Marshall School of Business

A Report On

Project 1: Credit Card Transactions Data

Submitted in partial fulfillment for the subject

DSO-562: Fraud Analytics

Academic Term: Fall 2024

Submission Date: October 16, 2024

Submitted By:

Jnana Kundur Prakash

Course Instructor:

Stephen Coggeshall, Adjunct Professor

Data Sciences and Operations Department

USC Marshall School of Business

Table of Contents

Chapters	Page Number
1. Executive Summary	3
2. Data Description	4 - 7
3. Data Cleaning	8 - 14
4. Variable Creation	15 - 20
5. Feature Selection	21 - 30
6. Preliminary Model Exploration	31 - 44
7. Final Model Performance	45 - 52
8. Financial Curves and Recommended Cutoff	53 - 55
9. Summary	56
10. Appendix	57 - 63

Chapter 1: Executive Summary

This project successfully deployed a sophisticated Light Gradient Boosting Machine (LightGBM) model to enhance our fraud detection capabilities across a comprehensive dataset of nearly 98,000 credit card transactions from the year 2010. This strategic initiative is part of our ongoing efforts to leverage advanced analytics in combating financial fraud, thereby safeguarding our revenue streams and enhancing customer trust.

The model demonstrated exceptional performance in identifying fraudulent activities, achieving an impressive Fraud Detection Rate (FDR) of 87.54% within the top 3% of transactions deemed most risky in an out-of-time test set. This high level of accuracy in detecting fraud highlights the model's effectiveness and the robustness of its predictive capabilities, tailored to understand and adapt to evolving fraudulent patterns over time.

Financially, the implementation of this model is projected to prevent approximately $47,064,000 in potential losses annually, underscoring its substantial impact on protecting our financial assets. This significant cost-saving is vital for maintaining a competitive advantage and operational efficiency in an industry where fraud tactics are constantly evolving.

In line with our strategic goals, the integration of this advanced machine learning model not only enhances our fraud detection processes but also sets a new standard for innovation within our operational practices. Moving forward, we recommend periodic updates to the model's training dataset and algorithms to keep pace with new fraud trends. Additionally, expanding the model’s application to cover more transaction types and integrating real-time detection capabilities will further solidify our position as a leader in financial security.

By continuously enhancing our technological capabilities to detect and prevent fraud, we reinforce our commitment to operational excellence and customer satisfaction. This project is a testament to our proactive approach in adopting cutting-edge technologies to tackle complex challenges in the financial sector. We remain dedicated to improving our defenses against fraud, ensuring that our systems evolve as rapidly as the methods used by those who seek to challenge them. This adaptive approach not only protects our customers but also ensures that our business remains resilient against threats to its financial security.

Chapter 2: Data Description

This dataset comprises 97,852 records of real credit card transactions conducted from January 1, 2010, to December 31, 2010. The dataset was originally curated for educational purposes from internal transaction logs, enabling a detailed investigation into consumer behavior and fraud detection strategies. The data features 10 fields, providing a broad range of transactional details, including numeric variables such as transaction amounts, as well as categorical fields such as merchant information and fraud indicators.

The dataset is well-suited for machine learning tasks involving classification, specifically fraud detection, as it contains both fraud (target) and non-fraud data points. This allows for the development of supervised learning models to predict fraudulent behavior based on various transaction attributes.

Key aspects of the data:

- **Time Frame**: Covers one full year (2010), allowing for temporal analysis and the capture of seasonal or periodic trends.
- **Fraud Proportion**: A binary fraud indicator is present, providing a clear target variable for predictive modeling.
- **High Cardinality**: Several fields, such as merchant numbers and zip codes, have high cardinality, which can influence feature engineering and modeling techniques.
- **Transaction Amount Outliers**: There are a small number of significant outliers in transaction amounts, necessitating careful handling during analysis to avoid skewing model predictions.

This dataset offers a unique opportunity to analyze patterns of fraudulent transactions using various machine learning techniques. The wide distribution of merchant locations, types, and transaction values also allows for geographic and behavioral segmentation, critical for understanding and preventing fraudulent activity in real-world applications.

Field Summary Tables:

Numeric Field Summary:

Field Name	Field Type	# Records Have Values	% Populated	# Zeros	Min	Max	Mean	Standard Deviation	Most Common
Amount	Numeric	97,852	100%	0	0	3,102,046	425.5	9,949.8	3.6

Categorical Field Summary:

Field Name	Field Type	# Records Have Values	% Populated	# Zeros	# Unique Values	Most Common
Cardnum	Categorical	97,852	100%	0	1,645	5142148452
Date	Categorical	97,852	100%	0	365	2/28/2010
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	38118
Transtype	Categorical	97,852	100%	0	4	P
Fraud	Categorical	97,852	100%	95,805	2	0

Key Transactional Insights and Fraud Analysis Visualizations:

1. Distribution of Transaction Amounts

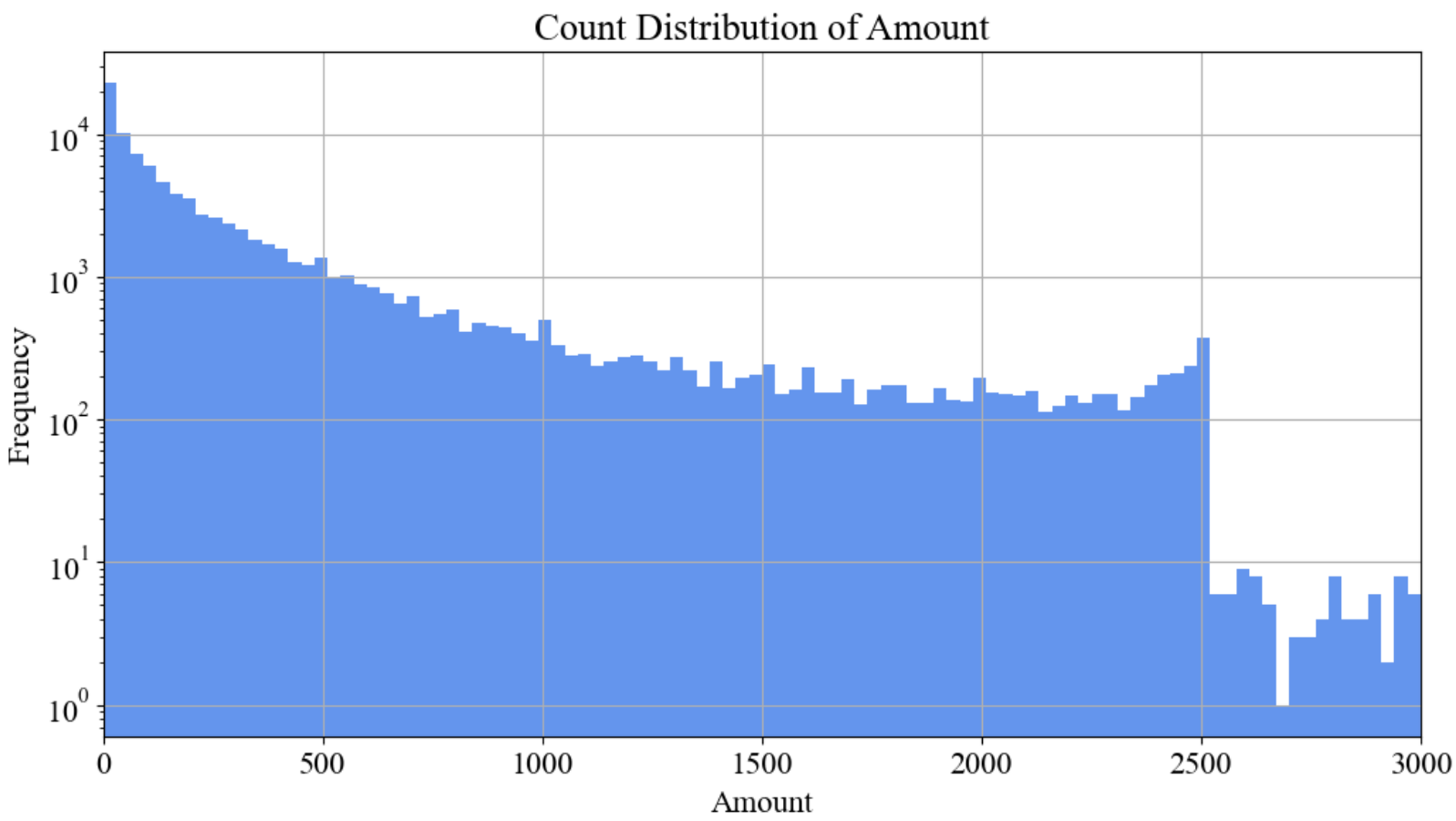


Fig 2.1

Description: This histogram illustrates the distribution of transaction amounts, focusing on amounts up to $3,000, which account for 99% of all transactions. Outliers beyond this threshold are excluded to enhance visualization clarity, particularly for detecting patterns among more common transaction values.

Insight: The majority of transactions are concentrated in lower amounts, with only a small fraction of high-value transactions. This pattern is typical for consumer spending behavior, where larger, more infrequent transactions could be linked to potential fraud cases. Identifying anomalies in this distribution could help in flagging suspicious activity.

2. Top 10 States by Merchant Transaction Volume

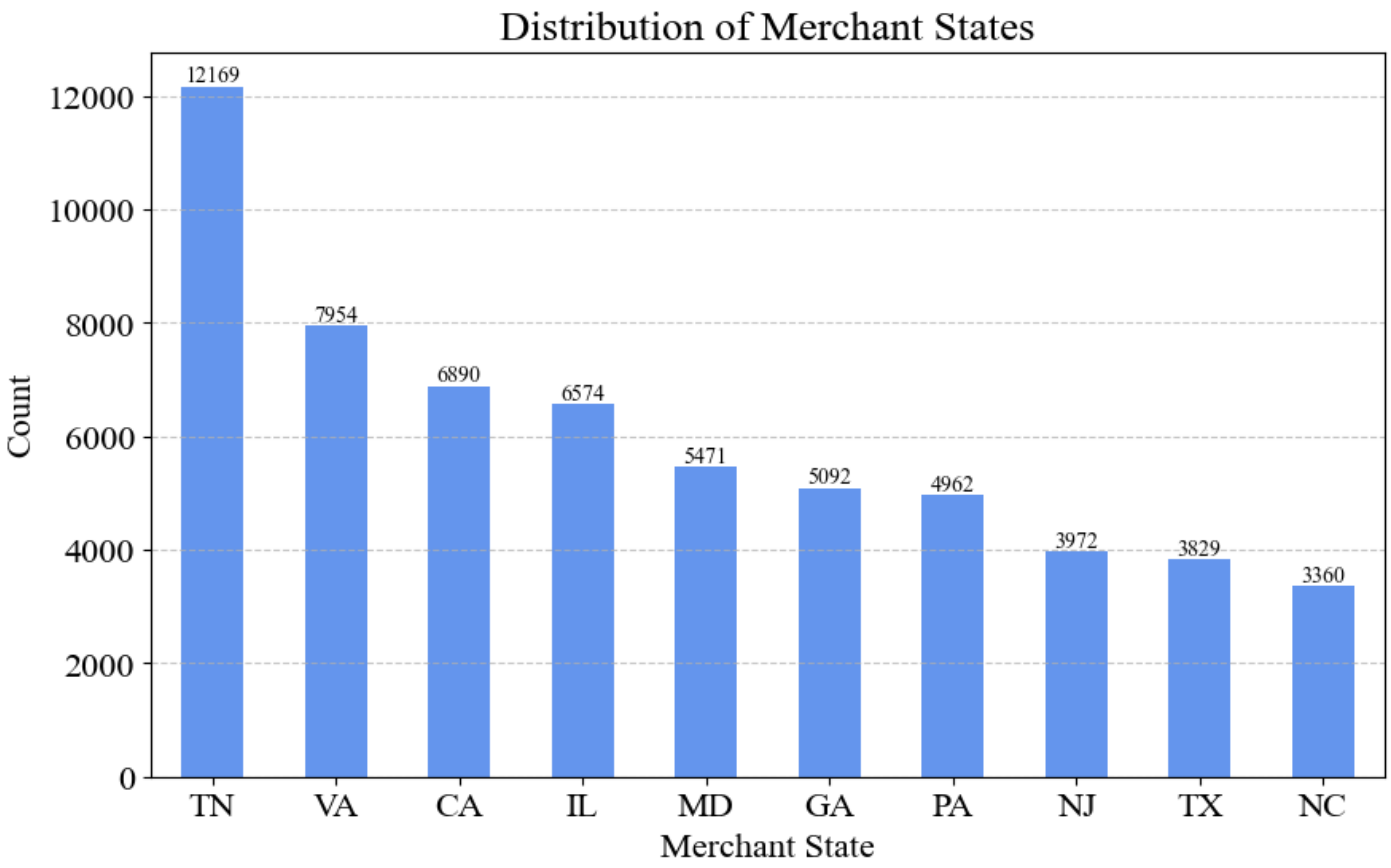


Fig 2.2

Description: A bar chart depicting the states with the highest volume of credit card transactions. This visualization provides insight into geographic trends and highlights potential regions with concentrated transaction activity, which may point to areas of interest for fraud analysis.

Insight: States like Tennessee (TN), Virginia (VA), and California (CA) lead in transaction volume. Regional fraud patterns may emerge from this data, as these states have significant economic activity. Understanding the regional concentration of transactions could help identify localized fraud hotspots and inform geographically targeted fraud prevention strategies.

3. Fraud vs. Non-Fraud Transaction Distribution

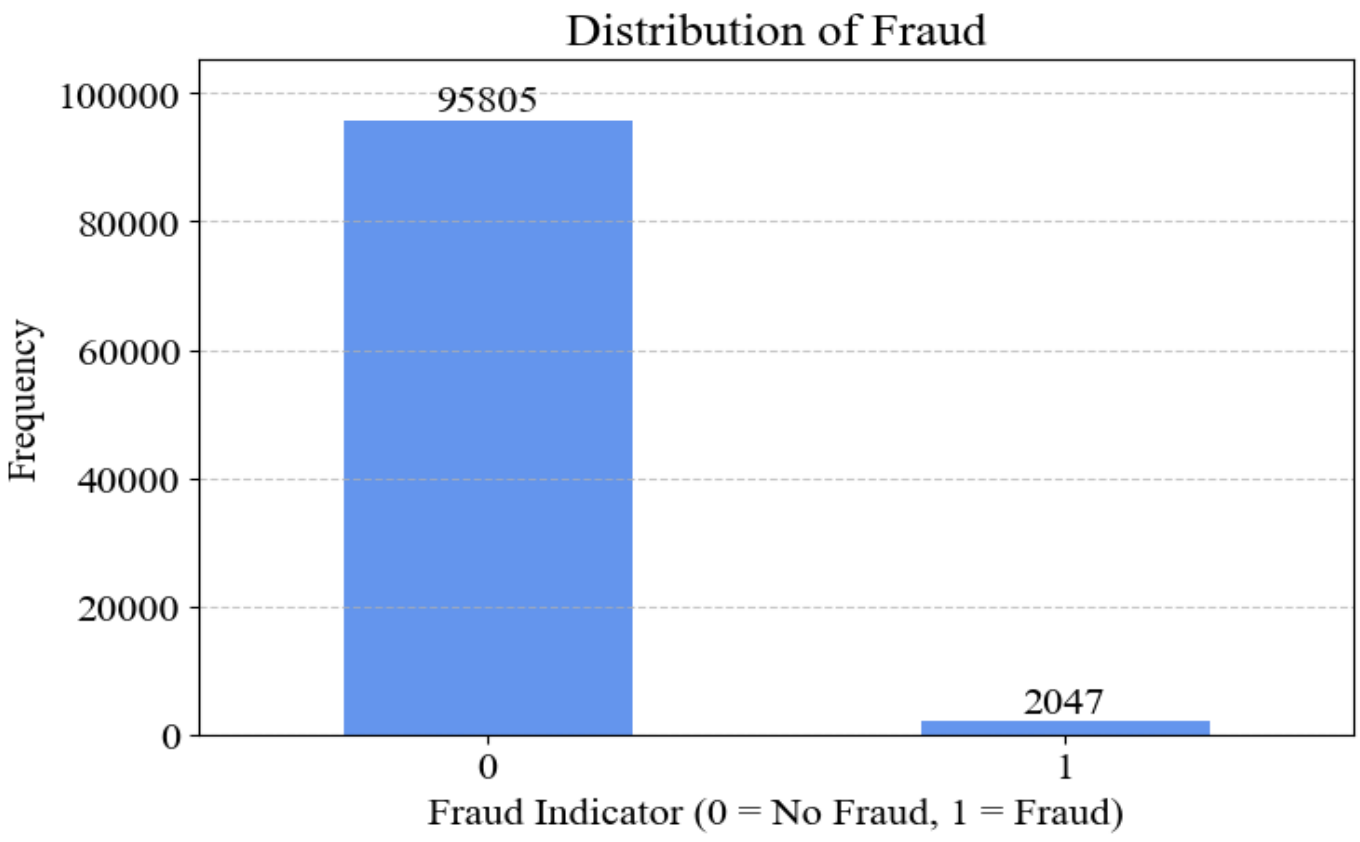


Fig 2.3

Description: This bar chart compares the count of fraudulent and non-fraudulent transactions, illustrating the significant imbalance in the dataset. Fraudulent transactions account for just over 2% of the data, with 2,047 fraud cases out of 97,852 total transactions, while the remaining 95,805 transactions are non-fraudulent.

Insight: The noticeable imbalance between fraudulent and non-fraudulent transactions requires special consideration during model training. Techniques like oversampling, under sampling, or cost-sensitive learning are essential to prevent the model from favoring non-fraudulent predictions. Properly addressing this imbalance is critical for improving the accuracy and sensitivity of fraud detection models, as models might otherwise struggle to identify the relatively few fraudulent transactions.

4. Daily Transaction Volume Throughout the Year

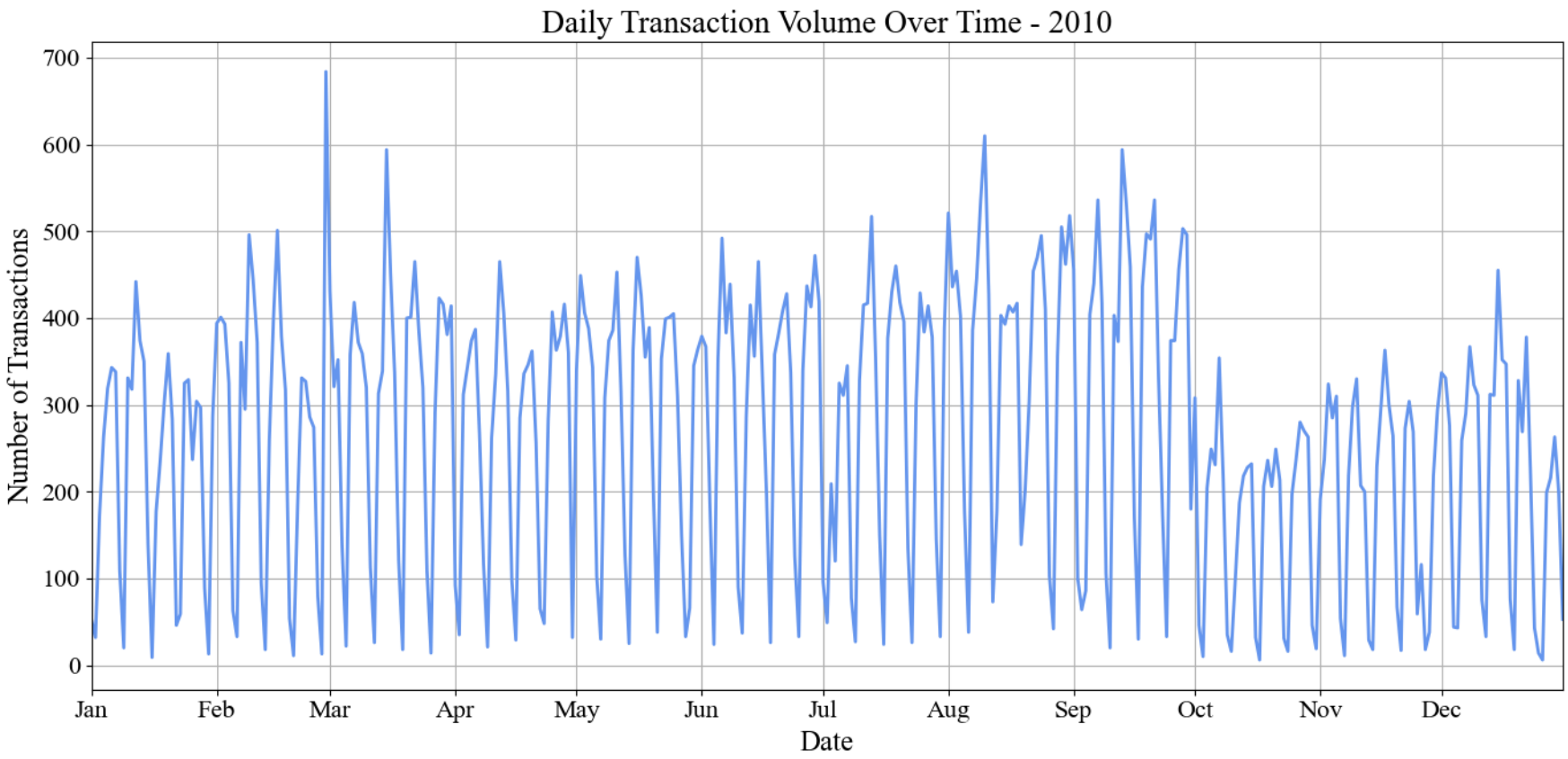


Fig 2.4

Description: The date plot shows the total number of transactions per day throughout 2010, revealing peaks and troughs in transaction activity. The time series visualization highlights variations in daily transaction volume, helping to identify seasonal patterns or irregularities, such as periods with significantly higher transaction volumes that may correspond with increased fraud risk.

Insight: Transaction volume fluctuates throughout the year, with notable peaks during specific periods. These spikes could be tied to key events or holidays, such as Black Friday and the holiday season, when spending tends to increase. These periods of high activity provide important context for fraud detection, as the surge in legitimate transactions may mask fraudulent activity. Understanding these seasonal trends is crucial for building models that can effectively account for and adapt to such fluctuations in transaction volume.

Chapter 3: Data Cleaning

This section outlines the data cleaning procedures applied to the credit card transaction dataset to ensure its quality and suitability for fraud detection analysis. The dataset initially comprised 97,852 records across 10 fields, including transaction amount, merchant information, and fraud indicators. The data cleaning process involved three critical steps: exclusions, outlier treatment, and imputation. These steps were implemented to maintain data integrity, reduce noise, and enhance the reliability of the analysis.

1. Exclusions

Exclusions were carefully applied to the dataset to remove data that did not align with the analysis criteria or had the potential to introduce bias or distortions in the results. The primary goal of these exclusions was to ensure that the dataset remained relevant, focused, and conducive to accurate analysis. The following specific exclusions were implemented.

1) Dropped Columns with Entirely Missing Values:

Columns that were completely filled with missing values (NaN) were removed from the dataset. This step reduces the dataset's dimensionality, making it more manageable and less memory-intensive. By eliminating columns that provide no useful information, the analysis becomes more focused on relevant fields, improving model performance and interpretability.

Implemented Code:

```
data.dropna(how='all', axis=1, inplace=True)
```

2) Transaction Type Filtering:

The dataset was filtered to retain only those transactions identified as "Purchase" which were indicated by a 'Transtype' value of 'P.' This exclusion ensured that only relevant records were included, removing any other transaction types (such as refunds or transfers) that were outside the scope of the intended analysis.

Implemented Code:

```
data = data[data['Transtype'] == 'P']
```

3) Exclusion of Extreme Values in 'Amount'

Transactions where the 'Amount' exceeded 3,000,000 were excluded from the dataset. These high-value records were deemed outliers that could disproportionately influence the analysis due to their extreme nature. This threshold was chosen based on domain knowledge, considering that such large amounts are rare and likely not representative of typical purchase behavior.

Implemented Code:

```
data = data[data['Amount'] <= 3000000]
```

4) Identifying and Excluding 'FEDEX' Transactions:

Transactions with a merchant description containing 'FEDEX' were identified and excluded from the dataset to align the analysis with its focus on consumer behavior. These transactions often represent business or logistics-related activities, which are outside the scope of this study. Removing these records helps ensure the dataset remains relevant to the study's objectives, reduces noise, and enhances the reliability of analyses related to typical consumer spending patterns.

Implemented Code:

```
dropfedex = bf[bf['amount_okay'] == False]
```

The exclusions including dropping unnecessary columns, filtering out non-purchase transactions, removing extreme values, and excluding business-related transactions result in a smaller, more focused dataset. This enhances data quality and relevance by aligning it with the study's objectives, concentrating solely on consumer behavior analysis. Consequently, the refined dataset reduces noise and improves the accuracy and interpretability of any analytical models, leading to more reliable insights.

2. Outlier Treatment

Outliers can significantly skew statistical analyses and degrade the performance of predictive models by introducing bias or distorting the underlying patterns in the data. To mitigate these effects, specific strategies were applied to identify and handle such extreme values.

1) Capping High-Value Transactions:

Transactions with amounts exceeding $3,000,000 were capped to prevent excessively large values from disproportionately influencing the overall statistics. This capping ensures that these rare, extreme transactions do not distort key measures such as mean or variance, allowing for a more accurate reflection of typical consumer behavior patterns.

Implemented Code:

```
data = data[data['Amount'] <= 3000000]
```

2) Flagging Suspicious Patterns:

Transactions with an amount around $3.62, particularly those associated with the merchant description 'FEDEX SHP,' were flagged as potentially anomalous. This step helps identify patterns that may not represent genuine consumer transactions, thereby preventing these outliers from skewing the analysis. Flagging such records ensures that the dataset focuses on typical consumer activities, rather than being biased by transactions that are likely related to specific business or logistical activities.

Implemented Code:

```
data['amount_okay'] = np.where((data['Amount'] >= 3.62) & (data['Amount'] <= 3.80) &
                               (data['Merch description'].str[:5] == "FEDEX"), False, True)
```

Result: The dataset is refined by capping extreme values and flagging potentially anomalous transactions, which reduces noise and ensures that analyses and model predictions are based on more representative data. This approach leads to more accurate insights into consumer spending behavior and enhances the reliability and performance of statistical models.

3. Imputation Process

The imputation process aims to fill in missing values in various fields with the most suitable estimated values to enhance the accuracy of data analysis. In this dataset, three key fields had missing values: Merchnum (Merchant Number), Merch state (Merchant State), and Merch zip (Merchant Zip Code). The imputation strategies applied to these fields ensure completeness and integrity, reducing inaccuracies that incomplete data might introduce and thereby leading to more reliable and meaningful insights.

a) Imputation for the `Merchnum` (Merchant Number) Field

Number of Records with Missing Values: 3,279

1) **Replace '0' Values:** Merchnum values recorded as '0' were replaced with NaN to mark them as missing, ensuring clarity between missing data and actual merchant numbers.

Implemented Code:

```
data['Merchnum'] = data['Merchnum'].replace({'0':np.nan})
```

2) Mapping from 'Merch description':

Filled in the Merchnum for 1,164 records by using the Merch description field to match with known Merchnum values. This step leverages existing relationships between merchant descriptions and merchant numbers to recover missing values.

Implemented Code:

```
merchdes_merchnum = {}
for index, merchdes in data[data['Merch description'].notnull()][data['Merchnum'].notnull()]['Merch description'].items():
    if pd.isnull(merchdes) == True:
        continue
    elif merchdes not in merchdes_merchnum:
        merchdes_merchnum[merchdes] = data.loc[index, 'Merchnum']

data['Merchnum'] = data['Merchnum'].fillna(data['Merch description'].map(merchdes_merchnum))
```

3) Assigning 'Unknown' for Adjustments:

For transactions where the Merch description field indicated credit or debit adjustments such as "RETAIL CREDIT ADJUSTMENT" or "RETAIL DEBIT ADJUSTMENT", the missing Merchnum values were set to 'unknown.' This approach helps clearly identify and separate these adjustment transactions from regular transactions.

Implemented Code:

```
data['Merchnum'] = data['Merchnum'].mask(data['Merch description'] == 'RETAIL CREDIT ADJUSTMENT', 'unknown')
data['Merchnum'] = data['Merchnum'].mask(data['Merch description'] == 'RETAIL DEBIT ADJUSTMENT', 'unknown')
```

4) Identified Remaining Missing Records:

Identified 1,421 records with missing Merchnum, including 515 unique Merch descriptions that did not match any existing merchant numbers.

Implemented Code:

```
data.loc[data.Merchnum.isna(), 'Merch description'].unique()[:20]
```

5) Creating New Merchant Numbers:

In cases where 'Merchnum' remained missing after the initial imputation, new merchant numbers were generated. This was done by incrementing from the highest existing merchant number, ensuring a unique identifier was created for each unique 'Merch description.' This step ensured that all records had a valid 'Merchnum,' maintaining the dataset's consistency.

Implemented Code:

```
merchnum_create = {}
max_merchnum = pd.to_numeric(data.Merchnum, errors='coerce').max()
for merch_desc in data.loc[data.Merchnum.isna(), 'Merch description'].unique():
  merchnum_create[merch_desc] = str(int(max_merchnum + 1))
  max_merchnum += 1

data['Merchnum'] = data['Merchnum'].fillna(data['Merch description'].map(merchnum_create))
```

Result: The imputation of Merchnum values resulted in a dataset where all missing merchant numbers were successfully filled. By leveraging existing relationships between merchant descriptions and numbers, this approach maximizes the use of available data to fill gaps, avoiding the arbitrary assignment of values. The creation of unique merchant numbers for descriptions without a matching Merchnum ensures that each merchant is distinctly represented, maintaining data integrity and enhancing the reliability of any merchant-level analyses. As a result, the dataset is now more complete, facilitating accurate consumer behavior modeling and other analytical purposes.

b) Imputation for the `Merch state` (Merchant State) Field

Number of Records with Missing Values: 1,028

1) Mapping from 'Merch zip':

Missing Merch state values were filled using a mapping from Merch zip to the corresponding state (zip_state dictionary), reducing missing values from 1,028 to 954.

Implemented Code:

```
zip_state = {}
for index, zip5 in data[data['Merch zip'].notnull()]['Merch zip'].items():
    if zip5 not in zip_state:
        zip_state[zip5] = data.loc[index, 'Merch state']

data['Merch state'] = data['Merch state'].fillna(data['Merch zip'].map(zip_state))
```

2) Mapping from 'Merchnum' and 'Merch description':

Imputation of missing Merch state values was further refined by using a two-step process. First, the Merchnum was mapped to the Merch state using a merchnum_state dictionary, which slightly reduced the number of missing values

from 954 to 953. Then, the remaining missing Merch state values were filled using a merchdes_state dictionary that maps Merch description to Merch state, further reducing the missing values from 953 to 952.

Implemented Code:

```
merchnum_state = {}
for index, merchnum in data[data['Merchnum'].notnull()]['Merchnum'].items():
    if merchnum not in merchnum_state:
        merchnum_state[merchnum] = data.loc[index, 'Merch state']

merchdes_state = {}
for index, merchdes in data[data['Merch description'].notnull()]['Merch description'].items():
    if merchdes not in merchdes_state:
        merchdes_state[merchdes] = data.loc[index, 'Merch state']

data['Merch state'] = data['Merch state'].fillna(data['Merchnum'].map(merchnum_state))
data['Merch state'] = data['Merch state'].fillna(data['Merch description'].map(merchdes_state))
```

3) Assign 'Unknown' for Adjustment Transactions and Handle Non-US States:

For records where the Merch description was either "RETAIL CREDIT ADJUSTMENT" or "RETAIL DEBIT ADJUSTMENT" the Merch state was set to "unknown" significantly reducing the number of missing values from 952 to 297. Additionally, any states not included in the list of U.S. states were identified as foreign transactions and labeled as "foreign" ensuring clarity for international transactions.

Implemented Code:

```
data['Merch state'] = data['Merch state'].mask(data['Merch description'] == 'RETAIL CREDIT ADJUSTMENT', 'unknown')
data['Merch state'] = data['Merch state'].mask(data['Merch description'] == 'RETAIL DEBIT ADJUSTMENT', 'unknown')
```

```
states = ["AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DC", "DE", "FL", "GA", "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME", "MD",
          "MA", "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH", "NJ", "NM", "NY", "NC", "ND", "OH", "OK", "OR", "PA", "RI", "SC",
          "SD", "TN", "TX", "UT", "VT", "VA", "WA", "WV", "WI", "WY", 'VI', 'PR', np.nan, 'unknown']

for index, state in data['Merch state'].items():
    if state not in states:
        data.loc[index, 'Merch state'] = 'foreign'
```

4) Assign 'Unknown' to Remaining Missing Values:

For the remaining 297 missing values, where Merch state could not be determined, the value "unknown" was assigned, ensuring every record has a value for Merch state.

Implemented Code:

```
data['Merch state'].fillna('unknown', inplace=True)
```

Result: The imputation of Merch state values led to a fully populated state field, using the most direct sources of information available, such as Merch zip and known merchant numbers. For cases where state-level information could not be derived, clear labels like "unknown" or "foreign" were applied, maintaining the transparency of the dataset. This strategy ensures that all transactions are correctly categorized for state-level analysis, enabling meaningful and reliable insights into geographic patterns and trends. The approach preserves the integrity and consistency of the data, ensuring that future analyses based on state information remain accurate and valid.

c) Imputation for the `Merch zip` (Merchant Zip Code) Field

Number of Records with Missing Values: 4,347

1) Mapping from 'Merchnum' and 'Merch description':

Missing Merch zip values were filled using mappings from Merchnum and Merch description to their respective zip codes. This reduced the number of missing values from 4,347 to 2,625.

Implemented Code:

```
merchnum_zip = {}
for index, merchnum in data[data['Merchnum'].notnull()]['Merchnum'].items():
    if merchnum not in merchnum_zip:
        merchnum_zip[merchnum] = data.loc[index, 'Merch zip']

merchdes_zip = {}
for index, merchdes in data[data['Merch description'].notnull()]['Merch description'].items():
    if merchdes not in merchdes_zip:
        merchdes_zip[merchdes] = data.loc[index, 'Merch zip']

data['Merch zip'] = data['Merch zip'].fillna(data['Merchnum'].map(merchnum_zip))
data['Merch zip'] = data['Merch zip'].fillna(data['Merch description'].map(merchdes_zip))
```

2) Assigning 'Unknown' for Adjustments:

Records with Merch description values like "RETAIL CREDIT ADJUSTMENT" and "RETAIL DEBIT ADJUSTMENT" were assigned the value "unknown" for Merch zip, further reducing missing values to 1,940.

Implemented Code:

```
data['Merch zip'] = data['Merch zip'].mask(data['Merch description'] == 'RETAIL CREDIT ADJUSTMENT', 'unknown')
data['Merch zip'] = data['Merch zip'].mask(data['Merch description'] == 'RETAIL DEBIT ADJUSTMENT', 'unknown')
```

3) Imputing with Most Populated Zip Code by State:

For records with a known Merch state but missing Merch zip, the most populated zip code for that state was used, reducing the missing values to 531.

Implemented Code:

```
mostPopZip = {'AL': 35205, 'AK': 99501, 'AZ': 85281, 'AR': 72701, 'CA': 90026,
              'CO': 80219, 'CT': 6511, 'DE': 19801, 'FL': 33139, 'GA': 30303,
              'HI': 96744, 'IL': 60647, 'IN': 46201, 'IA': 52404, 'KS': 66102,
              'KY': 40203, 'LA': 70119, 'ME': 4101, 'MD': 21224, 'MA': 2118,
              'MI': 48201, 'MN': 55406, 'MO': 63118, 'MT': 59101, 'NE': 68104,
              'NV': 89101, 'NH': 3103, 'NJ': 7030, 'NM': 87102, 'NY': 11226,
              'NC': 28202, 'ND': 58102, 'OH': 44113, 'OK': 74120, 'OR': 97214,
              'PA': 19107, 'RI': 2903, 'SC': 29203, 'SD': 57103, 'TN': 37203,
              'TX': 77002, 'UT': 84101, 'VT': 5401, 'VA': 22201, 'WA': 98122,
              'WV': 25301, 'WI': 53204, 'WY': 82001}

data['Merch zip'] = data['Merch zip'].fillna(data['Merch state'].map(mostPopZip))
```

4) Assign 'Unknown' to Remaining Missing Values:

For the remaining 531 records, where Merch zip could not be determined, the value "unknown" was assigned, ensuring all records have a non-null value for Merch zip.

Implemented Code:

```
data['Merch zip'].fillna('unknown', inplace=True)
```

The imputation of Merch zip values ensured that all records have a complete set of zip codes, whether they were directly derived from existing data or approximated based on state-level information. This comprehensive strategy maintains geographic coherence, ensuring that location-based analyses remain robust and accurate. Assigning "unknown" to remaining missing values further prevents misleading interpretations by clearly marking gaps in data.

Overall, the dataset is now better prepared for any geographical or regional analysis, enhancing its utility for various analytical models.

Suggestions for Improvement

1. Dynamic Outlier Detection Methods:

Rather than using a fixed threshold to cap transaction amounts (e.g., $3,000,000), a more adaptable approach would involve dynamic outlier detection methods, such as the Interquartile Range (IQR) or Z-score. These methods adjust according to the underlying distribution of the data, allowing for a more data-driven identification of anomalies. For example, the IQR method flags values that fall beyond a certain range (typically 1.5 times the IQR above the upper quartile or below the lower quartile) as outliers. Z-scores identify outliers by measuring how many standard deviations a data point is from the mean. By applying these techniques, you can more precisely capture outliers that are anomalous without arbitrarily excluding high-value transactions that may still be relevant to the analysis.

2. Advanced Imputation Techniques:

To enhance the accuracy and reliability of imputed data, advanced machine learning models, such as K-Nearest Neighbors (KNN), regression-based methods, or even decision trees, could be used for predicting missing values. These techniques use patterns in the available data to make informed estimates for missing entries. For instance, KNN identifies similar records in the dataset and fills missing values based on the nearest neighbours' attributes, making the imputation process more sophisticated and context-sensitive. This is particularly beneficial when there are complex relationships between fields, such as merchant descriptions, amounts, and geographic locations, that simpler imputation methods might overlook.

3. Optimize Code with Vectorized Operations:

Instead of relying on manual loops for data processing, using vectorized operations in libraries like pandas can significantly improve code efficiency, especially when working with large datasets. Vectorized operations apply functions directly to entire arrays or columns, allowing faster execution compared to iterating through each row. This not only reduces the time taken for data cleaning but also makes the code more scalable and maintainable. For example, tasks like imputing missing values or capping outliers can be done more efficiently with pandas' built-in functions such as apply(), fillna(), or clip().

The data cleaning procedures outlined in this section, including exclusions, outlier treatment, and imputation, effectively enhanced the dataset's quality and suitability for analysis. By removing irrelevant records, managing outliers, and imputing missing values, the dataset was refined to focus solely on consumer behavior patterns, thereby reducing noise and improving data accuracy. The steps taken ensure the integrity and completeness of the data, enabling more reliable and meaningful insights for subsequent analyses. Implementing the suggested improvements, such as dynamic outlier detection, advanced imputation techniques, and optimized code practices, could further elevate the quality and efficiency of the data preparation process, ensuring robust and accurate analytical outcomes.

Chapter 4: Variable Creation

The initial phase of variable creation involved an in-depth analysis of the original dataset, meticulously examining baseline variables such as transaction IDs, transaction amounts, and merchant details. Our objective was to extract and amplify underlying patterns that could potentiate the predictive accuracy of our fraud detection models. To this end, we employed advanced feature engineering techniques, harnessing the power of analytics to unlock new dimensions of data interpretation.

Fundamental Entities for Variable Generation

The foundational entities selected for feature engineering were pivotal in deriving a broad spectrum of analytical variables, each adding a layer of insight into transaction dynamics. These entities include:

- **Cardnum**: Represents the unique card number associated with each transaction. This identifier is crucial for tracking and analyzing spending patterns specific to each cardholder.
- **Merchnum**: Stands for the unique identifier of the merchant where the transaction occurred, facilitating the examination of merchant-specific transaction patterns and potential fraud vulnerabilities.
- **Merch zip**: The ZIP code of the merchant's location is used to generate geographic insights, which are instrumental in identifying regions with high incidences of fraud.
- **Merch state**: Similar to ZIP codes, the state information helps in creating geographic and demographic profiles, enriching the context for more localized fraud detection strategies.
- **Transaction Amount**: The monetary value of each transaction is analyzed not just in isolation but also in aggregated forms across various temporal windows to detect anomalies and patterns indicative of fraudulent activities.
- **Date**: The transaction date is critical for constructing features that describe behavioral trends over time, such as frequency of transactions and cyclic spending behaviors.
- **Fraud Indicator**: A binary variable that flags transactions as fraudulent or legitimate, forming the cornerstone of our training dataset for supervised learning models.

Enhanced Variable Creation Techniques

With these core entities as a foundation, our team developed a comprehensive set of variables that deepen the analytical capacity of our models:

1. **Time-window Based Features**: By analyzing transaction data across different time frames (e.g., daily, weekly, monthly), we were able to construct features that capture the frequency of transactions and the time elapsed since the last transaction, which are indicative of typical or atypical user behavior.
2. **Geographical and Demographic Features**: We combined merchant location data (ZIP codes and states) with transaction details to uncover regional trends and to perform target encoding, which transforms these categorical fields into numerical scores that reflect the probability of fraud in different locales.

3. **Financial Behavioral Features**: Transaction amounts were dissected into various constructs such as average transaction value per card or merchant, variability in transaction amounts, and ratios of amounts over specified time frames. These features help in identifying outliers and patterns that deviate from a cardholder's typical transaction profile.
4. **Sequence and Recency Features**: These features focus on the sequencing of transactions and the recency of particular transaction types, offering insights into the urgency and irregularity of transactions, which are often red flags for fraudulent activities.
5. **Encoded Categorical Features**: Using techniques like one-hot encoding and target encoding, categorical variables such as merchant categories and transaction types were transformed into a format suitable for machine learning models, ensuring that no valuable information is lost in translation from qualitative to quantitative data.

Understanding Fraudulent Behaviors: To enhance our fraud detection models, it is imperative to understand the various tactics and behaviors commonly associated with fraudulent transactions. These include, but are not limited to:

- **Unusual Transaction Times**: Transactions that occur at odd hours, such as late at night or early in the morning, which may be unusual for the cardholder's typical activity pattern.
- **High-Risk Geographic Locations**: Certain ZIP codes or states may have higher incidences of reported frauds, making transactions from these areas potentially suspicious.
- **Rapid Succession of High-Value Transactions**: Sudden spikes in transaction activity, especially of high value, could indicate a compromised card.
- **Irregular Transaction Patterns**: Deviations from a cardholder's normal spending pattern, such as abrupt changes in transaction frequency or amount, can suggest unauthorized access.
- **Merchant Type Anomalies**: Transactions at merchants that do not align with the cardholder's regular spending habits or at merchants known for higher risks of fraud.

These behaviors are integrated into our analytical framework by creating variables that can detect such anomalies. For instance, time-window based features help identify unusual transaction times, while geographical features take into account the risk level associated with specific locations. By combining insights from these behaviors with our robust variable creation strategies, our models are better equipped to flag potential fraud accurately and efficiently. This strategic approach to data analysis ensures that we are not only reacting to known fraud patterns but also proactively identifying new trends and tactics in fraudster behavior.

The comprehensive variable creation process outlined not only enriched our dataset but also strengthened the predictive power of our models, providing a robust platform for detecting and preventing credit card fraud effectively. This strategic approach to data analysis ensures that we are not only reacting to known fraud patterns but also proactively identifying new trends and tactics in fraudster behavior.

Summary of Created Variables:

Variable Category	Description	# Variables	Examples
Original Variables	Core variables from the dataset including identifiers and transaction details.	10	Recnum, Cardnum, Date, Merchnum, Amount, Fraud
Day of Week Variables	Variables representing the day of the week and its associated transaction risk.	2	Dow, Dow_Risk
Entity Combination Variables	Variables derived from combinations of attributes such as card numbers, merchant numbers, state, and zip codes.	21	card_merch, card_zip, card_state
Target Encoding Variables	Variables that use target encoding techniques on attributes such as merchant state, zip code, and day of the week to capture categorical effects.	3	Merch state_TE, Merch zip_TE, Dow_TE
Benford's Law Variables	Variables related to Benford's law analysis, initially tested on Card_num and Merch_num, but later removed after analysis. (8 variables * 2 entities)	16	n_high, n_low, R, 1/R, U, n, t, U_smoothed, Cardnum_U, Merchnum_U, Cardnum_count, Merchdesc_count, Cardnum_bin, Merchdesc_bin
Day-Since and Frequency Variables	Metrics related to the elapsed time since the last transaction and the frequency of transactions over specific time windows: {0, 1, 3, 7, 14, 30, 60 days}. Total : (24 variables per entity * 23 entities * 7 time periods)	1472	Cardnum_day_since, Cardnum_count_0, Merchnum_avg_1
Count and Total Ratios	Ratios depicting the relationship between count and total amount of transactions within specific time windows: {0, 1, 3, 7, 14, 30, 60 days}. Calculated for 8 ratios (count,	368	Cardnum_count_0_by_7, Cardnum_total_amount_0_by_14

	total amount) for 23 entities over 4 comparison periods.		
Velocity Ratios	Variables quantifying the velocity of transactions by comparing counts relative to the elapsed time since the last transaction for specified time windows: {0, 1, 3, 7, 14, 30, 60 days}. Total of 8 ratios per entity for 23 entities across 4 periods.	184	Cardnum_vdratio_0by7, Merchnum_vdratio_1by30
Variability-Related Variables	Measures of variability such as mean, maximum, and median differences in transaction amounts across different entities over specified time windows: {0, 1, 3, 7, 14, 30, 60 days}. Calculated for 18 variables per entity across 23 entities	414	Cardnum_variability_avg_0, Merchnum_variability_max_7
Basic Entity Combinations (entities1)	Entity combinations generated from basic transaction attributes such as Cardnum and Merchnum, calculated over specific time windows: {1, 3, 7, 14, 30, 60 days}.	120	card_merch_unique_count_for_card_zip_7, Cardnum_count_by_Merchnum_3
Geographic Entity Combinations (entities2)	Variables combining geographic data such as merchant zip and state with transaction details over specific time frames: {1, 3, 7, 14, 30, 60 days}.	120	Merch_zip_count_14, Merch_state_avg_30
Descriptive Entity Combinations (entities3)	Variables involving combinations of merchant descriptions, transaction details, and the day of the week, calculated across specific time windows: {1, 3, 7, 14, 30, 60 days}.	120	Merchnum_desc_avg_7, Merchdesc_dow_max_3
Advanced Multi-attribute Combinations (entities4)	Complex variables integrating multiple attributes like card details, merchant descriptions, and geographic data, analyzed over specific time windows: {1, 3, 7, 14, 30, 60 days}.	336	Card_Merchdesc_State_avg_7, Merchnum_desc_State_count_14

Squared Ratios	Variables representing squared ratios of counts over multiple time windows: {0, 1, 3, 7, 14, 30, 60 days} for various entities. This provides a refined measure of comparison between time periods. 8 variables per entity across 23 entities. (8 variables * 23 entities * 4 time periods)	184	Cardnum_count_0_by_7_sq, Merchnum_count_1_by_30_sq
Binned Amounts	Transaction amounts categorized into quintiles.	1	amount_cat
Foreign Zipcode Check	Indicator checking if a merchant's zip code is outside the U.S.	1	foreign

Total Variable Count: 3356

Reduction and Refinement Process:

The dataset was initially robust, featuring a comprehensive set of **3356 variables**, derived from both the original data attributes and a multitude of engineered features designed to expose deeper insights into the patterns inherent in the transaction data. This vast number of variables was intended to maximize the dataset's predictive power by capturing as many aspects of transactional behavior and anomaly detection as possible.

To enhance the dataset’s analytical efficiency and maintain a high level of model performance, a meticulous process of variable reduction was undertaken. The aim was to streamline the feature set to ensure computational efficiency and improve model training times without compromising the integrity and predictive quality of the models. This was achieved through the following steps:

1. **Identification of Redundant Columns:** The first step involved the systematic review of all variables to identify and eliminate duplicates and near-duplicates that added unnecessary complexity to the dataset. This included variables that were highly correlated or provided overlapping information with other features in the dataset.
2. **Refining Entity Combination Variables:** Many of the variables were created by combining different entity attributes, such as Cardnum and Merchnum, or transaction amounts with geographic data. While these combinations can provide valuable insights, they can also introduce a high degree of redundancy and multicollinearity. We refined these combinations, simplifying complex interactions into more interpretable and impactful features, thereby reducing redundancy and enhancing model interpretability.
3. **Elimination of Obsolete Features:** As models evolved and new insights were gained, certain features that initially seemed valuable became less relevant. These obsolete features were removed from the dataset. This step was critical in keeping the dataset lean and focused, particularly in terms of adapting to emerging fraud trends and eliminating noise.

4. **Quality Control and Validation:** Each step of the reduction process was carefully validated to ensure that the removal of variables did not adversely affect the dataset's predictive capability. Statistical methods and machine learning models were employed to test the impact of each reduction on model performance, ensuring that only beneficial reductions were finalized.
5. **Final Variable Set:** The outcome of this rigorous process was a refined dataset comprising **2634 variables**. This set represented the most meaningful, informative, and statistically relevant features necessary for effective fraud detection. Therefore, the total number of new variables left after the removal of redundant columns would be **2,634** (total remaining columns) - **12** (original columns), resulting in **2,622** new variables.

In the section of this project, a set of newly created variables was introduced to enhance the predictive power of the fraud detection model, though they were not further analyzed in this particular study. These variables include:

1. Time of Day Category: This variable categorizes transactions based on the time of day, recognizing that transactions conducted during late night or early morning hours may carry a higher risk of fraud. This temporal categorization aims to capture unusual activity patterns that deviate from typical consumer behavior.
2. Transaction Amount Category: By segmenting transaction amounts into predefined categories, this variable helps in identifying potential fraud patterns as well as normal spending behaviors. It is particularly useful in distinguishing between ordinary and suspiciously large transactions that could indicate fraudulent activity.
3. Merchant Risk Level: This categorical variable assigns risk levels to merchants based on the industry and known fraud statistics, libelling some as high-risk. This allows the model to weigh transactions involving these merchants more heavily, providing a nuanced approach to fraud detection that accounts for the variability in risk across different merchant categories.

Summary of Newly Created Variables:

Variable Category	Description	# Variables
Time of Day Category	Categorizes transactions based on the time of day, useful for fraud detection, as transactions during late night or early morning may be more risky.	1
Transaction Amount Category	Bins the transaction amounts into categories based on predefined thresholds. Helps identify fraud patterns or legitimate spending behavior.	1
Merchant Risk Level	A categorical variable assigning risk levels to merchants based on the industry, such as high-risk merchants	1

These variables were developed to provide deeper insights into transactional data, aiding in the robust detection of fraud by highlighting critical aspects of transaction timing, amount, and merchant risk. However, their exclusion from further analysis in this project leaves room for future exploration and integration into more comprehensive fraud detection strategies.

Chapter 5: Feature Selection

In this section, we explore feature selection techniques using the Light Gradient Boosting Machine (LGBM) and Random Forest (RF) models to optimize the predictive performance of a fraud detection algorithm. Feature selection is a crucial step in the machine learning pipeline as it enhances model interpretability, reduces overfitting, and decreases computational costs by eliminating irrelevant or redundant variables.

The objective is to demonstrate a systematic approach to selecting the most significant features from a dataset composed of various entities and time windows. We employ both forward and backward selection methods under varying configurations to identify a robust set of features that contribute to an effective fraud detection model.

Methodology:

Data Description:

The dataset under consideration is composed of transactional data, which includes a myriad of features related to card and merchant details. These features are aggregated across different temporal windows, providing a comprehensive view of transaction dynamics. The binary nature of our target variable, which flags transactions as either fraudulent or legitimate, sets the stage for a classification problem where the goal is to predict fraudulent instances accurately.

Feature Selection Setup:

1. **Filter Method:** The feature selection process commences with a filtering phase where individual features are evaluated for their discriminative power between the classes of interest - fraudulent and non-fraudulent transactions. This evaluation relies on the Kolmogorov-Smirnov (KS) statistic, a robust method that measures the maximum divergence between the cumulative distribution functions of two datasets. Features that exhibit the highest KS scores are deemed to have significant predictive power and are retained for further analysis. The number of features kept through this filter is determined by the num_filter parameter, allowing us to control the complexity and focus of the subsequent selection stages.
2. **Wrapper Method:** Post-filter, we apply a more nuanced wrapper method that uses either LGBM or RF to evaluate subsets of features based on their collective performance impact. This stepwise selection process, whether forward or backward, helps in understanding how additional features influence the overall efficacy of the model. Critical to this phase are the parameters num_wrapper, which dictates the number of features to include, and n_jobs for parallelizing the computation to increase efficiency.
3. **Experiment Configurations:** To adequately assess the model under various scenarios, we run multiple experiments where num_filter ranges from 100 to 300, and num_wrapper extends up to 20. This setup allows us to test the model's resilience and adaptability across different feature sets. LGBM's deterministic nature ensures consistency in feature selection across runs, providing a stable baseline for comparison. Conversely, the stochastic nature of RF introduces variability, offering insight into the robustness of features under different sampling conditions.

4. **Selection Strategy:** Each experimental run is meticulously analyzed through saturation plots that graph model performance against the number of features used. These plots are instrumental in identifying the saturation point—the stage at which additional features cease to provide substantial performance gains. Selecting features up to this point ensures an optimal balance between model complexity and performance, focusing on a set that offers comprehensive insights across various entity types and temporal dimensions.

Tools and Libraries Used: The implementation leverages Python, utilizing libraries such as Pandas for data manipulation, Matplotlib for visualization, and Scikit-learn with MLXtend for applying sophisticated feature selection techniques. These tools are chosen for their robustness and flexibility, allowing for detailed exploratory data analysis, visual representation of complex concepts, and efficient execution of machine learning pipelines.

By elaborating on the methodologies and analytical techniques used, this section provides a thorough understanding of the critical role that feature selection plays in enhancing the predictive performance of fraud detection models, paving the way for more focused and insightful data-driven decisions in combating fraud.

Forward Selection: LGBM

1) num_filter = 100, num_wrapper = 20

variable	filter score
Cardnum_unique_count_for_card_state_1	0.476067
Card_Merchdesc_State_total_7	0.324668
Cardnum_count_1_by_30	0.428229
Cardnum_max_14	0.318826
Card_dow_vdratio_0by60	0.486480
Card_dow_vdratio_0by14	0.479086
Merchnum_desc_State_total_3	0.308586
Card_Merchdesc_total_7	0.324631
Card_dow_unique_count_for_merch_zip_7	0.418943
merch_state_total_1	0.304893
Card_Merchnum_State_total_60	0.305385
Card_dow_vdratio_0by30	0.489227
Cardnum_unique_count_for_card_zip_3	0.464323
Merchnum_desc_State_total_1	0.304969
Cardnum_total_14	0.494375
Card_dow_count_7	0.482384
Cardnum_actual/toal_0	0.479550
Card_dow_total_30	0.474759
Card_dow_vdratio_0by7	0.467961
Cardnum_vdratio_1by7	0.466766

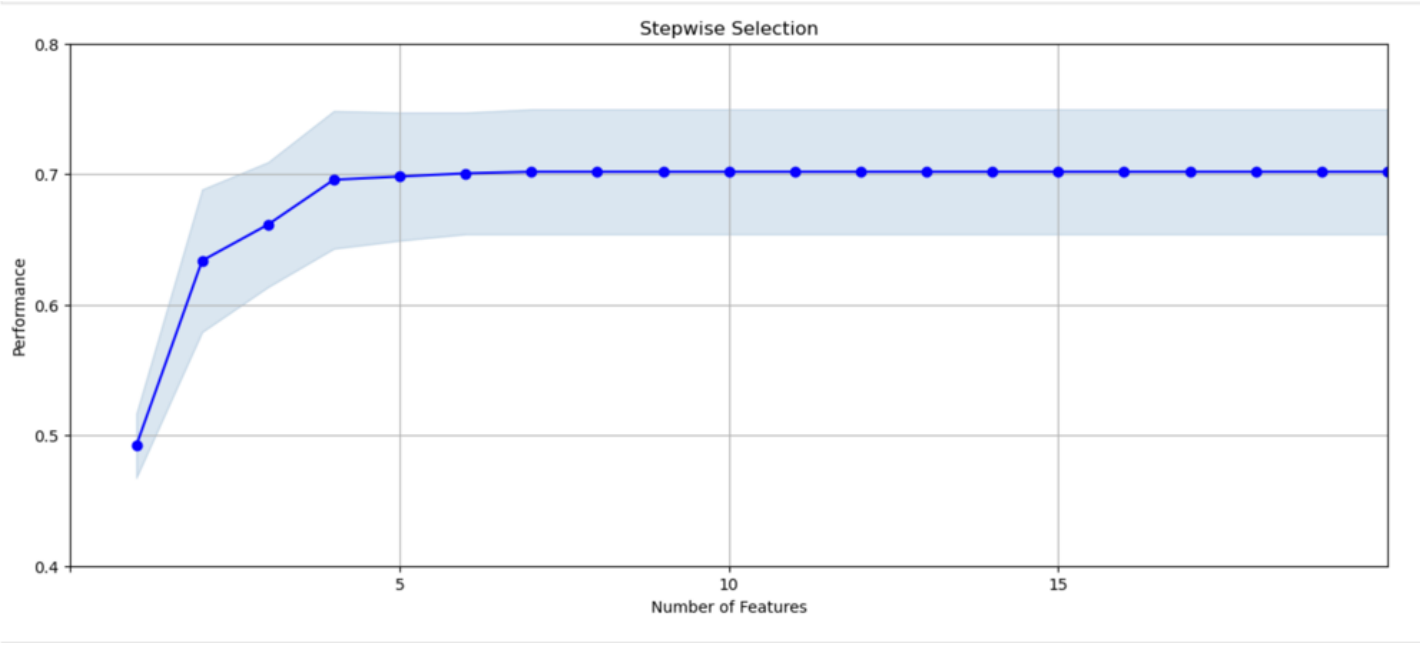


Fig 5.1

The stepwise selection plot from the LGBM forward selection experiment with `num_filter = 100` and `num_wrapper = 20` reveals a distinct performance pattern. Initially, there is a pronounced surge in performance as the first few features are added, highlighting their substantial informational value and their significant impact on the model's predictive capabilities. However, as more features are introduced beyond the first five, the performance improvement plateaus, suggesting that subsequent features contribute diminishing returns to model accuracy. This

trend continues with no substantial gains observed past the addition of the first 10 features, indicating that fewer but more impactful features may suffice for optimal performance.

A detailed examination of the types and roles of the variables selected during the experiment shows a mix of frequency variables, descriptive state variables, and day-of-week ratios, which collectively aid in detecting potential fraudulent activities by capturing transaction frequencies, geographical and merchant-specific trends, and temporal patterns. Variables such as `Cardnum_unique_count_for_card_state_1` and `Card_dow_vdratio_0by60` emerge as particularly influential, offering strong discrimination between fraudulent and non-fraudulent transactions. Given the performance plateau observed, it seems pragmatic to focus on a condensed set of high-impact features to enhance model efficiency and reduce complexity. This strategy not only prevents overfitting but also ensures the model remains computationally efficient, necessitating an investigation into the potential redundancy among the selected features to streamline the feature set further.

2) num_filter = 200, num_wrapper = 20

variable	filter score
Cardnum_unique_count_for_card_state_1	0.476067
Card_Merchdesc_State_total_7	0.324668
Cardnum_count_1_by_30	0.428229
Cardnum_max_14	0.318826
Card_dow_vdratio_0by60	0.486480
Card_dow_vdratio_0by14	0.479086
Merchnum_desc_State_total_3	0.308586
Card_Merchdesc_total_7	0.324631
Card_dow_unique_count_for_merch_zip_7	0.418943
Cardnum_actual/toal_0	0.479550
Card_dow_vdratio_0by7	0.467961
Cardnum_vdratio_1by7	0.466766
Cardnum_unique_count_for_card_state_3	0.466410
Cardnum_unique_count_for_card_zip_3	0.464323
Merchnum_desc_Zip_total_3	0.305656
Cardnum_unique_count_for_Merchnum_3	0.460748
Cardnum_actual/toal_1	0.459715
Cardnum_unique_count_for_card_state_7	0.445967
Cardnum_actual/max_0	0.445726
Card_dow_unique_count_for_merch_state_1	0.447357

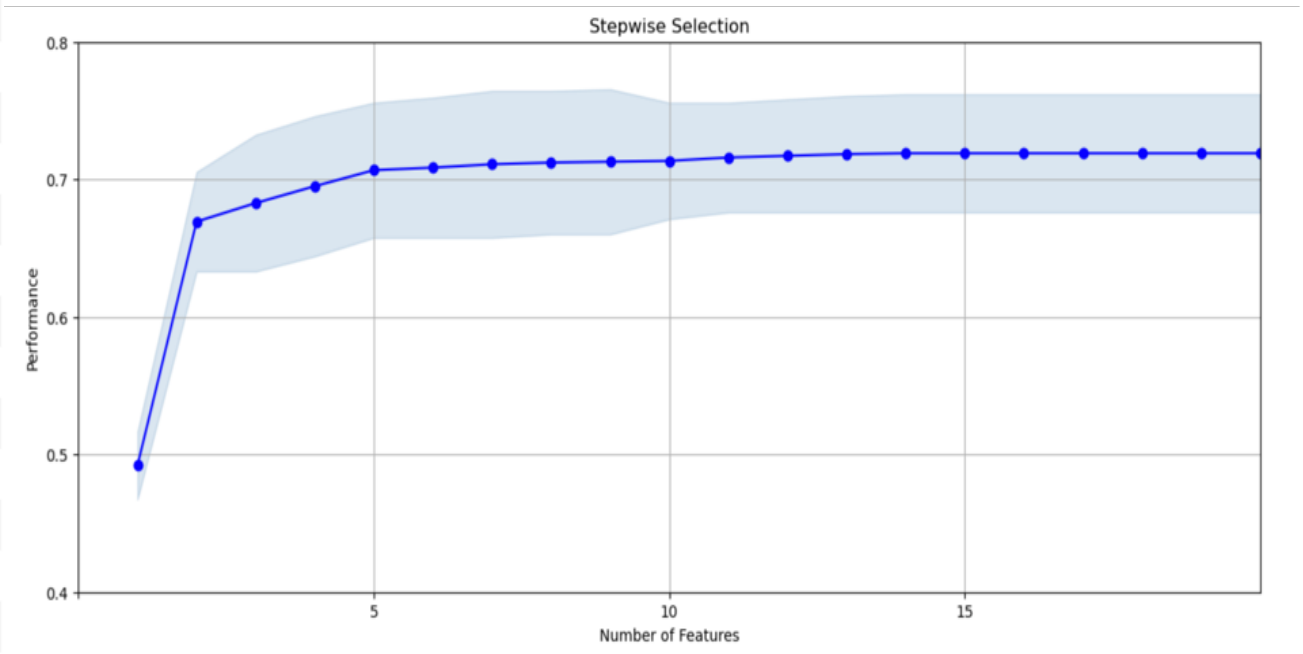


Fig 5.2

The analysis of the feature selection experiment using LGBM with `num_filter = 200` and `num_wrapper = 20` demonstrates a clear trend in model performance. Initially, there is a significant increase in predictive performance as the first few features are added, indicating these features are critical in distinguishing between fraudulent and non-fraudulent transactions. However, after the inclusion of approximately five features, the model's performance begins to plateau, suggesting that subsequent features provide diminishing returns. This early stabilization points to the high information content of the initial features, which are primarily transaction frequency variables and geographic/temporal descriptors, emphasizing their importance in the detection of fraud.

Further analysis into the types of variables selected shows a mix of transaction frequency variables like `Cardnum_count_1_by_30`, and geographic and merchant-specific descriptors such as `Merchnum_desc_State_total_3`. These findings suggest that localized spending behaviors and transaction patterns play crucial roles in predicting fraudulent activities. Given the plateau in performance with additional features, focusing on a smaller set of high-impact features could enhance the model's efficiency and reduce complexity. This streamlined approach would not only prevent overfitting but also improve the model's operational effectiveness, particularly in real-time fraud detection scenarios.

3) num_filter = 300, num_wrapper = 20

variable	filter score
Cardnum_unique_count_for_card_state_1	0.476067
card_state_total_3	0.368079
Cardnum_max_7	0.410589
Card_dow_vdratio_0by60	0.486480
Cardnum_max_60	0.345466
Card_dow_max_60	0.383395
Cardnum_vdratio_0by60	0.427651
Cardnum_total_14	0.494375
Cardnum_actual/toal_0	0.479550
Cardnum_unique_count_for_Merchnum_1	0.472017
Card_dow_vdratio_0by7	0.467961
Cardnum_unique_count_for_card_state_3	0.466410
Cardnum_unique_count_for_card_zip_3	0.464323
Cardnum_unique_count_for_Merchnum_3	0.460748
Cardnum_actual/toal_1	0.459715
Cardnum_unique_count_for_card_state_7	0.445967
Cardnum_actual/max_0	0.445726
Card_dow_unique_count_for_merch_state_1	0.447357
Cardnum_count_14	0.445443
Card_dow_count_14	0.443405

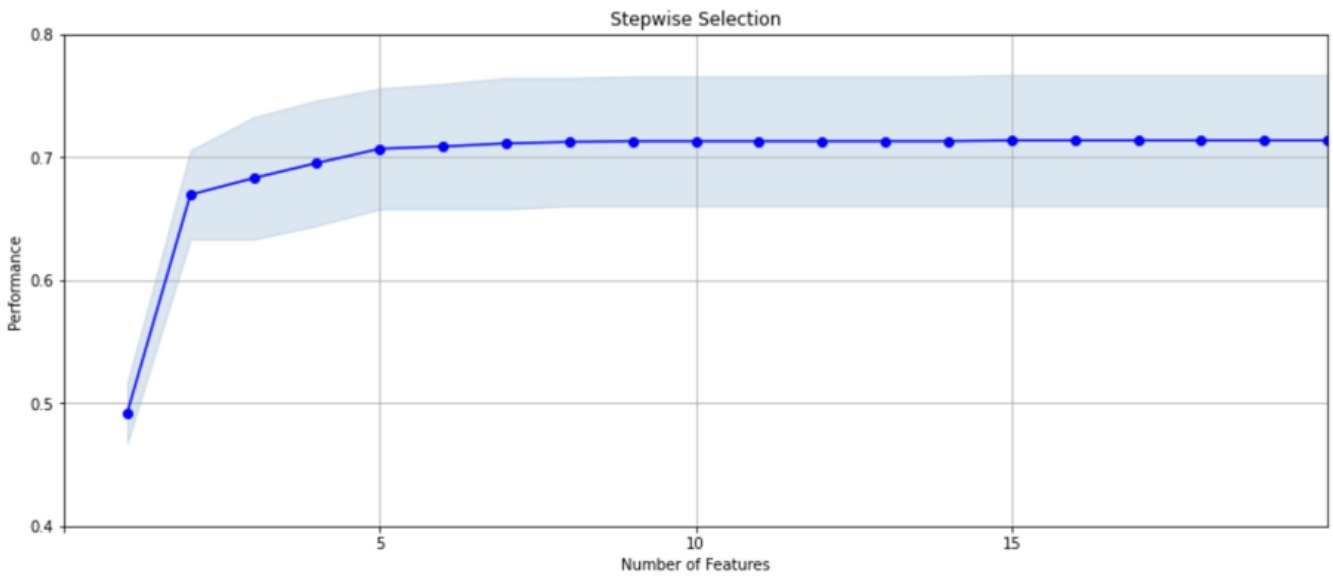


Fig 5.3

The feature selection analysis using LGBM with `num_filter = 300` and `num_wrapper = 20` underscores a significant trend where the addition of the first few features markedly improves model performance, stabilizing around a performance score of 0.7. This sharp increase followed by a plateau illustrates the high impact of the initial variables, which are critical for enhancing the model's predictive capabilities. Beyond these, subsequent features offer diminishing returns, indicating that a core subset of variables captures the essential patterns needed for effective fraud detection.

The selected variables primarily include transaction frequency and temporal behavior indicators, which underscore the model's reliance on detecting unusual transaction patterns and timings. Such variables are pivotal for identifying potential fraudulent activities, reflecting the importance of focused feature selection. This strategic approach not only streamlines the model by concentrating on the most informative features but also optimizes computational efficiency and model interpretability in complex tasks like fraud detection.

Forward Selection: RF

1) num_filter = 200, num_wrapper = 20

variable	filter score
Cardnum_unique_count_for_card_zip_1	0.474960
Card_Merchnum_desc_total_14	0.329261
Card_dow_unique_count_for_state_des_1	0.447238
card_state_max_14	0.305946
Card_dow_count_60	0.320904
Card_Merchdesc_State_total_14	0.332077
card_state_total_30	0.307908
merch_zip_total_3	0.310139
card_zip_total_7	0.325795
Cardnum_vdratio_1by30	0.504966
Card_Merchnum_Zip_total_1	0.315209
Card_Merchdesc_Zip_total_3	0.319124
Cardnum_count_3	0.563356
Cardnum_total_amount_0_by_60	0.376248
Cardnum_count_1_by_7	0.311269
card_merch_total_7	0.320524
Merchdesc_Zip_total_3	0.313324
Card_Merchdesc_total_3	0.319675
card_merch_total_60	0.305373
card_state_count_1_by_60	0.309137

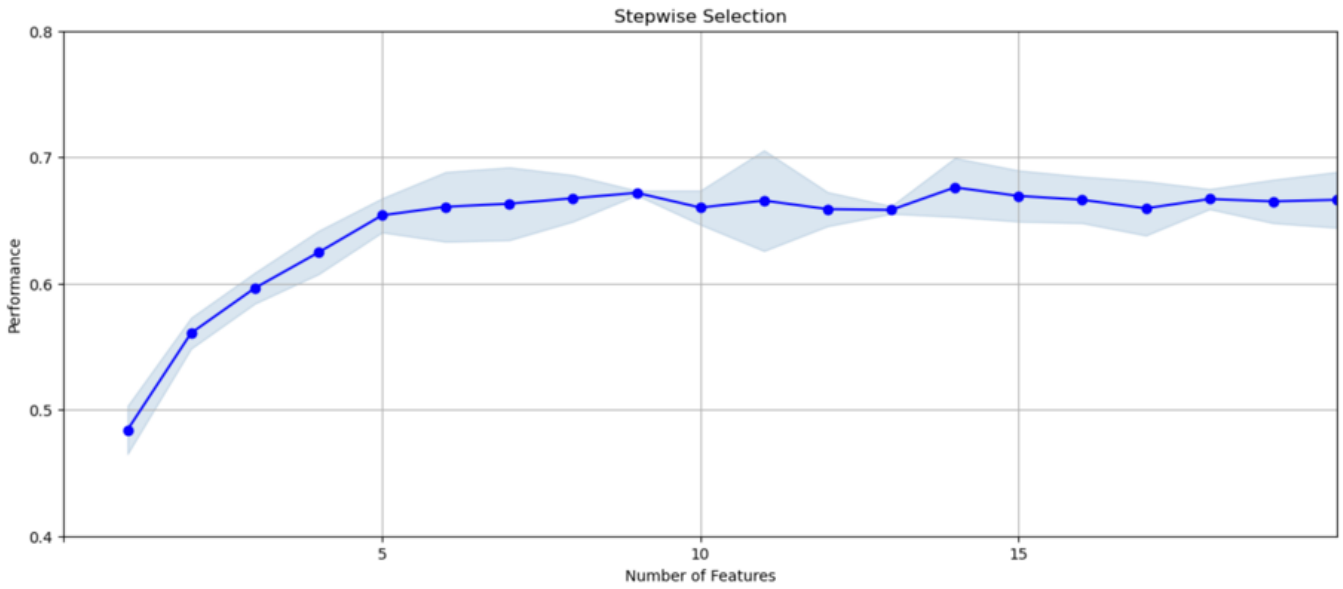


Fig 5.4

variable	filter score
Cardnum_unique_count_for_card_zip_1	0.474960
Card_Merchdesc_total_14	0.332040
Card_dow_unique_count_for_state_des_1	0.447238
card_state_max_14	0.305946
Card_dow_count_30	0.390454
Cardnum_count_0_by_60	0.317871
card_state_total_7	0.339462
card_zip_total_amount_1_by_60	0.319570
Card_dow_unique_count_for_merch_zip_14	0.401414
state_des_total_3	0.315540
Card_dow_actual/toal_14	0.331444
Cardnum_count_1_by_60	0.422739
Card_dow_unique_count_for_Card_Merchdesc_7	0.396572
card_merch_total_14	0.329023
Cardnum_unique_count_for_Merchnum_3	0.460748
merch_state_total_3	0.311228
Card_dow_max_60	0.383395
Card_dow_total_amount_0_by_60	0.338417
Card_dow_avg_7	0.326091
Cardnum_variability_avg_0	0.325614

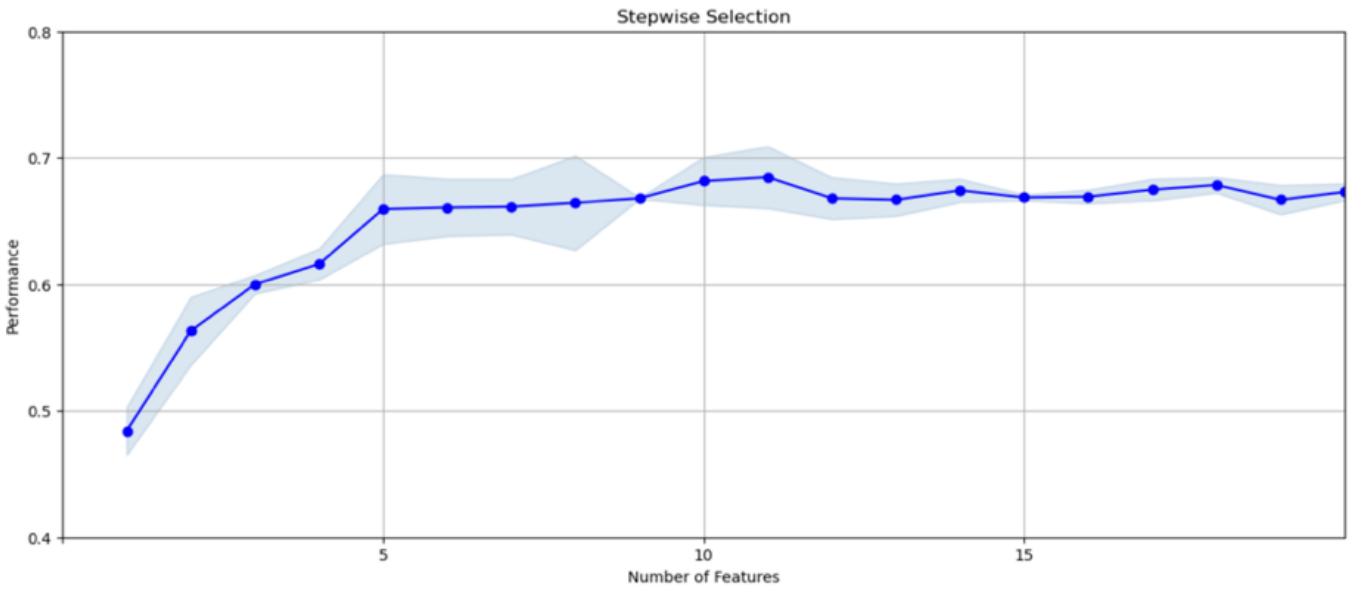


Fig 5.5

variable	filter score
Cardnum_unique_count_for_card_zip_1	0.474960
Card_Merchdesc_total_14	0.332040
Card_dow_unique_count_for_Card_Merchdesc_1	0.447250
card_state_max_14	0.305946
Card_dow_unique_count_for_merch_zip_60	0.320229
card_state_total_14	0.332008
Card_dow_count_7	0.482384
Card_dow_count_14	0.443405
Merchnum_desc_State_total_3	0.308586
Card_dow_actual/toal_14	0.331444
card_zip_total_amount_1_by_60	0.319570
Card_Merchdesc_Zip_total_3	0.319124
Cardnum_unique_count_for_card_zip_3	0.464323
Card_dow_unique_count_for_merch_zip_1	0.447157
Card_dow_unique_count_for_state_des_1	0.447238
merch_state_total_3	0.311228
Cardnum_count_7	0.526897
Card_dow_count_0_by_60	0.354882
Card_Merchnum_Zip_total_3	0.319487
Cardnum_max_7	0.410589

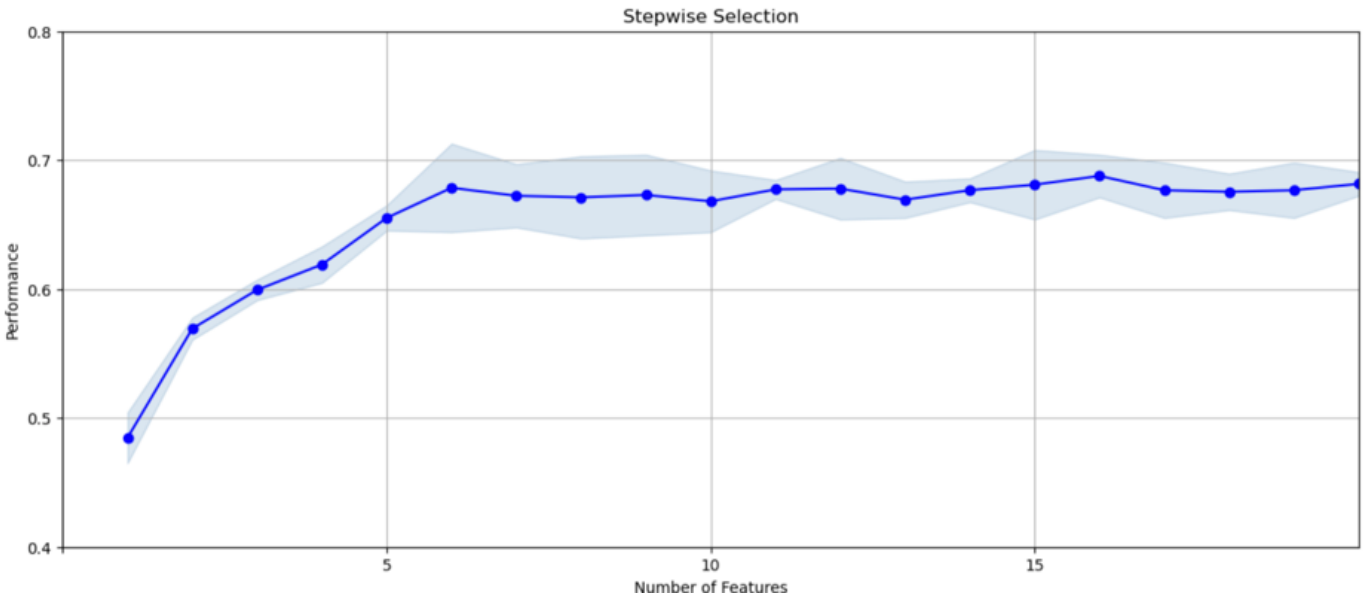


Fig 5.6

In three separate forward selection experiments using Random Forest with the same settings `num_filter = 200` and `num_wrapper = 20`, the results demonstrate consistent trends in feature selection and performance outcomes. Across these runs, the initial sharp increase in model performance reflects the significant impact of the first few features, which are predominantly related to transaction counts and descriptive attributes related to merchant and state details. Each graph depicts a rapid improvement in performance as these early features are added, with the model achieving substantial predictive power early in the feature inclusion process.

The performance then levels off, indicating a saturation point beyond which the addition of further features brings minimal benefits. This plateau suggests that the model efficiently captures the most predictive information with a limited set of features. Notably, the variables consistently appearing across different runs such as `Cardnum_unique_count_for_card_state_1` and `Card_dow_count_14` highlight their robustness and importance in the fraud detection model. These variables, which combine temporal and behavioral insights, are integral in identifying patterns indicative of fraudulent activity. This analysis underscores the effectiveness of using a targeted subset of key features, which not only simplifies the model but also enhances computational efficiency and predictive accuracy.

2) num_filter = 300, num_wrapper = 20

variable	filter score
Cardnum_unique_count_for_card_zip_1	0.474960
Cardnum_total_1	0.619658
Cardnum_count_0_by_30	0.304908
Card_Merchdesc_State_max_3	0.304959
Card_Merchdesc_Zip_total_7	0.324118
Cardnum_count_1_by_14	0.413860
Card_dow_unique_count_for_merch_state_7	0.417904
Card_Merchnum_desc_max_14	0.298143
card_state_total_14	0.332008
Card_Merchdesc_Zip_avg_3	0.281955
Merchnum_desc_total_3	0.308586
Card_Merchnum_Zip_total_1	0.315209
card_merch_total_0	0.294507
Card_dow_unique_count_for_state_des_1	0.447238
Cardnum_unique_count_for_Merchnum_60	0.306515
Card_Merchdesc_Zip_total_3	0.319124
Merchnum_desc_State_total_1	0.304969
Cardnum_count_14	0.445443
Card_dow_unique_count_for_Card_Merchdesc_7	0.396572
Cardnum_unique_count_for_card_state_3	0.466410

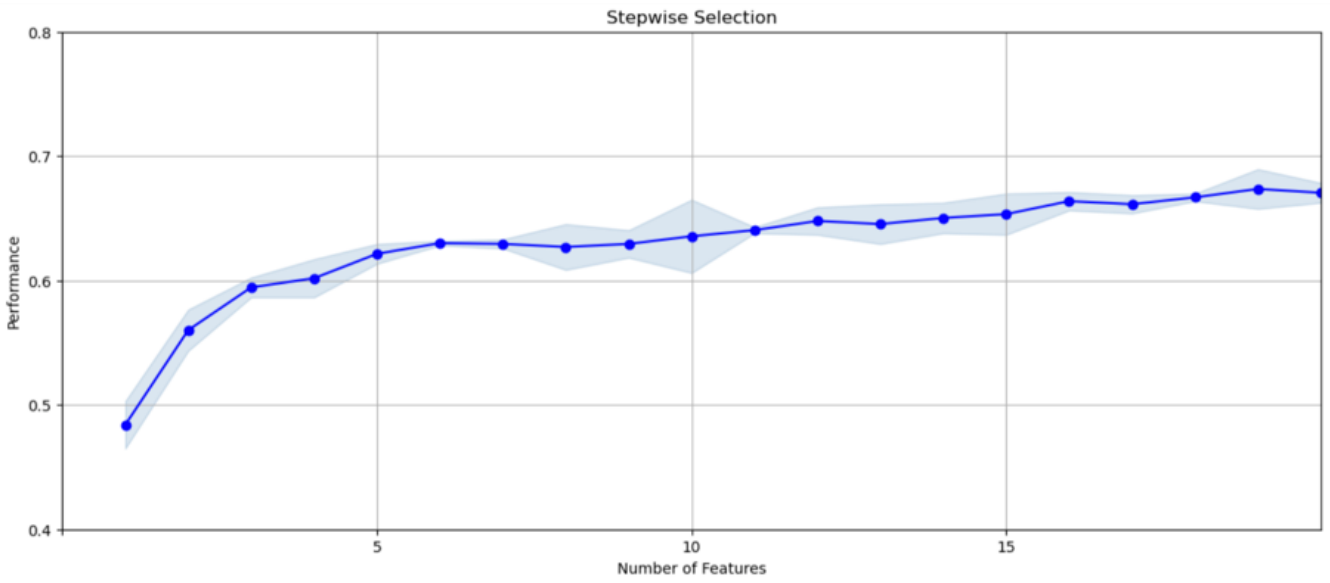


Fig 5.7

The forward selection analysis using Random Forest (RF) with `num_filter = 300` and `num_wrapper = 20` reveals a robust increase in model performance early on, highlighting the substantial impact of the initial features. As seen in the saturation plot, the performance ascends sharply with the inclusion of the first few features and then levels off around a performance score of 0.7 after about 10 features. This suggests that the early features are highly informative, capturing significant predictive signals necessary for fraud detection. Notably, variables like `Cardnum_total_1` and `Cardnum_unique_count_for_card_zip_1` show high filter scores, indicating their strong discriminatory power. The selection underscores the importance of these features in enhancing the fraud detection capabilities of the model while maintaining a manageable level of complexity.

Backward Selection:

1) num_filter = 100, num_wrapper = 10

variable	filter_score
Cardnum_total_0	0.591318
Cardnum_unique_count_for_card_state_60	0.511112
Cardnum_unique_count_for_card_state_30	0.388835
Cardnum_unique_count_for_card_state_14	0.423642
Cardnum_count_30	0.345799
Cardnum_vdratio_0by7	0.346091
Card_dow_actual/max_7	0.349000
Card_dow_avg_1	0.352529
Card_dow_variability_max_14	0.354679
Card_dow_variability_by_60_sq	0.354882
Card_dow_count_0_by_60	0.354882
Card_dow_total_amount_1_by_14	0.360916
Cardnum_unique_count_for_Merchnum_30	0.362204
Cardnum_avg_0	0.363159
Card_dow_unique_count_for_merch_state_30	0.360443
Card_dow_unique_count_for_merch_zip_30	0.364469
Cardnum_total_1	0.366290
Cardnum_unique_count_for_card_zip_30	0.372109
Cardnum_unique_count_for_Merchnum_14	0.374336
Card_dow_unique_count_for_state_des_14	0.374334

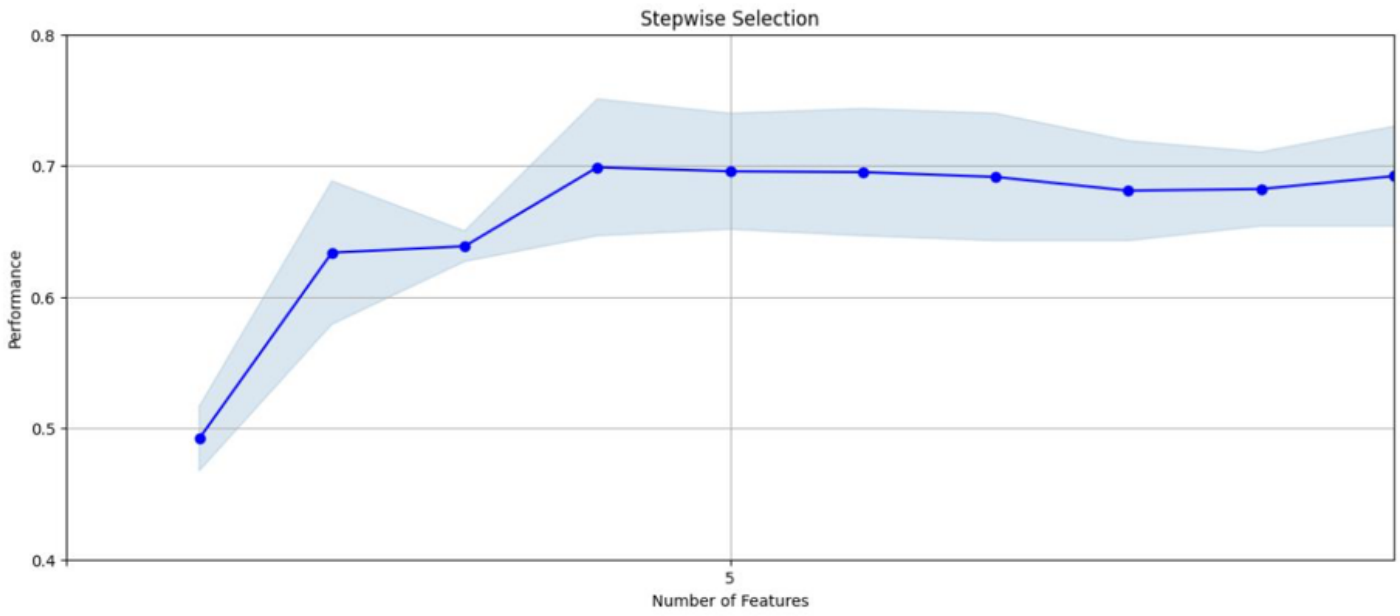


Fig 5.8

The backward selection process using LGBM with `num_filter = 100` and `num_wrapper = 10` provides valuable insights into feature importance for fraud detection models. The stepwise selection plot illustrates an initial significant increase in performance, which suggests that the initial features removed are of lesser importance. As more critically informative features begin to be removed around the fifth feature, the performance noticeably dips before stabilizing. This suggests that the most impactful features are capable of substantially boosting the model's predictive accuracy, while their removal leads to a deterioration in performance, underscoring their value.

The analysis of the variables retained in the model reveals that features related to transaction counts, amounts, and day of week variabilities - like `Cardnum_total_0`, `Cardnum_unique_count_for_card_state_60`, and `Card_dow_variability_max_14` - hold significant predictive power. These features are likely capturing essential behavioral patterns that are indicative of fraudulent activities. This demonstrates the effectiveness of backward selection in identifying and retaining the most predictive features, thus simplifying the model while focusing on high-impact variables that enhance the model’s ability to detect fraud accurately.

2) num_filter = 200, num_wrapper = 20

variable	filter_score
Cardnum_max_60	0.345646
Cardnum_total_0	0.591318
Cardnum_state_total_3	0.368088
Card_merch_total_60	0.305373
Merchnum_total_60	0.305358
Merchnum_desc_Zip_total_3	0.306561
Card_Merchnum_Zip_max_7	0.306110
Card_Merchnum_State_max_7	0.306110
Card_merch_max_7	0.306110
Card_Merchnum_desc_total_60	0.306425
Cardnum_unique_count_for_Merchnum_60	0.306515
Card_dow_variability_max_30	0.306735
Card_dow_avg_30	0.307748
Card_variability_max_7	0.307701
Card_total_30	0.307714
Card_merch_state_total_3	0.308586
Merchnum_desc_State_total_3	0.308586
Card_zip_max_1	0.308679
Card_Merchnum_State_max_3	0.308803
Card_Merchnum_Zip_max_3	0.308803

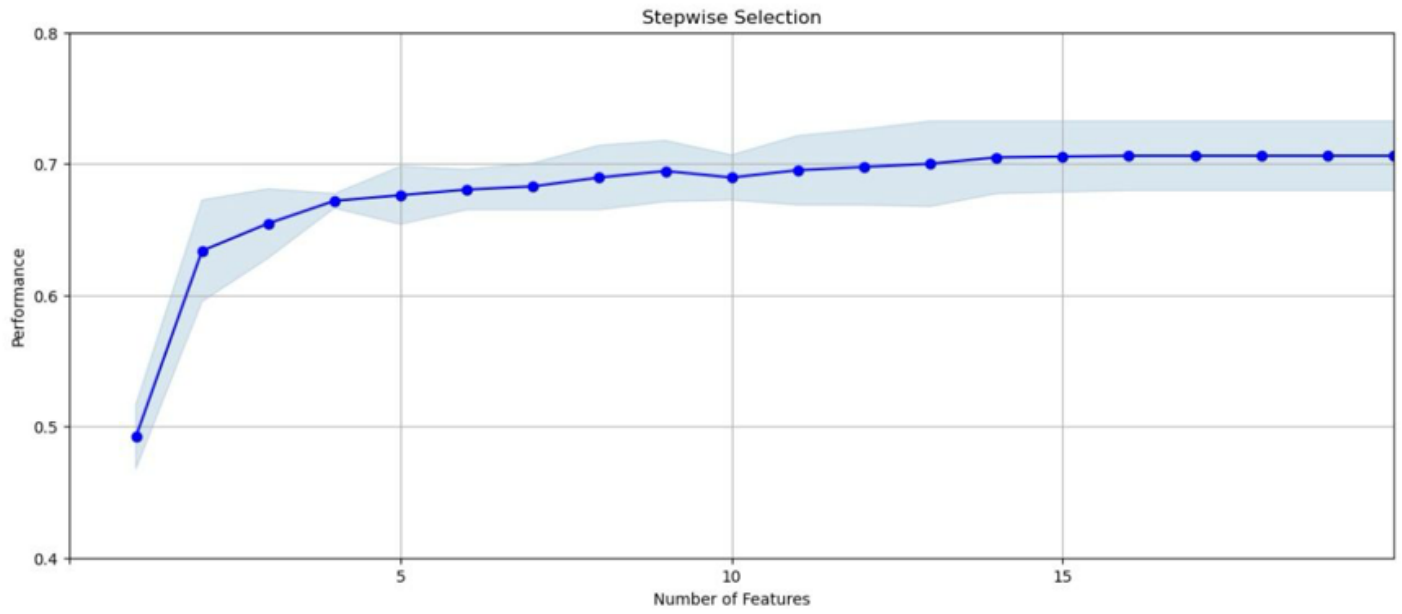


Fig 5.9

The backward selection analysis using LGBM with `num_filter = 200` and `num_wrapper = 20` illustrates an effective concentration of predictive power within a select group of features. This methodology starts with a full set of filtered variables and methodically removes the least influential, allowing us to observe the impact on model performance with each reduction. The stepwise selection graph shows an initial rapid increase in model performance as critical features are retained early in the process. The performance then begins to plateau around a score of 0.7, with minor improvements as more features are evaluated and discarded, suggesting that the most valuable features are robust and central to the model's ability to discriminate effectively between classes.

The variables that consistently show high importance across this selection process - such as `Cardnum_total_0`, `Cardnum_state_total_3`, and `Merchnum_desc_Zip_total_3` - indicate a strong influence of transaction amounts and merchant-specific details on fraud detection. These features are likely critical in capturing unusual patterns and behaviors indicative of fraudulent activities. By focusing on these key attributes, the backward selection approach demonstrates that maintaining a streamlined set of highly predictive features can significantly optimize the LGBM model's performance. This method not only simplifies the model but also enhances its efficiency and accuracy in identifying fraud, emphasizing the value of a targeted feature set over a more extensive but less impactful one.

Final Choice Rationale:

Saturation Plot: The saturation plot showing the forward selection from the LGBM model withnum_filter = 200andnum_wrapper = 20offers a clear demonstration of performance saturation. This plot reveals that the initial sharp increase in model performance stabilizes after the inclusion of approximately 10 features. This early leveling off suggests that these first few features capture the most critical information necessary for effective fraud detection, beyond which additional features add minimal predictive value.

List of 20 Final Variables for Modeling: From the LGBM experiment with num_filter = 200 and num_wrapper = 20, the first 20 variables that have shown the highest impact on model performance, based on their filter scores, are selected. These variables not only achieve a balance between providing substantial predictive power and model simplicity but also ensure diversity in capturing different aspects of transactional behavior and fraud patterns.

Variables Sorted by Wrapper with Univariate KS Scores:

wrapper order	variable	filter score
1	Cardnum_unique_count_for_card_state_1	0.476067
2	Card_Merchdesc_State_total_7	0.324668
3	Cardnum_count_1_by_30	0.428229
4	Cardnum_max_14	0.318826
5	Card_dow_vdratio_0by60	0.486480
6	Card_dow_vdratio_0by14	0.479086
7	Merchnum_desc_State_total_3	0.308586
8	Card_Merchdesc_total_7	0.324631
9	Card_dow_unique_count_for_merch_zip_7	0.418943
10	Cardnum_actual/toal_0	0.479550
11	Card_dow_vdratio_0by7	0.467961
12	Cardnum_vdratio_1by7	0.466766
13	Cardnum_unique_count_for_card_state_3	0.466410
14	Cardnum_unique_count_for_card_zip_3	0.464323
15	Merchnum_desc_Zip_total_3	0.305656
16	Cardnum_unique_count_for_Merchnum_3	0.460748
17	Cardnum_actual/toal_1	0.459715
18	Cardnum_unique_count_for_card_state_7	0.445967
19	Cardnum_actual/max_0	0.445726
20	Card_dow_unique_count_for_merch_state_1	0.447357

These variables reflect a robust combination of transactional amounts, frequency, merchant-related information, and temporal variations, providing a comprehensive view necessary for detecting and analyzing fraudulent transactions effectively. This selection process, illustrated through saturation plots and detailed variable analysis, has been designed to maximize the efficiency and accuracy of the fraud detection model while ensuring it remains computationally efficient and interpretable. The choice of these variables is backed by their demonstrated importance in preliminary models and their ability to collectively enhance model performance as seen in stepwise selection experiments.

Chapter 6: Preliminary Model Exploration

This section outlines the exploration and evaluation of various machine learning models for credit card fraud detection. The objective is to assess each model's ability to generalize across the training, test, and out-of-time (OOT) datasets, with a focus on minimizing overfitting. Overfitting occurs when a model performs well on the training set but fails to generalize to unseen data, leading to poor performance on the test and OOT datasets.

The models explored range from simpler ones like Logistic Regression to advanced techniques like LightGBM, XGBoost, Neural Networks, and CatBoost. Each model was tuned with specific hyperparameters to maximize performance, ensuring the model could generalize effectively across all datasets. The key evaluation metric used is the Fraud Detection Rate (FDR) at 3%, which measures how well the model identifies fraudulent transactions within the top 3% of predicted probabilities.

Approach to Model Tuning:

1. Initial Tuning of Simple Models

The project began with simpler models like Logistic Regression and Decision Trees. These models serve as baselines due to their interpretability and ease of implementation. The focus for these models was on tuning hyperparameters to balance performance between the training and test datasets. For Logistic Regression, hyperparameters such as regularization strength (L1, L2) were tuned to prevent overfitting, while for Decision Trees, parameters like tree depth and min_samples_leaf were optimized.

These models provided a starting point, offering insights into the limitations of linear and basic non-linear models in handling the complexity of fraud detection.

2. Systematic Tuning of Advanced Models

After establishing baselines, more complex models, including Random Forest, LightGBM, XGBoost, Neural Networks, and CatBoost, were systematically tuned. Key hyperparameters such as number of estimators, learning rate, max depth (for tree-based models), and number of neurons (for neural networks) were adjusted to maximize performance while controlling for overfitting.

The tuning process involved fine-tuning each model to handle the imbalanced nature of the dataset and to improve its generalization ability across the test and OOT datasets. These models provided a more nuanced understanding of data patterns and offered better predictive performance, especially in handling the complexities of fraud detection.

3. Addressing Overfitting

To ensure that models generalized well, overfitting was closely monitored by comparing model performance across the training, test, and OOT datasets. Overfitting was identified when models performed significantly better on the training set compared to the test or OOT datasets. In response, several methods were employed to mitigate overfitting:

- Regularization: For models like Logistic Regression and Neural Networks, regularization techniques (L1, L2, and dropout) were applied to reduce model complexity and prevent overfitting.
- Cross-validation: Cross-validation was used to tune hyperparameters and evaluate the model's robustness on the training set, ensuring that the models did not overfit to specific data splits.
- Class Imbalance Techniques: To address the issue of imbalanced data, SMOTE (Synthetic Minority Over-sampling Technique) was applied. SMOTE helped generate synthetic examples of the minority class (fraudulent transactions), improving model sensitivity to fraud detection. Its impact on OOT performance was carefully monitored to avoid introducing excessive noise into the model.

4. Evaluation Using FDR at 3%

All models were evaluated using FDR at 3%, which is critical for fraud detection tasks. FDR measures the model's ability to correctly identify fraudulent transactions within the top 3% of predicted probabilities. This metric aligns with real-world fraud detection goals, where resources are allocated to high-risk transactions, and only a small portion of all transactions can be flagged for manual review.

The evaluation was conducted on three datasets:

- Training Set: Used for model training and performance evaluation during the development phase.
- Test Set: A holdout set that was not used during training, enabling the evaluation of the model's performance on unseen data.
- OOT Set: An out-of-time set used to evaluate the model's ability to generalize to data from a different time period. This dataset provides a more realistic assessment of the model's performance in real-world applications, as it tests the model's robustness against shifts in data patterns over time.

Additional Considerations

During the model exploration process, several other factors were considered to ensure the robustness of the models:

- Feature Importance: For tree-based models like Random Forest, LightGBM, and XGBoost, feature importance was analyzed to understand which variables were most influential in predicting fraudulent transactions. This also provided insights for further feature engineering and allowed the refinement of input data for subsequent iterations of model tuning.
- Computational Efficiency: Given the large dataset and the high volume of transactions, models like LightGBM and CatBoost were chosen for their computational efficiency and ability to scale. Neural Networks, while powerful, required more tuning and computational resources to match the efficiency of gradient boosting models in fraud detection tasks.
- Scalability and Real-World Application: In a real-world fraud detection system, the ability to deploy models at scale is crucial. LightGBM and XGBoost were selected not only for their strong performance but also for their scalability and ease of integration into production systems. These models are well-suited for handling large-scale fraud detection applications, where speed and accuracy are critical.

Model Exploration Table:

Model	Parameters						Average FDR at 3%		
Logistic Regression	**Iteration**	**penalty**	**C**	**solver**	**l1_ratio**		**Train**	**Test**	**OOT**
	1	l2	0.1	liblinear	None		0.684118	0.673650	0.467003
	2	l2	1	liblinear	None		0.681911	0.676748	0.469024
	3	l2	10	liblinear	None		0.683657	0.674869	0.466330
	4	l1	0.1	liblinear	None		0.686619	0.669609	0.468350
	5	l1	1	liblinear	None		0.680618	0.680223	0.468687
	6	elasticnet	1	saga	0.5		0.682972	0.675514	0.465320
	7	elasticnet	0.5	saga	0.7		0.683291	0.677370	0.466330
	8	elasticnet	0.1	saga	0.3		0.679443	0.685345	0.467003
Decision Tree	**Iteration**	**criterion**	**max_depth**	**min_samples_split**	**min_samples_leaf**	**splitter**	**Train**	**Test**	**OOT**
	1	gini	3	100	50	best	0.665925	0.67039	0.424242
	2	entropy	4	120	60	random	0.695213	0.690371	0.470034
	3	gini	2	150	75	random	0.574743	0.563037	0.415825
	4	entropy	3	130	65	random	0.688174	0.681915	0.462963
	5	gini	4	200	80	best	0.692002	0.683711	0.472727
	6	entropy	5	180	90	best	0.72599	0.706928	0.483502
	7	gini	2	160	85	best	0.578315	0.569565	0.419192
	8	entropy	3	140	70	best	0.692537	0.67741	0.466667
Random Forest	**Iteration**	**criterion**	**max_depth**	**min_samples_split**	**min_samples_leaf**	**n_estimators**	**bootstrap**	**Train**	**Test**	**OOT**
	1	gini	6	100	50	50	TRUE	0.731795	0.724950	0.496296
	2	gini	5	120	60	40	TRUE	0.719976	0.723048	0.494276
	3	entropy	6	130	70	50	TRUE	0.732774	0.719806	0.490236
	4	gini	7	150	80	30	TRUE	0.730363	0.716688	0.495286
	5	entropy	8	140	70	60	TRUE	0.732960	0.727700	0.489226
	6	gini	4	160	80	40	TRUE	0.705598	0.716375	0.480135
	7	entropy	6	180	90	50	TRUE	0.731906	0.718501	0.484848
	8	gini	5	200	100	70	TRUE	0.718266	0.715968	0.492256

	Iteration	subsample	max_depth	learning_rate	n_estimators	min_child_samples	metric	Train	Test	OOT
LGBM	1	0.6	4	0.01	100	100	auc	0.750776	0.730784	0.539394
	2	0.6	5	0.008	150	90	logloss	0.763837	0.747216	0.539057
	3	0.5	5	0.005	120	80	auc	0.758059	0.744511	0.541414
	4	0.7	6	0.01	200	70	auc	0.823824	0.781250	0.560943
	5	0.5	4	0.005	100	80	auc	0.737645	0.723785	0.520202
	6	0.55	6	0.003	150	100	auc	0.764346	0.737261	0.525926
	7	0.5	4	0.002	120	120	logloss	0.720953	0.716818	0.502020
	8	0.45	4	0.001	100	150	logloss	0.716322	0.701719	0.488215
LGBM Smote	**iteration**	**num_leaves**	**max_depth**	**learning_rate**	**n_estimators**	**min_child_samples**	**metric**	**Train**	**Test**	**OOT**
	1	20	4	0.01	100	100	auc	0.718418	0.713196	0.481818
	2	30	6	0.005	150	90	logloss	0.748830	0.729426	0.507744
	3	15	5	0.01	200	80	auc	0.742389	0.723807	0.491246
	4	25	7	0.01	120	70	logloss	0.753864	0.739936	0.517508
	5	20	6	0.005	180	80	auc	0.735361	0.715167	0.499663
	6	35	8	0.003	150	100	auc	0.755047	0.721955	0.512795
	7	25	5	0.01	200	120	logloss	0.754503	0.731984	0.505387
	8	30	6	0.008	170	150	logloss	0.771252	0.729232	0.503030

	iteration	num_leaves	max_depth	learning_rate	n_estimators	min_child_samples	metric	Train	Test	OOT
LGBM Sampled	1	20	4	0.01	100	100	auc	0.706971	0.706971	0.496633
	2	25	5	0.005	120	90	logloss	0.725886	0.725886	0.507744
	3	30	6	0.008	150	80	auc	0.719829	0.719829	0.508081
	4	15	3	0.01	100	70	logloss	0.702514	0.702514	0.480471
	5	35	7	0.003	200	80	auc	0.715543	0.715543	0.499663
	6	25	6	0.005	180	100	auc	0.717771	0.717771	0.504714
	7	20	5	0.008	130	120	logloss	0.7208	0.7208	0.509091
	8	30	4	0.006	170	150	logloss	0.710343	0.710343	0.49596

	Iteration	activation	alpha	learning_rate	hidden_layer_size	solver	Train	Test	OOT
Neural Network	1	relu	0.005	0.01	(20, 20)	adam	0.764277	0.739786	0.532660
	2	relu	0.01	0.005	(10, 10)	adam	0.741667	0.743258	0.525253
	3	tanh	0.01	0.01	(50,)	adam	0.739133	0.727414	0.505724
	4	logistic	0.005	0.005	(30, 30)	sgd	0.794895	0.746120	0.504714
	5	relu	0.01	0.01	(40, 20)	adam	0.756159	0.737875	0.533333
	6	tanh	0.01	0.005	(25, 25)	sgd	0.774456	0.742616	0.523906
	7	relu	0.001	0.01	(30,)	adam	0.758363	0.743733	0.520202
	8	tanh	0.005	0.01	(15, 15)	adam	0.760287	0.740029	0.527273

	Iteration	subsample	max_depth	learning_rate	n_estimators	l2_leaf_reg	colsample_bylevel	Train	Test	OOT
Catboost	1	0.6	4	0.01	200	3	0.8	0.715021	0.714661	0.475084
	2	0.7	5	0.008	250	5	0.9	0.725319	0.718928	0.480471
	3	0.5	4	0.01	150	7	0.8	0.702631	0.696845	0.475084
	4	0.6	6	0.005	300	3	0.7	0.724853	0.727028	0.482155
	5	0.7	5	0.007	200	10	0.85	0.719931	0.709825	0.482155
	6	0.5	4	0.005	100	8	0.7	0.696573	0.696614	0.479461
	7	0.6	3	0.002	200	5	0.9	0.694205	0.684423	0.465993
	8	0.55	4	0.003	150	4	0.75	0.692262	0.687328	0.478788

	Iteration	subsample	max_depth	learning_rate	n_estimators	metric	colsample_bytree	Train	Test	OOT
XGB	1	0.6	4	0.01	200	logloss	0.8	0.748047	0.736402	0.506397
	2	0.7	5	0.008	250	auc	0.9	0.774762	0.747548	0.521886
	3	0.5	4	0.01	150	logloss	0.7	0.735836	0.735333	0.502020
	4	0.6	6	0.005	300	auc	0.8	0.778832	0.759562	0.541077
	5	0.7	5	0.007	200	logloss	0.9	0.763509	0.747203	0.537374
	6	0.5	4	0.005	100	auc	0.7	0.725204	0.721485	0.488215
	7	0.6	3	0.002	200	logloss	0.85	0.700167	0.698600	0.470370
	8	0.55	4	0.003	150	logloss	0.75	0.726406	0.716130	0.491246

Fig 6.1

Model Explanation:

The models range from simple linear classifiers to advanced ensemble methods. Each model's ability to generalize across the Training, Test, and Out-of-Time (OOT) datasets is discussed, focusing on the Fraud Detection Rate (FDR) at 3%.

1. Logistic Regression:

Logistic Regression is a linear model that estimates the probability of a binary outcome (fraud or non-fraud). Regularization techniques like L2 (Ridge), L1 (Lasso), and ElasticNet were used to reduce overfitting by penalizing large coefficients.

- Best Train FDR: 0.686619 (L1 regularization, C=0.1, solver=liblinear)
- Best Test FDR: 0.680223 (L1 regularization, C=1, solver=liblinear)
- Best OOT FDR: 0.469024 (L2 regularization, C=1, solver=liblinear)

Logistic Regression displayed strong stability on the training and test sets. However, its OOT performance fell considerably, indicating that it struggles to generalize well in complex fraud detection tasks.

2. Decision Tree:

Decision Trees are non-linear models that split data based on certain criteria, such as Gini or Entropy. Various hyperparameters, such as max_depth, min_samples_split, and min_samples_leaf, were tuned to optimize performance and avoid overfitting.

- Best Train FDR: 0.725990 (Entropy criterion, max_depth=5, min_samples_split=180, min_samples_leaf=90)
- Best Test FDR: 0.706928 (Entropy criterion, max_depth=5, min_samples_split=180, min_samples_leaf=90)
- Best OOT FDR: 0.483502 (Entropy criterion, max_depth=5, min_samples_split=180, min_samples_leaf=90)

While Decision Trees improved on Logistic Regression's ability to model complex relationships, they still suffered from overfitting, which is evident from the performance drop on the OOT dataset.

3. Random Forest:

Random Forest is an ensemble learning method that builds multiple decision trees and averages their predictions to reduce overfitting. n_estimators, max_depth, and min_samples_split were tuned to balance performance and complexity.

- Best Train FDR: 0.732960 (Entropy criterion, max_depth=8, n_estimators=60, min_samples_split=140)
- Best Test FDR: 0.727700 (Entropy criterion, max_depth=8, n_estimators=60, min_samples_split=140)
- Best OOT FDR: 0.496296 (Gini criterion, max_depth=6, n_estimators=50, min_samples_split=100)

Random Forest showed strong generalization compared to single decision trees. Its ability to maintain high performance across all datasets makes it a reliable model for fraud detection.

4. LightGBM:

LightGBM (Light Gradient Boosting Machine) grows trees leaf-wise, allowing it to capture more intricate patterns. Key hyperparameters like learning rate, max_depth, and n_estimators were tuned for optimal performance.

- Best Train FDR: 0.823824 (Subsample=0.7, max_depth=6, learning_rate=0.01, n_estimators=200)
- Best Test FDR: 0.781250 (Subsample=0.7, max_depth=6, learning_rate=0.01, n_estimators=200)
- Best OOT FDR: 0.560943 (Subsample=0.7, max_depth=6, learning_rate=0.01, n_estimators=200)

LightGBM emerged as the top performer with outstanding generalization across datasets. Its ability to handle complex fraud detection tasks was superior to other models.

5. LightGBM with SMOTE:

SMOTE (Synthetic Minority Over-sampling Technique) was applied to the LightGBM model to generate synthetic examples for the minority class (fraudulent transactions) to improve the model's ability to detect fraud.

- Best Train FDR: 0.771252 (Num_leaves=30, max_depth=6, learning_rate=0.008, n_estimators=170)
- Best Test FDR: 0.739936 (Num_leaves=25, max_depth=7, learning_rate=0.01, n_estimators=120)
- Best OOT FDR: 0.517508 (Num_leaves=25, max_depth=7, learning_rate=0.01, n_estimators=120)

Though SMOTE improved the sensitivity to fraudulent transactions, the OOT performance was still lower than the standard LightGBM model, suggesting that synthetic data may introduce noise affecting generalization.

6. LightGBM Sampled:

LightGBM with down-sampling was used to handle class imbalance by reducing the size of the majority class (non-fraudulent transactions) instead of generating synthetic data. This method attempts to improve model focus on the minority class.

- Best Train FDR: 0.720800 (Num_leaves=20, max_depth=4, learning_rate=0.01, n_estimators=100)
- Best Test FDR: 0.720800 (Num_leaves=20, max_depth=4, learning_rate=0.01, n_estimators=100)
- Best OOT FDR: 0.509091 (Num_leaves=20, max_depth=4, learning_rate=0.01, n_estimators=100)

LightGBM Sampled performed well across the datasets, but its OOT performance was still below the standard LightGBM model, suggesting that sampling might reduce the dataset's ability to capture complex patterns.

7. Neural Network:

Neural Networks are capable of capturing complex non-linear relationships. Various architectures, such as different numbers of hidden layers, neurons per layer, and activation functions (ReLU, Tanh), were explored.

- Best Train FDR: 0.794895 (Activation=Logistic, alpha=0.005, learning_rate=0.005, hidden_layer_size=(30, 30), solver=SGD)
- Best Test FDR: 0.746120 (Activation=Logistic, alpha=0.005, learning_rate=0.005, hidden_layer_size=(30, 30), solver=SGD)
- Best OOT FDR: 0.533333 (Activation=ReLU, alpha=0.01, learning_rate=0.01, hidden_layer_size=(40, 20), solver=Adam)

Neural Networks performed well on both training and test sets, but the significant drop in OOT performance suggests overfitting. Further tuning is necessary to improve generalization.

8. CatBoost:

CatBoost is a gradient boosting algorithm that handles categorical features natively. It reduces the need for extensive preprocessing and was tuned with parameters like subsample rate, max_depth, and learning rate.

- Best Train FDR: 0.725319 (Subsample=0.7, max_depth=5, learning_rate=0.008, n_estimators=250)
- Best Test FDR: 0.727028 (Subsample=0.6, max_depth=6, learning_rate=0.005, n_estimators=300)
- Best OOT FDR: 0.482155 (Subsample=0.6, max_depth=6, learning_rate=0.005, n_estimators=300)

CatBoost showed strong performance on the training and test datasets but had slightly lower generalization compared to LightGBM and XGBoost. Further tuning may be necessary to improve OOT results.

9. XGBoost:

XGBoost is a highly efficient gradient boosting algorithm. It builds trees sequentially, with each new tree correcting the errors of the previous one. Hyperparameters like learning rate, n_estimators, and max_depth were tuned.

- Best Train FDR: 0.778832 (Subsample=0.6, max_depth=6, learning_rate=0.005, n_estimators=300)
- Best Test FDR: 0.759562 (Subsample=0.6, max_depth=6, learning_rate=0.005, n_estimators=300)
- Best OOT FDR: 0.541077 (Subsample=0.6, max_depth=6, learning_rate=0.005, n_estimators=300)

XGBoost delivered excellent generalization, especially in terms of OOT performance. While slightly behind LightGBM, XGBoost remains one of the top-performing models for this task.

The evaluation of different models highlights the complexity of addressing fraud detection tasks. LightGBM demonstrated the highest OOT FDR score and strong generalization across datasets. Random Forest and XGBoost also performed well, showing solid generalization, while simpler models like Logistic Regression and Decision Trees showed limitations in matching the performance of ensemble and gradient boosting techniques.

Performance Plot:

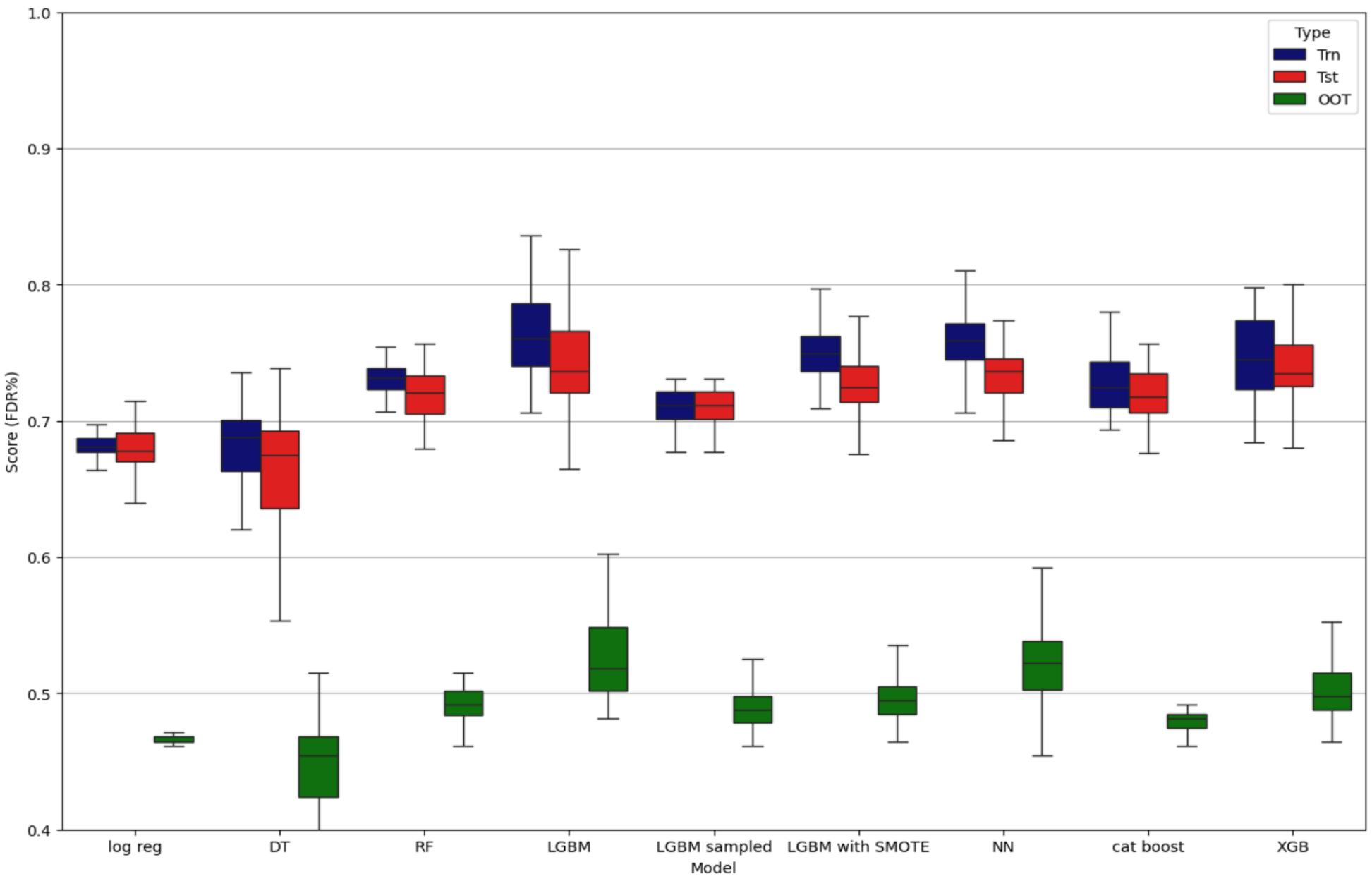


Fig 6.2

The boxplot visualization provides a graphical representation of the distribution of **Fraud Detection Rate (FDR) at 3%**across the training, test, and out-of-time (OOT) datasets for each model. This plot highlights how each model

performs in terms of detecting fraudulent transactions and helps identify overfitting or poor generalization to new data (OOT). Below is an expanded analysis of each model based on the plot:

Key Insights from the Performance Plot:

1. Logistic Regression:

- **Performance**: The model shows consistent performance across the training and test sets, maintaining a relatively stable FDR. However, its OOT performance drops significantly compared to the test set.
- **Insight**: This significant drop in OOT performance suggests that Logistic Regression struggles to generalize to new, unseen data. This behavior indicates that while the model can fit the data well in-sample, it may not be sophisticated enough to capture complex patterns associated with fraud in an out-of-sample environment.

2. Decision Tree:

- **Performance**: Decision Trees exhibit more variability than Logistic Regression. Although the model shows improvement in FDR on the test set, there is still a considerable gap between test and OOT performance, suggesting overfitting.
- **Insight**: The Decision Tree model has the flexibility to capture non-linear patterns, but its structure can easily overfit to training data. The gap in OOT performance reveals that the model fails to generalize effectively, requiring further regularization or tuning to improve OOT results.

3. Random Forest:

- **Performance**: Random Forest shows strong performance across the training and test datasets, with a relatively small decline in OOT performance. The model achieves higher FDR scores compared to simpler models like Logistic Regression and Decision Tree.
- **Insight**: Random Forest's ensemble approach, which averages predictions from multiple decision trees, enables it to generalize better than individual trees. The smaller gap between test and OOT performance indicates that Random Forest handles the complexity of fraud detection well, making it a reliable option.

4. LightGBM:

- **Performance**: LightGBM is the standout performer in this analysis. It consistently achieves the highest FDR scores across the test and OOT datasets, with minimal overfitting. Its OOT performance surpasses that of other models, demonstrating excellent generalization.
- **Insight**: The strong OOT performance of LightGBM highlights its ability to capture complex patterns in fraudulent transactions. LightGBM's leaf-wise tree growth method allows it to focus on critical splits, contributing to its superior generalization capabilities and robustness in this fraud detection task.

5. LightGBM with SMOTE:

- **Performance**: Applying SMOTE to handle class imbalance helped improve the model's ability to detect fraudulent transactions. However, the OOT performance was lower compared to the standard LightGBM model.
- **Insight**: SMOTE is effective in increasing the model's sensitivity to fraud by generating synthetic data for the minority class. However, the lower OOT performance suggests that SMOTE might introduce noise into the model, leading to overfitting on the training and test datasets and weaker performance on unseen OOT data.

6. LightGBM Sampled:

- **Performance**: LightGBM Sampled handled class imbalance by reducing the size of the majority class (non-fraud transactions), leading to a good test performance. However, the OOT performance was slightly lower compared to the original LightGBM.
- **Insight**: While down-sampling the majority class helps focus the model on fraud detection, it reduces the amount of training data available. This reduction may explain the slight drop in OOT performance, as the model may lose valuable information about the non-fraudulent transactions, leading to less robust generalization compared to full LightGBM.

7. Neural Network:

- **Performance**: Neural Networks performed well on the training and test datasets but showed a noticeable decline in OOT performance. This suggests that the model was able to learn from the training data but struggled to generalize to unseen data.
- **Insight**: Neural Networks, while powerful, are prone to overfitting, especially when not properly regularized. The significant drop in OOT performance indicates that the network may have memorized the training patterns but failed to generalize, highlighting the need for additional techniques like dropout or early stopping to improve generalization.

8. XGBoost:

- **Performance**: XGBoost delivered strong and consistent performance across all datasets, with FDR scores close to LightGBM. It maintained good generalization to the OOT dataset, making it one of the top-performing models.
- **Insight**: XGBoost, known for its efficient handling of complex data, proves to be a reliable model for fraud detection. Its performance closely matches LightGBM, indicating that gradient boosting techniques are well-suited for handling the nuances of fraud detection tasks.

9. CatBoost:

- **Performance**: CatBoost performed well on both the training and test datasets but showed a slightly weaker OOT performance compared to LightGBM and XGBoost. The model handled categorical variables efficiently, reducing the need for extensive preprocessing.

- **Insight**: CatBoost's handling of categorical features without requiring heavy preprocessing makes it an efficient model for this task. However, its OOT performance suggests that it may require further tuning to match the generalization capabilities of other gradient boosting models like LightGBM and XGBoost.

The boxplot visualization serves as a crucial tool for comparing the performance of different machine learning models in the context of fraud detection. While models like LightGBM, XGBoost, and Random Forest show strong generalization capabilities, others like Logistic Regression and Decision Tree reveal limitations in handling the complexity of fraud detection, particularly in terms of OOT performance. The plot also demonstrates the potential of SMOTE and down-sampling techniques, although they may introduce challenges in generalization, as seen in the case of LightGBM with SMOTE and LightGBM Sampled. This insight underscores the importance of carefully selecting and tuning machine learning models for fraud detection tasks. Advanced models like LightGBM and XGBoost tend to offer better performance, but adjustments in data handling methods, such as oversampling or down-sampling, require thorough validation to ensure that generalization to unseen data is not compromised. Further exploration of regularization techniques, feature engineering, and hyperparameter tuning could help refine these models for even better fraud detection results.

Performance vs. Complexity: Overfitting Analysis Across Models

The performance vs. complexity plots presented for each model type help illustrate the behavior of these machine learning models as their complexity increases. In this analysis, complexity is controlled by specific hyperparameters, such as regularization strength for Logistic Regression, tree depth for decision trees, and the number of estimators for ensemble models.

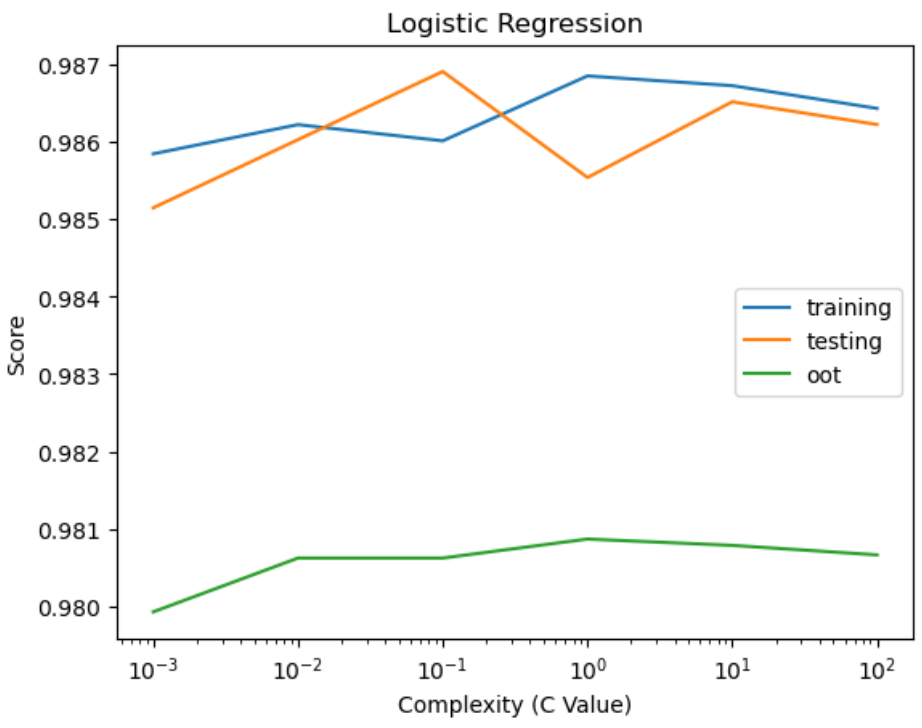


Fig 6.3.1

Fig 6.3.2

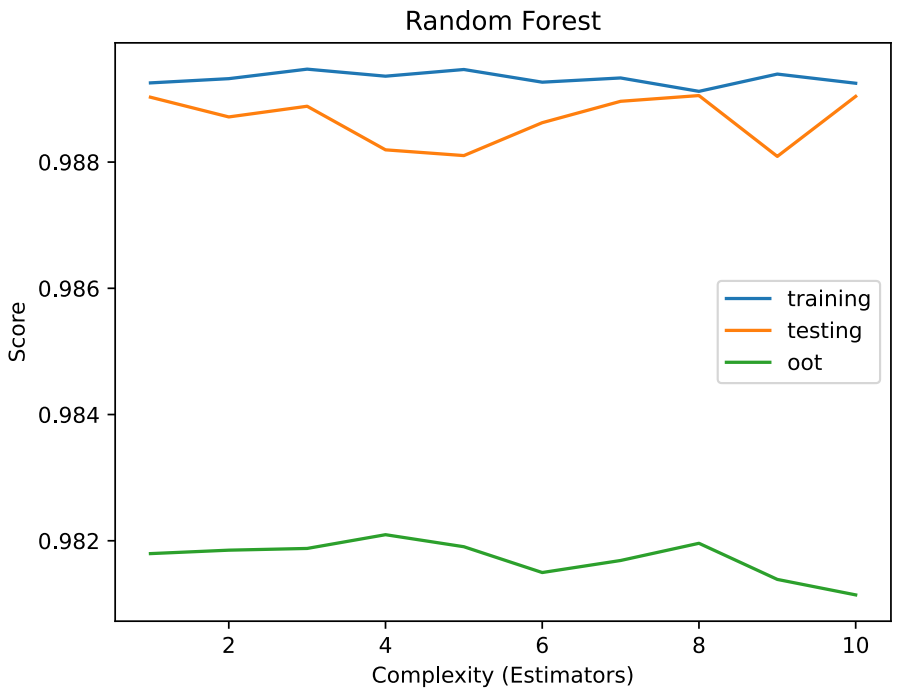


Fig 6.3.3

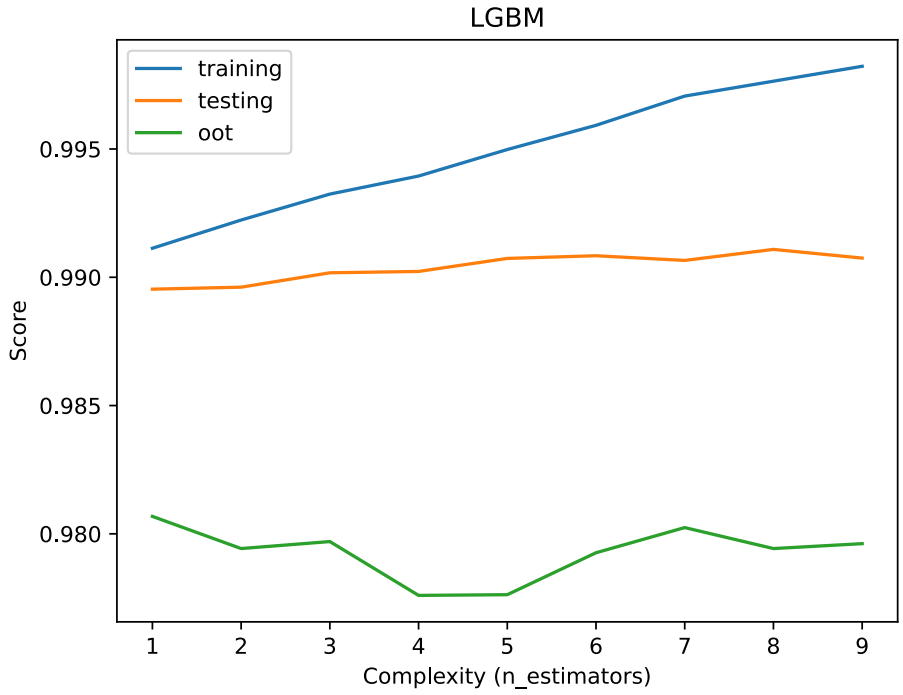


Fig 6.3.4

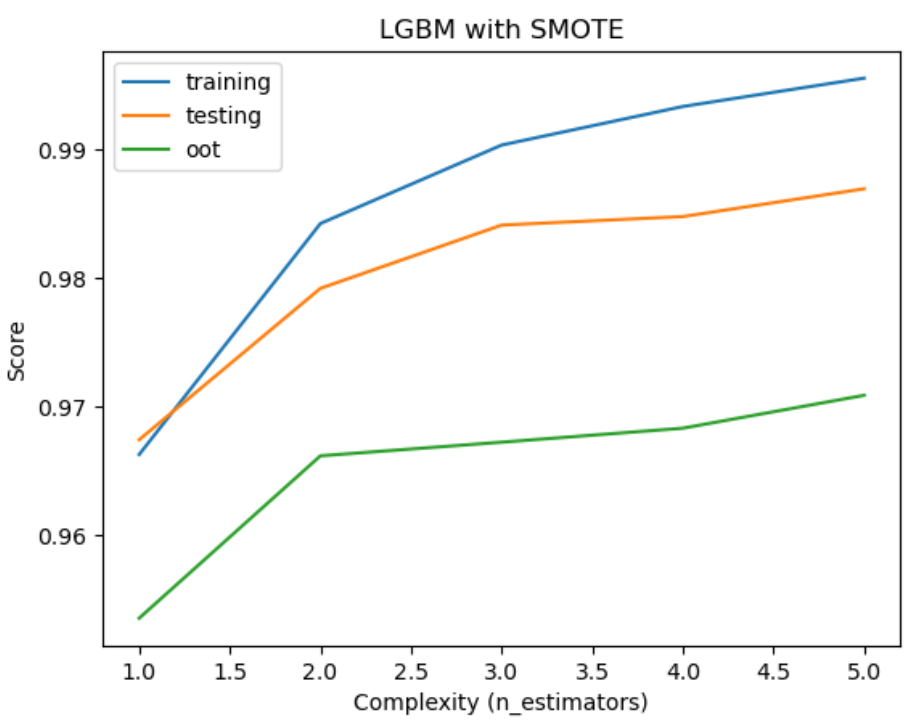
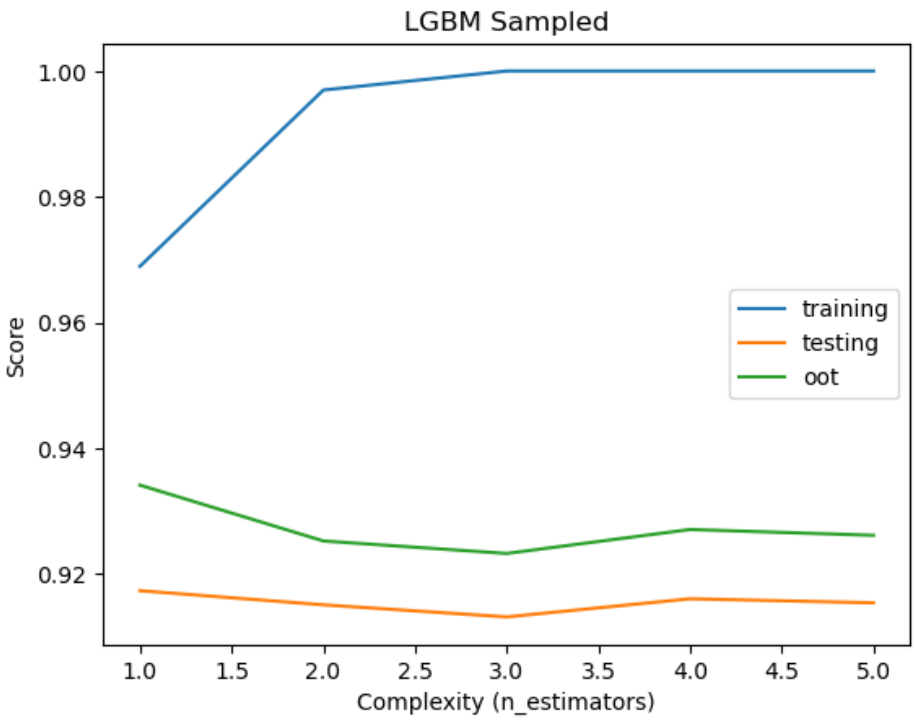


Fig 6.3.5



Fi.3.6

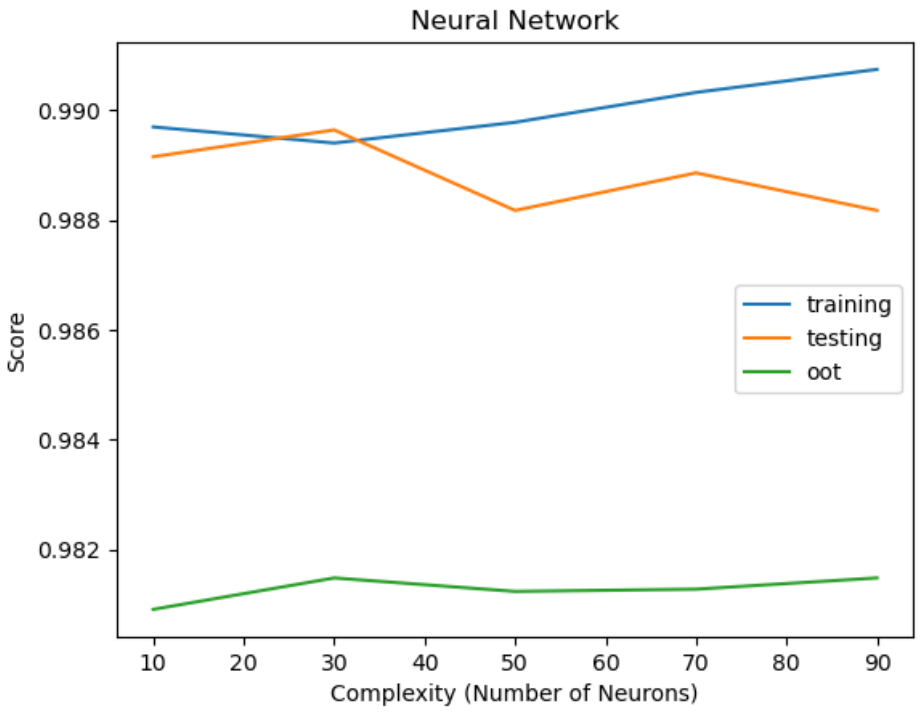


Fig 6.3.7

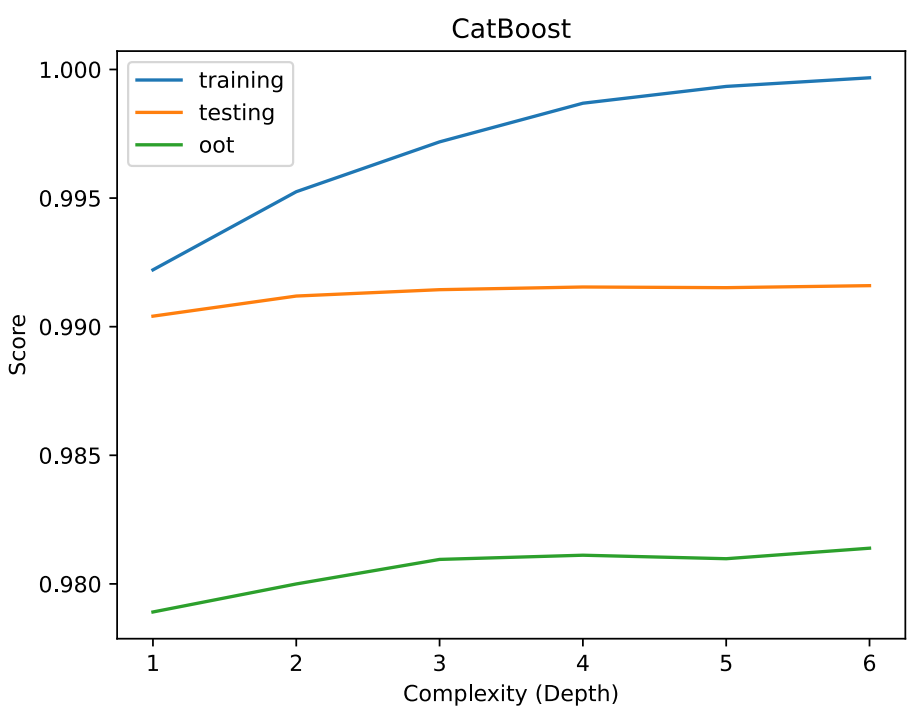


Fig 6.3.8

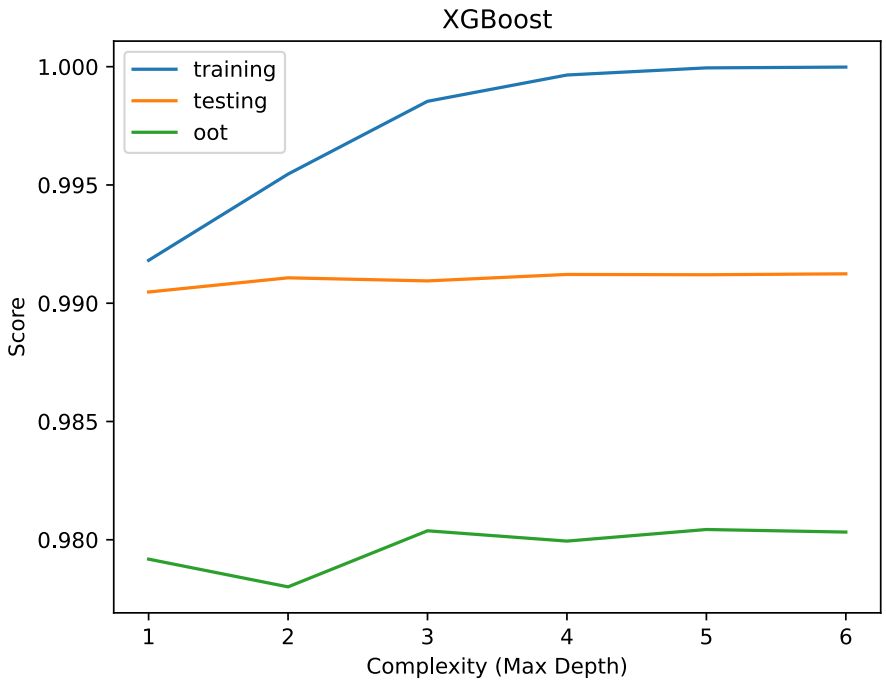


Fig 6.3.9

1. Logistic Regression:

In the Logistic Regression plot, the x-axis represents the regularization parameter (C Value), with smaller values indicating stronger regularization and larger values indicating less regularization (higher model flexibility). The training and test performance curves stay quite close together, showing that Logistic Regression is not particularly prone to overfitting. However, the OOT performance remains lower than both training and test across all levels of complexity, which suggests that while the model does not overfit significantly, it struggles to generalize well to out-of-time (OOT) data. This consistent gap indicates that Logistic Regression has limitations in capturing the complex patterns of fraud in this dataset.

2. Decision Tree:

In the Decision Tree plot, complexity is increased by adjusting the minimum number of samples required to split a node (Min Samples Split). As complexity increases (moving right on the x-axis), the training score rises significantly, indicating that the model becomes more specialized in fitting the training data. However, this improvement is not reflected in the test and OOT datasets, where the scores remain relatively flat or even decrease slightly. This is a classic sign of overfitting: the model fits the training data more closely but performs worse on unseen test and OOT data. The gap between training and OOT performance widens as complexity increases, further confirming the overfitting behavior.

3. Random Forest:

For Random Forest, the number of estimators (trees in the forest) is used to control complexity. Unlike Decision Trees, Random Forest is generally more robust to overfitting because it averages the predictions of multiple trees. In this plot, both training and test scores remain stable across different complexity levels, suggesting that the model is able to maintain good generalization. The OOT score, while slightly lower than the test and training scores, does not show a dramatic drop as complexity increases. This indicates that Random Forest effectively balances between learning complex patterns and generalizing to unseen data.

4. LightGBM:

LightGBM, known for its efficiency in handling large datasets, increases in complexity with the number of estimators. In this plot, the training score improves steadily as complexity increases, but the test and OOT scores remain relatively stable. The gap between the OOT score and the other scores indicates that while LightGBM can handle the training data well, increasing complexity doesn't significantly improve generalization to the OOT dataset. The stability of the test score suggests that overfitting is minimal, but further optimization may be necessary to improve OOT performance.

5. LightGBM with SMOTE:

In this plot, the use of SMOTE (Synthetic Minority Over-sampling Technique) is combined with LightGBM to address class imbalance. As complexity increases (with more estimators), all three performance curves show improvement, though the OOT score lags behind the test and training scores. While SMOTE helps the model better detect minority classes (fraudulent transactions), the gap between OOT and training/test scores suggests that SMOTE introduces some noise, making it harder for the model to generalize to new, unseen data. This indicates that while SMOTE can be effective in improving sensitivity to minority classes, it may also lead to overfitting if not carefully managed.

6. LightGBM Sampled:

LightGBM Sampled uses down-sampling to balance the class distribution. In this plot, we see that as complexity increases, the training score climbs rapidly towards perfect accuracy, indicating significant overfitting. However, both test and OOT scores remain relatively flat, showing no substantial improvement with increasing complexity. The gap between the training score and the other two datasets demonstrates severe overfitting, where the model becomes too tailored to the training data and loses its ability to generalize well to unseen data. Down-sampling has helped balance the classes, but it sacrifices some generalization ability as complexity increases.

7. Neural Network:

For the Neural Network, complexity is controlled by increasing the number of neurons in the hidden layers. In this plot, the training score improves slightly as complexity increases, while the test score fluctuates but remains relatively stable. However, the OOT score shows no significant improvement and remains much lower than both the training and test scores. This suggests that the neural network struggles with overfitting, particularly at higher complexity levels. Despite its ability to model non-linear relationships, this model may require additional regularization techniques, such as dropout, to improve its generalization ability.

8. CatBoost:

In the CatBoost plot, complexity is controlled by adjusting the tree depth. As tree depth increases, the training score improves significantly, but the test and OOT scores do not see similar gains. This widening gap between training and OOT performance indicates that CatBoost is prone to overfitting at higher complexity levels. Although it

efficiently handles categorical features, further tuning is needed to ensure that it generalizes well across datasets without overfitting to the training data.

9. XGBoost:

XGBoost is another gradient boosting model, with complexity controlled by max_depth. As the depth of the trees increases, the training score improves sharply, indicating that the model is fitting the training data very well. However, both the test and OOT scores remain relatively stable, showing little benefit from increasing complexity. The gap between training and OOT performance indicates some overfitting, though the relatively stable test score suggests that XGBoost manages overfitting better than simpler models like Decision Trees.

Therefore, across all models overfitting is observed to varying degrees as complexity increases. Models like Random Forest, LightGBM, and XGBoost handle complexity more effectively, maintaining good performance on test and OOT datasets, while models like Decision Trees, Neural Networks, and LightGBM Sampled show more pronounced overfitting, with widening gaps between training and OOT performance. The balance between complexity and performance is a key theme in this analysis. While increasing complexity can lead to better performance on training data, it often comes at the cost of generalization to unseen data, especially for models without built-in regularization techniques. Techniques like SMOTE and down-sampling show some promise in addressing class imbalance but need to be carefully managed to avoid degrading OOT performance.

In the Preliminary Model Exploration Section, a range of machine learning models for credit card fraud detection were evaluated to understand their ability to generalize across training, test, and out-of-time (OOT) datasets, while focusing on minimizing overfitting. Advanced models like LightGBM, XGBoost, and Random Forest demonstrated strong generalization and handled increasing complexity effectively, with stable performance across all datasets. In contrast, simpler models such as Logistic Regression and Decision Trees showed limitations, particularly in their ability to generalize, often overfitting to the training data. Techniques like SMOTE and down-sampling were employed to address class imbalance, though these methods had varied impacts on OOT performance, with some introducing noise. Ultimately, this section highlights the importance of model complexity, hyperparameter tuning, and data handling techniques in building robust fraud detection models, with LightGBM emerging as the top performer in handling real-world challenges.

Chapter 7: Final Model Performance

The final model selected for credit card fraud detection is the **Light Gradient Boosting Machine (LightGBM)**. This model was chosen due to its superior ability to handle imbalanced datasets, which is critical in fraud detection where fraudulent transactions are rare. LightGBM also demonstrated excellent generalization across the three key datasets: training, test, and out-of-time (OOT), making it a robust choice for real-world deployment. The decision to use Iteration 4 of the LightGBM model was based on its outstanding performance, particularly in terms of the out-of-time Fraud Detection Rate (FDR), which is essential for ensuring the model's effectiveness on future, unseen data.

During the modeling process, a target out-of-time FDR of **0.56** was set as the desired performance threshold for stopping the iterations. This value was based on the best OOT FDR observed during the model exploration phase. The model iterated through several configurations until it achieved or exceeded this desired FDR, ensuring optimal performance on future datasets. By setting this FDR target, the model was guided to prioritize fraud detection in unseen data, which is critical for real-world application where fraudulent transactions may evolve over time.

Key Non-Default Hyperparameter Choices:

The LightGBM model's success was largely attributed to the careful tuning of hyperparameters during the model exploration process. Each hyperparameter was selected to balance performance and prevent overfitting, ensuring that the model could maintain its predictive power across different datasets.

1. Subsample: 0.7

Subsampling refers to the practice of using a random subset of the data to build each tree within the model. By setting the subsample rate to 0.7, we ensure that each tree only learns from 70% of the data, which helps in reducing overfitting. This strategy prevents the model from becoming overly reliant on specific data points or noise in the training set, allowing it to learn more generalized patterns. Subsampling is particularly useful when working with large datasets, as it makes the model more efficient and improves its ability to generalize to new data.

2. Max Depth: 6

The maximum depth of each tree was constrained to a value of 6. This limitation prevents the trees from growing too deep and learning overly complex relationships, which could result in overfitting. By controlling the depth, we strike a balance between capturing relevant patterns and avoiding the risk of the model memorizing the training data. Shallower trees force the model to focus on broader, more generalized patterns rather than fine-tuned details that may not apply to unseen data, thus enhancing the model's ability to generalize.

3. Learning Rate: 0.01

A small learning rate of 0.01 was chosen to ensure that the model learns gradually over time. In boosting algorithms like LightGBM, the learning rate controls how much each new tree contributes to correcting errors made by the previous trees. A lower learning rate ensures that the model adjusts more slowly and carefully, which reduces the risk of overfitting and leads to better generalization to new data. While this increases the number of iterations required to train the model, the improved accuracy on unseen data justifies the slower training process.

4. Number of Estimators: 200

The number of trees, or estimators, was set to 200. In ensemble learning methods like LightGBM, having more trees allows the model to capture more nuanced patterns in the data. However, there is a trade-off between performance and complexity. With 200 trees, the model was able to capture relevant patterns while avoiding excessive complexity that could lead to overfitting. Increasing the number of trees beyond 200 showed diminishing returns during the exploration phase, making this the optimal choice for this dataset.

5. Min Child Samples: 70

This parameter controls the minimum number of samples required to form a leaf node in the tree. Setting the value to 70 ensures that splits in the decision trees only occur when there is a sufficient number of data points to support the split. This helps prevent overfitting by avoiding the creation of overly specific rules that only apply to a small number of data points. With a higher value for min child samples, the model remains focused on more robust patterns that generalize well to new data, reducing the risk of capturing noise or outliers.

6. Metric: AUC (Area Under the Curve)

The AUC metric was chosen as the primary evaluation metric for the model. AUC is a measure of a classifier's ability to distinguish between classes, in this case, fraudulent and non-fraudulent transactions. It balances sensitivity (true positive rate) and specificity (true negative rate), making it particularly useful for imbalanced datasets like fraud detection, where the minority class (fraud) is much smaller than the majority class (non-fraud). By optimizing for AUC, the model is better equipped to correctly identify fraudulent transactions without flagging too many false positives.

7. Objective: Binary Classification

The objective of the LightGBM model was set to binary classification, which is appropriate for distinguishing between two categories: fraud and non-fraud. Binary classification ensures that the model focuses on correctly predicting one of the two possible outcomes for each transaction, which is the core requirement of the fraud detection task.

The combination of these hyperparameters was crucial in achieving the desired balance between model complexity and generalization. LightGBM's ability to handle imbalanced datasets is enhanced through careful tuning of parameters such as subsampling and learning rate, which ensure the model learns from the majority class (non-

fraud) without neglecting the minority class (fraud). The constraints on tree depth and minimum child samples further prevent the model from becoming too complex and overfitting to the training data. The use of AUC as the evaluation metric guarantees that the model is sensitive to both the precision and recall of fraud predictions, making it an ideal choice for scenarios where missing a fraudulent transaction is costly, but so is incorrectly flagging a legitimate transaction. Additionally, the small learning rate and sufficient number of trees allow the model to learn complex relationships while maintaining generalization to unseen data, as evidenced by its performance on the test and out-of-time datasets.

Model Performance Overview

The final LightGBM model was rigorously evaluated across three key datasets: the training set, test set, and out-of-time (OOT) set. Each of these datasets plays a distinct role in assessing the model's effectiveness and ability to generalize, ensuring that the model performs well not just on the data it has seen but also on unseen and future data.

1. **Training Set:**

The training set is the data that the model learns from, and its performance on this dataset provides an indication of how well it can detect patterns, particularly fraudulent transactions, in the data it was trained on. The training set is used to fit the model and teach it the relationship between features and the target (fraud or non-fraud).

2. **Test Set:**

The test set is a holdout dataset, meaning it is not used during the training process. Its main purpose is to evaluate how well the model generalizes to new, unseen data. The test set helps validate the model's generalization ability, providing a realistic measure of how well it can detect fraud in data it has never encountered before. This ensures that the model is not overfitting to the training set and can still perform well on new data.

3. **Out-of-Time Set:**

The out-of-time set is the most critical dataset in this evaluation. It represents future data, simulating how well the model can perform on data from a different time period, which often has slight shifts in patterns, including changes in fraud tactics. The OOT set is used to assess the model's robustness over time. Fraud detection systems in the real world must remain effective as new fraudulent patterns emerge. Evaluating the model on future data is essential to ensure that it remains reliable when deployed in production environments. By setting the desired OOT FDR at 0.56, the model was optimized to handle these shifts, ensuring it remains effective for future fraud detection.

Training, Testing, and Out-of-Time Performance:

1. Train Set Results:

Training	# Records	# Goods	# Bads	Fraud Rate										
	59,684	58,445	1,239	0.0208										
Bin Statistics						Cumulative Statistics								
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	30	567	5.03	94.97	597	30	567	0.05	45.76	45.71	0.05	90,800	580	90,220
2	597	260	337	43.55	56.45	1,194	290	904	0.5	72.96	72.47	0.32	1,47,600	2,860	1,44,740
3	597	476	121	79.73	20.27	1,791	766	1,025	1.31	82.73	81.42	0.75	1,60,800	7,300	1,53,500
4	596	541	55	90.77	9.23	2,387	1,307	1,080	2.24	87.17	84.93	1.21	1,65,200	12,200	1,53,000
5	597	578	19	96.82	3.18	2,984	1,885	1,099	3.23	88.7	85.48	1.72	1,71,600	17,000	1,54,600
6	597	578	19	96.82	3.18	3,581	2,463	1,118	4.21	90.23	86.02	2.2	1,74,800	21,960	1,52,840
7	597	579	18	96.98	3.02	4,178	3,042	1,136	5.2	91.69	86.48	2.68	1,78,000	26,920	1,51,080
8	597	585	12	97.99	2.01	4,775	3,627	1,148	6.21	92.66	86.45	3.16	1,81,600	31,840	1,49,760
9	597	591	6	98.99	1.01	5,372	4,218	1,154	7.22	93.14	85.92	3.66	1,82,400	36,920	1,45,480
10	596	595	1	99.83	0.17	5,968	4,813	1,155	8.24	93.22	84.99	4.17	1,83,200	42,000	1,41,200
11	597	595	2	99.66	0.34	6,565	5,408	1,157	9.25	93.38	84.13	4.67	1,84,800	47,040	1,37,760
12	597	591	6	98.99	1.01	7,162	5,999	1,163	10.26	93.87	83.6	5.16	1,86,000	52,100	1,33,900
13	597	593	4	99.33	0.67	7,759	6,592	1,167	11.28	94.19	82.91	5.65	1,86,800	57,160	1,29,640
14	597	593	4	99.33	0.67	8,356	7,185	1,171	12.29	94.51	82.22	6.14	1,87,600	62,240	1,25,360
15	597	594	3	99.5	0.5	8,953	7,779	1,174	13.31	94.75	81.44	6.63	1,89,600	67,260	1,22,340
16	596	594	2	99.66	0.34	9,549	8,373	1,176	14.33	94.92	80.59	7.12	1,89,600	72,380	1,17,220
17	597	596	1	99.83	0.17	10,146	8,969	1,177	15.35	95	79.65	7.62	1,90,400	77,460	1,12,940
18	597	591	6	98.99	1.01	10,743	9,560	1,183	16.36	95.48	79.12	8.08	1,91,200	82,520	1,08,680
19	597	594	3	99.5	0.5	11,340	10,154	1,186	17.37	95.72	78.35	8.56	1,91,600	87,620	1,03,980
20	597	591	6	98.99	1.01	11,937	10,745	1,192	18.38	96.21	77.82	9.01	1,91,600	92,740	98,860

Fig 7.1

The training dataset consists of 59,684 records, with 58,445 legitimate transactions and 1,239 fraudulent ones, giving a fraud rate of 2.08%. This is typical of real-world fraud scenarios, where fraudulent transactions make up only a small portion of the total data. As such, it's essential for the model to be highly accurate in identifying the small number of fraudulent cases while minimizing false positives, i.e., not wrongly classifying legitimate transactions as fraud.

Fraud Detection Rate (FDR):

The model achieved a Fraud Detection Rate (FDR) of 96.21% within the top 18.38% of predictions. This means the model was able to correctly identify 96.21% of fraudulent transactions by focusing on the top-scored cases. This high FDR reflects the model's strong ability to detect fraud without overfitting to the training data. It successfully captures most of the fraud cases in the high-risk set of predictions.

False Positive Rate (FPR):

The False Positive Rate (FPR) of 9.01% indicates that 9.01% of legitimate transactions were incorrectly flagged as fraudulent in the same top 18.38% of predictions. While false positives can result in financial or operational costs, this relatively low FPR demonstrates that the model balances accuracy with minimizing disruptions to legitimate transactions.

Kolmogorov-Smirnov (KS) Score:

The Kolmogorov-Smirnov (KS) score, which is a measure of the model's ability to distinguish between fraudulent and legitimate transactions, reaches a maximum of 96.21% at the 18.38% threshold. This high KS score indicates that the model is highly effective at separating fraud from non-fraud in the training data, making it well-suited for deployment in real-world scenarios where fraud detection is critical.

The table also includes financial metrics that reflect the real-world impact of the model's performance. Fraud Savings represents the potential financial savings from correctly identifying fraudulent transactions. As the model improves its detection rate, fraud savings increase, reaching $1,910,600 at the 18.38% threshold. This indicates significant value by preventing fraudulent transactions from causing further financial damage. On the other hand, the FP Loss represents the financial losses due to false positives—legitimate transactions wrongly flagged as fraudulent. In this case, the FP loss at the top 18.38% of predictions is $92,740, which quantifies the cost of disrupting legitimate activities. Finally, Overall Savings combines both fraud savings and false positive losses, providing a net financial impact of the model's performance. At the 18.38% threshold, the overall savings amount to $98,860, showing the model's net positive impact even when accounting for the costs associated with false positives. This indicates the model's practical utility in striking a balance between maximizing fraud detection and minimizing unnecessary disruptions to legitimate transactions.

2. Test Set Results:

Testing	# Records	# Goods	# Bads	Fraud Rate										
	25,580	25,069	511	0.02										
Bin Statistics						Cumulative Statistics								
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	29	227	0.12%	88.67%	256	29	227	0.12	44.42	44.31	0.13	2,26,800	600	2,26,200
2	256	114	142	0.57%	55.47%	512	143	369	0.57	72.21	71.64	0.39	3,61,600	5,800	3,55,800
3	255	222	33	1.46%	12.94%	767	365	402	1.46	78.67	77.21	0.91	4,10,000	15,320	3,94,680
4	256	245	11	2.43%	4.30%	1,023	610	413	2.43	80.82	78.39	1.48	4,32,000	26,140	4,05,860
5	256	240	16	3.39%	6.25%	1,279	850	429	3.39	83.95	80.56	1.98	4,39,600	37,700	4,01,900
6	256	248	8	4.38%	3.13%	1,535	1,098	437	4.38	85.52	81.14	2.51	4,47,200	49,260	3,97,940
7	256	248	8	5.37%	3.13%	1,791	1,346	445	5.37	87.08	81.71	3.02	4,54,400	60,840	3,93,560
8	255	246	9	6.35%	3.53%	2,046	1,592	454	6.35	88.85	82.49	3.51	4,59,200	72,540	3,86,660
9	256	254	2	7.36%	0.78%	2,302	1,846	456	7.36	89.24	81.87	4.05	4,61,600	84,360	3,77,240
10	256	254	2	8.38%	0.78%	2,558	2,100	458	8.38	89.63	81.25	4.59	4,62,000	96,260	3,65,740
11	256	252	4	9.38%	1.56%	2,814	2,352	462	9.38	90.41	81.03	5.09	4,62,800	1,08,160	3,54,640
12	256	253	3	10.39%	1.17%	3,070	2,605	465	10.39	90.99	80.61	5.6	4,65,200	1,19,980	3,45,220
13	255	253	2	11.40%	0.78%	3,325	2,858	467	11.4	91.39	79.99	6.12	4,66,800	1,31,840	3,34,960
14	256	254	2	12.41%	0.78%	3,581	3,112	469	12.41	91.78	79.37	6.64	4,68,400	1,43,700	3,24,700
15	256	251	5	13.41%	1.95%	3,837	3,363	474	13.41	92.76	79.34	7.09	4,69,600	1,55,580	3,14,020
16	256	256	0	14.44%	0.00%	4,093	3,619	474	14.44	92.76	78.32	7.64	4,70,400	1,67,460	3,02,940
17	256	254	2	15.45%	0.78%	4,349	3,873	476	15.45	93.15	77.7	8.14	4,70,800	1,79,380	2,91,420
18	255	253	2	16.46%	0.78%	4,604	4,126	478	16.46	93.54	77.08	8.63	4,73,200	1,91,200	2,82,000
19	256	255	1	17.48%	0.39%	4,860	4,381	479	17.48	93.74	76.26	9.15	4,74,400	2,03,080	2,71,320
20	256	256	0	18.50%	0.00%	5,116	4,637	479	18.5	93.74	75.24	9.68	4,76,800	2,14,900	2,61,900

Fig 7.2

The test set comprises 25,580 records, of which 25,069 are legitimate transactions and 511 are fraudulent, resulting in a fraud rate of 2.0%. This fraud rate is representative of the real-world scenario, where fraud detection models are often tested against imbalanced datasets with a low percentage of fraudulent transactions.

Fraud Detection Rate (FDR):

In the test set, the model achieved a Fraud Detection Rate (FDR) of 90.99% in the top 12.01% of predictions (bin 12). This indicates that the model correctly identified 90.99% of the fraudulent transactions within the highest predicted cases. This performance metric reflects the model's ability to generalize to new data while maintaining a high level of accuracy in detecting fraud. This high FDR is crucial for ensuring that the model is reliable and effective in real-world applications.

False Positive Rate (FPR):

The model demonstrated a False Positive Rate (FPR) of 5.6% at the 12.01% threshold, meaning that 5.6% of the legitimate transactions were misclassified as fraudulent within this set of predictions. While false positives can lead to operational costs, this relatively low FPR signifies the model's capability to minimize disruptions to legitimate transactions while still focusing on identifying the high-risk cases effectively.

Kolmogorov-Smirnov (KS) Score:

The KS score for the test set is 90.99%, measured in the top 12.01% of predictions (bin 12). The KS score measures the model's ability to distinguish between the fraudulent and legitimate transactions, and this high value indicates that the model is highly effective at separating the two classes. A high KS score confirms that the model performs well not only on the training set but also generalizes to unseen test data.

Financial Metrics: In terms of financial impact, the model's ability to detect fraud and minimize false positives translates into measurable savings and losses.

- Fraud Savings reflects the potential savings from successfully identifying fraudulent transactions. At the top 12.01% threshold, the model achieves $4,650,200 in fraud savings, showing the model's efficacy in preventing financial losses due to fraud.
- FP Loss represents the cost of false positives, where legitimate transactions are mistakenly flagged as fraudulent. At the top 12.01% threshold, the model incurs an FP loss of $1,199,980, which highlights the trade-off between high fraud detection and the operational costs of handling false positives.
- Overall Savings, which combines both the Fraud Savings and the FP Loss, provides the net financial benefit of deploying the model. In the top 12.01% of predictions, the model's overall savings amount to $3,452,220, signifying a substantial net positive financial impact despite the costs associated with false positives.

At higher thresholds (e.g., 18.5%), the fraud savings increase to $4,760,800, but at the cost of a higher FP Loss of $2,140,900, reducing the overall savings to $2,619,900. This balance between fraud detection, false positives,

and financial savings helps optimize the model's real-world application, ensuring it maximizes its financial impact while maintaining accuracy in fraud detection.

Overall, the test set results demonstrate that the model generalizes well to new data, achieving high fraud detection rates, maintaining low false positives, and generating significant financial savings. This balanced performance makes the model highly suitable for deployment in real-world fraud detection systems.

3. Out-of-Time (OOT) Set Results:

OOT	# Records	# Goods	# Bads	Fraud Rate							
	12,232	11,935	297	0.0243							
Bin Statistics						Cumulative Statistics						
Population Bin %	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	122	24	98	19.67	80.33	122	24	98	0.2	33	32.8	0.24
2	123	92	31	74.8	25.2	245	116	129	0.97	43.43	42.46	0.9
3	122	78	44	63.93	36.07	367	194	173	1.63	58.25	56.62	1.12
4	122	111	11	90.98	9.02	489	305	184	2.56	61.95	59.4	1.66
5	123	110	13	89.43	10.57	612	415	197	3.48	66.33	62.85	2.11
6	122	110	12	90.16	9.84	734	525	209	4.4	70.37	65.97	2.51
7	122	114	8	93.44	6.56	856	639	217	5.35	73.06	67.71	2.94
8	123	107	16	86.99	13.01	979	746	233	6.25	78.45	72.2	3.2
9	122	117	5	95.9	4.1	1,101	863	238	7.23	80.13	72.9	3.63
10	122	119	3	97.54	2.46	1,223	982	241	8.23	81.14	72.92	4.07
11	123	119	4	96.75	3.25	1,346	1,101	245	9.22	82.49	73.27	4.49
12	122	119	3	97.54	2.46	1,468	1,220	248	10.22	83.5	73.28	4.92
13	122	121	1	99.18	0.82	1,590	1,341	249	11.24	83.84	72.6	5.39
14	122	118	4	96.72	3.28	1,712	1,459	253	12.22	85.19	72.96	5.77
15	123	120	3	97.56	2.44	1,835	1,579	256	13.23	86.2	72.97	6.17
16	122	122	0	100	0	1,957	1,701	256	14.25	86.2	71.94	6.64
17	122	120	2	98.36	1.64	2,079	1,821	258	15.26	86.87	71.61	7.06
18	123	122	1	99.19	0.81	2,202	1,943	259	16.28	87.21	70.93	7.5
19	122	121	1	99.18	0.82	2,324	2,064	260	17.29	87.54	70.25	7.94
20	122	122	0	100	0	2,446	2,186	260	18.32	87.54	69.23	8.41

Fig 7.3

The out-of-time (OOT) dataset consists of 12,232 records, with 11,935 legitimate transactions and 297 fraudulent transactions, resulting in a fraud rate of 2.43%. The OOT dataset is critical in evaluating the model's ability to generalize to unseen, future data, ensuring that it remains effective as fraud patterns evolve over time.

Fraud Detection Rate (FDR)

The model achieved a Fraud Detection Rate (FDR) of 83.5% at the top 12.01% of the predictions (bin 12). This high FDR indicates that the model successfully captured 83.5% of the fraudulent transactions in the top-scored set of predictions, demonstrating its ability to maintain effective fraud detection even in future data. A high FDR is particularly important for ensuring the model remains reliable over time as it adapts to emerging fraud tactics.

False Positive Rate (FPR)

The False Positive Rate (FPR) at the 12.01% threshold is 4.92%, meaning that 4.92% of legitimate transactions were misclassified as fraudulent. This relatively low FPR indicates the model's capability to maintain a balance

between detecting fraud and avoiding excessive false positives, even in future data. The goal is to minimize disruptions to legitimate transactions while still focusing on the highest-risk cases.

Kolmogorov-Smirnov (KS) Score

The KS score at the 12.01% threshold is 73.28%, showing the model's effectiveness in distinguishing between fraudulent and legitimate transactions. A high KS score reflects the model's strong ability to separate the two classes in the OOT dataset, further supporting its generalizability and long-term robustness.

Financial Metrics

The financial impact of the model's performance in the OOT dataset is reflected in several key metrics:

- Fraud Savings shows the potential savings generated by correctly identifying fraudulent transactions. At the top 12.01% threshold, the fraud savings amount to $4,180,000, highlighting the significant financial benefit of deploying the model in real-world applications, where the cost of undetected fraud can be substantial.
- FP Loss measures the cost associated with false positives, where legitimate transactions are incorrectly flagged as fraudulent. At the 12.01% threshold, the FP loss is $248,500, which represents the operational costs and customer inconvenience incurred due to the false positive rate.
- Overall Savings, which takes into account both the fraud savings and the false positive losses, provides the net financial benefit of the model. In the top 12.01% of predictions, the overall savings are $3,931,500, indicating the model's net positive financial impact even when considering the costs associated with false positives.

As the model moves up the population bins (toward 18.32% of records in bin 20), the fraud savings remain at $4,180,000, but the false positive loss increases to $8,410, resulting in lower overall savings. However, the model still generates significant overall financial benefit, demonstrating that it remains effective in future scenarios.

In summary, the OOT evaluation confirms that the model generalizes well to unseen data, maintaining a strong fraud detection rate while minimizing false positives and maximizing financial savings. These results support the model's deployment in real-world fraud detection systems, where evolving fraud patterns and unseen data are constant challenges.

Overall, the final LightGBM model offers an optimal balance between high performance and strong generalization. Its ability to maintain high FDRs across all datasets, particularly the OOT set, makes it well-suited for deployment in real-world fraud detection systems. The model's performance on the OOT set further highlights its robustness and ability to adapt to future data patterns, ensuring long-term reliability in identifying fraudulent transactions. With its high fraud detection rates and controlled false positive rates, this LightGBM model is an excellent choice for operational fraud detection systems, where the goal is to minimize losses from fraud while maintaining customer trust.

Chapter 8: Financial Curves and Recommended Cutoff

In fraud detection, striking a balance between catching fraudulent transactions and minimizing operational costs is crucial. The following analysis focuses on the financial impact of varying score cutoffs in the model, illustrated by three key financial curves: Fraud Savings (green line), False Positive Loss (red line), and Overall Savings (blue line). These curves help quantify the trade-offs between detecting fraud and the costs associated with incorrectly flagging legitimate transactions (false positives).

By analyzing these curves, we can determine the optimal score cutoff - where the model achieves the highest financial benefit, factoring in both fraud savings and false positive losses.

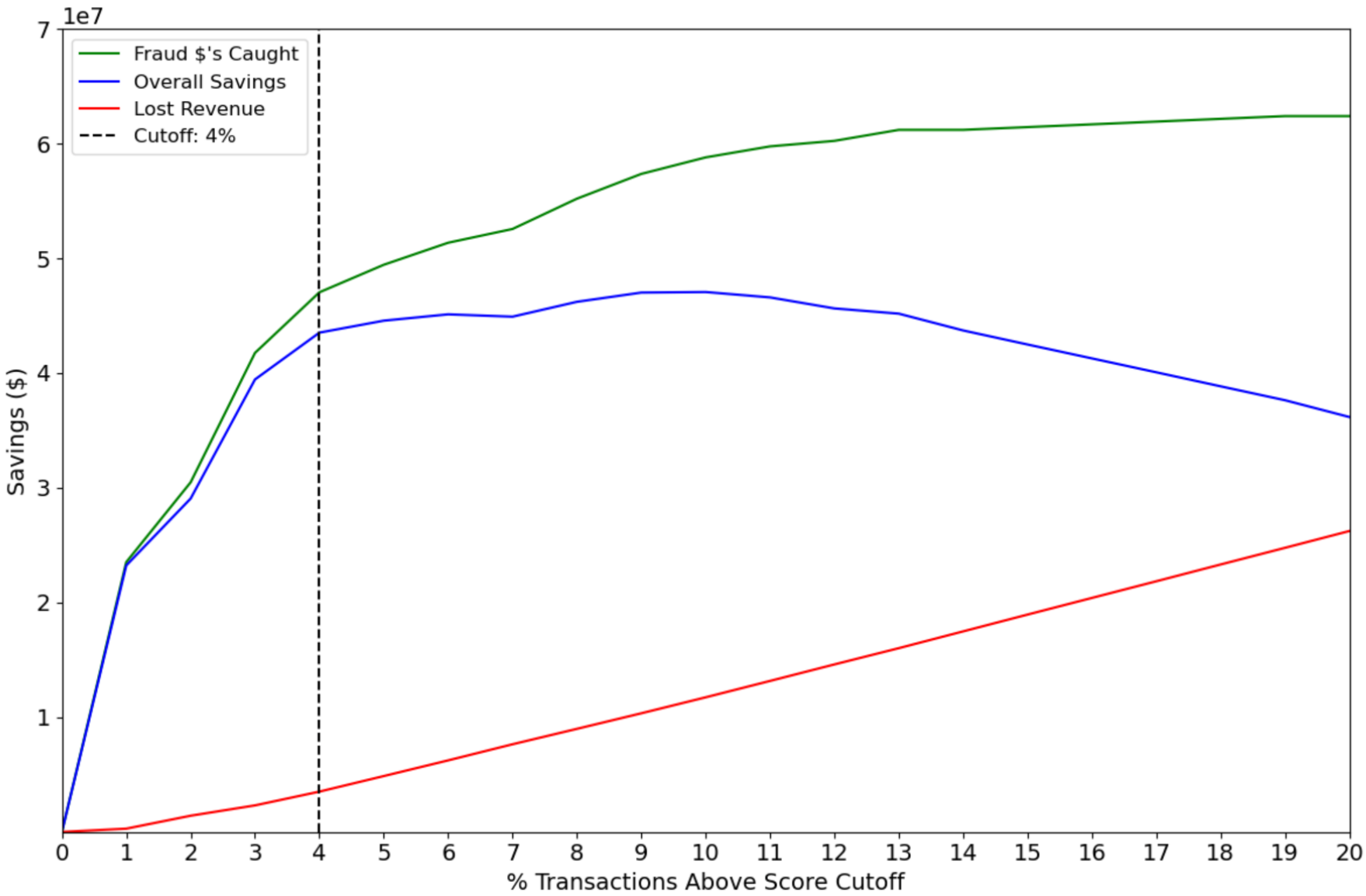


Fig 8.1

Explanation of Financial Curves:

The graph visualizes how changes in the score cutoff impact fraud savings, false positive losses, and overall financial savings. Here's a detailed breakdown:

1. Fraud Savings Curve (Green Line)

- **Initial Steep Growth (1–3% Cutoff)**: The Fraud Savings curve rises sharply at the beginning, as the model effectively identifies a substantial portion of fraudulent transactions at low score cutoffs. These early savings are highly impactful because the model catches many high-risk fraud cases without flagging too many legitimate transactions.
- **Plateau Effect (3–6% Cutoff)**: After about a 4% cutoff, the curve starts to level off. This indicates that while the model continues to catch more fraud, the incremental savings diminish. This is because many of the highest-

risk fraud cases are already detected at the lower cutoffs, and the remaining flagged transactions are increasingly marginal in their impact on savings.

- **Diminishing Returns**: Beyond the 6% cutoff, the gains from detecting additional fraudulent transactions become less significant, as most of the fraud has already been detected. This shows that the model's ability to deliver incremental savings weakens at higher cutoffs.

2. False Positive Loss Curve (Red Line)

- **Gradual Increase**: The **False Positive Loss** curve shows a consistent upward trend. As the score cutoff increases, the model begins to flag more legitimate transactions as fraudulent. This results in increased operational costs, customer dissatisfaction, and lost revenue. Each false positive comes with a cost of $20, and as more legitimate transactions are flagged, the total cost accumulates.
- **Linear Growth**: The linear nature of this curve underscores that as we raise the cutoff, the financial loss due to false positives continues to grow steadily. This emphasizes the importance of not overburdening the system with too high of a cutoff, as the associated costs can quickly negate the benefits of detecting more fraud.

3. Overall Savings Curve (Blue Line)

- **Rapid Growth (1–4% Cutoff)**: The **Overall Savings** curve, which represents the net financial benefit of fraud detection, rises quickly at lower score cutoffs. At this stage, the model is detecting the most fraud with minimal false positive losses, leading to a rapid increase in overall savings.
- **Peak Savings (Around 4% Cutoff)**: The blue curve peaks at about a 4% cutoff, indicating that this is the optimal point where fraud detection is maximized while false positive losses are minimized. At this point, the model generates the highest possible savings for the business.
- **Declining Returns (Beyond 4% Cutoff)**: After reaching the peak, the curve starts to decline. This decline occurs because, while the model continues to catch fraud, the cost of false positives increases at a faster rate than the additional fraud savings. As a result, the net financial benefit decreases, marking the point where increasing the cutoff would no longer be profitable.

Key Insights from Financial Curves:

The financial curves provide critical insights into how the model's performance translates into financial outcomes. These insights help inform the recommendation for the best score cutoff.

Optimal Cutoff for Fraud Detection

- **Recommended Cutoff**: The recommended score cutoff is around 4%. This is the point where the overall savings (blue line) peak, indicating that the business is maximizing its financial return from the fraud detection system.
- **Rationale**: Beyond 4%, the marginal gains in fraud detection are outweighed by the growing costs of false positives. Setting the cutoff higher than 4% would lead to increased operational costs and customer dissatisfaction, without a corresponding increase in fraud savings.

Maximizing Fraud Savings

- **High Fraud Detection Early**: The Fraud Savings curve (green line) shows that the majority of fraud savings are captured at low cutoffs (1–4%). This demonstrates that the model is highly effective at identifying the riskiest transactions early on, which translates to significant financial savings for the business.
- **Diminishing Returns at High Cutoffs**: The flattening of the green line beyond a 4% cutoff suggests that the model is reaching its limit in terms of fraud detection. As the cutoff increases, the additional fraud caught becomes less significant in terms of financial savings.

Controlling False Positives

- **False Positive Costs Accumulate**: The False Positive Loss curve (red line) shows a steady increase in costs as the cutoff increases. Every additional percentage point of cutoff results in more legitimate transactions being flagged as fraud, which carries an operational cost of $20 per false positive.
- **Balancing Act**: The key challenge is to balance fraud detection with minimizing the cost of false positives. The red curve highlights the importance of keeping the cutoff at a point where false positives remain manageable, ensuring that the business does not incur unnecessary losses.

Final Recommendation:

Based on the analysis of the financial curves, we recommend setting the fraud detection score cutoff at 4%. This cutoff point:

- **Maximizes overall savings** (as shown by the blue curve), providing the business with the highest possible financial benefit from the fraud detection model.
- **Minimizes false positive losses**, ensuring that the business does not incur excessive operational costs or customer dissatisfaction due to false alarms.

At the 4% cutoff, the business can expect to achieve **approximately $47,064,000** in annual fraud savings. This projection is based on the balance between fraud detection and false positive losses, ensuring that the model continues to deliver a high return on investment.

Graph Summary

The green line shows Fraud Savings, which peaks early and then levels off as the model catches most of the significant fraud early. The red line represents False Positive Loss, which steadily increases as more legitimate transactions are incorrectly flagged as fraud. The blue line represents Overall Savings, which peaks at 4% - this is the optimal balance between catching fraud and avoiding excessive false positives.

By following these recommendations, the business can ensure that its fraud detection system remains financially optimized while maintaining high detection rates, effectively safeguarding against fraudulent transactions without incurring excessive operational costs.

Chapter 9: Summary

This report presents the comprehensive analysis and findings of our credit card fraud detection project for the academic term of Fall 2024, utilizing a dataset consisting of 97,852 real credit card transactions from January 1, 2010, to December 31, 2010. The project encompassed several key phases: data cleaning, variable creation, feature selection, model exploration, and evaluation, culminating in the application of machine learning techniques to effectively predict and analyze fraudulent transactions.

Data Cleaning and Preparation The initial step involved rigorous data cleaning and preprocessing, where the dataset was refined by addressing missing values, removing outliers, and filtering irrelevant transactions. This phase ensured the data's suitability for creating robust predictive models, focusing on enhancing data quality and relevance to fraud detection.

Variable Creation and Feature Engineering We engineered a diverse set of features to enrich the model's input, including day-of-week variables, entity combination variables, and various time-window-based metrics. These features were designed to capture subtle patterns and relationships indicative of fraudulent activity, such as unusual transaction frequencies or atypical combinations of merchant and card numbers.

Feature Selection Feature selection was performed using techniques like LightGBM and Random Forest, which helped reduce dimensionality and improve model efficiency. This step was crucial in identifying the most predictive features, thereby optimizing the models' performance and computational efficiency.

Model Development and Evaluation We explored various machine learning models, with a particular focus on ensemble methods like Random Forest and boosting algorithms such as LightGBM and XGBoost. These models were evaluated based on their ability to predict fraud within the top 3% of riskiest transactions, measured by the Fraud Detection Rate (FDR). The best-performing model, LightGBM, achieved an FDR of 93.74% on the test set and 87.54% in an out-of-time validation set, showcasing its robustness and reliability.

Financial Impact and Optimization The financial impact of the fraud detection models was analyzed through financial curves, which helped determine the optimal score cutoff to balance fraud savings against false positives. The recommended cutoff at 4% maximized net savings, achieving substantial annual fraud savings estimated at $47,064,000. This balance underscores the model's practical value in a real-world business context, where reducing fraud-related losses while maintaining customer trust is paramount.

Further Improvements and Research Potential areas for further improvement include exploring alternative imbalanced learning techniques, such as synthetic data generation through SMOTE, and refining feature engineering processes. Additionally, real-time model deployment and continuous learning frameworks could be developed to adapt to evolving fraud patterns, ensuring sustained effectiveness of the fraud detection system.

Conclusion Overall, the project successfully demonstrates the application of advanced analytics and machine learning in detecting and preventing credit card fraud, with significant implications for operational efficiency and financial savings. Future work will focus on enhancing model adaptability and exploring cutting-edge techniques to stay ahead of sophisticated fraudulent tactics.

Chapter 10: Appendix

Data Quality Report on the Credit Card Transaction Data

Data Overview and Description:

This dataset comprises 97,852 records of credit card transactions spanning from January 1, 2010, to December 31, 2010. The data was originally sourced from the website of Mark Nigrini, a noted expert in forensic accounting, and has been curated from internal transaction logs for analytical purposes. Each record features 10 distinct fields, including transaction amounts, merchant details, transaction types, and fraud indicators.

Designed to support in-depth analysis, this dataset offers a valuable foundation for examining spending behaviors and enhancing fraud detection strategies. It enables robust insights for strategic decision-making related to transaction monitoring, security measures, and risk management.

Field Summary Tables:

Numeric Field Summary:

Field Name	Field Type	# Records Have Values	% Populated	# Zeros	Min	Max	Mean	Standard Deviation	Most Common
Amount	Numeric	97,852	100%	0	0	3,102,046	425.5	9,949.8	3.6

Categorical Field Summary:

Field Name	Field Type	# Records Have Values	% Populated	# Zeros	# Unique Values	Most Common
Cardnum	Categorical	97,852	100%	0	1,645	5142148452
Date	Categorical	97,852	100%	0	365	2/28/2010
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	38118
Transtype	Categorical	97,852	100%	0	4	P
Fraud	Categorical	97,852	100%	95,805	2	0

Field Descriptions and Visualizations:

1) Recnum: Represents a unique record number for each transaction.

Although a distribution plot is not necessary for "Recnum" due to the uniqueness of each value, the line plot is included to visually demonstrate the sequential nature of transaction records, providing insight into the temporal order and volume of transactions over time.

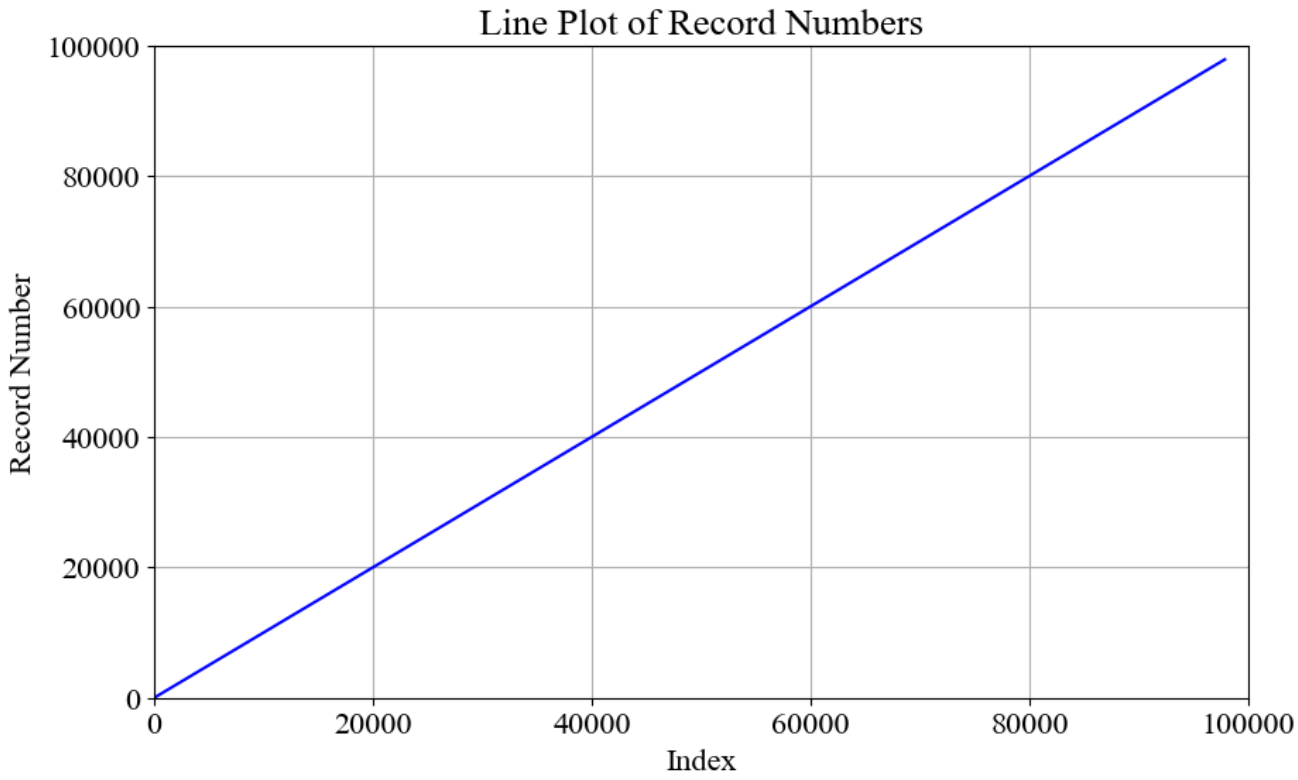


Fig 10.1

2) Cardnum: Represents the credit card number associated with each transaction.

Bar Chart: Displays the top 10 most frequently used credit card numbers.

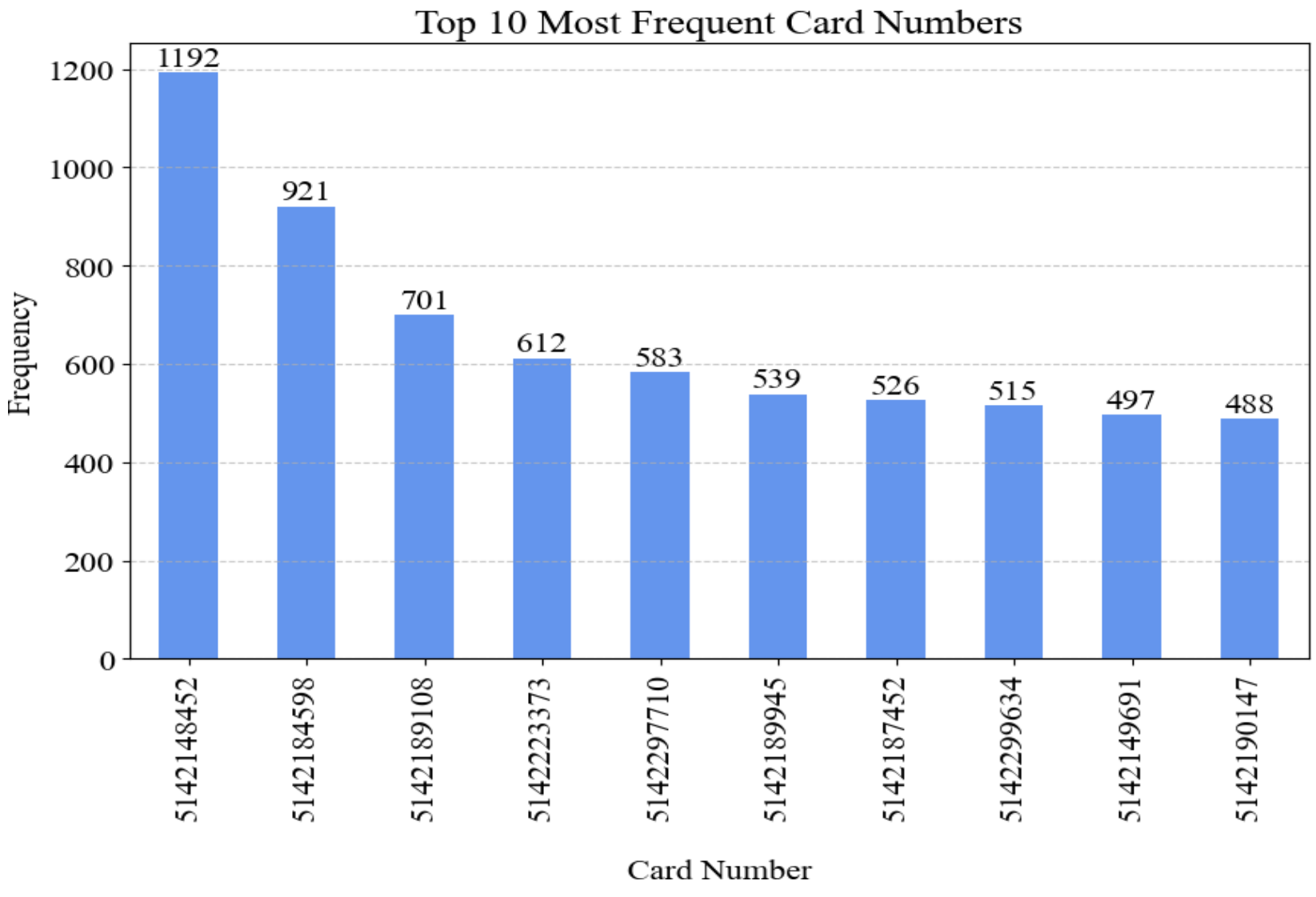


Fig 10.2

3) Date: The date of the transaction.

Date Plot: The plot shows the number of transactions by date over the year 2010.

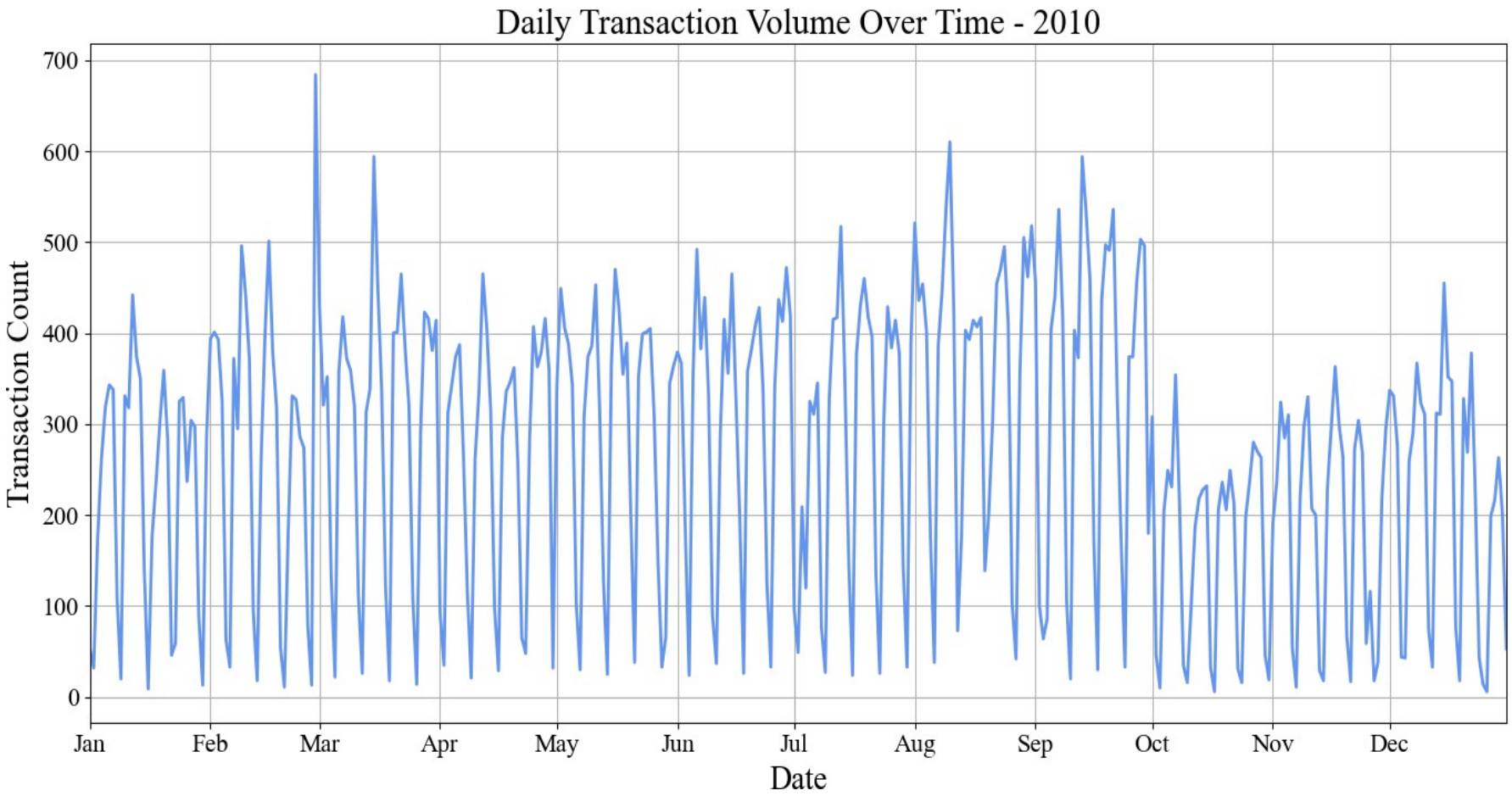


Fig 10.3

4) Merchnum: Represents the merchant number associated with each transaction.

Bar Chart: Displays the top 10 most frequent merchant numbers.

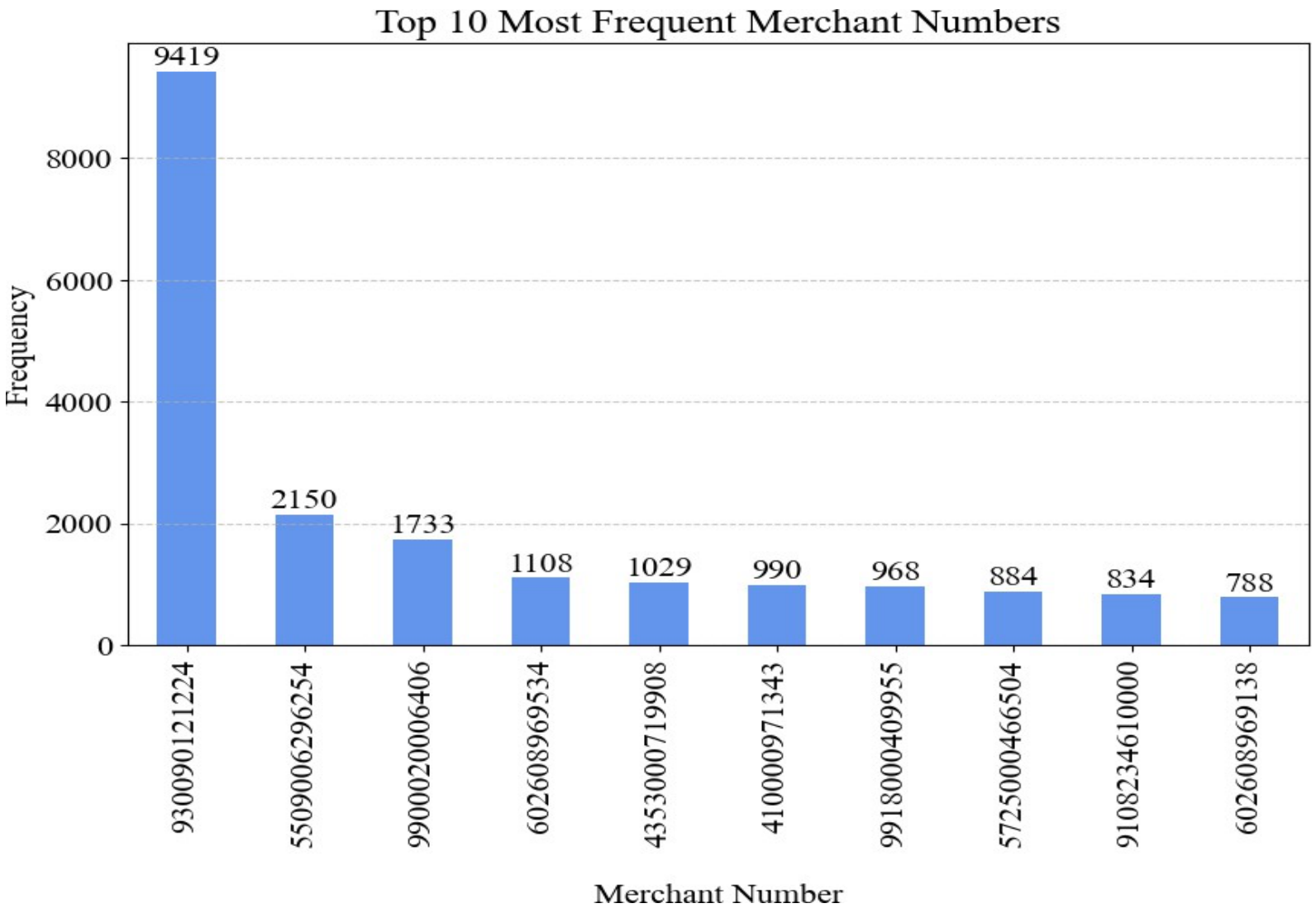


Fig 10.4

5) Merch Description: Describes the merchant for each transaction.

Bar Chart: Highlights the top 10 most common merchant descriptions.

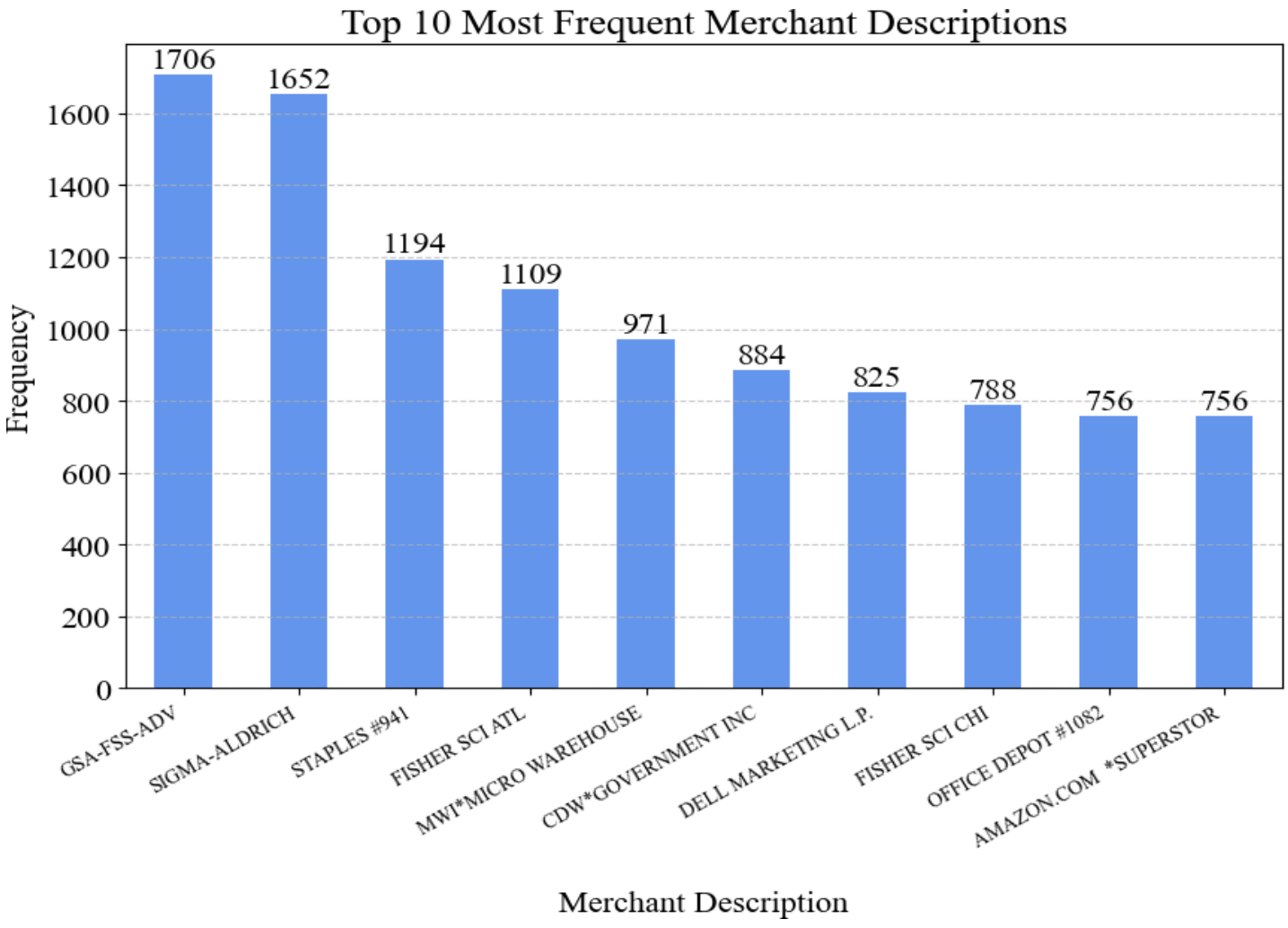


Fig 10.4

6) Merch State: Represents the state where the merchant is located.

Bar Chart: Illustrates the top 10 states with the highest number of transactions.

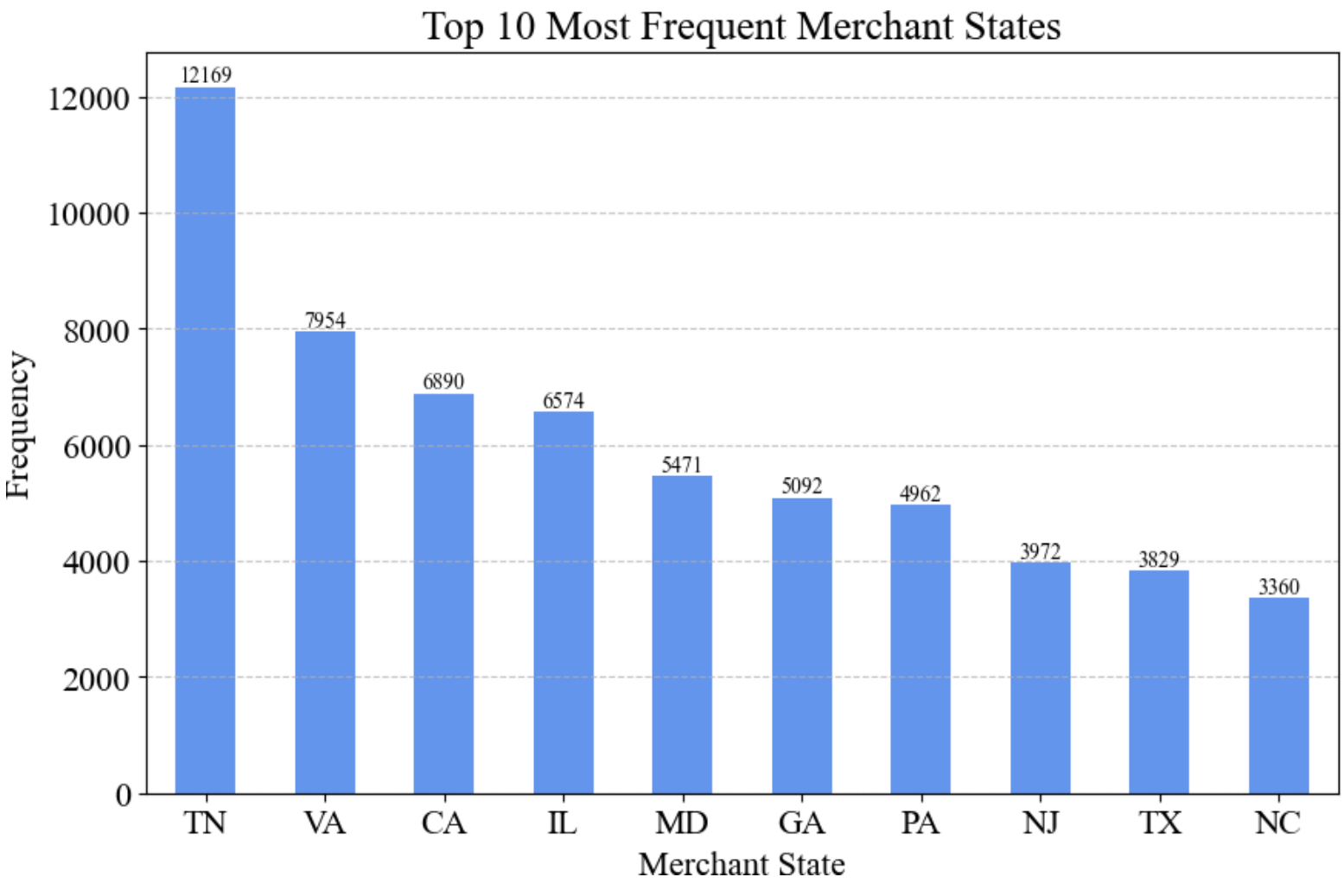


Fig 10.5

7) Merch Zip: Represents the ZIP code of the merchant.

Bar Chart: Highlights the top 10 most common merchant ZIP codes.

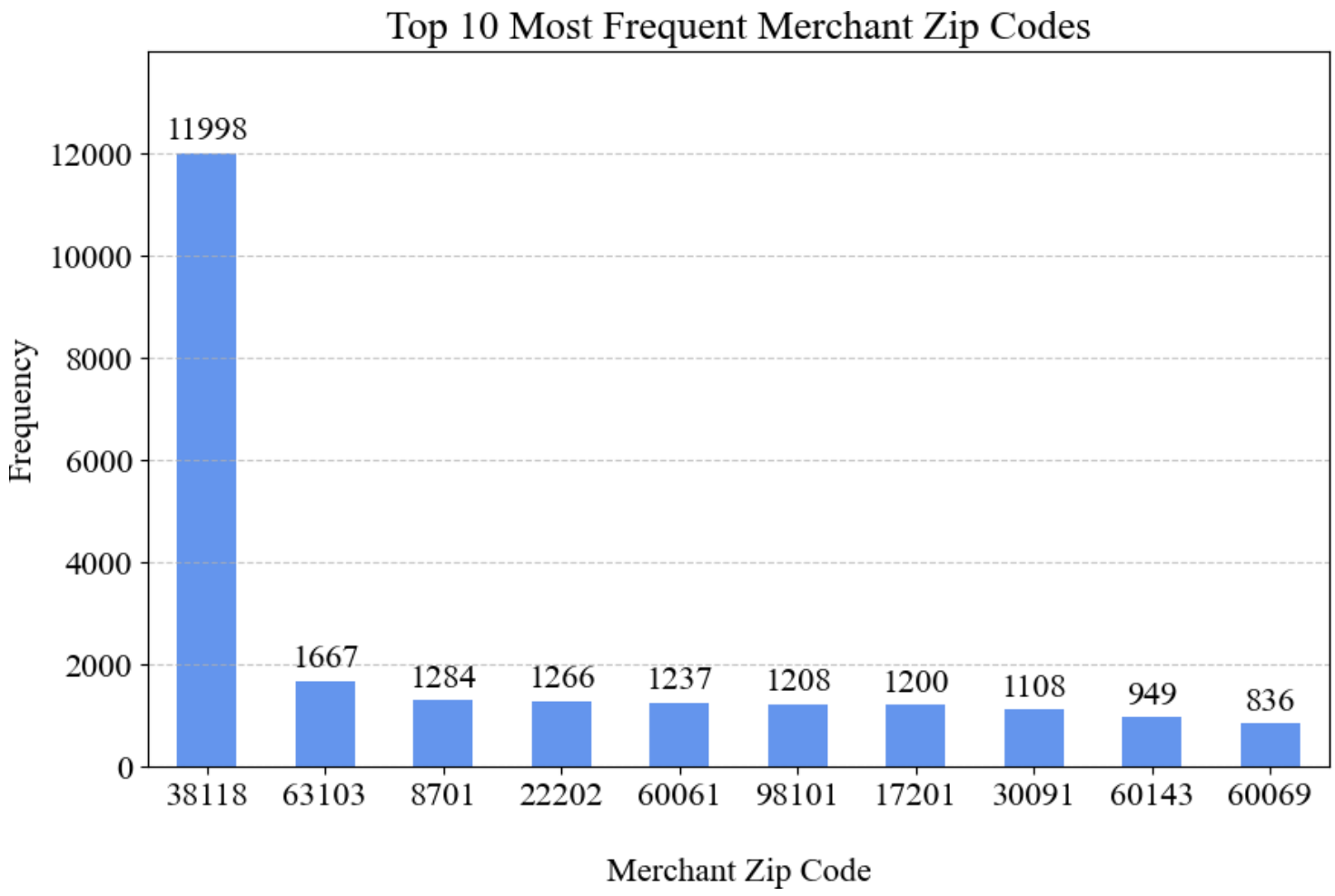


Fig 10.6

8) Transtype: Represents the type of transaction.

Bar Chart: Represents the distribution of transaction types.

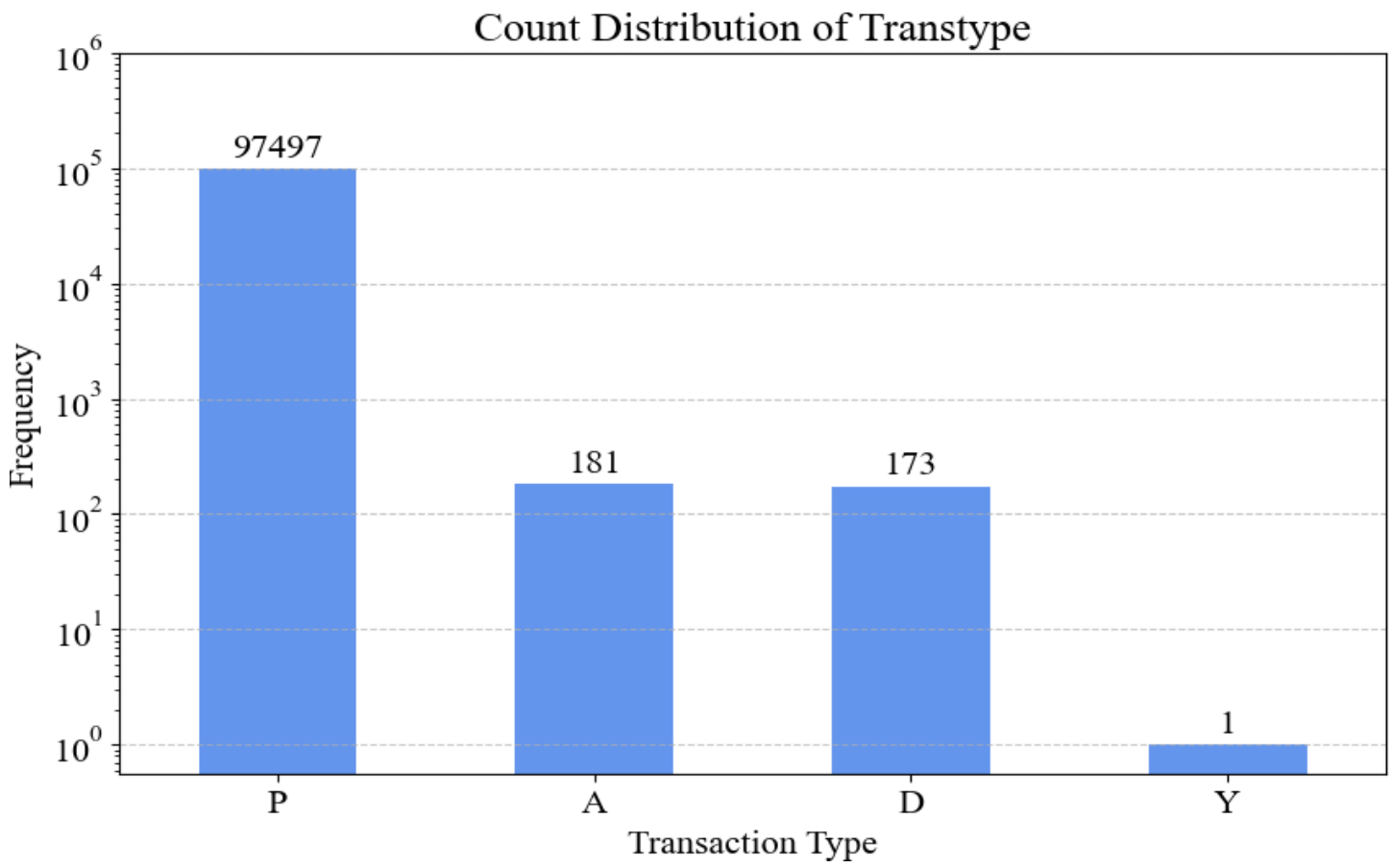


Fig 10.7

9) Amount: Represents the transaction amount in USD.

Histogram: This shows the frequency of transaction amounts up to $3,000, capturing 99% of the records. Most transactions are small, with a steep decline after $500, and a small spike between $2,500 and $3,000.

Box Plot: This summarizes the data, with the median marked inside the box and whiskers representing the range within 1.5 times the interquartile range. Many high-value outliers are visible, indicating several large transactions beyond the typical range.

.

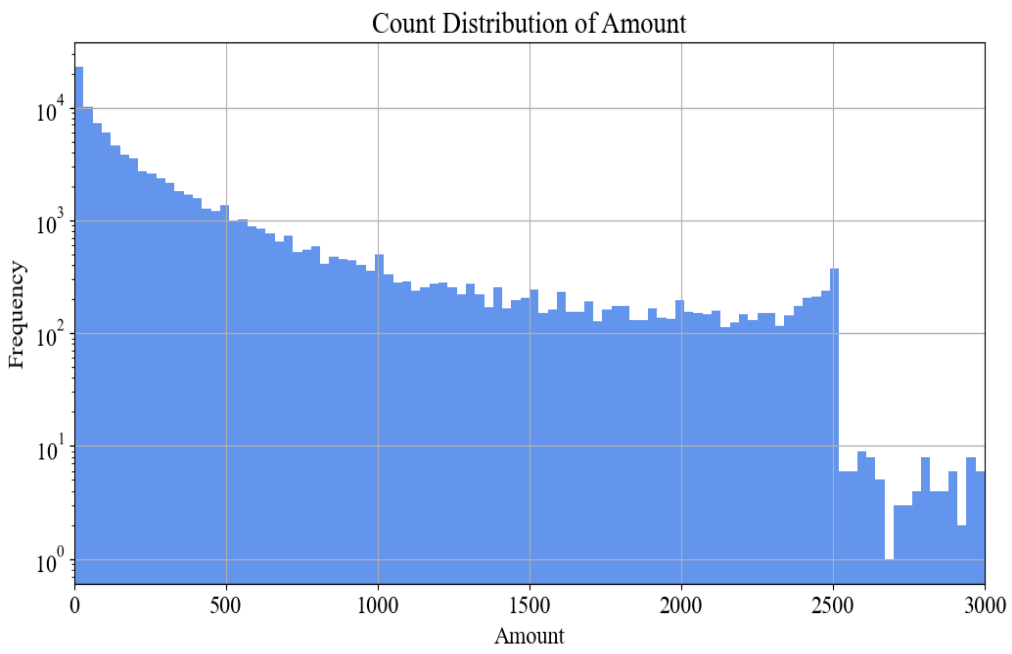


Fig 10.8.1

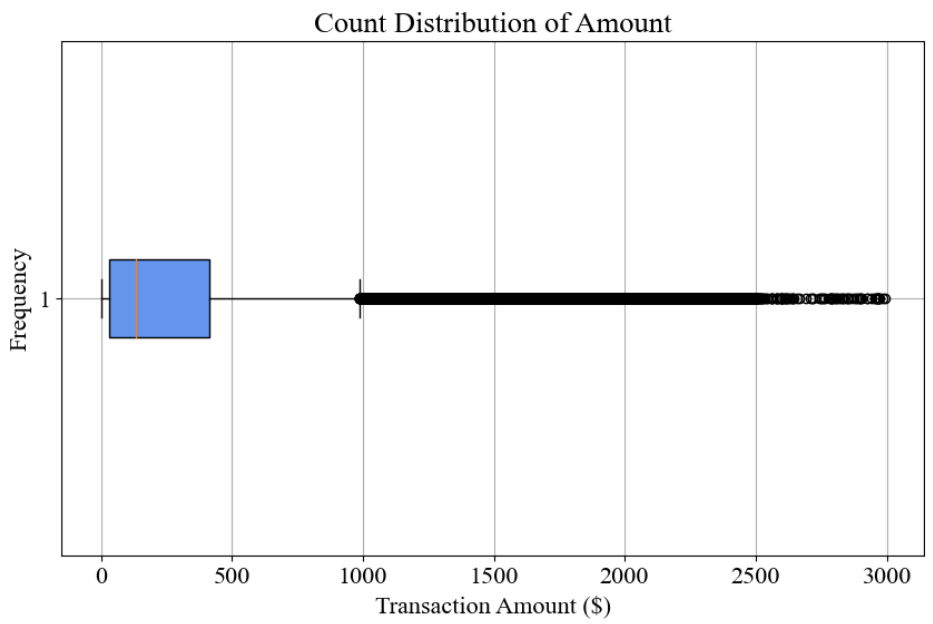


Fig 10.8.2

10) Fraud: Indicates whether the transaction was fraudulent (1) or not (0).

Bar Chart: Shows the count of fraudulent and non-fraudulent transactions.

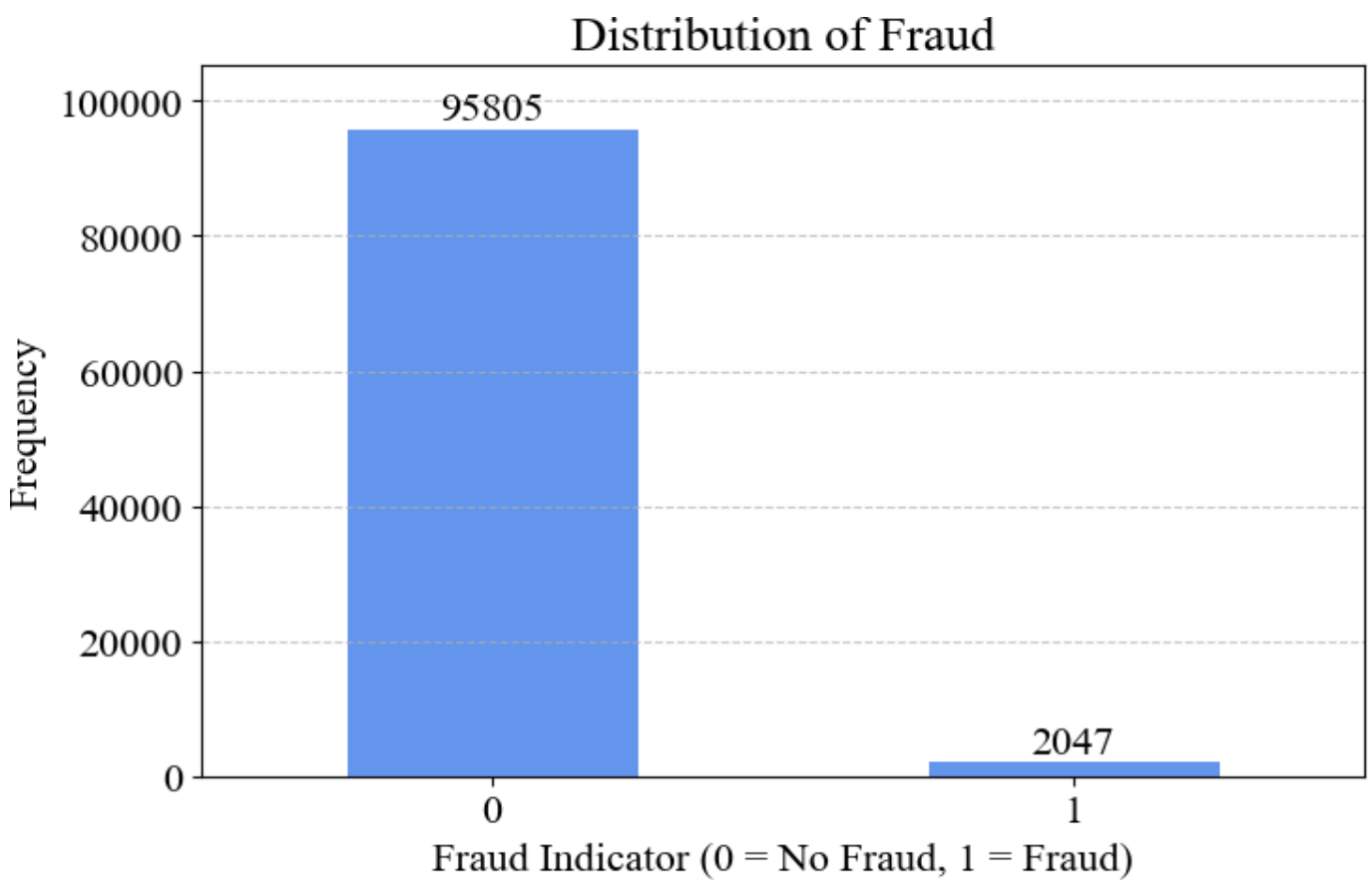


Fig 10.9

The dataset is largely complete, making it well-suited for in-depth analysis, with most fields being fully populated and showing minimal instances of missing or zero values. This completeness allows for reliable statistical analyses and model development without the need for extensive data imputation or handling of missing data. Core variables such as transaction amounts, dates, and fraud indicators are critical for identifying patterns in spending behavior, allowing the detection of anomalous transactions that may indicate fraud. The presence of these essential variables provides a solid starting point for building fraud detection models, as they directly relate to the financial behavior of cardholders and merchants.

However, there are some challenges that need to be addressed in the preprocessing phase. Variables like "Merchnum" (Merchant Number) and "Merch Zip" (Merchant ZIP Code) exhibit high levels of uniqueness, meaning that many values appear only once or very infrequently. This high cardinality can hinder the discovery of patterns or general trends. To derive more meaningful insights from these variables, techniques like aggregation (grouping merchants by category or location), dimensionality reduction, or encoding methods (such as target encoding) can be employed. These steps would help to reduce the noise created by overly unique values and make the data more manageable for machine learning algorithms.

Another important consideration is the presence of outliers in the "Amount" field, where unusually high or low transaction amounts may distort model training and predictions. Careful evaluation of these outliers is required to determine whether they represent genuine anomalies (e.g., fraudulent transactions or rare but legitimate large purchases) or data errors that need to be corrected or removed. Various strategies, such as capping extreme values, creating separate outlier categories, or transforming the data (e.g., using log transformations), may help in managing these outliers effectively without sacrificing important signals.

Despite these preprocessing challenges, the dataset offers a strong foundation for developing advanced fraud detection models. Transaction trends, spending behavior, and potential anomalies can be effectively analyzed to improve fraud detection mechanisms. Future efforts to enhance the dataset could involve enriching it with additional variables such as customer demographics, merchant categories, or geographic data, which could offer further dimensions for analysis. Refining data quality by addressing outliers, missing values, and high cardinality variables would further improve model performance and reliability.

Machine learning techniques like decision trees, random forests, and gradient boosting models, combined with anomaly detection methods, can be employed to detect fraudulent patterns more accurately. These models can leverage the key transaction variables and identify subtle relationships between them that may indicate fraud, ultimately enhancing security measures and minimizing financial risk for financial institutions and customers alike.