Machine Learning in Medicine - Labwork 1

Nguyen Quang Huy - 22BI13195

February 2025

I. Introduction

This report will describe in detail my experiments on the two well-known datasets for heartbeat classification, the MIT-BIH Arrhythmia Dataset and the PTB Diagnostic ECG Database [2]. In particular, I elected to implement the Extreme Gradient Boosting (XGBoost) algorithm for the classification task due to the significant class imbalance, which we will explore in more detail momentarily.

II. MIT-BIH dataset

Created by Massachusetts Institute of Technology and Beth Israel Hospital, the MIT-BIH dataset contains 48 half-hour ECG recordings from 47 patients, while PTB-XL consists of 21,837 clinical ECG recordings from 18,885 patients. It consists of 109,446 labeled heartbeats, with 187 feature vectors that constitutes a time-series signal, representing the electrical activity of the patient's heart over time. There are 5 classes for the 5 types of heartbeat: Normal Beats, Supraventricular Ectopic Beats, Ventricular Ectopic Beats, Fusion Beats and Unclassifiable Beats.

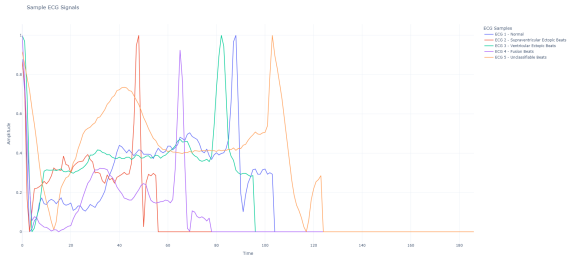

Figure 1: Sample ECG Signals, Represented as Amplitude over Time

One of the first key challenges this data presents is with the volume and dimensionality of our dataset, creating a lot of overhead during the training process that is computationally costly. In addition, this data also exhibits extreme class imbalance as seen within both the training and testing set.

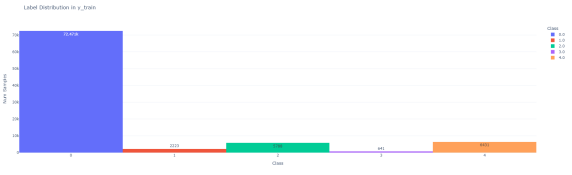

Figure 2: Label Distribution in training set

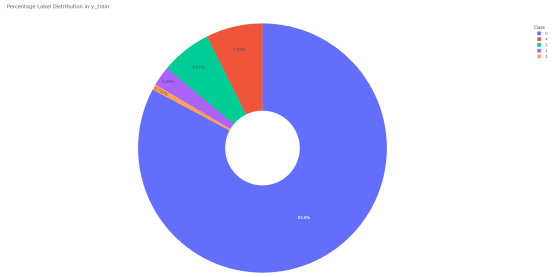

Figure 3: Percentage Label Distribution in training set

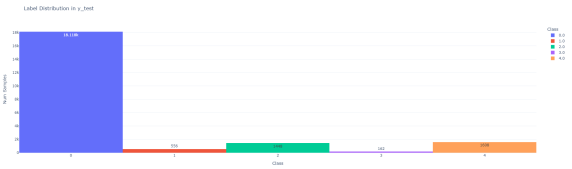

Figure 4: Label Distribution in testing set

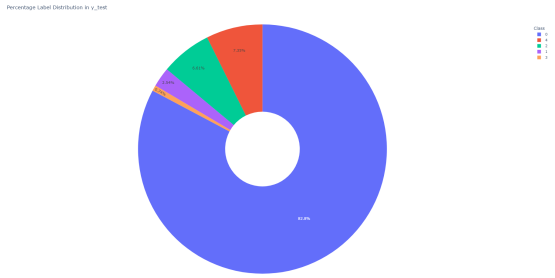

Figure 5: Percentage Label Distribution in testing set

From visually examining these plots, we can infer that across both the training and testing sets, normal heartbeats are more predominant, occupying over

80% of the samples within each set. In contrast, the smallest minority classes only account for 0.73% and 2.54% of our label distribution respectively. Here we find our first problem with this dataset: in normal class imbalance cases, the simple solution is often minority class oversampling (e.g. SMOTE) or majority class undersampling (e.g. RandomUndersampling). However, with the significant imbalance in this particular case, this leaves us with either a new dataset of mostly synthetic samples that invariably leads to overfitting, or losing the vast majority of our training samples. Therefore, there is no real way to account for the class imbalance in data preprocessing, thus necessitating an algorithm that can best propagate itself to handle this data.

Beyond the previous issue, we also look toward reducing the dimensionality of our dataset, using Principal Component Analysis. Principal Component Analysis (PCA) is a widely known and utilized technique in exploratory data analysis, dimensionality reduction and feature extraction [1]. The aim is to transform a high-dimensional dataset into a lower-dimensional space while still retaining the structure and underlying patterns or relationships among the variables. With our data, since the value range is already normalized to $[0, 1]$, we need not prepare our data any further.

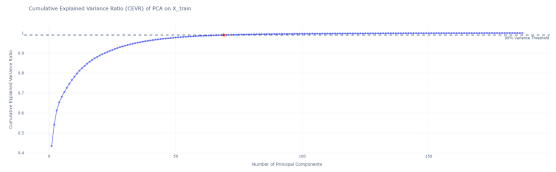

Figure 6: Cumulative Explained Variance Ratio (CEVR) of PCA on X_train

From inspecting the feature variance explained across principal components, we choose the number of components that accommodates for over 99% of the cumulative explained variance ratio. This allows us to maintain much of the information from the original data while significantly reducing its size, and we arrive at the ideal number of PC's being 69. Proceeding from this step, we fit the PCA model to the training set and transform the data a new space with 69 principal components. Then the same transformation is then applied to the test data.

With this data, we can begin the process of training and hyperparameter tuning. This is done using the XGBoost algorithm, which is particularly effective for our imbalanced dataset thanks to 2 key features. The sequential nature of boosting means each new trees will focus on fixing the errors of the old one. Since minority classes are most prone to being misclassified, the model is naturally inclined towards prioritizing learning these patterns. Additionally, XGBoost uses second-order gradient information (Hessian) to optimize the loss function, allowing for more precise updates, especially for hard-to-classify (often of the minority class) samples. To ensure that our XGBoost model can attain the best accuracy, we conducted an extensive optimization process by employing the hyperparameter tuning technique via the RandomizedSearchCV function from the sklearn library. This structure allows us to systematically explore the hyperparameter space we defined, and identify the best combination for our model. The following hyperparameters and their respective candidate values were included in our search space:

- n_estimators: 100, 200, 500
- max_depth: 3, 6, 9
- learning_rate: 0.01, 0.1, 0.2
- subsample: 0.7, 0.8, 1.0

This process was conducted with 500 iterations and cv = 5, which split our training data into 5 equal slices and assign their respective role as training/validation data to ensure our tuning does not result in overfitting. Moreover, we use balanced accuracy as opposed to normal accuracy as the scoring mechanism to optimize, which more aptly account for the imbalanced dataset. The balanced accuracy is calculated as:

$$\text{Balanced Accuracy} = \frac{\text{Recall} + \text{Specificity}}{2}$$

Table I shows the best performing hyperparameter combination, which we will implement for this data moving forward.

subsample	n_estimators	max_depth	learning_rate
1.0	500	6	0.2

Table I: The best hyperparameter combination found through RandomizedSearchCV

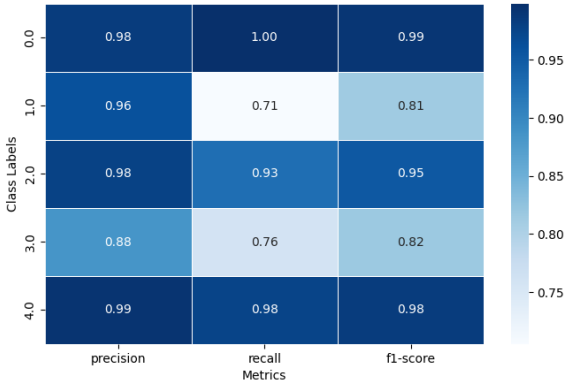

Figure 7: Classification Report Heatmap

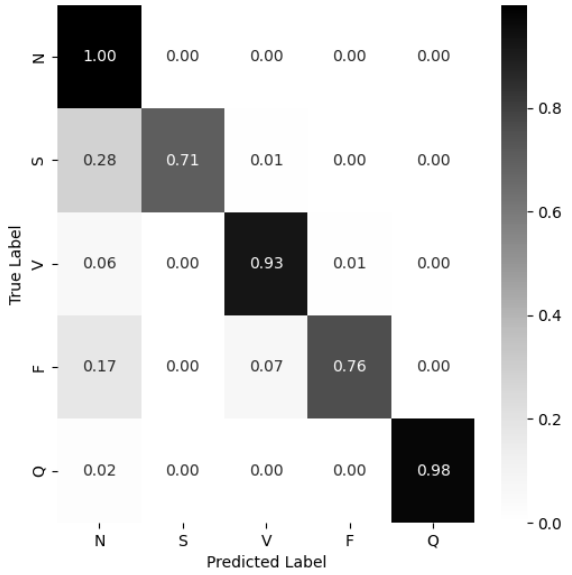

Figure 8: Normalized Confusion Matrix

With the final optimized version of the model, we were able to obtain the results as shown in Figure 7 and Figure 8. Most significantly, we achieved an accuracy of 98.3% and a balanced accuracy of 87.35%. The baseline paper ECG Heartbeat Classification: A Deep Transferable Representation did not report their balanced accuracy specifically, only that they achieved an accuracy of 95.9% [3] which we exceeded with our method.

III. PTBDB Dataset

The PTB Diagnostic ECG Database (PTBDB) is a dataset of ECG recordings from 290 subjects (148 with heart conditions, 142 healthy). It includes 15 simultaneous ECG leads sampled at 1 kHz, labeled with diagnoses like myocardial infarction, cardiomyopathy, and healthy controls. Similar to the MIT-BIH dataset it contains 187 feature vectors and 1 target vector containing 2 classes, normal and abnormal. It consists of 14,552 samples which also exhibits class imbalance, though not to the degree observed in the previous dataset.

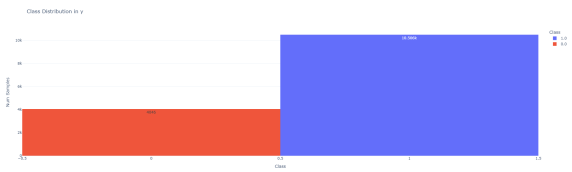

Figure 9: Label Distribution in the PTBDB dataset

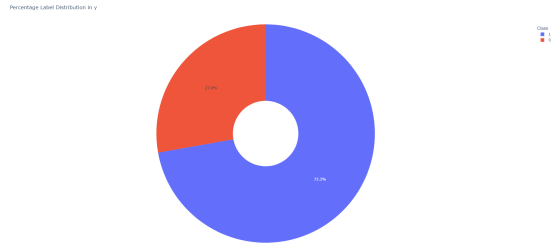

Figure 10: Percentage Label Distribution in the PTBDB dataset

Since we have more samples within the minority class, we can effectively handle the data imbalance. Specifically, we implemented random undersampling using the RandomUnderSampler from the imblearn library, reducing both classes to 4,046 samples. We then split the dataset into training and testing sets with the ratio of 80/20 respectively, stratifying our label distribution across these sets to maintain class balance. In addition, we use another method to reduce the dimensionality of our dataset based on the Pearson correlation coefficient between features. First, we compute a correlation matrix from the training data to measure the pairwise linear relationships between features. Then, features with a correlation greater than 0.9 are identified and these highly correlated features are removed from both the training and testing sets. This leaves us with 70 most relevant features to our training.

From here, the training and hyperparameter tuning process is similar to the previous dataset. We use XGBoost with the same hyperparameter search space, which resulted in the combination seen in Table II

subsample	n_estimators	max_depth	learning_rate
0.8	500	9	0.1

Table II: The best hyperparameter combination found for PTBDB dataset

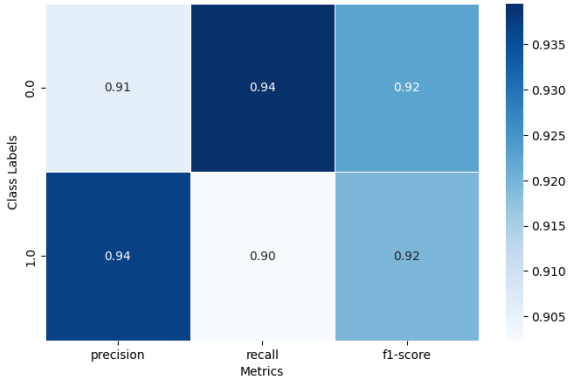

Figure 11: Classification Report Heatmap in PTBDB dataset

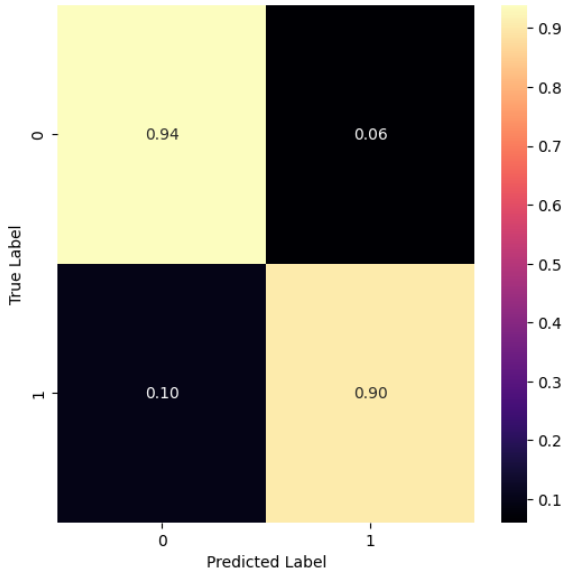

Figure 12: Normalized Confusion Matrix in PTBDB dataset

Finally, we achieved an accuracy of 92.09% which is equivalent to the balanced accuracy, since we normalized our data distribution.

REFERENCES

[1] Karl Pearson. *On lines and planes of closest fit to systems of points in space*. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2(11):559–572, 1901. DOI: 10.1080/14786440109462720.

[2] Shayan Fazeli. *ECG Heartbeat Categorization Dataset*. 2018. https://www.kaggle.com/datasets/shayanfazeli/heartbeat/data. [Accessed: Thursday 27th February, 2025].

[3] Mohammad Kachuee et al. *ECG Heartbeat Classification: A Deep Transferable Representation*. arXiv preprint arXiv:1805.00794, 2018. https://arxiv.org/pdf/1805.00794.pdf.