FRAUD DETECTION IN FINANCIAL TRANSACTIONS

Using Machine Learning

Submitted by,
NIMISHA DAVIS
DATA ANALYST

Submitted on,
15-02-2025

INTRODUCTION

In today's digital economy, transactions are increasing at an exponential rate. However, with this growth comes a significant rise in fraudulent activities, leading to financial losses and security threats. Fraud detection systems aim to identify suspicious transactions in real-time to minimize risks. This project focuses on detecting fraudulent transactions using machine learning techniques by analyzing patterns and anomalies in financial transaction data.

OBJECTIVE

The objective of the project "Fraud Detection in Financial Transactions Using Machine Learning" is to develop an accurate and efficient fraud detection system using machine learning techniques.

DATASET DESCRIPTION

The dataset consists of 11 variables:

- step: Represents a unit of time where 1 step equals 1 hour.
- type: Type of online transaction.
- amount: The amount of the transaction.
- nameOrig: Customer starting the transaction.
- oldbalanceOrg: Balance before the transaction.
- newbalanceOrig: Balance after the transaction.
- nameDest: Recipient of the transaction.
- oldbalanceDest: Initial balance of recipient before the transaction.
- newbalanceDest: The new balance of recipient after the transaction.
- isFraud: A binary label indicating whether the transaction is fraudulent (1) or not (0).
- isFlaggedFraud: Indicates if the transaction was flagged as suspicious by automated software.

Source : https://www.kaggle.com/datasets/drnimishadavis/onlinefraud

We're working with a dataset of shape (1,048,575, 11) but are selecting only the first 50,000 rows.

FEATURES AND TARGET VARIABLE

Features

- step: Represents a unit of time where 1 step equals 1 hour.
- type: Type of online transaction (e.g., PAYMENT, TRANSFER, CASH_OUT).
- amount: The amount of the transaction.
- oldbalanceOrg: Balance before the transaction.
- newbalanceOrig: Balance after the transaction.
- oldbalanceDest: Initial balance of recipient before the transaction.
- newbalanceDest: The new balance of recipient after the transaction.

Target

- isFraud: Target variable (1 if the transaction is fraudulent, 0 otherwise).

PROBLEM STATEMENT

Online fraud is a serious challenge faced by financial institutions, e-commerce platforms, and digital payment systems. Traditional rule-based fraud detection methods struggle to keep up with evolving fraud techniques. The goal of this project is to develop a machine learning model that can effectively identify fraudulent transactions with high accuracy while minimizing false positives.

Traditional rule-based fraud detection methods, which rely on predefined patterns and thresholds, often fail to keep up with evolving fraud techniques. As a result, there is an urgent need for a more dynamic and adaptive fraud detection system that can detect suspicious activities with high accuracy while minimizing false positives, which can result in inconvenience for legitimate users.

IMPORTANCE

Effective fraud detection is vital for safeguarding financial assets, maintaining the integrity of digital transactions, and protecting consumer trust. Machine learning models offer a robust solution by learning from historical transaction data and detecting hidden patterns indicative of fraudulent activities. By accurately identifying fraudulent transactions, such models can significantly reduce financial losses, prevent unauthorized access, and enhance overall transaction security. Moreover, minimizing false positives ensures a smoother user experience, preventing unnecessary disruptions. Ultimately, the successful implementation of machine learning in fraud detection will improve the safety, efficiency, and reliability of digital financial services, supporting the continued growth of online commerce and digital payments.

METHODOLOGY

1. DATA PREPROCESSING

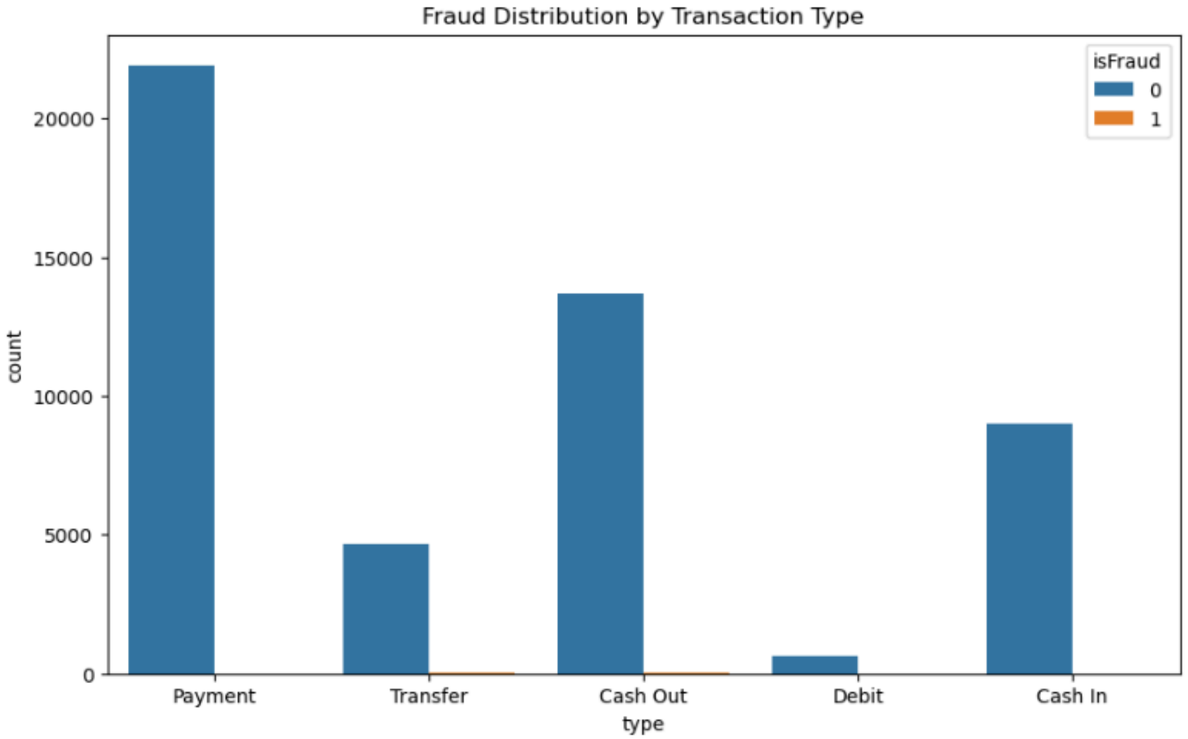
- **Handling missing values :**The dataset is cleaned by handling any missing or null values. For instance, columns with missing data are filled with appropriate values, such as the mean for numerical data.
- **Encoding variables :** Non-numeric columns, such as transaction types and account names, are encoded into numerical values using techniques like Label Encoding and custom mapping. This step ensures that categorical features can be processed by machine learning models.
- **Standardization:** To improve the performance of machine learning models, numerical features are standardized to ensure that all features are on a similar scale.
- **Outlier Detection and Handling :** Outliers were identified using boxplots and handled through the Interquartile Range (IQR) method to ensure data consistency and mitigate the impact of extreme values on model performance.

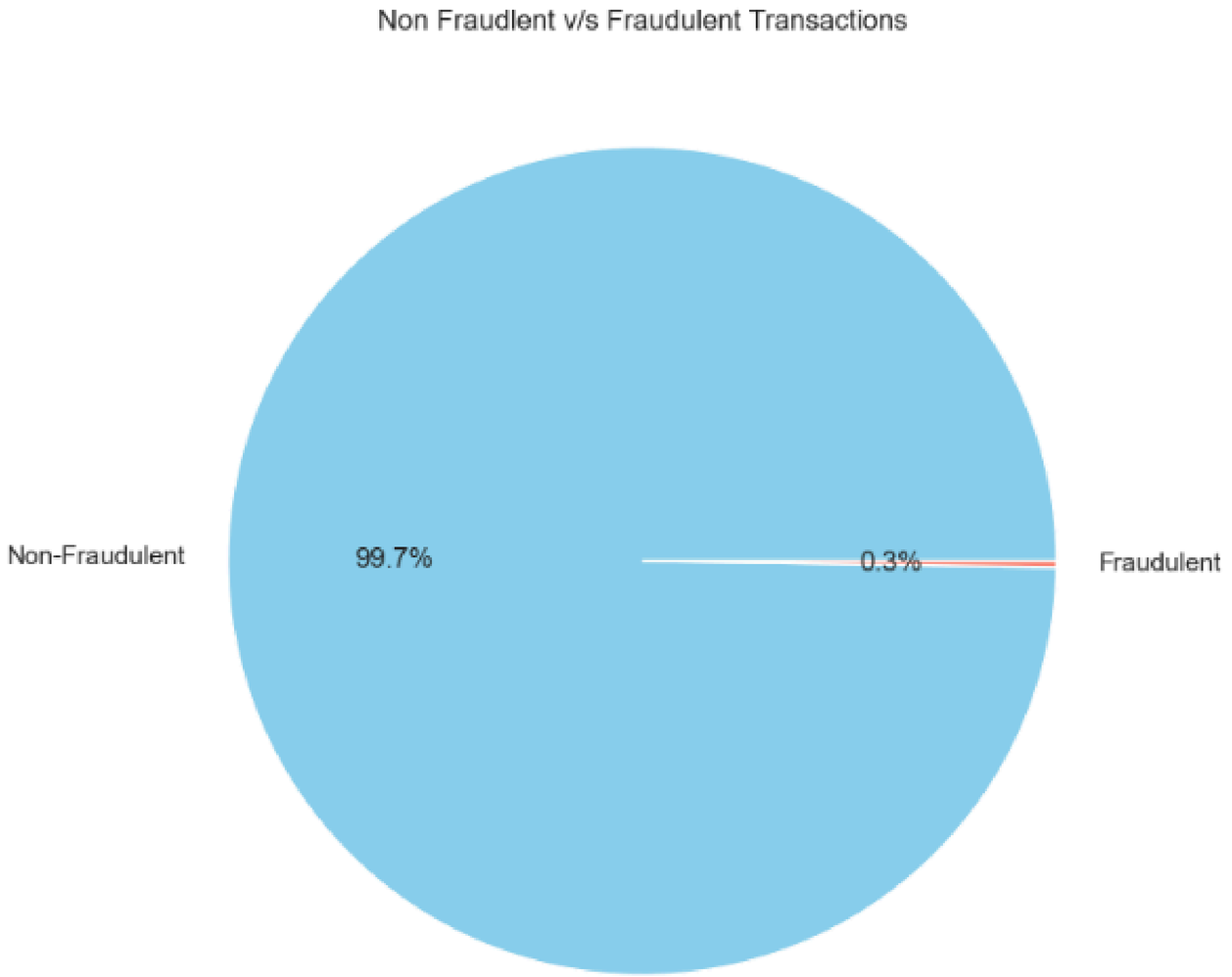
METHODOLOGY

2.EXPLORATORY DATA ANALYSIS (EDA)

Fraud Transactions Analysis

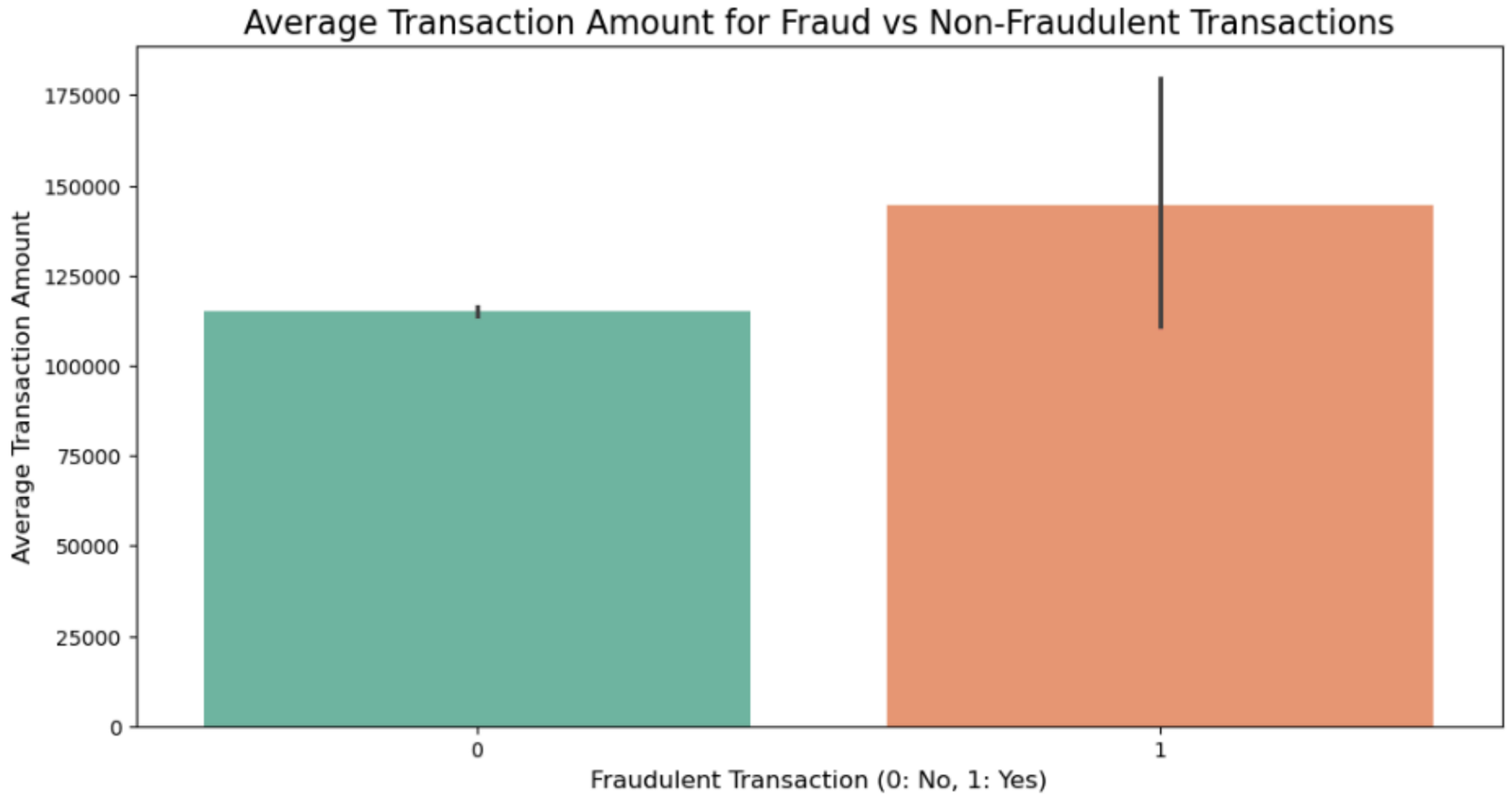
- The majority of transactions for all types are non-fraudulent (isFraud = 0), represented by the dark blue bars.
- Fraudulent transactions (isFraud = 1, shown in orange) are very minimal in comparison.
- Fraud is almost negligible in Payment, Cash In, and Debit transactions.





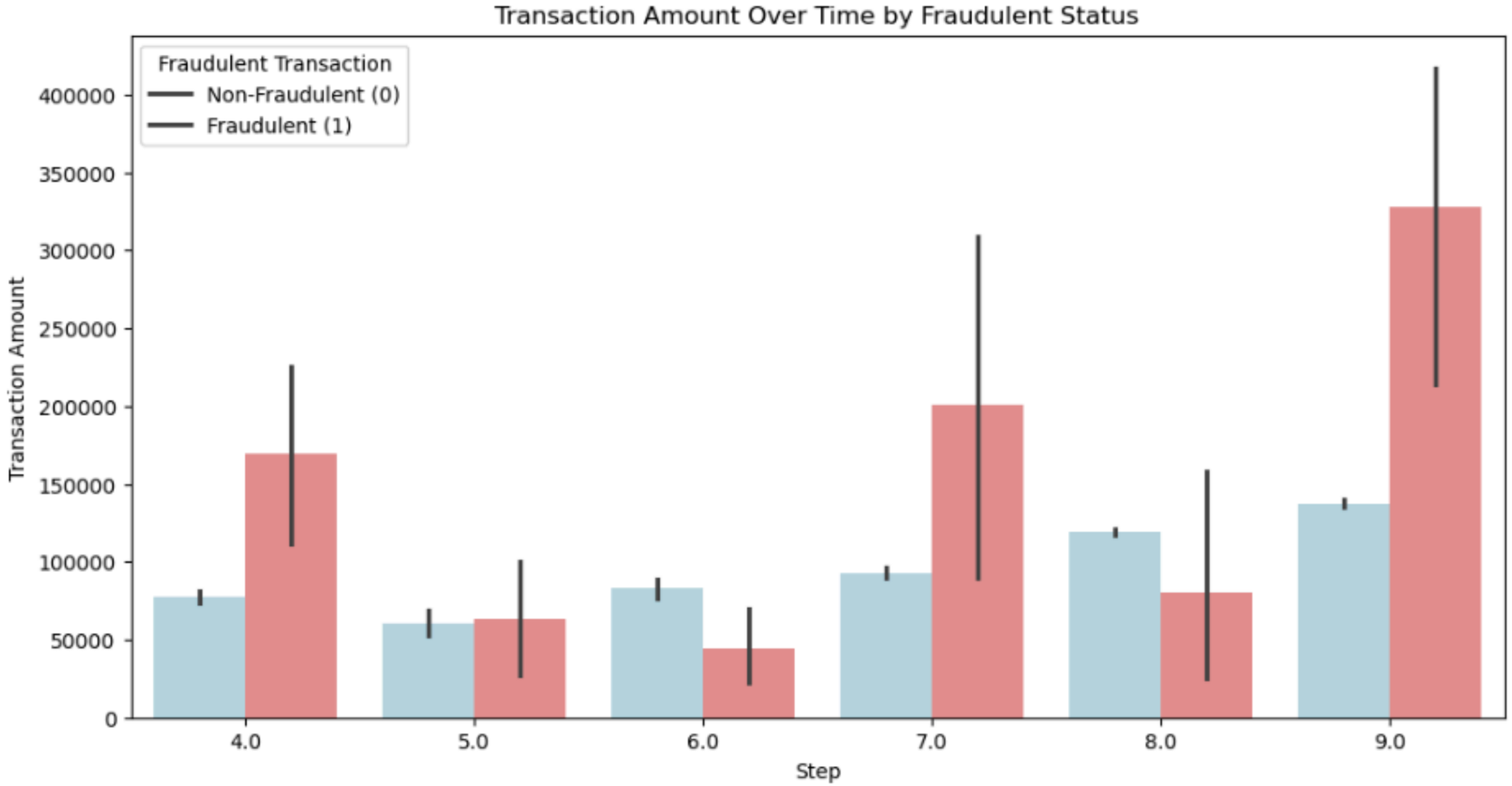
Fraudulent vs Non-Fraudulent Transaction Distribution

- Highly Imbalanced Data : Non-Fraudulent transactions dominate
- Fraud is Extremely Rare



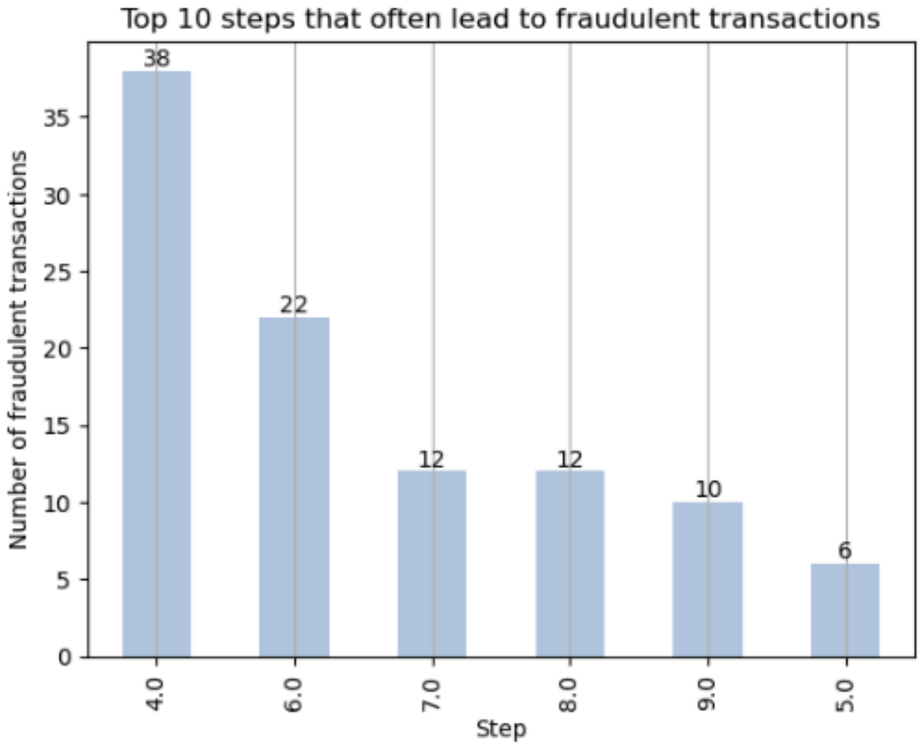
Average Transaction Amount for Fraud vs Non-Fraud

- Higher Average Transaction Amount for Fraudulent Transactions
- Greater Variance in Fraudulent Transactions
- Fraudulent Transactions are More Likely to Involve Large Sums



Transaction Amount Over Time by Fraudulent Status

- Fluctuations in Transaction Amounts Over Steps
- Fraudulent Transactions (Red) Show High Variability
- Non-Fraudulent Transactions (Blue) Are More Stable
- Steps (like Step 4, 7, and 9) have spikes in fraudulent transaction amounts

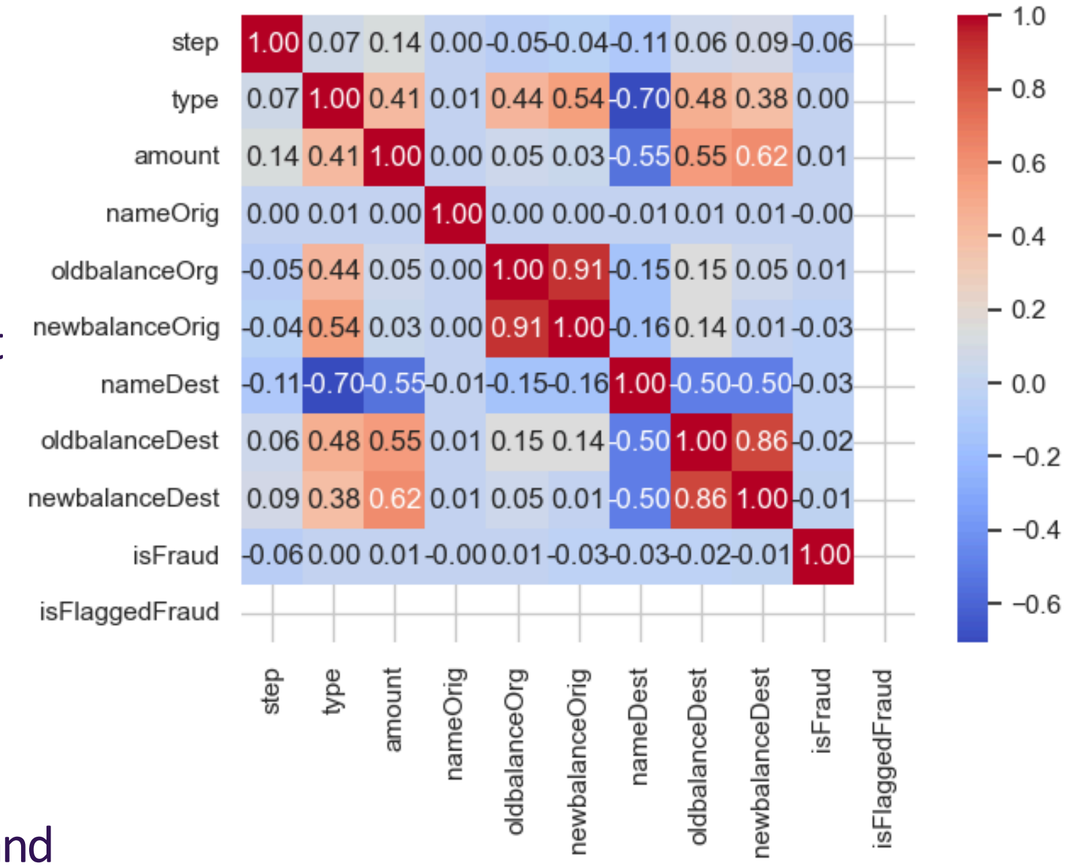


Top 10 Steps Leading to Fraudulent Transactions

- Step 4 has the highest fraudulent transactions (38 cases)
- Step 6 is the second most frequent fraud step (22 cases)
- Steps 7 and 8 have moderate fraud cases (12 each)
- Step 9 has slightly lower fraud (10 cases)
- Step 5 shows the lowest fraud count (6 cases)

Correlation Analysis

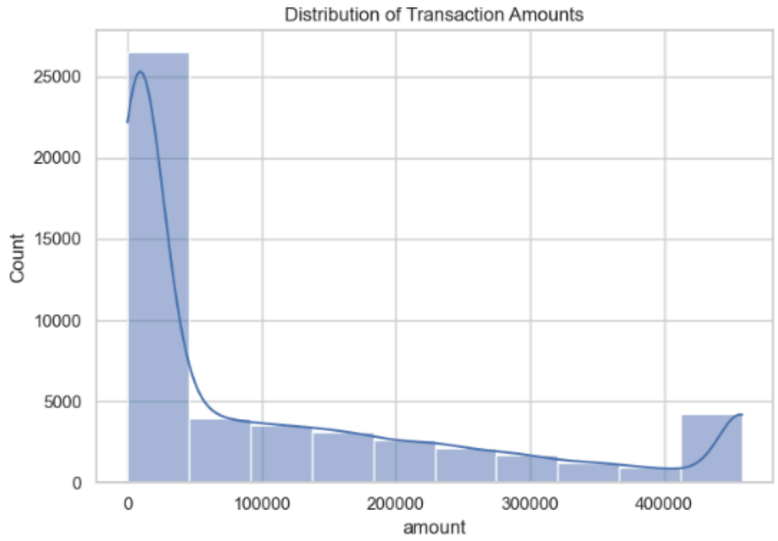
- We computed the Pearson Correlation which Measures linear relationships between variables.
- A heatmap was used to visualize the correlation matrix, making it easier to identify strong or weak relationships.
- Amount & Balance Features are Key: Features like amount, oldbalanceOrig, and oldbalanceDest are more informative for predicting fraud.
- Name-related Features Can Likely Be Dropped: Since nameOrig and nameDest have almost no impact, they might not be useful for fraud detection.



1 Histogram of Transaction Amounts (Left Plot)

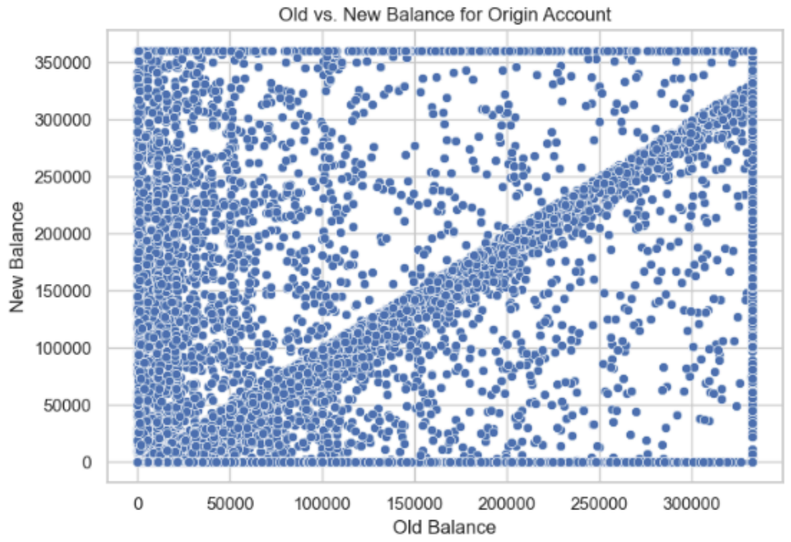
The distribution is right-skewed, meaning:

- Most transactions involve small amounts.
- A few transactions involve very large amounts, which could indicate potential fraud



2 Scatter Plot: Old vs. New Balance for Origin Account (Right Plot)

- The diagonal pattern suggests that in some cases, the new balance is exactly proportional to the old balance.
- However, many transactions do not follow a strict pattern, which could indicate:
  - Some accounts retain most of their balance after transactions.
  - Others have a significant reduction, possibly indicating large withdrawals or fraud.



Class distribution

- The dataset is highly imbalanced.
- There are 49,900 non-fraudulent transactions (Class 0) and only 100 fraudulent transactions (Class 1).
- Fraudulent transactions make up just 0.2% of the total data, which is a severe imbalance.

```
isFraud
0    49900
1      100
Name: count, dtype: int64
```

Descriptive Statistics

- Transaction Type: The dataset contains mostly Transfer transactions.
- Transaction Amounts: There are both very small and large transactions, with high variability.
- Balances: Origin and destination balances have large differences, with a much higher balance observed for origin accounts compared to destination ones.
- Fraudulent Transactions: Fraudulent transactions are extremely rare,.

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
count	50000.000000	50000.00000	5.000000e+04	50000.000000	5.000000e+04	5.000000e+04	50000.000000	5.000000e+04	5.000000e+04	50000.000000
mean	7.453800	1.40200	1.562645e+05	24999.500000	7.333085e+05	7.473080e+05	9525.570860	8.444788e+05	1.164464e+06	0.002000
std	2.056636	1.49049	3.243949e+05	14433.901067	2.202383e+06	2.239965e+06	8351.299295	2.433474e+06	2.915438e+06	0.044677
min	1.000000	0.00000	6.300000e-01	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000
25%	7.000000	0.00000	7.734573e+03	12499.750000	0.000000e+00	0.000000e+00	2870.750000	0.000000e+00	0.000000e+00	0.000000
50%	8.000000	1.00000	3.341482e+04	24999.500000	1.703072e+04	0.000000e+00	5873.500000	2.126000e+03	0.000000e+00	0.000000
75%	9.000000	2.00000	1.878390e+05	37499.250000	1.331716e+05	1.440106e+05	15998.250000	4.632050e+05	8.649264e+05	0.000000
max	9.000000	4.00000	1.000000e+07	49999.000000	2.850000e+07	2.860000e+07	28498.000000	3.010000e+07	3.200000e+07	1.000000

METHODOLOGY

3.MODELING

1. Model Choice: The following classification models were implemented for predicting fraudulent transactions:

- Logistic Regression
- Decision Trees
- Random Forest
- K-Nearest Neighbors (KNN)
- Support Vector Machines (SVM)
- Naïve Bayes
- Ensemble Models:
  - AdaBoost
  - Gradient Boosting
  - XGBoost

METHODOLOGY

3.MODELING

2. Scaling: Feature scaling was applied to standardize the data for certain models that are sensitive to feature magnitude.

3.Training and Testing: The dataset was split into training and test sets to validate the models. The training set was used to train the models, and the test set was used for evaluation.

INITIAL MODEL PERFORMANCE:

#	Model	Train Accuracy (%)	Test Accuracy (%)
0	Random Forest Classifier	100.0000	99.95
1	Logistic Regression	99.8200	99.81
2	SVM	99.8000	99.80
3	KNN	99.8500	99.85
4	Naive Bayes	99.7950	99.79
5	Decision Tree	99.8600	99.82
6	AdaBoost	99.8575	99.90
7	Gradient Boosting	96.5800	96.55
8	XGBoost	99.9125	99.97

- XGBoost achieved the highest accuracy of 99.97%, making it the best-performing model in this case.
- Random Forest Classifier followed closely with 99.95% accuracy, which is still a very strong performance.

HANDLING IMBALANCED DATA - SMOTE (SYNTHETIC MINORITY OVER-SAMPLING TECHNIQUE)

Model	Accuracy (%)	F1-score	AUC-ROC	Confusion Matrix
Random Forest Classifier	99.94	0.9994	0.999997	[[9969, 11], [1, 9979]]
Logistic Regression	91.34	0.9148	0.968727	[[8959, 1021], [707, 9273]]
SVM	94.17	0.9428	N/A	[[9206, 774], [389, 9591]]
KNN	99.54	0.9954	0.99854	[[9888, 92], [0, 9980]]
Naive Bayes	87.13	0.8817	0.958881	[[7821, 2159], [410, 9570]]
Decision Tree	96.37	0.9647	0.989266	[[9348, 632], [92, 9888]]
AdaBoost	97.77	0.9779	0.997868	[[9635, 345], [101, 9879]]
Gradient Boosting	99.79	0.9979	0.999961	[[9938, 42], [0, 9980]]
XGBoost	99.16	0.9916	0.999626	[[9853, 127], [41, 9939]]

- Random Forest and Gradient Boosting achieved near-perfect accuracy (99.94% and 99.79%).
- Confusion Matrix Insights: The number of false positives was low, and false negatives were minimized.
- Random Forest and Gradient Boosting emerged as the best models, with nearly perfect accuracy and high F1-scores.

HYPERPARAMETER TUNING

Model	Best Parameters	Training Accuracy (%)	Test Accuracy (%)
Random Forest Classifier	`{'n_estimators': 2500, 'min_samples_split': 6, 'min_samples_leaf': 3, 'max_features': 'sqrt', 'max_depth': 19, 'class_weight': {0: 1.02, 1: 0.98}, 'bootstrap': True}`	99.96	99.92
Logistic Regression	`{'C': 10}`	91.32	91.37
SVM	`{'kernel': 'rbf', 'C': 10}`	99.03	99.02
KNN	`{'n_neighbors': 3}`	99.81	99.65
Naive Bayes	`{'var_smoothing': 1e-09}`	87.09	87.13
Decision Tree	`{'min_samples_split': 5, 'max_depth': 20}`	99.99	99.86
AdaBoost	`{'n_estimators': 100}`	98.76	98.87
Gradient Boosting	`{'n_estimators': 100, 'learning_rate': 0.1}`	99.18	99.30
XGBoost	`{'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.1}`	99.77	99.74

- Top Performers: Random Forest Model have 99.96% training accuracy, with test accuracies 99.92%. This model benefit from fine-tuned hyperparameters and demonstrate excellent generalization.
- Based on test accuracy,Random Forest is the best choices.

MODEL PERFORMANCE ACROSS EACH STAGE

Model Performance Across Each Stage:

	Model	Accuracy_Initial	Accuracy_Oversampling	Accuracy_Hyperparameter_Tuning
0	Random Forest Classifier	99.95	99.939880	99.924850
1	Logistic Regression	99.81	91.342685	91.372745
2	SVM	99.80	94.173347	99.018036
3	KNN	99.85	99.539078	99.654309
4	Naive Bayes	99.79	87.129259	87.129259
5	Decision Tree	99.82	96.372745	99.864729
6	AdaBoost	99.90	97.765531	98.867735
7	Gradient Boosting	96.55	99.789579	99.298597
8	XGBoost	99.97	99.158317	99.744489

- Random Forest Classifier is a strong contender, achieving 99.92% accuracy, which is excellent and very close to 100%.

METHODOLOGY

4.OPTIMAL MODEL SELECTION

FINAL MODEL – RANDOM FOREST CLASSIFIER

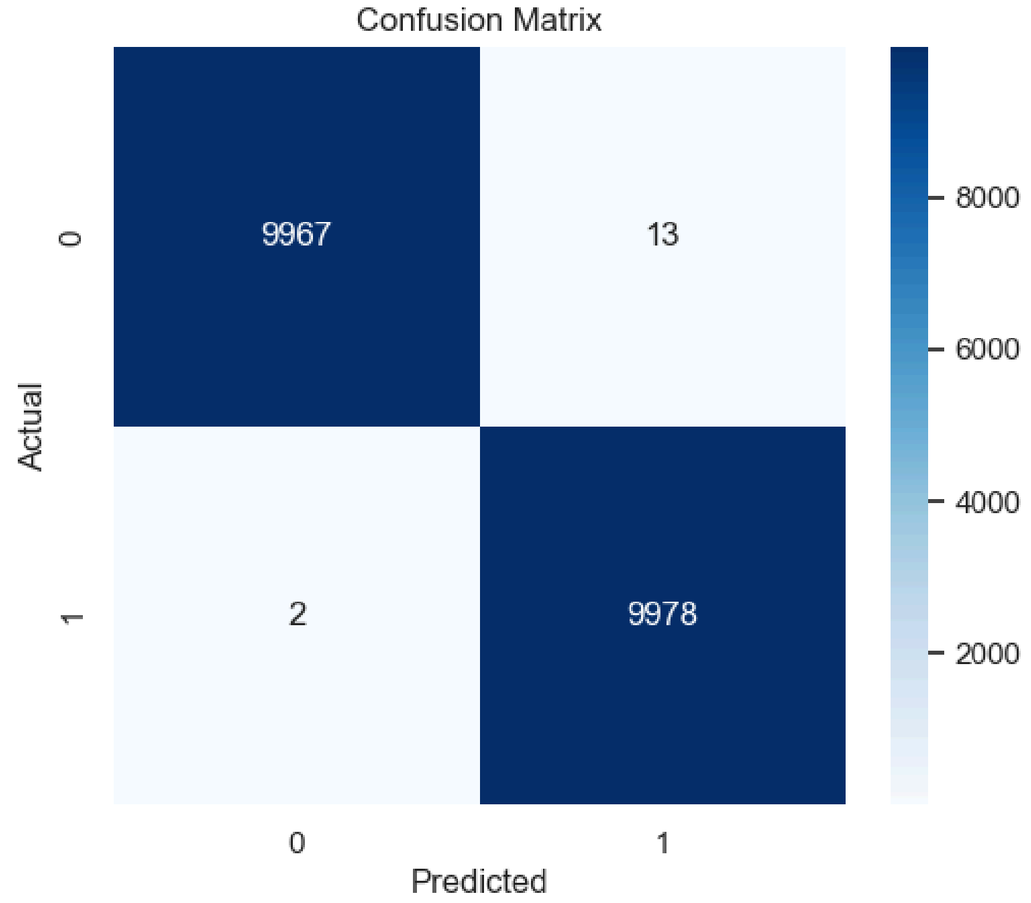
✅ Train Accuracy: 0.9996
✅ Test Accuracy: 0.9992

Confusion Matrix:

9967	13
2	9978

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	9980
1	1.00	1.00	1.00	9980
accuracy			1.00	19960
macro avg	1.00	1.00	1.00	19960
weighted avg	1.00	1.00	1.00	19960



KEY FINDINGS – RANDOM FOREST

✅ Train & Test Accuracy

- Training Accuracy: 99.96%
- Test Accuracy: 99.92%

Confusion Matrix:

- True Positives (9978): Correctly predicted positive cases.
- True Negatives (9967): Correctly predicted negative cases.
- False Positives (13): Incorrectly classified as positive.
- False Negatives (2): Incorrectly classified as negative.

Key Insight: Very few misclassifications, showing high precision & recall.

KEY FINDINGS – RANDOM FOREST

Classification Report:

- Precision (1.00): Almost no false positives.
- Recall (1.00): Almost no false negatives.
- F1-score (1.00): The harmonic mean of precision and recall is perfect.

Comprehensive Hyperparameter Tuning

- Well-Optimized Model: The near-perfect performance suggests that hyperparameter tuning has been performed effectively, optimizing for both bias and variance.

RECOMMENDATIONS

- Real-World Testing:
  - Test on unseen data for practical validation.
- Monitor for Overfitting:
  - Continuous monitoring post-deployment.

SOLUTIONS

📌 Real-World Testing

- Test on a completely unseen dataset.
- Deploy in a test environment before production.
- Conduct A/B Testing to compare with existing models.

📌 Monitor for Overfitting

- Implement Model Drift Detection.
- Track weekly/monthly performance metrics.
- Compare predicted vs. actual outcomes.

IMPACT

📌 1. Real-World Testing (Test on Unseen Data)

✅ Impact: Improves model generalization, ensuring reliable performance when deployed in real-world scenarios. Reduces the risk of unexpected failures.

📌 2. Monitor for Overfitting (Continuous Model Monitoring)

✅ Impact: Helps maintain long-term model accuracy and stability. Detects performance degradation early, allowing timely adjustments.

🔹 Overall Impact:

✔ More reliable and interpretable AI model.

✔ Prevents failures in real-world applications.

✔ Enhances long-term performance and adaptability.

CONCLUSION

This project successfully demonstrated the effectiveness of the Random Forest Classifier in achieving high accuracy for the given dataset. Through various techniques such as oversampling, hyperparameter tuning.

The Random Forest Classifier achieves exceptional performance with high accuracy, precision, recall, and F1-score.

✅ Minimal false positives & false negatives indicate robust generalization.

✅ The model is reliable and can be confidently deployed for predictions.

✅ Future Recommendations:

- Continuously monitor for data drift to maintain long-term performance.
- Deploy in a controlled environment and collect real-world feedback for further fine-tuning.

🚀 The model is ready for deployment, but ongoing monitoring and improvements are necessary to ensure consistent real-world performance.

THANK YOU

https://www.kaggle.com/drnimishadavis