Exploratory Data Analysis and Cleaning Report

Simulated Credit Card Fraud Dataset (2019–2020)

Mohamed Fouad Rabahi

September 19, 2025

1 Introduction

The goal of this project is to build a fraud detection pipeline using a realistic, simulated credit card transaction dataset generated with the *Sparkov Data Generation* tool. The dataset spans the period from **1 January 2019 to 31 December 2020**, covering transactions from approximately 1000 customers at 800 merchants.

Detecting fraud is a highly challenging task due to:

- **Severe class imbalance:** fraud is less than 1% of all transactions.
- **High dimensionality:** mixture of numerical, categorical, temporal, and location-based features.
- **Privacy concerns:** sensitive personally identifiable information (PII) must be anonymized.

This report describes the data preparation and exploratory analysis phase, ensuring the dataset is clean, anonymized, and ready for modeling.

2 Dataset Overview

- **Rows:** 1,296,675 transactions
- **Columns:** 23 raw features
- **Data types:** 12 object (categorical/text), 6 integer, 5 float
- **Duplicates:** 0
- **Missing values:** None in essential columns

The key features included:

- **Transaction details:** timestamp, amount, merchant, category, geographic location.
- **Customer details:** name, address, gender, date of birth, job.
- **Fraud label:** `is_fraud` (0 = legitimate, 1 = fraud).

3 Data Cleaning

3.1 Anonymization

To protect privacy, we anonymized sensitive information. Specifically:

- **Credit card numbers** (`cc_num`) were replaced with a hashed identifier (`cc_hash`).
- **Customer names** (`first`, `last`) were combined and hashed into a single `name_hash`.
- **Street addresses** (`street`) were dropped entirely, since geographic features at city/state/zip level remained sufficient.

As a result, the following raw PII columns were dropped from the dataset:

```
[cc_num, first, last, street]
```

3.2 Datetime Parsing

The column `trans_date_trans_time` was parsed into a proper timestamp variable `trans_datetime`. From this, we derived:

- Hour of day (`trans_hour`)
- Day of week (`trans_dow`)
- Month and year

3.3 Feature Engineering

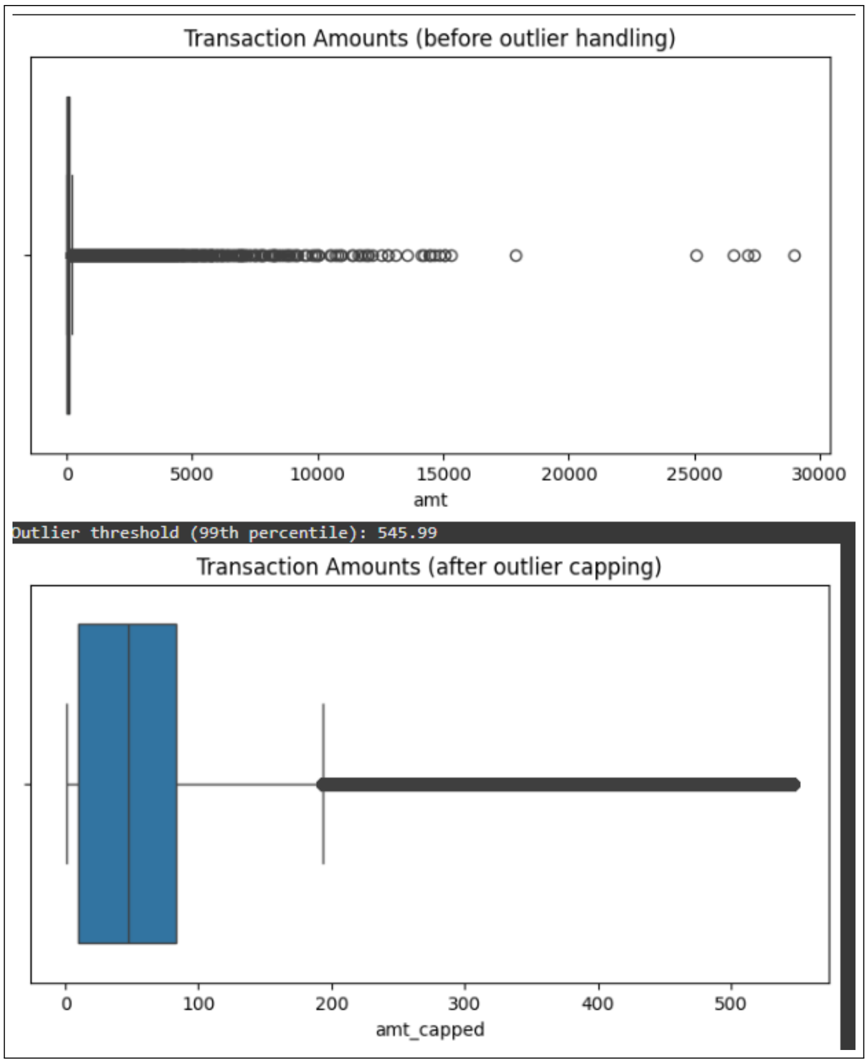
Additional features were created:

- **Log-transformed amount:** `amt_log1p = log(1+amt)`
- **Age of customer:** derived from date of birth
- **Transaction counts:** per customer and per merchant
- **Customer–merchant distance:** haversine distance using latitude/longitude

3.4 Handling Outliers

Transaction amounts showed a heavy-tailed distribution. To mitigate the effect of extreme values:

- We visualized transaction amounts with a boxplot.
- Defined the **99th percentile** as the threshold for outliers.
- All values above this threshold were capped to the 99th percentile.
- New features were created: `amt_capped` and its log transform `amt_log1p_capped`.



```
Outlier threshold (99th percentile): 545.99
```

Figure 1: Transaction amounts before and after outlier capping.

3.5 Handling Duplicates and Missing Values

No duplicate rows were present. For numeric fields with occasional missing values, the median was used for imputation.

4 Exploratory Data Analysis

4.1 Class Distribution

- **Total transactions:** 1,296,675
- **Legit:** 1,289,169
- **Fraud:** 7,506
- **Fraud ratio:** 0.58%

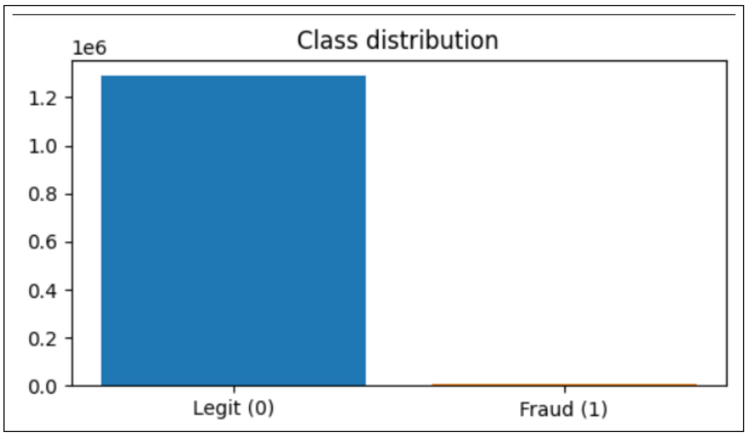


Figure 2: Distribution of legitimate vs fraudulent transactions. Legitimate (blue), Fraudulent (red).

4.2 Fraud by Category

Fraud was not uniformly distributed. The highest fraud rates appeared in:

- **shopping_net** (1.75%)
- **misc_net** (1.45%)
- **grocery_pos** (1.41%)

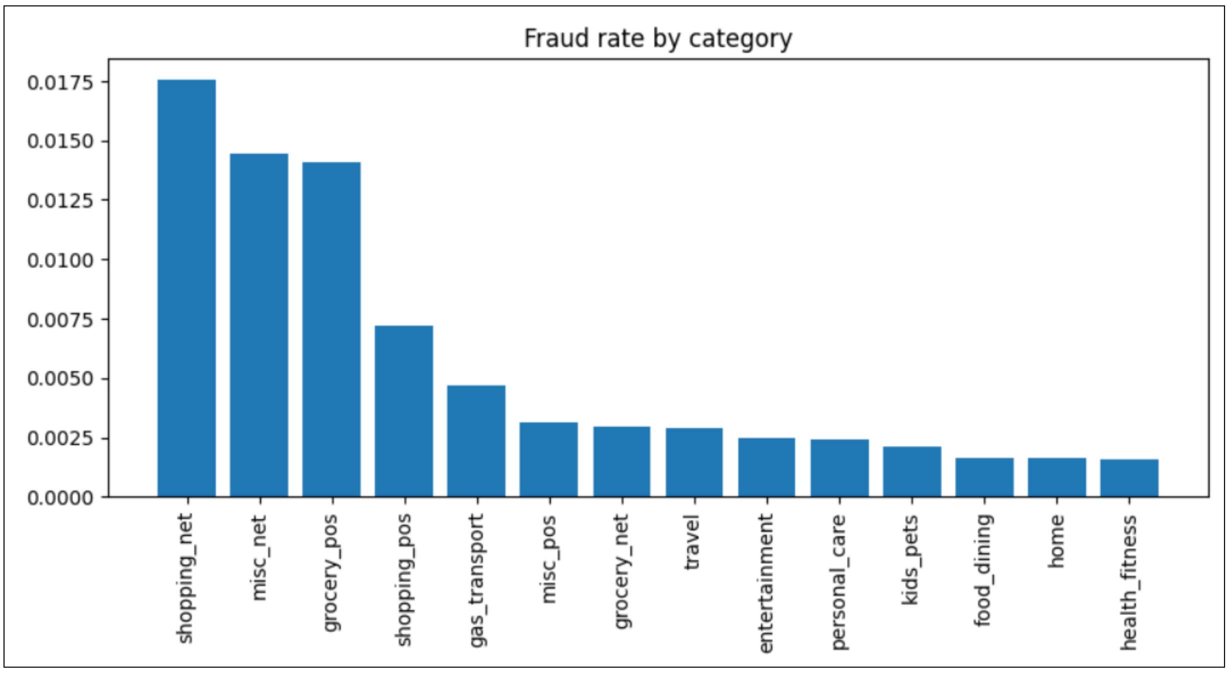


Figure 3: Fraud rate by transaction category.

4.3 Correlation Analysis

Correlations with the fraud label `is_fraud` showed:

- Amount (`amt`): 0.219
- Log amount (`amt_log1p`): 0.120
- Customer transaction count: 0.055
- Temporal features (hour, month, age): ∼0.012

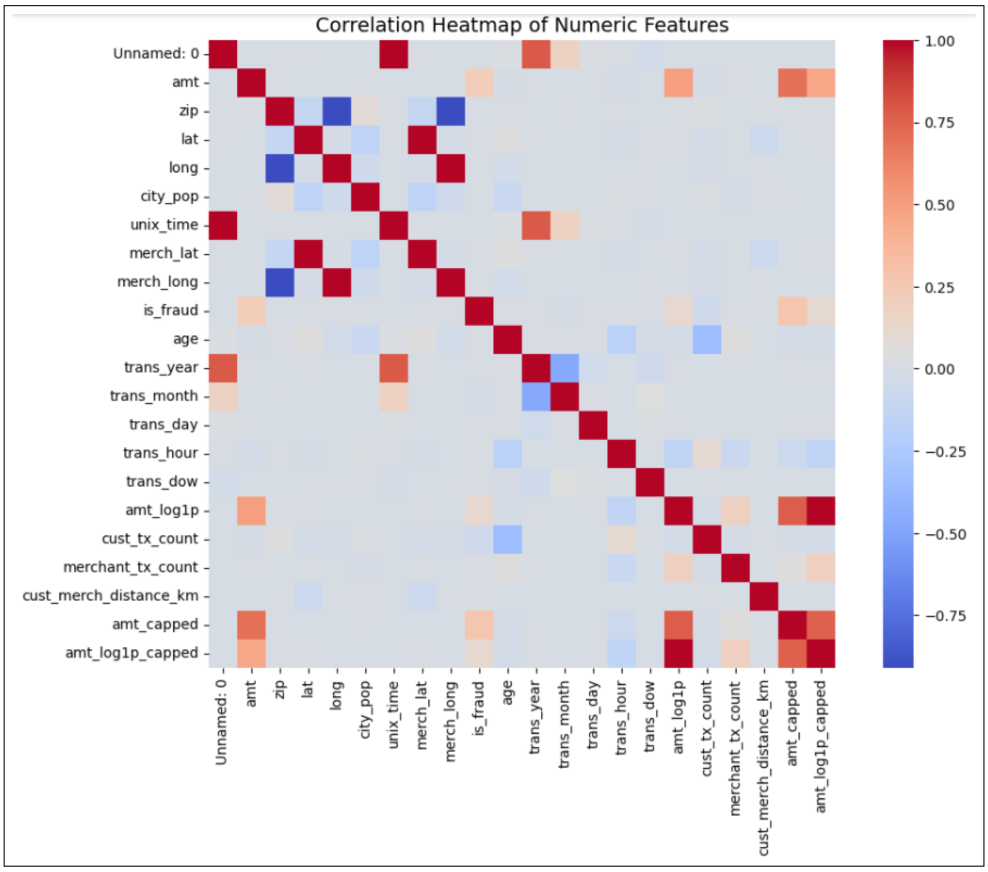


Figure 4: Heatmap of feature correlations with fraud label.

5 Conclusion and Next Steps

The dataset has been:

- Cleaned and anonymized (PII removed)
- Enhanced with engineered features
- Outliers handled through capping at the 99th percentile
- Verified to contain a realistic fraud ratio of ∼0.58%

Next steps:

1. Train/test split with chronological separation for realistic evaluation.
2. Train baseline models (Logistic Regression, Random Forest, XGBoost).

3. Handle class imbalance using class weights, SMOTE (on training set only), and threshold tuning.

4. Evaluate models using Precision-Recall AUC and business-driven metrics (Precision@K).