

# ***Credit Fraud Detection Report***

## **Table of Contents**

**1- About the Report**

**2- Why Are Imbalanced Datasets a Challenge in Fraud Detection?**

**3- Models and Techniques Used to Handle Class Imbalance**

**4- Analysis of All Models Across All Experiments**

**5- Best Model**

**6- side notes**

**7- Results for all models and experiments are based on the default threshold applied to the validation dataset.**

**8- Results for all models and experiments are based on the optimized threshold applied to the validation dataset.**

**9- conclusion**

# 1- About the Report

This report focuses on credit fraud detection. I will present all the techniques used to address the challenges of this imbalanced dataset(using classical machine learning techniques) and improve model performance. Each experiment is explained in detail, showing how different models performed with different strategies.

This report is designed not only to document my process and findings but also to serve as a practical guide for anyone working on similar machine learning projects, especially those dealing with imbalanced classification problems.

## 2- Why Are Imbalanced Datasets a Challenge in Fraud Detection?

Imbalanced datasets are very common in fraud detection problems, where fraud cases make up only a small percentage of all transactions. This creates several challenges:

- **Biased predictions:**  
If we train a model on the imbalanced data as it is, the model may learn to always predict the majority class (e.g., "not fraud"). This can result in high accuracy but poor fraud detection, because most fraud cases will be missed.
- **Mismatch with real-world data:**  
Resampling techniques like oversampling or undersampling change the class distribution in the training set. For example, turning 1% fraud cases into 35% changes the nature of the data(in real life). This can help the model learn to detect fraud better, but it no longer reflects the true distribution seen in real-world scenarios.
- **Performance depends on the dataset:**  
While techniques like oversampling can improve results, they don't always work well on every dataset. Their success depends on the data characteristics and the models used.

In this report, I will explain the different techniques I used to handle the imbalanced data and show how they affected the performance of various machine learning models.

### 3. Models and Techniques Used to Handle Class Imbalance

In this section, I describe the techniques and machine learning models I used to address the challenge of class imbalance in fraud detection.

#### Techniques:

To handle the imbalance between the majority (non-fraud) and minority (fraud) classes, I applied the following strategies:

1. **Undersampling** – Reducing the number of majority class samples to balance the dataset.
2. **Oversampling using SMOTE (Synthetic Minority Oversampling Technique)** – Generating synthetic examples for the minority class to increase its representation.
3. **Combined Sampling** – Applying both oversampling and undersampling to create a more balanced dataset.
4. **Cost-Sensitive Learning** – Assigning a higher cost to misclassifying fraud cases to force the model to focus more on the minority class.
5. **Using the Data As-Is** – Training the model on the original imbalanced dataset to serve as a baseline and see if the resampling techniques will improve the performance or harm it.

#### Models:

I experimented with a variety of machine learning algorithms to evaluate how well they perform with different sampling strategies:

1. **Logistic Regression**
2. **Random Forest**
3. **XGBoost (Extreme Gradient Boosting)**
4. **CatBoost**
5. **MLP (Multi-Layer Perceptron)**

## 4- Analysis of All Models Across All Experiments

For all models, RandomizedSearchCV with StratifiedKFold cross-validation was applied to select optimal hyperparameters. Each model's performance is evaluated using both default and optimized thresholds.

### 4.1 Logistic Regression

#### Default Threshold Performance:

- Raw data (with/without scaling): Model performed poorly, indicating underfitting ( $F1_{train} = 0.0682$ ,  $F1_{val} = 0.0616$ ).
- Cost-sensitive learning (with/without scaling): Similar underfitting behavior observed
- Resampling techniques: High overfitting, with a large gap between training and validation F1 scores.

#### Best Threshold Performance:

- Raw data: Significant improvement with threshold 0.99 ( $f\_score\_train = 0.7931$ ,  $f\_score\_val = 0.7932$ )
- Cost-sensitive learning: Maintained good performance with optimized threshold (like performance with raw data)
- Resampling techniques: Overfitting still happened, even after changing the threshold

#### Key Observation:

Logistic Regression performs best with raw data and improves a lot when the threshold is tuned. Resampling methods usually cause overfitting, so they're not a good choice for this model.

### 4.2 Random Forest Classifier

#### Default Threshold Performance:

- Raw data (with/without scaling): Best performance achieved with no overfitting or underfitting
- Resampling techniques: High overfitting, with a large gap between training and validation F1 scores.

### Best Threshold Performance:

- Raw data: Performance did not improve with optimized threshold (0.336), and the optimized threshold indicates lower model confidence
- Resampling techniques: Overfitting still happened, even after changing the threshold, and the gap between training and validation F1\_score increased, except for cost-sensitive learning, where gaps remained constant
- Overall: The Default threshold generally outperformed the optimized threshold

### Key Observation:

Random Forest achieves optimal performance with raw, unprocessed data using the default threshold. Increasing (**min\_samples\_leaf**) and setting (**class\_weight = None**) helps to prevent **overfitting** by limiting tree complexity and also using regularization techniques.

## 4.3 XGBoost and CatBoost Classifiers

### Default and Best Threshold Performance:

- All experiments: Both models showed consistent overfitting, regardless of whether scaling or regularization techniques were applied.
- Scaling effect: No noticeable improvement in performance.
- Threshold optimization: Did not reduce overfitting.

### Key Observation:

XGBoost and CatBoost consistently overfit in all scenarios. Since they degraded the performance, they were removed from the final voting classifier.

## 4.4 Multi-Layer Perceptron (MLP) Classifier

### Default Threshold Performance:

- Raw data with scaling: Delivered the best results among all individual models.
- Raw data without scaling: Performed poorly, confirming the need for scaling.
- Resampling techniques and cost-sensitive learning: Led to overfitting.
- Early stopping: Helped reduce overfitting when tuned correctly.

### **Best Threshold Performance:**

- Optimized threshold (0.8712): Results were similar to the default threshold.
- Confidence: Model showed higher prediction confidence at the optimized threshold.

### **Key Observation:**

The MLP Classifier was the top-performing individual model, working best with scaled data. Early stopping improves generalization, and threshold tuning boosts confidence without changing overall performance.

## **4.5 Voting Classifier**

### **Configuration 1 – All Models**

- Models included: Logistic Regression, Random Forest, XGBoost, CatBoost, MLP (scaled data).
- Performance: Overfitted in all experiments.
- Root cause:
  - XGBoost and CatBoost consistently overfitted in all settings, which negatively impacted the ensemble.
  - Resampling techniques (undersampling, oversampling, oversampling and undersampling) further increased overfitting since most models also overfitted under these methods.

### **Configuration 2 – Selected Models**

- Models included: Logistic Regression, Random Forest, MLP (scaled data).
- Best performance:
  - Raw data: Achieved best results with optimized threshold (0.7861).
  - Scores:  $f\_score\_train = 0.84$ ,  $f\_score\_val = 0.82$ .
- Resampling techniques: still overfitting.
- Other experiments: Performed well with raw data (scaled/unscaled) and cost-sensitive learning (scaled/unscaled).

**Key Observation:**

The best ensemble setup used the Voting Classifier with Logistic Regression, Random Forest, and MLP on raw scaled data. This configuration balanced performance and generalization, while avoiding the overfitting issues introduced by XGBoost and CatBoost.

**5- Best Model****Criteria Considered**

In real-world projects, the “best” model depends on the business’s priorities—such as agreed performance targets and acceptable error trade-offs. In this project, my primary selection metric was the F1-score.

Since two models achieved the same F1-score, I applied additional considerations :

- Cost of Missing Fraud vs Cost of False Alarms
- Customer Experience (too many blocks = angry customers)
- Investigation Capacity (Can your team handle 1000 alerts/day?)

**Top two models**

Two models stood out during evaluation:

- Random Forest (raw data or raw data with scaling)
- MLPClassifier (scaled data)

Random Forest results at default threshold

- f1\_score\_val = 0.8363

- pr\_auc\_val = 0.8525

MLPClassifier results at default threshold

- f1\_score\_val = 0.8323

- pr\_auc\_val = 0.7798

and also almost the same result for best threshold

Both models achieved almost the same F1 score at the default threshold. The only difference is that the Random Forest model had a higher PR-AUC. So, the choice of which model to use depends on your criteria — you can try both and select the one that performs best for your needs..

## MLPClassifier Results

Default Threshold (0.5):

- F1\_train = 0.8301
- F1\_val = 0.8323

train classification report for default threshold :

	precision	recall	f1-score	support
0	0.9996	0.9998	0.9997	170579
1	0.8910	0.7770	0.8301	305
accuracy			0.9994	170884
macro avg	0.9453	0.8884	0.9149	170884
weighted avg	0.9994	0.9994	0.9994	170884

val classification report for default threshold :

	precision	recall	f1-score	support
0	0.9997	0.9998	0.9997	56870
1	0.8675	0.8000	0.8324	90
accuracy			0.9995	56960
macro avg	0.9336	0.8999	0.9161	56960
weighted avg	0.9995	0.9995	0.9995	56960



### Optimized Threshold (0.8712):

- F1\_train = 0.8389
- F1\_val = 0.8353

### train classification report for the best threshold :

	precision	recall	f1-score	support
0	0.9996	0.9999	0.9997	170579
1	0.9115	0.7770	0.8389	305
accuracy			0.9995	170884
macro avg	0.9556	0.8885	0.9193	170884
weighted avg	0.9994	0.9995	0.9994	170884

### val classification report for the best threshold :

	precision	recall	f1-score	support
0	0.9997	0.9998	0.9998	56870
1	0.8875	0.7889	0.8353	90
accuracy			0.9995	56960
macro avg	0.9436	0.8944	0.9175	56960
weighted avg	0.9995	0.9995	0.9995	56960

## Random Forest Results

### Default Threshold (0.5):

- F1\_train = 0.8369
- F1\_val = 0.8363

### train classification report for default threshold :

	precision	recall	f1-score	support
0	0.9996	0.9999	0.9997	170579
1	0.9352	0.7574	0.8370	305
accuracy			0.9995	170884
macro avg	0.9674	0.8786	0.9183	170884
weighted avg	0.9995	0.9995	0.9994	170884

**val classification report for default threshold :**

	precision	recall	f1-score	support
0	0.9996	0.9999	0.9998	56870
1	0.9200	0.7667	0.8364	90
accuracy			0.9995	56960
macro avg	0.9598	0.8833	0.9181	56960
weighted avg	0.9995	0.9995	0.9995	56960

**train classification report for the best threshold :**

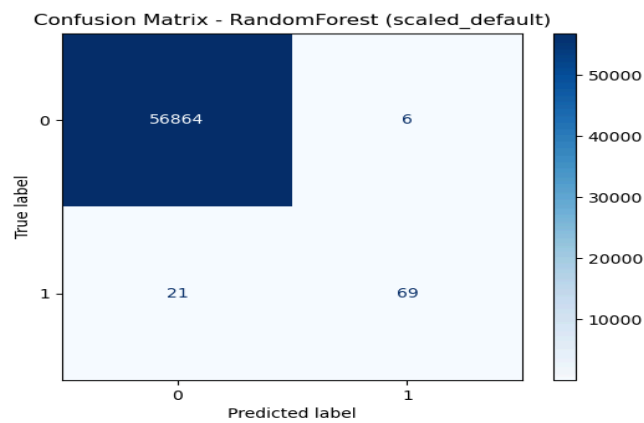
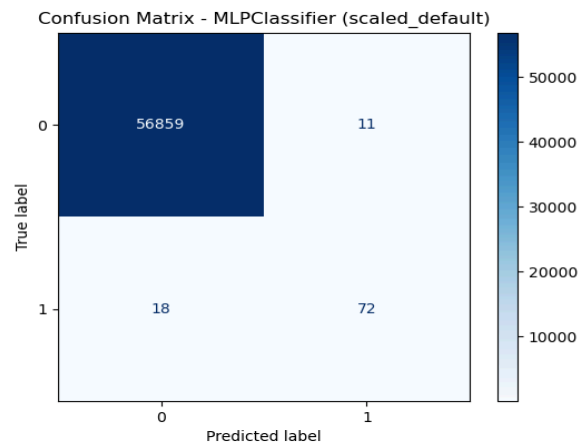
	precision	recall	f1-score	support
0	0.9996	0.9999	0.9998	170579
1	0.9245	0.8033	0.8596	305
accuracy			0.9995	170884
macro avg	0.9621	0.9016	0.9297	170884
weighted avg	0.9995	0.9995	0.9995	170884

**val classification report for the best threshold :**

	precision	recall	f1-score	support
0	0.9997	0.9998	0.9998	56870
1	0.8875	0.7889	0.8353	90
accuracy			0.9995	56960
macro avg	0.9436	0.8944	0.9175	56960
weighted avg	0.9995	0.9995	0.9995	56960

**which model i should use?**

**Now, let's take another point of view and look at the confusion matrix.**



MLP Classifier:

- True Positives (TP): 72
- False Positives (FP): 11
- False Negatives (FN): 18

Random Forest:

- True Positives (TP): 69
- False Positives (FP): 6
- False Negatives (FN): 21

Choose MLP if:

- Fraud losses are extremely costly
- You can handle customer complaints from false blocks
- Investigation capacity is high

Choose Random Forest if:

- Customer experience is critical
- Investigation resources are limited
- False alarms are expensive to resolve

## Conclusion

In the end, it's a trade-off. The decision should be made by looking at the problem from multiple perspectives, not just one.

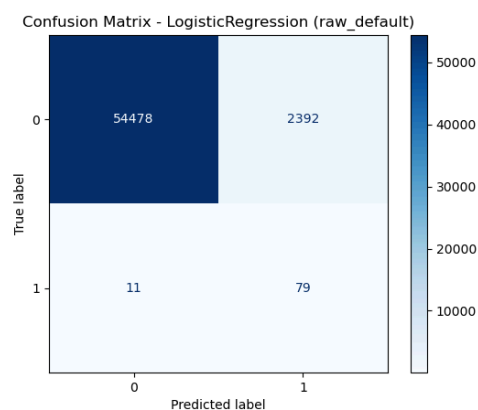
## 6- side notes

The best threshold greatly improved Logistic Regression's performance across all experiments and reduced false positive alarms.

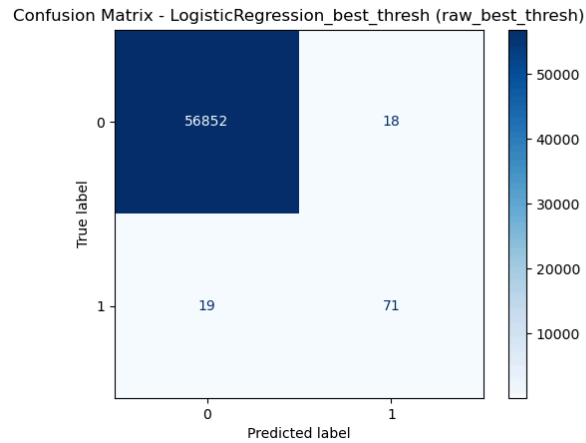
### 1- logistics regression with raw data :

model	f1_train	f1_val	f1_train_a	f1_val_at	pr_auc_va	best_threshold
LogisticRe	0.068293	0.061695	0.793103	0.793296	0.717582	0.999988

- confusion matrix for default threshold (0.5) :



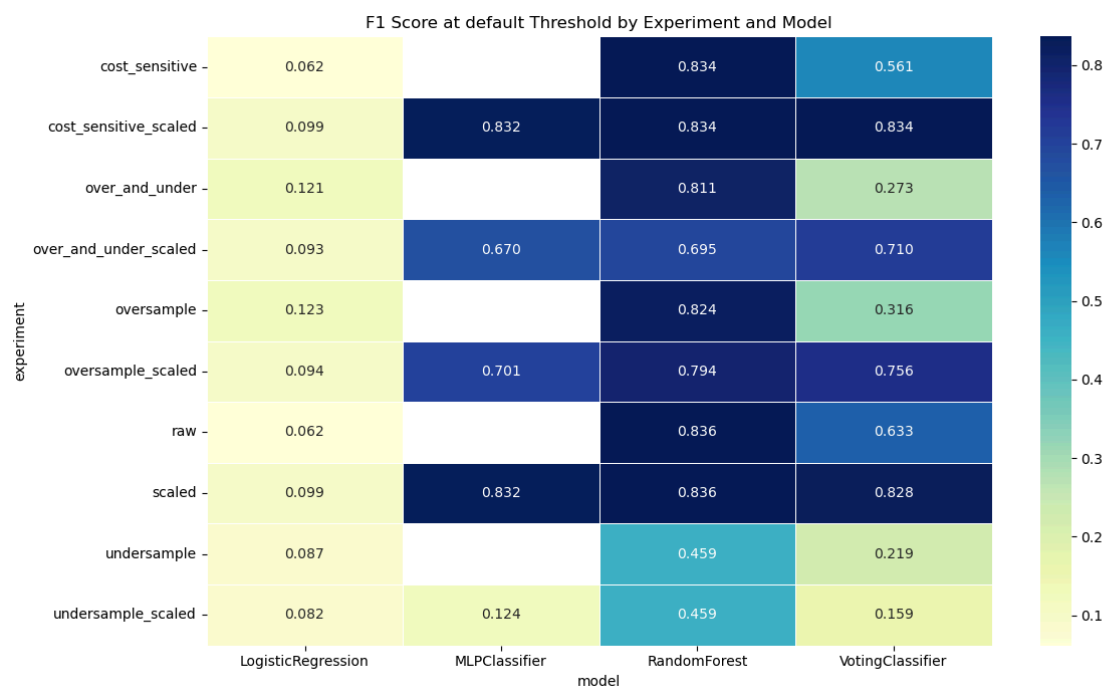
- confusion matrix for default threshold (0.999) :



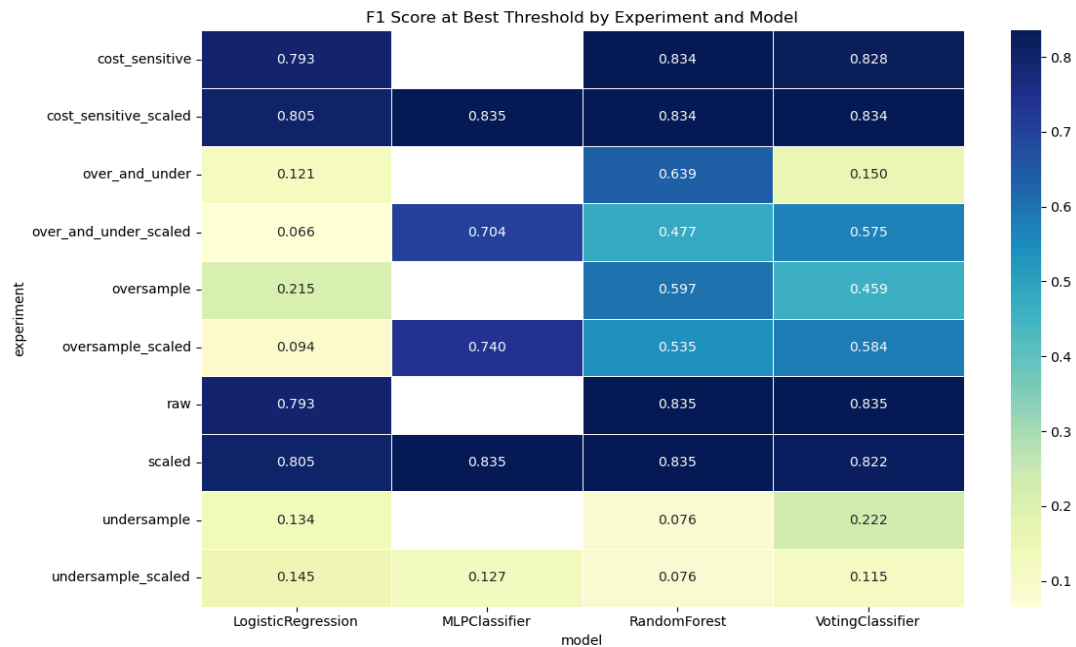
2- For scaling, I tried all the scalers and found little to no difference in performance. The only noticeable impact was on the MLPClassifier, where using a scaler increased the PR-AUC by about 2% and slightly affected the best threshold. For the Random Forest model, there was no change.

So, I decided to use StandardScaler, but you can also use RobustScaler if you prefer.

7- Results for all models and experiments are based on the default threshold applied to the validation dataset.



**8- Results for all models and experiments are based on the optimized threshold applied to the validation dataset.**



## 9- Conclusion

**1- Sampling techniques are useless in this project, and they harm the model performance more than improve it, especially (undersampling)**

**2- You should apply (RandomizedSearchCV) to determine the best parameters for your models. This will help you get the best performance**

