflow and reduces potential errors. CatBoost is highly effective in classification and regression tasks, delivering fast training times and superior accuracy. Its use of ordered boosting minimizes overfitting, especially on smaller datasets, making it a versatile and reliable choice for a wide range of real-world applications. Its balance of performance and ease of use makes it popular for industries dealing with complex, structured data.



- **Key Parameters**
  - Subsample: The process of using a fraction of the data to train each tree
    - Typical Values: values range between 0.5 to 1.0
    - Effect: a lower value may reduce overfitting but could also lead to underfitting if it's too small
    - Common: 0.7 - 0.9 is often used to reduce overfitting
  - Max depth: Limits how deep the tree can grow
    - Smaller depth (e.g., 3-5): results in a simpler model with fewer decision rules, which can generalize better but may underfit the data
    - Larger depth (e.g., 10-20): allows the tree to capture more complexity, but it may overfit by learning noise in the training data
  - Learning rate: Controls the contribution of each tree in the boosted ensemble
    - Effect: a smaller learning rate slows down the learning process but allows more trees to be built, typically resulting in better generalization, A larger learning rate can make the model fit faster but can cause overfitting or make the model unstable
    - Common: default is 0.1, for better generalization, reduce it to 0.01 or 0.05, but more trees (higher n_estimators) will be needed to compensate
  - L2 leaf reg: Penalizing large leaf values by applying L2 regularization
    - Default: 3
    - Smaller values (e.g., 1): allow more complex models but risk overfitting
    - Larger values (e.g., 10): make the model simpler by limiting how much each tree's leaf can influence the final prediction, reducing overfitting but possibly underfitting

6) Neural Network

- A Neural Network is a machine learning model inspired by the structure and function of the human brain. It consists of layers of interconnected nodes (neurons) where each connection has an associated weight. The network learns by adjusting these weights based on the input data. Neural networks are highly flexible and can model complex, non-linear relationships, making them suitable for a wide range of tasks, including classification, regression, image recognition, and natural language processing. They are especially effective when working with large datasets.



Artificial neural networks

- **Key Parameters**
  - Activation: Defines the activation function to be used in the hidden layers
    - relu (rectified linear unit): default, especially for deep layers
    - logistic: often used in binary classification tasks, used in simpler or older neural networks
    - tanh (hyperbolic tangent function): outputs values between -1 and 1, used in simpler or older neural networks
  - Alpha: L2 regularization term penalizing large coefficients
    - A higher value of alpha means more regularization, which can reduce overfitting but may cause underfitting
    - Default: 0.0001, a range of 0.0001 to 0.001 is typical
  - Learning_rate: Controls how much the model adjusts its weights with respect to the loss gradient
    - constant: a fixed learning rate
    - adaptive: adjusts the learning rate based on validation score improvement
    - 0.001 to 0.1: typical ranges for most neural networks, smaller values (like 0.001 or 0.01) often work best with deep network
  - Hidden_layer_size: Specifies the number of neurons in the hidden layers
    - 100: default, one hidden layer with 100 neurons
    - (50, 100), (150, 50), etc: increasing capacity from first to second layer
  - Solver: Determines the optimization algorithm used for weight updates
    - adam (default): a combination of RMSprop and momentum-based optimization, which is efficient for large datasets
    - sgd: stochastic Gradient Descent, often slower and requires careful tuning of learning rates
    - lbfgs: an optimizer that works well for smaller datasets but can be slow

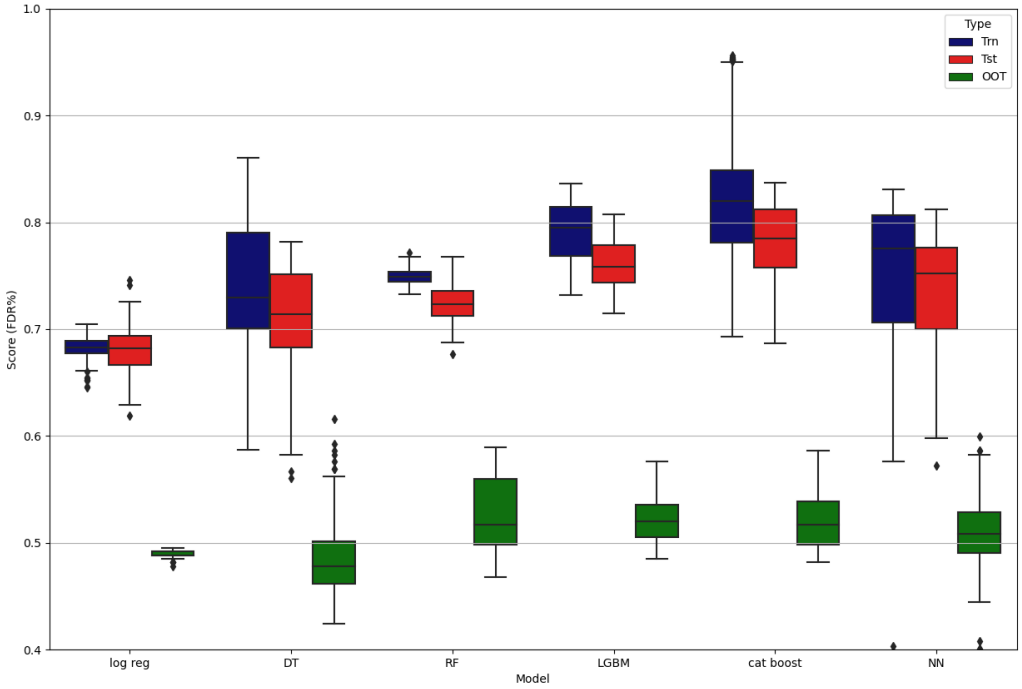
3. Model Exploration Table

Model		Parameters					Average FDR at 3%			
Type	**Ver**	**penalty**	**C**	**solver**	**l1_ratio**		**Trn**	**Tst**	**OOT**	**Trn-Tst**
Logistic Regression	1	L2	0.01	lbfgs	None		0.687	0.682	0.488	0.005
	2	L2	0.001	lbfgs	None		0.688	0.679	0.490	0.009
	3	L2	0.01	sag	None		0.691	0.675	0.486	0.016
	4	L1	0.001	liblinear	None		0.654	0.650	0.491	0.004
	5	Elastic Net	0.01	saga	0.5		0.685	0.685	0.492	0.000
	6	Elastic Net	0.01	saga	0.2		0.686	0.685	0.487	0.001
	7	Elastic Net	0.001	saga	0.2		0.685	0.676	0.492	0.009
Decision Tree	**Ver**	**criterion**	**max_depth**	**min_samples_leaf**	**splitter**		**Trn**	**Tst**	**OOT**	**Trn-Tst**
	1	gini	3	1	best		0.646	0.640	0.428	0.006
	2	gini	3	10	best		0.641	0.630	0.427	0.011
	3	gini	5	1	best		0.707	0.686	0.471	0.021
	4	gini	5	5	best		0.703	0.688	0.463	0.015
	5	gini	5	60	best		0.704	0.692	0.479	0.012

	6	gini	10	80	best		0.745	0.726	0.539	0.019
	7	entropy	5	40	best		0.729	0.718	0.484	0.011
	8	entropy	5	60	best		0.736	0.705	0.486	0.031
Random Forest	**Ver**	**criterion**	**max_depth**	**min_samples_leaf**	**n_estimator**	**bootstrap**	**Trn**	**Tst**	**OOT**	**Trn-Tst**
	1	gini	5	40	150	true	0.756	0.724	0.541	0.032
	2	gini	5	60	100	true	0.756	0.735	0.544	0.021
	3	gini	5	60	150	true	0.756	0.726	0.555	0.030
	4	gini	10	50	200	true	0.755	0.737	0.541	0.018
	5	gini	10	60	200	true	0.755	0.742	0.547	0.013
	6	entropy	5	40	100	true	0.756	0.725	0.529	0.031
	7	entropy	5	60	100	true	0.756	0.728	0.536	0.028
	8	entropy	5	60	150	true	0.757	0.734	0.543	0.023
	9	entropy	10	40	200	true	0.759	0.735	0.549	0.024
	10	entropy	10	60	200	true	0.754	0.727	0.512	0.027
Light LGBM	**Ver**	**subsample**	**max_depth**	**learning_rate**	**n_estimator**	**metric**	**Trn**	**Tst**	**OOT**	**Trn-Tst**
	1	0.7	5	0.01	100	auc	0.786	0.771	0.532	0.015
	2	0.7	5	0.01	200	auc	0.820	0.797	0.548	0.023
	3	0.7	5	0.001	100	auc	0.753	0.725	0.502	0.028
	4	0.7	10	0.001	100	auc	0.790	0.757	0.516	0.033
	5	0.7	10	0.001	200	binary_log loss	0.808	0.774	0.524	0.034
	6	0.8	5	0.001	200	binary_log loss	0.760	0.732	0.516	0.028
	7	0.8	10	0.001	200	binary_log loss	0.806	0.769	0.523	0.037
CatBoost	**Ver**	**subsample**	**max_depth**	**learning_rate**	**n_estimator**	**l2_leaf_reg**	**Trn**	**Tst**	**OOT**	**Trn-Tst**
	1	0.7	5	0.1	100	3	0.777	0.747	0.508	0.030
	2	0.7	5	0.1	150	3	0.796	0.784	0.515	0.012
	3	0.7	5	0.1	200	3	0.785	0.785	0.511	0.000
	4	0.7	10	0.1	100	3	0.830	0.792	0.532	0.038
	5	0.7	10	0.1	200	3	0.947	0.826	0.553	0.121
	6	0.8	5	0.1	100	3	0.773	0.760	0.503	0.013
	8	0.8	10	0.01	200	3	0.800	0.767	0.511	0.033
Neural Network	**Ver**	**activation**	**alpha**	**learning_rate**	**hidden_layer_size**	**solver**	**Trn**	**Tst**	**OOT**	**Trn-Tst**
	1	logistic	0.0001	constant	(100,)	adam	0.774	0.749	0.512	0.025
	2	logistic	0.0001	constant	(50, 100)	adam	0.784	0.763	0.514	0.021
	3	logistic	0.0001	adaptive	(100,)	adam	0.763	0.748	0.524	0.015
	4	logistic	0.0001	adaptive	(50, 100)	adam	0.778	0.749	0.513	0.029
	5	logistic	0.0001	adaptive	(150, 50)	adam	0.812	0.780	0.526	0.032
	6	relu	0.0001	adaptive	(100,)	adam	0.811	0.779	0.557	0.032
	7	relu	0.0001	adaptive	(100,)	sgd	0.706	0.697	0.480	0.009
	8	relu	0.0001	adaptive	(50, 100)	sgd	0.714	0.705	0.486	0.009

4. Summary Box Plot

- Based on the model exploration table, the best version of each model was selected for comparison across all model types. The primary criterion for selecting the optimal model is maintaining a minimal difference between training and testing performance, with an Out-of-Time (OOT) score consistently above 0.50.



- CatBoost shows good balance between training and testing, but the long whiskers represent high variability in the model's performance across iterations, which might indicate inconsistency and risk in certain datasets.
- LightGBM and Neural Network (NN) both perform well in terms of out-of-sample (OOT) data, though NN displays more variability, with long whiskers and a few outliers, suggesting greater instability and higher risk of overfitting.
- Decision Tree (DT) and Random Forest (RF), while showing a decent balance between training and testing, perform lower in terms of OOT, making them less ideal choices compared to CatBoost and LightGBM.
- Logistic Regression, while displaying minimal variance between training, testing, and OOT, achieves overall lower scores, indicating it may not capture the complexity of the dataset as effectively as more advanced models like CatBoost and LightGBM. However, its consistency and simplicity may make it preferable when aiming to avoid overfitting.

VII. Final Model Performance

1. Final Model Description

The final model, CatBoost's optimal hyperparameters were as follows: subsample set to 0.7, maximum depth of 5, learning rate of 0.01, 100 estimators, and AUC as the evaluation metric.

The model demonstrated a training accuracy of 90.54%, testing accuracy of 86.63%, and out-of-time (OOT) accuracy of 55.67%. The accompanying summary table presents population bin percentages, which indicate how the each data population bin, based on fraud score, corresponded to fraudulent transactions. Additionally, it highlights the potential savings that could be achieved by preventing these fraudulent transactions.

By examining the results, one can observe that targeting transactions with higher fraud scores leads to significant savings for the company, as shown by the fraud detection rate (FDR) and overall financial savings metrics in each bin.

Training	# Records	# Goods	# Bads	Fraud Rate
	59,684	10,810	1,127	0.0189

	Bin Statistics					Cumulative Statistics							Fraud Savings		
Population Bin %	# Records	# Goods	# Bads	%Goods	%Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR	Fraud Savings	FP Loss	Overall Savings
1	597	10	587	1.68%	98.32%	597	010	587	0.02%	49.87%	49.86	0.02	$ 234,800	$ 200	$ 234,600
2	597	243	354	40.70%	59.30%	1,194	253	941	0.43%	79.95%	79.52	0.27	$ 376,400	$ 5,060	$ 371,340
3	597	533	64	89.28%	10.72%	1,791	786	1,005	1.34%	85.39%	84.04	0.78	$ 402,000	$ 15,720	$ 386,280
4	596	566	30	94.97%	5.03%	2,387	1,352	1,035	2.31%	87.94%	85.62	1.31	$ 414,000	$ 27,040	$ 386,960
5	597	576	21	96.48%	3.52%	2,984	1,928	1,056	3.30%	89.72%	86.42	1.83	$ 422,400	$ 38,560	$ 383,840
6	597	581	16	97.32%	2.68%	3,581	2,509	1,072	4.29%	91.08%	86.79	2.34	$ 428,800	$ 50,180	$ 378,620
7	597	587	10	98.32%	1.68%	4,178	3,096	1,082	5.29%	91.93%	86.64	2.86	$ 432,800	$ 61,920	$ 370,880
8	597	588	9	98.49%	1.51%	4,775	3,684	1,091	6.30%	92.69%	86.40	3.38	$ 436,400	$ 73,680	$ 362,720
9	597	591	6	98.99%	1.01%	5,372	4,275	1,097	7.31%	93.20%	85.90	3.90	$ 438,800	$ 85,500	$ 353,300
10	596	594	2	99.66%	0.34%	5,968	4,869	1,099	8.32%	93.37%	85.05	4.43	$ 439,600	$ 97,380	$ 342,220
11	597	595	2	99.66%	0.34%	6,565	5,464	1,101	9.34%	93.54%	84.20	4.96	$ 440,400	$ 109,280	$ 331,120
12	597	596	1	99.83%	0.17%	7,162	6,060	1,102	10.36%	93.63%	83.27	5.50	$ 440,800	$ 121,200	$ 319,600
13	597	593	4	99.33%	0.67%	7,759	6,653	1,106	11.37%	93.97%	82.60	6.02	$ 442,400	$ 133,060	$ 309,340
14	597	593	4	99.33%	0.67%	8,356	7,246	1,110	12.38%	94.31%	81.92	6.53	$ 444,000	$ 144,920	$ 299,080
15	597	595	2	99.66%	0.34%	8,953	7,841	1,112	13.40%	94.48%	81.08	7.05	$ 444,800	$ 156,820	$ 287,980
16	596	592	4	99.33%	0.67%	9,549	8,433	1,116	14.41%	94.82%	80.40	7.56	$ 446,400	$ 168,660	$ 277,740
17	597	590	7	98.83%	1.17%	10,146	9,023	1,123	15.42%	95.41%	79.99	8.03	$ 449,200	$ 180,460	$ 268,740
18	597	595	2	99.66%	0.34%	10,743	9,618	1,125	16.44%	95.58%	79.14	8.55	$ 450,000	$ 192,360	$ 257,640
19	597	595	2	99.66%	0.34%	11,340	10,213	1,127	17.46%	95.75%	78.30	9.06	$ 450,800	$ 204,260	$ 246,540
20	597	597	0	100%	0%	11,937	10,810	1,127	18.48%	95.75%	77.28	9.59	$ 450,800	$ 216,200	$ 234,600

Testing	# Records	# Goods	# Bads	Fraud Rate
	25,580	25,007	547	0.0224

	Bin Statistics					Cumulative Statistics							Fraud Savings		
Population Bin %	# Records	# Goods	# Bads	%Goods	%Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR	Fraud Savings	FP Loss	Overall Savings
1	256	19	237	7.42%	92.58%	256	019	237	0.08%	41.36%	41.29	0.08	$ 94,800	$ 380	$ 94,420
2	256	87	169	33.98%	66.02%	512	106	406	0.42%	70.86%	70.43	0.26	$ 162,400	$ 2,120	$ 160,280
3	255	215	40	84.31%	15.69%	767	321	446	1.28%	77.84%	76.55	0.72	$ 178,400	$ 6,420	$ 171,980
4	256	229	27	89.45%	10.55%	1,023	550	473	2.20%	82.55%	80.35	1.16	$ 189,200	$ 11,000	$ 178,200
5	256	242	14	94.53%	5.47%	1,279	792	487	3.17%	84.99%	81.82	1.63	$ 194,800	$ 15,840	$ 178,960
6	256	249	7	97.27%	2.73%	1,535	1,041	494	4.16%	86.21%	82.05	2.11	$ 197,600	$ 20,820	$ 176,780
7	256	247	9	96.48%	3.52%	1,791	1,288	503	5.15%	87.78%	82.63	2.56	$ 201,200	$ 25,760	$ 175,440
8	255	249	6	97.65%	2.35%	2,046	1,537	509	6.15%	88.83%	82.68	3.02	$ 203,600	$ 30,740	$ 172,860
9	256	254	2	99.22%	0.78%	2,302	1,791	511	7.16%	89.18%	82.02	3.50	$ 204,400	$ 35,820	$ 168,580
10	256	251	5	98.05%	1.95%	2,558	2,042	516	8.17%	90.05%	81.89	3.96	$ 206,400	$ 40,840	$ 165,560
11	256	251	5	98.05%	1.95%	2,814	2,293	521	9.17%	90.92%	81.76	4.40	$ 208,400	$ 45,860	$ 162,540
12	256	254	2	99.22%	0.78%	3,070	2,547	523	10.19%	91.27%	81.09	4.87	$ 209,200	$ 50,940	$ 158,260
13	255	251	4	98.43%	1.57%	3,325	2,798	527	11.19%	91.97%	80.78	5.31	$ 210,800	$ 55,960	$ 154,840
14	256	251	5	98.05%	1.95%	3,581	3,049	532	12.19%	92.84%	80.65	5.73	$ 212,800	$ 60,980	$ 151,820
15	256	253	3	98.83%	1.17%	3,837	3,302	535	13.20%	93.37%	80.16	6.17	$ 214,000	$ 66,040	$ 147,960
16	256	252	4	98.44%	1.56%	4,093	3,554	539	14.21%	94.07%	79.85	6.59	$ 215,600	$ 71,080	$ 144,520
17	256	254	2	99.22%	0.78%	4,349	3,808	541	15.23%	94.42%	79.19	7.04	$ 216,400	$ 76,160	$ 140,240
18	255	252	3	98.82%	1.18%	4,604	4,060	544	16.24%	94.94%	78.70	7.46	$ 217,600	$ 81,200	$ 136,400
19	256	255	1	99.61%	0.39%	4,860	4,315	545	17.26%	95.11%	77.86	7.92	$ 218,000	$ 86,300	$ 131,700
20	256	254	2	99.22%	0.78%	5,116	4,569	547	18.27%	95.46%	77.19	8.35	$ 218,800	$ 91,380	$ 127,420

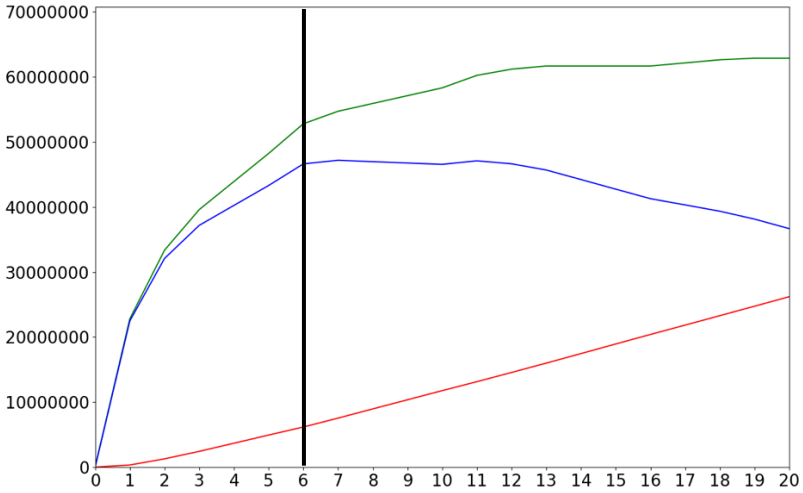
OOT	# Records	# Goods	# Bads	Fraud Rate
	12,232	11,935	262	0.0242

	Bin Statistics					Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	%Goods	%Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	122	27	95	22.13%	77.87%	122	027	095	0.23%	31.99%	31.76	0.28
2	123	79	44	64.23%	35.77%	245	106	139	0.89%	46.80%	45.91	0.76
3	122	96	26	78.69%	21.31%	367	202	165	1.69%	55.56%	53.86	1.22
4	122	104	18	85.25%	14.75%	489	306	183	2.56%	61.62%	59.05	1.67
5	123	105	18	85.37%	14.63%	612	411	201	3.44%	67.68%	64.23	2.04
6	122	103	19	84.43%	15.57%	734	514	220	4.31%	74.07%	69.77	2.34
7	122	114	8	93.44%	6.56%	856	628	228	5.26%	76.77%	71.51	2.75
8	123	118	5	95.93%	4.07%	979	746	233	6.25%	78.45%	72.20	3.20
9	122	117	5	95.90%	4.10%	1,101	863	238	7.23%	80.13%	72.90	3.63
10	122	117	5	95.90%	4.10%	1,223	980	243	8.21%	81.82%	73.61	4.03
11	123	115	8	93.50%	6.50%	1,346	1,095	251	9.17%	84.51%	75.34	4.36
12	122	118	4	96.72%	3.28%	1,468	1,213	255	10.16%	85.86%	75.70	4.76
13	122	120	2	98.36%	1.64%	1,590	1,333	257	11.17%	86.53%	75.36	5.19
14	122	122	0	100%	0%	1,712	1,455	257	12.19%	86.53%	74.34	5.66
15	123	123	0	100%	0%	1,835	1,578	257	13.22%	86.53%	73.31	6.14
16	122	122	0	100%	0%	1,957	1,700	257	14.24%	86.53%	72.29	6.61
17	122	120	2	98.36%	1.64%	2,079	1,820	259	15.25%	87.21%	71.96	7.03
18	123	121	2	98.37%	1.63%	2,202	1,941	261	16.26%	87.88%	71.62	7.44
19	122	121	1	99.18%	0.82%	2,324	2,062	262	17.28%	88.22%	70.94	7.87
20	122	122	0	100%	0%	2,446	2,184	262	18.30%	88.22%	69.92	8.34

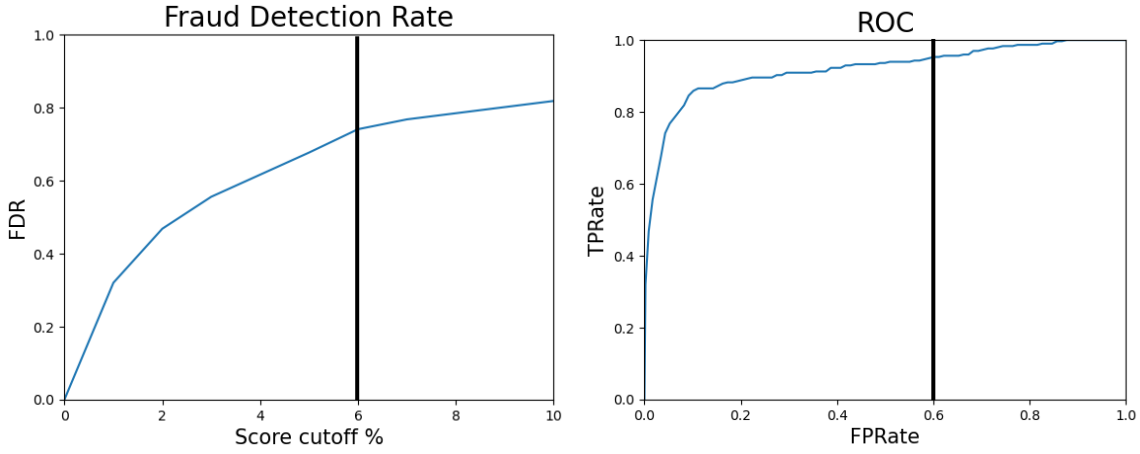
VIII. Financial Curves and Recommended Cutoff

This section evaluates the financial impact of fraud detection across score cutoffs, highlighting how the savings from fraud detection compare to the losses incurred from false positives.

- Green Line (Fraud Savings): Represents the total savings generated by detecting fraudulent transactions at each cutoff threshold.
- Red Line (False Positive Loss): Displays the losses associated with false positives, or the cost of misclassifying legitimate transactions as fraudulent.
- Blue Line (Overall Savings): Shows the net savings by subtracting the false positive losses from the fraud savings.



- Fraud Detection Rate (FDR): Analyzing the cutoff % and the corresponding FDR that maximizes savings while reducing costs from investigating flagged transactions
- ROC: Trade-off between True Positive Rate (TPR) and False Positive Rate (FPR)



Based on the plots, choosing bin 6 as the cutoff offers the maximum possible savings of $47.18 million. This means the business should consider taking action (e.g., flagging transactions or

denying access) on the top 6% of the highest-risk transactions. This cutoff strikes a middle ground, maximizing savings while keeping false positives at a level that minimizes disruptions for legitimate customers.

IX. General Summary

In this project, the primary goal was to detect and mitigate fraudulent credit card transactions by developing a robust machine learning model. The dataset consisted of over 97,000 transaction records from a U.S. government organization in Tennessee, with approximately 2,000 manually generated fraud cases for analysis. A comprehensive data cleaning process was conducted to ensure that only relevant purchase transactions were included, outliers were addressed, and missing values were imputed using several logical relationships between the data fields.

Next, variables were created based on transaction behaviors, such as frequency, amount variability, and velocity ratios. These variables were key to understanding the nuances of transaction patterns, particularly those associated with fraud. Feature selection methods, including both filter and wrapper techniques, were employed to reduce the dimensionality of the data and enhance model performance by selecting the most relevant features for training.

The project explored multiple machine learning models, including Logistic Regression, Decision Trees, Random Forests, LightGBM, CatBoost, and Neural Networks. Through extensive tuning and evaluation of hyperparameters, the CatBoost model was selected as the final model based on its balanced performance in training, testing, and out-of-time (OOT) data.

Financial analysis was performed to assess the impact of various score cutoffs on fraud detection. By setting a cutoff at 6%, the model achieved an FDR (Fraud Detection Rate) of 60% on OOT data, with projected annual savings of approximately $47.18 million. This cutoff represented a middle ground, balancing fraud detection with the costs associated with investigating false positives and ensuring minimal disruption to legitimate customers.

Finally, the project's success was evaluated through financial curves, ROC, and FDR plots, which helped identify the optimal score cutoff and quantify the savings from fraud detection efforts. This project demonstrates a strategic approach to fraud detection, leveraging advanced analytics to mitigate fraud risk while protecting customer trust and maximizing financial savings.

There are potential avenues for further enhancement throughout the process. For model improvement techniques, iterative training could be applied by focusing on specific score regions, such as eliminating borderline scoring records or training only on extreme records, to improve decision-making accuracy. Additionally, segmentation could create separate models for different transaction types or customer profiles, leading to more specialized and accurate models.

Handling imbalanced data is another area for improvement. Techniques like SMOTE (Synthetic Minority Oversampling Technique) can generate artificial fraud records to balance the dataset, helping the model learn from more fraudulent data points and improving fraud detection in imbalanced scenarios.

Lastly, score calibration could be further refined to ensure that the probabilities assigned to fraud scores are more interpretable and actionable for business stakeholders, aligning the model's output with business risk management strategies.