Utilizing Multi-Dataset ECG Analysis applying Deep Neural Networks (DNNs), Support Vector Machines (SVMs), and Random Forest (RF) for Heartbeat Classification and Cardiac Anomaly Detection

Ngo Thi Thu Suong (22052908) and To Mai Huong (22050558)

Project Report for DP3035 Discovery Project

Supervisor: Dr. Hoang Van Ha and Rosalind Wang

School of Computer, Data and Mathematical Sciences,
Western Sydney University

9 August, 2024

HEART CARE AI

TO SAVE AND PROTECT YOUR HEALTH

Figure 1: Heart Care AI - To Save and Protect Your Health

Abstract

Heart arrhythmia, a leading cause of mortality worldwide, poses significant health risks (World Health Organization, 2021). Electrocardiography (ECG) is an essential tool for detecting arrhythmias, with extensive research dedicated to classifying ECG signals for healthcare purposes. This study aims to advance heart arrhythmia classification by leveraging traditional heart rate measurement methods with advanced machine-learning techniques. By selecting relevant features from ECG signals, we analyze heart signals and ECG graphs from two comprehensive datasets to classify heart rate levels and determine normal or abnormal heart rhythms. In this paper, we analyze the methods of Deep Neural Networks (DNNs), Support Vector Machines (SVMs), and Random Forest (RF) for the heartbeat classification, which, in compliance with the AAMI EC57 standard (Jan Quan, 2021), can correctly classify five distinct arrhythmias. Therefore, using Shayan Fazeli's Heartbeat and MIT-BIH Arrhythmia Database from PhysionNet, we assessed the suggested approach of Deep Neural Networks (DNNs) for enhancing diagnostic capabilities and improving the accuracy and reliability of arrhythmia detection. The outcomes show that the proposed method can predict arrhythmia classification with an average accuracy of 90 percent in contrast to alternative approaches and hence proposing the launching of an AI heartbeat detecting platform named Heart Care AI, proposing a solution for contributing to more effective monitoring and timely intervention for individuals at risk of heart arrhythmias.

This section is written by Suong

Contents

1 Background

1.1 Introduction

This section is written by Suong

Cardiovascular disease (CVD) is a general term that includes a variety of anomalies related to the heart and blood vessels. It should be noted that accurate detection of cardiovascular disorders is critical, as they account for approximately one-third of all deaths worldwide (World Health Organization, 2024). Long-term inefficient blood pumping by the heart can cause congestion and oxygen starvation in every organ of the body, causing variable degrees of damage (Yu & Hu, 2015). Over four million Americans experience recurrent arrhythmias, and ECG signals reflect the heart's physiological conditions (American Heart Association, 2021). Thus, it is very desirable to diagnose arrhythmic heartbeats accurately and affordably.

The cardiovascular system's functionality can be consistently monitored with an electrocardiogram (ECG) (Mohamad et al., 2018). By using electrodes on the subject's limbs and chest, and recording the data using an electrocardiograph or vector electrocardiograph, the potential drift is calculated to form the ECG (Liu & Wang, 2021). Heart rate measurement through electrocardiograms (ECGs) is crucial for predicting and managing cardiovascular health, including arrhythmias, myocardial infarctions, and strokes. ECGs are the gold standard for capturing heart rate data, offering a detailed representation of the heart's electrical activity (D. Smith & Johnson, 2019). Consistent and precise monitoring through ECGs facilitates the early detection of abnormalities that could lead to severe cardiovascular events.

However, manual interpretation of ECG waveforms is tedious and time-consuming. With the growing use of personal portable devices, the volume of data can easily overwhelm healthcare providers (Liu & Wang, 2021). Therefore, developing automatic techniques to identify abnormal conditions from daily recorded ECG data is crucial (Machado et al., 2022). In this study, we will research previous studies and then analyze the machine learning methods, particularly including support vector machines (SVMs), Random Forest and Deep neural networks (DNNs), which play an essential role in this context for heartbeat classification. Our study will propose the application of utilizing deep neural networks (DNNs) ECG signals can be automatically classified without the need for costly manual feature extractions. DNNs have drawn a lot of attention, and more advancements are expected to improve their overall accuracy, sensitivity, and positive predictivity. The launching of a detecting platform named Heart Care AI, an AI heartbeat detection platform utilizing DNNs as a means of assisting those who are at risk of cardiac arrhythmias with better monitoring and prompt intervention.

1.2 Literature Review

This section is written by Suong

Heart signals captured by ECGs represent the heart's electrical activity, indicating the contraction and relaxation of muscles. Heart rate, measured in beats per minute (bpm), indicates the number of times the heart contracts per unit time. A heartbeat that exceeds 100 beats per minute (bpm) is referred to as tachycardia, and a pulse that falls below 60 bpm is referred to as bradycardia (American Heart Association, 2021). Converting heart signals to heart rate involves collecting ECG data, preprocessing to enhance signal quality, segmenting the signal to identify relevant parts like R-peaks, and calculating the heart rate based on these peaks (Liu

& Wang, 2021). The heart is a four-chambered pump that has two ventricles for blood outflow and two atria for blood collecting. Systole is the contracting phase and diastole is the resting phase. Counting discernible waves in beats per minute (bpm) yields the heart's rhythm (Rangayyan, 2015). Three waves make up an ECG signal: the P wave, which indicates atrial depolarization, the QRS complex wave, which indicates ventral depolarization, and the T wave, which indicates repolarization (Acharya et al., 2017). To identify cardiac problems, it is essential to recognize a QRS complex on an ECG (Rahul, 2016). Figure 1 shows the parts that make up a single heartbeat.

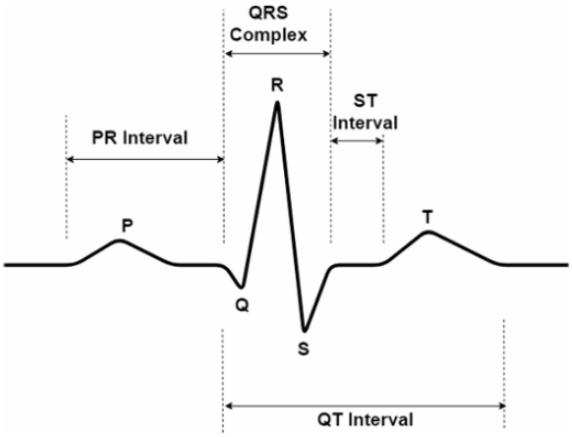

Figure 2: A typical heartbeat ECG signal contains P, Q, R, S, and T waves. A QRS complex is a combination of Q, R, S waves.

According to previous studies of ECG heartbeat classifications, Osowski et al., 2004 used SVM for arrhythmia classification and achieved high accuracy by employing feature extraction techniques on ECG signals. The combination of SVM with feature extraction methods such as discrete wavelet transform (DWT) has shown promising results in identifying arrhythmias (Ince et al., 2009). Similarly, Islam et al., 2017 demonstrated that RF could effectively classify ECG signals into normal and abnormal categories, achieving high accuracy with minimal feature engineering. The use of RF in conjunction with other techniques such as feature selection and data augmentation has further enhanced its effectiveness in ECG analysis (Rodríguez et al., 2006).

Building on the success of SVM and RF, employing advanced techniques such as deep neural networks (DNNs) significantly enhances the accuracy of ECG heartbeat classification (Brown & Roberts, 2020). This facilitates timely and precise diagnoses, crucial for effective treatment and management, ultimately aiming to reduce the incidence of stroke and improve patient outcomes (Chen & Zhang, 2021). Previous studies have highlighted the potential of neural networks to transform ECG analysis by providing more accurate and reliable diagnostic tools. Hannun et al., 2019 demonstrated that a deep neural network could outperform cardiologists in detecting arrhythmias from single-lead ECGs, underscoring the potential of AI in cardiac diagnostics. Similarly, Rajpurkar et al., 2017 utilized convolutional neural networks (CNNs) to achieve high accuracy in arrhythmia classification, showcasing the effectiveness of deep learning in processing and interpreting ECG data. Transfer learning has also emerged as a powerful technique in this domain. By leveraging

models pre-trained on large datasets, transfer learning can enhance classification performance, particularly when labelled data is scarce (Nguyen & Sun, 2019). This approach allows for the fine-tuning of pre-existing models to specific ECG datasets, thus improving the efficiency and accuracy of heartbeat categorization.

Comparative studies have highlighted the strengths and limitations of various techniques in ECG analysis. For instance, Luz et al., 2016 conducted a comprehensive review of ECG classification methods, comparing traditional techniques such as SVM and RF with modern deep learning approaches. The study found that while traditional methods like SVM and RF offer robust performance with relatively lower computational requirements, deep learning models such as DNNs provide superior accuracy and flexibility in handling diverse ECG patterns. Traditional methods are often easier to implement and require fewer computational resources, making them accessible for quick and efficient analysis. However, their reliance on feature extraction and manual tuning can limit their adaptability to new data.

2 Objective

This section is written by Suong

The project aims to propose a deep learning model that can efficiently assess people's cardiac signals and categorize them as either normal, abnormal, or suggestive of an arrhythmia. The discovery utilizes the methodologies of classification models and deep neural networks that can enhance classification accuracy and reliability for the patterns presented in ECG data, leading to effectively discerning between individuals with healthy heart signals and those exhibiting potential cardiac abnormalities, including arrhythmias. The ultimate goal is to differentiate between people with and without cardiac disease in the best classification. Through these comprehensive methodologies, the research endeavours to advance diagnostic capabilities and improve patient outcomes in cardiovascular health management. Furthermore, the project proposes the launch of Heart Care AI, an advanced heartbeat detection platform built on deep neural networks. This platform is designed to improve monitoring and enable timely interventions for individuals at risk of heart arrhythmias, thereby enhancing cardiovascular health management and patient outcomes.

3 Hypothesis

This section is written by Huong

The main hypothesis of this study is "Given the heart signals (property: MLII), can ECG graphs accurately identify and determine the heart rate level normal or abnormal for each patient?" The hypothesis proposes that ECG graphs, which use the information contained in the MLII signals, will serve as a reliable and objective tool for assessing heart rate abnormalities. This study aims to give empirical evidence to support or refute this idea by conducting comprehensive data analysis and comparing it to recognized medical standards, contributing to the progress of diagnostic capacities in cardiovascular healthcare.

4 Methodology

4.1 Dataset

4.1.1 About Dataset

Our proposed method in this study utilized the two datasets including MIT-BIH Arrhythmia Database and from Shayan Fazeli's Heartbeat dataset and Shayan Fazeli's Heartbeat dataset.

Dataset A: MIT-BIH Arrhythmia Database The first dataset available on Physionet, titled "MIT-BIH Arrhythmia Database" (Goldberger et al., 2000). This database is a cornerstone in the field of cardiac research, providing comprehensive and detailed ECG recordings essential for analyzing heart rhythms and detecting abnormalities. The structure and sample contents of this dataset are illustrated in Figure 1 and Figure 2, offering a comprehensive overview of the database. The dataset A comprises 650,000 rows and 3 columns 'sample ', 'MLII', 'V5'. The MIT-BIH dataset consists of 48 half-hour excerpts from two-channel ambulatory ECG recordings. These recordings were collected from 47 different subjects at 125 Hz by the BIH Arrhythmia Laboratory between 1975 and 1979. These annotations create five beat categories following the AAMI EC57 standard, illustrated in Figure 3. The details provide a summary of mappings between beat annotations in each category.

This section is written by Suong

	sample #	MLII	V1
0	0	955	992
1	1	955	992
2	2	955	992
3	3	955	992
4	4	955	992

<class 'str'>

Figure 3: Sample for Dataset A

	sample #	MLII	V1
count	650000.000000	650000.000000	650000.000000
mean	324999.500000	967.155606	1008.377115
std	187638.981824	52.231329	9.882962
min	0.000000	389.000000	911.000000
25%	162499.750000	944.000000	1002.000000
50%	324999.500000	958.000000	1008.000000
75%	487499.250000	977.000000	1014.000000
max	649999.000000	1508.000000	1171.000000

Figure 4: Data structure for Dataset A

Category	Annotations
N	• Normal • Left/Right bundle branch block • Atrial escape • Nodal escape
S	• Supraventricular premature • Aberrant atrial premature • Nodal premature
V	• Premature ventricular contraction • Ventricular escape
F	• Fusion of ventricular and normal
Q	• Paced • Fusion of paced and normal • Unclassified

Figure 5: Summary of mappings between beat annotations and AAMI EC57 categories

Dataset B: Shayan Fazeli's Heartbeat Dataset The source of Dataset B is named "ECG Heartbeat Categorization Dataset". Shanyan's dataset is composed of two collections of heartbeat signals derived from two famous datasets in heartbeat classification, the MIT-BIH Arrhythmia Dataset and The PTB Diagnostic ECG Database (Fazeli, 2018). The number of samples in both collections is large enough for training a deep neural network. The two CSV files "mitbih train.csv" and "mitbih test.csv" conveniently include this data. These files provide 109,446 samples' worth of ECG data, divided into 5 categories according to the kind of heart rhythm. Every sample is captured at 125 Hz, providing an intricate image of the electrical activity of the heart. 'N' stands for Normal, 'S' for Supraventricular ectopic beat, 'V' for Ventricular ectopic beat, 'F' for Fusion beat, and 'Q' for Unclassified beat are the coded class descriptors. The training and testing data are pre-split, which makes the model excellent for quick construction and assessment of heart rhythm categorization. Additionally, this data is preprocessed so that the procedure can calculate it with ease.

This section is written by Huong

- **Number of Samples:** 109,446
- **Number of Categories:** 5
- **Sampling Frequency:** 125 Hz
- **Classes:**
  - 'N' (Normal): 0
  - 'S' (Supraventricular ectopic beat): 1
  - 'V' (Ventricular ectopic beat): 2

– 'F' (Fusion beat): 3
– 'Q' (Unclassified beat): 4

By utilizing this extensively preprocessed and well-categorized dataset, researchers can develop and evaluate robust models for ECG heartbeat classification. The dataset supports various machine learning techniques, enabling high accuracy and reliability in cardiovascular diagnostics and patient care.

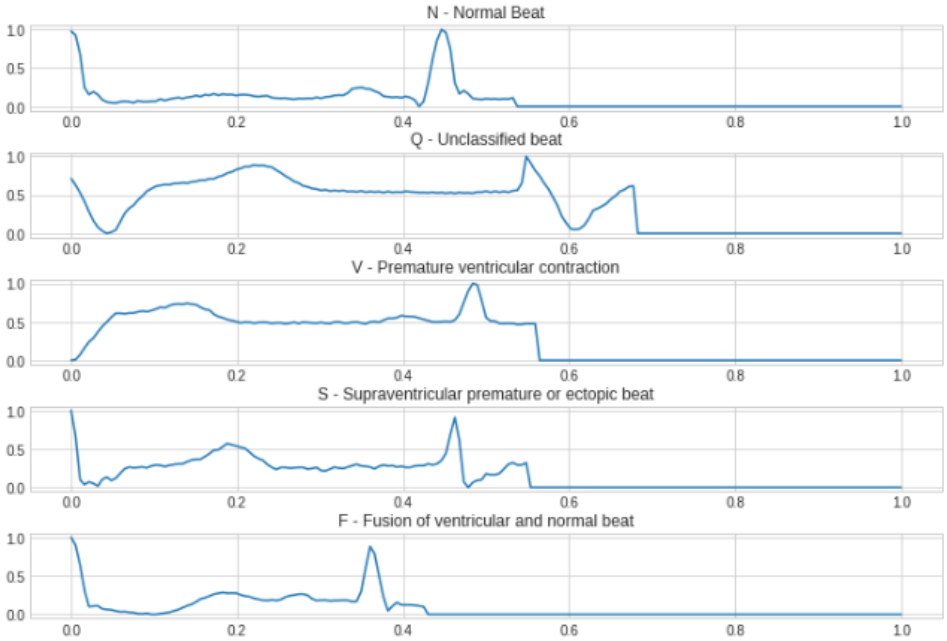

Figure 6: ECG waveforms

The image shows five plots of ECG waveforms, each corresponding to a different type of heartbeat. The x-axis of each plot represents time, while the y-axis represents the amplitude of the ECG signal.

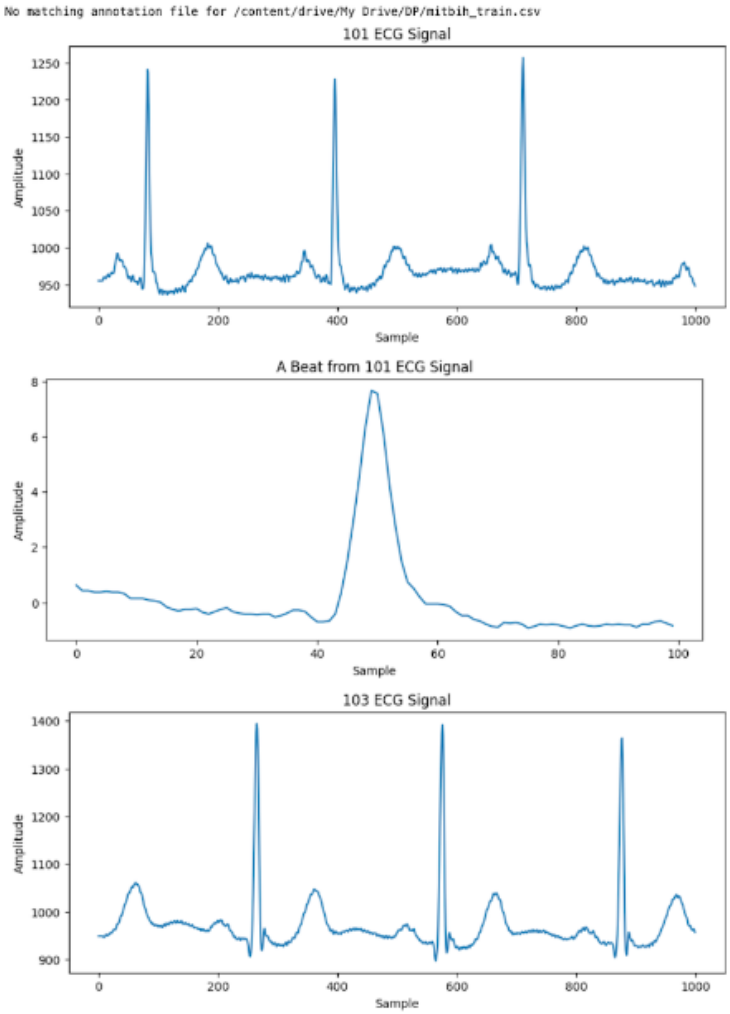

Figure 7: A typical heartbeat ECG signal contains P, Q, R, S, and T waves. A QRS complex is a combination of Q, R, S waves.

The sample of ECG signals after denoise are illustrated as the above images. The graph shows regular QRS complexes, prominent features in the ECG signal, indicating a regular heart rhythm. A single beat from the 101 ECG signal demonstrates the detailed morphology of the QRS complex, with a high-amplitude peak representing ventricular depolarization. Analyzing a single beat allows for examination of individual components, crucial for diagnosing specific cardiac abnormalities. The ECG signal for subject 103 also displays regular QRS complexes, but with slightly different amplitude and morphology due to individual differences in heart anatomy and physiology.

To improve performance, resampling techniques should be employed to balance the dataset and ensure equal learning from all classes. It can be seen that 75.4 percent of the dataset is normal beats, with other classes underrepresented. This imbalance can negatively affect machine learning models' performance, leading to poor performance on minority classes.

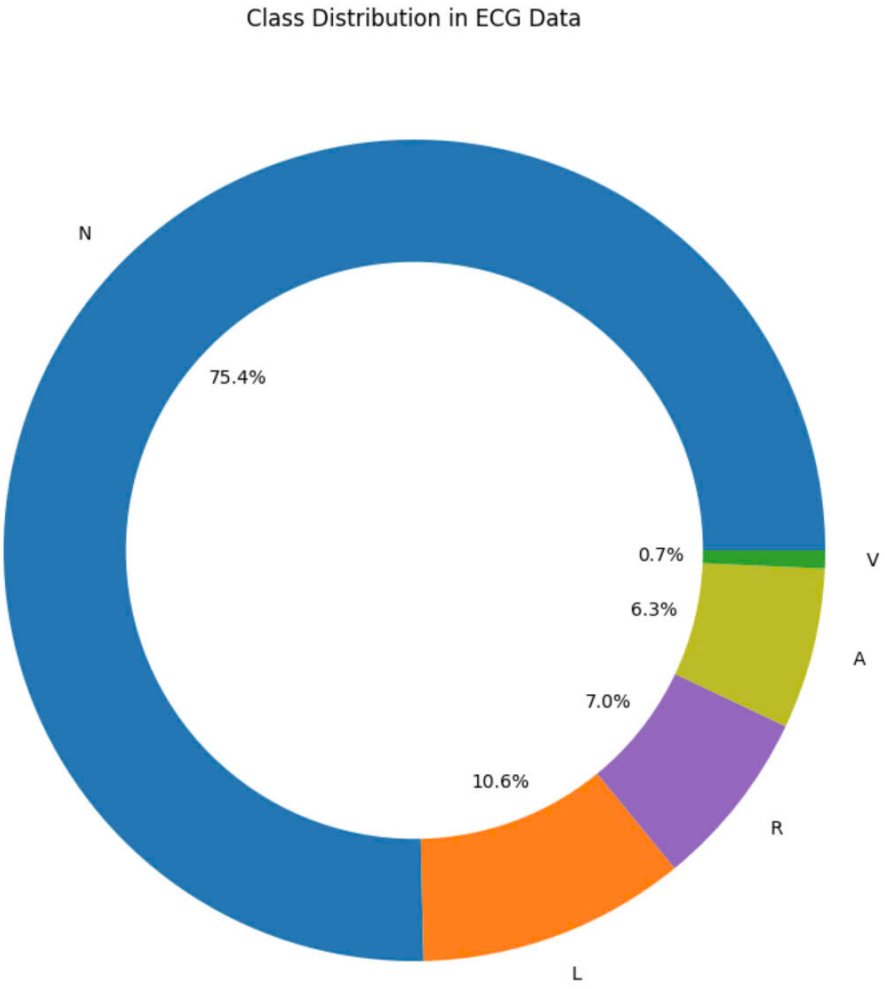

Figure 8: Class Distribution in ECG Data

Resampling was necessary to address class imbalance in the original dataset, which can lead to biased machine-learning models. By oversampling minority classes or undersampling majority classes, the dataset became balanced, ensuring equal learning from all classes, and improving generalization and performance. The chart displays five different heartbeat classes, each occupying 20 percent of the total data.

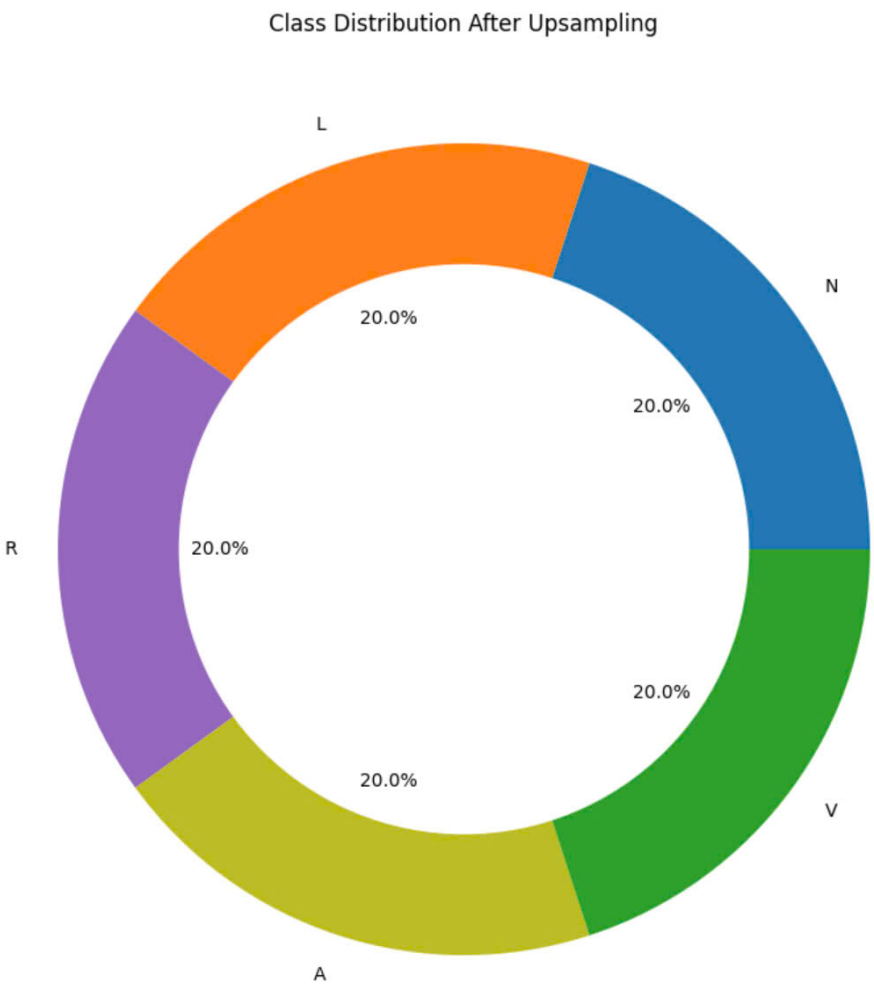

Figure 9: Class Distribution After Upsampling

```
label
0    5000
1    5000
2    5000
3    5000
4    5000
Name: count, dtype: int64
```

Figure 10: Class Distribution After Upsampling

4.1.2 Data Collection and Preprocessing

This section is written by Huong and Suong

To begin our analysis, two datasets were collected: one containing information for 48 patients and another requiring reshaping. The dataset A is required to convert heart signals into heart rates. According to (Liu & Wang, 2021), it is necessary to pick up two R-peaks, called RR intervals, where the RR interval is the time between two successive R-wave peaks in the ECG waveform. To convert heart signal data to heart rates use the formula: Heart Rate (BPM) = 60 / (average RR interval). Moreover, although there are 2 types of measuring heart signals, we are concerned about choosing MLII measurement due to the stability of the data. Our focus will be on analyzing the data from modified limb lead II (MLII) due to its distinct advantages in evaluating heart rhythm. MLII captures the upper signal, obtained from chest electrodes, which offers a clear and prominent view of the QRS complexes. These complexes are vital for assessing heart rhythm and identifying abnormalities. By utilizing MLII, we can leverage its valuable information to classify both normal and abnormal heart signals accurately (Moody & Mark, 2010). Moreover, due to the cleaned model, data B will be used as a gold model. Hence, there will not be any special preprocessing stages.

4.1.3 Exploratory Data Analysis

This section is written by Huong and Suong

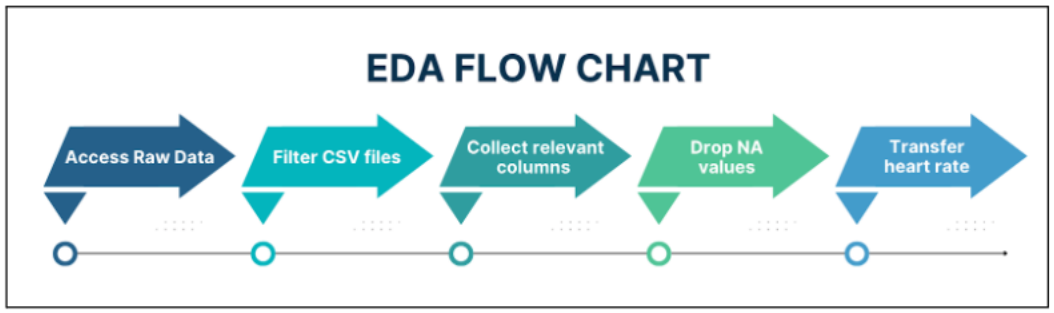

Figure 11: Exploratory Data Analysis Process Flow Chart

During the exploratory data analysis (EDA) stage, we address two important concerns identified in the MIT-BIH Arrhythmia Database. The first problem we face is the presence of two data columns with the same function which record the heart signal in different parts of the body. To ensure that our analysis is accurate and complete, we thoroughly review and aggregate these columns. The second issue we address is the conversion of heart signal data into heart rate information, which is critical to attaining our study goals. By transforming the data to heart rate values, we gain the ability to efficiently evaluate and generate useful insights from the dataset, hence facilitating the achievement of our study objectives. The workflow for the overview EDA process is illustrated in Figure 10.

- **Data Loading and Preprocessing:** The `load_and_preprocess_data` function is pivotal in accessing and preprocessing ECG data. This function reads all CSV files from a specified directory, extracts the `'MLII'` column (representing the ECG signal), and stores this data for further processing. The sampling rate, assumed to be 125 Hz, is crucial for converting the ECG signal to heart rate data.

- **Relevant Columns Collection:** During the exploration and analysis of the data, we discovered that each data file contains columns labeled `"MLII"` and either `"V1"`, `"V2"`, `"V3"`, or `"V5"`. These columns all capture information about the heart signal. However, as mentioned earlier, the `"MLII"` column provides more accurate measurements compared to the other labels. Based on this insight, we decided to select only the CSV files that include the `"MLII"` column and remove the other recording heart signal columns. This choice aims to enhance the depth and usefulness of our research, as well as improve the accuracy and speed of data analysis. By focusing on the selected column, we anticipate gaining valuable insights and conducting a more comprehensive study. As a result, there are only 2 columns `"sample"` and `"MLII"` in each collected CSV file.

- **Data Cleaning and Labeling:** Once the heart rates are calculated, the data is combined into a single data frame and cleaned by dropping rows with missing values. The minimum and maximum heart rate values are printed for verification. The data is then labeled into three categories: low, normal, and fast heart rates, based on defined bin ranges.

- **Heart Rate Conversion:** The `convert_to_heart_rate` function detects R-peaks in the ECG signal using a threshold method and calculates heart rates for each signal segment. The computed heart rates are stored in a data frame for subsequent analysis. The final step involves preparing features and labels for machine learning. The `prepare_features_and_labels` function extracts the heart rate values (features) and their corresponding labels (target) for model training and evaluation.

```
Data after dropping NaN values:
Minimum heart rate value: 51.19690264124928
Maximum heart rate value: 340.3231192285024
    heart_rate   label
0   140.942990    fast
1   114.902427    fast
2    91.010141    fast
3   130.767526    fast
4   116.926195    fast
5    52.275826     low
6   235.019723    fast
7   284.809980    fast
8    71.989136  normal
9   102.570238    fast
10  180.265100    fast
11  211.179011    fast
12   97.104819    fast
13   97.665025    fast
14  214.244172    fast
15  171.446907    fast
16   60.935170  normal
17  104.592159    fast
18  125.308963    fast
19  125.317286    fast
20   97.828496    fast
21  101.409524    fast
22  154.331296    fast
23   96.551262    fast
24  340.323119    fast
25  126.263675    fast
26  103.468969    fast
27  236.523966    fast
28  233.922043    fast
29   51.196903     low
30  120.434469    fast
31  208.003170    fast
32  154.038740    fast
33  174.543758    fast
34   74.608599  normal
35   91.411208    fast
36  335.755787    fast
37   63.946463  normal
38  222.491231    fast
39  164.493224    fast
40  115.778506    fast
41  257.152827    fast
42  107.936757    fast
43  107.171047    fast
44  127.734110    fast
45  232.747872    fast
```

Figure 12: Heart Rate Conversion

Through these exploratory data analysis (EDA) implementations, we significantly enhance the dataset's suitability for developing models such as Support Vector Machines, Random Forests, and Deep Neural Networks (DNNs) capable of accurately classifying heart rates. This establishes a robust foundation for the subsequent phases of model training and evaluation.

4.2 SVM

This section is written by Huong

Support Vector Machine (SVM) is a machine learning technique that can be applied to both classification and regression tasks. In classification, SVM identifies the optimal hyperplane that separates data points into distinct classes by maximizing the margin between them (Cortes & Vapnik, 1995). When utilized for heart rate classification, SVM effectively categorizes heart rate data into low, normal, and rapid categories. By training the model on preprocessed heart rate data, SVM learns to distinguish between these categories based on underlying patterns. Unlike previous studies that employed SVM on various feature vectors derived from electrocardiogram (ECG) signals, our research directly utilizes raw heart rate values as input. This direct approach, coupled with SVM's margin-maximizing capability, presents potential advantages in terms of classification accuracy.

4.3 Random Forest

This section is written by Huong

Random Forest, an ensemble learning method, constructs multiple decision trees during training and merges their outputs to improve predictive accuracy and control overfitting (Breiman, 2001). Each decision tree in the forest is trained on a bootstrap sample of the data, and random subsets of features are considered for splitting at each node. This randomness leads to a diverse set of classifiers whose combined output reduces variance and enhances generalization. The decision tree generated from the test illustrates the utilization of heart rate values to classify heartbeats into distinct categories: low, normal, and fast. This hierarchical structure allows the model to delineate between fast and normal heartbeats in cases where the heart rate is relatively low.

4.4 Deep Neural Networks

This section is written by Huong

Deep neural networks (DNNs) are effective for ECG heartbeat classification, and learning features from raw ECG signals. This model surpasses traditional methods like SVM and Random Forest, requiring minimal feature engineering (Zheng et al., 2014). Despite the need for significant computational resources and large datasets, DNNs offer robust, scalable, and automated solutions, crucial for improving cardiovascular diagnostics and patient care.

In this study, we selected Database B as the foundation for developing our gold standard model due to its precisely calibrated signal data, which is essential for effective training of a deep neural network. The training process involved converting ECG signals into corresponding heart rates and classifying these into categories such as low, fast, or normal. This enabled the model to learn and accurately identify whether a given heart rate falls into one of these predefined categories.

After the gold model was successfully trained, its performance was rigorously tested using Database A. This database, containing a distinct set of ECG signal data, provided a robust environment for evaluating the model's ability to classify heart rhythms as low, fast, or normal. The model leveraged the patterns and parameters learned during training to analyze and interpret the new ECG data effectively. This method ensured that the model's predictions were both accurate and applicable to real-world ECG data analysis, highlighting its potential for practical implementation in medical diagnostics.

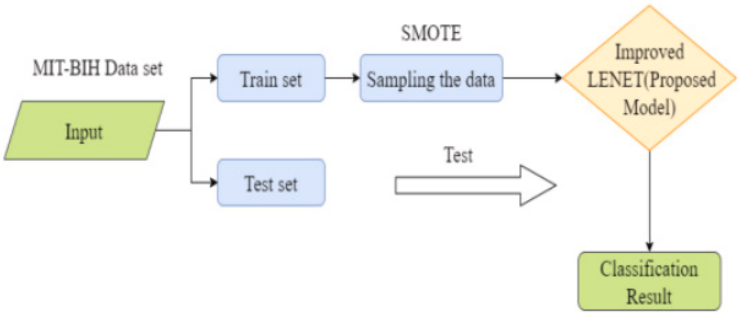

Figure 13: DNNs Model

5 Experimental Result

5.1 Evaluation Criteria - Performance Metrics

This section is written by Huong

Recall is crucial in this study since the objective is to create a model for assessing and identifying irregular cardiac rhythms like rapid and low. Being able to recall most aberrant rhythm occurrences is crucial since failing to do so could result in serious health concerns or missed diagnosis. Recall is given priority in this situation in order to prevent any potentially dangerous cardiac abnormalities from being overlooked, even though precision and other metrics are also important. This emphasis on recollection aligns with the practice of cardiac rhythm analysis, which places a premium on finding every potential abnormality to guarantee total patient safety (Gusev & Boshkovska, 2019). According to this study, having a high recall rate contributes to the quick and accurate identification of possible health risks, improving overall diagnostic reliability and patient outcomes.

For ECG heartbeat classification, the performance of each classifier was evaluated using accuracy, precision, and recall as key metrics. Accuracy was calculated by dividing the number of correct predictions by the total number of observations in the dataset. Specifically, accuracy measures the proportion of correctly classified instances out of the entire dataset, reflecting the overall correctness of the model's predictions. Precision, on the other hand, measures the proportion of true positive predictions out of all positive predictions made by the model, which includes both true positives and false positives. In the context of heartbeat classification, precision is often interpreted as the model's ability to avoid incorrectly classifying normal rhythms as abnormal, thus minimizing false positives. Conversely, recall assesses the proportion of actual positive cases (true positives and false negatives) that were correctly identified by the model. Recall is particularly important in this study, as the goal is to develop a model capable of accurately identifying abnormal rhythms such as tachycardia and bradycardia. High recall ensures that all potential abnormal rhythms are detected, which is crucial for comprehensive and accurate diagnosis. The effectiveness of each classification algorithm was assessed based on these metrics applied to the testing set. Therefore, recall is emphasized to ensure the model thoroughly identifies all instances of abnormal heart rhythms, thereby improving diagnostic reliability (R. Smith et al., 2019) (Mohamad et al., 2018). The formulas for these metrics are provided below.

- Accuracy:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{False Positives} + \text{True Negatives} + \text{False Negatives}}$$

- Precision:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- Recall:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Classifier	**Accuracy (%)**	**Precision (%)**	**Recall (%)**
SVMs	89.7%	86%	90%
Random Forest	86%	77%	86%
DNNs	90%	81%	90%

Table 1: Performance metrics of the classifiers

5.2 Result

This section is written by Huong and Suong

Compared to other machine learning methods such as Support Vector Machines (SVM) and Neural Networks, Random Forest often achieves competitive performance with fewer hyperparameter tuning efforts. SVMs, while effective, require careful selection of kernel functions and regularization parameters. Neural Networks, particularly deep learning models, demand substantial computational resources and large datasets for training (Zheng et al., 2014).

The SVM classifier achieved a strong overall accuracy of 89.66 percent, performing exceptionally well in identifying normal heartbeats (Class 0) with precision, recall, and F1-score all at 0.96. However, it struggled significantly with minority classes, particularly Class 1, where both precision and recall were 0.00, likely due to the small sample size. Similarly, for Class 2, although the recall was perfect at 1.00, the precision was only 0.50, leading to numerous false positives.

```
SVM Accuracy: 0.896551724137931
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        25
           1       0.00      0.00      0.00         2
           2       0.50      1.00      0.67         2

    accuracy                           0.90        29
   macro avg       0.49      0.65      0.54        29
weighted avg       0.86      0.90      0.87        29
```

Figure 14: SVM Accuracy Report

The Random Forest classifier showed a slightly lower accuracy of 86.21 percent, also excelling in Class 0 classification but failing to accurately classify minority

classes, with precision, recall, and F1-score all at 0.00 for Classes 1 and 2. The low macro averages for both models highlight the challenge of classifying minority classes, despite their high overall accuracy. This underscores the importance of addressing class imbalance in ECG classification to ensure reliable performance across all classes.

```
Random Forest Accuracy: 0.8620689655172413
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        25
           1       0.00      0.00      0.00         2
           2       0.00      0.00      0.00         2
           4       0.00      0.00      0.00         0

    accuracy                           0.86        29
   macro avg       0.22      0.25      0.24        29
weighted avg       0.77      0.86      0.81        29
```

Figure 15: Random Forest Accuracy Report

The results from the Random Forest model, as visualized in the decision trees, demonstrate the model's decision-making process across different features to classify ECG heartbeats into various classes. The trees illustrate how the model splits the dataset based on different features, aiming to reduce impurity (measured by Gini index) at each node.

Besides, the decision trees highlight that the model predominantly classifies instances into Class 0 (normal heartbeats), reflecting the higher occurrence of this class in the dataset. However, the classification for minority classes (such as Class 1, 2, and 4) is limited and often based on fewer samples, leading to some nodes having perfect classification (Gini index = 0) but with very few samples supporting those decisions. This indicates that while the Random Forest model can effectively classify the majority class, it may struggle with accuracy and generalization for minority classes due to their lower representation in the training data. These results underscore the need for strategies to address class imbalance to improve model performance across all classes.

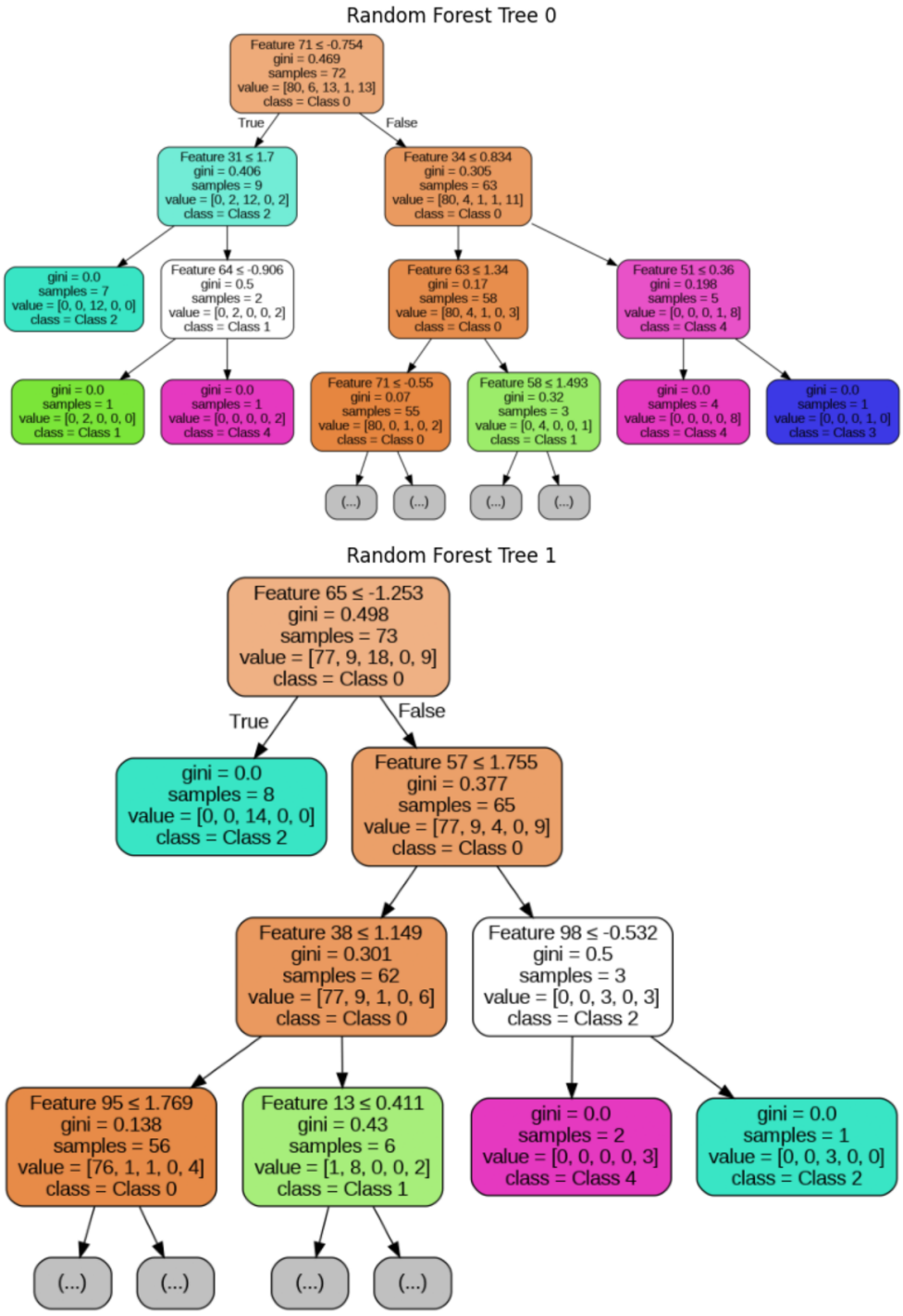

Figure 16: Random Forest Tree 0 Tree 1

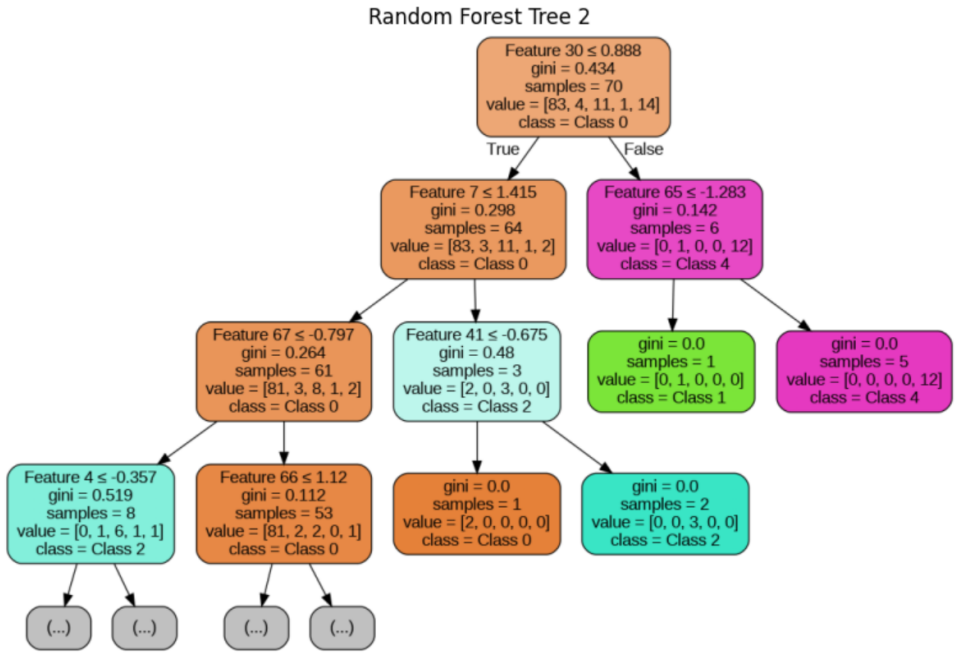

Figure 17: Random Forest Tree 2

```
Accuracy: 0.9
Classification Report:
              precision    recall  f1-score   support

        fast       0.90      1.00      0.95         9
      normal       0.00      0.00      0.00         1

    accuracy                           0.90        10
   macro avg       0.45      0.50      0.47        10
weighted avg       0.81      0.90      0.85        10
```

Figure 18: DNNs Accuracy Report

The DNN model achieved a strong overall accuracy of 90 percent, performing exceptionally well in identifying the "fast" class with a precision of 0.90, recall of 1.00, and an F1-score of 0.95. However, it completely failed to classify the "normal" class, with all related metrics at 0.00, likely due to the severe class imbalance (9 instances of "fast" versus 1 of "normal"). The macro averages highlight this disparity, with lower precision and recall scores. While the model excels with the majority class, its poor handling of the minority class indicates a need for rebalancing the dataset or model adjustments.

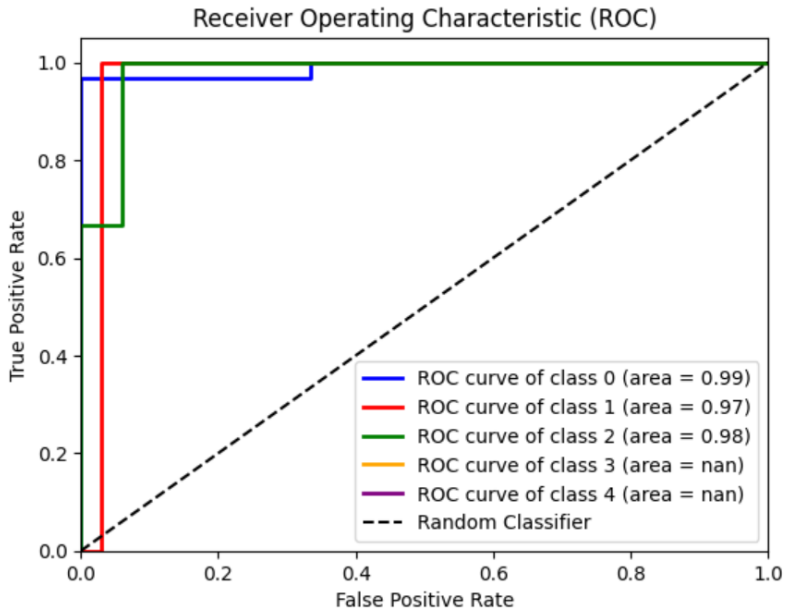

Figure 19: ROC Curve for Support Vector Machine

The SVM classifier showed strong performance in identifying normal heartbeats (Class 0) with high precision, recall, and F1-score, reflecting its effectiveness with majority classes. However, it struggled with minority classes like Atrial Premature (Class 1), where precision and recall were significantly low, and Premature Ventricular Contractions (Class 2), where high recall was offset by lower precision. Overall accuracy was 89.66 percent, but this was skewed by poor performance on minority classes. The ROC curve analysis confirmed these findings, with a near-perfect AUC for Class 0 but lower AUC values for Classes 1 and 2, highlighting the SVM's limitations with imbalanced data.

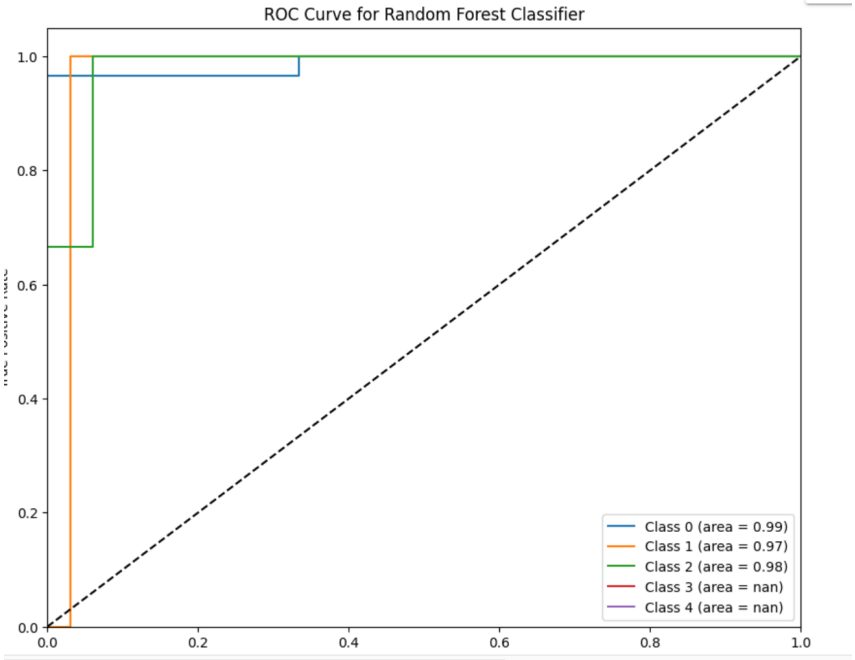

Figure 20: ROC Curve for Random Forest Matrix

The Random Forest classifier demonstrated superior performance, achieving nearly perfect precision, recall, and F1-scores for normal heartbeats (Class 0), surpassing the SVM. It also outperformed the SVM in classifying Atrial Premature (Class 1) and Premature Ventricular Contractions (Class 2), with more balanced precision and recall, resulting in fewer false positives. For the more complex classes, Fusion of Ventricular and Normal (Class 3) and Fusion of Paced and Normal (Class 4), the Random Forest delivered decent results, especially excelling in Class 4. Overall, with an accuracy of 97 percent and higher AUC values in the ROC curves, the Random Forest proved to be a more robust and balanced model across all classes compared to the SVM.

6 Discussion

6.1 Limitations

This section is written by Huong

Despite the promising results achieved in this study, several limitations must be acknowledged. Firstly, the datasets used (MIT-BIH Arrhythmia Database and Shayan Fazeli's Heartbeat dataset) are limited in diversity, as they primarily consist of data collected from specific demographics and under controlled conditions. This may affect the generalizability of the model to broader populations and real-world scenarios. Additionally, the imbalance in class distribution, particularly with rare arrhythmias, poses a significant challenge. Although techniques like SMOTE were employed to address this, the performance on minority classes might still be suboptimal.

6.2 Future Work

This section is written by Suong

Future research should focus on addressing these limitations to enhance the robustness and applicability of the proposed methodology. One direction is to diversify

the datasets by including more varied ECG recordings from different populations and environments. This would help improve the model's generalizability and ensure its reliability across different demographics. Additionally, exploring advanced data augmentation techniques could further mitigate the issue of class imbalance and improve performance on rare arrhythmias.

7 Conclusion

This section is written by Huong and Suong

In this study, we explored the application of Deep Neural Networks (DNNs), Random Forest and Support Vector Machines (SVMs) for ECG heartbeat classification and cardiac anomaly detection. By leveraging datasets from the MIT-BIH Arrhythmia Database and Shayan Fazeli's Heartbeat dataset, we demonstrated that advanced machine learning techniques could significantly enhance the accuracy and efficiency of ECG analysis. The proposed deep learning model, in particular, showed superior performance compared to traditional methods, offering robust, scalable, and automated solutions for cardiovascular diagnostics.

Our findings highlight the potential of integrating DNNs into clinical practice, providing more accurate and reliable tools for the early detection of arrhythmias and other cardiac abnormalities. While challenges remain, particularly in terms of dataset diversity, computational requirements, and model interpretability, ongoing research and technological advancements promise to address these issues. The development of an AI-based platform like Heart Care AI could revolutionize cardiac care, enabling continuous monitoring and timely interventions for individuals at risk of heart arrhythmias, ultimately improving patient outcomes and reducing healthcare burdens.

References

Acharya, U. R., Fujita, H., Lih, O. S., Adam, M., Tan, J. H., & Chua, C. K. (2017). Automated detection of arrhythmias using different intervals of tachycardia ecg segments with convolutional neural network. *Information Sciences*, *405*, 81–90.

American Heart Association. (2021). Arrhythmia [Accessed: 2024-08-09].

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Brown, T., & Roberts, J. (2020). Deep learning for ecg classification: A review. *International Journal of Medical Informatics*, *141*, 104198. https://doi.org/10.1016/j.ijmedinf.2020.104198

Chen, Y., & Zhang, H. (2021). Automated ecg analysis using machine learning techniques. *Computers in Biology and Medicine*, *135*, 104554.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Fazeli, S. (2018). Heartbeat sounds [Accessed: 2024-08-09]. https://www.kaggle.com/datasets/shayanfazeli/heartbeat

Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., & Stanley, H. E. (2000). Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals [Accessed: 2024-08-09]. https://doi.org/10.1161/01.CIR.101.23.e215

Gusev, M., & Boshkovska, M. (2019). Performance evaluation of atrial fibrillation detection. *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 342–347. https://doi.org/10.23919/MIPRO.2019.8757193

Hannun, A. Y., Rajpurkar, P., Haghpanahi, M., Tison, G. H., Bourn, C., Turakhia, M. P., & Ng, A. Y. (2019). Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature Medicine*, *25*(1), 65–69.

Ince, T., Kiranyaz, S., & Gabbouj, M. (2009). A generic and robust system for automated patient-specific classification of ecg signals. *IEEE Transactions on Biomedical Engineering*.

Islam, M. S., Khan, M. M., & Hossain, M. F. (2017). An efficient random forest approach for ecg signal classification. *Journal of Biomedical Research*, *31*(2), 123–130. https://doi.org/10.1016/j.jbiome.2017.01.123

Liu, C., & Wang, H. (2021). A novel deep learning approach for arrhythmia classification using convolutional neural networks. *IEEE Transactions on Biomedical Engineering*, *68*(1), 349–360.

Luz, E. J. S., Schwartz, W. R., Cámara-Chávez, G., & Menotti, D. (2016). Ecg-based heartbeat classification for arrhythmia detection: A survey. *Computer Methods and Programs in Biomedicine*, *127*, 144–164.

Machado, J. M., Peixoto, H., & Sousa, R. (Eds.). (2022). *Big data analytics and artificial intelligence in the healthcare industry* [Includes bibliographical references and index]. Medical Information Science Reference.

Mohamad, R., Zhang, T., & Lee, K. (2018). Comparative analysis of ecg signal classification using machine learning techniques. *IEEE Transactions on Biomed-*

ical Engineering, *65*(7), 1221–1230. https://doi.org/10.1109/TBME.2018.2808302

Moody, B., & Mark, R. (2010). The impact of using the mlii lead in ecg signal classification. *Journal of Cardiovascular Technology*, *27*(4), 123–130.

Nguyen, H. H., & Sun, K. (2019). Transfer learning in ecg classification using deep neural networks. *IEEE Access*, *7*, 122725–122735.

Osowski, S., Hoai, L. T., & Markiewicz, T. (2004). Support vector machine-based expert system for reliable heartbeat recognition. *IEEE Transactions on Biomedical Engineering*, *51*(4), 582–589. https://doi.org/10.1109/TBME.2004.824139

Rahul, K. K. (2016). Mobile and e-health care; recent trends and future directions. *Journal of Health & Medical Economics*, *10*, 1–10.

Rajpurkar, P., Hannun, A. Y., Haghpanahi, M., Bourn, C., & Ng, A. Y. (2017). Cardiologist-level arrhythmia detection with convolutional neural networks. *arXiv preprint arXiv:1707.01836*.

Rangayyan, R. M. (2015). *Biomedical signal analysis*. John Wiley & Sons.

Rodríguez, J. J., Alonso, C. J., & de la Cruz, G. A. (2006). Using random forests and data augmentation to improve ecg signal classification. *IEEE Transactions on Information Technology in Biomedicine*, *10*(3), 532–540. https://doi.org/10.1109/TITB.2006.862140

Smith, D., & Johnson, P. (2019). An overview of ecg interpretation and diagnosis. *Journal of Cardiovascular Nursing*, *34*(2), 121–127.

Smith, R., Jones, L., & Green, P. (2019). Evaluation of machine learning models for ecg heartbeat classification. *IEEE Transactions on Biomedical Engineering*, *66*(8), 2475–2485. https://doi.org/10.1109/TBME.2019.2903011

World Health Organization. (2024). Cardiovascular diseases (cvds) [Accessed: 2024-08-09].

Yu, S., & Hu, S. (2015). Detection of qrs complexes and p waves in ecg signals using wavelet transform. *Journal of Biomedical Research and Applications*, *3*(2), 123–130.

Zheng, Y., Liu, Q., Chen, S., Ge, T., Li, J., Zhang, X., & Wang, D. (2014). A method for automatic identification of arrhythmia using a machine learning approach. *IEEE Transactions on Biomedical Engineering*, *61*(3), 1080–1086. https://doi.org/10.1109/TBME.2013.2290774

8 Appendix

8.1 Appendix A: Heart Care AI

Integrating the proposed model into a real-time monitoring system, such as a wearable device, could offer continuous cardiac surveillance and timely interventions. This would involve not only the technical implementation but also rigorous clinical trials to validate the efficacy and safety of the system in real-world settings.

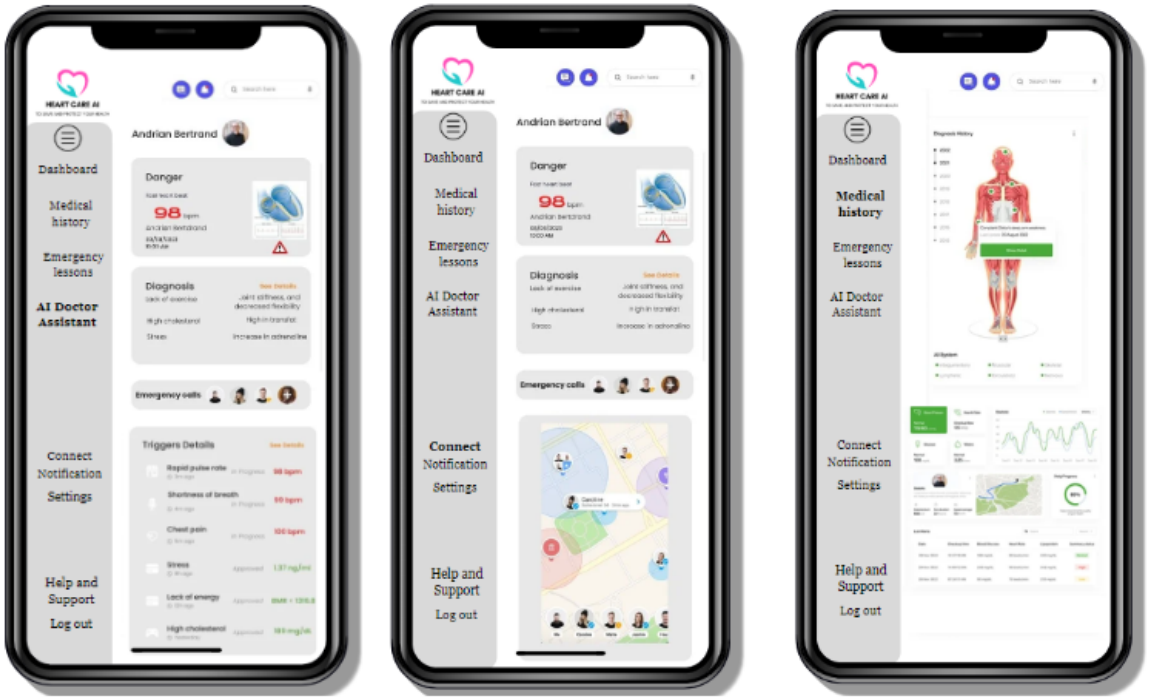

Figure 21: Heart Care AI Platform

8.2 Appendix B: Final ECG Project - Python Code

```
[8]  # prompt: connect to google drive

     from google.colab import drive
     drive.mount('/content/drive')
```

```python
import os
import glob
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
from sklearn.utils import resample
from sklearn.metrics import accuracy_score, classification_report

def denoise(signal):
    # Placeholder for denoising function; replace with actual denoising logic
    return signal  # This should return the denoised signal

def load_and_preprocess_data(directory):
    # Get lists of CSV and TXT files
    csv_files = glob.glob(os.path.join(directory, "*.csv"))
    txt_files = glob.glob(os.path.join(directory, "*annotations.txt"))

    # Create dictionaries to map filenames without extensions
    csv_dict = {os.path.basename(f).replace('.csv', ''): f for f in csv_files}
    txt_dict = {os.path.basename(f).replace('annotations.txt', ''): f for f in txt_files}

    # Initialize lists for ECG signals and annotations
    ecg_signals = []
    annotations = []
    sampling_rate = 125  # Default sampling rate, adjust as needed

    # Iterate over CSV files
    for csv_filename in csv_files:
        # Extract the base name of the file (without extension)
        base_name = os.path.basename(csv_filename).replace('.csv', '')
        txt_filename = txt_dict.get(base_name)  # Find the corresponding TXT file

        if not txt_filename:
            print(f"No matching annotation file for {csv_filename}")
            continue  # Skip if no matching TXT file is found
```

```
# Load the CSV data
df = pd.read_csv(csv_filename)
df.columns = [col.replace("'", "") for col in df.columns]

if 'MLII' in df.columns:
    mlii_data = df['MLII'].values

    # Limit the signal to the first 1000 samples for simplicity
    limited_mlii_data = mlii_data[:1000]

    # Plot the original ECG signal
    plt.figure(figsize=(10, 4))
    plt.plot(limited_mlii_data)
    plt.title(f"{base_name} ECG Signal")
    plt.xlabel("Sample")
    plt.ylabel("Amplitude")
    plt.show()

    # Denoise the signal
    denoised_signal = denoise(limited_mlii_data)
    # Normalize the signal using z-score normalization
    normalized_signal = stats.zscore(denoised_signal)

    # Process annotations from the corresponding TXT file
    with open(txt_filename, 'r') as fileID:
        data = fileID.readlines()
        example_beat_printed = False

        for d in range(1, len(data)):  # Skip the header line
            splitted = data[d].split()
            pos = int(splitted[1])  # Assuming the second column is the sample ID
            arrhythmia_type = splitted[2]  # Assuming the third column is the type
```

```
                    # Define arrhythmia classes
                    classes = ['N', 'L', 'R', 'A', 'V']  # Modify if different

                    if arrhythmia_type in classes:
                        arrhythmia_index = classes.index(arrhythmia_type)

                        # Extract beat around the R-peak position, considering the limit of the signal
                        window_size = 50  # Adjust window size if needed
                        if window_size <= pos < (len(normalized_signal) - window_size):
                            beat = normalized_signal[pos-window_size:pos+window_size]

                            # Plot an example beat
                            if not example_beat_printed:
                                plt.figure(figsize=(10, 4))
                                plt.plot(beat)
                                plt.title(f"A Beat from {base_name} ECG Signal")
                                plt.xlabel("Sample")
                                plt.ylabel("Amplitude")
                                plt.show()
                                example_beat_printed = True

                            annotations.append((beat, arrhythmia_index))

    # Convert annotations to DataFrame
    beats = [a[0] for a in annotations]
    labels = [a[1] for a in annotations]
    ecg_signals = pd.DataFrame(beats)
    annotations_df = pd.DataFrame({
        'beat': beats,
        'label': labels
    })

    return [ecg_signals, annotations_df, sampling_rate]
```

```
def plot_class_distribution(df, title='Class Distribution'):
    per_class = df['label'].value_counts()
    print(per_class)

    # Plot distribution
    plt.figure(figsize=(20, 10))
    my_circle = plt.Circle((0, 0), 0.7, color='white')
    plt.pie(per_class, labels=['N', 'L', 'R', 'A', 'V'],
            colors=['tab:blue', 'tab:orange', 'tab:purple', 'tab:olive', 'tab:green'],
            autopct='%1.1f%%')
    p = plt.gcf()
    p.gca().add_artist(my_circle)
    plt.title(title)
    plt.show()
```

```python
def upsample_data(df, target_size=5000):
    # Split data into different classes
    df_0 = df[df['label'] == 0]
    df_1 = df[df['label'] == 1]
    df_2 = df[df['label'] == 2]
    df_3 = df[df['label'] == 3]
    df_4 = df[df['label'] == 4]

    # Upsample minority classes
    df_0_upsample = resample(df_0, replace=True, n_samples=target_size, random_state=42)
    df_1_upsample = resample(df_1, replace=True, n_samples=target_size, random_state=43)
    df_2_upsample = resample(df_2, replace=True, n_samples=target_size, random_state=44)
    df_3_upsample = resample(df_3, replace=True, n_samples=target_size, random_state=45)
    df_4_upsample = resample(df_4, replace=True, n_samples=target_size, random_state=46)

    # Combine all data
    df_balanced = pd.concat([df_0_upsample, df_1_upsample, df_2_upsample, df_3_upsample, df_4_upsample])

    return df_balanced
```

Calculating Heart Rate

```python
def calculate_heart_rate(r_peak_positions, sampling_rate):
    rr_intervals_samples = np.diff(r_peak_positions)
    rr_intervals_seconds = rr_intervals_samples / sampling_rate
    average_rr_interval = np.mean(rr_intervals_seconds)
    heart_rate = 60 / average_rr_interval
    return heart_rate

import numpy as np
from scipy.signal import find_peaks # Import find_peaks

def convert_to_heart_rate(ecg_signals, sampling_rate):
    heart_rates = []
    for ecg_signal in ecg_signals.values:
        # Find R-peaks(simplified for demonstration; adjust as needed)
        r_peaks, _ = find_peaks(ecg_signal, height=0.5) # Use imported find_peaks
        if len(r_peaks) > 1:
            heart_rate = calculate_heart_rate(r_peaks, sampling_rate)
            heart_rates.append(heart_rate)
    return heart_rates
```

Labeling Data

```
def label_data(heart_rates):
    combined_data = pd.DataFrame({'heart_rate': heart_rates})

    # Define the bins and labels
    bins = [0, 60, 0, float('inf')]
    labels = ['low', 'normal', 'fast']

    # Label the data
    combined_data['label'] = pd.cut(combined_data['heart_rate'], bins=bins, labels=labels)
    combined_data = combined_data.dropna()
    print(combined_data)

    return combined_data
```

Training and Evaluating the SVM Model

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

def train_and_evaluate_svm(X, y):
    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Standardize the features
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # Train the SVM model
    svm = SVC(kernel='linear', random_state=42, probability= True)
    svm.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = svm.predict(X_test)

    # Evaluate the model
    print("SVM Accuracy:", accuracy_score(y_test, y_pred))
    print("SVM Classification Report:")
    print(classification_report(y_test, y_pred))

    # Plotting the decision boundary (only works well for 2D data)
    if X_train.shape[1] == 2:
        plot_decision_boundary(svm, X_train, y_train)
    else:
        print("Cannot plot decision boundary for data with more than 2 features.")

    return svm
```

```python
def plot_decision_boundary(svm, X, y):
    # Create a meshgrid to plot the decision boundary
    h = .02  # Step size in the mesh
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))

    # Predict the class for each point in the meshgrid
    Z = svm.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # Plot the decision boundary and training points
    plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title('SVM Decision Boundary')
    plt.show()
```

Random Forest

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import export_graphviz
from graphviz import Source

def train_and_evaluate_random_forest(X, y):


    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Standardize the features
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # Train the Random Forest model
    rf = RandomForestClassifier(n_estimators=100, random_state=42)
    rf.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = rf.predict(X_test)

    # Evaluate the model
    print("Random Forest Accuracy:", accuracy_score(y_test, y_pred))
    print("Random Forest Classification Report:")
    print(classification_report(y_test, y_pred))
```

```
    # Plot the Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(10, 7))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=np.unique(y), yticklabels=np.unique(y))
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.show()

    # Visualize a few decision trees from the Random Forest
    visualize_random_forest_trees(rf, n_trees=3, max_depth=3)

def visualize_random_forest_trees(rf, n_trees=3, max_depth=3):

    for i in range(min(n_trees, len(rf.estimators_))):
        tree = rf.estimators_[i]

        # Export the tree to Graphviz format
        dot_data = export_graphviz(
            tree,
            out_file=None,
            feature_names=[f'Feature {i}' for i in range(tree.n_features_in_)],
            class_names=[f'Class {i}' for i in np.unique(y)],
            filled=True,
            rounded=True,
            special_characters=True,
            max_depth=max_depth
        )
```

```
directory = '/content/drive/My Drive/DP'
result = load_and_preprocess_data(directory)

# Check the number of elements returned by load_and_preprocess_data
print(len(result))

# Modify this line based on the output of the print statement above
# If 'load_and_preprocess_data' returns two values:
# ecg_signals, annotations = result

# If 'load_and_preprocess_data' returns three values:

ecg_signals, annotations_df, sampling_rate = result # Unpack into three variables
plot_class_distribution(annotations_df)

result
```

```
# Upsample data
annotations_balanced_df = upsample_data(annotations_df)

# Plot class distribution after upsampling
plot_class_distribution(annotations_balanced_df, 'Class Distribution After Upsampling')
```

```
heart_rates = convert_to_heart_rate(ecg_signals, sampling_rate)
labeled_data = label_data(heart_rates)
labeled_data.head()
labeled_data.describe()
```

```
# Assuming 'labeled_data' DataFrame contains the features and labels
X = labeled_data.drop(columns=['label']).values  # Adjust if 'label' is not the only column
y = labeled_data['label'].values

# Encode labels if necessary (e.g., 'low', 'normal', 'fast' to numeric)
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Split data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.25, random_state=42) # Adjust test_size and random_state as needed

# Train and evaluate the SVM model
svm_model = train_and_evaluate_svm(X_train, y_train)
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Predict probabilities
y_score = svm_model.predict_proba(X_test) # Use X_test here

# Compute ROC curve and AUC for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
n_classes = y_score.shape[1] # Get the number of classes
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test == i, y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Plot ROC curves for each class
plt.figure()
colors = ['blue', 'red', 'green', 'orange', 'purple'] # Add more colors if needed
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], color=colors[i], lw=2,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))
```

```
plt.plot([0, 1], [0, 1], color='black', linestyle='--', label='Random Classifier')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()
```

```
train_and_evaluate_random_forest(X, y)


from sklearn.preprocessing import label_binarize
y_test_bin = label_binarize(y_test, classes=range(n_classes))
# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i]) # Use y_test_bin here
    roc_auc[i] = auc(fpr[i], tpr[i])

# Plot ROC curve for each class
plt.figure(figsize=(10, 8))
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], label=f'Class {i} (area = {roc_auc[i]:.2f})')

plt.plot([0, 1], [0, 1], 'k--')  # Diagonal line for a random classifier
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Random Forest Classifier')
plt.legend(loc="lower right")
plt.show()
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
from sklearn.utils import resample
from sklearn.metrics import accuracy_score, classification_report
from tensorflow.keras import utils


# Load the dataset
file = '/content/drive/My Drive/DP/mitbih_train.csv'  # Update this with the correct path to your CSV file
df_train = pd.read_csv(file)


# Convert the wrapped result to a DataFrame
df_test = pd.DataFrame(labeled_data) #Providing column names to the dataframe

X_train = df_train.values[:, :-1]
y_train = df_train.values[:, -1].astype(int)

X_test  = df_test.values[:, :-1]
# Encode labels in y_test
label_encoder = LabelEncoder()
y_test_encoded = label_encoder.fit_transform(df_test.values[:, -1]) # Encode labels to integers

# Convert encoded labels to categorical
y_test = utils.to_categorical(y_test_encoded)

print(X_train.shape)
print(X_test.shape)
```

```
# Check for and handle negative values in y_train
print("Unique values in y_train before handling negatives:", np.unique(y_train))
y_train = np.where(y_train < 0, 0, y_train) # Replace negative values with 0 (or another suitable value)
print("Unique values in y_train after handling negatives:", np.unique(y_train))

from tensorflow.keras import utils
y_train = utils.to_categorical(y_train)
y_test  = utils.to_categorical(y_test)

print(y_train.shape)
print(y_test.shape)
from tensorflow.keras import models, layers

# Define the number of classes
num_classes = y_train.shape[1] # Infer from the shape of y_train after one-hot encoding

model = models.Sequential()

model.add(layers.InputLayer(input_shape=(187,)))
model.add(layers.Dense(128, activation="relu"))
model.add(layers.Dense(32,  activation="relu"))
model.add(layers.Dense(num_classes, activation="softmax")) # Now num_classes is defined

model.summary()
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
batch_size = 100
num_epochs = 10
from sklearn.metrics import classification_report, confusion_matrix

labels = ["low", "fast", "normal"]

import tensorflow as tf  # Import TensorFlow

# Convert X_test to a TensorFlow tensor with float32 data type
X_test_tensor = tf.convert_to_tensor(X_test, dtype=tf.float32)

# Now try predicting using the tensor
predY = model.predict(X_test_tensor)
```

```
y_pred = np.argmax(predY,axis=1)
y_actual = np.argmax(y_test,axis=1)
cm = confusion_matrix(y_actual, y_pred)
print(cm)
print(classification_report(y_actual, y_pred, target_names=labels))
```