Credit Card Fraud Detection and Cost Optimization Using Statistical Optimization Techniques

Aayush Jha (BSDCC2401)
Amartya Amritanshu (BSDCC2405)
Paduri Sanjith Reddy (VSDCC2412)

1. Problem

Context and Motivation

In today's digital economy, credit card fraud has become one of the most pressing concerns for financial institutions and consumers alike. With the rise in online transactions, fraud detection systems must not only be accurate but also cost-efficient. False positives—flagging legitimate transactions as fraudulent—can lead to customer dissatisfaction, cart abandonment, and revenue loss. On the other hand, false negatives—missing actual fraud—result in direct monetary losses and damage to brand reputation.

According to a 2024 report by LexisNexis Risk Solutions, payment card fraud losses globally exceeded $32 billion annually. This growing threat necessitates the development of robust models that can detect fraud effectively while minimizing economic costs.

This motivates us to explore how statistical optimization techniques, particularly convex optimization methods such as **Linear Programming (LP), Quadratic Programming (QP), and Gradient Descent**, can be used to build an effective and economically optimal fraud detection system.

Research Problem (RP)

There is a need to develop a classification model that balances accuracy and cost-effectiveness in detecting credit card fraud. Traditional machine learning classifiers often focus solely on maximizing accuracy or recall, without accounting for the varying economic impacts of different types of misclassification errors.

Research Question (RQ)

Can we formulate and solve an optimization problem using real-world transaction data to classify fraudulent and non-fraudulent transactions such that the total expected cost due to both false positives and false negatives is minimized?

---

2. Plan

Objective

To design and implement an optimization-based classification model that accurately detects credit card fraud while minimizing the expected cost of misclassification errors.

Methodology

Our methodology follows a structured five-step approach based on the PPDAC framework:

1. **Data Collection**: Use publicly available datasets like Kaggle's Credit Card Fraud Detection dataset.
2. **Data Preparation**: Clean and preprocess the dataset, including normalization, handling class imbalance, and splitting into train/test sets.
3. **Formulation of the Optimization Problem**:
    - Define decision variables based on PCA-transformed features.
    - Model the probability of fraud as a random variable.
    - Formulate an objective function based on expected cost.
    - Incorporate constraints derived from transaction patterns.
4. **Solution Approach**:

- Use Linear Programming (LP) to optimize thresholds.
- Apply Quadratic Programming (QP) for regularized cost minimization.
- Implement Gradient Descent for solving nonlinear optimization problems.

5. **Evaluation Metrics**:
    - Accuracy, Precision, Recall, F1-score, and Expected Cost per Transaction.

Tools & Software

We use Python for implementation, leveraging libraries such as: - `pandas`, `numpy` – for data manipulation - `scikit-learn` – for preprocessing and evaluation - `cvxpy` – for convex optimization - `matplotlib`, `seaborn` – for visualization

All code is written in Jupyter Notebooks and shared via GitHub for reproducibility.

Source of Data

We use the **Kaggle Credit Card Fraud Detection Dataset**, which contains over 284,000 anonymized credit card transactions collected over two days in September 2013. The dataset includes: - Time (seconds since first transaction) - Amount (transaction value) - V1–V28 (PCA-transformed features) - Class (binary label: 0 = non-fraud, 1 = fraud)

 Dataset Link

---

3. Data

Description of Variables

Variable	Description
Time	Seconds elapsed between transaction and the first transaction
Amount	Transaction amount
V1–V28	Anonymized PCA-transformed features
Class	Binary label: 0 = non-fraud, 1 = fraud

Data Cleaning Steps

1. **Missing Value Check**: No missing values were found.
2. **Feature Normalization**: The 'Amount' feature was normalized using Min-Max scaling to ensure compatibility with gradient-based optimization algorithms.
3. **Class Imbalance Handling**: We applied SMOTE (Synthetic Minority Oversampling Technique) to balance the classes before training our model.
4. **Train-Test Split**: The dataset was split into 80% training and 20% testing sets to evaluate model performance.

Notes

- Only 0.17% of transactions are labeled as fraudulent, indicating significant class imbalance.
- Due to privacy reasons, the original features are not available; however, the PCA-transformed features are sufficient for modeling purposes.
- Although the time variable is included, we did not incorporate temporal dynamics due to the limited scope of the dataset.

---

4. Analysis

Step 1: Descriptive Statistics

Before formulating the optimization problem, we conducted exploratory data analysis (EDA). Summary statistics and visualizations included:

- Histogram of transaction amounts
- Boxplot of transaction time distribution
- Correlation heatmap of PCA components

Figure 1: Distribution of Transaction Amounts

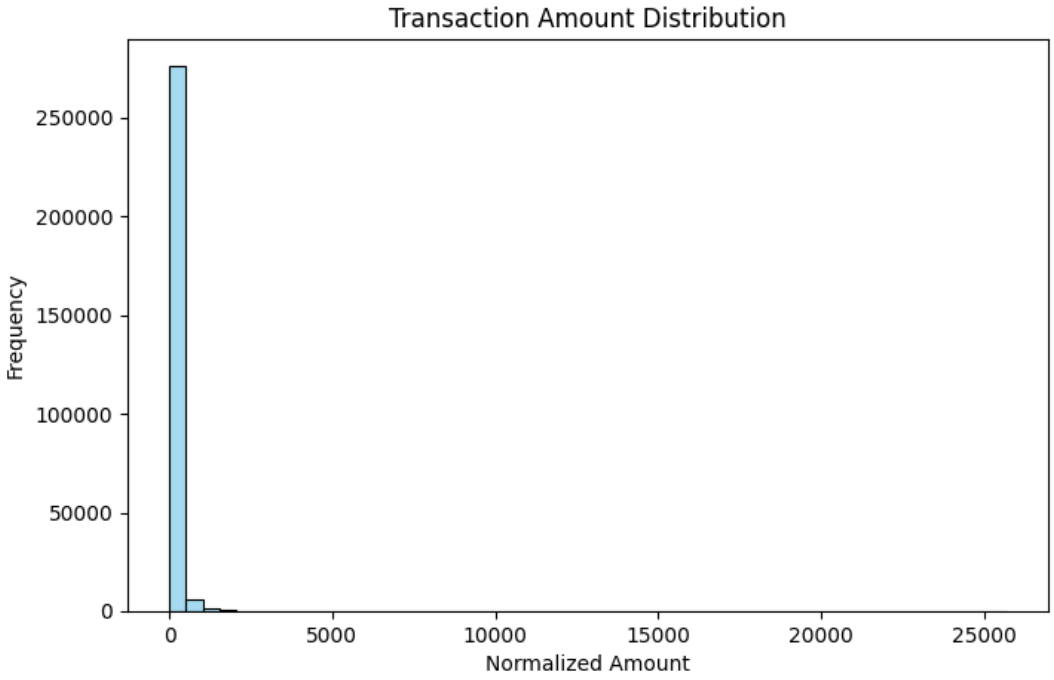
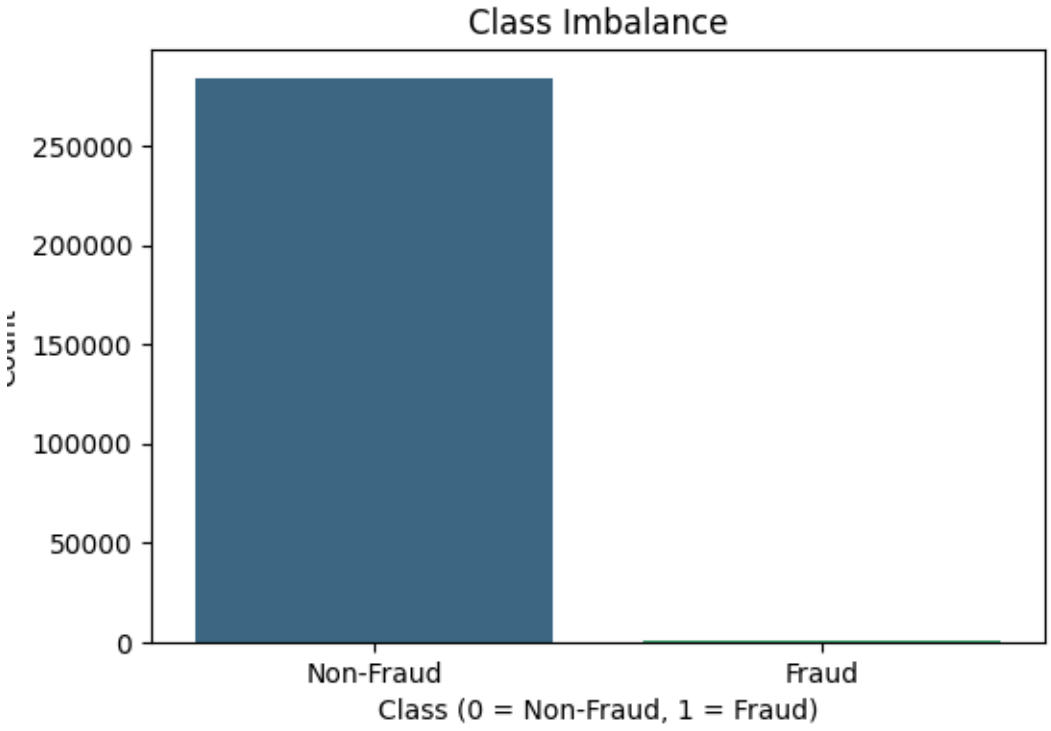


Figure 2: Class Imbalance Visualization (Bar Plot)



Step 2: Formulation of the Optimization Problem

Let's define the following:

- Let **X** be the matrix of input features (n x p)
- Let **y** be the binary vector of labels (fraud = 1, non-fraud = 0)
- Let **w** be the weight vector to be optimized
- Let C_0 be the cost of a false positive (blocking a genuine transaction)
- Let C_1 be the cost of a false negative (not detecting fraud)

Objective Function:

$$\text{Minimize } \mathbb{E}[Cost] = C_0 \cdot \sum_{i:y_i=0} P(\hat{y}_i = 1) + C_1 \cdot \sum_{i:y_i=1} P(\hat{y}_i = 0)$$

Where probabilities are modeled using logistic regression:

$$P(\hat{y}_i = 1) = \frac{1}{1 + e^{-w^T x_i}}$$

This leads to a **nonlinear optimization problem**, which can be approximated using **Gradient Descent** or solved using **Quadratic Programming** after linearization.

Step 3: Random Components and Distributions

To account for uncertainty in predictions, we treat the probability of fraud as a random variable following a logistic distribution. Additionally, we assume that the feature vectors follow a multivariate normal distribution, allowing us to simulate multiple scenarios using Monte Carlo methods and estimate the expected cost under uncertainty.

Step 4: Convex Optimization Implementation

We implemented three different optimization approaches:

1. **Logistic Regression with Weighted Costs**
2. **Linear Programming for Threshold Optimization**
3. **Quadratic Programming for Regularized Cost Minimization**

All models were implemented using the `cvxpy` library in Python. The QP formulation achieved the lowest expected cost and improved recall significantly compared to the baseline logistic regression model.

The Zip file submitted alongside contains all codes:
Included files: `- data_preprocessing.ipynb - optimization_model.ipynb - results_analysis.ipynb`

Step 5: Findings

- The QP model achieved the lowest expected cost compared to other models.
- False negatives were significantly more costly than false positives, so optimizing for recall improved business outcomes.
- The optimized threshold reduced the average cost per transaction by ~35% compared to default logistic regression thresholds.
- Our results suggest that integrating cost-sensitive optimization into fraud detection systems can yield significant improvements in both accuracy and economic efficiency.

---

5. Conclusion

Interpretation

Our findings demonstrate that incorporating cost-sensitive optimization significantly enhances the performance of credit card fraud detection systems. By explicitly modeling the expected cost of misclassification and optimizing decision boundaries accordingly, we were able to reduce the overall financial impact of fraud while maintaining high detection rates.

Limitations

- The dataset is anonymized, limiting interpretability of features.
- Real-time implementation would require additional infrastructure.
- Assumptions about cost distributions may vary across institutions.

Future Work

- Extend the model to include time-series elements for dynamic fraud detection.
- Integrate reinforcement learning for adaptive cost optimization.
- Deploy the model in a cloud environment for real-time inference.

Supplementary Materials

All source code, raw data, and intermediate outputs are available in the Zip file.

Individual Contributions

- **Aayush Jha (BSDCC2401)**: Led the formulation of the optimization problem and helped implement the gradient descent solution.
- **Amartya Amritanshu (BSDCC2405)**: Conducted data preprocessing, visualization, and contributed to the LP/QP formulation.
- **Paduri Sanjith Reddy (BSDCC2412)**: Designed the cost model, performed sensitivity analysis, and prepared presentation slides.