Finding Fraudsters: Detecting Credit Card Fraud with Machine Learning

Zamiul Alam, Alex Gekow, Kazi Aatish Imroz, Roshan Joshi, Kayode Oyedele

June 2025

Executive Summary

Introduction

Credit card fraud is a big problem in the world of e-commerce, and difficult one to eliminate. In 2024, it was reported that consumers lost more than $12.5 billion to fraud [1]. As such, it is imperative that the fraudulent transaction detection system is efficient. This study uses boosted decision trees (BDT) to predict whether a transaction is fraudulent or not. The dataset used is obtained from **IEEE-CIS Fraud Detection** Kaggle competition.[2] The evaluation of the models is done using area under the curve (AUC) of receiver operating characteristic (ROC) curve via submission of fraud probability scores of test set to kaggle.

StakeHolders

The stakeholders include:

1. Credit card companies
2. Credit card holders
3. Merchants
4. Fraud prevention companies

Methods and Models

The most challenging part of the study was handling the features in the datasets. These features are divided into transaction features and identity features. There are 433 features in total, many of which are sparse (filled with NaNs) and have masked meanings. This meant selecting useful features was difficult. Different methods were tried for feature reduction:

- Remove features that are highly correlated with other features with unique values.
- Remove features that are highly correlated with other features but have less correlation to the fraud label.
- Remove features if they have low feature importance in model trained with simple classifier.

[1] https://www.ftc.gov/news-events/news/press-releases/2025/03/new-ftc-data-show-big-jump-reported-losses-fraud-125-billion-2024

[2] Addison Howard et al. *IEEE-CIS Fraud Detection*. `https://kaggle.com/competitions/ieee-fraud-detection`. Kaggle. 2019.

In addition, sparse features were also pre-processed in different way - the NaNs were either filled with dummy values or replaced with mean/median/mode. However, masked nature of features meant that there was no good way to decide if either of those two methods was a good choice. In the end, the features that were highly important in a few of the best-trained models were selected.

Boosted Decision Trees were used to train model for classification as they perform well out-of-the-box, are faster to train, and higher dimensionality of features does not affect them as much. Different BDT frameworks were tested which included XGBoost, LightGBM, and CatBoost. Hyper-parameter optimization was tested with optuna library which samples hyper-parameters n times using smart algorithms and optimizes them based on given scoring function (f1 score in our case). The final model from all three libraries is aggregated in an ensemble where the final output is a weighted average of the individual model outputs.

Results

Out of the models tried, XGBoost was the best performing one on the Kaggle test set. It got an AUC of 0.905 on 80% of test set, and 0.928 on remaining 20%. However, in as highly imbalanced dataset as this, AUC is not as appropriate of a metric for performance as a metric like f1 score or AUCPRC (area under precision recall curve). In the test dataset, split from training set (10% of training set), the AUPRC for the best model was 0.911.

Model	AUC (Private, using kaggle test set)	AUC (Public, using kaggle test set)	AUPRC (Splitting the training set)
XGBoost	0.905	0.928	0.911
LightGBM	0.896	0.925	0.868
CATBoost	0.890	0.914	0.893
Ensemble	0.907	0.933	

Figure 1: Comparison of AUC and AUPRC scores for LightGBM, XGBoost, CatBoost, and Ensemble (combination of the three models).

Future Directions

In the future, hyper-parameter optimization using full grid search can be done rather than sampling different combinations of hyper-parameters. However, this requires good compute power. In addition, the nature of the problem means that anomaly detection might be a suitable algorithm to be tried on this dataset. So, one can try to optimize anomaly detection algorithms, using both tree-based and neutral network based methods.