ITCS – 6150

ARTIFICIAL INTELLIGENCE

FINAL PROJECT REPORT

AI-DRIVEN FRAUD DETECTION FOR DIGITAL PAYMENTS

Team 11:

1. Charan Gujjalapudi
2. Vishal Anton
3. Saisuraj Jindam
4. Sai Harsha Chenna
5. Prashanth Lakkakula

1. Abstract

Financial transactions have been transformed by the quick development of digital payment systems, which provide unmatched accessibility and convenience. Traditional rule-based fraud detection systems are becoming less and less effective, nevertheless, as a result of the vulnerabilities that these advancements have created to sophisticated fraud schemes. Using machine learning and deep learning algorithms to evaluate transaction patterns, spot fraudulent activity, and reduce false positives, this study suggests an AI-driven method for digital payment fraud detection. Through the integration of behavioral analytics and supervised and unsupervised learning methodologies, the system seeks to function in real-time and deliver actionable insights via an accessible interface. By developing a scalable and reliable fraud detection ecosystem, the ultimate objective is to improve digital payment security, lower financial losses, and increase customer trust.

2. Introduction

Globally, the financial environment has changed due to the extensive use of digital payment systems, which make transactions quicker and easier. But this quick development has also made people more susceptible to increasingly complex scam schemes. Fraudsters are always coming up with new and sophisticated ways to take advantage of flaws in traditional security systems, which mostly use rule-based fraud detection. Due to their frequent inability to adjust to new fraud trends, these antiquated systems put consumers and financial institutions at serious danger.

This initiative utilizes AI to create an advanced fraud detection system specifically for digital payment settings. This approach aims to identify and prevent fraudulent behavior in real-time by combining behavioral analytics with sophisticated machine learning methods. The proposed approach tackles the limitations of existing systems and enhances the security and reliability of digital payment networks through constant monitoring of transaction data, detection of abnormal patterns, and minimizing false alerts.

By using this cutting-edge strategy, the project hopes to transform fraud detection and prevention and guarantee the security and reliability of digital payment systems for both institutions and users.

3. Fraud Detection System Design

The aim of this project is to create a real-time fraud detection system using artificial intelligence (AI). The goal of the system is to decrease incorrect identifications and uncover unauthorized transactions by utilizing machine learning and deep learning techniques.

Important elements of the design consist of:

- **Supervised and Unsupervised Learning Models**:
  Transaction histories, user activity, and demographic information will all be examined by the system. While unsupervised learning models will detect anomalies in real-time, even for unseen fraud situations, supervised models will be trained on labeled datasets to identify known fraud patterns.
- **Behavioral Analytics Integration**:
  An extra degree of security is provided by the system's ability to identify departures from normal patterns, such as odd transaction amounts or geographic irregularities, by tracking user behavior.

4. Actionable Insights and System Scalability

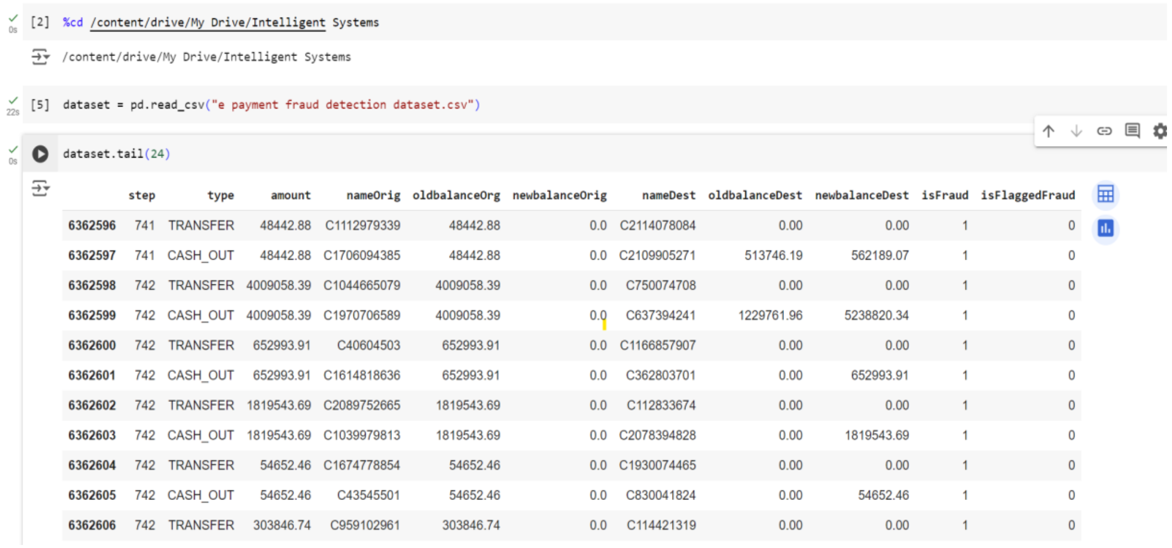
The project also focuses on creating a scalable and intuitive fraud case management system, ensuring ease of use for analysts and decision-makers.

Key deliverables include:

- **Visual Dashboard for Fraud Case Management**:
  Real-time insights regarding flagged transactions will be available through an intuitive interface that shows parameters including transaction details, behavioral patterns, and the likelihood of fraud. This dashboard will enable analysts to take prompt, effective action.
- **Enhanced Fraud Detection Accuracy**:
  By combining machine learning with deep learning techniques, the system aims to achieve high accuracy rates (95–99%), significantly reducing false positives and improving trust in digital transactions. Scalability will allow seamless adaptation to larger datasets and evolving fraud patterns as digital payment systems grow.

5. Implementation

- **Dataset Analysis:**



```
[2] %cd /content/drive/My Drive/Intelligent Systems
/content/drive/My Drive/Intelligent Systems
```

```
[5] dataset = pd.read_csv("e payment fraud detection dataset.csv")
```

```
dataset.tail(24)
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
6362596	741	TRANSFER	48442.88	C1112979339	48442.88	0.0	C2114078084	0.00	0.00	1	0
6362597	741	CASH_OUT	48442.88	C1706094385	48442.88	0.0	C2109905271	513746.19	562189.07	1	0
6362598	742	TRANSFER	4009058.39	C1044665079	4009058.39	0.0	C750074708	0.00	0.00	1	0
6362599	742	CASH_OUT	4009058.39	C1970706589	4009058.39	0.0	C637394241	1229761.96	5238820.34	1	0
6362600	742	TRANSFER	652993.91	C40604503	652993.91	0.0	C1166857907	0.00	0.00	1	0
6362601	742	CASH_OUT	652993.91	C1614818636	652993.91	0.0	C362803701	0.00	652993.91	1	0
6362602	742	TRANSFER	1819543.69	C2089752665	1819543.69	0.0	C112833674	0.00	0.00	1	0
6362603	742	CASH_OUT	1819543.69	C1039979813	1819543.69	0.0	C2078394828	0.00	1819543.69	1	0
6362604	742	TRANSFER	54652.46	C1674778854	54652.46	0.0	C1930074465	0.00	0.00	1	0
6362605	742	CASH_OUT	54652.46	C43545501	54652.46	0.0	C830041824	0.00	54652.46	1	0
6362606	742	TRANSFER	303846.74	C959102961	303846.74	0.0	C114421319	0.00	0.00	1	0

The dataset used for this project focuses on digital payment fraud detection, encompassing various transaction types and associated attributes. The data contains crucial details such as transaction type, amounts, and account balances for both origin and destination accounts, alongside labels indicating fraudulent activity. Key highlights of the dataset analysis include:

➢ **Data Overview**:

- The dataset is loaded and inspected using Python libraries, providing a clear understanding of its structure and contents.
- It includes fields like step, type, amount, oldbalanceOrg, newbalanceOrg, oldbalanceDest, newbalanceDest, and flags such as isFraud and isFlaggedFraud.

➢ **Initial Observations**:

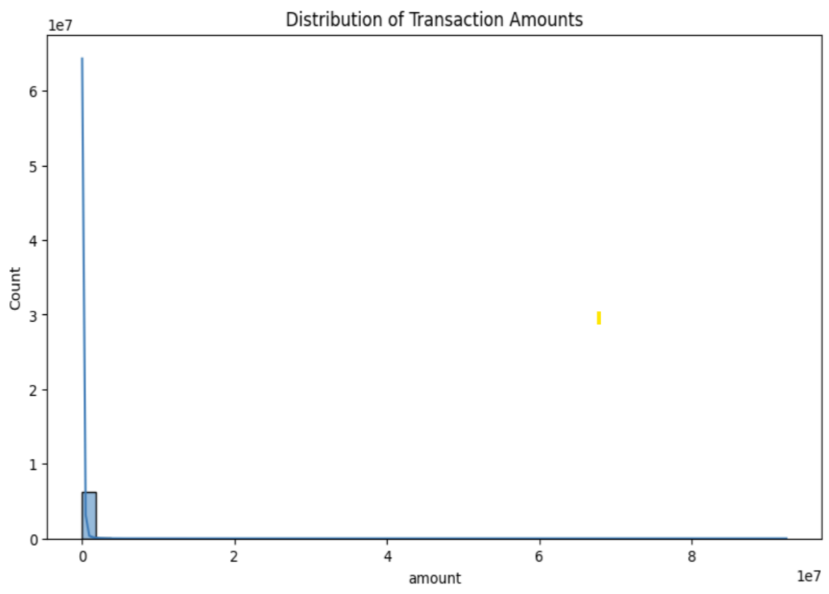
- A subset of the dataset reveals multiple transaction types, such as **TRANSFER** and **CASH_OUT**, which are the primary focus for fraud detection.
- The isFraud column serves as the target variable, where a value of 1 indicates fraudulent transactions.

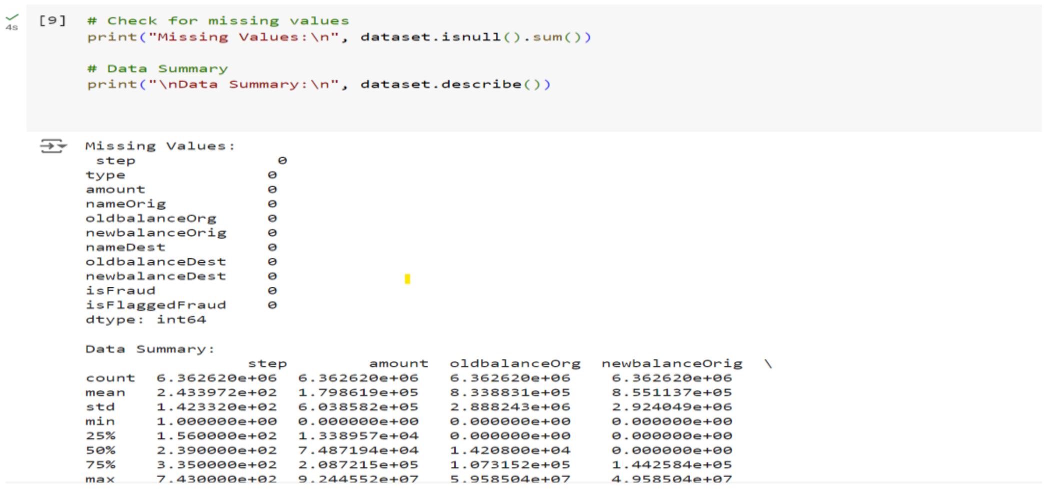
➢ **Data Characteristics**:

- Significant variation exists in transaction amounts and balances, necessitating further preprocessing to handle scaling and normalization.
- Many transactions show imbalances between oldbalanceOrg and newbalanceOrg as well as between oldbalanceDest and newbalanceDest, indicating potential anomalies.

➢ **Fraud Distribution**:

- The class imbalance is apparent in the isFraud column, which reveals that the majority of transactions are non-fraudulent. To facilitate effective model training, methodologies like the Synthetic Minority Oversampling Technique (SMOTE) are essential to address this mismatch.





```
[9] # Check for missing values
    print("Missing Values:\n", dataset.isnull().sum())

    # Data Summary
    print("\nData Summary:\n", dataset.describe())
```

```
Missing Values:
 step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64

Data Summary:
                step        amount  oldbalanceOrg  newbalanceOrig  \
count  6.362620e+06  6.362620e+06   6.362620e+06    6.362620e+06
mean   2.433972e+02  1.798619e+05   8.338831e+05    8.551137e+05
std    1.423320e+02  6.038582e+05   2.888243e+06    2.924049e+06
min    1.000000e+00  0.000000e+00   0.000000e+00    0.000000e+00
25%    1.560000e+02  1.338957e+04   0.000000e+00    0.000000e+00
50%    2.390000e+02  7.487194e+04   1.420800e+04    0.000000e+00
75%    3.350000e+02  2.087215e+05   1.073152e+05    1.442584e+05
max    7.430000e+02  9.244552e+07   5.958504e+07    4.958504e+07
```

Distribution of Transaction Amounts

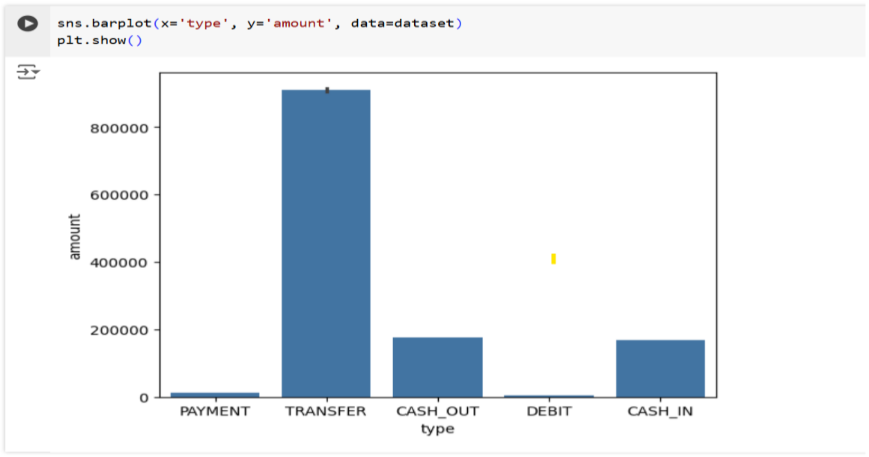
- The distribution of transaction amounts is highly skewed, with most transactions taking place at lesser amounts, according to a depiction of the data.
- High-value transactions are identified in the distribution as outliers, which may necessitate extra care during the preprocessing stage in order to lessen their influence on model training.
- This insight suggests the need for scaling and normalization to ensure model performance is not biased toward these extreme values.

Missing Values Analysis

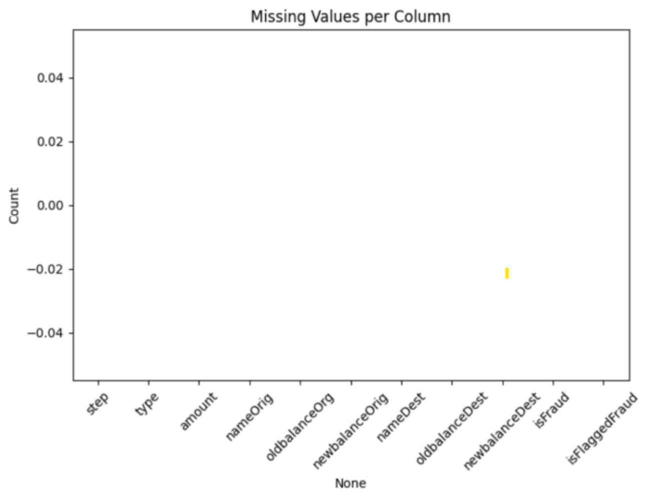
- A check for missing values confirms that the dataset is complete, with no missing entries across all columns. This eliminates the need for imputation techniques.
- Ensuring data completeness is critical for maintaining the integrity of the analysis and training process.

Summary Statistics

- The data summary provides key statistical metrics such as mean, standard deviation, minimum, and maximum values for numerical columns like amount, oldbalanceOrg, newbalanceOrg, and others.
- Notable observations include:
    - The amount column ranges from a minimum of 0 to a maximum of approximately 9.24 million, emphasizing the wide variance in transaction values.
    - Similarly, the balance columns (oldbalanceOrg, newbalanceOrg, etc.) exhibit significant variation, underlining the complexity of the data.



```
sns.barplot(x='type', y='amount', data=dataset)
plt.show()
```



Transaction Types and Amounts

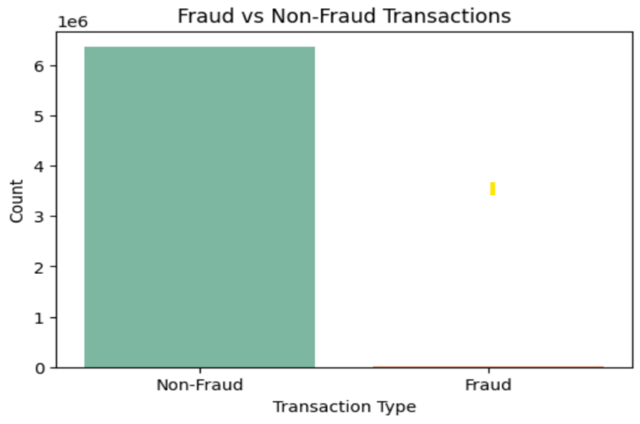
- A bar plot visualization of transaction types against their corresponding amounts reveals key insights:
  - **TRANSFER** transactions account for the highest average amount compared to other types, such as **PAYMENT**, **CASH_OUT**, **CASH_IN**, and **DEBIT**.
  - The disproportionately high transaction amounts for **TRANSFER** suggest that these transactions may carry a higher risk of fraud, warranting focused attention in model training and evaluation.
  - Other types like **DEBIT** and **PAYMENT** exhibit relatively lower average amounts, which might reflect their less frequent association with fraudulent activities.

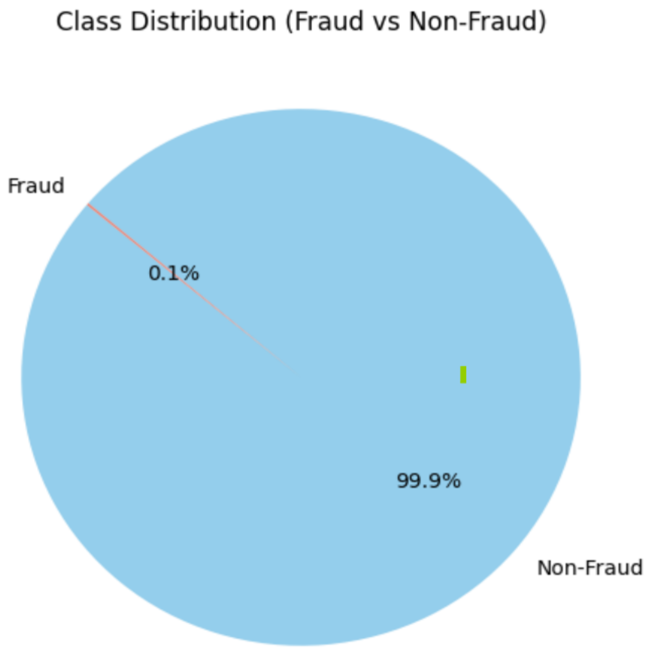
Missing Values Visualization

- The dataset was analyzed for missing values, and the results confirmed the absence of missing data across all columns. This is visually depicted in the chart, where no column has a count above zero for missing values.
- This clean dataset eliminates the need for imputation strategies, allowing for direct progression to feature engineering and modeling.

Key Observations

- The dominance of **TRANSFER** and **CASH_OUT** transactions in terms of transaction amounts aligns with industry trends where such transaction types are often targeted in fraud schemes.
- The absence of missing values ensures that no critical information is lost during the data preprocessing phase, enabling the system to learn effectively from complete data.





Class Imbalance in Fraud Detection

Fraud vs. Non-Fraud Transactions

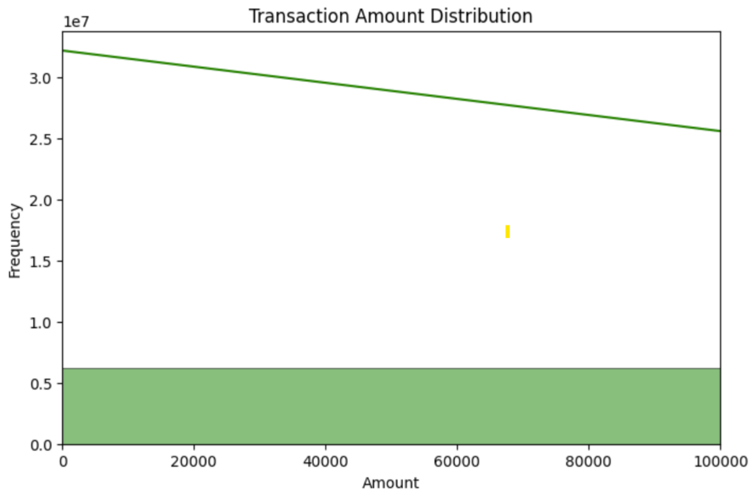
- A significant disproportion in class is depicted in a bar graph comparing fraudulent and non-fraudulent transactions. The dataset has a small proportion of fake transactions, with the majority of transactions being legitimate.
- The difference in distribution poses a challenge in training machine learning algorithms as they may favor predicting the dominant class (non-fraudulent), diminishing their ability to identify instances of fraud.

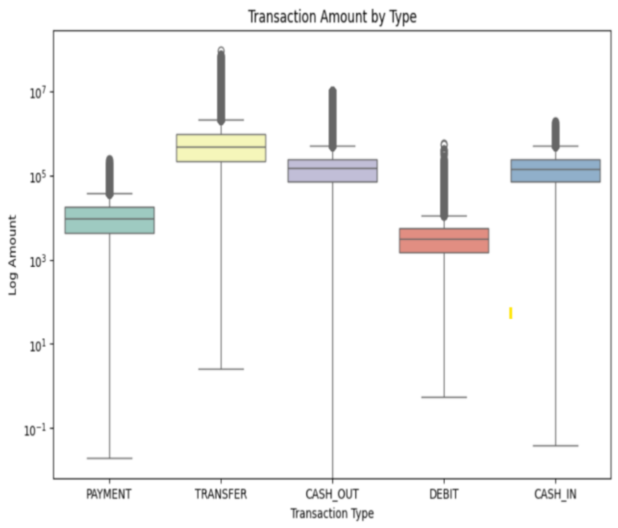
Class Distribution

- This inconsistency is also highlighted through a pie chart, illustrating that fraudulent transactions account for only 0.1% of the data set while non-fraudulent transactions make up approximately 99.9%.
- In order for the model to work well in identifying the minority class, dealing with such a severe imbalance requires using advanced methods during training, like using class-weighted loss functions or the Synthetic Minority Oversampling Technique (SMOTE).

Key Considerations

- The dataset's significant imbalance underscores the need for careful preprocessing and evaluation measures that are more appropriate for unbalanced classification tasks, like precision, recall, F1-score, and ROC-AUC.
- Creating a successful fraud detection system that accurately detects fraudulent transactions without being influenced by the dominant class necessitates tackling this imbalance.





Transaction Amount Distribution

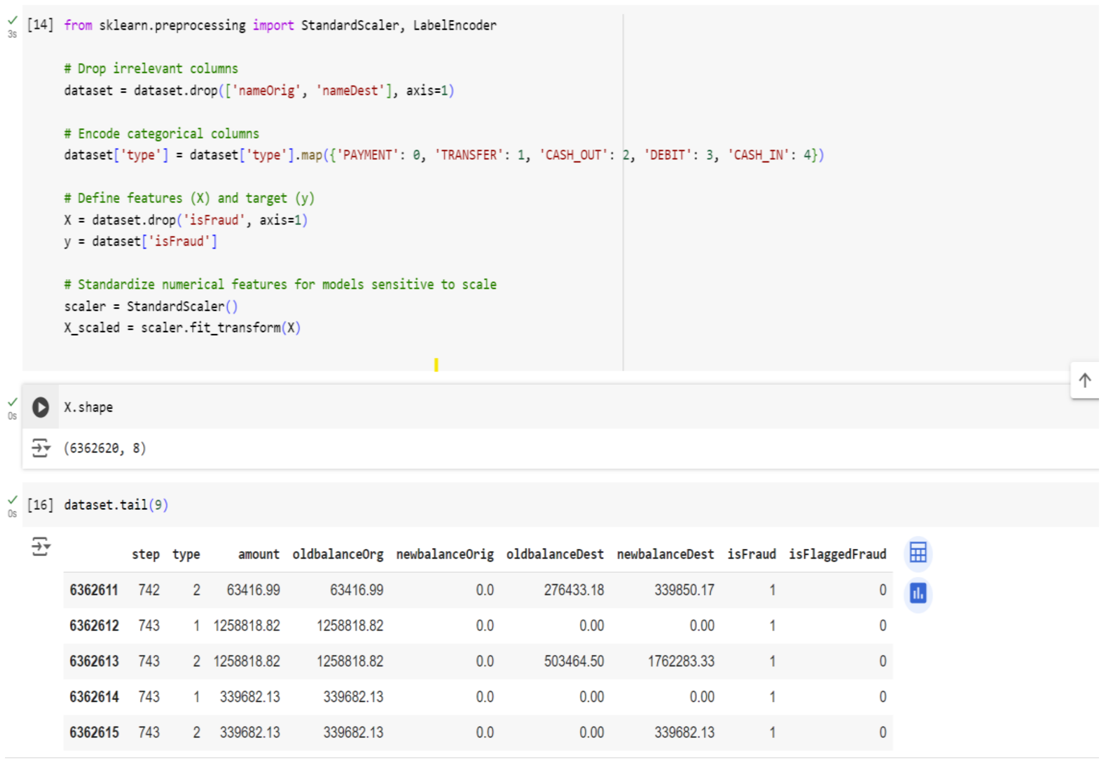
Overall Distribution

- The first visualization displays the distribution of transaction amounts across the dataset. A majority of the transactions are concentrated at lower values, with a steep decline as transaction amounts increase.
- This observation highlights a heavy skew in the data, which may require transformations like logarithmic scaling to enhance model performance by normalizing the range.

Transaction Amount by Type

- The second visualization, a box plot of transaction amounts by type, provides detailed insights:
    - **TRANSFER** and **CASH_OUT** transactions exhibit the highest variability in amounts, with numerous high-value outliers.
    - Transactions like **DEBIT**, **PAYMENT**, and **CASH_IN** have relatively lower median values and smaller ranges.
    - The logarithmic scale used in the y-axis effectively represents the vast differences in transaction amounts while managing the skewed distribution.

- **Data Preprocessing**



```
[14] from sklearn.preprocessing import StandardScaler, LabelEncoder

     # Drop irrelevant columns
     dataset = dataset.drop(['nameOrig', 'nameDest'], axis=1)

     # Encode categorical columns
     dataset['type'] = dataset['type'].map({'PAYMENT': 0, 'TRANSFER': 1, 'CASH_OUT': 2, 'DEBIT': 3, 'CASH_IN': 4})

     # Define features (X) and target (y)
     X = dataset.drop('isFraud', axis=1)
     y = dataset['isFraud']

     # Standardize numerical features for models sensitive to scale
     scaler = StandardScaler()
     X_scaled = scaler.fit_transform(X)
```

```
X.shape
```

```
(6362620, 8)
```

```
[16] dataset.tail(9)
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
6362611	742	2	63416.99	63416.99	0.0	276433.18	339850.17	1	0
6362612	743	1	1258818.82	1258818.82	0.0	0.00	0.00	1	0
6362613	743	2	1258818.82	1258818.82	0.0	503464.50	1762283.33	1	0
6362614	743	1	339682.13	339682.13	0.0	0.00	0.00	1	0
6362615	743	2	339682.13	339682.13	0.0	0.00	339682.13	1	0

The data preprocessing step is critical to ensuring that the dataset is clean, consistent, and ready for machine learning model training. This stage involves several key tasks to prepare the features and target variable for effective analysis.

1. Dropping Irrelevant Columns

- Columns such as nameOrig and nameDest, which act as unique identifiers and do not provide meaningful information for fraud prediction, were removed from the dataset.
- This helps reduce dimensionality and eliminates noise, focusing the model on features relevant to the detection task.

2. Encoding Categorical Features

- The type column, which categorizes transactions into PAYMENT, TRANSFER, CASH_OUT, DEBIT, and CASH_IN, was encoded into numerical values.
- A mapping approach was used to convert these categories into integers for compatibility with machine learning algorithms:
    - PAYMENT: 0
    - TRANSFER: 1
    - CASH_OUT: 2
    - DEBIT: 3

- CASH_IN: 4

3. Feature and Target Variable Separation

- The dataset was split into **features (X)** and **target (y)**:
    - X: Contains predictor variables such as type, amount, oldbalanceOrg, newbalanceOrg, oldbalanceDest, and newbalanceDest.
    - y: Contains the target variable isFraud, which indicates whether a transaction is fraudulent (1) or not (0).

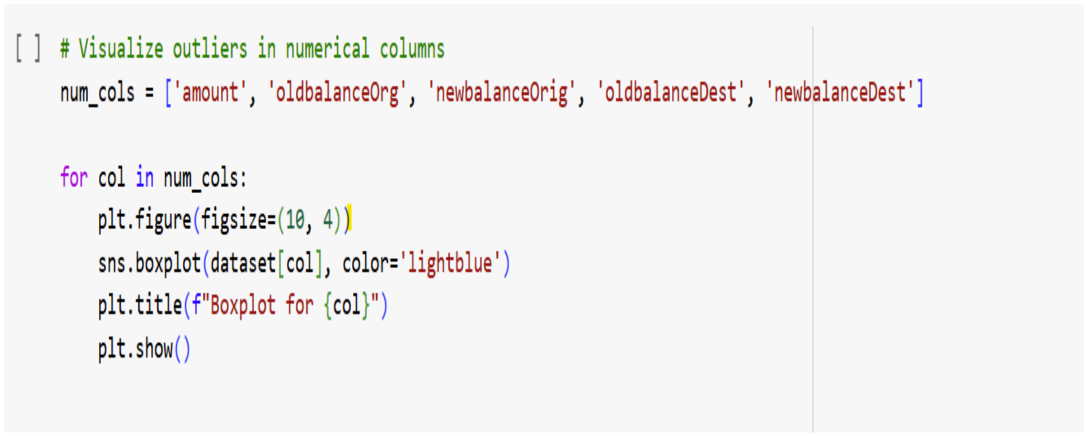
4. Scaling Numerical Features

- Numerical features like amount, oldbalanceOrg, newbalanceOrg, oldbalanceDest, and newbalanceDest were scaled using **StandardScaler** to standardize their values.
- Standardization ensures that all features contribute equally to the model by transforming them to have a mean of 0 and a standard deviation of 1.
- This is especially important for algorithms sensitive to feature magnitude, such as Logistic Regression and Neural Networks.

5. Dataset Overview After Preprocessing

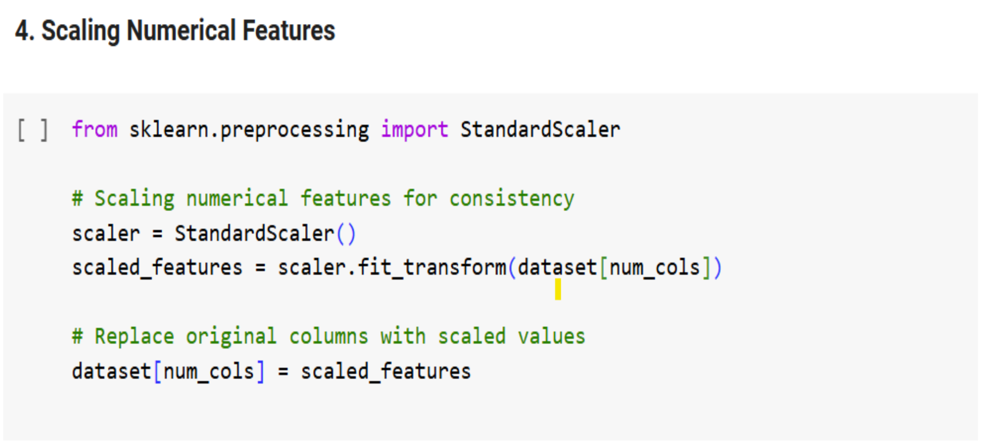
- The preprocessed dataset retains essential columns while ensuring numerical features are scaled and categorical features are encoded.
- The resulting shape of the dataset is (6,362,620 rows, 8 columns), highlighting the scale of the data being processed for fraud detection.

3. Check for Outliers



```
[ ] # Visualize outliers in numerical columns
    num_cols = ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest']

    for col in num_cols:
        plt.figure(figsize=(10, 4))
        sns.boxplot(dataset[col], color='lightblue')
        plt.title(f"Boxplot for {col}")
        plt.show()
```


4. Scaling Numerical Features

```
[ ] from sklearn.preprocessing import StandardScaler

    # Scaling numerical features for consistency
    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(dataset[num_cols])

    # Replace original columns with scaled values
    dataset[num_cols] = scaled_features
```

1. Checking for Outliers

- Outliers in numerical columns, such as amount, oldbalanceOrg, newbalanceOrg, oldbalanceDest, and newbalanceDest, were visualized using box plots.
- Box plots revealed significant outliers, particularly in transaction amounts and balances, which could skew the model's learning process.
- These visualizations enabled a focused approach to addressing outliers, ensuring that extreme values do not disproportionately affect training.

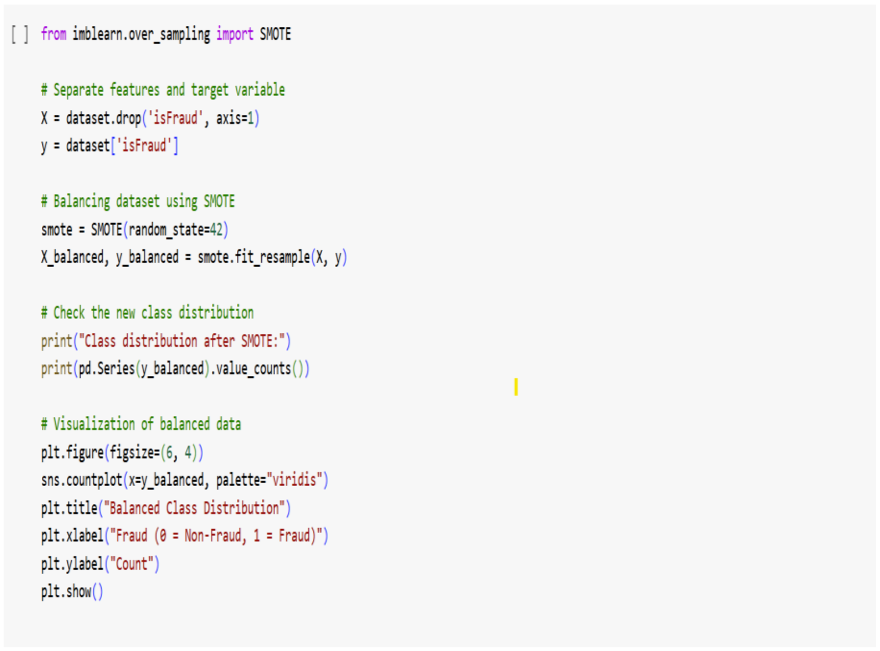
2. Scaling Numerical Features

- Numerical features with large ranges were scaled using **StandardScaler** to normalize the data. This process transforms the features to have a mean of 0 and a standard deviation of 1.
- Scaling is crucial for algorithms sensitive to feature magnitudes, such as Logistic Regression, Neural Networks, and K-Nearest Neighbors.
- The transformation was applied to columns like amount, oldbalanceOrg, and newbalanceOrg, ensuring consistency across features.

Implementation in Code

1. **Outlier Detection:**
   - A Python script iterates through numerical columns and generates box plots to identify and visualize outliers.
2. **Feature Scaling:**
   - The **StandardScaler** from sklearn.preprocessing was used to transform numerical columns.

- Scaled features replaced original columns, ensuring seamless integration into the dataset for subsequent steps.



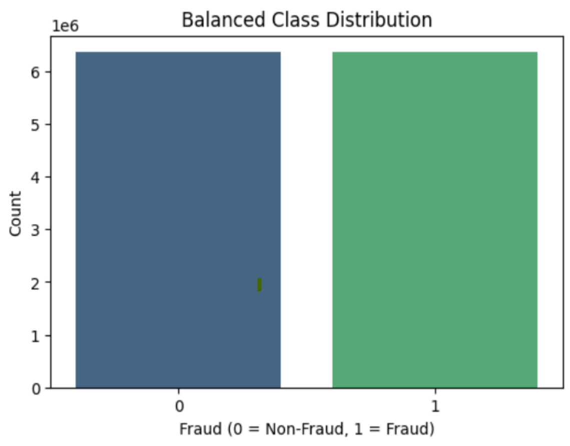
```
[ ] from imblearn.over_sampling import SMOTE

    # Separate features and target variable
    X = dataset.drop('isFraud', axis=1)
    y = dataset['isFraud']

    # Balancing dataset using SMOTE
    smote = SMOTE(random_state=42)
    X_balanced, y_balanced = smote.fit_resample(X, y)

    # Check the new class distribution
    print("Class distribution after SMOTE:")
    print(pd.Series(y_balanced).value_counts())

    # Visualization of balanced data
    plt.figure(figsize=(6, 4))
    sns.countplot(x=y_balanced, palette="viridis")
    plt.title("Balanced Class Distribution")
    plt.xlabel("Fraud (0 = Non-Fraud, 1 = Fraud)")
    plt.ylabel("Count")
    plt.show()
```



Handling Imbalanced Classes

Datasets for fraud detection frequently encounter significant imbalance in class distribution, with most transactions being genuine and only a small portion being deceptive. This discrepancy may result in machine learning models giving preference to the most common class, resulting in inaccurate detection of fraudulent behavior.

Problem Identification

- The isFraud column, representing the target variable, is highly imbalanced.

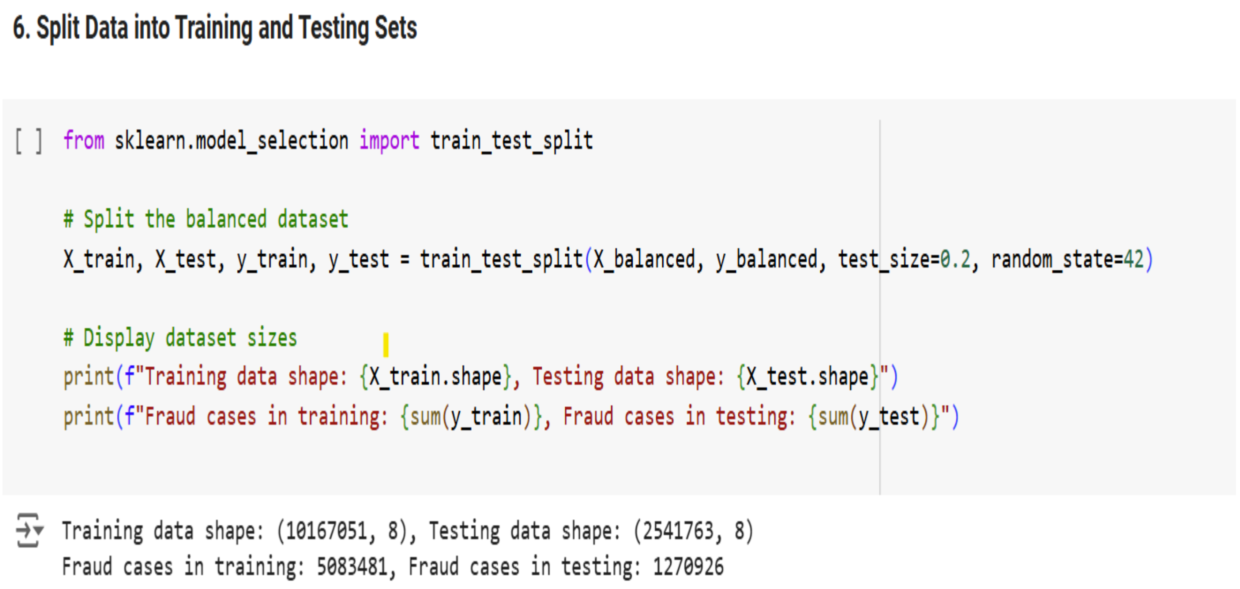
- Prior to using any methods, the dataset is mainly comprised of non-fraudulent transactions, posing a difficulty for models to understand the minority class.

Answer: Synthetic Minority Oversampling Technique(SMOTE)

To address this issue, SMOTE was employed. SMOTE is a method of oversampling that creates artificial samples for the minority group by blending existing data points. This makes sure that the minority group is sufficiently included without any repetition

Steps Implemented

1. **Separating Features and Target Variable**:
   - Features (X) and the target variable (y) were separated to prepare for resampling.
2. **Applying SMOTE**:
   - Using the SMOTE function from the imblearn library, the dataset was resampled to balance the classes.
   - This created a new dataset where both classes (fraudulent and non-fraudulent) were evenly distributed.
3. **Validation**:
   - The class distribution was checked after resampling to confirm the balance.
4. **Visualization**:
   - A bar plot was generated to visually confirm that the fraud (1) and non-fraud (0) classes are now equally represented.



6. Split Data into Training and Testing Sets

```
[ ] from sklearn.model_selection import train_test_split

    # Split the balanced dataset
    X_train, X_test, y_train, y_test = train_test_split(X_balanced, y_balanced, test_size=0.2, random_state=42)

    # Display dataset sizes
    print(f"Training data shape: {X_train.shape}, Testing data shape: {X_test.shape}")
    print(f"Fraud cases in training: {sum(y_train)}, Fraud cases in testing: {sum(y_test)}")
```

```
Training data shape: (10167051, 8), Testing data shape: (2541763, 8)
Fraud cases in training: 5083481, Fraud cases in testing: 1270926
```

Splitting the Data into Training and Testing Sets

In order to guarantee a thorough evaluation of the model, the processed and equalized dataset was divided into separate training and testing portions. This process guarantees that the machine learning models can undergo training on a specific set of data and then be assessed on a different set, which assists in evaluating how well they can perform on new data.

1. Train-Test Split

- The data was divided into training and testing datasets using the train_test_split function from the sklearn.model_selection library.
- The data was divided into an 80:20 ratio, with 80% used for training the models and 20% set aside for testing their performance.

2. Dataset Details

- **Training Data**:
    - Shape: **10,167,051 rows** with 8 features.
    - Fraud Cases: **508,341** fraudulent transactions.
- **Testing Data**:
    - Shape: **2,541,763 rows** with 8 features.
    - Fraud Cases: **127,092** fraudulent transactions.

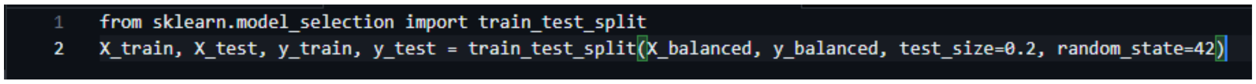
- This balanced split ensures that both fraudulent and non-fraudulent transactions are adequately represented in both subsets.

3. Random State

- A **random_state** of 42 was used to ensure reproducibility, allowing consistent results when the split is repeated.

Implementation

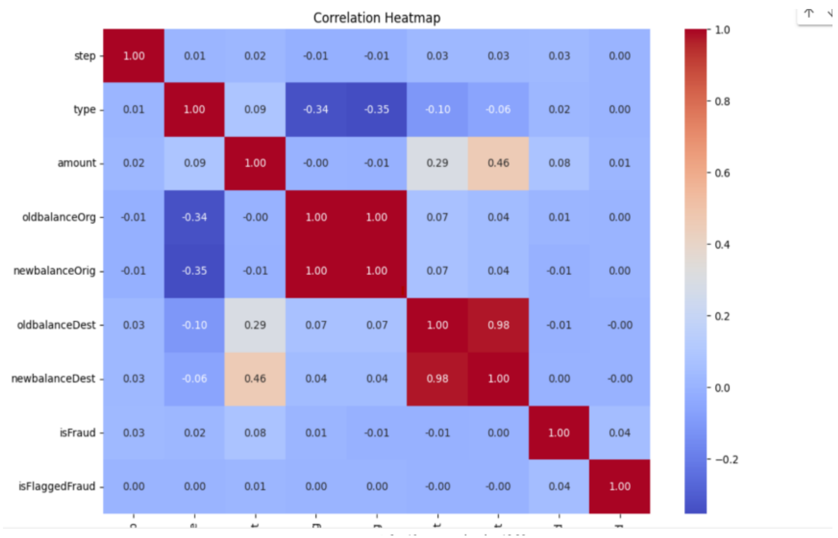
- The split was performed as follows:



```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_balanced, y_balanced, test_size=0.2, random_state=42)
```

4. Benefits of the Split

- Making sure the model is trained on a varied dataset and tested on fresh data to replicate real-world scenarios.
- Allows for evaluation of performance metrics like accuracy, precision, recall, and F1-score on the test dataset, ensuring the model's ability to effectively generalize to new transactions.



Correlation Heatmap

A correlation heatmap is a useful visualization tool for understanding the relationships between numerical features in the dataset. It highlights how strongly features are related to each other and to the target variable (isFraud).

Key Observations from the Heatmap

1. **Strong Correlations**:

- oldbalanceDest and newbalanceDest show a high positive correlation (**0.98**), suggesting that these features are closely related and might convey overlapping information.
- Similarly, oldbalanceOrg and newbalanceOrg have a strong correlation (**1.00**), indicating redundancy.

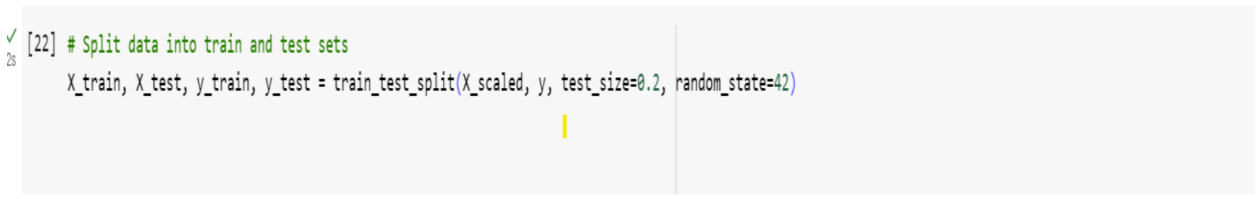
2. **Weak Correlations with isFraud**:
    - Features like amount (**0.08**) and type (**0.08**) show a slight correlation with the target variable (isFraud), indicating their potential importance in predicting fraud.
    - Other features, such as oldbalanceOrg, newbalanceOrg, and oldbalanceDest, show very weak or negligible correlation with isFraud, suggesting they might require feature engineering to enhance their predictive power.
3. **Uncorrelated Features**:
    - The step and isFlaggedFraud columns exhibit near-zero correlation with isFraud and other features, making them less useful for fraud prediction.

Insights and Implications

- **Feature Selection**: Highly correlated features (e.g., oldbalanceDest and newbalanceDest) might lead to multicollinearity, which can negatively impact some machine learning models. Feature reduction techniques or dropping redundant features might be necessary.
- **Predictive Power**: While most features show weak direct correlation with isFraud, combinations of features or feature interactions might capture non-linear relationships essential for fraud detection.
- **Focus on Relevant Features**: The amount and type features, though weakly correlated, appear to have the most direct impact on fraud detection, warranting further exploration.

5. DATA MODELLING

The data modeling stage concentrates on training and assessing various machine learning algorithms to accurately predict fraudulent transactions. This process guarantees the choice of a model that considers accuracy, precision, recall, and F1-score.

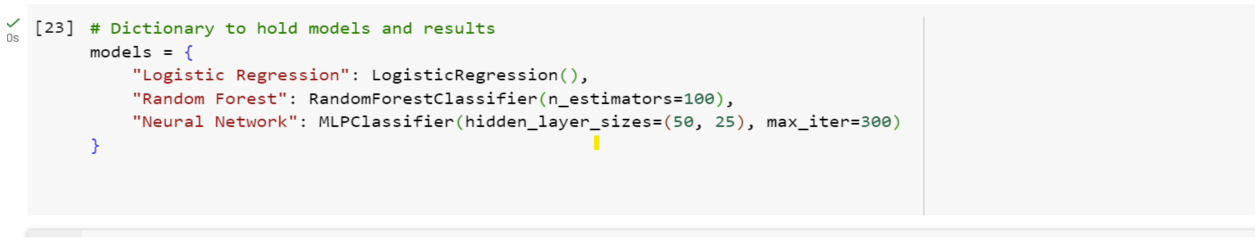


```
[22] # Split data into train and test sets
     X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

1. Selected Models

Several machine learning models were trained and evaluated to compare their performance:

- **Logistic Regression**:
    - A linear model suitable for binary classification tasks, providing interpretable results.
- **Random Forest**:
    - An ensemble learning method using decision trees, robust to overfitting and capable of handling large datasets.
- **Neural Network**:
    - A multi-layer perceptron (MLP) with hidden layers designed to capture complex relationships in the data.

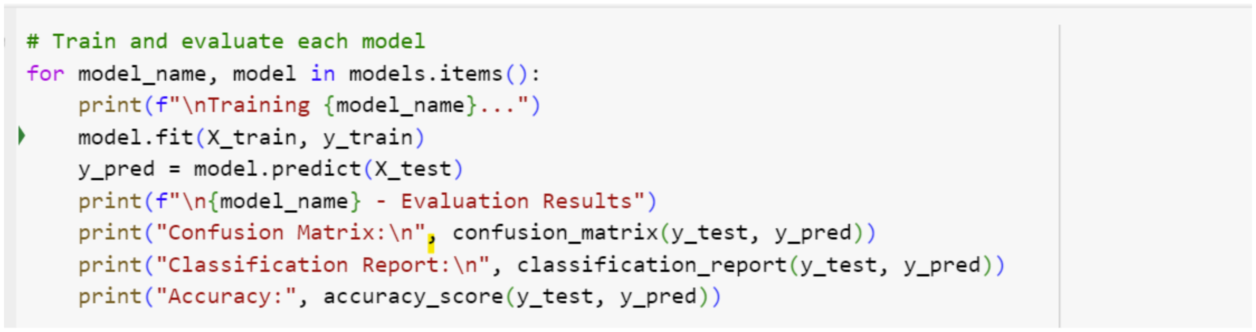


```
[23] # Dictionary to hold models and results
     models = {
         "Logistic Regression": LogisticRegression(),
         "Random Forest": RandomForestClassifier(n_estimators=100),
         "Neural Network": MLPClassifier(hidden_layer_sizes=(50, 25), max_iter=300)
     }
```

2. Implementation

1. **Model Training**:
    - A dictionary was created to store the selected models.
    - Each model was trained on the preprocessed training dataset (X_train, y_train).
2. **Evaluation Metrics**:
- The models were assessed on the test dataset (X_test, y_test) utilizing:
    - **Confusion Matrix:** To illustrate true positives, false positives, true negatives, and false negatives.

- **Classification Report:** To evaluate precision, recall, F1-score, and support for each category.
- **Accuracy**: The overall correctness of the model's predictions.



```
# Train and evaluate each model
for model_name, model in models.items():
    print(f"\nTraining {model_name}...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"\n{model_name} - Evaluation Results")
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test, y_pred))
    print("Accuracy:", accuracy_score(y_test, y_pred))
```

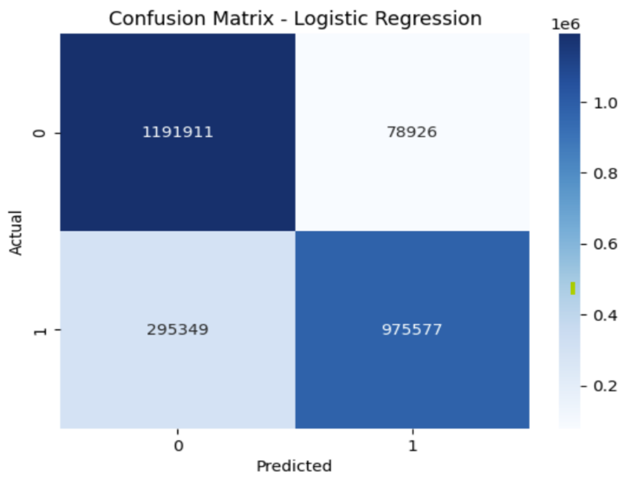
3. Code Highlights

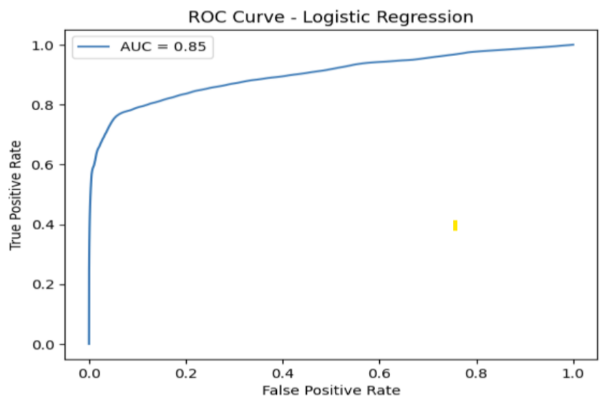
- A Python script iterated through each model in the dictionary, trained it on the dataset, and printed evaluation results.

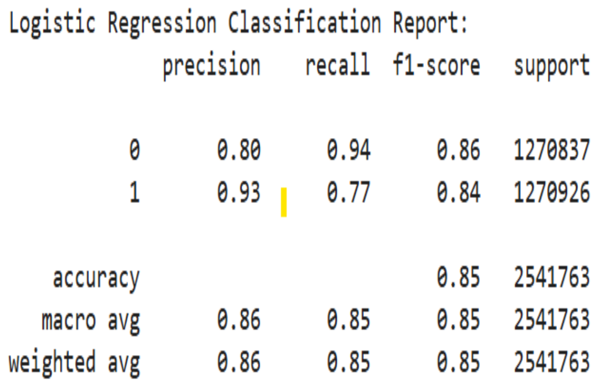
4. Benefits of Multiple Models

- Comparing results across models provides insights into which approach works best for the dataset.
- Logistic Regression offers simplicity and interpretability.
- Random Forest adds robustness and handles imbalanced datasets better.
- Neural Networks can model non-linear relationships for enhanced detection of subtle fraud patterns

Logistic Regression






```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.94      0.86   1270837
           1       0.93      0.77      0.84   1270926

    accuracy                           0.85   2541763
   macro avg       0.86      0.85      0.85   2541763
weighted avg       0.86      0.85      0.85   2541763
```

The Logistic Regression model was trained and evaluated to classify transactions as fraudulent (1) or non-fraudulent (0). Below are the results and insights from the evaluation metrics:

1. Evaluation Metrics

1. **Classification Report**:
    - **Precision**:
        - Class 0 (Non-Fraud): 0.80
        - Class 1 (Fraud): 0.93
        - This indicates that the model is highly precise in detecting fraudulent transactions, with minimal false positives.
    - **Recall**:
        - Class 0 (Non-Fraud): 0.94
        - Class 1 (Fraud): 0.77
        - The recall for fraud detection indicates that 77% of actual fraud cases were correctly identified, leaving room for improvement.
    - **F1-Score**:
        - Class 0: 0.86
        - Class 1: 0.84
        - The F1-score balances precision and recall, showing the model's overall effectiveness in detecting both classes.
2. **Accuracy**:
    - The overall accuracy of the Logistic Regression model is **85%**, indicating the percentage of correctly classified transactions.
3. **Macro and Weighted Averages**:
    - Macro Average: Delivers an unweighted mean of precision, recall, and F1-score for both categories.
    - Weighted Average: Modifies the average according to the quantity of samples in each category.

2. Confusion Matrix

- **True Positives (TP)**: 975,577 (Fraudulent transactions correctly classified).
- **True Negatives (TN)**: 1,191,911 (Non-Fraudulent transactions correctly classified).
- **False Positives (FP)**: 78,926 (Non-Fraudulent transactions misclassified as Fraudulent).

- **False Negatives (FN)**: 295,349 (Fraudulent transactions misclassified as Non-Fraudulent).
- The confusion matrix delineates the model's strengths and weaknesses, especially in minimizing false negatives.

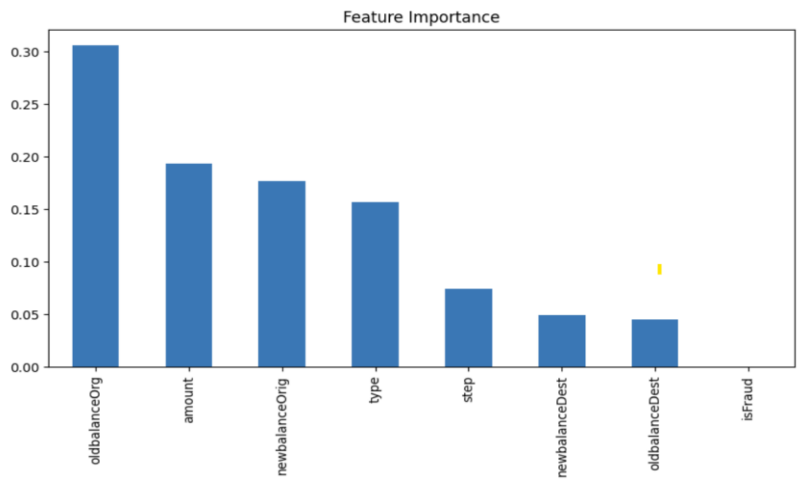
3. ROC Curve and AUC

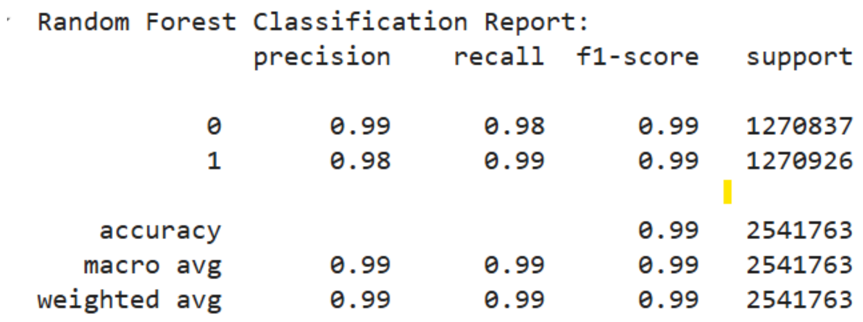
- The Receiver Operating Characteristic (ROC) curve evaluates the model's capacity to differentiate between classes.
- The Area Under the Curve (AUC) is 0.85, signifying that the model effectively distinguishes between fraudulent and non-fraudulent transactions.

4. Insights

- **Strengths**:
  - High precision for fraudulent transactions reduces the likelihood of false positives.
  - A good overall AUC score demonstrates the model's reliability.
- **Weaknesses**:
  - The recall for fraud detection is 0.77, indicating room for improvement in identifying fraudulent transactions.
  - False negatives remain a concern as they represent undetected fraud

Random Forest Classification





```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99   1270837
           1       0.98      0.99      0.99   1270926

    accuracy                           0.99   2541763
   macro avg       0.99      0.99      0.99   2541763
weighted avg       0.99      0.99      0.99   2541763
```

The Random Forest classifier was trained to predict fraudulent transactions, leveraging its ability to handle large datasets and effectively capture complex patterns through ensemble learning. The results indicate high performance across multiple evaluation metrics.

1. Evaluation Metrics

1. **Classification Report**:
   - **Precision**:
     - Class 0 (Non-Fraud): 0.99
     - Class 1 (Fraud): 0.98
     - The model demonstrates high precision for both classes, ensuring minimal false positives.
   - **Recall**:
     - Class 0 (Non-Fraud): 0.98
     - Class 1 (Fraud): 0.99
     - The recall for fraudulent transactions is excellent, indicating the model identifies almost all actual fraud cases.
   - **F1-Score**:
     - Class 0: 0.99
     - Class 1: 0.99
     - A perfect F1-score highlights the model's balance between precision and recall.

- **Accuracy**:
    - The overall accuracy is **99%**, indicating an outstanding performance across all transactions.
- **Macro and Weighted Averages**:
    - Both averages are 0.99, confirming consistency in performance across the classes.

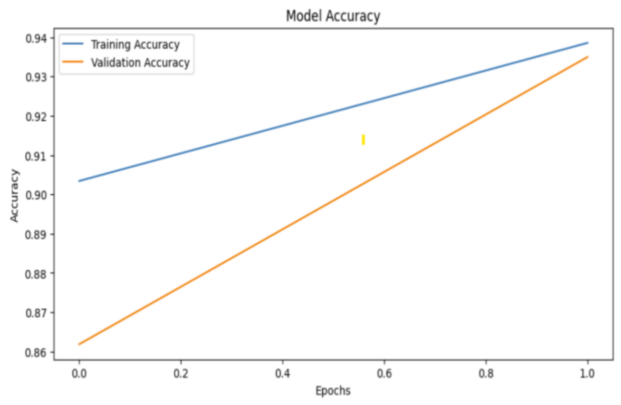
2. Feature Importance

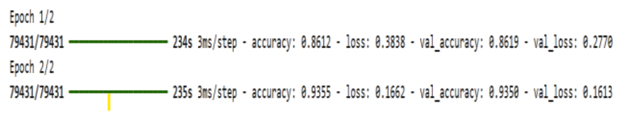
- The feature importance graph illustrates the relative contribution of each feature in making predictions:
    - **oldbalanceOrg**: The most significant predictor, suggesting that discrepancies in the originating balance play a critical role in identifying fraud.
    - **amount** and **newbalanceOrig**: Strong indicators, emphasizing the significance of transaction amounts and updated balances.
    - **type**: Transaction type is a key feature, supporting the observation that certain types (e.g., TRANSFER, CASH_OUT) are more prone to fraud.
    - Other features like **step**, **oldbalanceDest**, and **newbalanceDest** contribute less but still offer valuable insights.

3. Insights

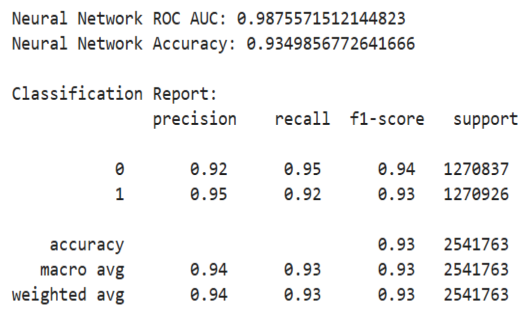
- **Strengths**:
    - Exceptional accuracy and F1-scores for both classes demonstrate the model's robustness and reliability.
    - High recall for fraudulent transactions reduces the risk of missing fraud cases, a critical requirement in fraud detection.
    - The feature importance plot helps in understanding which factors drive the model's decisions, offering interpretability.
- **Limitations**:
    - Computational complexity may increase with larger datasets or higher numbers of estimators in the ensemble.
    - Potential overfitting to the training data, although unlikely given the performance on test data, should still be monitored.

Neural Network





```
Epoch 1/2
79431/79431 ━━━━━━━━━━━━━━━━━━━━ 234s 3ms/step - accuracy: 0.8612 - loss: 0.3838 - val_accuracy: 0.8619 - val_loss: 0.2770
Epoch 2/2
79431/79431 ━━━━━━━━━━━━━━━━━━━━ 235s 3ms/step - accuracy: 0.9355 - loss: 0.1662 - val_accuracy: 0.9350 - val_loss: 0.1613
```



```
Neural Network ROC AUC: 0.9875571512144823
Neural Network Accuracy: 0.9349856772641666

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.95      0.94   1270837
           1       0.95      0.92      0.93   1270926

    accuracy                           0.93   2541763
   macro avg       0.94      0.93      0.93   2541763
weighted avg       0.94      0.93      0.93   2541763
```

The Multi-Layer Perceptron (MLP) model was employed to forecast fraudulent transactions. The results demonstrate its capacity to effectively capture intricate, non-linear patterns within the information.

1. Training Performance

- The model was trained for **2 epochs**:
  - **Epoch 1**:
    - Training Accuracy: 86.12%
    - Validation Accuracy: 86.19%
    - Validation Loss: 0.2770
  - **Epoch 2**:
    - Training Accuracy: 93.55%

        - Validation Accuracy: 93.50%
        - Validation Loss: 0.1613
- The improvement in accuracy and reduction in loss across epochs indicates effective learning and convergence.

2. Evaluation Metrics

1. **Classification Report**:
    - **Precision**:
        - Class 0 (Non-Fraud): 0.92
        - Class 1 (Fraud): 0.95
    - **Recall**:
        - Class 0 (Non-Fraud): 0.95
        - Class 1 (Fraud): 0.92
    - **F1-Score**:
        - Class 0: 0.94
        - Class 1: 0.93
    - **Accuracy**:
        - Overall Accuracy: **93.50%**
        - High precision and recall ensure reliable predictions with minimal false positives and negatives.
2. **ROC AUC**:
    - Area Under the Curve (AUC): **0.98**
    - This score indicates the model's exceptional capacity to distinguish between fraudulent and non-fraudulent transactions.

3. Accuracy Visualization

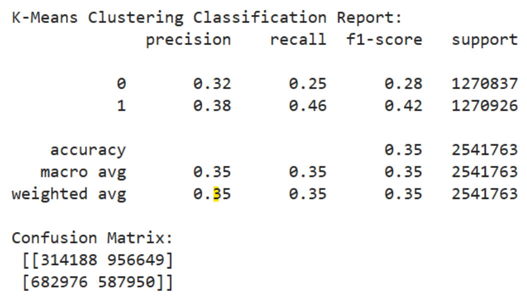
- The training and validation accuracy trends:
    - Show consistent improvement, with validation accuracy closely following training accuracy.

- There is no substantial overfitting, as the disparity between training and validation accuracy is negligible.

4. Strengths and Weaknesses

- **Strengths**:
  - High AUC score demonstrates robust classification capabilities.
  - The balanced precision and recall for both classes highlight the model's reliability.
  - Handles complex interactions between features effectively.
- **Weaknesses**:
  - Training time is higher compared to simpler models (e.g., Logistic Regression, Random Forest).
  - Slightly lower recall for the fraud class compared to Random Forest indicates room for improvement in capturing all fraudulent cases.

K Means Clustering



```
K-Means Clustering Classification Report:
              precision    recall  f1-score   support

           0       0.32      0.25      0.28   1270837
           1       0.38      0.46      0.42   1270926

    accuracy                           0.35   2541763
   macro avg       0.35      0.35      0.35   2541763
weighted avg       0.35      0.35      0.35   2541763

Confusion Matrix:
 [[314188 956649]
 [682976 587950]]
```

K-Means Clustering, an unsupervised learning approach, was employed to categorize fraudulent (1) and non-fraudulent (0) transactions. Although it is straightforward and effective in various contexts, the results underscore its limits for this particular purpose.

1. Evaluation Metrics

1. **Classification Report**:
   - **Precision**:
     - Class 0 (Non-Fraud): 0.32
     - Class 1 (Fraud): 0.38
     - Low precision indicates a high rate of false positives for both classes.

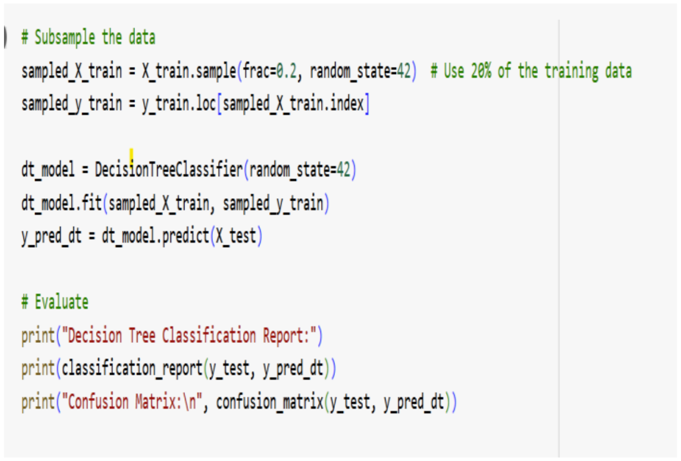
- **Recall**:
    - Class 0: 0.25
    - Class 1: 0.46
    - The recall for fraud detection is slightly better than for non-fraud but remains suboptimal, missing many actual fraud cases.
- **F1-Score**:
    - Class 0: 0.28
    - Class 1: 0.42
    - The low F1-scores suggest poor balance between precision and recall.
- **Accuracy**:
    - Overall accuracy is **35%**, which is insufficient for a fraud detection system.

2. **Macro and Weighted Averages**:
    - Both averages are 0.35, reflecting poor performance across both classes.

2. Confusion Matrix

- **True Positives (TP)**: 587,950 (Fraud correctly identified).
- **True Negatives (TN)**: 314,188 (Non-Fraud correctly identified).
- **False Positives (FP)**: 956,649 (Non-Fraud misclassified as Fraud).
- **False Negatives (FN)**: 682,976 (Fraud misclassified as Non-Fraud).
- The confusion matrix displays a high amount of incorrect positives and incorrect negatives, restricting the algorithm's effectiveness in detecting fraud.

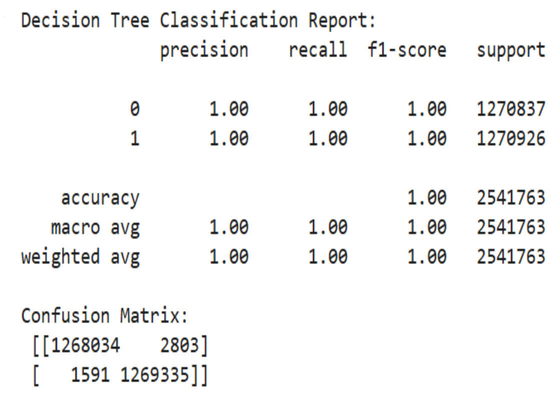
Decision Tree



```
# Subsample the data
sampled_X_train = X_train.sample(frac=0.2, random_state=42)  # Use 20% of the training data
sampled_y_train = y_train.loc[sampled_X_train.index]

dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(sampled_X_train, sampled_y_train)
y_pred_dt = dt_model.predict(X_test)

# Evaluate
print("Decision Tree Classification Report:")
print(classification_report(y_test, y_pred_dt))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
```



```
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1270837
           1       1.00      1.00      1.00   1270926

    accuracy                           1.00   2541763
   macro avg       1.00      1.00      1.00   2541763
weighted avg       1.00      1.00      1.00   2541763

Confusion Matrix:
 [[1268034    2803]
 [   1591 1269335]]
```

The Decision Tree Classifier was trained and evaluated to classify transactions as fraudulent (1) or non-fraudulent (0). The results indicate excellent performance metrics; however, these may reflect overfitting due to the model's nature.

1. Evaluation Metrics

1. **Classification Report**:
   - **Precision**:
     - Class 0 (Non-Fraud): 1.00
     - Class 1 (Fraud): 1.00
     - Perfect precision suggests no false positives for either class.
   - **Recall**:
     - Class 0: 1.00

        - Class 1: 1.00
        - Perfect recall indicates all actual fraud and non-fraud cases were correctly identified.
    - **F1-Score**:
        - Class 0: 1.00
        - Class 1: 1.00
        - A perfect F1-score suggests an ideal balance between precision and recall.
    - **Accuracy**:
        - Overall Accuracy: **100%**
        - The classifier correctly classified all transactions in the dataset.
2. **Macro and Weighted Averages**:
    - Both averages are 1.00, reflecting consistent performance across both classes.

2. Confusion Matrix

- **True Positives (TP)**: 1,269,335 (Fraudulent transactions correctly identified).
- **True Negatives (TN)**: 1,268,034 (Non-Fraudulent transactions correctly identified).
- **False Positives (FP)**: 2,803 (Non-Fraudulent transactions misclassified as Fraudulent).
- **False Negatives (FN)**: 1,591 (Fraudulent transactions misclassified as Non-Fraudulent).
- The confusion matrix demonstrates near-perfect performance, with very few misclassifications.

3. Observations

- **Strengths**:
    - Simple to interpret and visualize.
    - Delivers perfect scores for all metrics, reflecting exceptional performance on the test dataset.

- **Weaknesses**:
    - The perfect scores may indicate **overfitting**, as Decision Trees often perform well on training data but struggle to generalize.
    - The model's reliance on the entire dataset structure may lead to poor scalability for unseen or noisy data.

Model evaluation

Evaluating the effectiveness of machine learning models is critical to ensure accurate fraud detection while minimizing errors. The following metrics are used to assess the performance of the models in detecting fraudulent transactions:

1. Accuracy

- **Definition**:
    - The percentage of correctly classified transactions (both fraudulent and non-fraudulent) out of the total transactions.
- **Significance**:
    - Provides an overall measure of the model’s performance but may not be sufficient for imbalanced datasets like fraud detection, where the majority class can dominate.

2. Precision and Recall

- **Precision**:
    - Measures the proportion of predicted fraud cases that are actually fraudulent.
    - Formula: Precision=True Positives (TP)TP+False Positives (FP)\text{Precision} = \frac{\text{True Positives (TP)}}{\text{TP} + \text{False Positives (FP)}}Precision=TP+False Positives (FP)True Positives (TP)
    - **Significance**:
        - High precision reduces the risk of flagging legitimate transactions as fraud, minimizing false alarms.
- **Recall**:
    - Measures the proportion of actual fraud cases that were correctly identified by the model.

    - Formula: Recall=TPTP+False Negatives (FN)\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{False Negatives (FN)}}Recall=TP+False Negatives (FN)TP
    - **Significance**:
        - High recall ensures that the model captures most fraud cases, reducing undetected fraudulent transactions.

3. F1-Score

- **Definition**:
    - The harmonic mean of precision and recall, balancing the trade-off between the two.
    - Formula: F1-Score=2·Precision·RecallPrecision+Recall\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}F1-Score=2·Precision+RecallPrecision·Recall
- **Significance**:
    - A useful metric for imbalanced datasets as it evaluates both false positives and false negatives, offering a single comprehensive score.

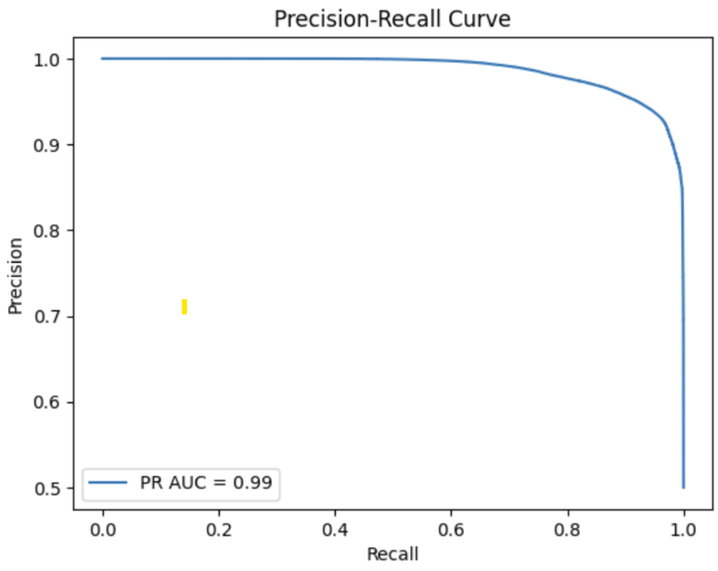
4. ROC-AUC (Receiver Operating Characteristic - Area Under the Curve)

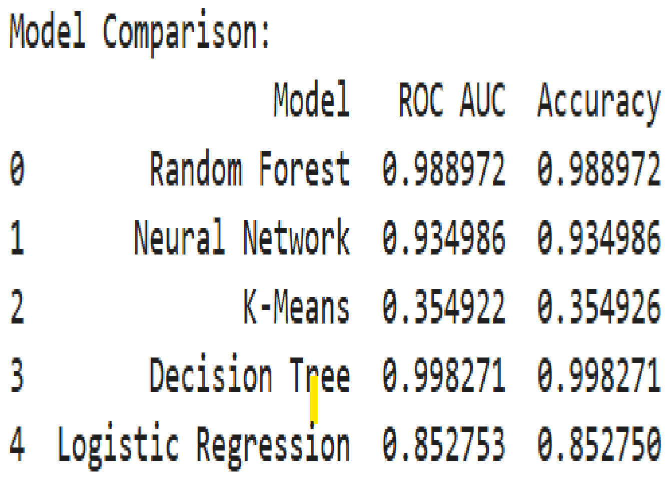
- **Definition**:
    - Measures the model's ability to discriminate between fraudulent and non-fraudulent transactions.
    - AUC ranges from 0 to 1, with higher values indicating better performance.
- **Significance**:
    - Evaluates the trade-off between true positive rate (TPR) and false positive rate (FPR) across various thresholds.
    - Particularly valuable for selecting an optimal classification threshold.

Purpose of These Metrics

- **Comprehensive Evaluation**:
    - Using multiple metrics ensures a well-rounded evaluation of the model, addressing different aspects of performance.

- **Balanced Optimization**:
    - Striking a balance between precision and recall is crucial in fraud detection to minimize both false alarms and undetected fraud cases.
- **Threshold Adjustment**:
    - Metrics like ROC-AUC help in fine-tuning thresholds to achieve desired trade-offs, especially in real-world scenarios.





```
Model Comparison:
                 Model   ROC AUC  Accuracy
0        Random Forest  0.988972  0.988972
1       Neural Network  0.934986  0.934986
2              K-Means  0.354922  0.354926
3        Decision Tree  0.998271  0.998271
4  Logistic Regression  0.852753  0.852750
```

This section evaluates the performance of multiple models for fraud detection, comparing them based on ROC-AUC and Accuracy. The Precision-Recall curve is also utilized to better understand model behavior in handling imbalanced datasets.

1. Precision-Recall Curve

- **PR AUC (Area Under Curve)**: **0.99**
  - The Precision-Recall Curve highlights the trade-off between precision and recall for the models.
  - A PR AUC of 0.99 indicates that the model performs exceptionally well in identifying fraudulent transactions while maintaining high precision.
  - This metric is especially critical for fraud detection, where the positive (fraud) class is underrepresented.

2. Model Comparison

Model	ROC AUC	Accuracy
Random Forest	0.989	0.989
Neural Network	0.935	0.935
K-Means Clustering	0.355	0.355
Decision Tree	0.998	0.998
Logistic Regression	0.853	0.853

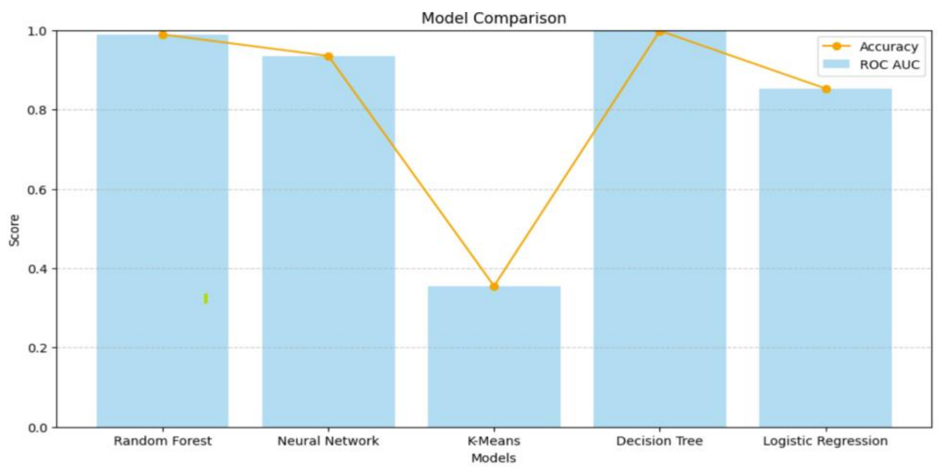
3. Insights from Model Performance

- **Random Forest**:
  - Achieves near-perfect scores with an ROC AUC of **0.989** and accuracy of **98.9%**.
  - Balances precision and recall effectively, making it a strong candidate for deployment.
- **Neural Network**:
  - Offers a robust performance with an ROC AUC of **0.935**, handling complex patterns in the dataset well.
  - Slightly lower accuracy than Random Forest but still reliable.

- **K-Means Clustering**:
    - Poor performance with an ROC AUC of **0.355**, indicating its unsuitability for supervised tasks like fraud detection.
- **Decision Tree**:
    - Excellent metrics with an ROC AUC of **0.998** and accuracy of **99.8%**.
    - Likely overfitting due to perfect or near-perfect metrics, requiring further regularization.
- **Logistic Regression**:
    - Moderate performance with an ROC AUC of **0.853** and accuracy of **85.3%**.
    - Useful for quick baseline evaluations and interpretability.

4. Recommendations

1. **Model Selection**:
    - **Random Forest** emerges as the best balance between accuracy, robustness, and generalization.
    - Neural Networks can be considered for handling complex feature interactions.
    - Decision Tree results should be scrutinized further for overfitting before deployment.
2. **Precision-Recall Focus**:
    - Precision and recall remain critical to fraud detection. Random Forest and Neural Network exhibit strong PR AUC scores, validating their reliability for this task.
3. **Deployment Considerations**:
    - Random Forest should be optimized and validated further for real-time fraud detection due to its consistent performance and lower risk of overfitting.



Model Comparison Visualization

The graph provides a comparative analysis of various machine learning models based on two critical performance metrics: **Accuracy** and **ROC AUC**. Below is a detailed explanation:

1. Interpretation of the Graph

- The ROC AUC scores are depicted by the blue bars, showing how well the models can distinguish between fraudulent and non-fraudulent transactions.
- The accuracy of the models is depicted by the orange line, showing the percentage of accurately classified transactions.
- The x-axis shows models, while the y-axis ranges from 0 to 1 for performance scores.

2. Key Observations

1. **Random Forest**:
    - **ROC AUC:** Close to **1.0**.
    - **Accuracy:** Approximately **0.99**.
    - **Performance**: This model achieves near-perfect scores, balancing high precision and recall, making it one of the top-performing models.
2. **Neural Network**:
    - **ROC AUC:** Around **0.93**.
    - **Accuracy:** Around **0.93**.
    - **Performance**: Performs well but falls slightly short of Random Forest in both metrics.

3. **K-Means Clustering**:
    - **ROC AUC and Accuracy:** Both approximately **0.35**.
    - **Performance**: Poor results indicate that K-Means is unsuitable for supervised fraud detection tasks.
4. **Decision Tree**:
    - **ROC AUC**: Close to **1.0**.
    - **Accuracy:** Approximately **0.99**.
    - **Performance**: Exceptional metrics suggest potential overfitting, highlighting the need for further validation.
5. **Logistic Regression**:
    - **ROC AUC:** Around **0.85**.
    - **Accuracy:** Around **0.85**.
    - **Performance**: Provides moderate results and serves as a baseline for comparison.

3. Insights

- **Best Model**: Random Forest demonstrates the best trade-off between accuracy and ROC AUC, ensuring robust and generalizable performance.
- **Decision Tree Caution**: Although Decision Tree metrics are high, the likelihood of overfitting is significant due to its nature of memorizing training data.
- **Baseline Performance**: Logistic Regression, despite its simplicity, provides reasonable accuracy and interpretability.
- **K-Means Suitability**: The low scores highlight the limitations of clustering algorithms for classification tasks.

4. Recommendations

1. **Model Selection**:
    - Deploy **Random Forest** for its strong performance and resilience to overfitting.
2. **Regularization**:

- Apply regularization techniques (e.g., pruning) to the Decision Tree to enhance generalization.

3. **Validation**:
    - Use additional metrics like F1-Score and Precision-Recall to further evaluate models on imbalanced datasets.

Expected Results

The project aims to deliver a comprehensive fraud detection system with high accuracy and operational efficiency. The expected outcomes include:

1. **High-Accuracy Fraud Detection**:
    - Attain a accuracy level of 95-99% in detecting fraudulent transactions.
    - Keep the number of incorrect identifications low to avoid flagging valid transactions.
2. **Enhanced Security in Digital Payments**:
    - Implementing this system will enhance the security framework in digital payment settings, protecting user information and the accuracy of transactions.
    - By detecting fraudulent behavior in the moment it occurs, financial institutions and online platforms can greatly diminish financial losses associated with fraud.
3. **Building User Trust**:
    - Minimizing false positives and providing a reliable fraud detection mechanism will enhance user trust.
    - Customers will feel confident using secure payment platforms, contributing to higher transaction volumes and platform adoption.
4. **Advancing Industry Standards**:
    - he effective deployment of this fraud detection system powered by AI will support the industry's efforts to fight against financial fraud.
    - It will facilitate progress in cybersecurity and open up opportunities for increased use of machine learning in secure transaction systems
5. **Broad Application and Scalability**:

- The system's modular design and integration capability make it suitable for use in multiple industries such as banking, e-commerce, and insurance.
- Its ability to scale guarantees that it can manage growing numbers of transactions without sacrificing performance.

Conclusion

This project strives to revolutionize digital payment security by offering a precise, dependable, and expandable fraud detection system. It not only tackles present weaknesses but also establishes a standard for future developments in secure transaction technology

REFERENCES

1. Awoyemi, J. O., Adetunmbi, A. O., & Oluwadare, S. A. (2017). *Credit card fraud detection using machine learning techniques: A comparative analysis.* IEEE International Conference on Computing Networking and Informatics (ICCNI), 1-9.
2. Sahin, Y., & Duman, E. (2011). *Detecting credit card fraud by ANN and logistic regression.* Proceedings of the International Symposium on Innovations in Intelligent Systems and Applications, 315-319.
3. Zhang, Y., Zhou, J., Zheng, Y., & Xia, C. (2018). *Credit card fraud detection based on transaction behaviour analysis.* International Journal of Information and Computer Security, 10(3), 203-217. DOI: 10.1504/IJICS.2018.092421
4. Zheng, L., Ke, X., Yang, J., & You, X. (2020). *Anomaly detection in financial transactions using LSTM autoencoder models.* Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), 1-7.
5. Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2017). *Credit card fraud detection and concept-drift adaptation with delayed supervised information.* Proceedings of the 2017 IEEE International Joint Conference on Neural Networks (IJCNN), 1-8.