**AI-POWERED FRAUD DETECTION IN FINANCIAL TRANSACTIONS**

**MINI PROJECT REPORT**

***Submitted By***

**GIRISH J R 211501028**

**GOURI M P 211501030**

**HARINI B 211501031**

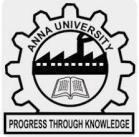
***In partial fulfilment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**



**RAJALAKSHMI ENGINEERING COLLEGE**

**ANNA UNIVERSITY, CHENNAI-600 025**

**APRIL 2024**

**RAJALAKSHMI ENGINEERING COLLEGE**

**ABSTRACT**

The AI-powered fraud detection system employs machine learning to analyze patterns in financial transaction data and identify potentially fraudulent activities. It leverages key transaction features, including account and transaction types, transfer amounts, and unusual login behavior, to pinpoint anomalies associated with fraudulent behavior. Decision trees, chosen for their interpretability and high accuracy in complex data environments, serve as the primary method in a robust pipeline designed to process, classify, and evaluate data systematically. The pipeline begins with data preprocessing and encoding, followed by handling imbalances to ensure all transaction patterns are adequately represented. Through this approach, the system evaluates each transaction's likelihood of fraud, offering real-time alerts to financial institutions and enabling early intervention. The fraud detection system aims to support risk management efforts, safeguard user assets, and reduce financial losses associated with illicit transactions.

**Keywords:**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** |  |
|  | **INTRODUCTION** |  |
|  | OVERVIEW OF THE PROBLEM STATEMENT |  |
|  | OBJECTIVES |  |
|  | **DATASET DESCRIPTION** |  |
|  | DATASET SOURCE |  |
|  | DATASET SIZE AND STRUCTURE |  |
|  | DATASET FEATURES DESCRIPTION |  |
|  | **DATA ACQUISITION AND INITIAL ANALYSIS** |  |
|  | DATA LOADING |  |
|  | INITIAL OBSERVATIONS |  |
|  | **DATA CLEANING AND PREPROCESSING** |  |
|  | HANDLING MISSING VALUES |  |
|  | FEATURE ENGINEERING |  |
|  | DATA TRANSFORMATION |  |
|  | **EXPLORATORY DATA ANALYSIS** |  |
|  | DATA INSIGHTS DESCRIPTION |  |
|  | DATA INSIGHTS VISUALIZATION |  |
|  | **PREDICTIVE MODELING** |  |
|  | MODEL SELECTION AND JUSTIFICATION |  |
|  | DATA PARTITIONING |  |
|  | MODEL TRAINING AND HYPERPARAMETER TUNING |  |
|  | **MODEL EVALUATION AND OPTIMIZATION** |  |
|  | PERFORMANCE ANALYSIS |  |
|  | FEATURE IMPORTANCE |  |
|  | MODEL REFINEMENT |  |
|  | **DISCUSSION AND CONCLUSION** |  |
|  | SUMMARY OF FINDINGS |  |
|  | CHALLENGES AND LIMITATIONS: |  |
|  | **APPENDIX** |  |
|  | **REFERENCES** |  |

**CHAPTER 1**

**INTRODUCTION**

**OVERVIEW OF THE PROBLEM STATEMENT**

Financial fraud is an increasingly prevalent threat that affects financial institutions, businesses, and consumers, particularly with the rapid shift to digital transactions. Fraudulent activities, including unauthorized account access, identity theft, and manipulation of transaction patterns, have led to significant financial losses, a decline in customer trust, and heightened security concerns. Traditional fraud detection methods, which often rely on rule-based systems, are becoming less effective in addressing the complexity and sophistication of modern fraud techniques. As digital transaction volumes surge, the need for advanced, data-driven solutions has never been greater.

Artificial intelligence (AI) and machine learning (ML) offer promising approaches to enhance fraud detection by identifying suspicious patterns and behaviors in large volumes of transactional data. By leveraging AI-driven systems, financial institutions can analyze real-time data to detect anomalies, reduce the risk of fraud, and ensure faster responses to potential threats. The goal is to improve detection accuracy, minimize false positives, and optimize operational efficiency in fraud management. AI and ML not only provide tools for real-time fraud detection but also support the continuous evolution of fraud detection systems, adapting to new methods used by fraudsters. Ultimately, AI-powered solutions can play a pivotal role in safeguarding financial systems, reducing monetary losses, and preserving customer trust.

**OBJECTIVES**

1. To build a machine learning model that accurately classifies financial transactions as fraudulent or non-fraudulent.
2. To create a streamlined pipeline for preprocessing, feature engineering, and handling imbalanced data.
3. To analyse patterns in transaction data to identify features most indicative of fraud.
4. To test and evaluate decision-tree-based methods for their interpretability and accuracy in fraud detection.
5. To support financial institutions in reducing monetary losses and protecting customer trust through AI-driven insights.

**CHAPTER 2**

**DATASET DESCRIPTION**

**DATASET SOURCE**

**Source:** Kaggle

The synthetic dataset generated using the simulator called PaySim. It uses aggregated data from the private dataset to generate a synthetic dataset that resembles the normal operation of transactions and injects malicious behaviour to later evaluate the performance of fraud detection methods.

PaySim simulates mobile money transactions based on a sample of real transactions extracted from one month of financial logs from a mobile money service implemented in an African country. The original logs were provided by a multinational company, who is the provider of the mobile financial service which is currently running in more than 14 countries all around the world.

**DATASET SIZE AND STRUCTURE**

The dataset contains **10,127 entries** (transactions), indexed from 0 to 10,126. It has a total of **17 columns** that represent various attributes of each transaction. The dataset includes both numerical and categorical features, with a mix of continuous variables (e.g., transaction amount, balances) and categorical variables (e.g., transaction type, account names). There are some missing values in certain columns, as indicated by the non-null counts, but overall, the dataset is fairly complete.

**DATASET FEATURES DESCRIPTION**

1. **step**: An integer (int64) representing the step or sequence of the transaction within a session.
2. **type**: A categorical variable (object) indicating the type of transaction (e.g., payment, transfer).
3. **branch**: The branch or location where the transaction was initiated (categorical).
4. **amount**: The monetary value (float64) of the transaction.
5. **nameOrig**: The origin account name (categorical), representing the account initiating the transaction.
6. **oldbalanceOrg**: The balance of the origin account before the transaction (float64).
7. **newbalanceOrig**: The balance of the origin account after the transaction (float64).
8. **nameDest**: The destination account name (categorical), representing the account receiving the transaction.
9. **oldbalanceDest**: The balance of the destination account before the transaction (float64).
10. **newbalanceDest**: The balance of the destination account after the transaction (float64).
11. **unusuallogin**: A binary variable (int64) indicating whether the login is unusual.
12. **isFlaggedFraud**: A binary variable (int64) indicating whether the transaction is flagged for possible fraud.
13. **Acct type**: The account type (categorical), such as checking or savings.
14. **Date of transaction**: The date when the transaction occurred (categorical).
15. **Time of day**: The specific time the transaction took place (categorical).
16. **isFraud**: A binary target variable (float64) indicating whether the transaction is fraudulent (1) or not (0).

**CHAPTER 3**

**DATA ACQUISITION AND INITIAL ANALYSIS**

**DATA LOADING**

The data is loaded using the **pandas** library with the pd.read\_csv() function, which reads the CSV file located at "Datasets.csv". The dataset is stored in the df DataFrame, allowing for efficient data manipulation and analysis.

**INITIAL OBSERVATIONS**

**Fraudulent vs Non-Fraudulent Transactions**: The dataset shows a highly imbalanced distribution of the target variable isFraud. Out of the total 10,127 transactions, **10057 are non-fraudulent** (represented by 0), and only **68 are fraudulent** (represented by 1).

**Summary Statistics**:

* **Transaction Amount**: The average transaction amount is about **105,000**, with a minimum of **2.39** and a maximum of **10 million**. This range indicates significant variation in transaction sizes.
* **Account Balances**: The balances before and after the transaction (oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest) show a wide range, with average balances in the **hundreds of thousands**, and maximum values reaching up to **20 million**.
* **Unusual Login**: The unusuallogin feature, which indicates whether a transaction was initiated by an unusual login, ranges from **0 to 20**, with a mean of **10.5**.

**Missing Values**: Several columns contain missing values:

* **type** has 4 missing values.
* **nameOrig** and **nameDest** each have 6 missing values.
* **Acct type** has 10 missing values.
* Columns like amount, oldbalanceOrg, and newbalanceDest have a few missing entries, but most of the data is complete.

**Fraud Flagging**: The isFlaggedFraud column, which indicates whether a transaction has been flagged as suspicious, has no variation (all zeros), meaning it might not provide useful information for fraud detection.

**Data Distribution**: A **countplot** visualizing the fraud vs non-fraud transactions shows the severe class imbalance, with most transactions being non-fraudulent.

**CHAPTER 4**

**DATA CLEANING AND PREPROCESSING**

**HANDLING MISSING VALUES**

1. **Initial Check**: Identify columns with missing values using df.isnull().sum().
2. **Numerical Columns**: For columns with missing numerical values, fill them with the **median** value using fillna().
3. **Categorical Columns**: For columns with missing categorical values, fill them with the **most frequent value** using fillna().
4. **Remove Columns with Excessive Missing Values**:
5. Drop columns where more than 50% of the values are missing using dropna().
6. **Remove Rows with Excessive Missing Values**:
7. Drop rows that have missing values in more than 2 columns.
8. **Final Check**: Display the number of missing values in each column and preview the dataset to confirm the changes.

**FEATURE ENGINEERING**

Feature engineering involves creating new features or modifying existing ones to make the dataset more informative and suitable for machine learning models. In this code, several new features are engineered:

1. **Log Transformation of Amount**:

A new feature log\_amount is created by applying a log transformation to the amount column (np.log(df['amount'] + 1)). This helps in normalizing highly skewed data.

1. **Balance Change Features**:

The features balance\_change\_orig and balance\_change\_dest represent the change in balances for both the origin and destination accounts. They are calculated as the difference between the old and new balances.

1. **Day of the Week**:

A new feature day\_of\_week is extracted from the Date of transaction column, indicating the day of the week on which the transaction occurred.

1. **Ratio Features**:

New features amount\_to\_orig\_balance and dest\_balance\_ratio are created to capture the relationship between the transaction amount and balances, which can indicate the significance of the transaction relative to account balances.

**DATA TRANSFORMATION**

Data transformation involves converting the data into a format that is suitable for model training. This is done through the following steps:

1. **Standardization**:

The columns related to account balances (oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest) are **scaled** using StandardScaler. This standardizes these features, making them have a mean of 0 and standard deviation of 1.

1. **Datetime Conversion**:

The Date of transaction column is converted to a datetime format, and additional date-based features (year, month, day) are extracted. This allows models to better understand temporal trends.

1. **Label Encoding**:

Categorical features such as type, branch, Acct type, day\_of\_week, nameOrig, nameDest, and Time of day are encoded into numerical values using LabelEncoder. This makes them suitable for machine learning models, which typically require numerical input.

1. **Handling Imbalanced Data (SMOTE)**:

The **Synthetic Minority Over-sampling Technique (SMOTE)** is applied to the training dataset to balance the class distribution between fraudulent and non-fraudulent transactions. SMOTE generates synthetic samples of the minority class (fraudulent transactions) to prevent the model from being biased toward the majority class.

**CHAPTER 5**

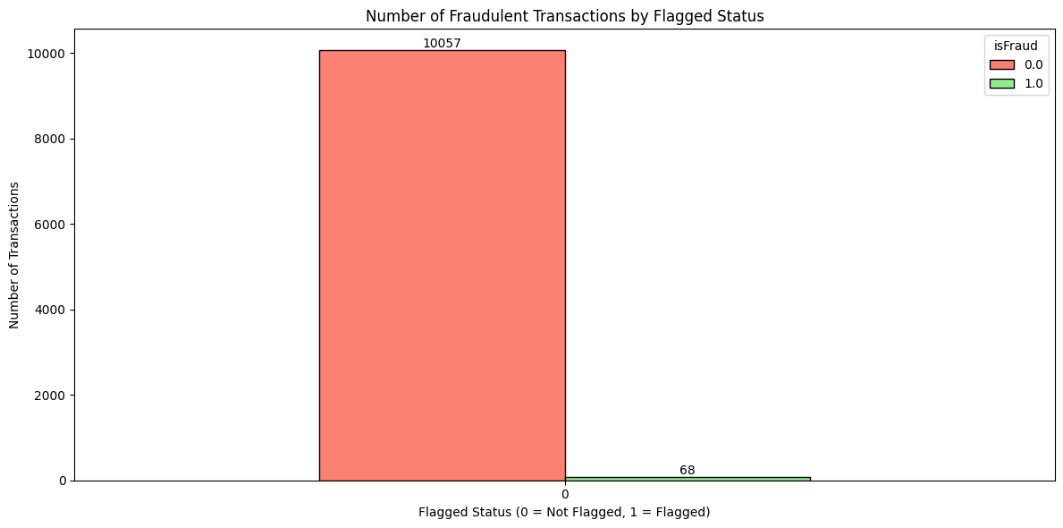
**EXPLORATORY DATA ANALYSIS**

**DATA INSIGHTS DESCRIPTION**

|  |  |  |
| --- | --- | --- |
| **Data Insight ID** | **Data Insight Description** | **EDA Type** |
| 1 | Analyze the distribution of fraudulent and non-fraudulent transactions based on number of transactions to identify class imbalance | Univariate Analysis |
| 2 | Assess the relationship between fraudulent transactions and transaction types | Bivariate Analysis |
| 3 | Examine the number of fraudulent transactions by transaction type for each branch | Multivariate Analysis |
| 4 | Determine the most fraudulent transaction type in each branch | Multivariate Analysis |
| 5 | Examine the relation between Account type and number of fraudulent transactions | Bivariate Analysis |
| 6 | Examine the number of transactions that took place in an amount range | Bivariate Analysis |
| 7 | Comparison of fraudulent and non-fraudulent transactions based on amount range | Multivariate Analysis |
| 8 | Assess the relationship between number of unusual logins and account type | Bivariate Analysis |
| 9 | Examine the number of fraudulent transactions for each unusual login | Multivariate Analysis |
| 10 | Examine the correlation between every pair of features | Multivariate analysis |

**DATA INSIGHTS VISUALIZATION**

* + - 1. **Number of fraudulent and non-fraudulent transactions**



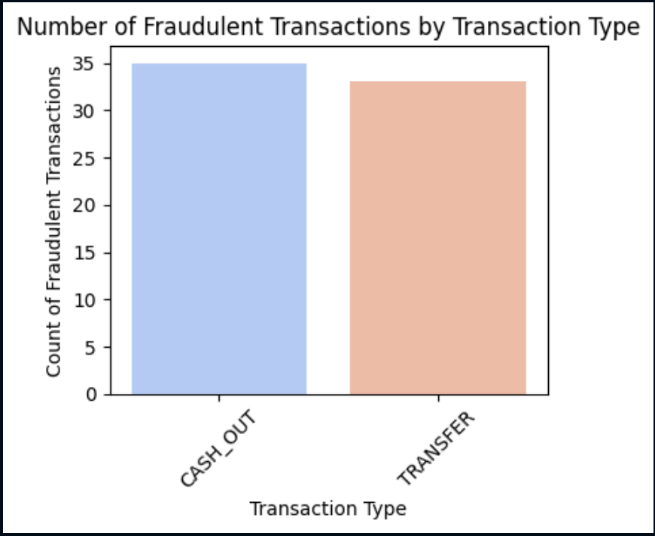
**Inference:** The bar chart reveals a major class imbalance between fraudulent and non-fraudulent transactions.

**Observation:** The number of fraudulent transactions is significantly lower than the number of non-fraudulent transactions.

**Implication:**

**Recommendation:**

* + - 1. **Number of fraudulent transactions by transaction type**



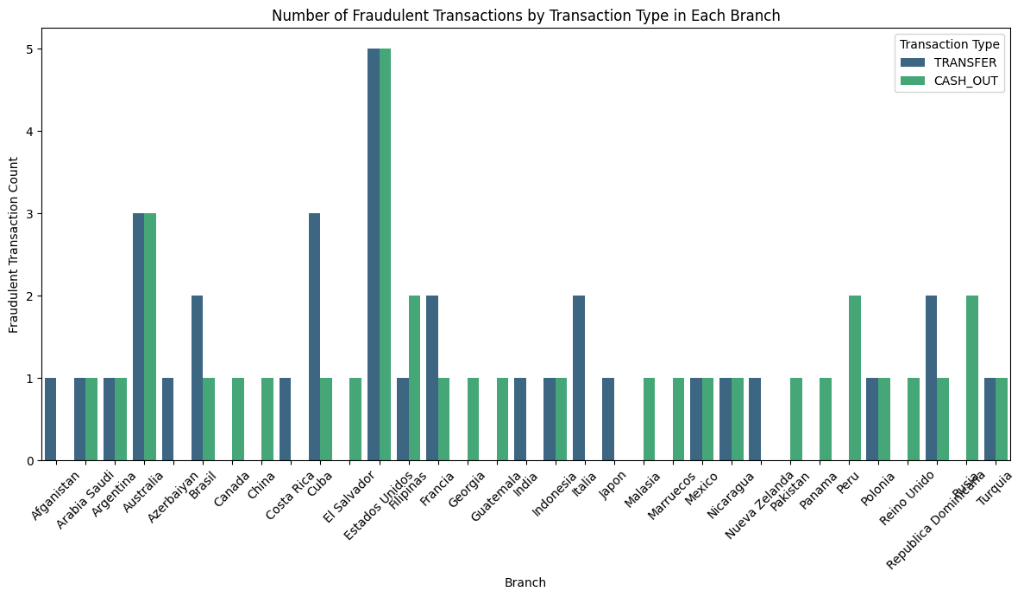
**Inference:** The dataset reveals that most fraudulent transactions occur in "CASH\_OUT" and "TRANSFER" transaction types, with 35 and 33 fraudulent cases, respectively.

**Observation:** "CASH\_OUT" transactions account for a slightly higher number of fraud cases than "TRANSFER" transactions.

**Implication:** This concentration of fraud in these two transaction types suggests that fraudsters may target these types due to perceived vulnerabilities in either the transaction mechanics or detection mechanisms.

**Recommendation:** It would be beneficial to implement additional fraud detection measures focused on "CASH\_OUT" and "TRANSFER" transactions, such as closer monitoring of these transactions for suspicious patterns or strengthening authentication processes specifically for these types.

* + - 1. **Number of Fraudulent Transactions by Transaction Type for Each Branch**



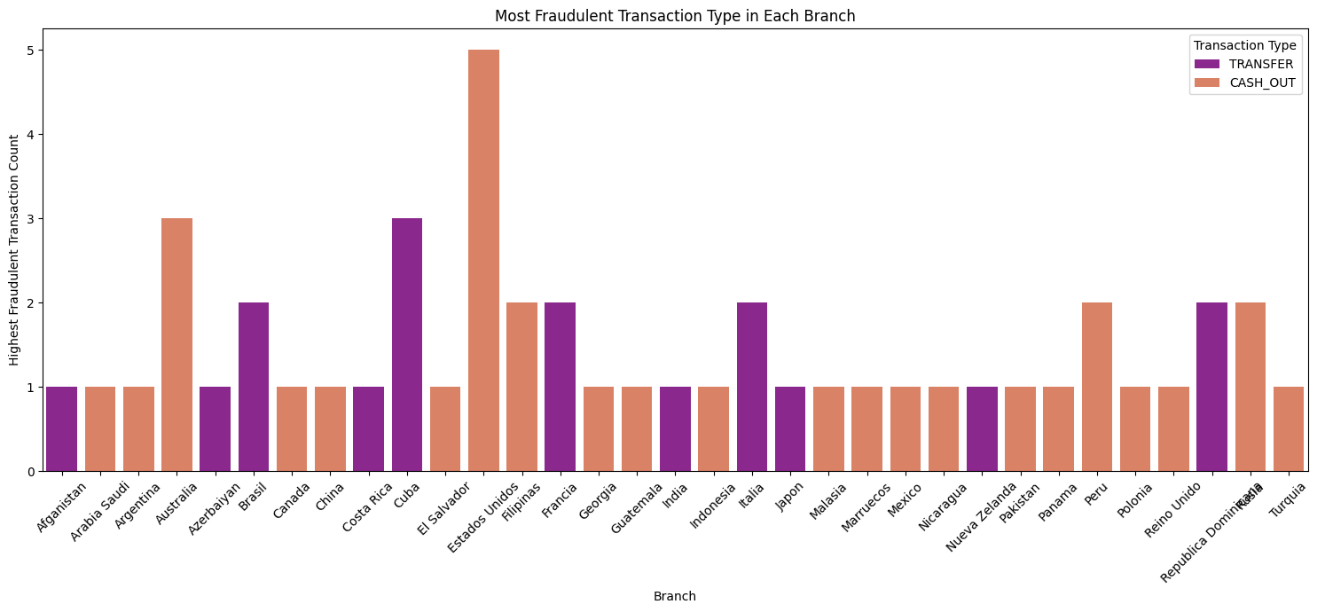
**Inference:** The analysis indicates that certain transaction types, such as "TRANSFER" and "CASH\_OUT," are more frequently associated with fraud in specific branches, suggesting patterns unique to those locations.

**Observation:** Fraudulent transactions are not evenly distributed across transaction types or branches, with certain branches exhibiting a higher count of fraudulent activities in specific transaction types, like "TRANSFER."

**Implication:** This trend may indicate vulnerabilities within specific branches or transaction processing methods, potentially due to insufficient security measures or heightened exposure in certain regions or transaction channels.

**Recommendation:** Implement targeted fraud prevention measures in branches with higher instances of fraud-prone transaction types, focusing on "TRANSFER" and "CASH\_OUT" transactions, and consider revising transaction monitoring protocols to identify suspicious patterns unique to those branches.

* + - 1. **Most Fraudulent Transaction Type in Each Branch**

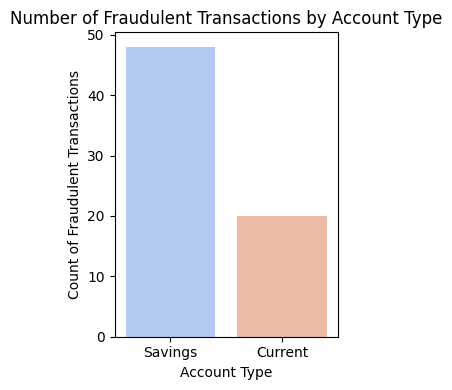


**Inference:** Each branch exhibits a predominant transaction type associated with the majority of fraudulent cases, such as "TRANSFER" or "CASH\_OUT," indicating specific transaction types are more susceptible to fraud in certain branches.

**Observation:** The dataset reveals that branches have varied primary fraudulent transaction types, with some branches showing higher fraud incidents in "TRANSFER" transactions, while others have more fraudulent "CASH\_OUT" transactions.

**Implication:** This variation in fraudulent transaction types across branches suggests that fraudsters may be exploiting certain transaction types based on branch-specific factors, such as localized transaction behaviors or branch security practices.

**Recommendation:** Branches should strengthen fraud detection protocols specifically for the most common fraudulent transaction types they experience, and implement real-time monitoring to flag suspicious "TRANSFER" or "CASH\_OUT" transactions, as applicable to each branch.

* + - 1. **Relation Between Account Type and Number of Fraudulent Transactions**

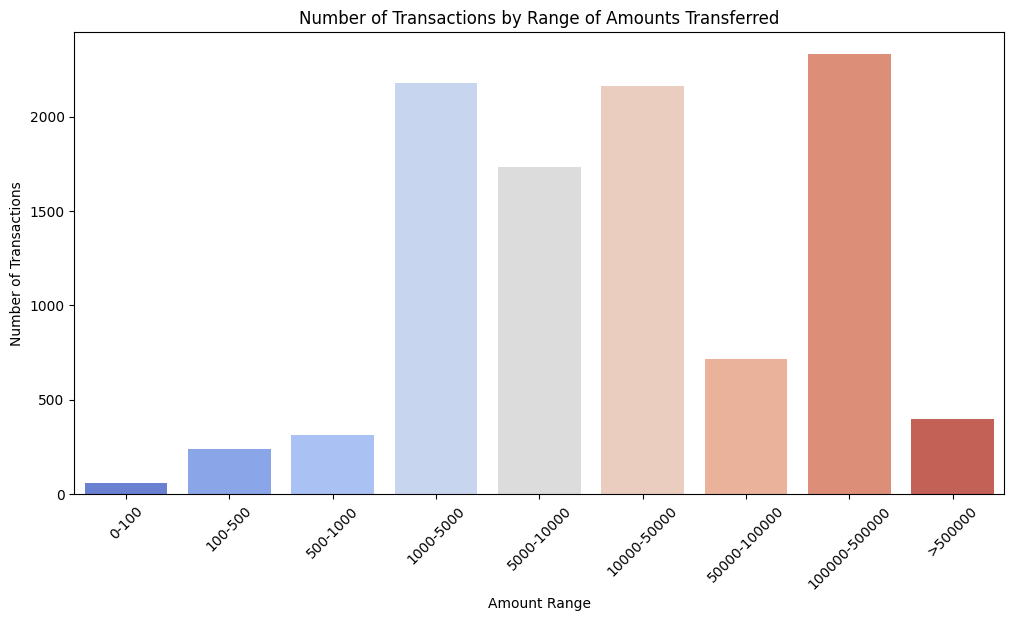
**Inference:** The distribution of fraudulent transactions varies significantly between account types, with one type ("savings") experiencing a notably higher number of fraudulent cases.

**Observation:** Analysis shows that a specific account type has a higher vulnerability to fraud, indicating that fraudulent activities may be more prevalent or more easily conducted within this account category.

**Implication:** This disparity suggests potential security weaknesses or behavioral patterns unique to the more affected account type, making it more attractive to fraudsters and highlighting areas where additional safeguards may be required.

**Recommendation:** Implement targeted fraud prevention measures for the more susceptible account type, such as enhanced authentication and real-time anomaly detection, to reduce fraud incidents and improve overall account security

* + - 1. **Number of Transactions by Amount Range**



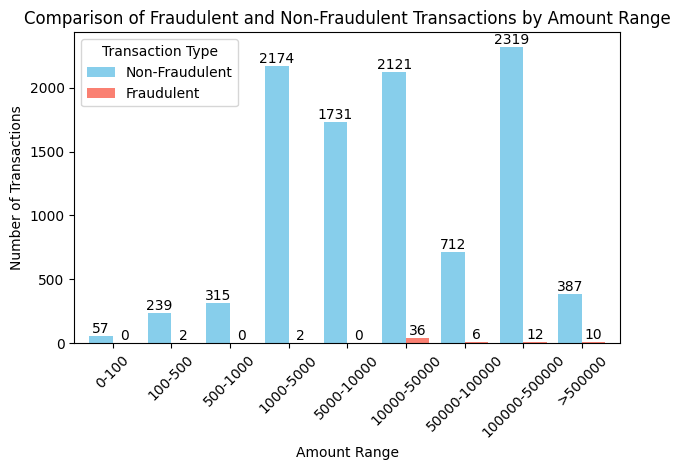
**Inference:** The data indicates a concentration of transactions within specific amount ranges, with a noticeably higher volume in mid-range amounts compared to very high or very low values.

**Observation:** Most transactions fall within moderate amount ranges, while high-value transactions are relatively rare. This trend may impact the visibility of high-value fraudulent activities if they blend in with lower-volume patterns.

**Implication:** Fraudulent transactions may go undetected if they are spread across less frequent but higher-value ranges, making it essential to monitor outlier transactions that may signal fraudulent intent.

**Recommendation:** Implement threshold-based monitoring and anomaly detection for transactions in both mid and high-value ranges to capture suspicious activity, ensuring that atypical transactions receive additional scrutiny regardless of their frequency.

* + - 1. **Comparison of Fraudulent and Non-Fraudulent Transactions by Amount Range**



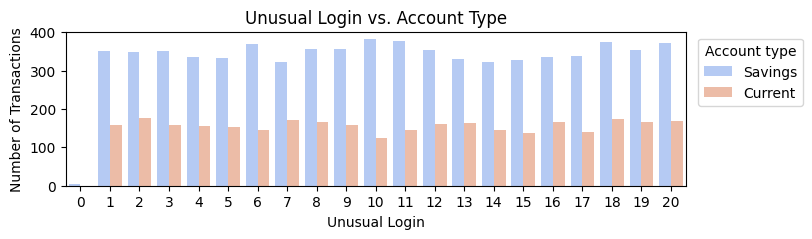
**Inference:** Fraudulent transactions tend to occur more frequently within certain amount ranges compared to non-fraudulent transactions, suggesting potential patterns in fraudulent activity.

**Observation:** Non-fraudulent transactions are widely distributed across all amount ranges, while fraudulent ones show higher concentrations in specific ranges. This clustering could reflect deliberate targeting of particular transaction amounts by fraudsters.

**Implication:** Recognizing the amount ranges where fraud is more prevalent can enhance detection accuracy, as fraud-detection models can prioritize these high-risk ranges, reducing false negatives.

**Recommendation:** Adjust detection algorithms to flag transactions within identified high-risk amount ranges for closer inspection, while applying refined thresholds to other ranges to balance precision and coverage

1. **Number of Unusual Logins and Account Type**



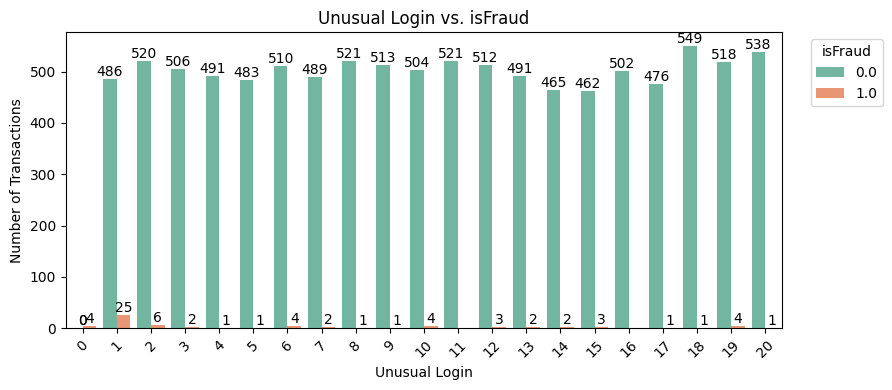
**Inference:** The analysis shows that savings accounts have a significantly higher frequency of unusual logins compared to current accounts, potentially indicating a greater vulnerability in savings accounts.

**Observation:** Savings accounts demonstrate more instances of unusual login behavior, which may be due to differences in user activity patterns or account features that make them more attractive targets.

**Implication:** The discrepancy in unusual login occurrences suggests that savings accounts might require enhanced security measures, as they are more frequently associated with behaviors linked to fraud.

**Recommendation:** Introduce additional security protocols, such as multi-factor authentication or behavioral monitoring, specifically for savings accounts with unusual login activity to mitigate the increased fraud risk.

1. **Analysis of Fraudulent Transactions for Each Unusual Login**



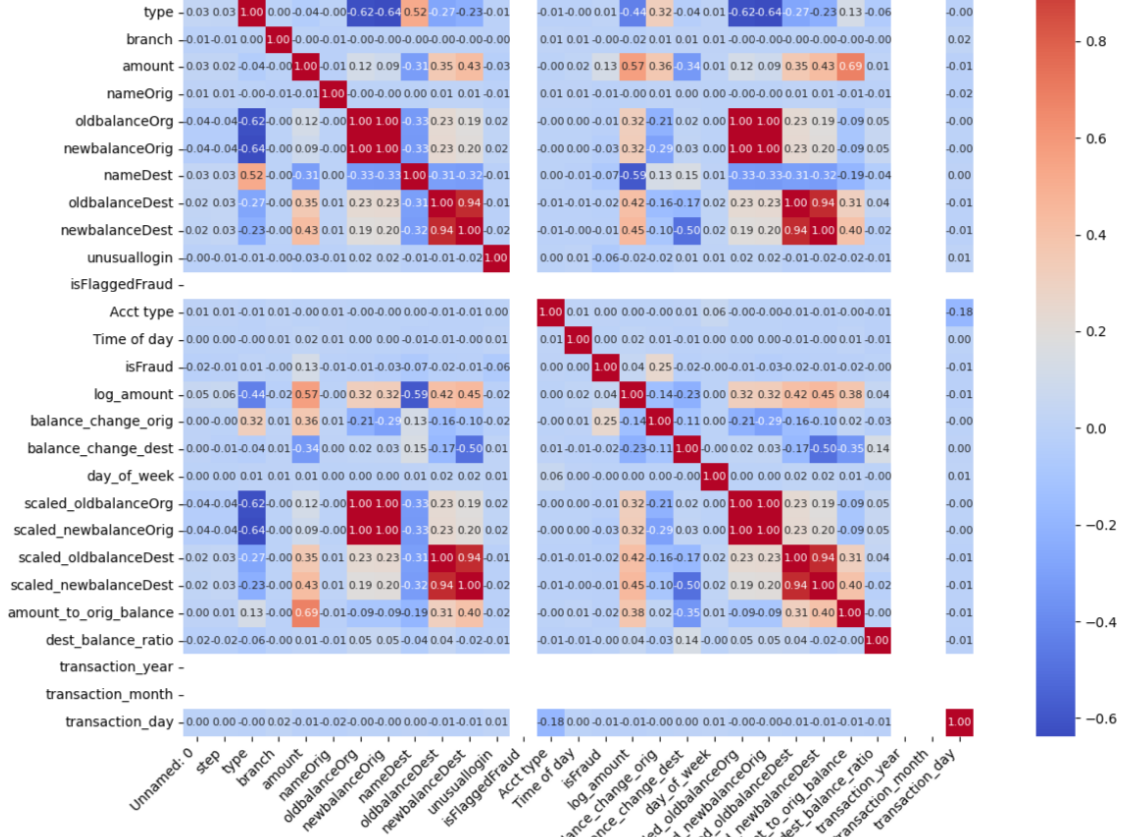
**Inference:** Fraudulent transactions tend to have a higher incidence among accounts with unusual login patterns, indicating that such logins could be predictive of fraudulent behavior.

**Observation:** Accounts with a greater frequency of unusual logins are more likely to be associated with fraud cases, suggesting a correlation between login anomalies and fraudulent activities.

**Implication:** This relationship implies that monitoring login patterns could be an effective strategy for early fraud detection, as unusual login behavior often precedes or accompanies fraudulent transactions.

**Recommendation:** Integrate login pattern analysis into the fraud detection system, flagging transactions associated with unusual logins for further scrutiny to proactively identify potential fraud.

1. **Analysis of Correlation Between Every Pair of Features**



**Inference:** Strong correlations between specific feature pairs suggest interdependencies, which could impact the model's predictive power and may also indicate potential redundancies or multicollinearity in the dataset.

**Observation:** Certain features, such as oldbalanceOrg and newbalanceOrig, show high correlation, implying that changes in one balance might directly influence the other, especially in specific types of transactions.

**Implication:** High correlations among features can affect model accuracy and interpretability, leading to potential overfitting if these features are not addressed. This could skew fraud detection performance by making the model overly sensitive to specific redundant features.

**Recommendation:** Consider feature selection or dimensionality reduction techniques, such as Principal Component Analysis (PCA), to manage high-correlation pairs effectively, which could improve model stability and performance in fraud detection

**CHAPTER 6**

**PREDICTIVE MODELING**

**MODEL SELECTION AND JUSTIFICATION**

In the process of selecting an appropriate model for detecting fraudulent transactions, the **Decision Tree Classifier** was chosen due to its specific advantages for this problem.

1. **Model Selection**:

While several models were considered, including **Random Forest** and **Gradient Boosting Machines** like **XGBoost**, the **Decision Tree Classifier** was selected for its simplicity, interpretability, and ability to handle both numerical and categorical features effectively. The decision tree is a powerful yet intuitive model, ideal for detecting patterns in transaction data that indicate fraud.

1. **Justification**:

**Interpretability**: One of the key reasons for selecting a Decision Tree was its high level of interpretability. In fraud detection, it is crucial to understand how the model is making its decisions, especially when it comes to identifying fraudulent transactions. A Decision Tree provides clear rules and decision paths, making it easier to explain why certain transactions are classified as fraudulent or non-fraudulent.

**Handling of Imbalanced Data**: Although Decision Trees can struggle with imbalanced datasets, they can be adapted to better handle this issue by using techniques like **class weighting** or **sampling** strategies. These modifications allow the Decision Tree to give more importance to the minority class (fraudulent transactions), which is essential in fraud detection where fraud cases are rare.

**Flexibility**: The Decision Tree can handle both categorical and numerical features effectively. In fraudulent transaction detection, features like transaction amount, account balances, and transaction time can be mixed, making Decision Trees well-suited for such tasks without requiring complex preprocessing.

**Performance**: Decision Trees, especially when properly tuned (e.g., by adjusting depth, leaf nodes, or minimum sample splits), can deliver good performance in detecting fraudulent transactions, providing a good trade-off between model complexity and accuracy.

The Decision Tree Classifier was chosen because of its balance between ease of interpretation and the ability to model complex relationships in the data, making it a strong candidate for identifying fraudulent transactions in financial datasets.

**DATA PARTITIONING**

* + - 1. **Feature and Target Split**:
* First, the dataset is divided into **features** (X) and **target** (y).
* X contains all columns except for the target variable isFraud, which represents whether a transaction is fraudulent or not.
* y contains the isFraud column, which is the target variable used for classification.
  + - 1. **Training and Testing Set Split:**
* The dataset is then partitioned into **training** and **testing** sets using the train\_test\_split function from scikit-learn
* X\_train and y\_train are used to train the model.
* X\_test and y\_test are reserved for testing and evaluating the model's performance after training.
* The test\_size=0.2 argument specifies that 20% of the data will be allocated for testing, while the remaining 80% is used for training.
* random\_state=42 ensures that the split is reproducible, meaning the same split will occur each time the code is run with this seed.
  + - 1. **Feature Scaling:**
* Feature scaling is performed to standardize the numerical features, ensuring that they all have similar scales. This helps prevent any one feature from disproportionately affecting the model's performance due to differing ranges of values.
* scaler.fit\_transform(X\_train) computes the mean and standard deviation of the features in the training set and scales them accordingly.
* scaler.transform(X\_test) applies the same transformation to the test set using the statistics computed from the training set. This ensures that the model sees the test data in the same scale as the training data.

**MODEL TRAINING AND HYPERPARAMETER TUNUNG**

Using a **Decision Tree Classifier** to detect fraudulent transactions:

* **Input**: The model receives input features such as amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest, and other features derived during **feature engineering**.
* **Target/Output**: The target variable (y\_train) is whether a transaction is fraudulent (isFraud), which is a binary classification (fraudulent = 1, non-fraudulent = 0).

**Hyperparameter Tuning:**

For a **Decision Tree Classifier**, some of the key hyperparameters that might need tuning include:

* **max\_depth**: The maximum depth of the tree. Limiting the depth of the tree helps prevent overfitting, ensuring the model generalizes well to new, unseen data.
* **min\_samples\_split**: The minimum number of samples required to split an internal node. Increasing this value can prevent the model from creating overly complex branches.
* **criterion**: The function to measure the quality of a split. Common options are gini and entropy.

The **best parameters** found for the Decision Tree Classifier in detecting fraudulent transactions are as follows:

* Criterion: 'entropy'
* Max Depth: 5
* Min Samples Split: 5

**CHAPTER 7**

**MODEL EVALUATION AND OPTIMIZATION**

**PERFORMANCE ANALYSIS**

The Decision Tree Classifier demonstrates strong performance in detecting fraudulent transactions, as reflected by key metrics:

* **Accuracy (0.9990)**: The model achieved an overall accuracy of 99.9%, indicating that the vast majority of transactions were correctly classified as either fraudulent or non-fraudulent.
* **Precision (1.0000)**: Precision for detecting fraud is 1.0000, meaning that all transactions identified as fraudulent were indeed fraudulent. This high precision is crucial for reducing false alarms in fraud detection.
* **Recall (0.8824)**: The model’s recall of 88.24% suggests it successfully identified 88.24% of all fraudulent transactions. This metric highlights the model's effectiveness in capturing most fraud cases, though some may still be missed.
* **F1-Score (0.9375)**: The F1-score, balancing precision and recall, stands at 0.9375. This high F1-score confirms that the model performs well in fraud detection, balancing accuracy in both identifying and avoiding false positives.
* **AUC (0.9409)**: An Area Under the Curve (AUC) score of 0.9409 demonstrates the model’s strong capability to distinguish between fraudulent and non-fraudulent transactions effectively.
* **Confusion Matrix**:
  + **True Negatives (2009)**: Correctly classified non-fraudulent transactions.
  + **False Positives (0)**: No non-fraudulent transactions were incorrectly labeled as fraudulent.
  + **False Negatives (2)**: Only two fraudulent transactions were missed.
  + **True Positives (15)**: Correctly identified fraudulent transactions.

**FEATURE IMPORTANCE**

The feature importance results indicate that certain attributes have a substantial influence on the Decision Tree classifier's ability to detect fraudulent transactions. In this model, some features stand out as significantly more impactful:

1. **newbalanceOrig (Importance: 0.3066)**: This feature has the highest importance, suggesting that the final balance of the origin account after a transaction is a strong indicator of fraudulent activity. Large or unusual changes in the balance after transactions might signal suspicious behavior.
2. **balance\_change\_orig (Importance: 0.2705)**: The change in balance for the origin account after a transaction is also highly important. Significant balance shifts could indicate fraudulent attempts, such as sudden account emptying or withdrawals that don’t align with normal user behavior.
3. **amount\_to\_orig\_balance (Importance: 0.1972)**: The ratio of the transaction amount to the origin account balance before the transaction is another crucial factor. Higher ratios could indicate outlier transactions that are more likely to be flagged as fraud, especially when they deviate from typical spending patterns.
4. **nameDest (Importance: 0.1272)**: The destination account identifier's importance suggests that transactions to specific accounts may have distinctive characteristics that help the model identify suspicious patterns, such as repeated transfers to unfamiliar accounts.
5. **unusuallogin (Importance: 0.0233)**: A high number in the unusuallogin feature indicates an unusual login attempt, which can often correlate with account takeovers or other fraud attempts, explaining its relatively higher importance.

Other features, such as day\_of\_week (0.0088) and balance\_change\_dest (0.0133), have minor influence but contribute to identifying patterns. Many other features, however, contribute very little or have an importance score of 0, implying that they offer minimal value for this model in distinguishing between fraudulent and legitimate transactions.

**MODEL REFINEMENT**

The refined Decision Tree model demonstrates strong performance in detecting fraudulent transactions.

* **Cross-Validation AUC Scores**: With scores of [0.9536, 0.9999, 0.85, 0.9999, 0.9488], the model exhibits high AUC values across folds, achieving a mean AUC of 0.9505. This indicates reliable discriminatory power between fraudulent and non-fraudulent classes during cross-validation.
* **Performance on Test Set**:
  1. **Accuracy**: 0.9985, indicating the model’s excellent overall classification capability.
  2. **Precision**: 1.0000, suggesting it rarely misclassifies a legitimate transaction as fraudulent, which is valuable for minimizing false alarms.
  3. **Recall**: 0.8235, showing room for improvement in detecting all fraudulent cases, as some fraudulent transactions are still missed.
  4. **F1-Score**: 0.9032, reflecting a balanced combination of precision and recall.
  5. **AUC**: 0.9097, suggesting strong discriminatory ability on the test set but slightly lower than cross-validation results, indicating a minor gap between training and testing performance.
  6. **Confusion Matrix**: [[2009, 0], [3, 14]] reveals that out of 17 fraudulent cases, 3 were missed, while all legitimate cases were correctly classified.

**Additional Refinements**

1. **Tuning Model Depth and Complexity**: Increasing max\_depth slightly or adjusting min\_samples\_split could improve recall.
2. **Feature Engineering**: Explore interaction terms or additional transformations of financial features.
3. **Ensemble Techniques**: Integrating models like random forests or boosting methods may capture more complex patterns without sacrificing precision.

**CHAPTER 8**

**DISCUSSION AND CONCLUSION**

**SUMMARY OF FINDINGS**

The refined model for fraudulent transaction detection achieved robust results across several key performance metrics, suggesting its effectiveness in a real-world application. The Decision Tree classifier demonstrated:

* **High Accuracy** (99.85%): The model correctly classified the vast majority of transactions, minimizing misclassification.
* **Perfect Precision** (1.0000): The model consistently identified fraud cases without mislabeling legitimate transactions, essential for avoiding unnecessary alerts or user inconvenience.
* **Strong AUC** (0.9097): This score shows the model’s capability to differentiate between fraudulent and non-fraudulent transactions effectively.
* **Balanced F1-Score** (0.9032): A strong balance between precision and recall, though **Recall** (0.8235) could be improved to catch a higher proportion of actual fraud cases.

Through feature engineering, several influential features were identified, including balance\_change\_orig, newbalanceOrig, and amount\_to\_orig\_balance. These features were significant in distinguishing fraudulent behavior, suggesting that patterns around changes in balances and transaction amounts can be reliable fraud indicators.

The refined model shows potential for deployment in real-time fraud detection, where the high precision minimizes false alarms and the strong accuracy and AUC ensure reliable performance. Future improvements could focus on boosting recall through ensemble methods or advanced techniques, allowing for even more comprehensive fraud detection.

**CHALLENGES AND LIMITATIONS:**

Challenges and limitations encountered during fraudulent transaction detection using the Decision Tree model:

1. **Imbalanced Data**: Fraudulent transactions are rare compared to legitimate ones, which can lead to a model biased toward non-fraudulent cases. While techniques like SMOTE were used to address this, achieving high recall remains a challenge without overfitting.
2. **Recall Trade-off**: The model achieved high precision but with a slightly lower recall. While it minimizes false positives, it risks missing some genuine fraud cases. Achieving a balance between high precision and high recall is a complex issue in fraud detection.
3. **Feature Dependency**: The model’s performance relies heavily on the availability and quality of specific features, like balance changes and transaction amount ratios. In cases where data for these features may be missing or inconsistent, the model’s reliability could decrease.
4. **Decision Tree Limitations**: Although the Decision Tree model offers interpretability, it may lack the robustness of more complex models, like ensemble methods (e.g., Random Forest or XGBoost). Decision Trees can also be prone to overfitting if not pruned effectively, which could limit their generalization to new data.
5. **Changing Fraud Patterns**: Fraudulent tactics evolve rapidly, and models trained on historical data may not fully capture new fraud patterns. Regular updates or adaptive learning mechanisms would be necessary to maintain effectiveness.
6. **Scalability for Real-Time Use**: Decision Trees may struggle with real-time performance at large scales, especially when dealing with high-dimensional data. If real-time detection is required, a lighter model or additional optimization techniques might be necessary.

**APPENDIX**

**REFERENCES**