Fraud Detection Report

Project title: Credit card fraud detection

Author: Ahmed diab

Date: September 2025

Abstract:

This project credit card fraud detection aims to able to classify the fraud transactions in by given 31 features and 56960 examples but there are 28 feature are anonymous, also the data is highly unbalanced with just 0.172% positive class ration, so we focuses on F1-score then recall, and try several algorithms (logistic regression, random forest, voting classifier, xgboost, light-boost, cat-boost) we reach for the highest f1-score from random forest with [Val_f1score: 89% and Test_f1score: 80%], after applying preprocessing remove duplicates, transform time, apply MinMaxScaling, apply Ovesampling with factor: 80

This project inspired from Kaggle competition: Kaggle Fraud Detection.

Introduction:

Accurately classify the fraud transactions is critical to prevent the fraud transactions and reduce thefts and crimes, but most of the transactions are normal transactions so we don’t want to detect them as fraud so we try to build efficient model to detect fraud detection only; after deploying it in banks this will reduce the fraud transactions.

Data Set Description:

Sources: inspired by Kaggle "Fraud Detection" with some changes

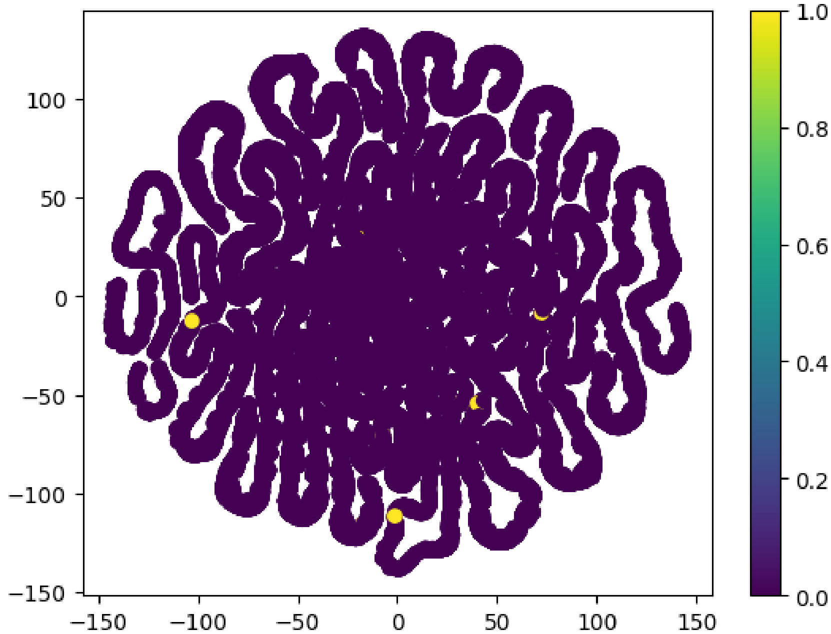
Size: 56960 columns & 31 feature for Train_val data

Features:

- Time: show the seconds between the first transaction and each transaction
- from: V1 to V28 are anonymous features
- Amount: the number of transactions happening in this time
- Class: The target feature 1 or 0

EDA:

- Target (Class) are highly unbalanced with 0.172% positive class



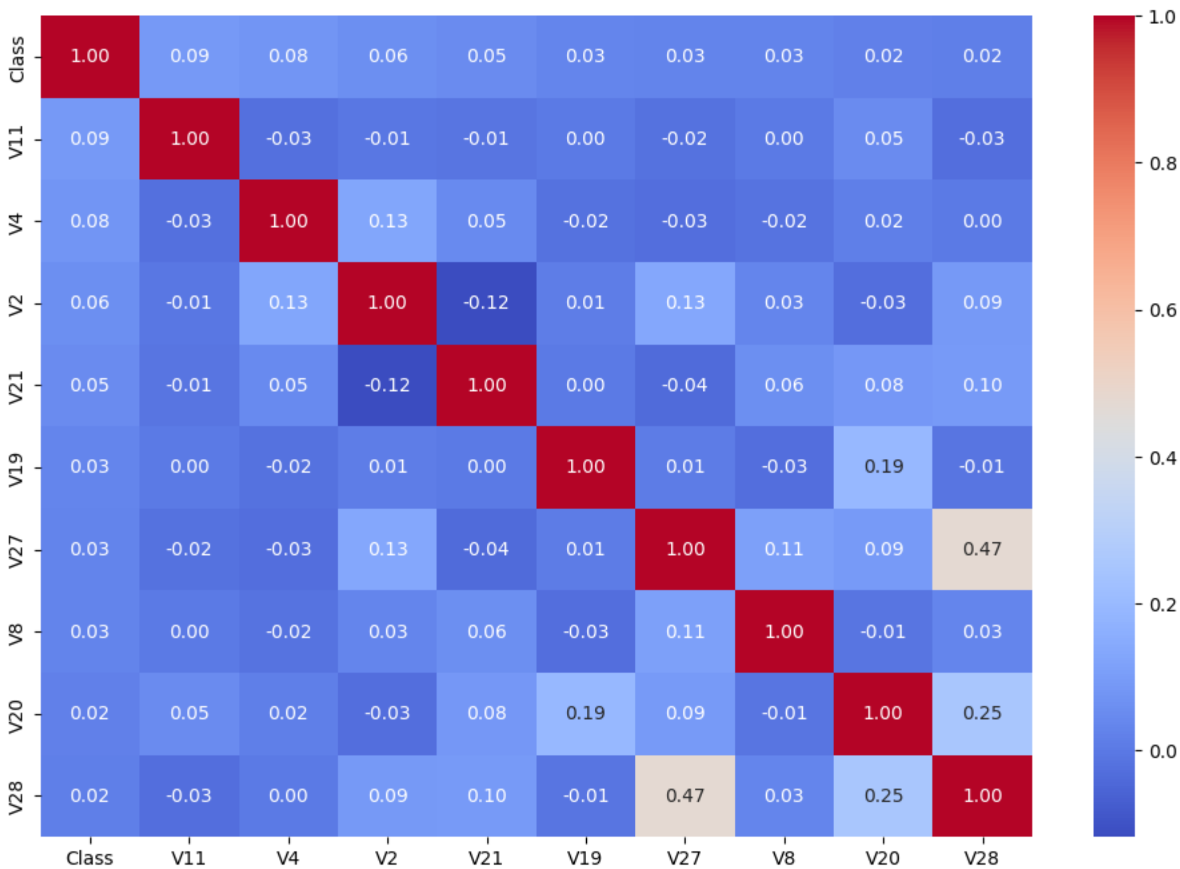
- The Data is Highly imbalanced with only 0.00157% positive class

- All features (except target) are floats

- Most of the features are Anonymous and normalized, 28 features

- the data is very weak (very low correlation)

Highest 10 feature correlation:



- there is no missing data (Clean)

Methodology:

Data Processing:

- The data was clean, no missing data
- There were 62 duplicate examples, drop them
- keep or drop outlier, keep them
- Change time from seconds to hours
- Apply MinMaxScaing to scale the data
- Try oversampling minority with factor=80

Models:

- Baseline Model: Logistic Regression
- Tree based Models: Random Forest Classifier
- Boosting technique: Xgboost, Light boost, Cat boost
- K Nearest Neighbors

Metrices:

So, the data is highly unbalanced we cannot use Accuracy as metric wee will use f1-score and recall, precision

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Results:

Model	F1-score data as it is	F1-score undersampling	F1-score oversampling
Logistic Regression	88%	80%	89%
Random Forest	88%	88%	89%

xgboost	77%	88%	89%
lightboost	79%	84%	83%
catboost	88%	80%	89%
KNN	56%	62%	88%

Random Forest:

We chose the Random Forest classifier model with parameters: Max_depth=9 and n_estimators = 50

Results:

Val_f1-score: f1-score 89% & Recall 81%

```
ud_Detection/credit_fraud_train.py
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     11362
           1       1.00      0.81      0.89       142

    accuracy                           1.00     11504
   macro avg       1.00      0.90      0.95     11504
weighted avg       1.00      1.00      1.00     11504

0.006114300150308123 0.2698072805139186 0.89
```

Test: f1-score 80% & Recall 82%

```
/ML_Live_Slides/HWs/Projects/Credit_Card_Fraud_Detection/test.py
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56812
           1       0.78      0.82      0.80        96

    accuracy                           1.00     56908
   macro avg       0.89      0.91      0.90     56908
weighted avg       1.00      1.00      1.00     56908
```

KNN:

We try k nearest neighbors (knn) classifier with the same preprocessing (cleaning, scaling, sampling) and use PCA to do feature reduction to reduce the time for prediction and reduce time in train-val from 1.33 to 0.44 seconds but it reduces the f1-score form 88% to 85% so we will choose without PCA.

We tunning to get the best n-neighbor 60, n_component 4

Results:

Train_Val: f1-score 88% & Recall 79%

```
venv\Scripts\python.exe e:/ML_Live_Slides/HWs/Projects/Credit_Card_Fraud_Detection/Try_KNN/knn_model_train.py
 """
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     11362
           1       1.00      0.79      0.88       142

    accuracy                           1.00     11504
   macro avg       1.00      0.89      0.94     11504
weighted avg       1.00      1.00      1.00     11504

Time Taken for Preiction 2.3627095222473145
0.18333333333333332 0.9 0.8098591549295775
```

Test: f1-score 75% & Recall 75%

```
venv\Scripts\python.exe e:/ML_Live_Slides/HWs/Projects/Credit_Card_Fraud_Detection/Try_KNN/test_knn.py
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56812
           1       0.75      0.75      0.75        96

    accuracy                           1.00     56908
   macro avg       0.87      0.87      0.87     56908
weighted avg       1.00      1.00      1.00     56908

10.439418315887451
```

Conclusion:

The model saved in model.pkl file it can now predict if the transaction is normal or fraud with 80% f1-score from tree Based algorithm random classifier with and can transformed as api and deployed for bank