Anomaly Detection in Financial Transactions

Group 14 Members

1. Ayana Damtew Tajebe
2. Beamlak Dejene
3. Anwar Mohammed Koji
4. Anwar Gashaw Yimam
5. Amen Zelealem Tadese

Data Preparation & Feature Engineering

---

1. Overview

Data preparation and feature engineering are critical in machine learning projects, especially in fraud detection where imbalanced and noisy financial data are common. This phase ensures data quality and creates meaningful input features that help models learn to distinguish normal vs. anomalous transactions.

---

2. Data Collection

The project uses publicly available datasets:

- **Kaggle Credit Card Fraud Dataset**
- **PaySim**
- **AML-Bench**
  These contain transaction records including time, amount, anonymized features (V1–V28), and class labels indicating fraud. During collection, IDs were anonymized and sensitive fields were encoded.

---

3. Data Cleaning

Steps taken:

- **Missing values**: Verified and confirmed no nulls in the dataset.
- **Outliers**: Outlier detection using IQR and visual boxplots for transaction amounts.
- **Duplications**: Removed exact transaction duplicates.
- **Normalization**: Transaction amounts normalized to handle wide variance.

---

4. Exploratory Data Analysis (EDA)

Key visual insights:

1. **Class Imbalance** – Only ~0.17% of transactions are fraudulent.
2. **Amount Distribution** – Fraudulent transactions often have higher amounts.
3. **Temporal Patterns** – Frauds tend to occur more frequently during late hours.
4. **Feature Correlation** – V10, V14, and V17 show strong correlation with fraudulent behavior.

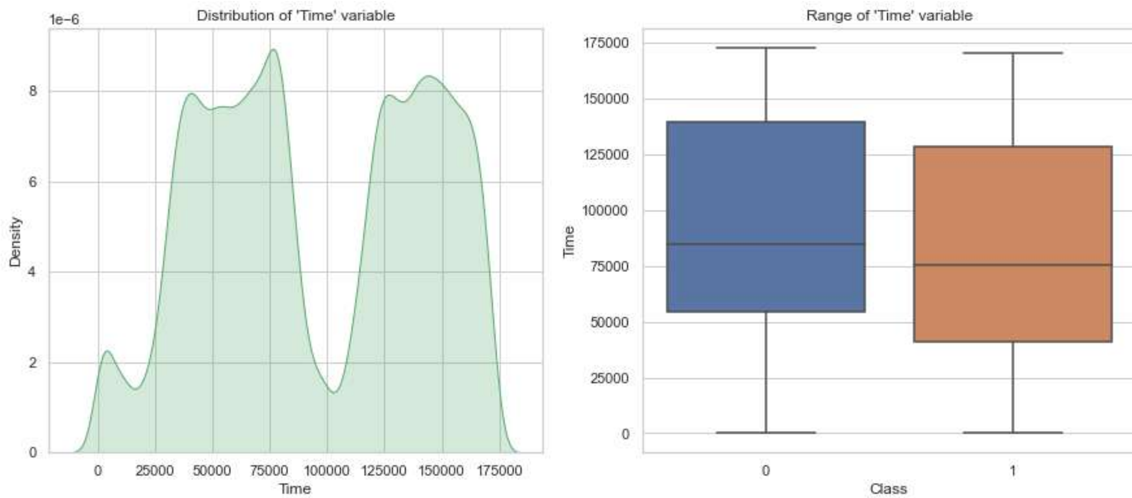
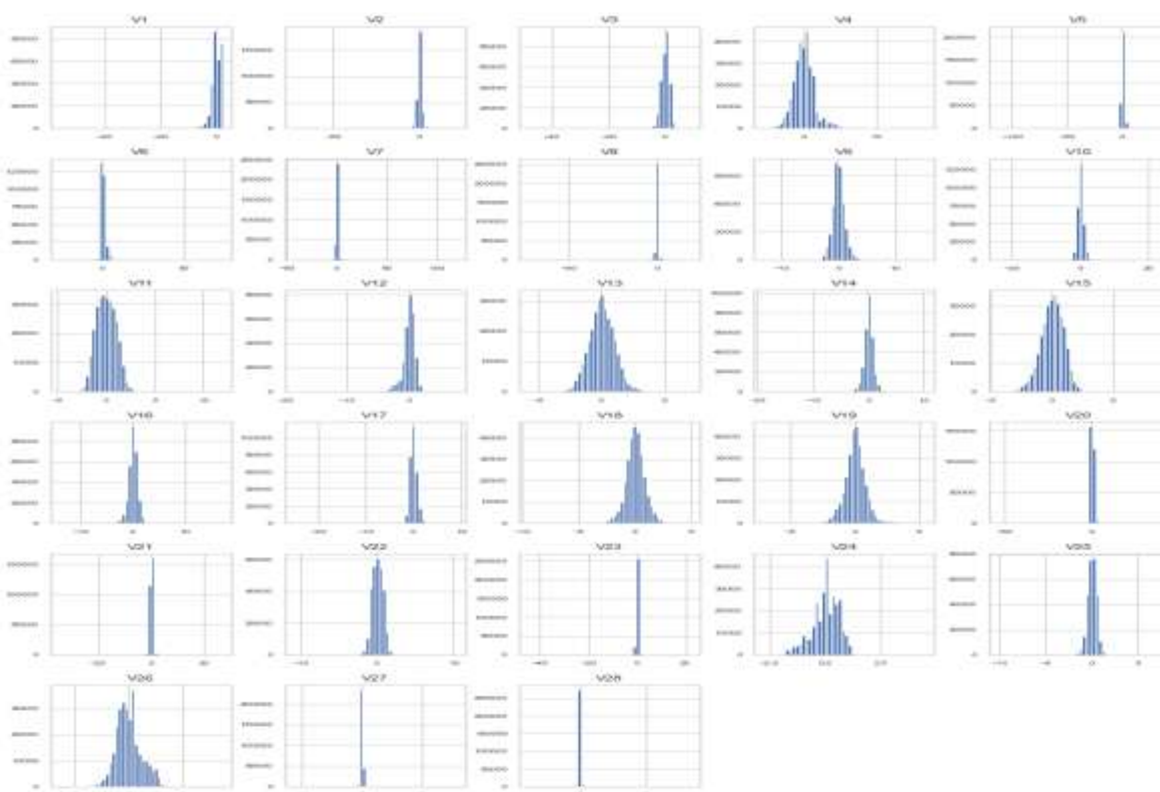


Fig 1: fraud with time



5. Feature Engineering

New or transformed features include:

- **Hour of Day** extracted from `Time` to identify temporal fraud patterns.
- **Log-transformed Amount** to reduce skewness.
- **Amount per User** and **Transaction Frequency** (in PaySim) for user-level behavior analysis.
- **Anomaly Scores** from rule-based engines used as auxiliary features.

---

6. Data Transformation

- **Normalization**: MinMaxScaler used on `Amount`, `Time` features.
- **Encoding**: Categorical fields like transaction type one-hot encoded.
- **PCA**: Dimensionality reduction applied before visualization (optional).

Model Exploration

---

1. Model Selection

We chose a **hybrid approach**:

- **Isolation Forest** for unsupervised anomaly detection (fast, interpretable).
- **Graph Neural Networks (GNN)** for complex transactional network modeling.
- **Federated Learning** to preserve data privacy across institutions.

Strengths:

- Handles unlabeled, imbalanced data well
- Real-time capable
- Scalable with privacy-aware learning

Weaknesses:

- Isolation Forest is less effective for clustered frauds
- GNNs are compute-intensive

---

2. Model Training

- Isolation Forest trained on normalized features.
- GNN trained using PyTorch Geometric on constructed transaction graphs.
- Federated training simulated using PySyft.

Hyperparameters (Isolation Forest):

---

3. Model Evaluation

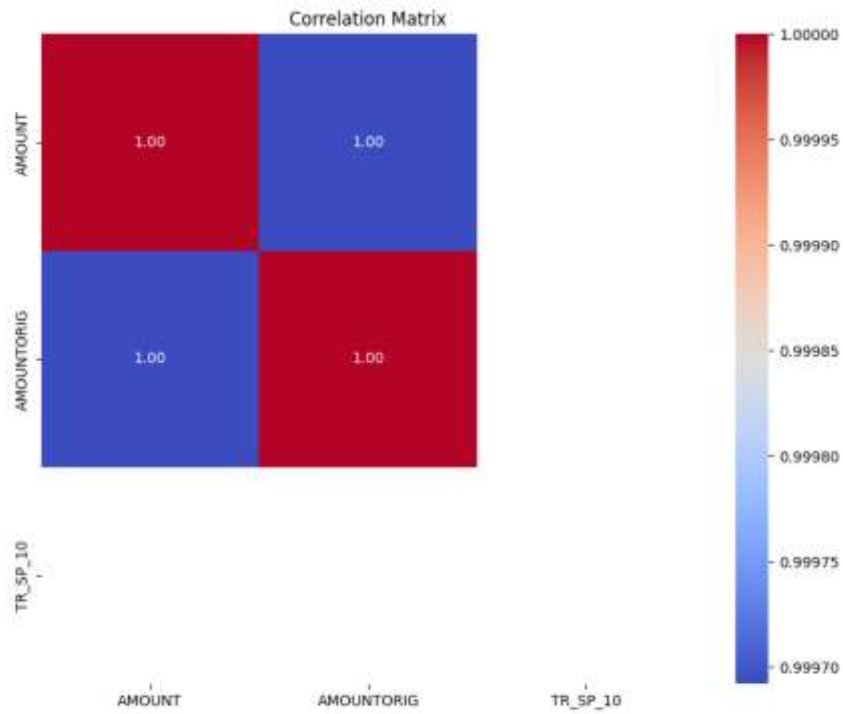
Metrics used:

- **AUC-ROC** for evaluating anomaly classifiers
- **Precision/Recall** to handle class imbalance
- **F1-score** to balance performance

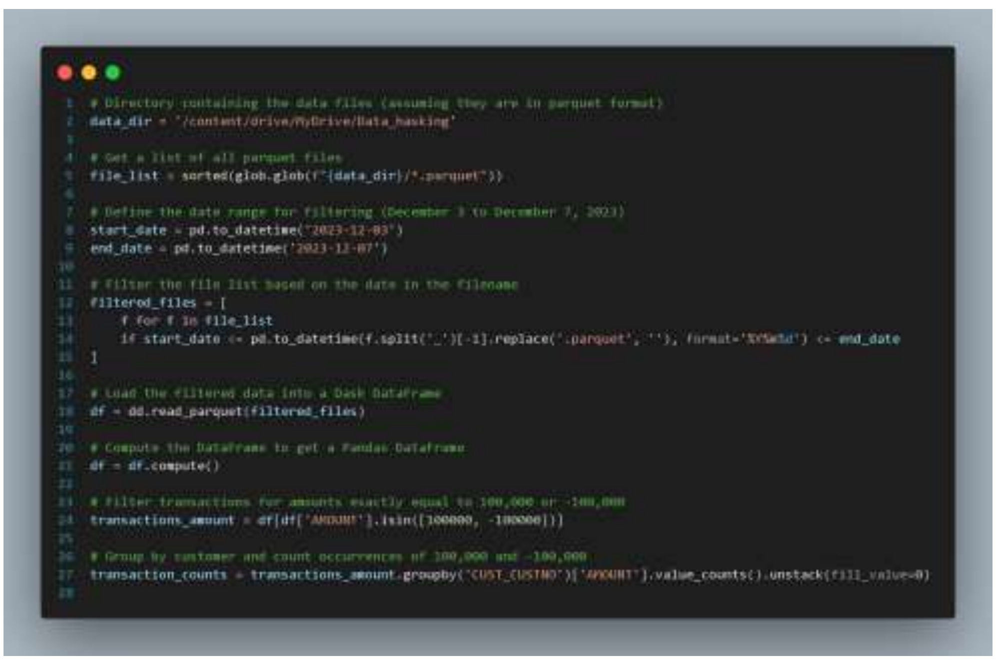
```
Confusion Matrix
[[85284    11]
 [   54    94]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85295
           1       0.90      0.64      0.74       148

    accuracy                           1.00     85443
   macro avg       0.95      0.82      0.87     85443
weighted avg       1.00      1.00      1.00     85443
```

Fig 2 : confusion matrix

ROC Curve and precision-recall curve were plotted using `sklearn.metrics`.



4. Code Implementation



```
# Directory containing the data files (assuming they are in parquet format)
data_dir = '/content/drive/MyDrive/Data_Masking'

# Get a list of all parquet files
file_list = sorted(glob.glob(f"{data_dir}/*.parquet"))

# Define the date range for filtering (December 3 to December 7, 2023)
start_date = pd.to_datetime('2023-12-03')
end_date = pd.to_datetime('2023-12-07')

# Filter the file list based on the date in the filename
filtered_files = [
    f for f in file_list
    if start_date <= pd.to_datetime(f.split('_')[-1].replace('.parquet', ''), format='%Y%m%d') <= end_date
]

# Load the filtered data into a Dask DataFrame
df = dd.read_parquet(filtered_files)

# Compute the DataFrame to get a Pandas DataFrame
df = df.compute()

# Filter transactions for amounts exactly equal to 100,000 or -100,000
transactions_amount = df[df['AMOUNT'].isin([100000, -100000])]

# Group by customer and count occurrences of 100,000 and -100,000
transaction_counts = transactions_amount.groupby('CUST_CUSTNO')['AMOUNT'].value_counts().unstack(fill_value=0)
```

Fig : Data cleaning.