EU Citizen Credit Card Fraud Detection

Presentor: Ali Asadnia

Introduction

In today's digital age, credit card fraud is a major concern for financial institutions and customers. Our project is to experiment multiple machine learning techniques to detect fraudulent transactions.

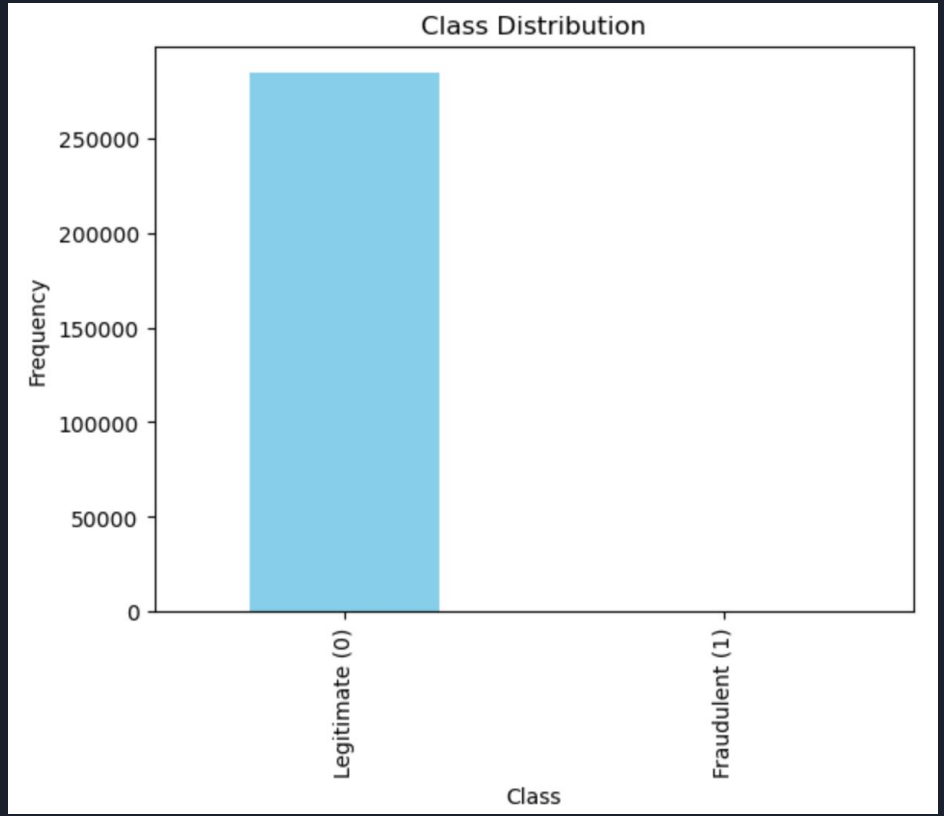
Fraudulent transactions follow complex patterns that are challenging to model with traditional programming. Therefore, it is essential to use data mining and machine learning techniques to learn from existing patterns and effectively identify unseen fraudulent transactions.

The Dataset

- The dataset used for this project contains more than 280,000 transactions.
- It splits the columns into 31 features, 28 of which have been PCA transformed. The rest are Time, Amount, and Class.

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	0.215153	69.99	0

Class Distribution



Preprocessing

The preprocessing was done in three steps:

- Duplicate Removal: The number of duplicates found was 1081.
- Missing values: There were no null/missing values in this dataset.
- Normalization: The values of the attributes “Time” and “Amount” were transformed using a “Robust Scaler” into values in the range (-1, 1) to follow the order of the rest of the features V1 through V28.
- Data Split: The dataset was split into training and testing data using a stratified method, the percentages used were 70% and 30% respectively

Cross-validation

- This step was carried out to compare different model performances on this data.
- To make sure the models have learned patterns and not the noise, it is necessary to test them against unseen data, that’s why I used 10-fold-cross-validation.

Rebalancing the Training Features and Labels

- Undersampling: used for the majority class “Normal” which has 99.8% support, by applying the “RandomUnderSampler” class from the imblearn package with a sampling strategy of 0.9.

- Oversampling: used for the minority class “Fraud” which has a bit over 1% support in the entire dataset. I used SMOTE not to train our model on duplicate samples. The sampling strategy was of 0.5.

Supervised Learning/Classification

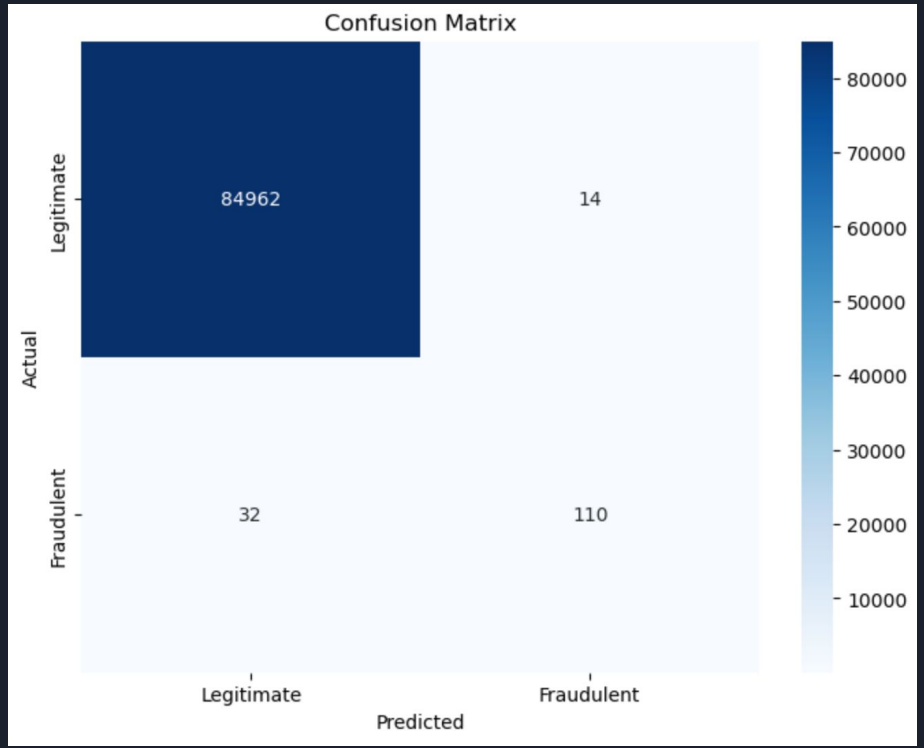
- Random Forest(RF)
- Gradient Boosting(GB)
- AdaBoost
- Naive Bayes(NB)

Classification Best Result Belongs to RF

	Metrics	Results
0	Accuracy	0.999460
1	Precision	0.887097
2	Recall	0.774648
3	F1_score	0.827068

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	84976
1	0.89	0.77	0.83	142
accuracy			1.00	85118
macro avg	0.94	0.89	0.91	85118
weighted avg	1.00	1.00	1.00	85118



Unsupervised Learning

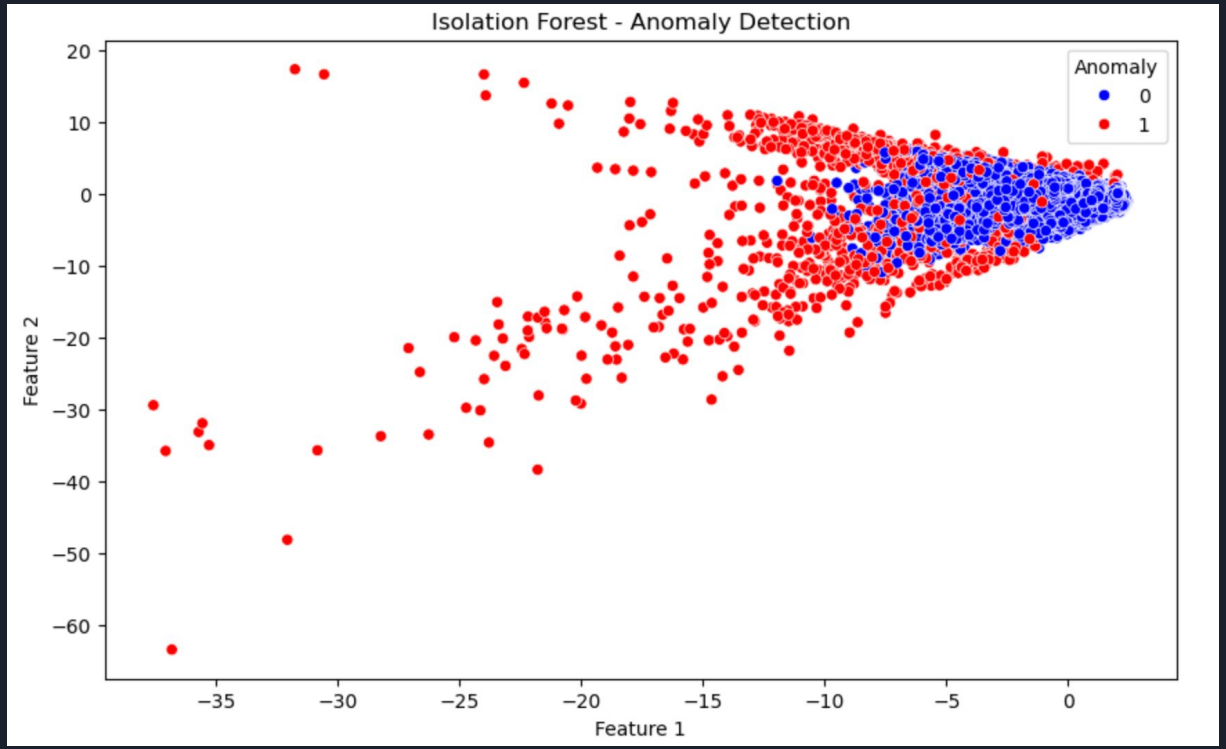
Anomaly Detection

- Isolation Forest
- Local Outlier Factor

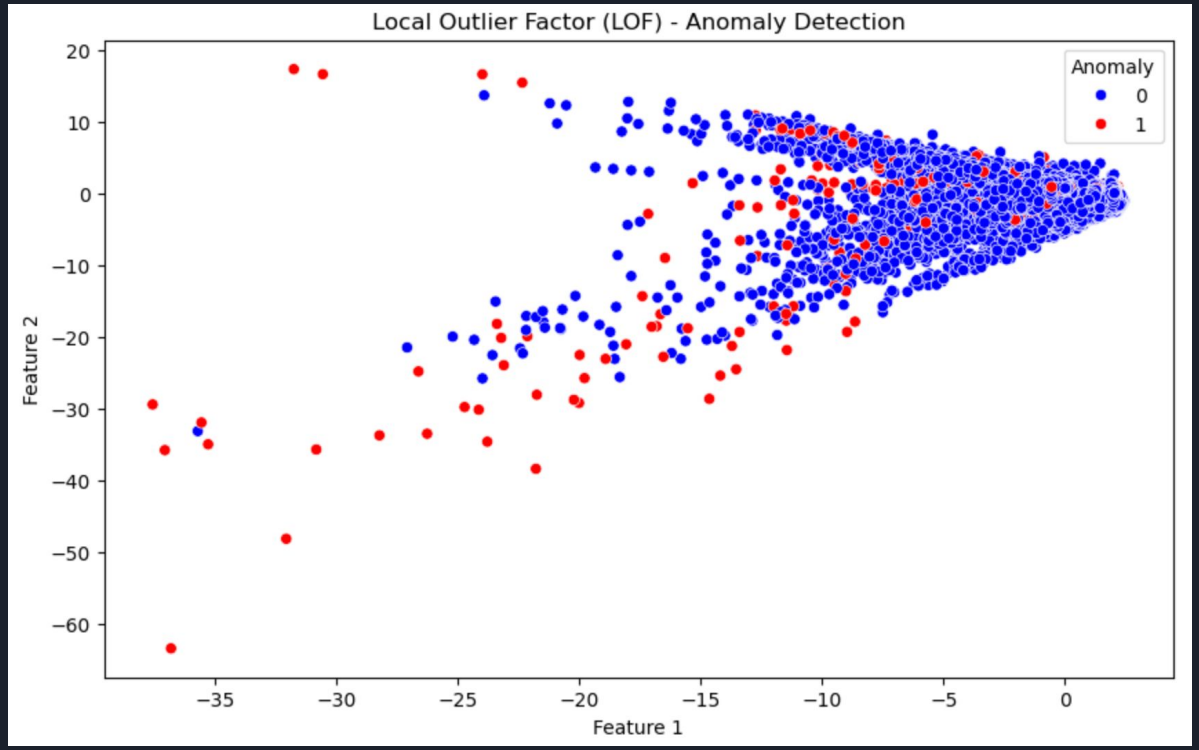
Clustering

- DBSCAN

Isolation Forest Anomaly Detection



Local Outlier Factor Anomaly Detection



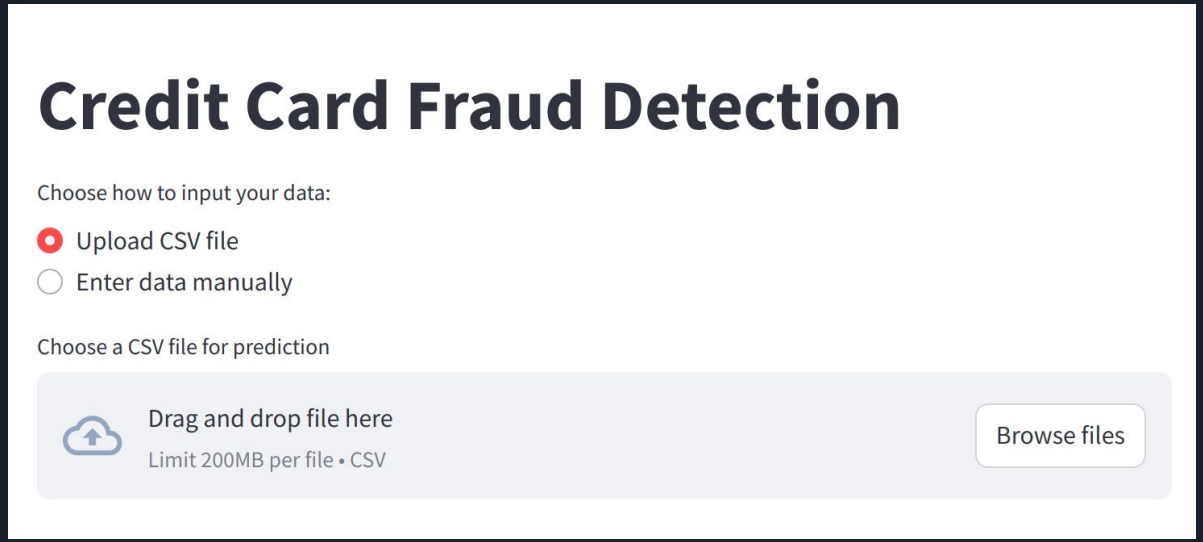
DBSCAN

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.06	0.12	84976
1	0.00	1.00	0.00	142
accuracy			0.06	85118
macro avg	0.50	0.53	0.06	85118
weighted avg	1.00	0.06	0.11	85118

Deployment

For the deployment of the fraud detection model, I utilized the Random Forest algorithm, which demonstrated the best performance during the model selection process. To provide a user-friendly interface for interacting with the model, I developed a web application using the Streamlit library in Python.



Address of the Website:
https://machine-learning-projects-frauddetection.streamlit.app/

Thank you for the attention