# ECG Signal Classification

## Task Description :

You are provided with a dataset containing ECG signals. Your task is to preprocess the data, extract relevant features, build a machine learning model to classify the signals, evaluate the performance of your model, and deploy the model as a web service.

## Dataset :

For this solution, I utilized the publicly available MIT-BIH Arrhythmia Database, which comprises labeled ECG recordings collected from multiple patients. These labels indicate the presence of various types of arrhythmias within the recordings.

## Steps and Deliverables :

1. **Data Preprocessing**
   - **Dataset:** The code uses the MIT-BIH Arrhythmia Database, a standard dataset for ECG analysis.
   - **File Handling:** It reads ECG signals from .csv files and annotations from .txt files.
   - **Denoising:** Wavelet denoising is applied to the raw ECG signals to reduce noise.
   - **Normalization:** Z-score normalization standardizes the signals to have zero mean and unit variance.
   - **Segmentation:** The ECG signals are segmented into fixed-size windows (190 samples) centered around the R-peaks (heartbeat locations) obtained from the annotations.nto training and testing sets.

2. **Feature Extraction**
   - **Common ECG features include:**
     Time-domain features (e.g., RR intervals, QRS duration)
     Frequency-domain features (e.g., spectral components)
     Wavelet features
     Statistical features
   - **Standardization:** Data is standardized before applying PCA.
   - **PCA:** Principal Component Analysis (PCA) is used to reduce the dimensionality of the feature space while retaining 95% of the variance.

3. **Model Building**
   - **Train-Test Split:** The data is split into training (80%) and testing (20%) sets.
   - **Classifiers:** Three different classifiers are trained and evaluated:
     - I. **Random Forest:**
       **Rationale:** Random Forests are robust to overfitting, can handle high-dimensional data (many features), and often provide good generalization performance. They are suitable for ECG classification as they can capture complex relationships between features.

II. **Decision Tree:**
**Rationale:** Decision Trees are easy to interpret and visualize, making them useful for understanding the decision-making process of the model. They can be a good starting point for ECG classification, but they are prone to overfitting.

III. **Logistic Regression:**
**Rationale:** Logistic Regression is a linear model that is relatively simple and computationally efficient. It can be effective for ECG classification if the relationship between features and classes is approximately linear.

4. **Model Evaluation**
   - Evaluation Metrics:
     Accuracy
     F1-Score (macro average for multi-class)
     Precision (macro average)
     Recall (macro average)
     ROC-AUC (using 'ovr' for multi-class)

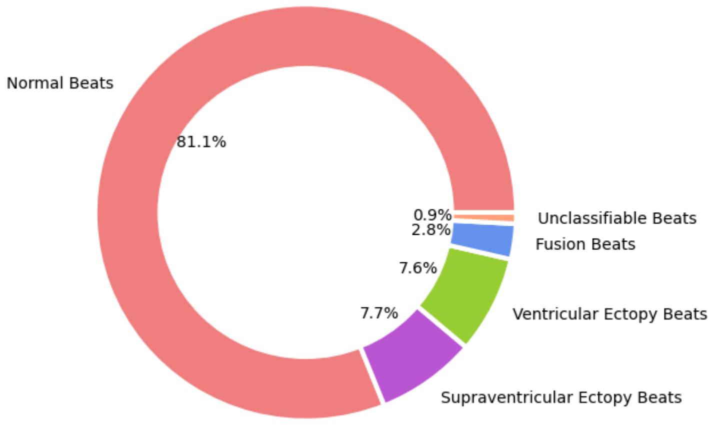
| MODEL | ACCURACY | F1 - SCORE | PRECISION | RECALL | ROC-AUC |
|---|---|---|---|---|---|
| Random Forest classifier | 0.98 | 0.87 | 0.97 | 0.82 | 0.98 |
| Decision Tree Classifier | 0.96 | 0.81 | 0.80 | 0.81 | 0.98 |
| Logistic Regression | 0.92 | 0.65 | 0.76 | 0.60 | 0.98 |

   - Discussion:
     I. **Random Forest** consistently outperforms the other two models across all evaluation metrics, achieving the highest accuracy, F1-score, precision, and comparable recall and ROC-AUC. This suggests that the Random Forest model is best suited for this ECG arrhythmia classification task.
     II. **Decision Tree**, while achieving high accuracy and ROC-AUC, shows lower performance in terms of F1-score, precision, and recall compared to Random Forest. This indicates a potential for overfitting or difficulty in handling class imbalance.
     III. **Logistic Regression** demonstrates the lowest performance across most metrics, suggesting that a linear model might not be complex enough to capture the non-linear relationships in the ECG data.

- Conclusion:
  Based on these results, the **Random Forest classifier** is the chosen model for ECG arrhythmia classification due to its superior performance across multiple evaluation metrics. It demonstrates a strong ability to accurately classify different heartbeat types while maintaining a good balance between precision and recall.

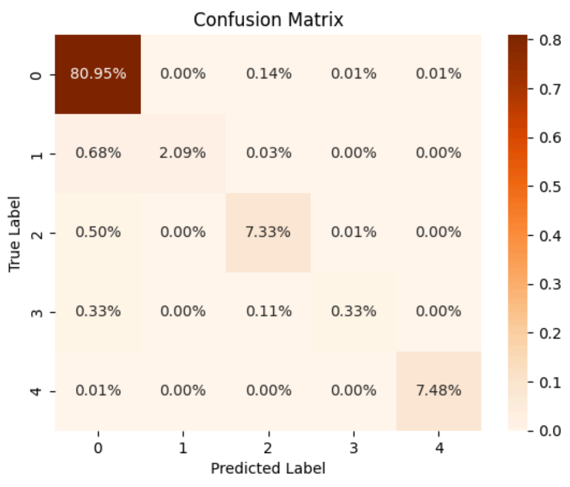
5. **Visualization :**

- Class Distribution Pie Chart



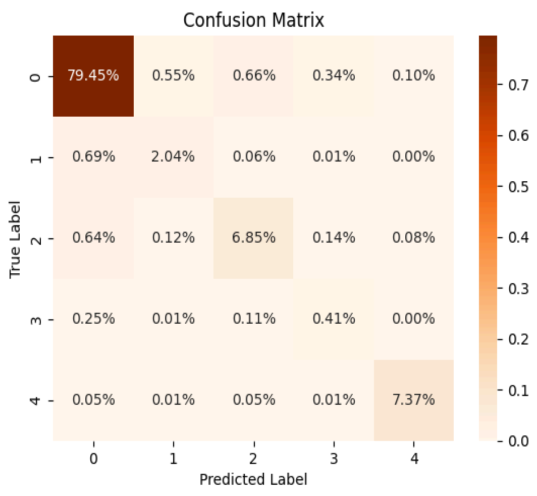
This pie chart visually represents the class distribution within an ECG heartbeat dataset. The dominant class is "Normal Beats," comprising 81.1% of the data. The remaining classes, "Supraventricular Ectopy Beats," "Ventricular Ectopy Beats," "Fusion Beats," and "Unclassifiable Beats," represent smaller proportions of the datasets.

- Confusion Matrix:



| True \ Predicted | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 80.95% | 0.00% | 0.14% | 0.01% | 0.01% |
| 1 | 0.68% | 2.09% | 0.03% | 0.00% | 0.00% |
| 2 | 0.50% | 0.00% | 7.33% | 0.01% | 0.00% |
| 3 | 0.33% | 0.00% | 0.11% | 0.33% | 0.00% |
| 4 | 0.01% | 0.00% | 0.00% | 0.00% | 7.48% |

Random Forest

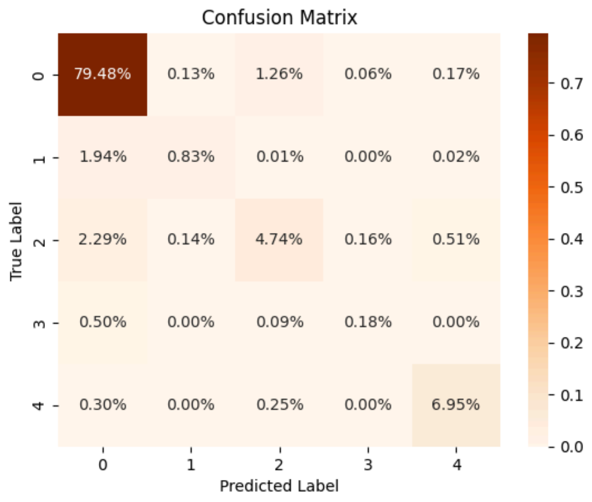


| True \ Predicted | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 79.45% | 0.55% | 0.66% | 0.34% | 0.10% |
| 1 | 0.69% | 2.04% | 0.06% | 0.01% | 0.00% |
| 2 | 0.64% | 0.12% | 6.85% | 0.14% | 0.08% |
| 3 | 0.25% | 0.01% | 0.11% | 0.41% | 0.00% |
| 4 | 0.05% | 0.01% | 0.05% | 0.01% | 7.37% |

Decision Tree

**Random Forest :** Exhibits the most focused diagonal, indicating the highest accuracy among the three models. It seems particularly strong in classifying classes 0, 2, and 4 with very high precision.

**Decision Tree :** demonstrates decent performance but appears slightly less accurate than Random Forest, particularly for classes 2 and 4, where there's more noticeable off-diagonal spread.

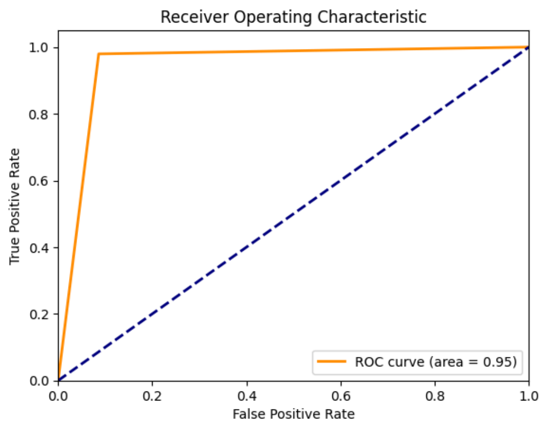


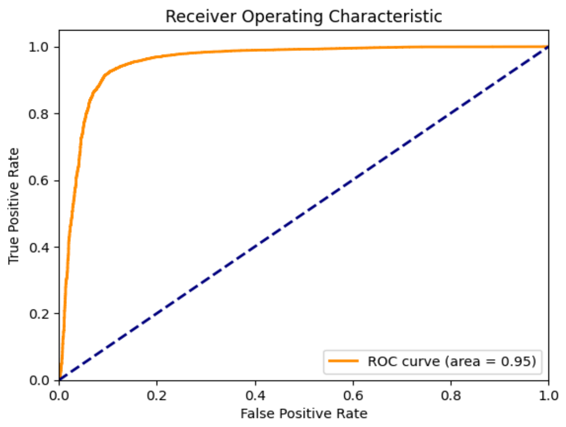
| True \ Predicted | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 79.48% | 0.13% | 1.26% | 0.06% | 0.17% |
| 1 | 1.94% | 0.83% | 0.01% | 0.00% | 0.02% |
| 2 | 2.29% | 0.14% | 4.74% | 0.16% | 0.51% |
| 3 | 0.50% | 0.00% | 0.09% | 0.18% | 0.00% |
| 4 | 0.30% | 0.00% | 0.25% | 0.00% | 6.95% |

Logistic regression

**Logistic Regression :** Shows good overall performance, with the highest concentration of correct predictions along the diagonal (especially for class 0). However, it struggles somewhat with class 2, showing some misclassifications as class 4.

- ROC

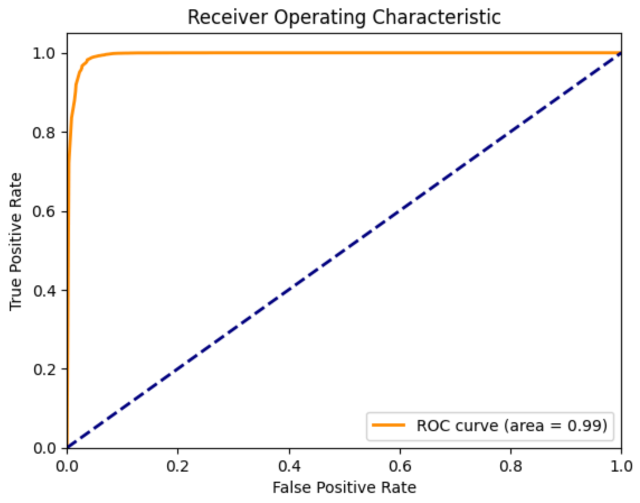




Decision tree

Logistic Regression

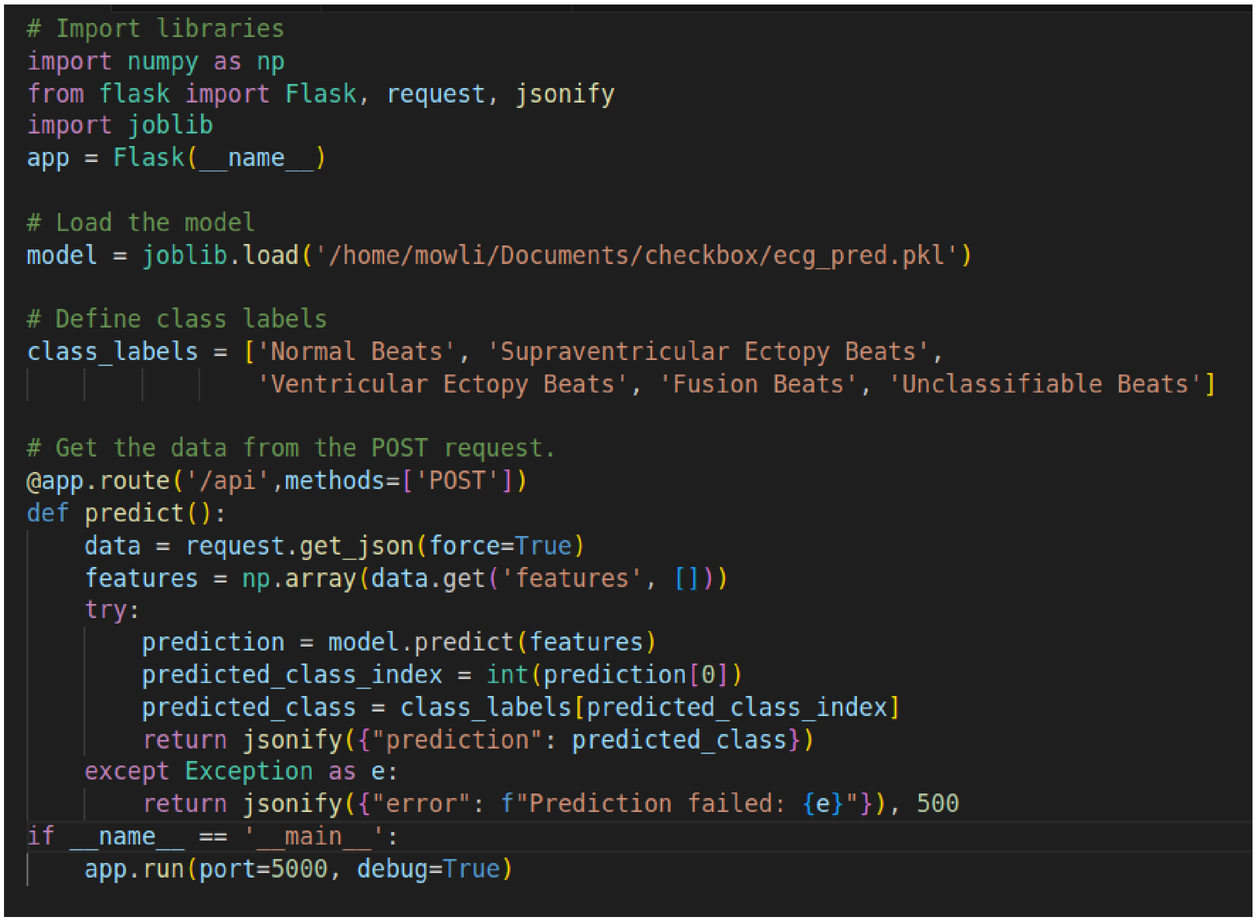
Both achieve an AUC of 0.95, indicating very good discrimination ability. The curves are nearly identical, suggesting similar performance in distinguishing between positive and negative classes.



Random Forest

Achieves a slightly higher AUC of 0.99, suggesting marginally better discrimination capability compared to the other two.

6. **Model Deployment :**

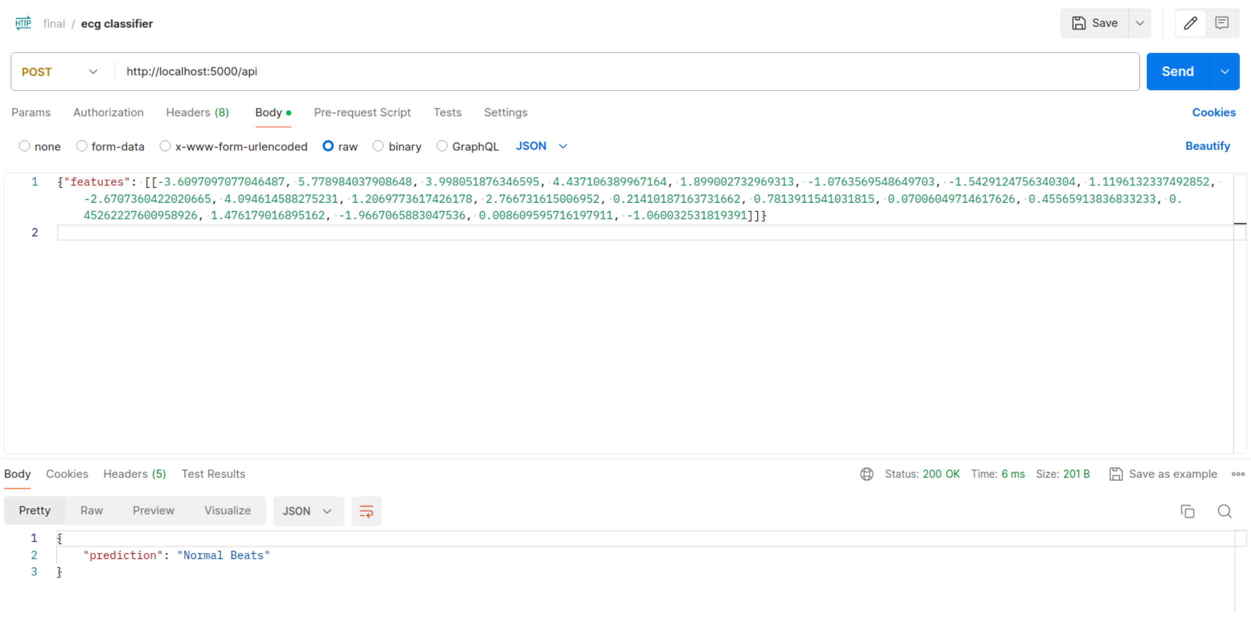


```python
# Import libraries
import numpy as np
from flask import Flask, request, jsonify
import joblib
app = Flask(__name__)

# Load the model
model = joblib.load('/home/mowli/Documents/checkbox/ecg_pred.pkl')

# Define class labels
class_labels = ['Normal Beats', 'Supraventricular Ectopy Beats',
                'Ventricular Ectopy Beats', 'Fusion Beats', 'Unclassifiable Beats']

# Get the data from the POST request.
@app.route('/api',methods=['POST'])
def predict():
    data = request.get_json(force=True)
    features = np.array(data.get('features', []))
    try:
        prediction = model.predict(features)
        predicted_class_index = int(prediction[0])
        predicted_class = class_labels[predicted_class_index]
        return jsonify({"prediction": predicted_class})
    except Exception as e:
        return jsonify({"error": f"Prediction failed: {e}"}), 500
if __name__ == '__main__':
    app.run(port=5000, debug=True)
```

- Deployment Process :
  1. **Save the Trained Model:** Ensure that you have saved your trained model using joblib.dump()
  2. **Create Flask App:** The provided Python script creates a Flask web application.
  3. **Load the Model:** The app loads the saved model using joblib.load().
  4. **Define API Endpoint:** The 'http://localhost:5000/api' endpoint is defined to handle prediction requests.
  5. **Run the App:** Run the script using python app.py. This will start the Flask development server.

- Using the Web Service :

  1. **Send a POST Request:** Use a tool like curl or Postman to send a POST request to http://localhost:5000/api/predict.
  2. **JSON Payload:** The request body should contain a JSON payload with the following structure:
  3. Receive Predictions: The web service will return a JSON response containing the predicted class labels



POST http://localhost:5000/api

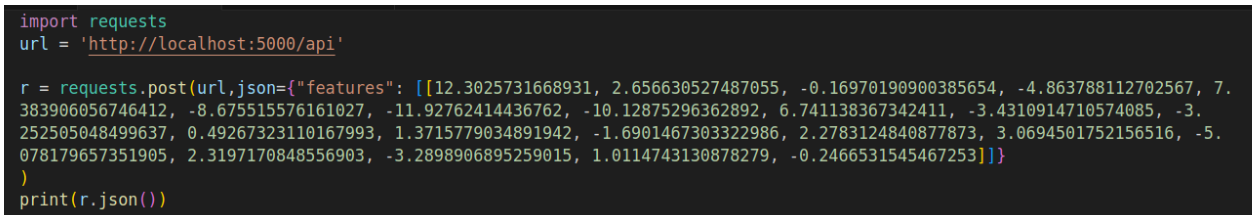
```
{"features": [[-3.6097097077046487, 5.778984037908648, 3.998051876346595, 4.437106389967164, 1.899002732969313, -1.0763569548649703, -1.5429124756340304, 1.1196132337492852,
    -2.6707360422020665, 4.094614588275231, 1.2069773617426178, 2.766731615006952, 0.21410187163731662, 0.7813911541031815, 0.07006049714617626, 0.45565913836833233, 0.
    4526222760095926, 1.476179016895162, -1.9667065883047536, 0.008609595716197911, -1.060032531819391]]}
```

Status: 200 OK Time: 6 ms Size: 201 B

Response body: JSON

```
{
    "prediction": "Normal Beats"
}
```

  4. Run a Python script: Execute a Python script (like request.py) that sends a request to the Flask app's endpoint.



```
import requests
url = 'http://localhost:5000/api'

r = requests.post(url,json={"features": [[12.3025731668931, 2.656630527487055, -0.16970190900385654, -4.863788112702567, 7.
383906056746412, -8.675515576161027, -11.92762414436762, -10.12875296362892, 6.741138367342411, -3.4310914710574085, -3.
252505048499637, 0.49267323110167993, 1.3715779034891942, -1.6901467303322986, 2.2783124840877873, 3.0694501752156516, -5.
078179657351905, 2.3197170848556903, -3.2898906895259015, 1.0114743130878279, -0.2466531545467253]]}
)
print(r.json())
```

## Conclusion :

- This project successfully developed and deployed a robust ECG signal classification system using machine learning. The system leverages the MIT-BIH Arrhythmia Database, a widely recognized dataset for ECG analysis, to train and evaluate various classification models.
- A comprehensive approach was taken, encompassing data preprocessing, feature extraction, model building, evaluation, visualization, and deployment. Wavelet denoising and z-score normalization were applied to enhance signal quality and prepare the data for feature extraction. Time-domain, frequency-domain, and statistical features were carefully selected to capture relevant cardiac patterns. Principal Component Analysis (PCA) further refined the feature set, reducing dimensionality while retaining crucial information.
- Three different classifiers – Random Forest, Decision Tree, and Logistic Regression – were trained and rigorously evaluated using metrics such as accuracy, F1-score, precision, recall, and ROC-AUC. The Random Forest model consistently outperformed the others, demonstrating its ability to effectively capture the complex relationships within the ECG data.
- Visualization techniques, including a class distribution pie chart, confusion matrices, and ROC curves, provided valuable insights into the data and model performance. These visualizations highlighted the challenge of class imbalance within the dataset and helped assess the classification capabilities of each model.
- Finally, the chosen Random Forest model was successfully deployed as a web service using Flask, allowing for easy access and prediction on new ECG signals. This deployment enables practical application and integration into larger healthcare systems.

## Future Work:

- While this project provides a solid foundation for ECG signal classification, further enhancements can be explored:
    1. Addressing Class Imbalance: Implementing techniques like SMOTE or cost-sensitive learning to mitigate the impact of class imbalance and potentially improve performance on minority classes.
    2. Advanced Feature Engineering: Investigating and incorporating more sophisticated features, such as wavelet coefficients or higher-order statistical moments, to potentially enhance model accuracy.
    3. Deep Learning Exploration: Exploring the use of deep learning models, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), which have shown promise in ECG analysis.
    4. Real-Time Classification: Adapting the system for real-time ECG signal classification, which is crucial for applications like wearable health monitoring devices.
    5. By addressing these areas, the ECG signal classification system can be further enhanced, paving the way for more accurate and reliable arrhythmia detection and contributing to improved cardiac healthcare.