Credit Card Fraud Detection

Project Report

Submitted By:
Vishal Tyagi

Project Duration:
04 November 2024 - 05 January 2025

Internship Program:
Infosys Springboard 5.0

Mentor's Name:
Muvendiran

Submission Date:
December 21, 2024

Contents

1 Executive Summary

This project focuses on building a robust credit card fraud detection system using machine learning models. The dataset used contains 284,807 transactions, with a significant class imbalance. Through detailed data analysis, feature engineering, and model comparison, we identified the Random Forest model as the most effective with a high ROC-AUC score.

2 Introduction

Credit card fraud is a critical issue in the financial sector. This project aims to create an effective fraud detection system to mitigate associated risks. Machine learning techniques were employed to analyze and predict fraudulent transactions based on historical data.

2.1 Objectives

- Develop accurate fraud detection models.
- Address class imbalance in the dataset.
- Compare multiple machine learning algorithms.
- Implement and evaluate oversampling techniques like SMOTE.

3 Dataset Analysis

3.1 Dataset Overview

The dataset contains 31 columns, including 28 PCA-transformed features, and two key features: Time and Amount. The target variable `Class` indicates transaction type (0: Legitimate, 1: Fraudulent).

3.2 Initial Observations

- Total Transactions: 284,807
- Legitimate Transactions: 284,315 (99.83%)
- Fraudulent Transactions: 492 (0.17%)
- Severe class imbalance with a ratio of approximately 578:1.

3.3 Exploratory Data Analysis (EDA)

3.3.1 Class Distribution

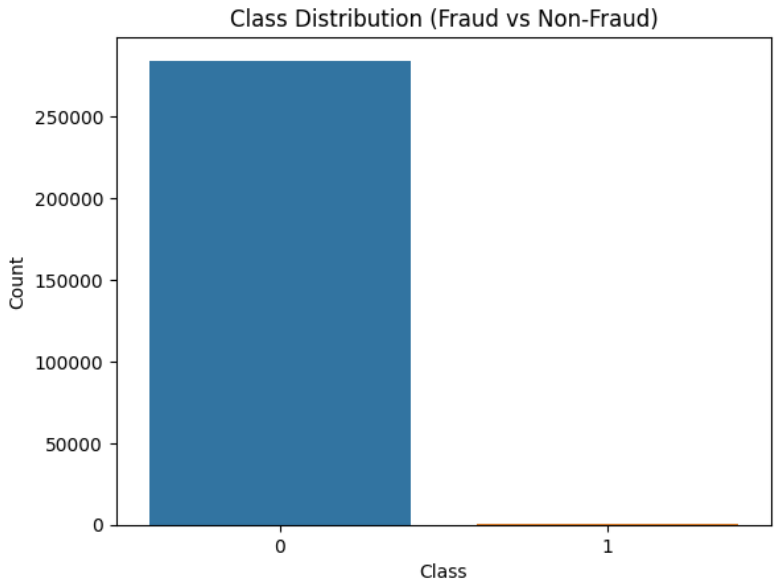

Figure 1: Distribution of Legitimate and Fraudulent Transactions.

3.3.2 Transaction Amount Distribution

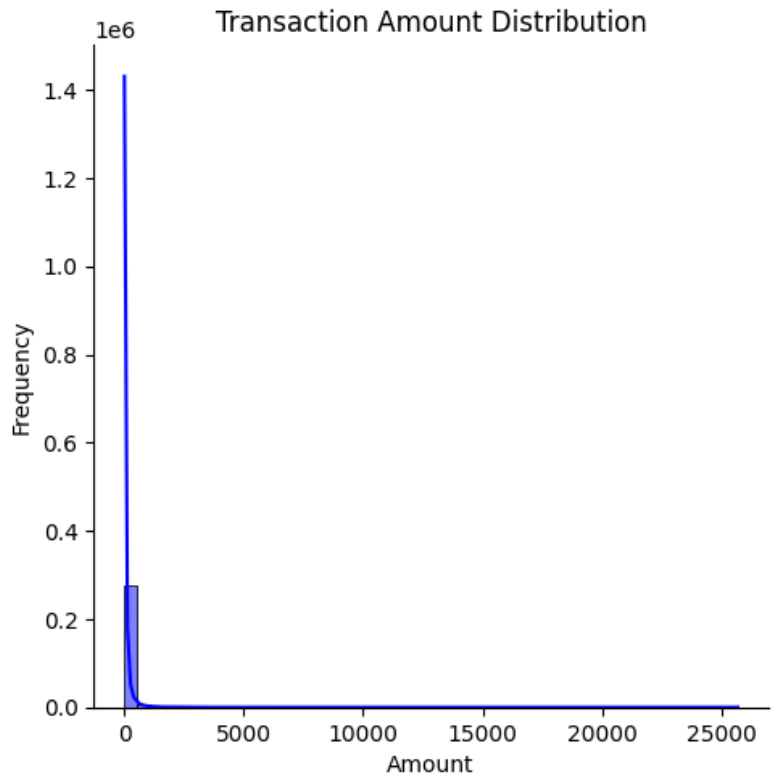

Figure 2: Transaction Amounts Across All Transactions.

3.3.3 Correlation Analysis

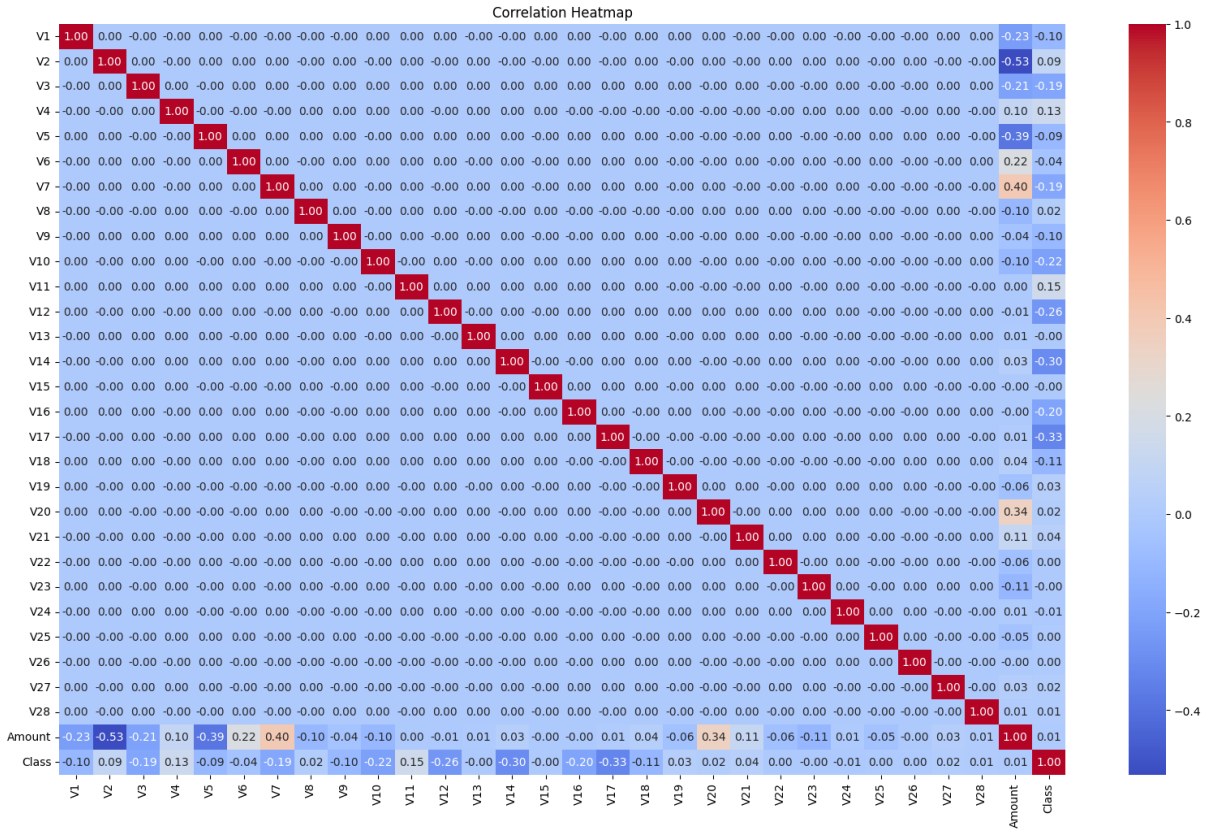

Figure 3: Heatmap Showing Feature Correlations.

3.3.4 Outliers

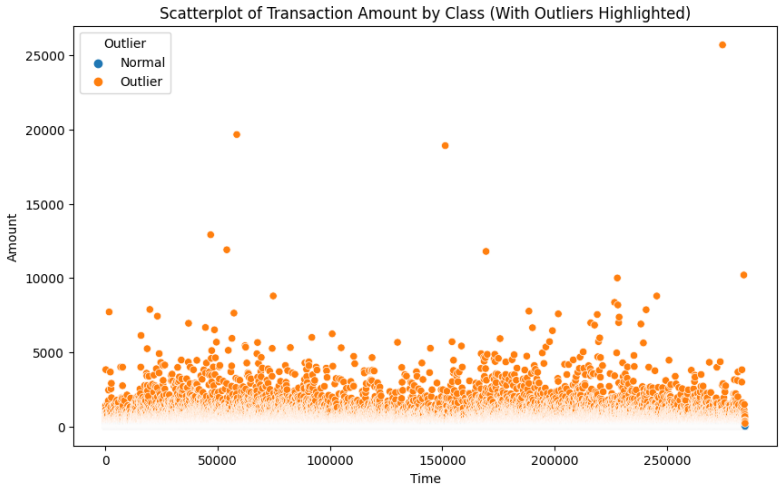

Figure 4: Detection of Outliers in Transaction Data.

3.3.5 Removal of Outliers

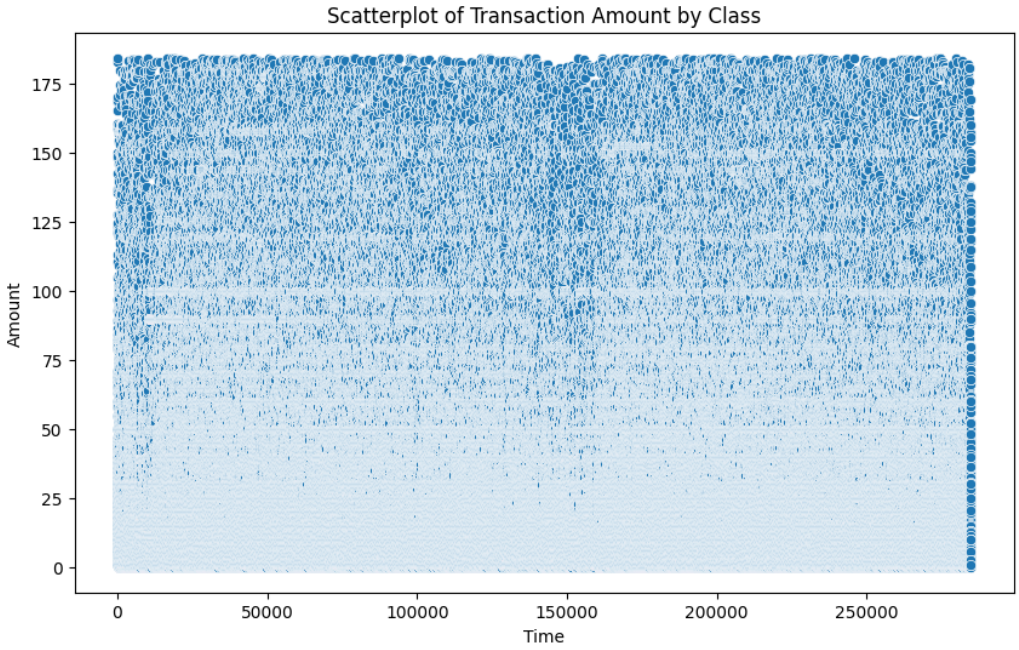

Figure 5: Visualization After Outlier Removal.

4 Methodology

4.1 Data Preprocessing

- Removed `Time` feature as it showed minimal correlation with fraud.
- Standardized `Amount` feature using `StandardScaler`:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df['Amount'] = scaler.fit_transform(df[['Amount']])
```

4.2 Handling Class Imbalance

Synthetic Minority Oversampling Technique (SMOTE) was used to balance the dataset:

```
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
```

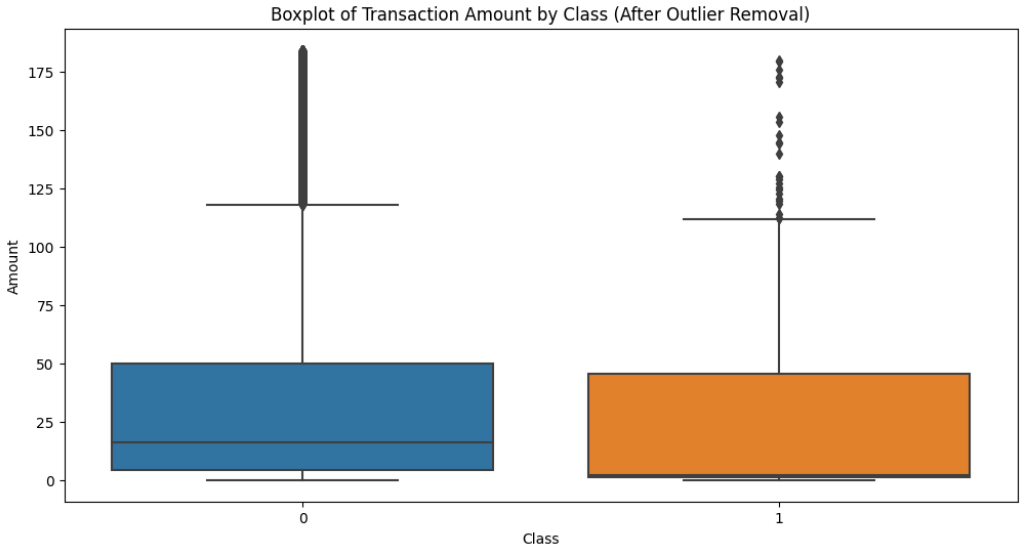

Figure 6: Balanced Class Distribution Using SMOTE.

5 Model Implementation

5.1 Logistic Regression

```
from sklearn.linear_model import LogisticRegression
logistic_model = LogisticRegression(max_iter=1000, random_state
    =42)
logistic_model.fit(X_resampled, y_resampled)
```

5.2 Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_resampled, y_resampled)
```

5.3 Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_resampled, y_resampled)
```

5.4 XGBoost

```
from xgboost import XGBClassifier
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='
    logloss', random_state=42)
xgb_model.fit(X_resampled, y_resampled)
```

5.5 Support Vector Machine

```
from sklearn.svm import SVC
svm_model = SVC(probability=True, random_state=42)
svm_model.fit(X_resampled, y_resampled)
```

5.6 LightGBM

```
from lightgbm import LGBMClassifier
lgbm_model = LGBMClassifier(random_state=42)
lgbm_model.fit(X_resampled, y_resampled)
```

5.7 CatBoost

```
from catboost import CatBoostClassifier
catboost_model = CatBoostClassifier(verbose=0, random_state=42)
catboost_model.fit(X_resampled, y_resampled)
```

5.8 Model Comparison

Model	ROC-AUC	Precision	Recall	F1-Score
Logistic Regression	0.9911	0.08	0.92	0.15
Decision Tree	0.8729	0.47	0.75	0.58
Random Forest	0.9668	0.90	0.85	0.87
XGBoost	0.9756	0.83	0.86	0.84
SVM	0.9827	0.13	0.91	0.23
LightGBM	0.9600	0.68	0.85	0.75
CatBoost	0.9858	0.57	0.87	0.69

Table 1: Performance Metrics of Models.

5.9 Comparison of ROC-AUC Scores

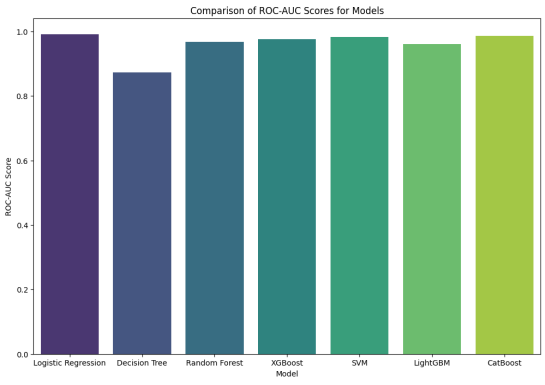

Figure 7: Comparison of ROC-AUC Scores Among Models.

6 Conclusion

6.1 Key Takeaways

- Random Forest demonstrated the highest overall performance.
- SMOTE effectively handled the class imbalance.
- Standardized features enhanced model accuracy.

6.2 Future Recommendations

- Explore deep learning methods for fraud detection.
- Develop a real-time fraud detection system.
- Investigate cost-sensitive learning approaches.

Acknowledgments

Special thanks to Infosys Springboard 5.0 and mentor Mr. Muvendiran for guidance and support throughout this project.

Declaration

I hereby declare that this project titled *Credit Card Fraud Detection* has been completed by me under the guidance of Mr. Muvendiran as part of the Infosys Springboard 5.0 internship program.

Vishal Tyagi
Date: December 21, 2024