Final Project - ECG Arrhythmia Detection

The **electrocardiogram (ECG)** is a fundamental diagnostic tool used to measure the electrical activity of the heart. It plays a crucial role in identifying and diagnosing cardiac arrhythmias conditions where the heart beats too fast, too slow, or irregularly. Accurate detection and classification of arrhythmias are essential for timely medical intervention.

This project is designed to guide the complete process of ECG signal analysis using the **MIT-BIH Arrhythmia Database**. The objective is to develop a practical and efficient system for preprocessing, feature extraction, and classification of ECG signals using machine learning techniques. The project is implemented in **Python** due to its simplicity, readability, and the availability of a wide range of scientific and machine learning libraries that support efficient signal processing and data analysis.

The following Python libraries are recommended for implementation:

- **wfdb** for reading and handling MIT-BIH data.
- **matplotlib** and **pandas** for data visualization and feature engineering.
- **NumPy** and **SciPy** and for numerical computations and signal processing.
- **scikit-learn** for implementing machine learning models.

The project is structured into **four main sections**, each addressing a specific phase of ECG analysis:

1. Dataset Exploration and Visualization
2. Signal Preprocessing
3. Feature Extraction
4. Arrhythmia Detection and Classification

Each section includes a description and a set of implementation requirements. All parts should be documented clearly with code, output, and brief commentary or interpretation.

1. Dataset Exploration and Visualization

The purpose of this section is to familiarize with the MIT-BIH Arrhythmia Database and to explore the structure, format, and content of the ECG recordings. This includes loading ECG signals, identifying the recording channels, and inspecting annotated heartbeat events. Visualization of the signals alongside annotation marks is essential for understanding the nature of the data and for verifying subsequent signal processing steps.

Requirements:

- Download the dataset (full ZIP archive) from the following official source: MIT-BIH Arrhythmia Database - PhysioNet Archive (.zip)
- Unzip the downloaded file and locate the records 100, 101, and 102 (each record consists of a .dat, .hea, and .atr file).
- Load each record using the **wfdb** Python library:
  - Read the ECG signals.
  - Extract metadata like sampling frequency, channel names, and recording duration.
- Plot a 10-second segment from one channel (e.g., MLII) of the ECG signal:
  - Convert sample indices to time in seconds.
  - Label axes clearly (Time [s], Amplitude [mV]).
- Load and overlay heartbeat annotations:
  - Extract annotation symbols and their corresponding sample locations.
  - Superimpose these annotations on the ECG plot using colored markers.
  - Include a legend indicating the meaning of each annotation symbol.
- Summarize the distribution of different heartbeat types present in the visualized segment.

2. Signal Preprocessing and Noise Removal

This section focuses on improving the quality of the raw ECG signal through several preprocessing techniques. These steps help remove various types of noise and artifacts that commonly affect ECG recordings, allowing for more accurate downstream analysis such as peak detection and classification. It is required to implemet:

a. **Baseline Wander Removal:**
Low-frequency noise, typically due to patient movement or respiration, is removed by applying a high-pass filter that attenuates frequencies below 0.5 Hz.

b. **Power Line Interference Removal:**
ECG signals often contain 50 Hz (or 60 Hz depending on region) interference from electrical power sources. A notch filter is designed to suppress this interference without affecting the rest of the signal.

c. **Noise Reduction and Smoothing:**
General high-frequency noise is suppressed using a low-pass filter or a band-pass filter, preserving relevant signal components while improving signal-to-noise ratio.

Note: Each filtering step is visualized to compare the original and the cleaned signal.

Requirements:

- Implement baseline wander removal using Fourier Transform.
- Implement a notch filter to eliminate 50 Hz powerline interference.
- Apply smoothing or bandpass filtering to reduce high-frequency noise while preserving ECG morphology.
- Plot and compare the signal before and after each preprocessing step.
- Comment on the observed differences in signal shape and quality.

3. R-Peak Detection and Heart Rate Calculation

This section focuses on detecting the R-peaks in the ECG signal and using them to compute important cardiac features. The R-peak corresponds to the peak of the QRS complex and is the most prominent feature in a typical ECG cycle. By detecting these peaks, it is possible to calculate the **RR intervals** (the time between successive R-peaks), which can then be used to estimate the **heart rate** and observe its variation over time.

This analysis is essential for identifying irregularities such as **arrhythmias** and forms the basis of heart rate variability studies.

Requirements:

- Use the preprocessed ECG signal obtained from Section 2.
- Implement a peak detection method (such as using a threshold, scipy.signal.find_peaks, or a simple derivative-based method) to detect R-peaks.
- Plot the detected R-peaks on the ECG signal to verify their accuracy.
- Calculate the RR intervals and visualize them using a histogram or line plot.
- Compute the average heart rate from the RR intervals and report the value.
- Provide observations regarding heart rate variability and any abnormalities (e.g., bradycardia or tachycardia) if present.

4. Arrhythmia Detection and Classification

The primary objective of this section is to classify the ECG signal into **normal** or **abnormal** categories based on the heart rate and RR intervals obtained in the previous section.
By analyzing the heart rate patterns and the variability between RR intervals, one can classify the ECG as either "Normal" or indicative of an **arrhythmia**. Arrhythmias are abnormalities in the heart's rhythm and can be categorized into different types based on the detected irregularities.

The heart rate variability (HRV) and RR interval analysis form the foundation for detecting certain types of arrhythmias. Based on this, the project will implement a basic classification model (either a simple thresholding method or a machine learning classifier such as decision trees or random forests) to detect the presence of arrhythmias.

Optional Learning Resource: For an introduction to machine learning with **Scikit-learn**, refer to the Official Scikit-learn tutorial or YouTube: Machine Learning with Scikit-learn or any available resources.

Requirements:

- Based on the average heart rate computed in Section 3, determine whether the heart rate is normal (between 60-100 beats per minute).
- If the heart rate is abnormal (either too high or too low), label the ECG as potentially indicating **arrhythmia**.
- Implement a machine learning classifier (e.g., logistic regression, decision tree, or random forest) to classify the ECG signal into **normal** and **abnormal** categories. You can use extracted features from the RR intervals and heart rate data.
- Apply the classifier to a set of labeled test data (from the MIT-BIH database) to assess the performance (accuracy, precision, recall).
- Provide a detailed analysis of classification results, including any false positives or false negatives, and comment on the accuracy of arrhythmia detection.
- Display a confusion matrix and other relevant performance metrics (such as ROC curve, F1 score, etc.) to evaluate the effectiveness of the classifier.

Submission regulations:

1. The assignment is to be completed in groups of 2-3 students.
2. Each group should submit a report including the code, screenshots for the output and suitable comments (report can be submitted as a Jupyter notebook or as a collection of code snippets, with all necessary documentation provided for clarity).
3. The submission deadline is on **Friday, May 16th, at 11:59 PM**.
4. Submissions should be made through this link: Project Submission Form.
5. Copied reports will take zero.