Credit Card Fraud Detection Project Report

23.04.2025

Awab M Erwa
Machine Learning Practitioner

Overview

In this project, I developed a classification model to distinguish between fraudulent and non-fraudulent credit card transactions made by European cardholders in September 2013. The dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. This task was a competition held by Kaggle.

EDA Findings

1. As shown in Figure 1, the majority of transaction amounts fall within the range of 0 to 200.

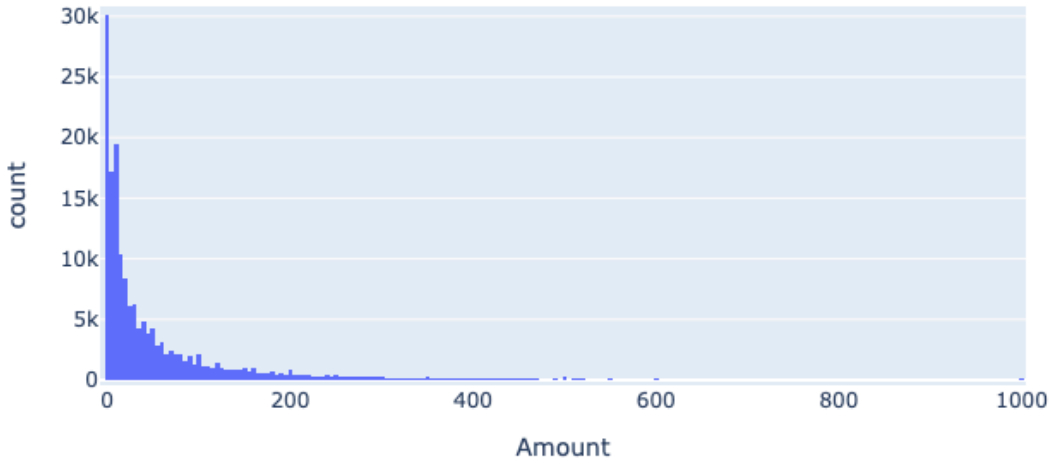

Figure 1

2. A large transaction amount does not necessarily indicate fraud; in fact, most high-value transactions are non-fraudulent, as illustrated in Figure 2.
3. The features V3, V4, V7, V10, V11, V12, V14, V16, V17, and V18 exhibit the highest correlation with the target class (fraud or non-fraud).
4. The Time and Amount features show a low correlation with the target class.

Figure 2

Feature Engineering

In the feature engineering step, following the removal of duplicate records, I performed several tasks, including:

Scaling and Transformation

First, I applied Min-Max scaling to the data, followed by a logarithmic transformation. These preprocessing steps contributed to an improvement in the model's performance.

Over/Under Sampling

Finally, to address the class imbalance in the dataset, I employed two techniques: undersampling using RandomUnderSampler and oversampling using SMOTE.

Learned Lesson

I experimented with transforming the data using PolynomialFeatures at various degrees; however, this approach negatively impacted the model's performance. Additionally, in this particular task, Min-Max scaling yielded better results compared to StandardScaler.

Modeling

In the modeling phase, I trained four models, namely:

LogisticRegression

For this model, after experimenting with various hyperparameters and techniques, the LogisticRegression model that achieved the highest validation F1-score was configured with the following parameters and techniques: default scikit-learn parameters, a threshold of 0.5, a polynomial degree of 2, and the use of undersampling only.

RandomForestClassifier

The RandomForestClassifier model that achieved the best validation F1-score was configured with the following parameters and techniques: max_depth=7, n_estimators=5, a threshold of 0.5, a polynomial degree of 1, and a combination of undersampling and oversampling. In this case, I introduced two techniques to address class imbalance: first, I incorporated class weights in the model parameters, assigning a weight of 0.5 to class 0 and 1 to class 1, which implies that errors in class 0 incur half the penalty compared to class 1. Additionally, I set warm_start=1 in the model, trained it on the undersampled and oversampled dataset, and then fine-tuned the model using the entire dataset.

MLPClassifier(Neural Network)

The MLPClassifier model that achieved the best validation F1-score was configured with the following parameters and techniques: one hidden layer with a size of 25, a learning rate of 0.1, a threshold of 0.1, a polynomial degree of 2, a combination of undersampling and oversampling, and warm_start=1.

VotingClassifier

I trained a VotingClassifier that combined two models: LogisticRegression and RandomForestClassifier. I used the best configurations for both models, as described earlier, with a threshold of 0.2 and a polynomial degree of 1. This ensemble model yielded the best results.

Figure 3 illustrates the results of the models on the validation dataset, while Figure 4 presents the data and modeling pipeline.

Metrics	LogisticRegression	MLPClassifier(NN)	RandomForsetClassifier	VotingClassifier
F1-Score	0.81	0.82	0.84	0.85
PR-AUC	0.65	0.68	0.71	0.73

Figure 3

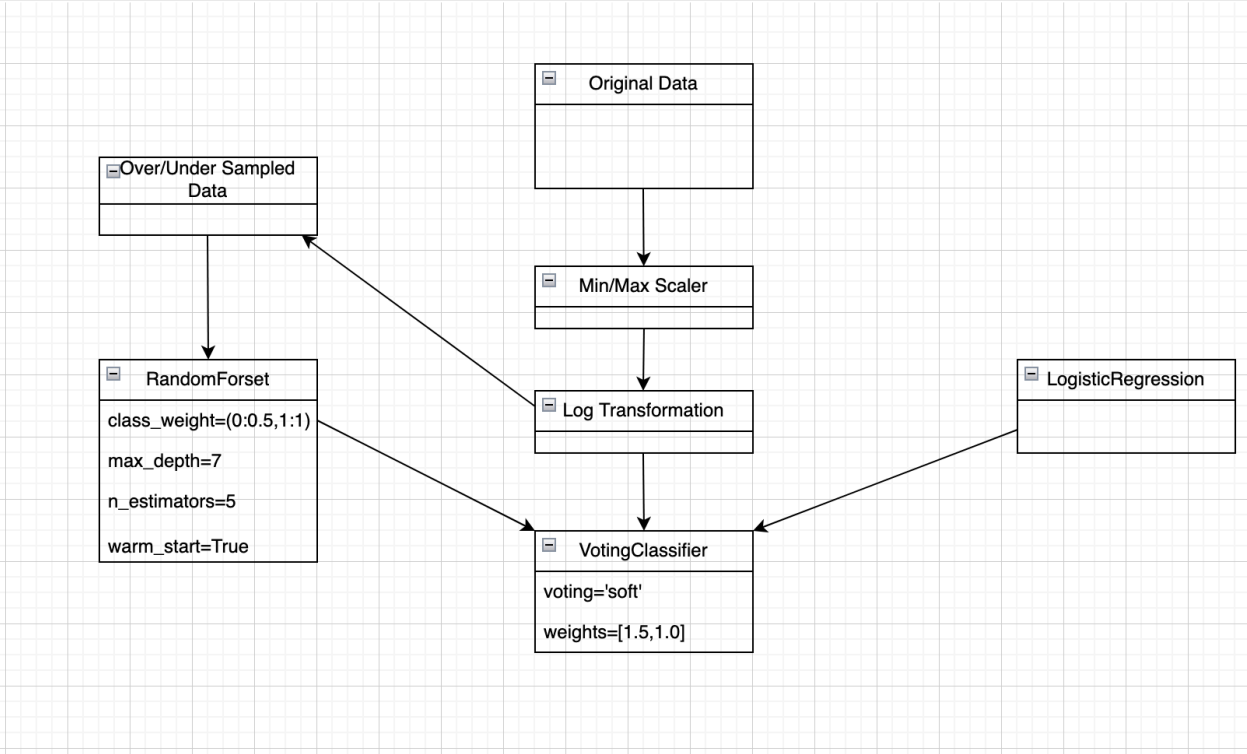

Figure 4