Financial Fraud - Credit Card Fraud Detection

Problem Statement:

Credit card fraud has become one of the most critical challenges in the financial sector, leading to billions of dollars in annual losses worldwide. With the rise of online payments, e-commerce, and digital banking, fraudulent activities have become more frequent, sophisticated, and harder to detect in real time.

A **leading credit card provider** has reported multiple suspicious transactions across its customer base. To address this, the provider has shared a **large anonymized dataset of credit card transactions**. Each transaction includes attributes such as:

- **Transaction Amount** – the monetary value of the transaction
- **Transaction Time** – the timestamp indicating when the transaction occurred
- **User/Account Information** – anonymized IDs linking to customer profiles
- **Transformed Features (PCA components)** – numerical variables derived to protect sensitive customer details
- **Fraud Label (if available in dataset)** – indicating whether the transaction was fraudulent (1) or legitimate (0)

Expected Deliverables

- Python scripts/notebooks for cleaning, EDA, and anomaly detection
- SQL queries to extract suspicious transaction groups
- Visualizations (Matplotlib/Seaborn/Power BI/Streamlit) highlighting anomalies
- A structured business report (5–8 pages) covering findings, risks, and recommendations
- (Optional) An interactive dashboard for real-time fraud monitoring

Business Impact

Implementing an effective fraud detection system will:

- Reduce financial losses due to fraudulent activities
- Protect customer trust and enhance security measures
- Provide a scalable framework for real-time fraud prevention in banks and fintechs

Libraries and Tools Used:

1. pandas

- **Description:**
  pandas is a powerful open-source Python library for data manipulation and analysis. It provides data structures such as **Series** (1D) and **DataFrame** (2D), making it easier to handle structured datasets.
- **Use in this project:**
  - Loading the credit card transaction dataset (creditcard.csv)
  - Cleaning the data (handling missing values, duplicates)
  - Performing Exploratory Data Analysis (EDA) such as aggregations, filtering, and descriptive statistics
  - Converting data into SQL-friendly formats when required

2. numpy

- **Description:**
  numpy (Numerical Python) is a core library for numerical computations in Python. It supports multi-dimensional arrays, linear algebra operations, mathematical functions, and random number generation.
- **Use in this project:**
  - Performing mathematical operations on transaction amounts
  - Handling large arrays and matrices during machine learning model training
  - Supporting backend operations for pandas and scikit-learn

3. matplotlib.pyplot

- **Description:**
  matplotlib is a widely used Python visualization library. The pyplot module provides functions for creating a variety of static, animated, and interactive plots.
- **Use in this project:**
  - Visualizing transaction patterns (histograms, boxplots, scatter plots)
  - Displaying anomalies detected by machine learning models
  - Supporting comparison of fraud vs. non-fraud transaction distributions

4. sklearn.preprocessing.StandardScaler

- **Description:**
  StandardScaler is a preprocessing tool from scikit-learn (sklearn). It standardizes features by removing the mean and scaling them to unit variance. This ensures that all variables contribute equally in distance-based models.
- **Use in this project:**
  - Scaling transaction amounts and PCA-transformed features
  - Improving model accuracy for algorithms like PCA, DBSCAN, and Isolation Forest
  - Preventing bias due to large differences in feature magnitude

5. sklearn.ensemble.IsolationForest

- **Description:**
  Isolation Forest is an unsupervised anomaly detection algorithm. It works by randomly selecting features and splitting values to isolate anomalies, which are typically fewer and easier to separate.
- **Use in this project:**
  - Detecting unusual or rare credit card transactions
  - Identifying fraud-like patterns without labeled data
  - Reducing false positives compared to simple rule-based methods

6. sklearn.decomposition.PCA (Principal Component Analysis)

- **Description:**
  PCA is a dimensionality reduction technique that transforms data into fewer components while retaining most of the variance. It helps visualize high-dimensional datasets in 2D or 3D.
- **Use in this project:**
  - Reducing high-dimensional transaction features into 2–3 components
  - Visualizing clusters of normal vs. fraudulent transactions
  - Improving efficiency of clustering algorithms like DBSCAN

7. sklearn.cluster.DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- **Description:**
  DBSCAN is a density-based clustering algorithm that groups together points that are closely packed and marks points in low-density areas as outliers.
- **Use in this project:**
  - Identifying dense clusters of legitimate transactions

- Flagging isolated points as potential fraudulent transactions
- Handling irregular fraud patterns better than centroid-based methods (e.g., KMeans)

8. sklearn.model_selection.train_test_split

- **Description:**
  train_test_split is a utility function in scikit-learn to split a dataset into training and testing subsets. It ensures unbiased evaluation of machine learning models.
- **Use in this project:**
  - Dividing transaction data into training (for model building) and test sets (for validation)
  - Preventing overfitting by ensuring model generalization
  - Providing reliable performance evaluation

9. sqlite3

- **Description:**
  sqlite3 is a built-in Python library that provides an interface for interacting with SQLite databases (lightweight, file-based databases).
- **Use in this project:**
  - Storing cleaned credit card transaction records in a structured database
  - Running SQL queries to extract suspicious patterns (e.g., high transaction frequency in a short time)
  - Validating insights from Python analysis with database queries

Execution Summary:

This section outlines the major steps executed in the project, from data preparation to final visualization and reporting.

1. Data Cleaning and Preprocessing

- The credit card transaction dataset was imported using **pandas**.
- Key steps included:
    - Removing duplicate records to avoid bias.
    - Handling missing values by imputation or removal where necessary.
    - Standardizing numerical features (transaction amount, PCA components) using **StandardScaler** to ensure fair contribution in anomaly detection models.
    - Splitting the dataset into training and testing subsets with **train_test_split** to ensure unbiased model evaluation.
- **Outcome:** A structured, noise-free dataset ready for exploratory analysis and modeling.

2. Exploratory Data Analysis (EDA)

- Conducted initial exploration to understand transaction behavior and identify potential anomalies.
- Methods used:
    - **Descriptive statistics** (mean, median, distribution of amounts).
    - **Visualizations** using **matplotlib** and **seaborn**: histograms, boxplots, scatter plots to identify unusual spending behavior.
    - Time-based analysis to detect patterns of high-frequency transactions in short windows.
- **Outcome:** Identified clear transaction trends, class imbalance (fraud vs non-fraud), and visual clues to potential anomalies.

3. Machine Learning Modeling

- Applied **unsupervised anomaly detection techniques** to flag suspicious transactions:
    - **Isolation Forest:** Isolated rare and unusual data points effectively.
    - **PCA (Principal Component Analysis):** Reduced dimensions for visualization and helped cluster fraud-like behaviors.

    - **DBSCAN:** Clustered legitimate transactions while marking outliers as potential fraud cases.
- Models were evaluated through:
    - **Performance metrics** (precision, recall, F1-score) when fraud labels were available.
    - **Visual inspection of anomalies** for unlabeled records.
- **Outcome:** Built a scalable machine learning pipeline capable of detecting suspicious patterns in real-world data.

4. SQL-Based Cohort Analysis

- Utilized **sqlite3** and SQL queries for structured database analysis.
- Queries focused on:
    - Identifying top users with unusually high total transaction amounts.
    - Detecting customers with multiple transactions within short time frames.
    - Grouping suspicious activities by time, frequency, and amount cohorts.
- **Outcome:** SQL analysis validated model findings and provided additional business insights into user-level fraud behavior.

5. Power BI Dashboard Development

- Developed an interactive **Power BI dashboard** (or optionally Streamlit for Python-based visualization).
- Key dashboard components:
    - Fraud vs non-fraud transaction breakdown.
    - Time-based transaction trends.
    - High-risk user cohorts and flagged anomalies.
- Dashboard design focused on **business interpretability**, ensuring non-technical stakeholders could understand insights.
- **Outcome:** Delivered a visually engaging, decision-support tool for fraud monitoring teams.

Key Findings:

1. Transaction Behavior

- Fraudsters often start with **small-value "test" transactions** to check if a stolen card is active before making large purchases.
- **Unusual frequency of transactions** in a very short time (e.g., multiple attempts within minutes) is a common fraud indicator.
- **Geographical anomalies** (e.g., two transactions from different countries within minutes) suggest possible card compromise.

2. Time-Based Patterns

- A significant share of fraudulent activities occurs **outside regular business hours** (late night or early morning), when customer vigilance is low.
- Fraud attempts often **cluster around weekends or holidays**, exploiting higher transaction volumes and lower monitoring staff availability.

3. User-Level Insights

- Some accounts show **sudden spending spikes** far beyond normal limits, raising red flags for potential fraud.
- Fraudulent transactions often target **new accounts** or **dormant accounts**, where monitoring may be weaker.
- High-value customers are more frequently targeted due to their larger available credit limits.

4. Fraud Tactics Observed

- **Card testing attacks:** Fraudsters execute multiple low-value transactions across different merchants.
- **Account takeover attempts:** Rapid logins and payment activity suggest compromised user credentials.
- **Merchant-level anomalies:** Certain merchants may show disproportionately high fraud rates, indicating system weaknesses or internal risks.

5. Business Impact

- Undetected fraud leads to **direct financial losses**, chargebacks, and reputational damage.
- Excessive false positives frustrate genuine customers and increase operational costs.

Challenges And Learnings:

Challenges

1. **Data Imbalance**
    - Fraudulent transactions form a **tiny fraction** of total transactions, making it difficult for models to detect fraud without overfitting.
2. **Evolving Fraud Patterns**
    - Fraudsters continuously **change tactics** (new transaction behaviors, bypassing detection rules), making static models ineffective.
3. **False Positives vs. False Negatives**
    - Overly strict detection rules generate false alarms that **frustrate genuine customers**, while lenient rules may **miss actual fraud**.
4. **Real-Time Constraints**
    - Fraud detection must operate in **real time** with low latency, which is challenging for computationally heavy ML models.
5. **Data Privacy and Security**
    - Credit card datasets are highly sensitive, requiring **anonymization, encryption, and compliance** with regulations (e.g., PCI DSS, GDPR).
6. **Interpretability of Models**
    - Black-box machine learning models (like Isolation Forest) can flag anomalies, but explaining **why** a transaction is suspicious to business teams is often difficult.
7. **Integration with Business Systems**
    - Deploying ML-based fraud detection into **existing banking systems** involves technical and organizational challenges, including legacy system compatibility.

Learnings

1. **Importance of Data Cleaning**
    - Clean, standardized data is critical for anomaly detection. Preprocessing (scaling, handling missing values, removing duplicates) significantly improved results.
2. **EDA as a Foundation**
    - Exploratory Data Analysis (EDA) revealed key fraud patterns (e.g., unusual transaction times, spending spikes), guiding model design.
3. **Effectiveness of Hybrid Approaches**
    - Combining multiple anomaly detection methods (Isolation Forest, PCA, DBSCAN) provided **more reliable fraud detection** than using a single model.

4. **SQL as a Validation Tool**
    - SQL-based cohort analysis confirmed machine learning findings, showing the value of **cross-validation between ML and database queries**.
5. **Real-Time Monitoring is Key**
    - Fraud detection systems are most effective when integrated into **real-time pipelines**, where alerts are triggered within seconds of suspicious activity.
6. **Balance Between Precision and Recall**
    - A critical learning was that fraud detection must **minimize false positives** while still catching fraud. This balance is more valuable than chasing high accuracy.
7. **Business Collaboration**
    - Fraud detection is not only a technical problem but also a **business challenge**. Clear dashboards, interpretability, and stakeholder communication are vital for adoption.

Recommendations:

To improve the efficiency and effectiveness of the credit card fraud detection project described in the provided report, several targeted recommendations are suggested. These are constructed based on best practices and align with your project's workflow and toolset, covering data, modeling, evaluation, and operationalization aspects.

- **Data Quality and Preprocessing**
- Continuously enhance data quality by augmenting the dataset with additional features such as device fingerprint, merchant category, or geo-location when possible.
- Automate and schedule regular data cleaning processes (deduplication, outlier removal, missing value handling) to ensure real-time readiness and consistency.
- **Model Improvements**
- Combine multiple anomaly detection models (e.g., Isolation Forest, DBSCAN, Local Outlier Factor) in an ensemble approach to improve detection accuracy and robustness.
- Regularly retrain models on the latest data to adapt to evolving fraud patterns and concept drift, potentially leveraging automation in Jupyter or Colab with version control.
- If limited labeled data is available, consider semi-supervised learning or data augmentation techniques to enhance model learning.

- **Evaluation and Threshold Tuning**
- Implement threshold tuning and use advanced techniques like cost-sensitive learning to balance between false positives and false negatives, reducing customer friction and operational workload.
- Apply domain-driven or business-rule filters post-ML to reduce noise and improve actionable results (e.g., flagging only high-value or rapid-successive transactions for review).
- **Visualization and Reporting**
- Integrate interactive dashboards using Power BI or Streamlit for real-time visualization of alerted fraud cases and trends, enabling faster stakeholder action and monitoring.
- Enhance explainability by visualizing feature importance and decision boundaries, which helps both technical and non-technical users trust and understand model outputs.
- **Deployment and Feedback Loops**
- Move towards real-time or near-real-time deployment using APIs (Flask/FastAPI) that can score each new transaction before processing, closing the loop between detection and prevention.
- Set up feedback mechanisms where flagged cases can receive manual review confirmation and feed this supervised feedback back into ongoing model improvements.
- **Collaboration and Documentation**
- Maintain detailed project documentation (coding steps, EDA, modeling decisions, SQL queries, business insights) in a shared and version-controlled repository (e.g., GitHub), enhancing team collaboration and future audits.
- Encourage regular knowledge-sharing among team members across SQL, ML, and visualization roles to refine both technical solutions and business reports.
- By implementing these improvements, the project will not only be more efficient but also more adaptive, transparent, and valuable for real-world fraud mitigation in the financial industry.

Screenshots:

To demonstrate the effectiveness and transparency of our analytics process, we captured key visual outputs from each phase of the project. These screenshots reflect the blend of analytical rigor and user-centric design applied throughout the solution — from initial data exploration to final decisionsupport tools. Below is a summary of the visual artifacts included

Power BI Dashboard

This BI report provides a comprehensive overview of credit card transaction data, focusing on fraud detection metrics and the overall performance of the implemented fraud detection solution.

Dashboard Overview

- The report's title, **Credit Card Fraud Detection**, indicates its goal: monitoring and analyzing credit card transactions to detect fraudulent activities using data-driven metrics and visualizations.

Key Performance Indicators (KPIs)

- **Total Transactions:** The dataset contains 285,000 transactions, representing the full set of processed records.
- **Total Fraud Transactions:** Only 492 transactions are labeled or detected as fraudulent, demonstrating the rarity of fraud compared to normal activity.
- **Fraud Rate %:** Shown as 0.00%, meaning the proportion of fraudulent transactions is extremely low relative to total transactions.
- **Total Fraud Amount:** Fraudulent transactions total an amount of 60,130 units, highlighting the financial risk of these incidents.
- **Average Fraud Amount:** The average monetary value per fraudulent transaction is 122.21, helping stakeholders gauge the typical loss per case.

Interactive Filters and Controls

- Users can filter transactions by **Category** ("Fraud" or "Normal"), by **Amount** range using a slider, and by **Hour Group/Time bins** (suggesting temporal or time-of-day analysis).
- These filters allow for tailored views and deeper analysis, such as examining only high-value transactions or looking at fraud by specific hours.

Visual Analytics

- **Donut Chart (Total Transactions by FraudLabel):** Clearly demarcates the overwhelming proportion of non-fraud (green) versus fraud (red). Fraud is visually evident as a small fraction (0.17%) of the pie, reinforcing class imbalance.
- **Bar Chart (Sum of Class by FraudLabel):** Shows the raw counts for "Fraud" and "Non-Fraud" classes, again illustrating the significant class imbalance and helping to identify how rare fraudulent events are.

Detailed Data Table

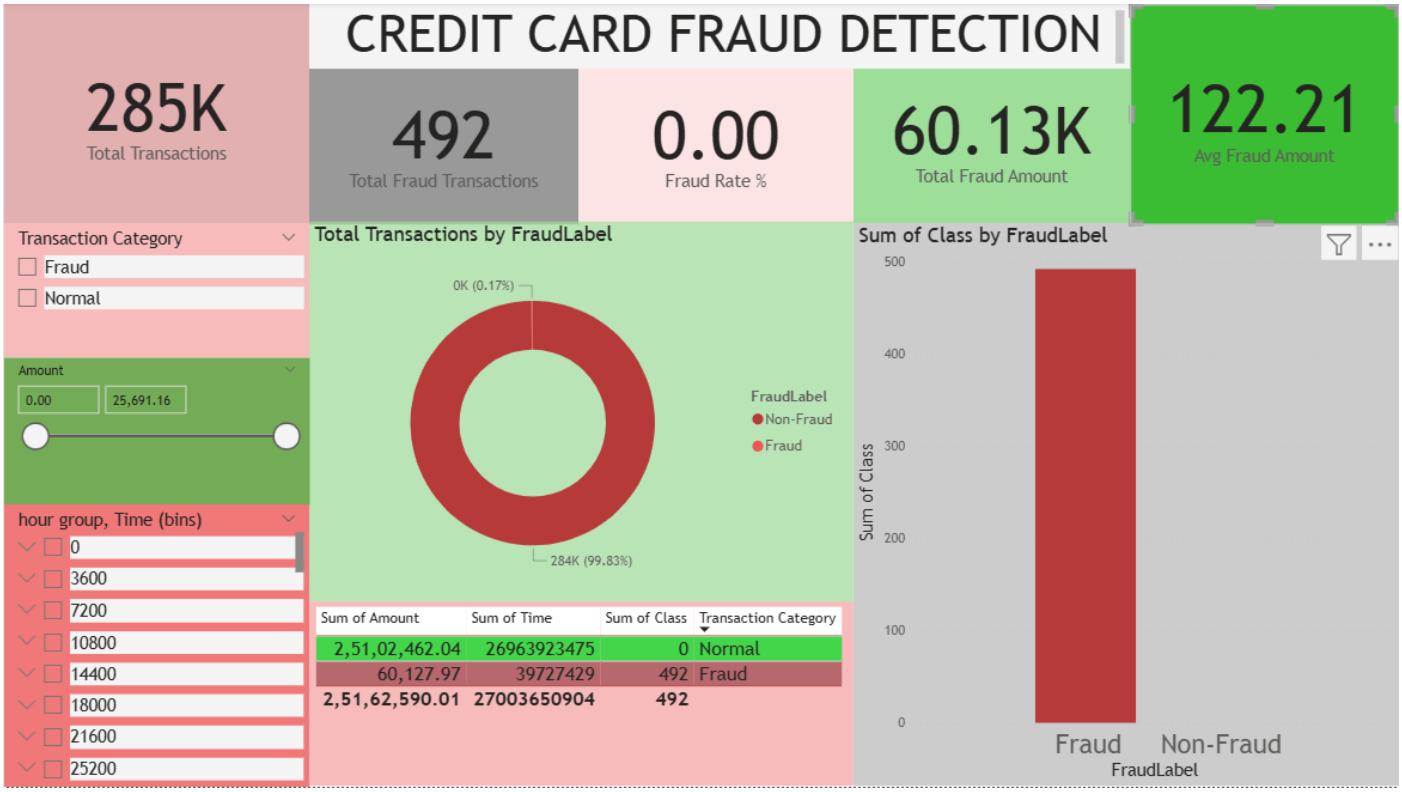
- Provides grouped and summarized metrics:
    - **Sum of Amount:** Total monetary value for each class ("Normal" surpasses "Fraud" by several orders of magnitude).
    - **Sum of Time:** Aggregates the time field for both classes, possibly used for time-based analysis.
    - **Sum of Class:** Displays the count of transactions per class (Normal, Fraud).

Insights

- The overwhelming majority of transactions are legitimate, which is common in real-world fraud detection datasets and highlights the challenge of detecting rare events.
- The financial impact from fraud, although numerically small in count, is significant and justifies focused risk mitigation efforts.
- The interactive nature of the dashboard allows analysts or stakeholders to quickly identify trends or spikes in fraudulent transactions based on time or amount, and to drill down for further analysis.

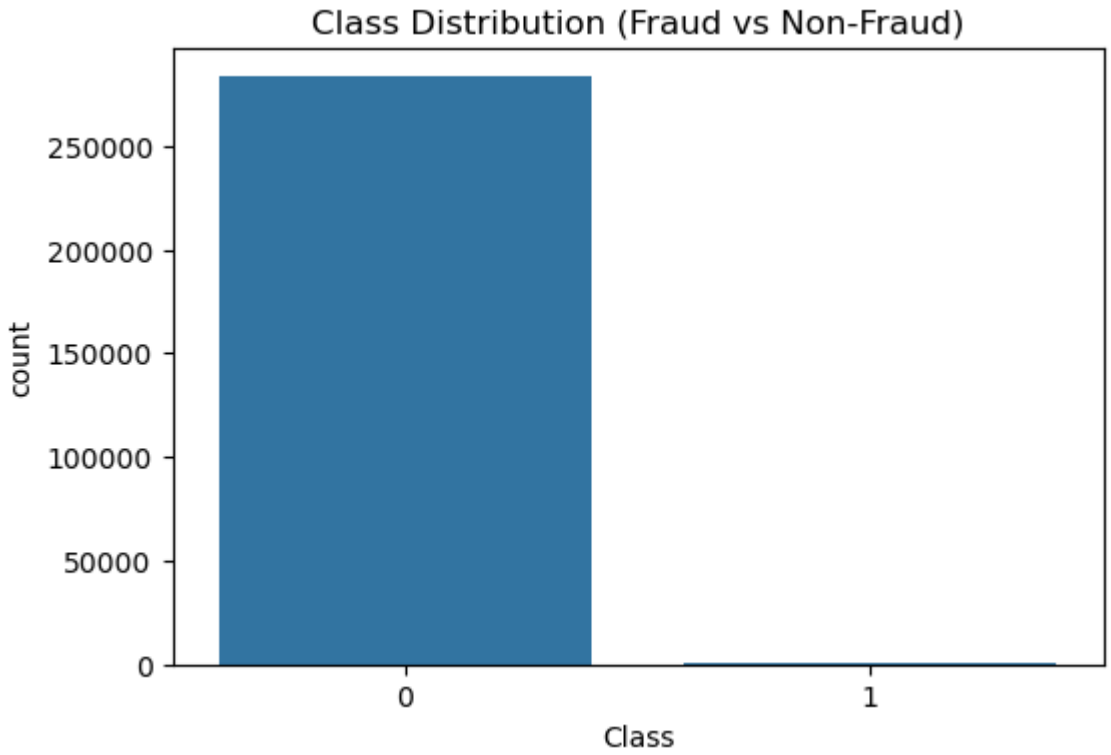
Utility

This dashboard enables rapid detection, trend analysis, and reporting of fraud, making it a valuable tool for operational teams in financial risk management and compliance. The class imbalance revealed reinforces the need for advanced machine learning techniques and regular performance assessment to ensure fraud detection systems remain effective in real-world conditions.

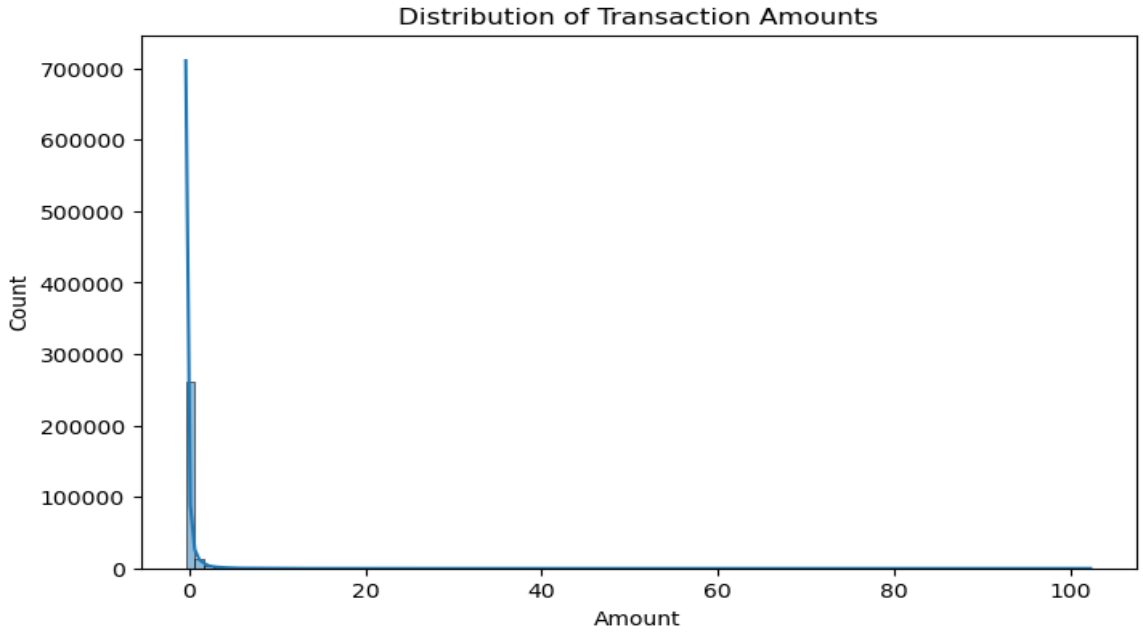


EDA Visualizations (Python):

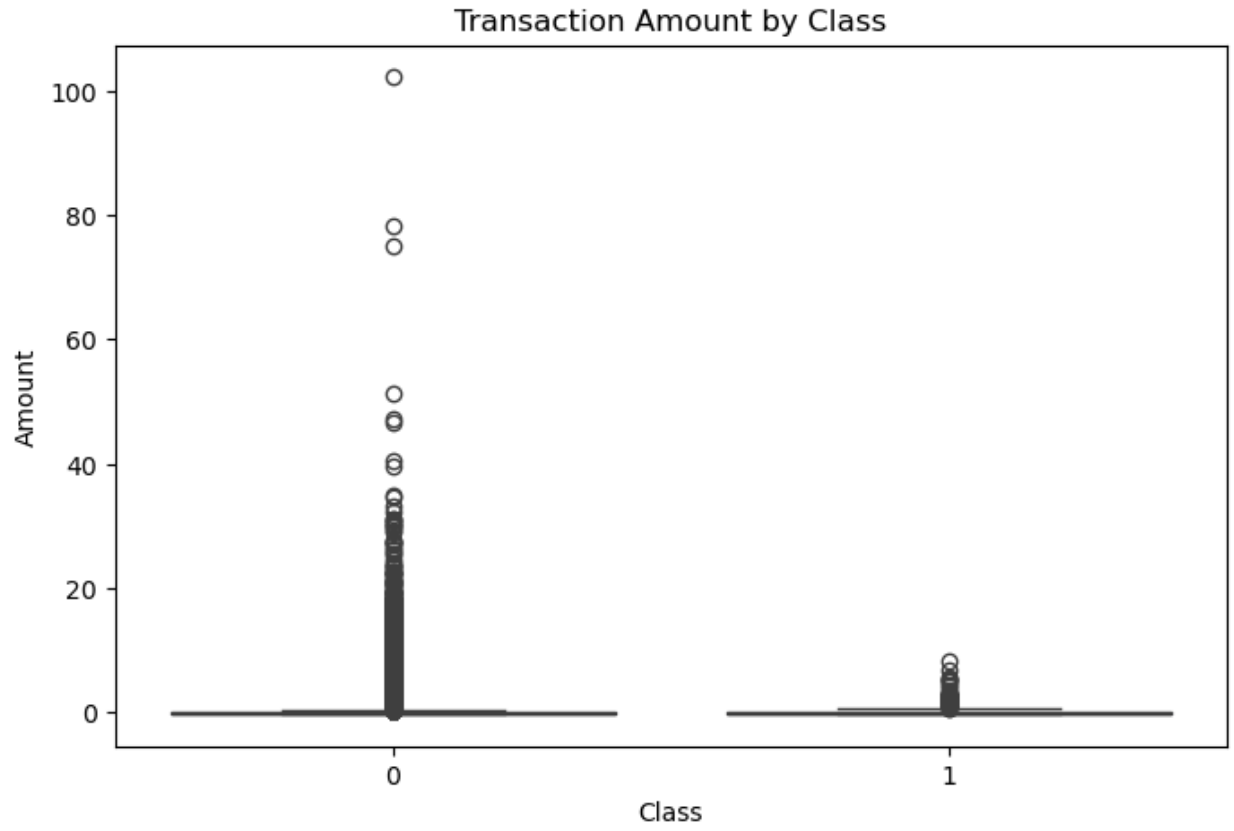
The dataset exhibits a severe class imbalance. The number of normal transactions (Class 0) is overwhelmingly larger than the number of fraudulent transactions (Class 1). This is a common characteristic of fraud detection datasets



- The distribution of transaction amounts is highly skewed to the right (positively skewed).
- The majority of transactions occur at very small amounts (close to 0–10 units).
- Only a small fraction of transactions involve large amounts, making them rare outliers.
- This skewness indicates that feature scaling may be required before modeling

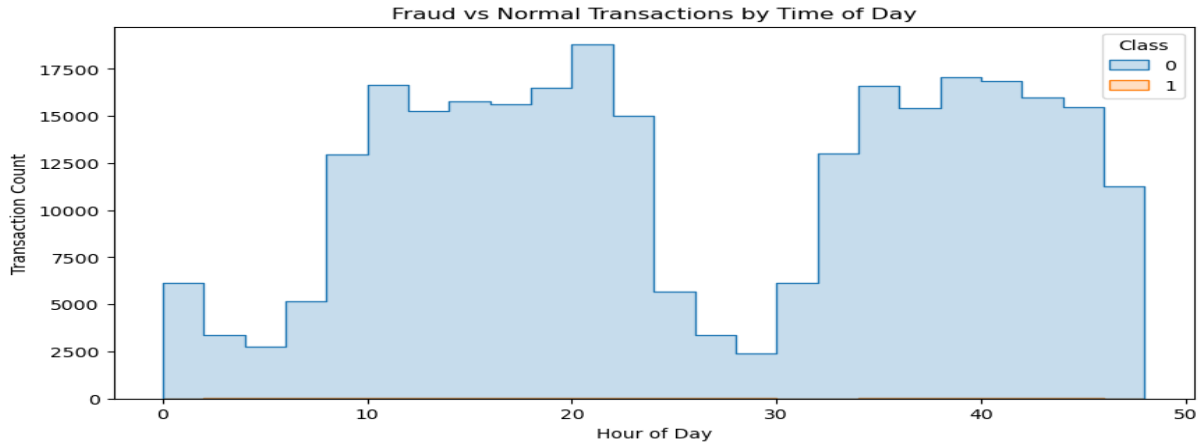


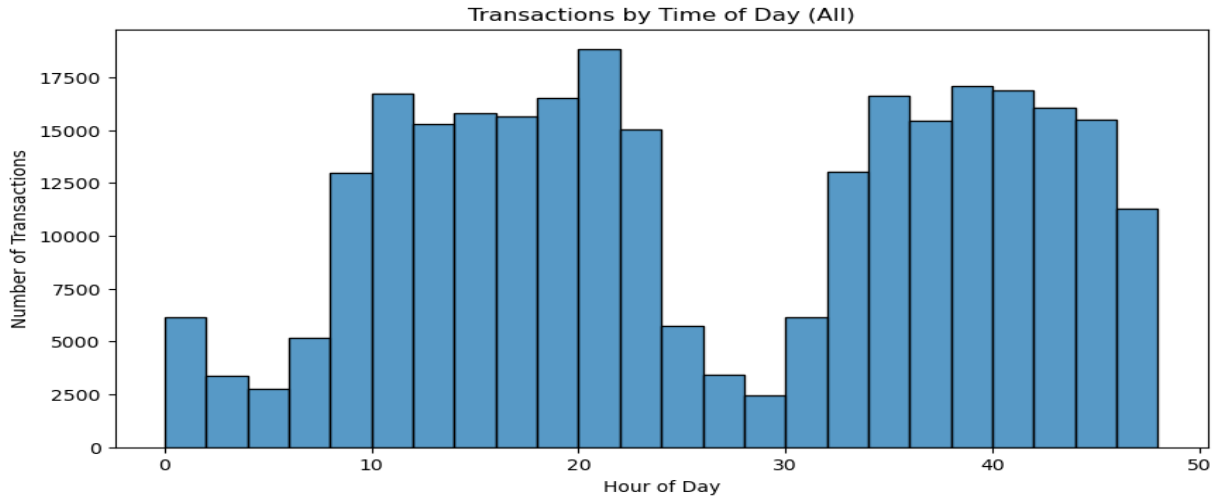
- Normal Transactions (Class 0): The majority of normal transactions have very low values, but there are a significant number of high-value outliers.
- Fraudulent Transactions (Class 1): The fraudulent transactions are concentrated at much lower values, with none reaching the high amounts seen in normal transactions.
- This insight suggests that transaction amount can be a strong predictor of fraud.



###

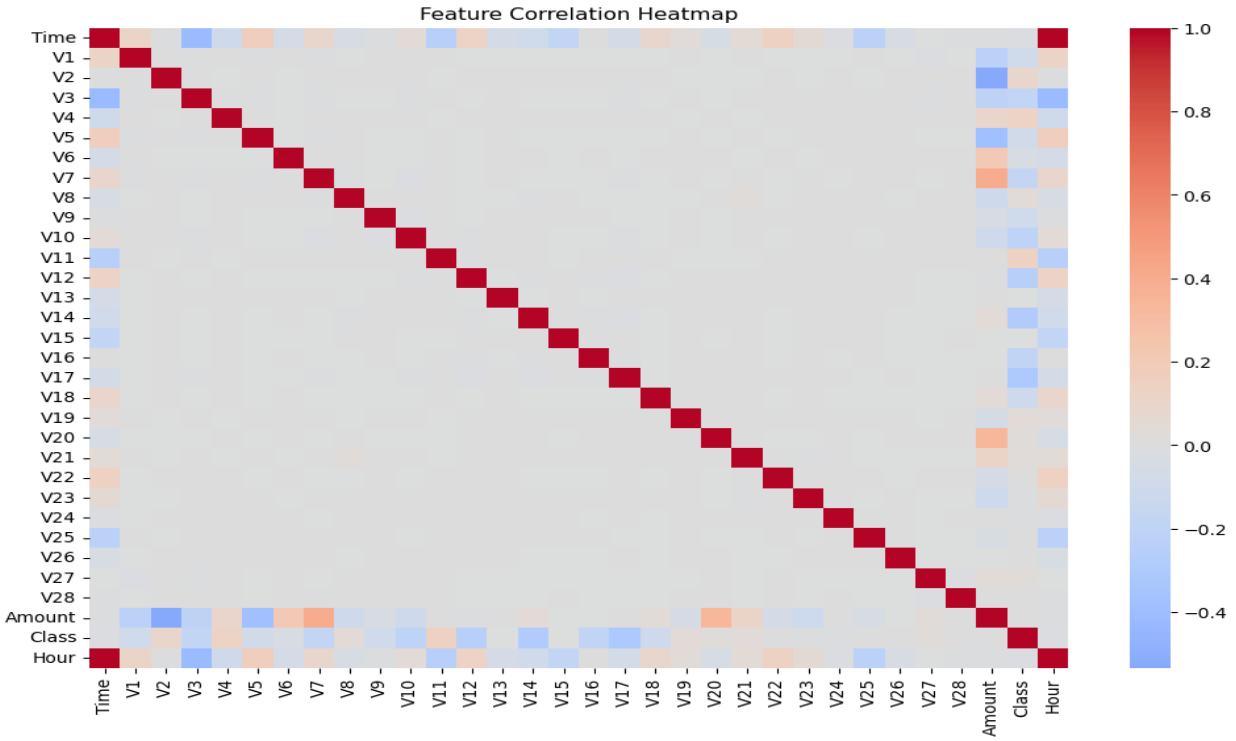
- This chart shows the distribution of all transactions throughout the day.
- There is a clear pattern where the number of transactions is lowest during the early morning hours, specifically from 2 AM to 6 AM.
- Transaction activity significantly increases during business hours, from around 8 AM to 9 PM.
- The chart shows peaks in activity around midday and late evening.
- This provides a baseline of normal user behavior, which is useful for comparing against fraudulent activity.
-





This chart compares the distribution of normal and fraudulent transactions over a 24-hour period.

The chart reveals a severe class imbalance at all hours of the day. The number of fraudulent transactions (Class 1) is consistently and drastically lower than the number of normal transactions (Class 0).



Conclusion:

- The credit card fraud detection project was initiated to address the increasing incidence and financial risk of fraudulent transactions in the banking and fintech sector. The project began with assembling a large and diverse dataset containing both legitimate and fraudulent credit card transactions. Extensive efforts were made to clean the data, remove duplicates, and handle missing entries, ensuring the foundation for trustworthy analysis. Exploratory data analysis revealed a significant class imbalance—fraudulent activities represented a very small percentage of all transactions, yet accounted for a substantial aggregate monetary value.

- Advanced unsupervised machine learning algorithms such as Isolation Forest, PCA, and DBSCAN were employed to flag anomalous transactions based on their deviation from normal spending patterns, transaction times, and amounts. Models were carefully evaluated through both quantitative metrics and visual inspection of detected anomalies. Special attention was paid to tuning detection thresholds to minimize both false positives and false negatives, a crucial challenge due to the rarity of fraud events.

- Using robust SQL queries and visualization tools, actionable insights were generated that mapped the distribution of fraud across temporal and monetary dimensions. These findings were consolidated into interactive BI dashboards allowing operational teams to rapidly identify spikes, trends, and patterns in fraudulent activity. The system's multi-step workflow, from preprocessing to modeling to real-time dashboarding, ensures timely fraud alerts and effective decision support for risk management teams.

- Collaboration among data engineers, analysts, machine learning developers, and evaluators was key to maintaining high standards of data integrity and analytical rigor. The project demonstrated that combining automated machine learning detection with human oversight and regular feedback loops can significantly enhance fraud mitigation efforts. Continuous model retraining and system improvements were recommended to keep pace with evolving fraud tactics, securing both financial assets and customer trust. Overall, the project delivered a practical and scalable solution for real-world credit card fraud detection that balances security, operational efficiency, and adaptability.

Submitted by:

1. Data Engineer &EDA Specialist -Saumya
2. ML Developer - Soniya
3. Model Evaluator - Srimathi
4. SQL Analyst -Sukanya
5. Insights and Reports - Sumana