Table of Contents

[Section 1 1](#_Toc188960516)

[Importance of Fraud Detection in Financial Transactions 1](#_Toc188960517)

[Why Synthetic Datasets Are Used in Fraud Detection 1](#_Toc188960518)

[Benefits of Using Synthetic Data 1](#_Toc188960519)

[Limitations of Using Synthetic Data 2](#_Toc188960520)

[Dataset Information 2](#_Toc188960521)

[Selecting a Suitable Machine Learning Model for Fraud Detection 2](#_Toc188960522)

[Rationale for Choosing Random Forest: 2](#_Toc188960523)

[Comparison with Other Models: 2](#_Toc188960524)

[Python Implementation of Random Forest 3](#_Toc188960525)

[Section 2 20](#_Toc188960526)

[Display transaction volumes, fraud rates, and key trends in synthetic dataset 20](#_Toc188960527)

[Visualizes model metrics confusion matrix and Precision-Recall curves to assess fraud detection effectiveness. 21](#_Toc188960528)

[Section 3 22](#_Toc188960529)

[1. Data Exploration 22](#_Toc188960530)

[2. Trends by Type 22](#_Toc188960531)

[3. Rate of Fraud by Step 22](#_Toc188960532)

[4. Account Balance-Based Fraud Pattern (Bottom Right): 23](#_Toc188960533)

[5. Overall Analysis 23](#_Toc188960534)

[Actionable Steps to Prevent Fraud 24](#_Toc188960535)

[1. Strengthen Transaction Monitoring 24](#_Toc188960536)

[2. Target High-Risk Transaction Types 24](#_Toc188960537)

[3. Focus on Low-Balance Account Transactions 25](#_Toc188960538)

[4. Address Fraudulent Patterns 25](#_Toc188960539)

[5. Improve Customer Communication 26](#_Toc188960540)

[6. Use Advanced AI Models 26](#_Toc188960541)

[7. Regulatory and Collaborative Efforts 27](#_Toc188960542)

[Conclusion 27](#_Toc188960543)

[References 28](#_Toc188960544)

[Appendix 29](#_Toc188960545)

[Github Link 29](#_Toc188960546)

# Section 1

## Importance of Fraud Detection in Financial Transactions

Fraud detection in financial transactions is crucial for maintaining the integrity and security of financial systems. It ensures that fraudulent activities are identified and prevented before they can cause significant financial losses. Following are the fields where the fraud detection helps

**Protect Financial Assets:** By detecting fraudulent transactions early, financial institutions can prevent unauthorized access to customer accounts and prevent financial losses.

**Build Trust:** Effective fraud detection mechanisms build customer confidence in financial systems, ensuring users feel safe and secure when making transactions.

**Comply with Regulations:** Many financial institutions are required to comply with anti-money laundering (AML) and know-your-customer (KYC) regulations, which necessitate robust fraud detection systems to avoid legal penalties.

**Improve Operational Efficiency:** Fraud detection algorithms help reduce the manual effort needed to review transactions, making the process faster and more efficient.

## Why Synthetic Datasets Are Used in Fraud Detection

Synthetic data offers a solution to privacy concerns, as it replicates real data without containing sensitive customer information, ensuring privacy and data protection. Unlike real transaction datasets, which are often restricted due to privacy and legal issues, synthetic datasets provide an accessible alternative for simulating real-world scenarios.

In fraud detection, where fraudulent transactions are rare and lead to imbalanced datasets, synthetic data can be generated to simulate more fraudulent instances, improving model training. Additionally, synthetic data allows for the creation of diverse and targeted examples of fraudulent behavior, helping to refine models and enhance their effectiveness in detecting fraud.

## Benefits of Using Synthetic Data

Synthetic datasets offer several advantages, including enhanced data privacy as they do not involve real customer information, making them safe for research and development. They are highly customizable, allowing researchers to tailor data with specific fraud patterns for targeted testing. Additionally, synthetic data is readily available in large quantities, addressing limitations in accessing real transaction data and enabling the simulation of rare fraudulent cases to overcome data imbalance.

## Limitations of Using Synthetic Data

Synthetic data has limitations. It may lack the realism of actual transactions, failing to capture the complexities of real-world fraud. This can lead to overfitting, where models perform well in controlled settings but struggle with real data. The variability in synthetic datasets may also be limited, reducing model robustness. Furthermore, poorly designed synthetic data generation processes can introduce biases, potentially impacting the accuracy of fraud detection models.

## Dataset Information

The report is based on the "Synthetic Financial Datasets for Fraud Detection" sourced from Kaggle. This comprehensive dataset comprises over 6,360,000 financial transaction records, capturing 11 key data attributes while adhering to ethical data standards and supporting the evaluation of fraud detection models.

## Selecting a Suitable Machine Learning Model for Fraud Detection

Model Selected is Random Forest

### Rationale for Choosing Random Forest:

Random Forest is well-suited for synthetic datasets due to its ability to handle diverse feature sets and identify the most relevant ones using its built-in feature importance mechanism. As an ensemble method, it combines multiple decision trees, making it robust to feature variability and capable of capturing complex interactions between features.

Synthetic datasets may contain noise or artifacts, but Random Forest is resistant to overfitting by averaging results across trees, unlike models like Logistic Regression. It also effectively handles imbalanced data, such as fewer fraudulent transactions, using techniques like class weighting and subsampling to improve sensitivity to fraud.

Balancing performance and interpretability, Random Forest allows stakeholders to understand feature importance in predicting fraud, unlike more opaque models like Neural Networks. It is computationally efficient, scalable for large synthetic datasets, and easy to implement using libraries like scikit-learn. Additionally, its insights into feature contributions can help refine synthetic data generation processes for future improvements.

### Comparison with Other Models:

**Logistic Regression:**

**Limitations:** While simple and interpretable, Logistic Regression might struggle with capturing non-linear relationships inherent in fraud detection, especially with complex synthetic data.

**Gradient Boosting**

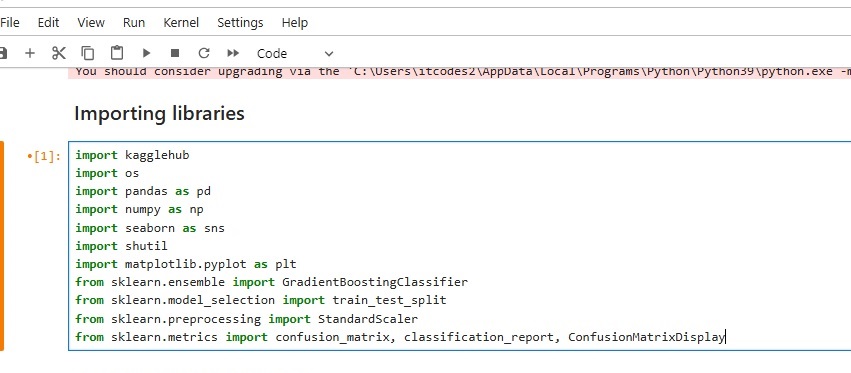
**Pros and Cons:** Gradient Boosting models like XGBoost or LightGBM offer high accuracy and are robust to overfitting. However, they require careful tuning and are computationally more intensive than Random Forest, which might be a drawback with large synthetic datasets.

**Neural Networks:**

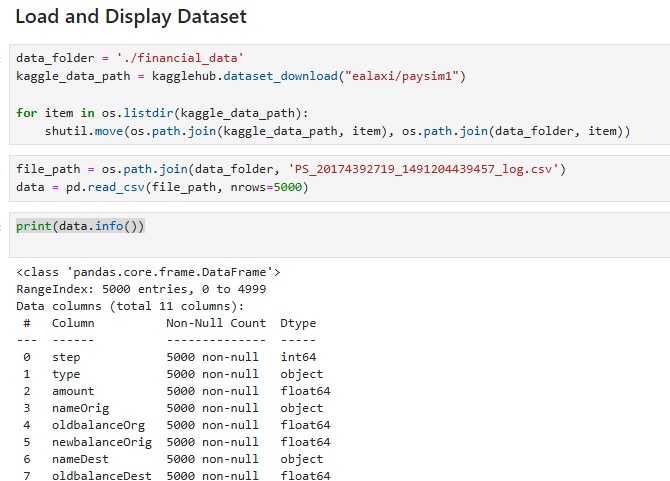
**Limitations:** While powerful, Neural Networks require large amounts of data and computational resources. They also need extensive hyperparameter tuning and are less interpretable, which could be a challenge with synthetic data that may not perfectly mimic real-world complexity.

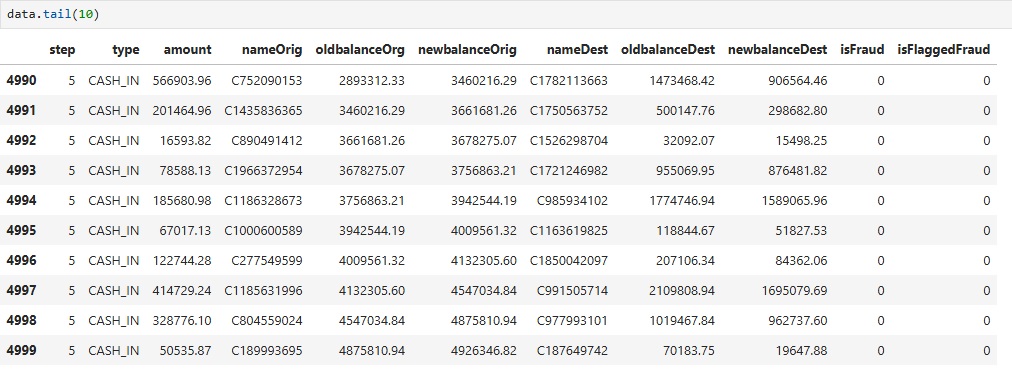
### Python Implementation of Random Forest

Notebook file link of Githun is provided in the Appendix section



Above code will Import libraries like **pandas, numpy, kagglehub and seaborn** which are needed for loading , preprocessing dataset and display visualization in python code. **Sklearn** is imported for random forest algorithm implementation. If these libraries are not installed one can install it using **pip install** command.





**Above Python code does the following**

data\_folder specifies the directory where financial data will be stored.

kaggle\_data\_path uses kagglehub.dataset\_download() to download a dataset (Paysim1).

Move is used to iterates over all files in the **kaggle\_data\_path** and moves them to the **data\_folder.**

Read CSV file constructs the full path to a specific CSV file in the **data\_folder** and reads the first 5000 rows of the file into a pandas DataFrame.

Tail(10) will display last 10 rows: of the DataFrame.

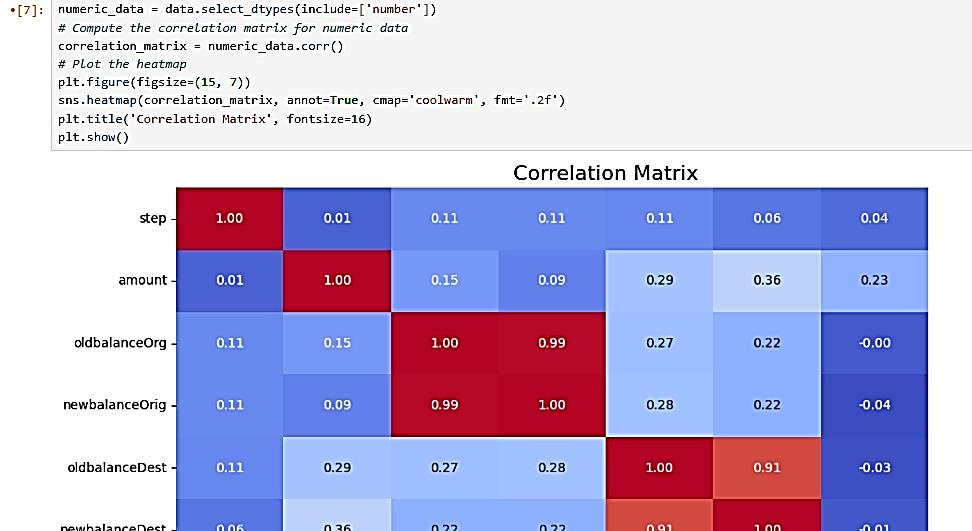


The code calculates the percentage distribution of fraud (isFraud = 1) and non-fraud (isFraud = 0) transactions in the dataset, rounds these percentages to two decimal places, and returns the result.

data['isFraud'].value\_counts() counts the occurrences of each unique value in the isFraud column and displays the result as a series.

data.isnull(): Checks the entire DataFrame for missing (null) values, returning a DataFrame of True for null entries and False otherwise.

sum(): Sums up the number of True values (null values) for each column, effectively counting the missing values column-wise.



**numeric\_data = data.select\_dtypes(include=['number'])**

filters the dataset to keep only numeric columns.

**correlation\_matrix = numeric\_data.corr()**

computes and stores a correlation matrix to analyze relationships between these numeric columns. The correlation matrix is useful for identifying patterns or dependencies between variables.

**plt.figure(figsize=(15, 7))**

**sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt='.2f')**

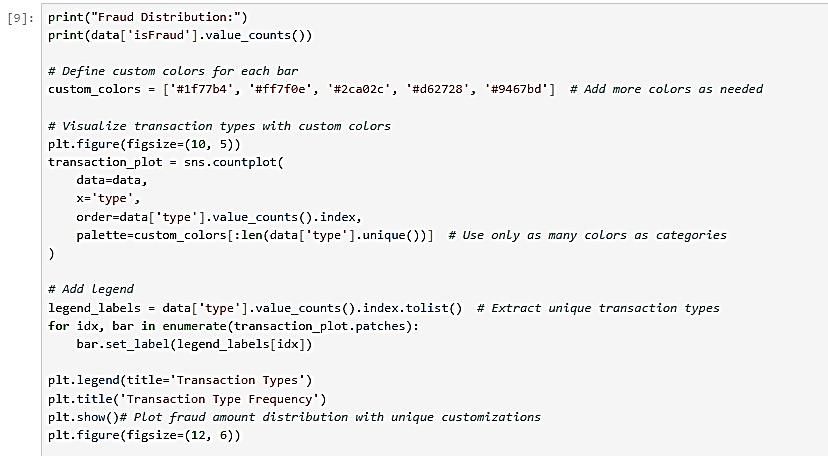
**plt.title('Correlation Matrix', fontsize=16)**

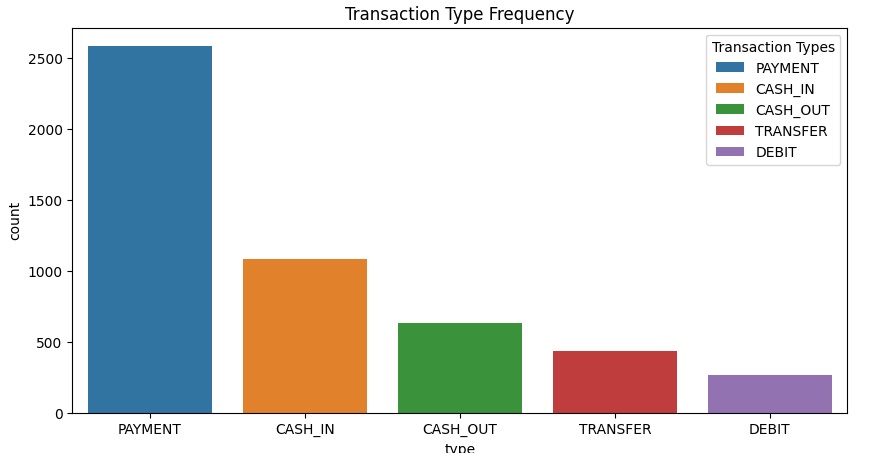
**plt.show()**

Above code will plot a visualization of co-relation metrics.

**numeric\_data.corr()['isFraud'].sort\_values(ascending=False)[1:]**

Will display Correlation within the taget values.





**custom\_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']**

Define Custom Colors The colors will be applied to each unique category in the type column of the dataset.

**plt.figure(figsize=(10, 5))** creates a new figure for the plot with specified dimensions (10x5 inches).

**transaction\_plot = sns.countplot(...)**

Plots a count of each unique transaction type using Seaborn’s countplot.

**data=data: Uses the data DataFrame for plotting.**

**x='type':** Sets the type column as the x-axis (categorical variable).

**order=data['type'].value\_counts().index: Orders the bars based on the frequency of each type (most to least common).**

palette=custom\_colors[:len(data['type'].unique())]: Assigns colors to bars, ensuring the number of colors matches the number of unique transaction types.

**legend\_labels = data['type'].value\_counts().index.tolist()**

Extracts the names of transaction types (e.g., "TRANSFER," "PAYMENT") in order of their frequency to use as legend labels.

**for idx, bar in enumerate(transaction\_plot.patches):**

Loops through all bars in the plot and assigns corresponding labels from legend\_labels.

**bar.set\_label(legend\_labels[idx])**

Sets a unique label for each bar based on the transaction type.

**plt.legend(title='Transaction Types')**

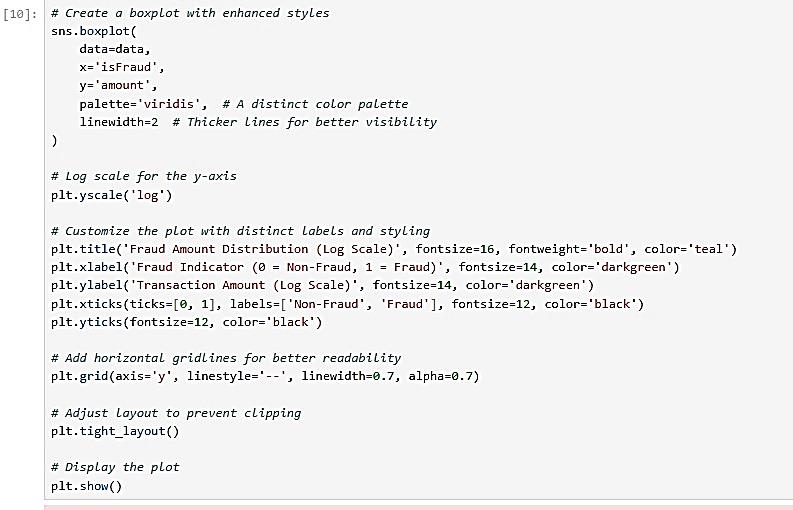
Adds a legend to the plot, titled "Transaction Types."

**plt.title('Transaction Type Frequency')**

Sets the title for the plot which is Transaction type frequency

**plt.show()**

Displays the plot.on the screen to display graph



This code generates a boxplot using Seaborn to visualize the distribution of transaction amounts (y='amount') for fraud (x='isFraud') and non-fraud transactions:

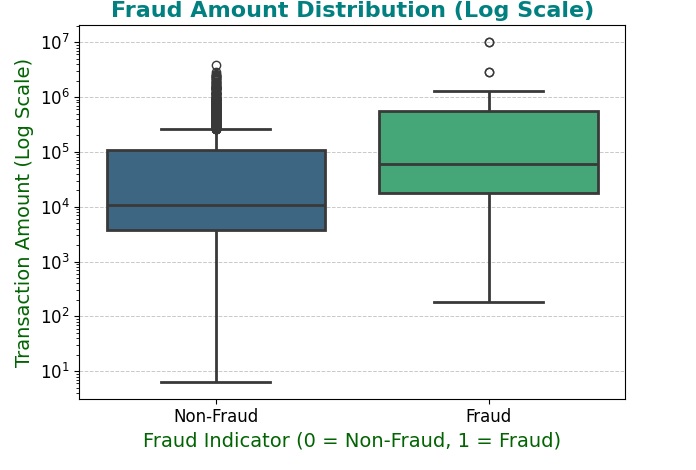
Boxplot shows the spread of transaction amounts, with isFraud=0 (non-fraud) and isFraud=1 (fraud) categories on the x-axis.

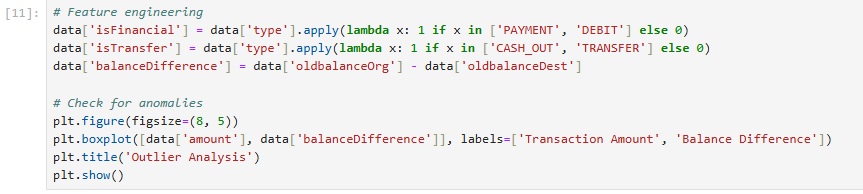
Logarithmic Scale: The y-axis is set to a log scale to handle large transaction amounts effectively.

Styling: Titles, axis labels, ticks, gridlines, and layout are customized for clarity and readability.

Color Palette: Uses the **“viridis”:** color palette for distinct visuals.

The plot is displayed, helping identify differences in transaction amounts between fraud and non-fraud cases.





**Feature Engineering**

isFinancial: Creates a new column indicating if the transaction type is "PAYMENT" or "DEBIT" (1 if true, otherwise 0).

isTransfer: Creates another column indicating if the transaction type is "CASH\_OUT" or "TRANSFER" (1 if true, otherwise 0).

balanceDifference: Adds a new column that computes the difference between oldbalanceOrg (original sender's balance) and oldbalanceDest (original receiver's balance).

**Outlier Detection (Box Plot)**

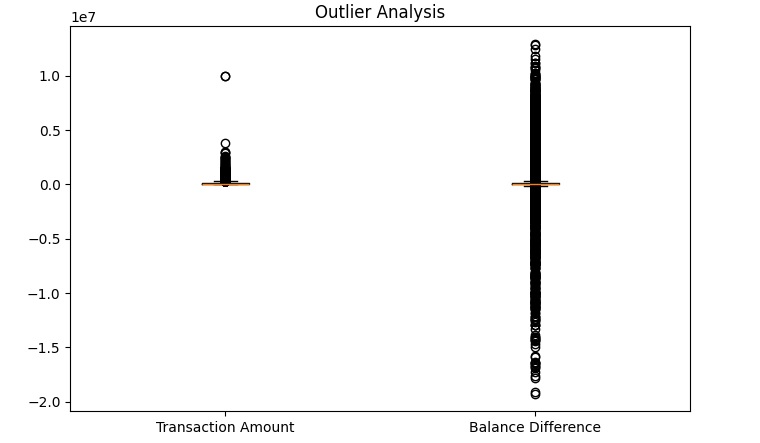
Visualizes the distribution of amount and balanceDifference using a boxplot to identify anomalies (outliers) in transaction amounts and balance differences.

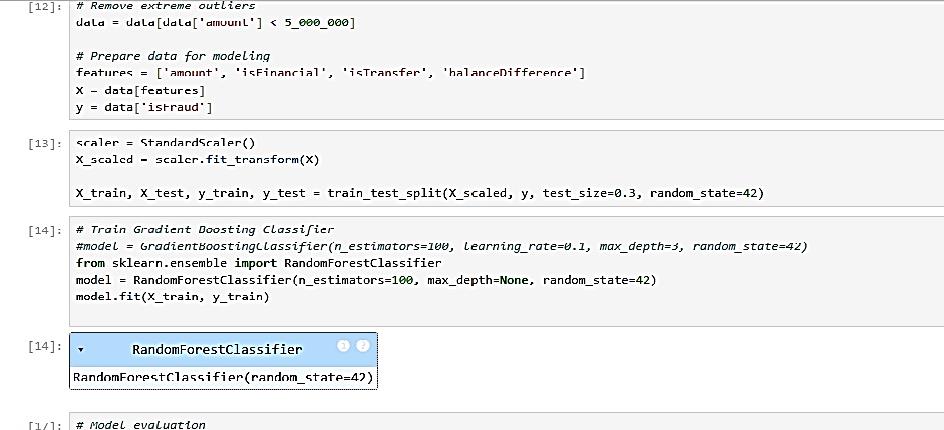
**Plot Customization:**

The boxplot is labeled with "Transaction Amount" and "Balance Difference" for clear interpretation and titled "Outlier Analysis."

**Visualization :**

The plot helps detect outliers in these features for further analysis.





The code processes the data, removes outliers, scales features, splits data for training/testing, and trains a machine learning model to detect fraudulent transactions.

**Remove Extreme Outliers**

Filters the dataset to exclude transactions with an amount greater than 5,000,000, reducing the impact of extreme outliers.

**Prepare Features and Labels**

features: Selects columns (amount, isFinancial, isTransfer, balanceDifference) as input features.

X and y: X contains the features, and y is the target column (isFraud), indicating whether a transaction is fraudulent.

**Feature Scaling**

StandardScaler: Normalizes the features in X for better performance in machine learning models.

X\_scaled: Transformed, scaled version of X.

**Train-Test Split**

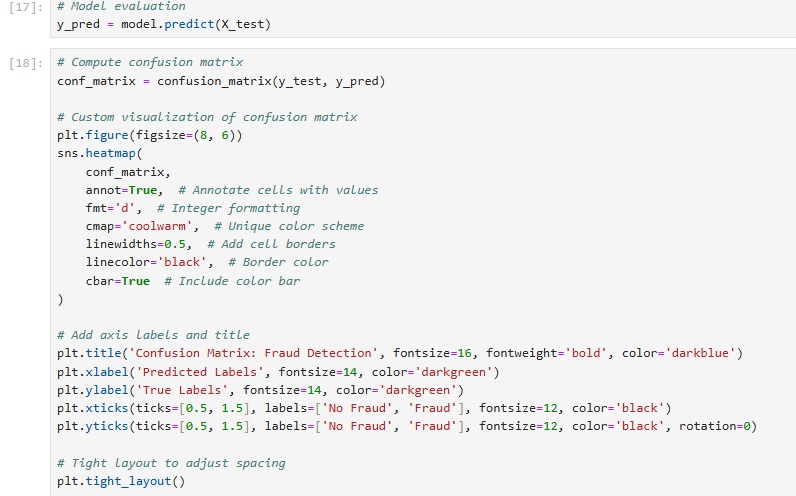
Splits the dataset into training (70%) and testing (30%) sets for model evaluation, using a random seed for reproducibility.

Train a Model:

**Random Forest Classifier**

Initializes and trains a Random Forest model with 100 trees (n\_estimators=100) and no depth limit (max\_depth=None) for classification.

The model learns patterns to classify transactions as fraud or non-fraud using the training data (X\_train and y\_train).



This code evaluates the model's performance by visualizing the confusion matrix, helping to understand how well the model distinguishes between fraudulent and non-fraudulent transactions.

**Predict Outcomes**

y\_pred = model.predict(X\_test): Uses the trained model to predict labels (fraud or no fraud) for the test dataset (X\_test).

**Confusion Matrix Calculation**

conf\_matrix = confusion\_matrix(y\_test, y\_pred): Compares the true labels (y\_test) with the predicted labels (y\_pred) to create a confusion matrix, showing counts of true positives, true negatives, false positives, and false negatives.

**Visualization of the Confusion Matrix**

A heatmap is plotted using Seaborn to display the confusion matrix.

**Styling**

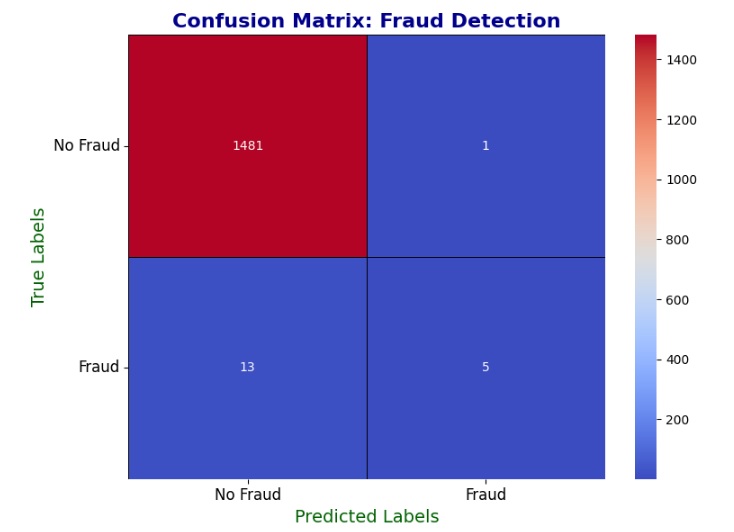
Cells are annotated with integer values (fmt='d') for clarity. Custom color scheme (cmap='coolwarm') is applied. Cell borders (linewidths=0.5) are included for better separation.

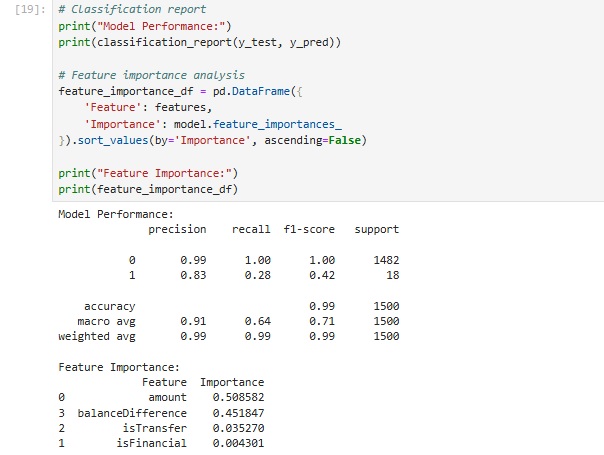
**Plot Customization**

Titles and labels are added to clearly explain what the matrix represents. Axes are labeled as "Predicted Labels" and "True Labels," with ticks indicating "No Fraud" and "Fraud."

**Layout Adjustment**

plt.tight\_layout() ensures the plot elements are properly spaced without clipping.





The classification report evaluates how model performs.The feature importance analysis identifies the most impactful features, providing insights for feature selection and model refinement.

**Classification Report**

classification\_report(y\_test, y\_pred):

Generates a detailed report on the model's performance, including metrics like precision, recall, F1-score, and support for each class (fraud and non-fraud).

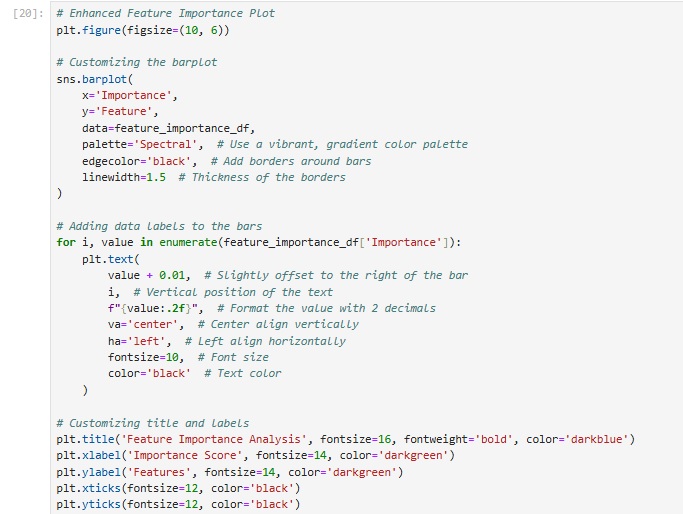
Helps assess the quality of predictions.

**Feature Importance Analysis**

**model.feature\_importances\_df:** Extracts the importance of each feature used in the Random Forest model. Higher values indicate features with a greater impact on predictions. DataFrame Creation creates a table with two columns:Feature: Names of input features **Importance** is used for **c**orresponding importance scores. **Sorting** features is used by importance in descending order for better analysis.

**Print Results**

Displays the model's classification performance and the importance of each feature, help in interpreting features most influential in predicting fraud.

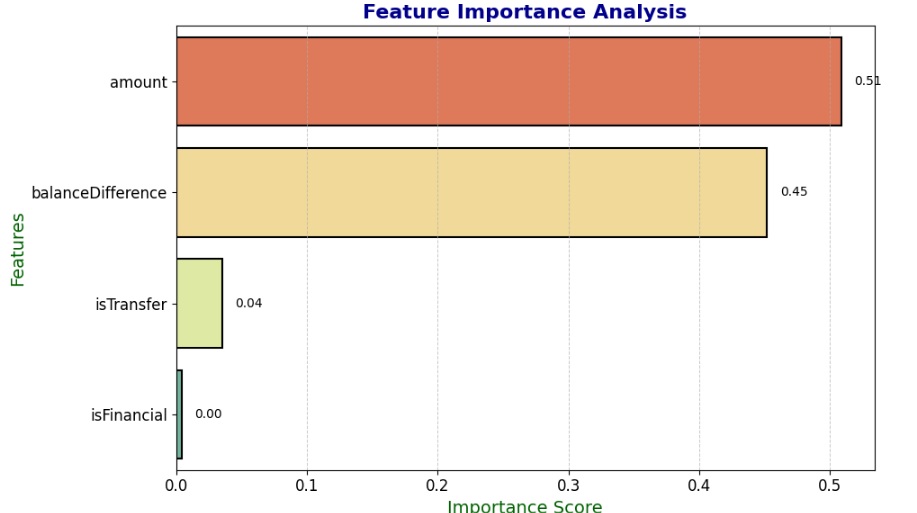


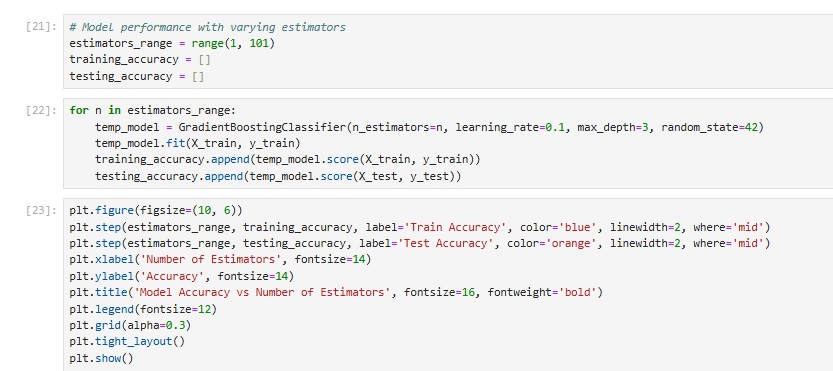
This code visualizes the importance of features using a horizontal bar plot. It provides a clear, visual representation of feature importance, helping to quickly identify the most influential features in the model.

**Bar Plot**

**sns.barplot** creates a horizontal bar chart showing feature importance scores from feature\_importance\_df.

**Data Labels** Iterates through each feature's importance score and adds a numeric label next to the corresponding bar, formatted to two decimal places for readability.





This code evaluates how the number of estimators in a GradientBoostingClassifier impacts both training and testing accuracy.

**Define Range and Lists :**

estimators\_range: Defines a range of 1 to 100 estimators for the model.

training\_accuracy and testing\_accuracy: Empty lists to store accuracy values.

**Loop Over Estimators**

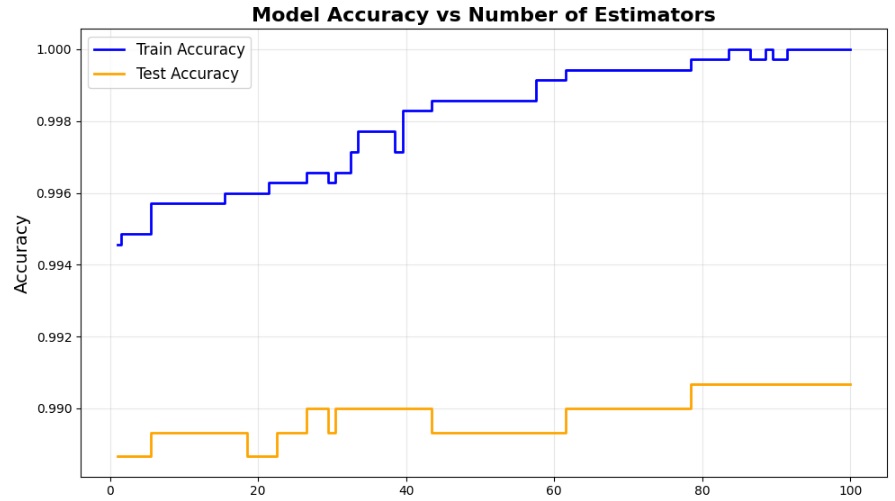
For each value of n, a new Randomforest classifier is trained with n\_estimators=n, learning\_rate=0.1, and max\_depth=3.

The model is fitted using X\_train and y\_train, and accuracy scores are calculated for both training and testing datasets.

**Plot Accuracy**

**Step Plot:** A line plot (plt.step) is created to show the relationship between the number of estimators and model accuracy for both training and testing sets.

Axis labels, title, and legend are added for clarity.





**Confusion matrix:** Helps evaluate the model's performance by summarizing correct and incorrect predictions.

**Precision-recall curve:** Assesses the trade-off between precision and recall at various thresholds, saving it for further analysis or visualization.

**Confusion Matrix to DataFrame**

conf\_matrix\_df: Converts the confusion matrix into a DataFrame with labeled rows and column and save to CSV files confusion\_matrix.csv

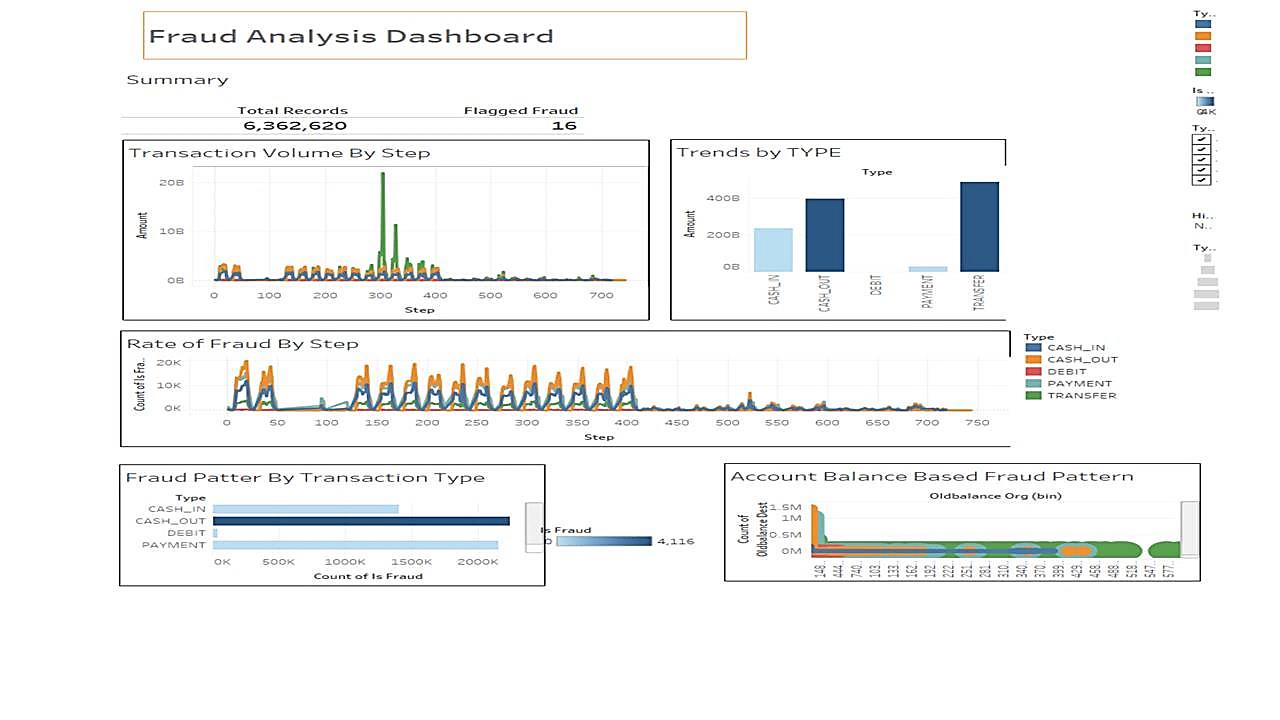
**precision\_recall\_curve**

Computes the precision, recall, and thresholds using the predicted probabilities (model.predict\_proba(X\_test)[:, 1]) for the positive class. DataFrame (pr\_curve\_df) is created with precision, recall, and thresholds, excluding the last threshold value

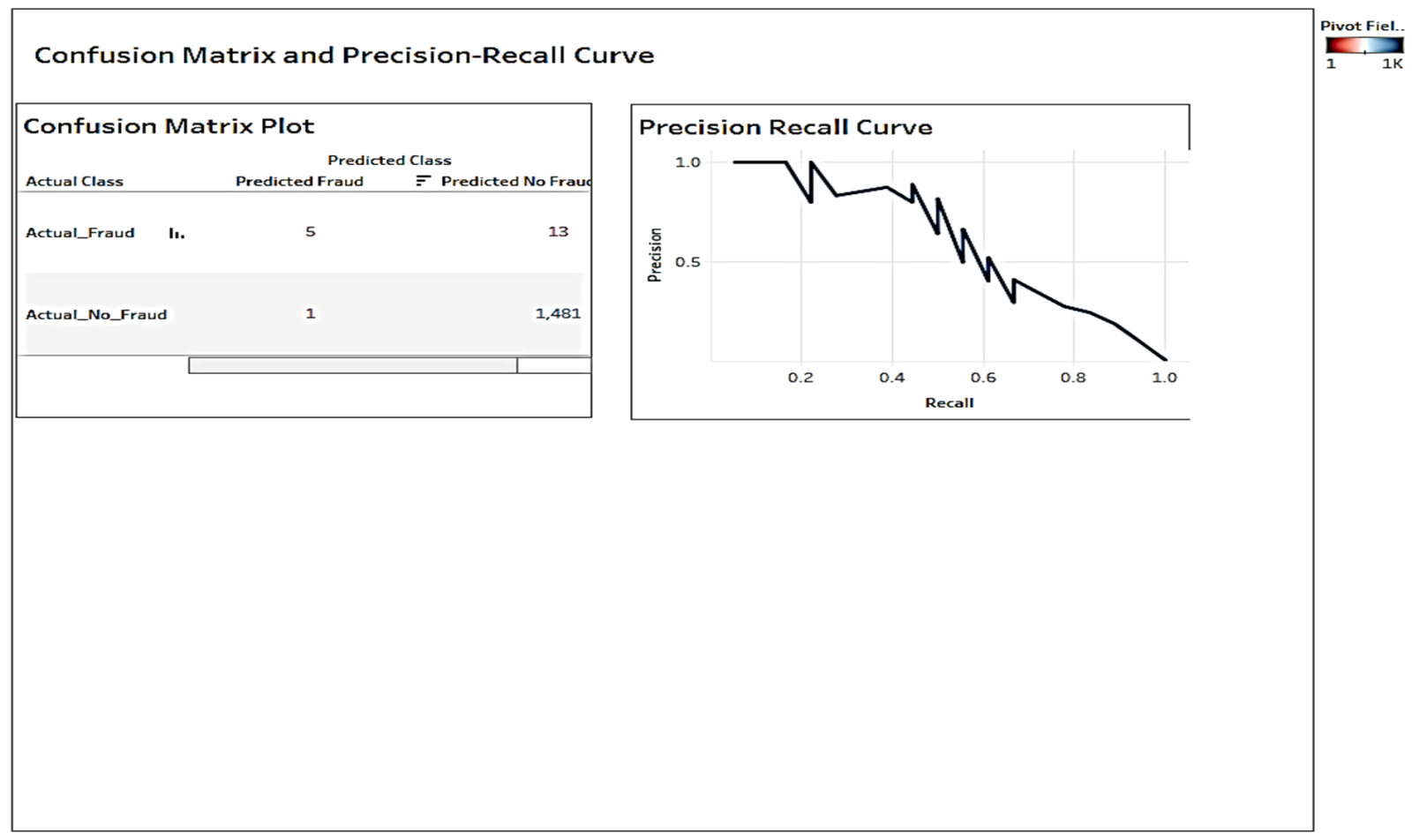
The precision-recall curve data is saved to a CSV file (precision\_recall\_curve.csv).

# Section 2

## Display transaction volumes, fraud rates, and key trends in synthetic dataset



## Visualizes model metrics confusion matrix and Precision-Recall curves to assess fraud detection effectiveness.



A confusion matrix evaluates the performance of fraud detection models by showing true positives (fraud correctly identified), true negatives which is non-fraud correctly identified, false positives which is non-fraud misclassified as fraud, and false negatives. It enables calculation of key metrics like precision , recall , an ability to detect fraud, and F1-score which is balance between precision and recall. This is especially useful in imbalanced datasets common in fraud detection. The matrix helps assess trade-offs, identify model weaknesses, and guide improvements, such as reducing missed frauds (false negatives) or unnecessary alerts , false positives, ensuring effective and balanced fraud detection.

# Section 3

## 1. Data Exploration

Transaction Volume by Step

**Observation:**

* The peak in transaction volume occurs around Step 300.
* This spike suggests abnormally high activity in a short time, warranting investigation.

**Analysis:**

* The volume reaches close to 20 billion at the peak.
* Consistently low transaction volume is seen outside the peak, indicating routine transactions.

## 2. Trends by Type

**Observation:**

TRANSFER transactions dominate the dataset with the highest total amount, exceeding 400 billion.

Other types like CASH\_OUT also show significant amounts (~200 billion), whereas DEBIT and PAYMENT have minimal contribution.

**Analysis:**

Fraud detection efforts should prioritize TRANSFER and CASH\_OUT transactions, as they handle large sums and are potential targets for fraud.

## 3. Rate of Fraud by Step

**Observation:**

* The number of fraud cases rises and falls cyclically, peaking consistently around certain steps e.g. Steps 100, 300, 500
* Each peak corresponds to about 20,000 fraudulent transactions, while the troughs have near-zero fraud cases.

**Analysis:**

This pattern indicates periodic fraud attempts, potentially automated or time-specific.

**Fraud Pattern by Transaction Type (Bottom Left):**

**Observation:**

* TRANSFER and CASH\_OUT transactions account for the majority of fraud cases, with TRANSFER surpassing 2 million cases and CASH\_OUT around 1.5 million cases.
* Fraudulent DEBIT and PAYMENT transactions are negligible.

**Analysis:**

Fraud prevention should focus on TRANSFER and CASH\_OUT, as these are the most exploited transaction types.

**3. Insights**

## 4. Account Balance-Based Fraud Pattern (Bottom Right):

**Observation:**

* Transactions with low OldBalance Org and OldBalance Dest values are the most frequent, indicating fraud targeting accounts with minimal balances.
* Scattered fraudulent cases are visible at higher balance ranges (e.g., 60M+), but these are fewer.

**Analysis:**

Fraud detection should address low-balance accounts specifically, as they seem to attract most fraud attempts.

**Summary Metrics (Top Row):**

* Total Records: 6,362,620 transactions.
* Flagged Fraud: 16 flagged cases, a very small fraction of the total.

Observation: The dataset likely includes manually flagged cases alongside machine-detected fraud.

## 5. Overall Analysis

**Transaction Trends:**

* The high volume of TRANSFER transactions (~400B) correlates with high fraud counts, confirming this type is a critical fraud target.
* Peaks in fraud activity are time-sensitive (Steps 100, 300, 500), suggesting fraud attempts might be automated or periodic.

**Fraud Targeting:**

* Fraud attempts often focus on low-balance accounts, which may evade stricter monitoring applied to high-balance accounts.

**Insights for Action:**

* Allocate fraud detection resources to TRANSFER and CASH\_OUT transaction types.
* Investigate peaks at specific steps to identify patterns in fraudulent activity timing.

## Actionable Steps to Prevent Fraud

Based on the analysis of the dashboard and trends, here are specific measures financial institutions can implement to minimize fraudulent activity:

### 1. Strengthen Transaction Monitoring

**Insight:**

Fraud is concentrated in specific transaction types (TRANSFER and CASH\_OUT) and peaks periodically at certain time steps.

**Actions**

* Use machine learning algorithms to monitor TRANSFER and CASH\_OUT transactions in real-time.
* Focus on abnormal patterns like sudden spikes in transaction volumes or amounts exceeding historical trends.
* Flag transactions occurring during unusual time windows (e.g., steps with consistent peaks like 100, 300, 500).

**Dynamic Threshold Rules**

Set thresholds for high-risk transactions e.g., transfer amounts exceeding historical averages for individual accounts Dynamically adjust these thresholds based on user behavior and transaction history.

**Real-World Application**

Banks like HSBC and JPMorgan Chase use real-time fraud detection platforms powered by AI to analyze millions of transactions and identify anomalies instantly. (Sharma, 2024)

### 2. Target High-Risk Transaction Types

**Insight**

It is clear from the visualization that the most fraudulent activities occur in TRANSFER and CASH\_OUT transactions.

**Actions**

* Mandate **Two-Factor Authentication (**2FA) for all high-value or high-risk transactions, especially in TRANSFER and CASH\_OUT categories.
* Utilize biometric authentication for added security like fingerprint or facial recognition
* Implement short delays (e.g., 15–30 minutes) for large TRANSFER transactions, allowing fraud detection systems or users to flag unauthorized activity.

**Real-World Application:**

Companies like PayPal and Zelle delay suspicious transactions and notify users immediately via email or app alerts to verify the activity. (Vieras, 2025)

### 3. Focus on Low-Balance Account Transactions

**Insight**

Fraud disproportionately targets accounts with low balances (e.g., <$1,000), likely to avoid detection.

**Actions**

* Flag transactions that result in a near-zero balance, particularly for CASH\_OUT or TRANSFER.
* Use predictive models to identify accounts likely to be targeted based on past fraud trends.
* Run awareness campaigns for low-balance account holders, educating them about common fraud schemes and phishing attempts.

**Real-World Application**

Platforms like Revolut and Venmo warn customers about suspicious transactions and enforce account limits to mitigate low-balance fraud risks.

### 4. Address Fraudulent Patterns

**Insight**

Fraudulent activity exhibits cyclical patterns (e.g., time-sensitive peaks around Steps 100, 300, 500).

**Actions**

* Use time-series models to predict when fraud peaks might occur and tighten monitoring during those windows.
* Block accounts involved in repeated suspicious activities linked to these time patterns.
* Leverage network analysis to uncover connected accounts or repeat offenders.
* Collaborate with law enforcement to dismantle organized fraud networks.

**Real-World Application**

Institutions like Mastercard use network analysis to track fraud patterns across multiple accounts and block related activity at its source. (Mastercard, 2024)

### 5. Improve Customer Communication

**Insight**

Fraud rates may escalate due to a lack of proactive communication with users.

**Actions**

* Notify customers immediately about unusual transactions or login attempts.
* Provide easy options (e.g., one-click verification) for customers to flag or confirm transactions.
* Allow users to set spending limits, freeze accounts, or whitelist trusted payees.

**Real-World Application**

Banks like Citibank provide mobile app tools for customers to monitor and control their transactions in real-time. (CNA, 2024)

### 6. Use Advanced AI Models

**Insight**

Manual review of flagged transactions is inefficient for large datasets.

**Actions**

* Train models on historical fraud data to predict the likelihood of fraud before transactions are processed.
* Use deep learning models to identify subtle patterns overlooked by traditional systems.
* Monitor user behavior (e.g., login frequency, IP address changes) to detect unusual activity.

**Real-World Application:**

Financial institutions like Barclays deploy AI-powered tools that reduce false positives while improving fraud detection rates. (Columbus, 2020)

### 7. Regulatory and Collaborative Efforts

**Insight**

Fraud is often cross-institutional, requiring industry collaboration.

**Actions**

* Share anonymized fraud data across institutions to identify emerging threats and patterns.
* Ensure adherence to regulations like PSD2 (Payment Services Directive) and implement mandatory fraud detection mechanisms.

**Real-World Application**

Organizations like SWIFT enforce compliance and fraud monitoring across global financial institutions. (Swift, 2024)

# Conclusion

Fraud detection in financial transactions is critical for safeguarding assets, maintaining trust, and ensuring compliance with regulations. Through Python and Tableau analysis, we identified significant patterns and key features such as transaction type, amount, and account differences that influence fraudulent activities. Python enabled efficient data preprocessing, feature engineering, and model evaluation, while Tableau provided interactive visualizations like confusion matrices and precision-recall curves, making insights accessible to stakeholders. The analysis highlights the importance of robust fraud detection systems that balance high accuracy with low false positives to prevent financial losses, protect customer trust, and enhance overall system security.

# References

CNA, 2024. *itibank's new anti-scam measure restricts users' access to app if risky permission settings detected.* [Online]   
Available at: https://www.channelnewsasia.com/singapore/citibank-mobile-app-anti-scam-measure-restrict-access-if-risky-permission-settings-detected-3777016  
[Accessed 25 01 2024].

Columbus, L., 2020. *How Barclays Is Preventing Fraud With AI.* [Online]   
Available at: https://softwarestrategiesblog.com/2020/06/27/how-barclays-is-preventing-fraud-with-ai/  
[Accessed 26 1 2024].

Mastercard, 2024. *Mastercard accelerates card fraud detection with generative AI technology.* [Online]   
Available at: https://newsroom.mastercard.com/news/press/2024/may/mastercard-accelerates-card-fraud-detection-with-generative-ai-technology/  
[Accessed 25 01 2024].

Sharma, P., 2024. *How Did HSBC Implement and Integrate AI for Fraud Detection?.* [Online]   
Available at: https://www.orgevo.in/post/how-did-hsbc-implement-and-integrate-ai-for-fraud-detection  
[Accessed 24 01 2024].

Swift, 2024. *20204.* [Online]   
Available at: https://www.swift.com/news-events/news/swift-launch-ai-powered-fraud-defence-cross-border-payments  
[Accessed 25 01 2024].

Vieras, B. &. M. D. &. J. A. &. M. T., 2025. The Impact of Real-Time Financial Fraud Detection on Financial Institutions' Reputation and Customer Trust..

# Appendix

## Github Link

<https://github.com/star07D/Detecting-Fraud-Using-Machine-Learning-and-Data-Visualization>