Team :

Yassin Ehab —> 20216117

Radwa Belal —> 20217005

Menna Elminshawy —> 20217011

TA :Mira

Credit Card Fraud detection:

Introduction:

One of the important tasks is to detect if there’s any fraud transactions happening, and it’s too important to detect it in early stages before any damage occurs.

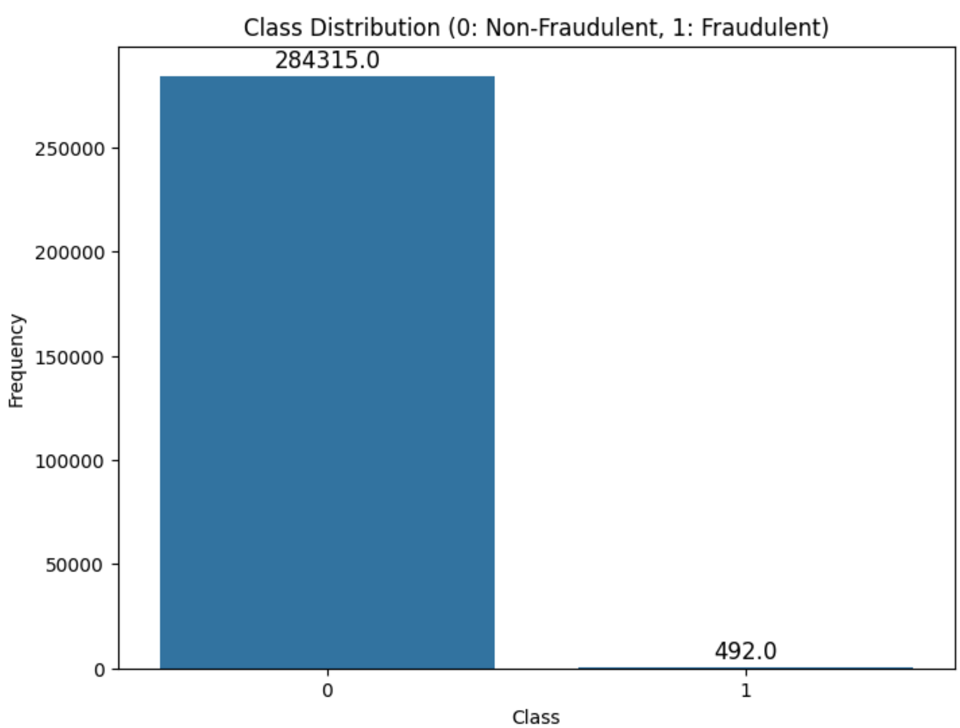
Objective:

The objective of this assignment is to develop a credit card fraud detection system using machine learning techniques. Additionally, you will explore various balancing techniques to handle the highly unbalanced nature of the dataset and compare their impact on the classification performance

Methodology:

1- First of all, starting with exploring the dataset. We found out that there’s no missing values, 0 nulls.

2- Seeing the difference between the 2 classes: fraudulent and non-fraud and here’s the result:



To fix this imbalancing problem we used different techniques including the following:

- Random Oversampling
- SMOTE
- Random Undersampling
- Class Weights

3- We used logistic regression to classify whether the row/object is a normal transaction or fraudulent.

```
print("\n--- Training on Original Unbalanced Data ---")
model_orig = LogisticRegression(random_state=42, max_iter=1000)
model_orig.fit(X_train, y_train)


y_pred_orig = model_orig.predict(X_test)

print("\nEvaluation Metrics (Original Data):")
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_orig))
print("\nClassification Report:\n", classification_report(y_test, y_pred_orig,
target_names=['Non-Fraud (0)', 'Fraud (1)']))
print(f"Accuracy: {accuracy_score(y_test, y_pred_orig):.4f}")
print(f"Precision (Fraud): {precision_score(y_test, y_pred_orig):.4f}")
print(f"Recall (Fraud): {recall_score(y_test, y_pred_orig):.4f}")
print(f"F1-Score (Fraud): {f1_score(y_test, y_pred_orig):.4f}")
```

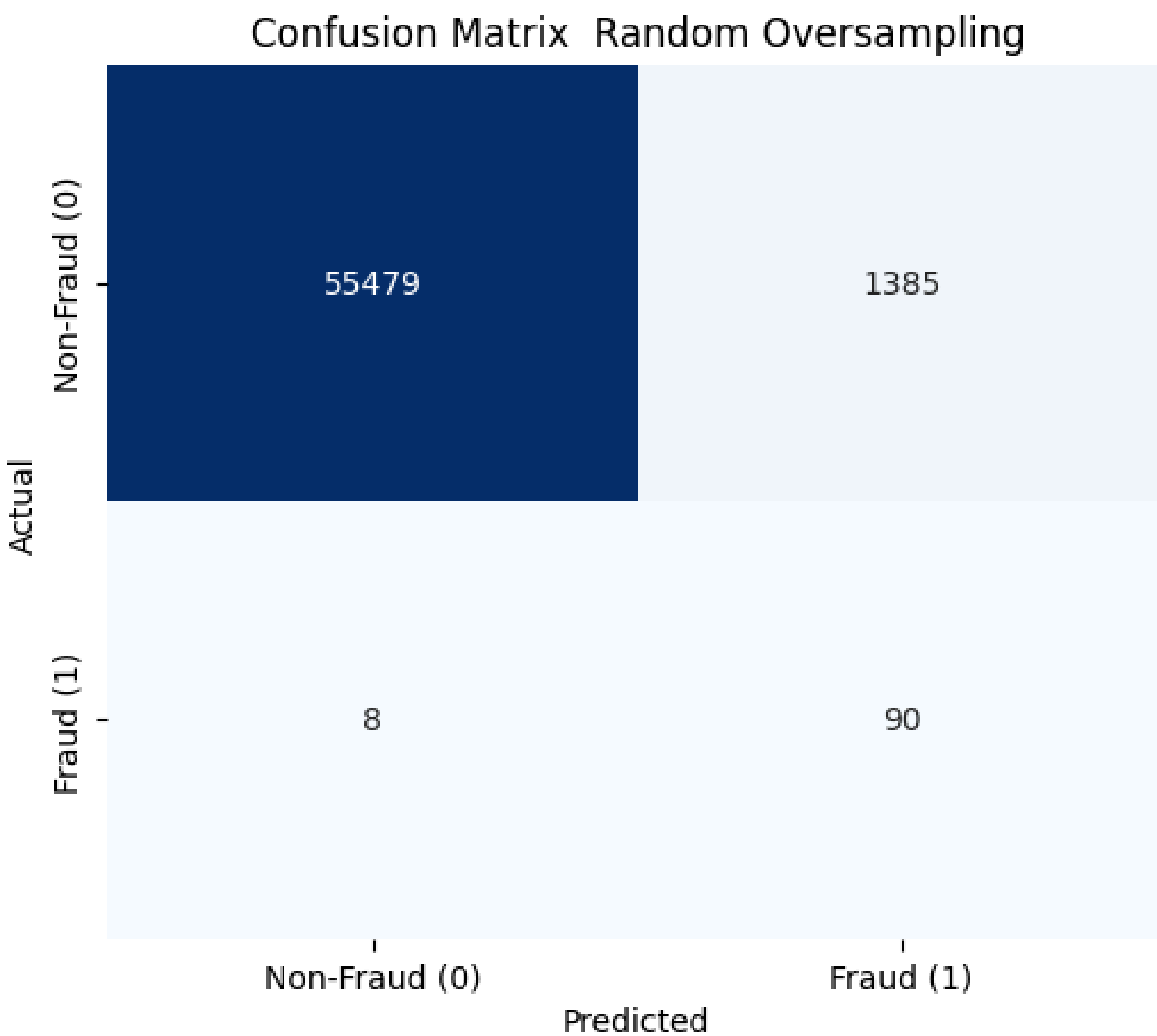
And it gave us accuracy: (this result is without using any balancing techniques)

```
Accuracy: 0.9992
Precision (Fraud): 0.8289
Recall (Fraud): 0.6429
F1-Score (Fraud): 0.7241
```

And this result is after using one of the techniques: (oversampling)

Precision dropped because oversampling **shifted the model's focus to catching fraud**, increasing false positives.

```
Accuracy: 0.9755
Precision (Fraud): 0.0610
Recall (Fraud): 0.9184
F1-Score (Fraud): 0.1144
```



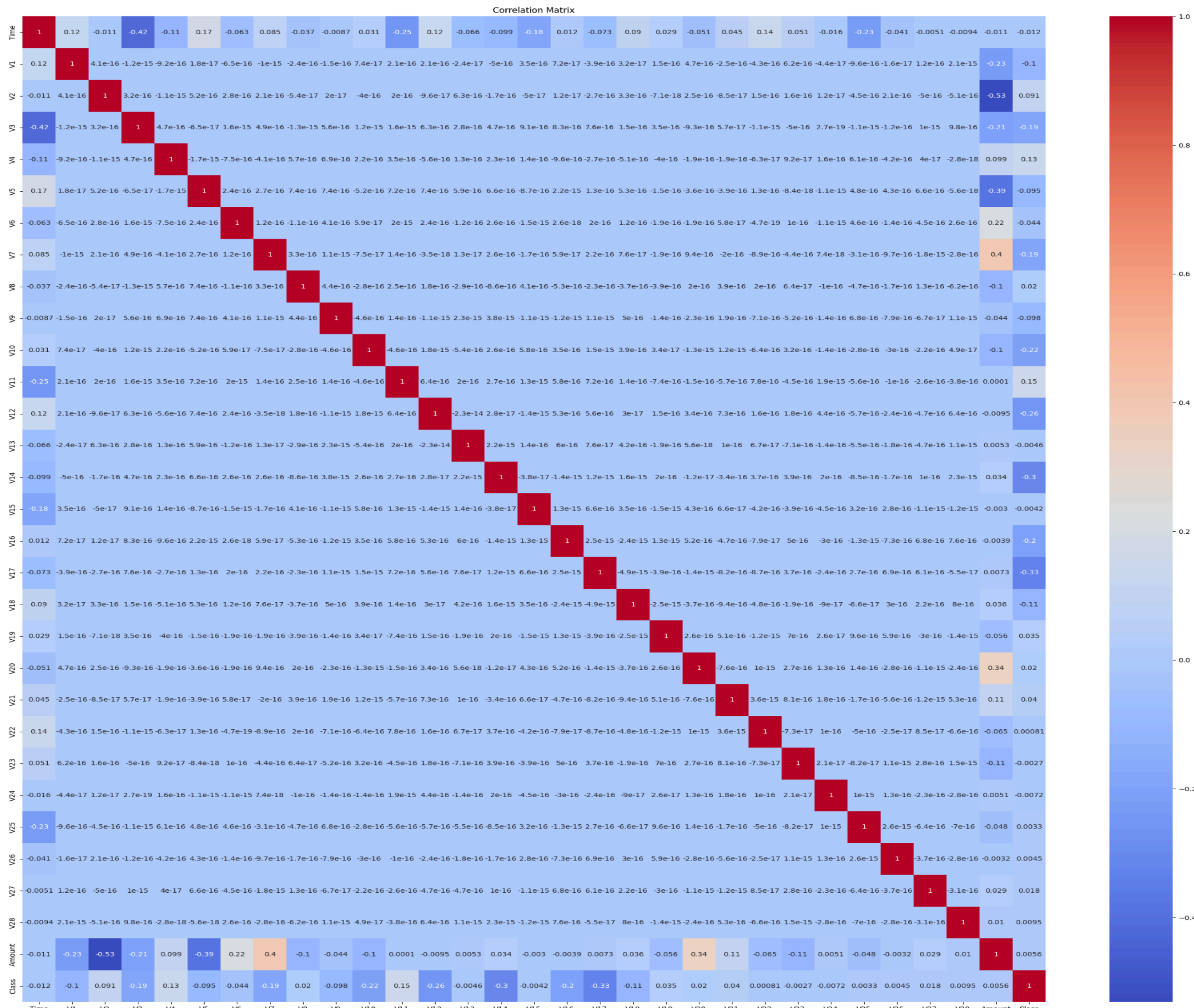
Comparison between imbalancing handling techniques:

Name of tech	Key advantages	Key limitations	Ideal use case
Oversampling	Improves recall	Overfitting risks	Small imbalanced datasets
Undersampling	Faster training	Loss of majority-class info	Large datasets
SMOTE	No data loss, better generalization	Noisy synthetic samples	Moderate imbalance
Class weights	No data modification	Less effective for extreme imbalance	Models supporting weighted loss
Threshold	Quick to implement	Doesn't fix data imbalance	Fine-tuning deployed models for business needs (e.g., fraud detection favoring recall).

Metric	**Original**	**Random Oversampling**	**SMOTE**	**Random Undersampling**	**Class Weights**	**Threshold = 0.2**
Accuracy	**0.999157**	**0.975545**	**0.974211**	**0.960324**	**0.975528**	**0.902777**
Precision (Fraud)	**0.828947**	**0.061017**	**0.058027**	**0.038429**	**0.060976**	**0.016358**
Recall (Fraud)	**0.642857**	**0.918367**	**0.918367**	**0.918367**	**0.918367**	**0.938776**
F1-Score (Fraud)	**0.724138**	**0.114431**	**0.109157**	**0.073770**	**0.114358**	**0.032157**

Exploratory data analysis

Correlation matrix



Splitting the Data:

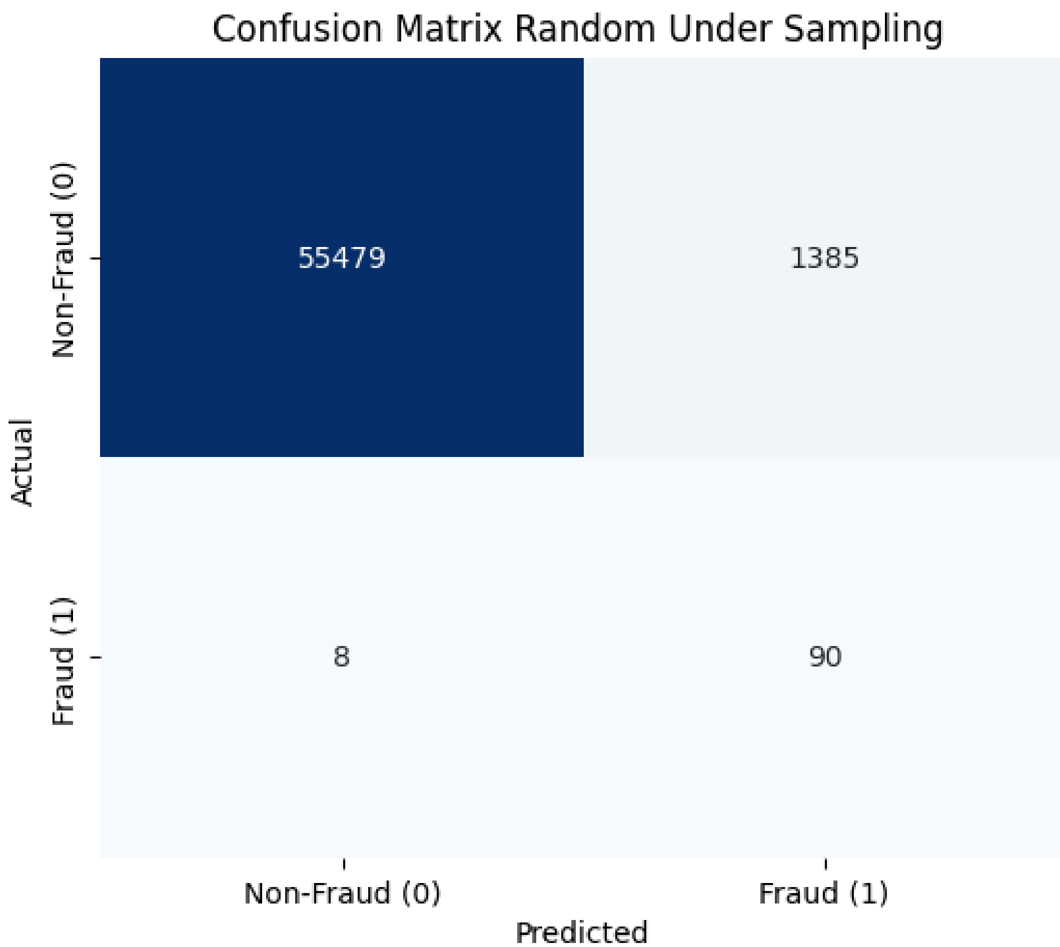
```
ta Splitting:
Shape of X_train: (227845, 30)
Shape of X_test: (56962, 30)
Shape of y_train: (227845,)
Shape of y_test: (56962,)

Class distribution in y_train:
Class
0    0.998271
1    0.001729
Name: proportion, dtype: float64

Class distribution in y_test:
Class
0    0.99828
1    0.00172
```

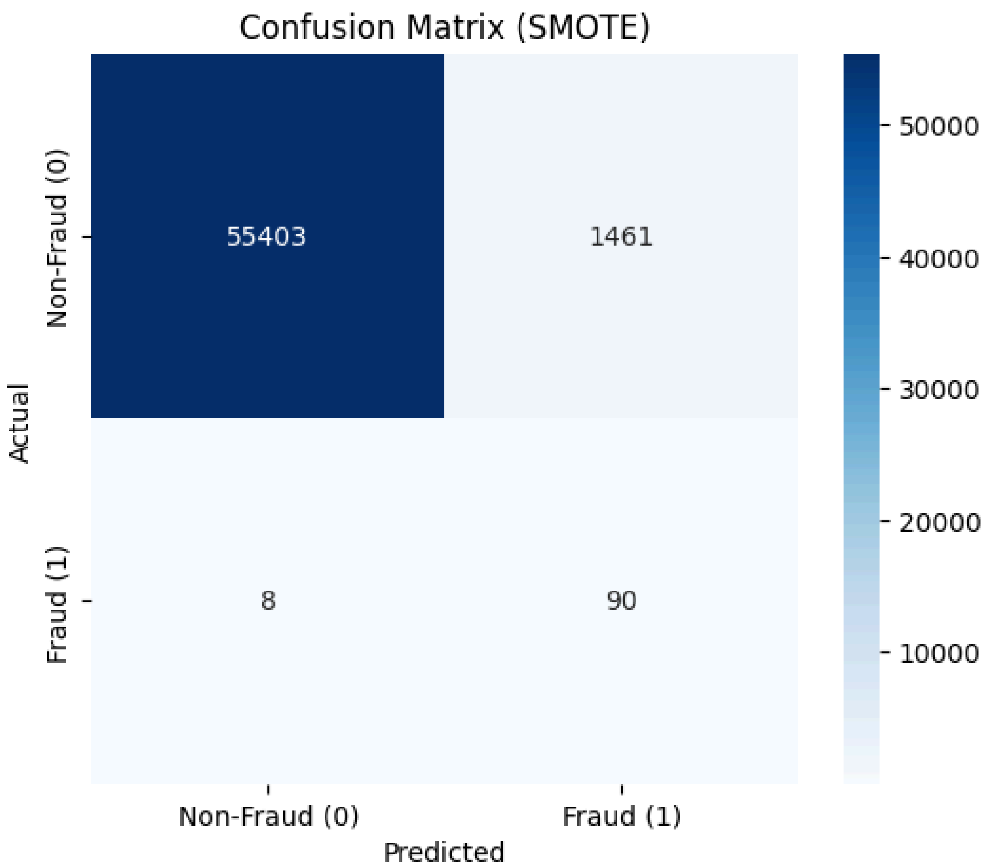
Applying Random Under Sampling

```
Accuracy: 0.9603
Precision (Fraud): 0.0384
Recall (Fraud): 0.9184
F1-Score (Fraud): 0.0738
```



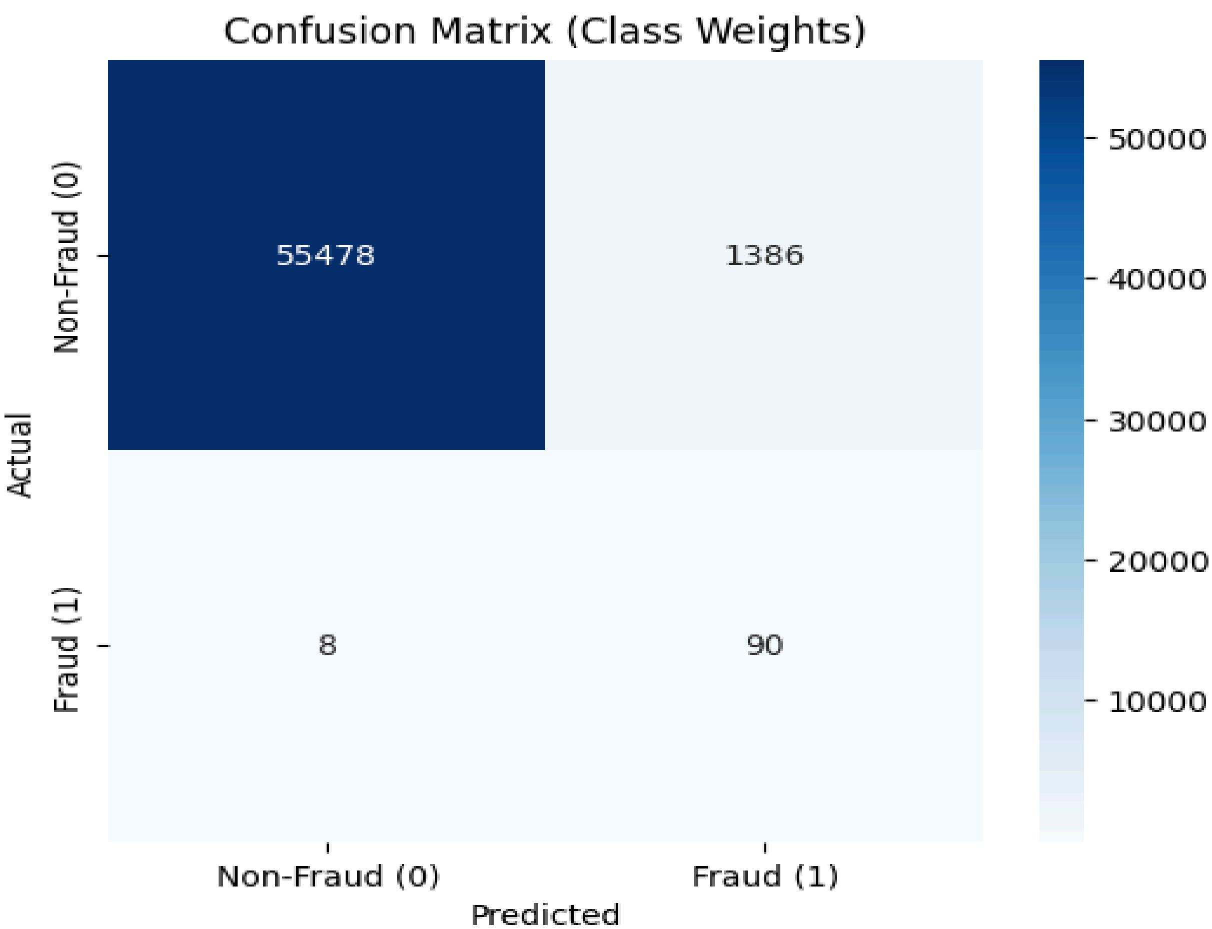
Applying SMOTE

```
Accuracy: 0.9742
Precision (Fraud): 0.0580
Recall (Fraud): 0.9184
F1-Score (Fraud): 0.1092
```



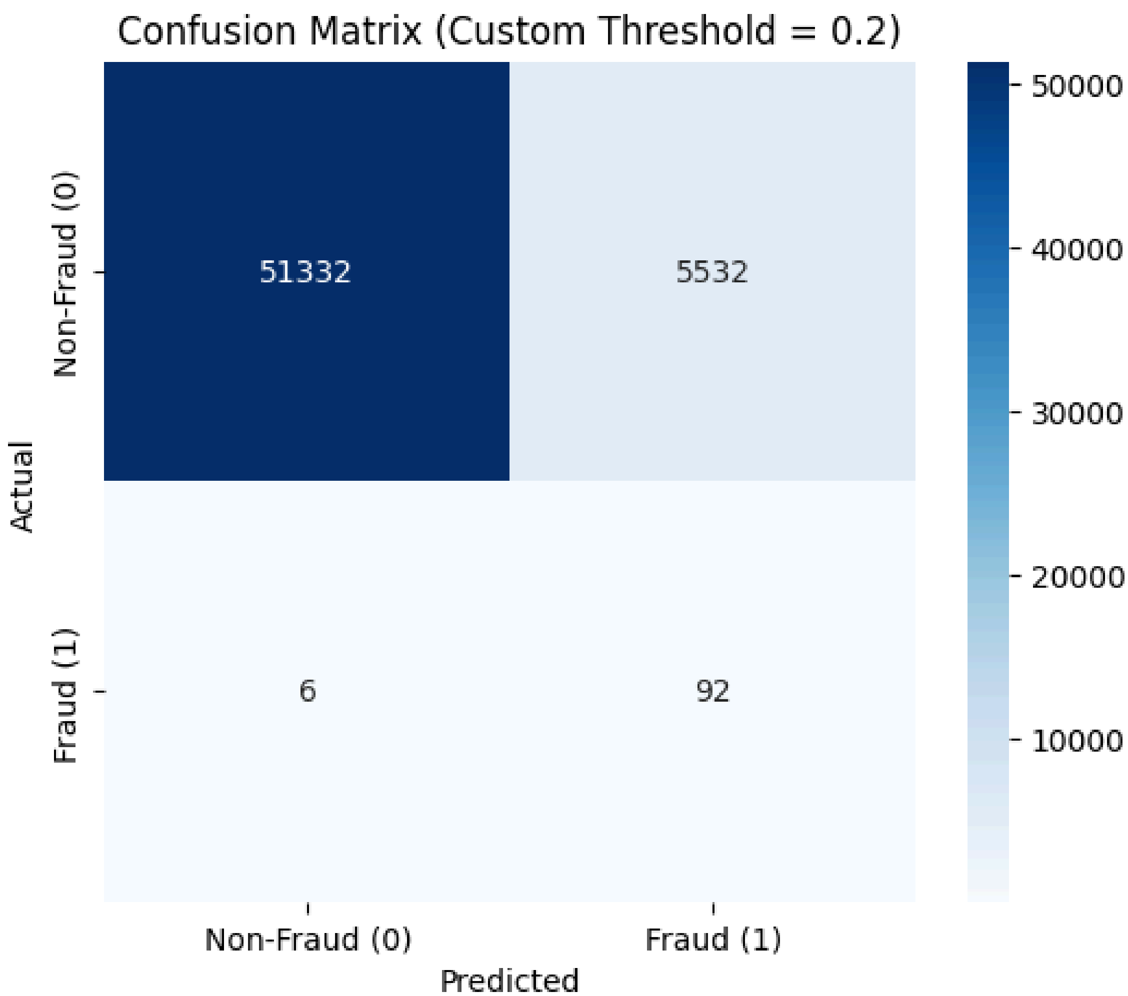
Class Weights

```
Accuracy: 0.9755
Precision (Fraud): 0.0610
Recall (Fraud): 0.9184
F1-Score (Fraud): 0.1144
```



Threshold = 0.2

```
Accuracy: 0.9028
Precision (Fraud): 0.0164
Recall (Fraud): 0.9388
F1-Score (Fraud): 0.0322
```



Conclusion:

This project focused on detecting credit card fraud using logistic regression and various techniques to handle class imbalance. The original unbalanced model had high accuracy (99.92%) and precision (82.89%) but low recall (64.29%), meaning many fraud cases were missed.

Balancing methods like Oversampling, SMOTE, Undersampling, Class Weights, and Threshold adjustment significantly improved recall (over 91%) but reduced precision, causing more false positives.

Among all, class weights and SMOTE offered a decent trade-off, though still with low precision.

Threshold tuning (0.2) gave the highest recall (93.88%) but the lowest precision.

Key Takeaway:

There's always a trade-off between recall and precision. In fraud detection, high recall is often prioritized to catch more fraud, even at the cost of some false alarms. SMOTE may not be ideal for real-world use due to synthetic data risks. Threshold tuning and class weighting are practical options in real systems.