Credit-Card Fraud detection using Machine Learning.

Jaafar Ben Khaled

Overview

01

The Challenge: Credit card fraud results in significant financial losses globally. Traditional detection methods often fall short in identifying fraudulent transactions promptly and accurately.

02

The Solution: Implement machine learning models to analyze transaction patterns and detect anomalies indicative of fraud.

03 DataSet

- Number of Transactions : 284,807
- Fraud ratio : 492 / 284 807 ≈ 0.173%

Column	Type	Description
Time	Float64	Seconds elapsed since the first transaction in the dataset.
V1-V28	Float64	Original features transformed for confidentiality and de-correlation.
Amount	Float64	Transaction amount in euros.
Class	int64	1 : Normal (non-Fraud) 0 : Fraud

```
<class 'pandas.core.frame.DataFrame'>
Index: 283726 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    283726 non-null  float64
 1   V1      283726 non-null  float64
 2   V2      283726 non-null  float64
 3   V3      283726 non-null  float64
 4   V4      283726 non-null  float64
 5   V5      283726 non-null  float64
 6   V6      283726 non-null  float64
 7   V7      283726 non-null  float64
 8   V8      283726 non-null  float64
 9   V9      283726 non-null  float64
 10  V10     283726 non-null  float64
 11  V11     283726 non-null  float64
 12  V12     283726 non-null  float64
 13  V13     283726 non-null  float64
 14  V14     283726 non-null  float64
 15  V15     283726 non-null  float64
 16  V16     283726 non-null  float64
 17  V17     283726 non-null  float64
 18  V18     283726 non-null  float64
 19  V19     283726 non-null  float64
 20  V20     283726 non-null  float64
 21  V21     283726 non-null  float64
 22  V22     283726 non-null  float64
 23  V23     283726 non-null  float64
 24  V24     283726 non-null  float64
 25  V25     283726 non-null  float64
 26  V26     283726 non-null  float64
 27  V27     283726 non-null  float64
 28  V28     283726 non-null  float64
 29  Amount  283726 non-null  float64
 30  Class   283726 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 69.3 MB
```

Five Questions We Aim to Answer:

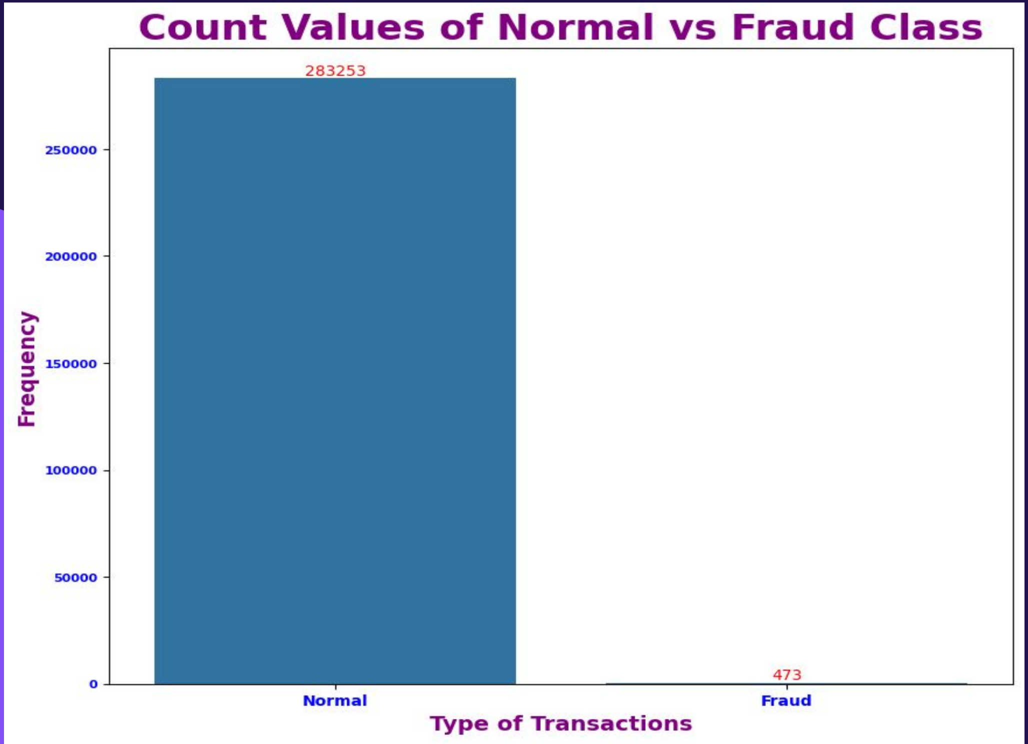
1. How can machine learning effectively detect fraudulent transactions in a large real world dataset?
2. How do the “Amount” distributions differ between fraud and normal?
3. What is the best way to handle imbalanced fraud data?
4. Why is accuracy misleading for imbalanced data, and what metrics better evaluate performance?
5. Which classification model performs best in terms of precision and recall for fraud detection?

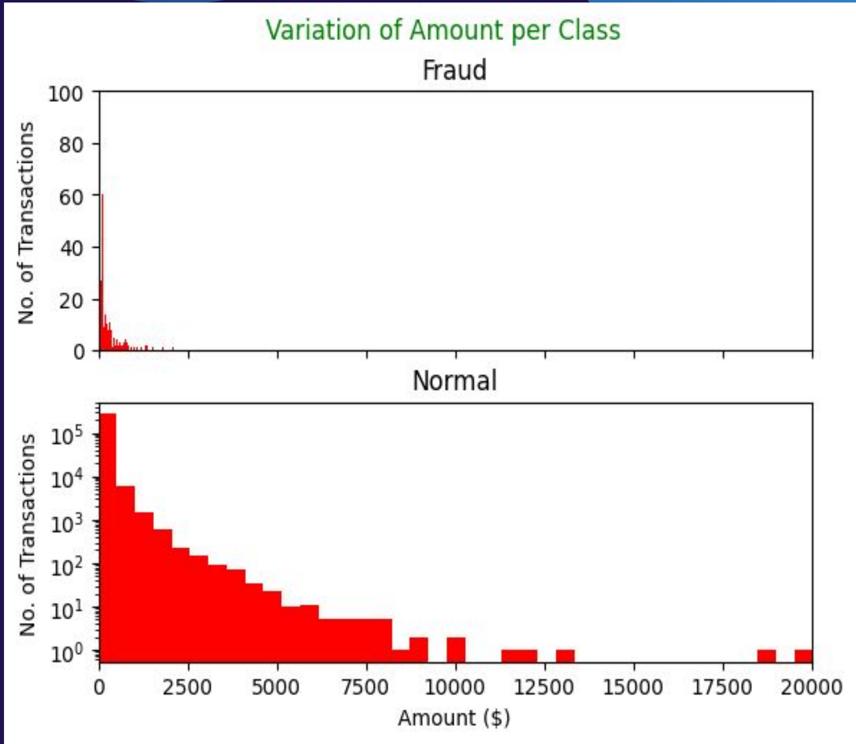
Technical Approach

1. **Import Libraries**
   - Pandas for Data Handling
   - Seaborn & Matplotlib for Visualization
   - Scikit-Learn for modeling
2. **Data Preprocessing**
   - Removes Duplicates
   - Checked for missing values
   - Scaled the “Amount” feature
   - Handling unbalanced dataset

3. **Train–Test Split**
   - Followed a 70%-30% split
   - We never touch the test set again until final evaluation, so it remains a realistic, imbalanced hold-out.

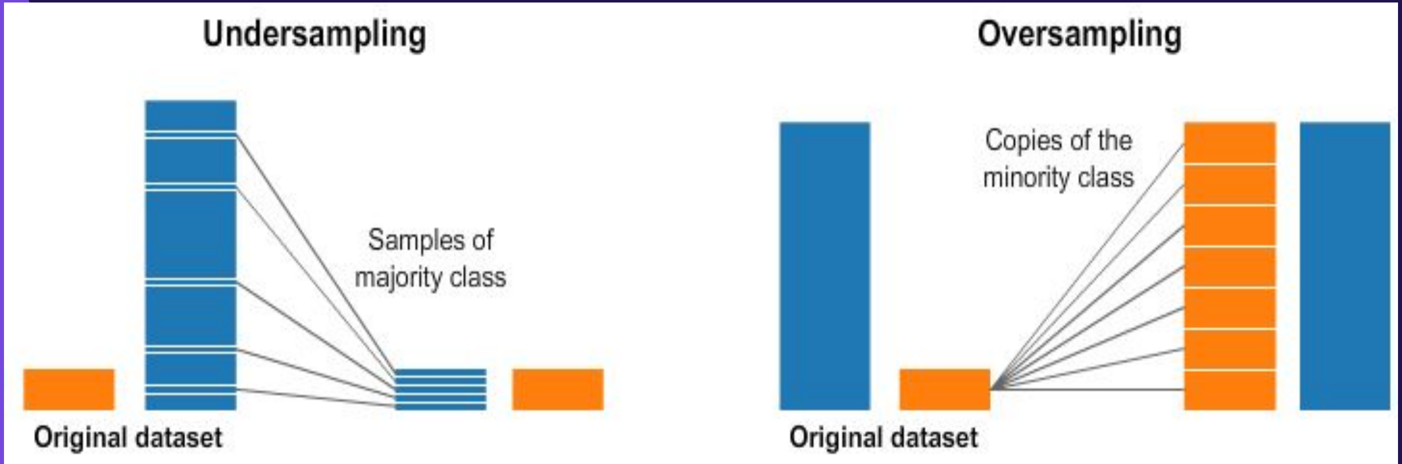
EDA & Visualization





How can we deal with highly unbalanced data?

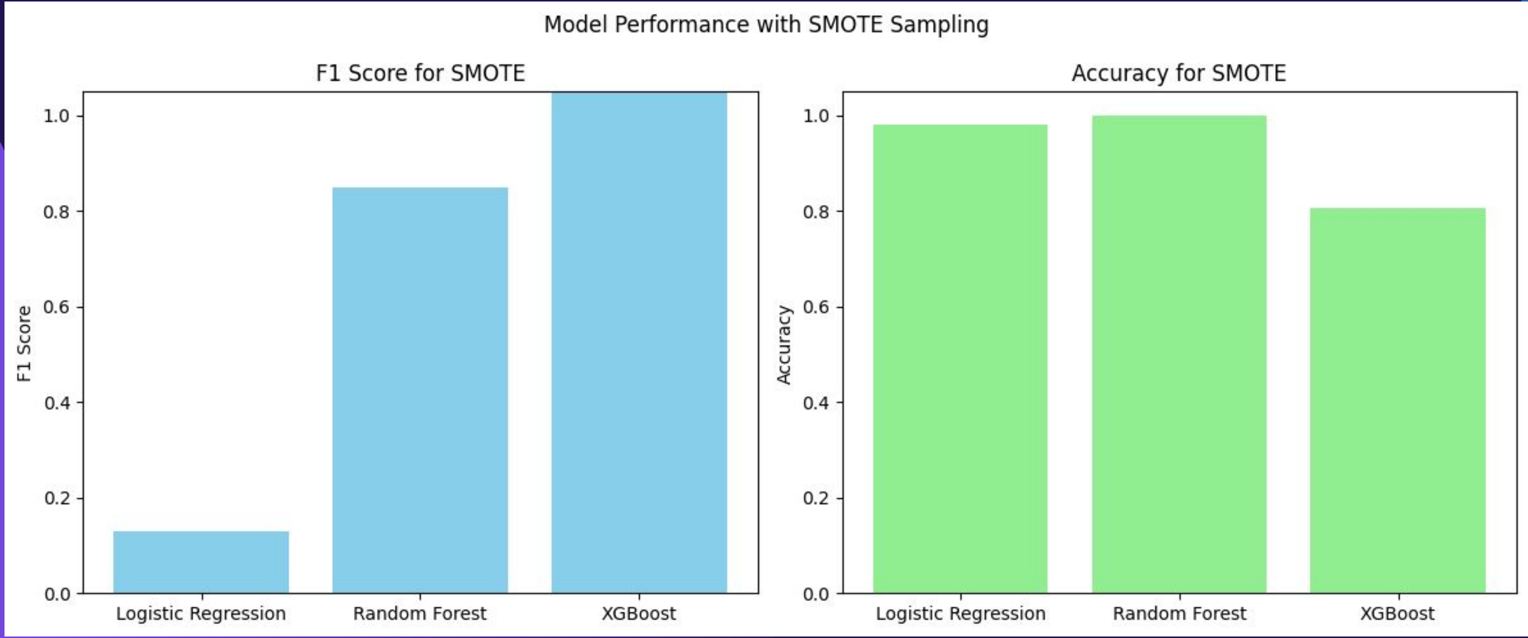
- **OverSampling the Minority Class:** We want our model to pay equal attention to fraud cases without throwing away any normal data.
- **Undersampling the Majority Class:** Balance the classes by keeping all fraud but reducing normal data so training is faster and simpler.
- **Smote:** To create *new* fraud examples instead of just copying existing ones

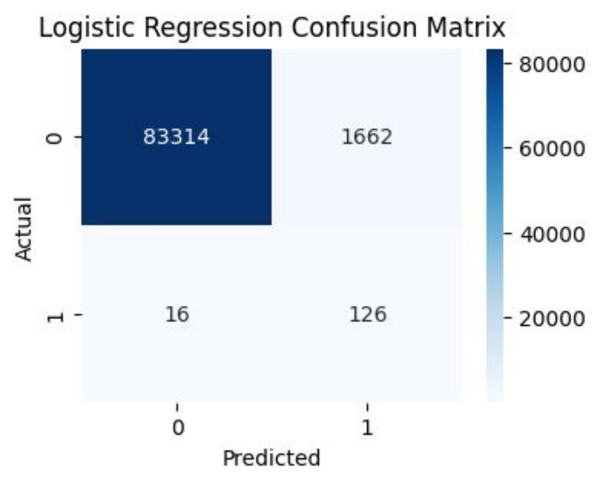


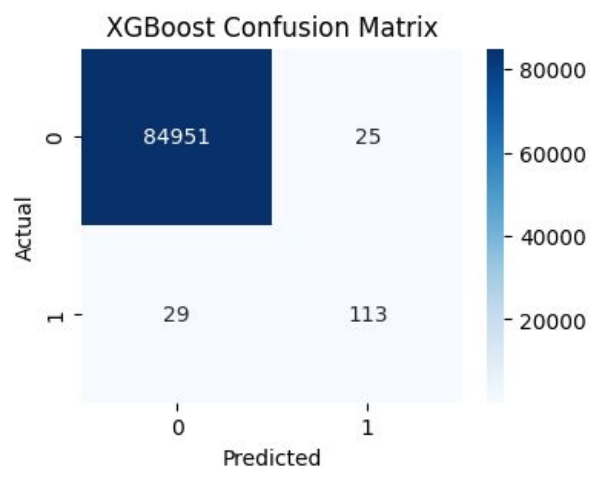
ML Models & Evaluation

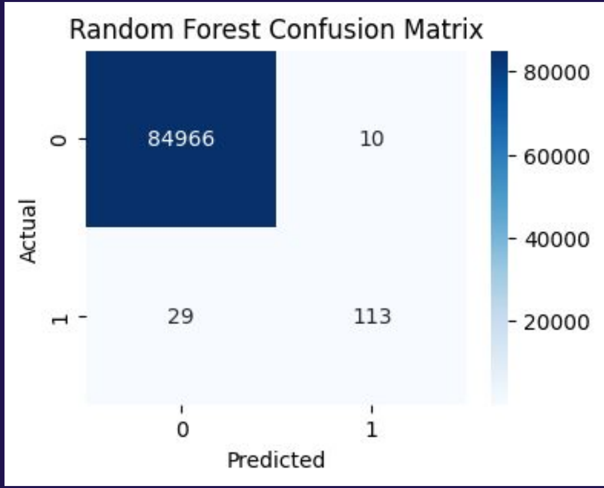
Model used :

1. Logistic Regression with
2. Random Forest with
3. XGBoost with









Logistic Regression:

- finds nearly 89% of the fraud cases (high recall)
- it only gets 7% of those fraud predictions right (very low precision)

XGBoost:

- it misses 29 frauds but makes slightly more false fraud predictions (25 instead of 10)

Random Forest:

- It only misses 29 fraud cases and makes 10 false fraud predictions.

Main Takeaways & What's Next

- **Imbalanced data** was the biggest challenge in fraud detection. First, the results were too perfect because we used SMOTE wrong (before train/test split). Fixing it gave us a realistic results.

- trying other resampling methods such as Random Oversampling or Undersampling. Then, comparing their performance using the same models to see which method gives the best results in identifying fraud.

Thank you
for your attention.