Report on Fraud Detection Model Performance

---

1. The Problem with Accuracy

WHY PRECISION and RECALL are ROBUST MEASURES

Accuracy is a misleading metric for imbalanced datasets because the majority class overwhelmingly skews the result. In the dataset, non-fraudulent transactions make up 99.83% of the data. A ‘lazy’ model that predicts "not fraud" for every single transaction would achieve 99.83% accuracy. While this sounds impressive, the model would be completely useless as it would fail to identify a single fraudulent case, which is the entire point of the exercise.

Precision and recall are robust because they focus on the performance of the minority class (fraud), which is the class we are most concerned with.

- **Precision** answers: "Of all the transactions we flagged as fraud, how many were actually fraudulent?" It is not affected by the large number of correctly identified non-fraudulent transactions (True Negatives).
- **Recall** answers: "Of all the actual fraudulent transactions, how many did we successfully flag?" It is only concerned with how many frauds we caught out of the total that existed.

---

2. How SMOTE Works and Its Limitations

SMOTE (Synthetic Minority Over-sampling Technique) generates "synthetic" data points for the minority class, rather than simply duplicating existing ones. For each fraudulent transaction, it:

1. Finds its *k*-nearest neighbours (other nearby fraudulent transactions).
2. Randomly selects one of those neighbours.
3. Creates a new, synthetic data point at a random spot along the line segment connecting the original transaction and its chosen neighbour.

The Limitation of SMOTE is that it can create noise

It does not consider the distribution of the majority class when creating new samples. This can lead to the creation of synthetic samples in regions that overlap with the majority class, blurring the decision boundary.

However, in this case, the hyperparameter tuning was highly effective at mitigating this issue. The GridSearchCV selected a model that was regularised enough to learn the patterns from the synthetic data without overfitting.

---

3. Concept of Clustering for Oversampling (CBO)

Clustering-Based Oversampling is a more intelligent way to oversample. Instead of treating all minority samples as one group (like SMOTE), it first tries to find underlying sub-groups within them. The process is:

1. Take only the minority class data (the fraudulent transactions).
2. Use a clustering algorithm like K-Means to group these fraudulent transactions into several distinct clusters. I used the elbow method to determine the appropriate number of clusters.
3. Perform oversampling within each cluster separately.

This ensures that the oversampling respects the natural structure of the minority class. If there are different "types" of fraud, each type gets boosted equally, creating a more representative and higher-quality synthetic dataset.

---

4. Concept of Clustering for Undersampling (CBU)

Clustering-Based Undersampling applies the same logic to the majority class to reduce information loss. The process is:

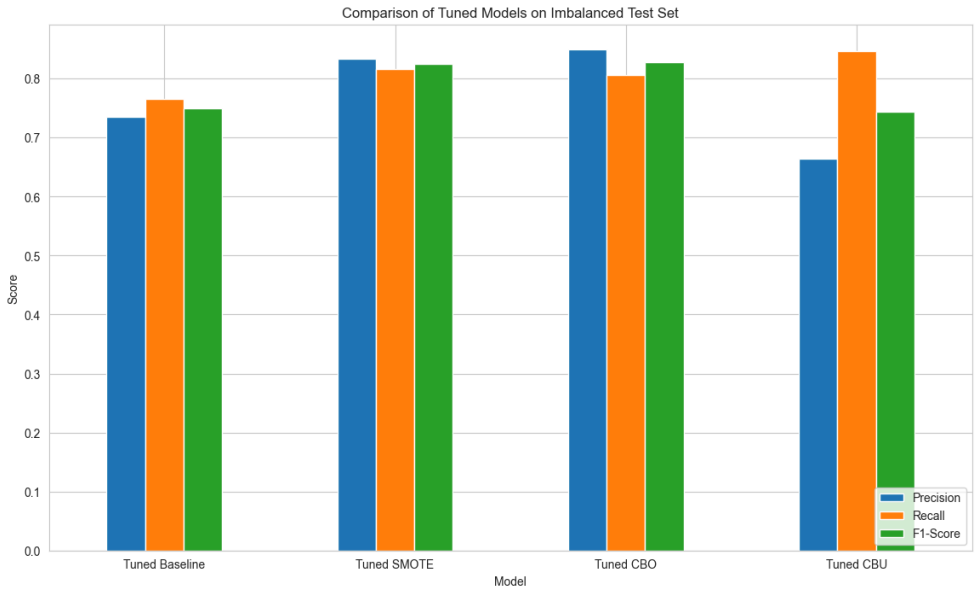
1. Take only the non-fraudulent transactions.
2. Use K-Means to group them into clusters (I used **k=4** based on the elbow plot).
3. Remove samples from each cluster proportionally until the number of non-fraudulent samples matches the fraudulent ones.

This creates a smaller but more diverse sample of the majority class compared to just random removal.

---

5. Benefits, Drawbacks, and Best Performing Method

Model	(Precision, Recall, F1-Score)
Tuned Baseline	0.7353, 0.7653, 0.7500
Tuned CBO	0.8495, 0.8061, 0.8272
Tuned SMOTE	0.8333, 0.8163, 0.8247
Tuned CBU	0.6640, 0.8469, 0.7444



The best-performing method is the **Tuned CBO (Clustering-Based Oversampling**) model.

- Why it performed best:

  The CBO model achieved the highest F1-Score (0.8272), indicating the best balance between precision and recall. It also had the highest precision (0.8495), meaning it produced the fewest false alarms among the resampling methods.

- Benefits & Drawbacks:
  - CBO/SMOTE: Their major benefit is clear—they dramatically improved the model's performance over the baseline by providing a balanced dataset, leading to F1-scores above 0.82.
  - CBU: Its drawback is evident in its low precision (0.6640) and an F1-score lower than the baseline. By throwing away most of the majority class data, the model lost too much information to build a reliable profile of non-fraudulent transactions, even with clustering.

---

6. How Clustering Addresses SMOTE's Limitations

In the results, CBO slightly outperformed SMOTE. While SMOTE generates samples anywhere between neighbours, CBO only generates samples *within* the identified fraud clusters.

This structured approach creates a cleaner, more representative set of synthetic data. This resulted in the **CBO model achieving a higher precision (0.8495 vs. 0.8333)** than the SMOTE model, as it was less confused by potentially "noisy" synthetic samples.

---

7. The Necessity of Threshold Tuning

Even though your models performed very well "out of the box," threshold tuning is a crucial final step for optimisation.

What is being done: The models were trained on 50/50 balanced data, so they used a default probability threshold of 0.5 to decide "fraud" vs. "not fraud." The test set, however, is not 50/50. Threshold tuning analyses the model's raw probability scores (`.predict_proba()`) and finds

a new, optimal threshold that maximises the F1-score specifically for the real-world, imbalanced test data.

Why it is necessary: It calibrates the model's sensitivity

For CBO and SMOTE Models:

The "Optimal Threshold" is listed as 1.0000, and more importantly, the F1-Scores are identical to the scores before calibration (CBO: 0.8272, SMOTE: 0.8247).
This means that the hyperparameter tuning was so effective that the models' default 0.5 threshold was already the optimal threshold for maximising the F1-score.

For the CBU Model:

This is where thresholding shows its true value. Before calibration, the CBU model had an F1-score of 0.7444. After finding an optimal threshold of 0.8261, its performance jumped significantly to an F1-score of 0.7788.