DA5401 A3: Addressing Class Imbalance with Clustering and Resampling

Objective: This assignment aims to challenge your understanding of class imbalance, unsupervised learning (clustering), and its application in improving the performance of a supervised classification model. You will use clustering to create a more representative training set for both the minority and majority classes through oversampling and undersampling, and assess the impact on a Logistic Regression classifier.

---

1. Problem Statement

You are working for a fraud detection company. You've been given a highly imbalanced credit card transaction dataset where a very small fraction of transactions are fraudulent (the minority class). Standard classification algorithms often perform poorly on such datasets, as they tend to classify most instances as the majority class to achieve high overall accuracy, ignoring the minority class.

Your task is to utilize **clustering-based oversampling** and **clustering-based undersampling** to create a more representative and improved training sample. You will then compare the performance of a Logistic Regression classifier on four different training sets: the original imbalanced data, data balanced using a naive oversampling method (SMOTE), data balanced using a clustering-based oversampling approach (CBO), and data balanced using a clustering-based undersampling approach (CBU).

You will submit a Jupyter Notebook with your complete code, visualizations, and a plausible story that explains your findings. The notebook should be well-commented, reproducible, and easy to follow.

Dataset: The dataset is available on Kaggle: Credit Card Fraud Detection.

---

2. Tasks

Part A: Data Exploration and Baseline Model [10 points]

1. **Load and Analyze the Dataset:** Load the `creditcard.csv` dataset. This dataset has already been pre-processed using PCA, so no feature engineering is required. **[1]**
2. **Analyze Class Distribution:** Print the class distribution (count of fraudulent vs. non-fraudulent transactions). A pie chart or bar plot would be a good visualization. Clearly state the degree of imbalance. **[2]**

3. **Baseline Model:**
    - Split the original dataset into training and testing sets. **Crucially, ensure the test set retains its original imbalance**. **[2]**
    - Train a Logistic Regression classifier on the imbalanced training data. This will be your **Model 1**. **[2]**
    - Evaluate the model's performance on the test set. Pay close attention to metrics robust to imbalance, such as **Precision, Recall, and the F1-score** for the minority (fraudulent) class. Explain why accuracy is a misleading metric in this scenario. **[3]**

Part B: Resampling Approaches [25 points]

1. **Naive Oversampling (SMOTE): [5]**
    - Apply the **SMOTE** (Synthetic Minority Over-sampling Technique) algorithm to the training data. This method generates synthetic samples for the minority class. **[3]**
    - Explain how SMOTE works and its potential limitations, such as generating noisy samples if the minority class is not well-defined. **[2]**
    - **Citation:** N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
2. **Clustering-Based Oversampling (CBO): [10]**
    - Explain the concept of using clustering for oversampling to ensure diversity and explain how it achieves this goal. **[2]**
    - Use a clustering algorithm, such as **K-Means,** to identify a few clusters within the **training data of the minority class only**. You can choose k based on intuition or the Elbow Method. **[5]**
    - Oversample from each minority cluster to create a new, balanced dataset. The goal is to ensure that all sub-groups are well-represented, thereby avoiding the creation of synthetic samples in regions with no actual data. **[3]**
3. **Clustering-Based Undersampling (CBU): [5]**
    - Explain the concept of using clustering for undersampling. The idea is to find sub-groups within the majority class and strategically remove instances to maintain a representative sample while reducing its size. **[2]**
    - Use a clustering algorithm to find clusters within the **training data of the majority class only**. **[5]**
    - Undersample from each cluster. For instance, you could remove samples from clusters that are closer to the minority class, or you could simply undersample each cluster proportionally to its size to preserve the original distribution of the majority class. **[3]**
    - The final training set will comprise all instances of the minority class and the selected subset of majority class instances.

Note: Libraries like `imblearn` provide convenient implementations for both SMOTE and clustering-based resampling methods, such as `ClusterCentroids` for undersampling.

Part C: Model Comparison and Analysis [15 points]

1. **Train and Evaluate Models: [5]**
   - **Model 2 (SMOTE):** Train a Logistic Regression classifier on the training data balanced with **SMOTE**. Evaluate its performance on the same, imbalanced test set from Part A.
   - **Model 3 (CBO):** Train a Logistic Regression classifier on the training data balanced with your **clustering-based oversampling approach**. Evaluate its performance on the same, imbalanced test set.
   - **Model 4 (CBU):** Train a Logistic Regression classifier on the training data balanced with your **clustering-based undersampling approach**. Evaluate its performance on the same, imbalanced test set.
2. **Performance Comparison [5]:** Create a summary table or bar chart comparing the Precision, Recall, and F1-score of the four models (Baseline, SMOTE, CBO, and CBU) for the minority class.
3. **Conclusion and Recommendations: [5]**
   - Discuss the benefits and drawbacks of each resampling method for this problem. Which method performed the best and why?
   - Explain how the clustering-based approaches address the limitations of a naive method like SMOTE.
   - Conclude with a recommendation on which resampling strategy the company should adopt.

---

3. Submission Guidelines

- The assignment is due in **7 days**.
- Submit a single Jupyter Notebook (`*.ipynb`) with all your code, visualizations, and answers to the conceptual questions.
- Your notebook should be self-contained and run without errors. Use markdown cells to structure your answers and explanations.
- Ensure your code is clean, readable, and well-commented.

Evaluation Criteria:

- Correct implementation of all four models.
- Correct use of evaluation metrics suitable for imbalanced datasets.
- Clear and insightful comparison of the models' performance.

- Demonstrated understanding of class imbalance, the limitations of naive resampling, and the benefits of a clustering-based approach.

Good luck!