# A Self-Contained ECG Classification and Arrythmia Detection

Submitted in partial fulfillment of the requirements of the degree of Bachelor of Engineering in Electronics and Telecommunication Engineering

**Rahul Bala, Nibin Varghese, Sairaj Parkhe, Sanket Salekar**

Department of Electronics and Telecommunication Engineering SIES GRADUATE SCHOOL OF TECHNOLOGY
Nerul, Navi Mumbai 400706 University of Mumbai

DOI: 10.29322/IJSRP.14.06.2024.p15039

Paper Received Date: 19th June 2024
Paper Acceptance Date: 20th June 2024
Paper Publication Date: 26th June 2024

***Abstract-*** Automated Electrocardiogram (ECG) classification and arrhythmia detection represent a cutting-edge frontier in cardiac diagnostics, poised to revolutionize the identification and management of cardiovascular disorders. Leveraging advanced machine learning methodologies, this report delves into the development and implementation of an automated ECG analysis system. Specifically, the integration of the state-of-the-art EfficientNetB7 model and the versatile Random Forest classifier is investigated for its efficacy in discerning complex patterns within ECG signals and categorizing them into distinct arrhythmia classes.

The EfficientNetB7 model, renowned for its prowess in image recognition tasks, is repurposed to extract features from ECG signals. By treating ECG signals as image-like representations, the model captures nuanced patterns and characteristics indicative of various cardiac abnormalities. Subsequently, the extracted features are fed into a Random Forest classifier, known for its robustness and interpretability, for classification into clinically relevant arrhythmia categories.

This approach harnesses the power of deep learning for feature extraction and ensemble learning for classification, promising to transcend the limitations of traditional manual interpretation methods. Through an in-depth analysis of theoretical foundations, methodological approaches, and potential applications, this study aims to elucidate the transformative potential of computational techniques in cardiac diagnostics. By offering expedited diagnosis and enhanced diagnostic accuracy, automated ECG analysis opens doors to personalized, data-driven approaches to cardiovascular healthcare, ultimately improving patient outcomes and revolutionizing clinical practice.

## I. Introduction

### 1.1 Introduction and Need of the Project.

E Electrocardiogram (ECG) classification and arrhythmia detection constitute a crucial frontier in modern cardiology, providing indispensable insights into cardiac health and facilitating early intervention for cardiovascular disorders. In this era of rapid technological advancement, the convergence of machine learning and biomedical engineering offers unprecedented opportunities to enhance the efficiency and accuracy of ECG analysis.

The code provided underscores the utilization of advanced machine learning techniques, particularly the integration of the EfficientNetB7 model and the Random Forest classifier, to automate ECG analysis. This amalgamation of state-of-the-art algorithms holds immense promise in discerning intricate patterns within ECG signals and categorizing them into clinically relevant arrhythmia classes.

The EfficientNetB7 model, pre-trained on the ImageNet dataset, epitomizes the pinnacle of deep learning architectures for image recognition tasks. By leveraging its hierarchical feature representations and efficient network scaling techniques, the EfficientNetB7 model demonstrates unparalleled proficiency in extracting discriminative features from images, including ECG signals when repurposed for this task.

In the context of ECG classification and arrhythmia detection, the EfficientNetB7 model serves as a potent tool for feature extraction. By treating ECG signals as image-like representations and employing transfer learning, the model can capture subtle variations and complex patterns indicative of different cardiac abnormalities. Through the process of feature extraction, the model transforms raw ECG data into high-dimensional feature vectors, encoding essential information for subsequent classification.

The integration of the EfficientNetB7 model and the Random Forest classifier represents a formidable approach to automated ECG analysis. By capitalizing on the strengths of deep learning for feature extraction and ensemble learning for classification, this methodology holds promise in revolutionizing cardiac diagnostics. Through the seamless fusion of cutting-edge algorithms and biomedical expertise, automated ECG analysis paves the way for personalized, data-driven approaches to cardiovascular healthcare, ultimately enhancing patient outcomes and advancing the frontier of modern cardiology.

Complementing the feature extraction capabilities of the EfficientNetB7 model, the Random Forest classifier offers a robust and interpretable framework for classification tasks. Encompassing an ensemble of decision trees, the Random Forest classifier harnesses the wisdom of multiple weak learners to achieve robust performance in handling high-dimensional data and mitigating overfitting.

In the context of ECG classification, the Random Forest classifier acts as the final arbiter, leveraging the extracted features to categorize ECG signals into distinct arrhythmia classes. Through its ability to capture non-linear relationships and model complex decision boundaries, the Random Forest classifier demonstrates remarkable efficacy in distinguishing between different cardiac conditions, thereby facilitating accurate diagnosis and timely intervention.

## 1.2 Problem Statement

The problem statement focuses on the difficulties associated with manual ECG signal interpretation, which causes delays in cardiac arrhythmia diagnosis and treatment. The existing reliance on expert analysis restricts the scalability and accessibility of cardiac healthcare services, especially in areas with limited access to specialist medical expertise. Furthermore, manual ECG interpretation takes time and reduces clinical workflow efficiency, potentially leading to missed opportunities for early intervention.

The proposed project aims to solve these problems by creating a self-contained ECG classification and arrhythmia detection system. The system will analyse ECG readings automatically using machine learning and signal processing techniques, allowing for the speedy and precise diagnosis of cardiac problems. This technical innovation holds substantial contributions for society.

These challenges are addressed by the development of a self-contained ECG classification and arrhythmia detection system, which automates signal analysis, improves access to cardiac healthcare, enables timely diagnosis and treatment, streamlines clinical workflows, empowers patients in self-monitoring, and promotes cardiovascular research and innovation. This technology development has the potential to revolutionize cardiac care delivery, ultimately improving patient outcomes and expanding the area of medical science.

## 1.3 Organization of the Report

Chapter 1 of this report outlines the necessity for undertaking the project and proposes an enhanced system to address the shortcomings prevalent in existing systems. It delves into the challenges faced by current systems, highlighting the need for innovation and improvement. By presenting a comprehensive overview of the project's objectives and the proposed solutions, this chapter sets the stage for the subsequent discussions on literature review, investigation, results, and conclusions. Chapter 2 provides a thorough review of existing literature relevant to the project. It explores previous research, methodologies, and technologies employed in similar endeavors. This section serves to contextualize the project within the broader academic and professional landscape, identifying gaps in knowledge and areas for further exploration. By synthesizing the findings from various sources, this chapter lays the groundwork for the subsequent phases of the project.

Chapter 3 offers a detailed account of the present investigation, including the methodologies, software utilized, and libraries installed during the project's execution. It provides insights into the practical aspects of implementing the proposed system, outlining the workflow, tools, and techniques employed. This section elucidates the project's working mechanisms, enabling readers to grasp the technical intricacies involved in its development. Chapter 4 presents the results obtained from the project and engages in a comprehensive discussion thereof. It analyzes the findings in relation to the project's objectives, drawing conclusions based on empirical evidence. Additionally, this chapter explores the implications of the results, addressing their significance within the broader context of the field. Through critical examination and interpretation, it offers valuable insights into the project's outcomes.

Chapter 5 encapsulates the conclusions drawn from the project's findings and outlines potential avenues for future research and development. It synthesizes the key insights gleaned throughout the report, reaffirming the significance of the project's contributions.

Moreover, this chapter delineates prospective areas for expansion, innovation, and improvement, charting a course for continued advancement in the field.

# Chapter 2 Literature Survey

Table 2.1: Literature Survey of Different Papers

| **Published Year** | **Title** | **Data Set** | **Methodology** | **Algorithms Used** | **Metrics** | **Result** |
|---|---|---|---|---|---|---|
| 2020 | Quantized Convolutional Neural Network toward Real-time Arrhythmia Detection in Edge Device | MIT BIH Arrhythmia | 1. Data set is used for training and testing.<br>2. Five standard metrics were used to measure the performance of the model. | a. 10-CNN<br>b. Long Short-Term Memory (LSTM)<br>c. Bidirectional recurrent Neural Network | • accuracy<br>• sensitivity<br>• specificity<br>• precision<br>• fl-score | 1. The CNN model was found to be faster than all the other algorithms used.<br>2. Jetson Nano was found to be significantly faster than Raspberry Pi |
| 2020 | ECG Heartbeat classification using CNN | MIT BIH Arrhythmia | 1. Pre – processing<br>2. Feature Extraction<br>3. Then the signal is made to pass through ReLu, Pooling layers, Fully Connected layer and ADAM Optimizer | a. Kalman filter, Bayesian filter<br>b. Beat by Beat Analysis<br>c. SVM classifier with hermite transform. | • Precision<br>• Fl scores<br>• false-positive data (Fp)<br>• false-negative data (Fn)<br>• true-negative (Tn) | 1. This model has achieved 97.17% overall accuracy.<br>2. After 5-fold cross validation the presented algorithm achieves an accuracy of 97.36% and fl score of 99.83% |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2020 | Technological Analysis of ECG Classification based on Machine Learning and Deep Learning Techniques | MIT BIH Arrhythmia | 1. Review of technology - es<br>2. Pre-Processing<br>3. Feature Extraction | a. Genetic algorithm and Wavelet Transform<br>b. Kaiser Window<br>c. Adaptive filter<br>d. FIR and lIR filters | - | 1. There seems to be a wide use of SVM's CNN and ANN as classification techniques.<br>2. It's very clear that wavelet transforms have been favored by many researchers. |
| 2020 | A Review on Arrhythmia Classification Using ECG Signals | MIT BIH Arrhythmia, UCI Machine Learning Repository | 1. Pre-Processing<br>2. Feature Extraction<br>3. Feature dimension reduction/ optimization | a. Support Vector Machine<br>b. Modular Neural Network<br>c. Linear classifiers Deep Neural Networks | • Precision<br>• recall<br>• Fl scores<br>• true-positive data(Tp)<br>• Specificity (S)<br>• Detection Error Rate (DER) | 1. Highest accuracy achieved is 100% with 2 classes of arrhythmia (Normal and abnormal classes)<br>2. Samples considered for testing are very less |
| 2020 | Classification of ECG Arrhythmia With Machine Learning Techniques | MIT BIH Arrhythmia | 1. MIT BIH database is used for training.<br>2. Classificati o n<br>3. Wave transformat i on | a. Back propagation algorithm with Multilayer Perceptron (MLP) Classifier<br>b. kernel - adatron algorithm (K- A) With Support Vector Machine | Interoperability | 1. Using wave transformation, the result obtained was quick. |

Table 2.1: Literature Survey of Different Papers

[1] In the paper Muhammad IIham et al. found that the process of quantization introduces quantization noise where the quantized models may not be able to capture complex patterns and features in the data as effectively as full-precision models. The high computational demands of CNNs can lead to increased energy consumption where the CNNs are often regarded as black- box models, making it challenging to understand why a particular prediction was made.[2] Instead of using manually designed features as most of the existing ECG classification works do, Xuexiang Xu et al. have extracted data-driven non-linear features using convolutional neural network.
CNNs are often considered black-box models, which means it can be challenging to interpret their decisions. They are susceptible to overfitting if not properly regularized or if the dataset is imbalanced or insufficient. They require high-quality, well-labeled training data. Over time, the performance of a trained CNN model may deteriorate due to changes in data distribution.[3] BH Nisal Sudila et al. proposes the analysis of the early statistical approaches made in classification of ECGs to modern techniques. ECG data can be noisy and may require preprocessing and cleaning before it can be used effectively.
Whereas, Deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) can be computationally intensive and require significant computing resources for training and deployment. Approval and regulation of AI-based medical devices can be a lengthy and complex process, which can delay their adoption in clinical practice. Low- quality data can lead to inaccurate classifications.[4] Mohebbanaaz et al. presents a survey on issues concerned in ECG denoising, feature extraction, optimization, and classification. Furthermore, methods used to analyze the performance are also discussed. Many novel algorithms have been proposed for analyzing and classification of arrhythmia.
In some cases, there may be a limited amount of ECG data available for training machine learning models, which can affect the model's ability to generalize to different populations or arrhythmia types. Classifying the classes of arrhythmia becomes difficult. Moreover, performance of a classifier is evaluated based on its accuracy and reliability, but the energy consumption aspect is often neglected. There is a need of study on heat beat arrhythmia problem and provide an extension to database with all classes of arrhythmia. [5] In this paper by Shweta Jambukia et al., the P, Q, R, S, and T waves in ECG signals are classified using some machine learning techniques. In the work to be done, MLP (Multi-Layer Perceptron) and SVM (Support Vector Machine) classification techniques which are not compared with each other using these signals will be compared. It is aimed to develop a method to improve the calculation time and standard classification performance of MLP and SVM, and it is aimed to contribute to the informed consciousness of this work. Training the data set is difficult. Compared to other methods, DFT has the worst similarity rate. The DCT, DWT and DFT methods gave almost the same results when the maximum noise ratio was compared.

## 3.1 Objectives

# Chapter 3

**Report On Present Investigation**

Developing an automated system capable of accurately classifying Electrocardiogram (ECG) signals into distinct arrhythmia categories represents a critical advancement in streamlining diagnostic processes within clinical settings. By meticulously minimizing both false positives and false negatives, this system ensures the delivery of reliable diagnostic outcomes, thereby facilitating prompt and precise diagnosis, which is paramount for effective patient management.

Furthermore, the design of such a solution must intricately balance efficiency and scalability to adeptly handle the vast volumes of ECG data encountered in clinical practice. This entails accommodating diverse workloads and data volumes to maintain optimal performance across varied healthcare environments. In addition to technical robustness, preserving the interpretability of the automated system emerges as a cornerstone for fostering clinical acceptance and trust. Through transparent decision-making processes and intuitively presented outputs, clinicians are empowered to comprehensively understand and validate the recommendations provided by the system, thus enhancing its integration into clinical workflows. Moreover, the development of a solution that effectively generalizes across diverse patient populations, ECG acquisition settings, and hardware platforms is imperative. This ensures its broad applicability and sustained effectiveness in real-world clinical scenarios, transcending the inherent variability in data characteristics encountered across different healthcare contexts.

## 3.2 System Architecture and Design

### 3.2.1. Software/Online Services Used:

In conjunction with Google Colab and Jupyter Notebook, forms the backbone of this project's development environment for ECG classification and arrhythmia detection. Working with Python in Google Colab offers unparalleled access to computational resources, alleviating the constraints posed by local hardware limitations. This cloud-based infrastructure streamlines the workflow, enabling seamless collaboration and facilitating rapid experimentation with various machine learning algorithms and techniques.
Leveraging Python within Google Colab and Jupyter Notebook revolutionizes the approach to ECG classification and arrhythmia detection. Python serves as the primary programming language, offering a vast ecosystem of libraries and tools tailored for machine learning and data analysis tasks. Within Google Colab, Python's compatibility seamlessly integrates with the cloud-based infrastructure, granting us access to scalable computational resources essential for training complex machine learning models.
The collaborative features embedded within Google Colab facilitate seamless teamwork, allowing multiple team members to collaborate on the same notebook in real-time. This enables parallel experimentation and knowledge sharing, enhancing productivity and accelerating the development process. Additionally, Python's versatility enables to integrate various machine learning libraries such as TensorFlow, Scikit-learn, and Keras, empowering to explore diverse algorithms and techniques for ECG classification.
Jupyter Notebook complements the workflow by providing an interactive environment where Python code, visualizations, and explanatory text converge in a single document. This facilitates code readability, reproducibility, and documentation, enhancing the transparency and interpretability of the work. Moreover, Jupyter Notebook's support for Markdown enables users to annotate our code with explanations, insights, and references, fostering clear communication and facilitating collaboration among team members.
Overall, from a technical perspective, Python within Google Colab and Jupyter Notebook streamlines the development process, enabling collaborative exploration, efficient experimentation, and transparent documentation of efforts in advancing ECG classification and arrhythmia detection.

**Libraries Used:**

**Numpy:** NumPy is a fundamental package for scientific computing with Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. In the code snippet, NumPy is used for array manipulation and mathematical operations, such as stacking arrays and converting lists to NumPy arrays.

**Tensorflow:** TensorFlow is an open-source machine learning framework developed by Google for building and training machine learning models, particularly deep learning models. In the code snippet, TensorFlow's Keras API is utilized for loading and preprocessing images, as well as for leveraging a pre-trained EfficientNetB7 model for feature extraction.

**Flask:** Flask is a lightweight WSGI (Web Server Gateway Interface) web application framework in Python. It's commonly used for building web applications and APIs. Although Flask is not explicitly used in the provided code snippet, it can be integrated into the project for serving the trained machine learning model as a web service or for building a web-based user interface.

**MySQLdb**: MySQLdb is a Python module that provides an interface to MySQL database server. It allows Python programs to connect to a MySQL database and perform database operations such as querying and updating data. MySQLdb is not directly used in the provided code snippet, but it could be incorporated into the project if there's a need to store or retrieve data from a MySQL database.

## 3.3 Methodology

### 3.3.1 Block Diagram

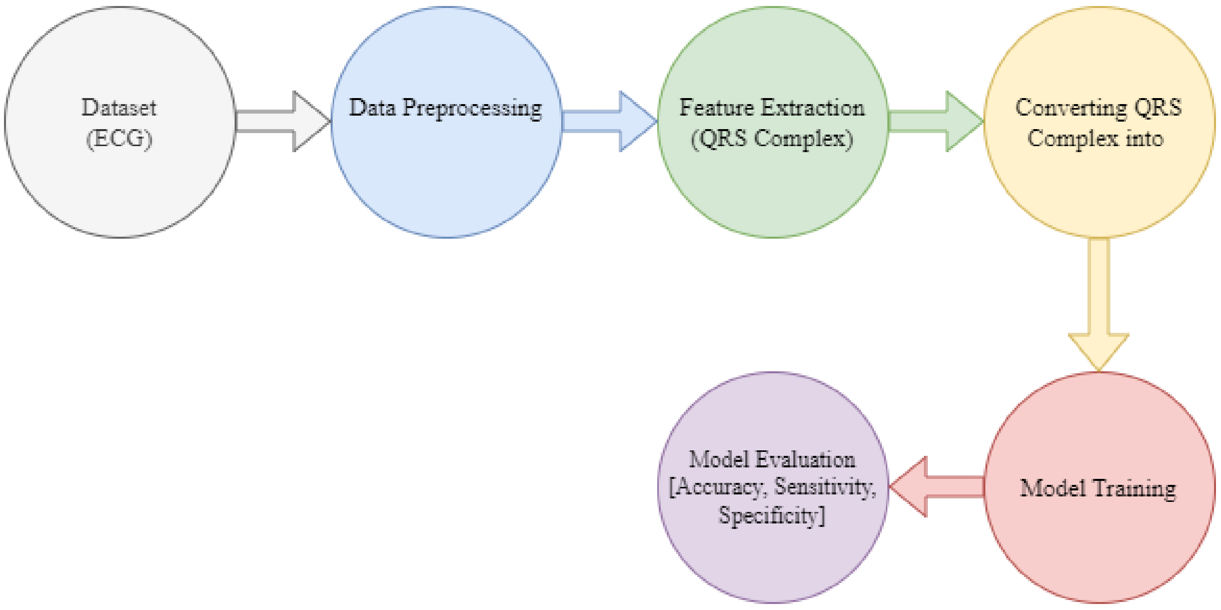

Dataset (ECG)
Data Preprocessing
Feature Extraction (QRS Complex)
Converting QRS Complex into
Model Training
Model Evaluation [Accuracy, Sensitivity, Specificity]

**Figure 3.3.1 Block Diagram of the Model**

### 3.3.2 Model Implementation

This methodology describes a methodical way to creating and validating the model for ECG classification. By following these steps, we can create a model that can automatically interpret ECG data and possibly aid in arrhythmia identification.

**Dataset:** The dataset block plays a foundational role in shaping the study's outcomes. This section outlines the specifics of the dataset used, including its characteristics, scope, and relevance to the research objectives. The MIT-BIH Arrhythmia Database is a widely recognized and extensively used dataset in the field of ECG analysis. It comprises long-term ECG recordings collected from a diverse set of subjects, capturing a range of normal and abnormal cardiac conditions. The dataset is annotated with expert-verified labels, providing a valuable resource for training, and evaluating arrhythmia detection algorithms. Details regarding the size of the dataset, including the number of recordings, the duration of each recording, and the number of subjects, are crucial components of the dataset block. Information about the distribution of normal and arrhythmic patterns within the dataset is highlighted, offering insights into the challenges and complexities inherent in arrhythmia detection.
ECG recordings in the MIT-BIH Arrhythmia Database are annotated by experienced cardiologists, ensuring accurate labelling of normal and abnormal cardiac events. The annotation protocol includes detailed labels for each beat, allowing for precise analysis of arrhythmic patterns. The dataset comprises 48 half-hour ECG recordings from 47 subjects, with one recording containing two separate leads. Each recording is sampled at 360 Hz and
is annotated with expert-verified labels for different arrhythmia classes. The dataset block outlines the specific types of arrhythmias covered in the MIT-BIH Arrhythmia Database. Common arrhythmia classes, such as atrial fibrillation, ventricular tachycardia, and premature ventricular contractions, are detailed. This information aids in understanding the diversity of arrhythmic patterns that the proposed algorithms aim to identify.

**Data Acquisition:** This is the first stage in collecting ECG data. This data could come from a variety of sources, including medical databases and wearable devices.
**Data Preprocessing**: Data preprocessing is a critical step in machine learning projects, playing a pivotal role in ensuring that raw data is transformed into a format suitable for model training. In the context of training a Random Forest classifier for image classification, data preprocessing involves several key tasks aimed at preparing the dataset for effective learning and generalization. The first step in data preprocessing is loading the image data from the specified directory. In this project, we utilized the 'load_images_and_labels function' to read images and their corresponding labels. This function iterates through the directory structure, extracting image paths and class labels. By loading images into memory, we create a foundational dataset structure essential for subsequent preprocessing steps.
Once images are loaded, they undergo resizing to ensure uniformity in dimensions. Images in the dataset are resized to a fixed size of 32x32 pixels using the 'load_img' function from the Keras preprocessing module. Resizing images to a consistent size is crucial for maintaining consistency across the dataset and ensuring compatibility with the model architecture. Normalization of pixel values

follows resizing, a critical step in standardizing data for model training. Normalization scales pixel values to a range between 0 and 1, typically achieved by dividing each pixel value by 255.0. This process enhances model convergence by stabilizing gradient descent optimization and ensuring that all input features contribute equally to model training. Label encoding transforms categorical class labels into numerical format, a requirement for many machine learning algorithms. We employed the 'Label Encoder' from the Scikit-learn library to convert class labels into numeric representations. Each unique class label is assigned a unique integer, enabling the model to process and learn from the labelled data effectively.

**Feature extraction (QRS Complex), Converting QRS complex into image:**
This block involves detection of QRS signals and converting it into images. Once the QRS complexes are identified, relevant features are extracted from these complexes. These features typically included duration, amplitude, morphology, and timing characteristics of the QRS complex.

**Algorithm:**
Random Forest constructs a predefined number of decision trees during training. Each decision tree is trained on a random subset of the training data and a random subset of features. At each node of the decision tree, the algorithm selects the best split among a random subset of features. The split criterion, such as Gini impurity or entropy, is used to determine the optimal separation of data points.
The decision trees are grown recursively by partitioning the feature space until a stopping criterion is met. This stopping criterion can be a maximum tree depth, minimum number of samples in leaf nodes, or others. After all trees are constructed, predictions are made for each individual tree. For classification tasks, the final prediction is determined by majority voting among the trees, while for regression tasks, it's the average of the predictions made by all trees. Random Forest is known for its robustness to overfitting, making it suitable for classification tasks where the dataset may have noise or outliers. Despite constructing multiple decision trees, Random Forest is computationally efficient, especially for large datasets. This efficiency allows for faster model training and prediction. Random Forest provides insights into feature importance, allowing for better understanding of which features contribute the most to the classification task. This information can aid in feature selection and model interpretation. Random Forest can effectively capture nonlinear relationships between features and target variables, making it suitable for image classification tasks where complex patterns may exist.

The EfficientNetB7 model used in this project is known for its efficiency in image classification tasks. EfficientNetB7 is one variant of the EfficientNet family, which was developed by Google researchers to optimize both model size and accuracy. The 'B7' denotes the largest variant in terms of the number of parameters and computational complexity.

EfficientNet models are designed based on a compound scaling method that uniformly scales the network width, depth, and resolution. This scaling allows for better performance compared to traditional methods that only scale one of these dimensions. By efficiently balancing these aspects, EfficientNet achieves state-of-the-art accuracy while maintaining computational efficiency, making it suitable for resource-constrained environments.

In this code, EfficientNetB7 is utilized as a feature extractor. Instead of training the entire model from scratch, which can be computationally expensive, only the pre-trained weights from the ImageNet dataset are used. The 'include_top=False' argument ensures that the fully connected layers (top layers) responsible for ImageNet's original classification task are not included. By setting 'pooling='avg'', the global average pooling layer is used to summarize the spatial information of the extracted features, resulting in a fixed-size feature vector for each input image.

The 'extract_features' function takes an image path as input, preprocesses the image to fit the input size required by EfficientNetB7 (224x224 pixels), converts it to an array, and then passes it through the EfficientNetB7 model to extract features. These features represent high-level representations of the input images learned by EfficientNetB7 through its training on ImageNet.

After extracting features from the images in the dataset, a RandomForestClassifier is trained using these features. RandomForest is a popular ensemble learning method that builds multiple decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees as the final prediction. The trained classifier is then evaluated on both the training, validation, and test sets to assess its performance in terms of accuracy.

Overall, by leveraging the feature extraction capabilities of EfficientNetB7 and the learning power of RandomForest, this code aims to classify ECG images and detect arrhythmias efficiently and accurately.

**Data Splitting for training and testing:**

Developing a machine learning project for ECG classification and arrhythmia detection follows a structured process, commencing with data acquisition and concluding with drawing meaningful conclusions from the data. One indispensable sub-step, essential for obtaining insightful findings, is data splitting. This technique involves dividing the dataset into two subgroups.

A portion of the data is designated for evaluation or testing, while the other part is employed for training the model. During this training phase, the model is exposed to the intricacies of ECG signals, learning patterns, features, and relationships within the training dataset. The training set serves as the foundation for model development, allowing for the comparison of multiple models or the estimation of various parameters.

The significance of this procedure lies in assessing how well the trained model performs on fresh, untested data and its ability to generalize beyond the training set. Given the dynamic nature of ECG signals and the variability in arrhythmias, it is crucial to ensure that the model can adapt to unseen instances effectively.

Data splitting is particularly employed to prevent overfitting in the context of ECG classification. Overfitting occurs when the model fits the training data too closely, capturing noise and specific details that may not be representative of the broader patterns in ECG signals. By utilizing a separate testing dataset, the model's performance is rigorously evaluated, ensuring that it can generalize to new, unseen ECG signals consistently.

In summary, data splitting is an integral step in the development of an ECG classification and arrhythmia detection model. It enables effective model evaluation, comparison of performance, and assessment of generalization capabilities on fresh, untested ECG data. This practice is crucial for building robust and reliable models tailored for accurate arrhythmia detection in diverse ECG signal patterns.

**Training:**

The cornerstone of developing an ECG classification and arrhythmia detection model lies in the utilization of a dedicated training set, a substantial subset of the dataset specifically allocated for training the model's parameters. This pivotal phase enables the model to assimilate knowledge regarding patterns, features, and relationships within the ECG data, enhancing its capability to make accurate predictions when faced with new, unseen instances. The training set plays a critical role in achieving model convergence, ensuring that the model captures the intricate complexities inherent in ECG signals. Throughout the training phase, the model learns from a labelled dataset, a fundamental step in the field of machine learning. This learning process enables the model, tailored for ECG signals, to generate predictions or classifications on novel, unseen data. The primary objective is to identify patterns and features indicative of various heart diseases or arrhythmias, enhancing the model's diagnostic capabilities. The input for the training dataset is comprised of ECG signals, serving as the foundation for the training phase. Each ECG signal in the dataset corresponds to a specific class or type of arrhythmia. Before inputting the data into the model, preprocessing procedures are implemented to ensure consistency and optimize the model's capacity to discern relevant elements within the ECG signals. These preprocessing steps contribute significantly to refining the dataset, preparing it for the subsequent phases of effective model training.

In summary, the training set serves as the backbone of the ECG classification and arrhythmia detection model, allowing the model to learn and generalize from labelled data. This foundational process empowers the model to make accurate predictions on new and unseen instances of ECG signals, thus enhancing its diagnostic efficacy in detecting various arrhythmias and heart conditions.

**Testing:**

The process of evaluating a model's effectiveness involves the use of testing data – a set of ECG signals deliberately withheld from the model during its training phase. This unseen dataset serves as a critical component for assessing the model's performance, generalization abilities, and accuracy when applied to novel instances. Following the construction of the model using the training data, the testing data is introduced to test and validate its capabilities. The significance of this "unseen" data lies in its role as an independent measure, allowing for an objective evaluation of the model's adaptability to new and diverse ECG signals.

Various performance metrics, including accuracy, sensitivity, specificity, and precision, are computed using the testing data. These metrics provide valuable insights into how well the model can accurately classify different types of arrhythmias and normal ECG patterns, serving as benchmarks for its diagnostic capabilities. Keeping the testing data concealed until the evaluation phase is paramount. During the training phase, the model is exposed to a labelled dataset of ECG signals, enabling it to learn patterns and features associated with various classes of arrhythmias. The deliberate separation of testing data prevents the model from memorizing specific instances, ensuring that its performance is not artificially inflated. The need for effective generalization is emphasized in ECG classification to ensure the model can accurately analyse new, unseen ECG signals. By isolating the testing data until evaluation, the model relies on the knowledge gained from the training set without leaning on specific details of the testing dataset. This approach guards against overfitting, where the model becomes too tailored to the training data, potentially hindering its performance on unseen instances. The analysis of testing data results serves a dual purpose. It not only provides a comprehensive evaluation of the model's accuracy but also offers insights for potential fine-tuning or optimization. Adjustments can be made to enhance the model's performance, ensuring its reliability in accurately detecting various arrhythmias.

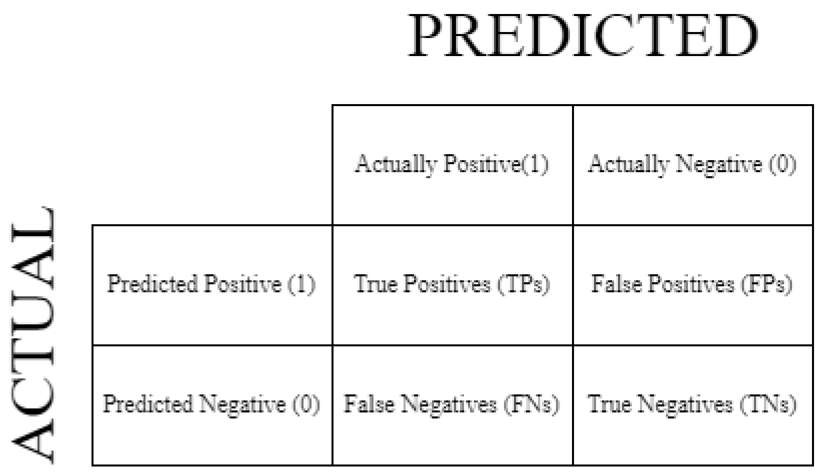
In the final evaluation phase, when the testing data is revealed to the model, it acts as a litmus test for the model's capacity to make precise predictions on fresh and unencountered ECG signals. This rigorous practice guarantees that the model's performance is robust and not solely dependent on memorization, instilling confidence in its ability for accurate arrhythmia detection across a diverse range of patient data.

**Model Evaluation:**
Model evaluation is crucial in the context of ECG (Electrocardiogram) classification and arrhythmia detection. In this domain, evaluating the performance of a machine learning model is essential for understanding its effectiveness in accurately identifying different cardiac conditions from ECG signals. This process involves the use of various evaluation metrics to gain insights into the model's strengths and weaknesses. ECG classification models are designed to analyse electrocardiographic signals and categorize them into specific cardiac conditions or normal patterns. The evaluation process helps assess how well the model distinguishes between various arrhythmias and normal heart rhythms. Commonly used metrics for ECG classification include sensitivity, specificity, accuracy, precision, recall, and F1-score. Model evaluation helps understand the model's performance on different types of ECG data and identify areas for improvement. It also aids in selecting the most suitable model architecture, hyperparameters, and feature representations for the task at hand.
Model evaluation continues to play a crucial role in the ongoing monitoring of ECG classification models. As the model is deployed in real-world scenarios, continuous assessment helps ensure its reliability and effectiveness over time. Regular monitoring allows for the detection of potential drifts in performance and the need for model updates or retraining.

**Model Evaluation Parameters:**
[True Positive (TP): Instances where the model correctly predicts the positive class (correctly identifies an arrhythmia).
Example: The model predicts arrhythmia, and the actual ECG signal has an arrhythmia.
True Negative (TN): Instances where the model correctly predicts the negative class (correctly identifies a normal ECG).
Example: The model predicts a normal ECG, and the actual ECG signal is normal.
False Positive (FP): Instances where the model incorrectly predicts the positive class (incorrectly identifies an arrhythmia when there isn't one).
Example: The model predicts arrhythmia, but the actual ECG signal is normal.
False Negative (FN): Instances where the model incorrectly predicts the negative class (incorrectly identifies a normal ECG when there is an arrhythmia).
Example: The model predicts a normal ECG, but the actual ECG signal has an arrhythmia.]

**Accuracy:** The ratio of correctly predicted instances to the total instances. Provides an overall measure of how well the model is performing across all classes (arrhythmia and normal ECG). However, it may not be the best metric if there is an imbalance between the classes.
Formula: $$\frac{True\ Positive+True\ Negative}{True\ Positive+True\ Negative+False\ Positive+False\ Negative}$$
**Sensitivity/Recall/True Positive Rate (TPR):** The ability of the model to correctly identify positive instances (arrhythmia). High sensitivity is crucial in ECG classification because it indicates the model's ability to detect most instances of arrhythmia, reducing false negatives.
Formula: $$\frac{True\ Positive}{True\ Positive+\ False\ Negative}$$

**Specificity/True Negative Rate (TNR):** The ability of the model to correctly identify negative instances (normal ECG). High specificity is important to minimize false positives, ensuring that normal ECG signals are correctly identified.
Formula: $$\frac{True\ Negative}{True\ Negative+FalsePositive}$$



PREDICTED

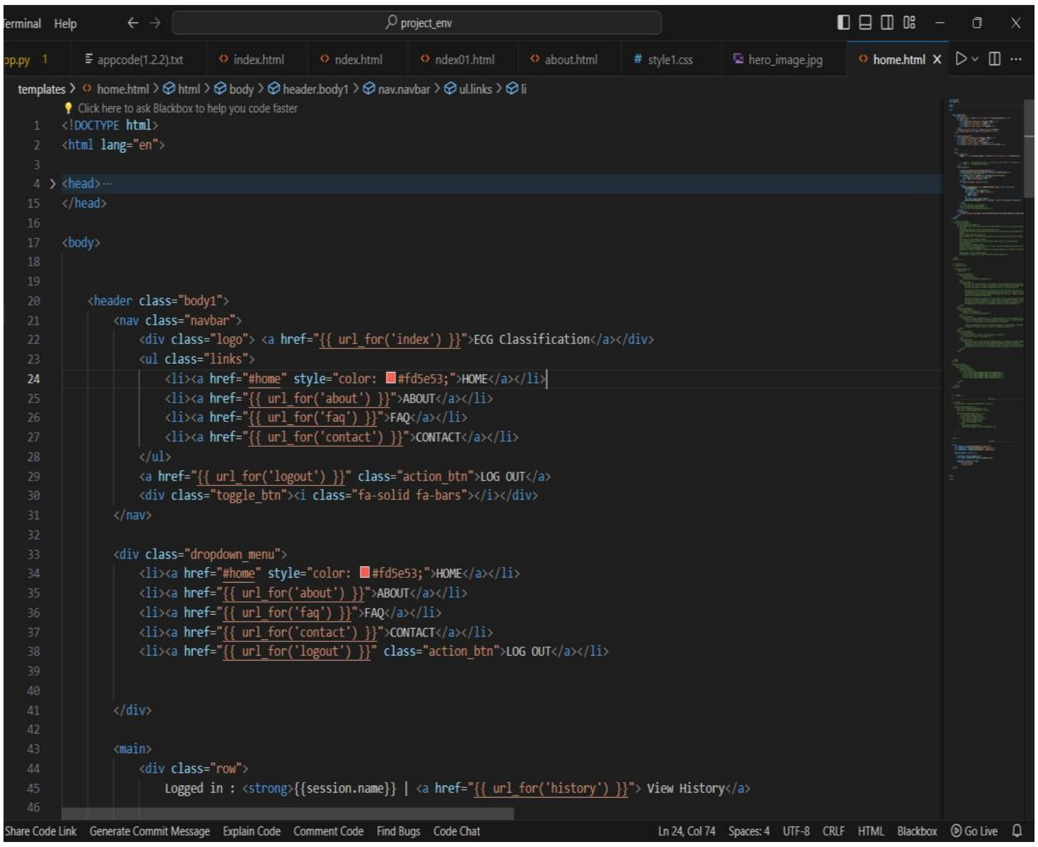
ACTUAL

| | Actually Positive(1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | True Positives (TPs) | False Positives (FPs) |
| Predicted Negative (0) | False Negatives (FNs) | True Negatives (TNs) |

Fig 3.3.2.1 Confusion Matrix

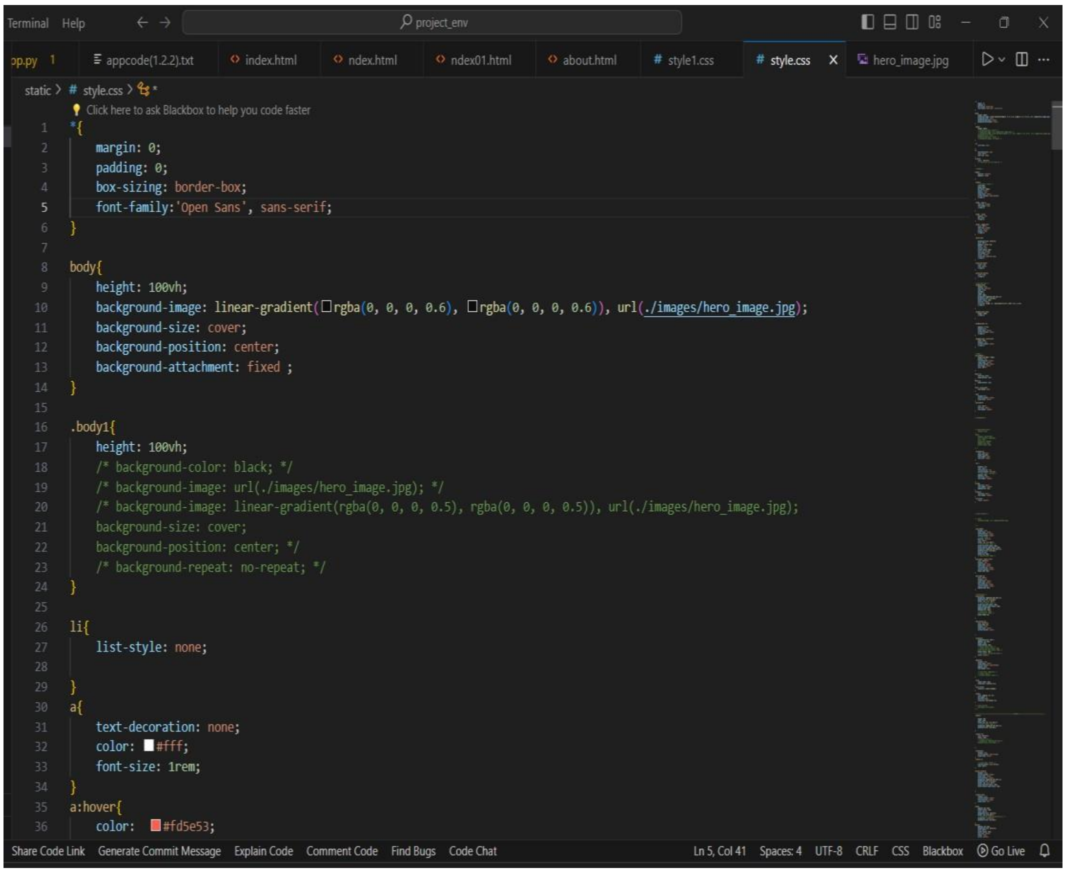
### 3.3.3 Model Deployment on Website

1) The website part was initiated by creating the frontend interface using HTML, CSS, and JavaScript from scratch. This involved designing the layout, styling elements, and implementing interactive features to enhance user experience.



```html
<!DOCTYPE html>
<html lang="en">

<head>···
</head>

<body>


    <header class="body1">
        <nav class="navbar">
            <div class="logo"> <a href="{{ url_for('index') }}">ECG Classification</a></div>
            <ul class="links">
                <li><a href="#home" style="color: #fd5e53;">HOME</a></li>
                <li><a href="{{ url_for('about') }}">ABOUT</a></li>
                <li><a href="{{ url_for('faq') }}">FAQ</a></li>
                <li><a href="{{ url_for('contact') }}">CONTACT</a></li>
            </ul>
            <a href="{{ url_for('logout') }}" class="action_btn">LOG OUT</a>
            <div class="toggle_btn"><i class="fa-solid fa-bars"></i></div>
        </nav>

        <div class="dropdown_menu">
            <li><a href="#home" style="color: #fd5e53;">HOME</a></li>
            <li><a href="{{ url_for('about') }}">ABOUT</a></li>
            <li><a href="{{ url_for('faq') }}">FAQ</a></li>
            <li><a href="{{ url_for('contact') }}">CONTACT</a></li>
            <li><a href="{{ url_for('logout') }}" class="action_btn">LOG OUT</a></li>


        </div>

        <main>
            <div class="row">
                Logged in : <strong>{{session.name}} | <a href="{{ url_for('history') }}"> View History</a>

```

Fig 3.3.3.1 HTML

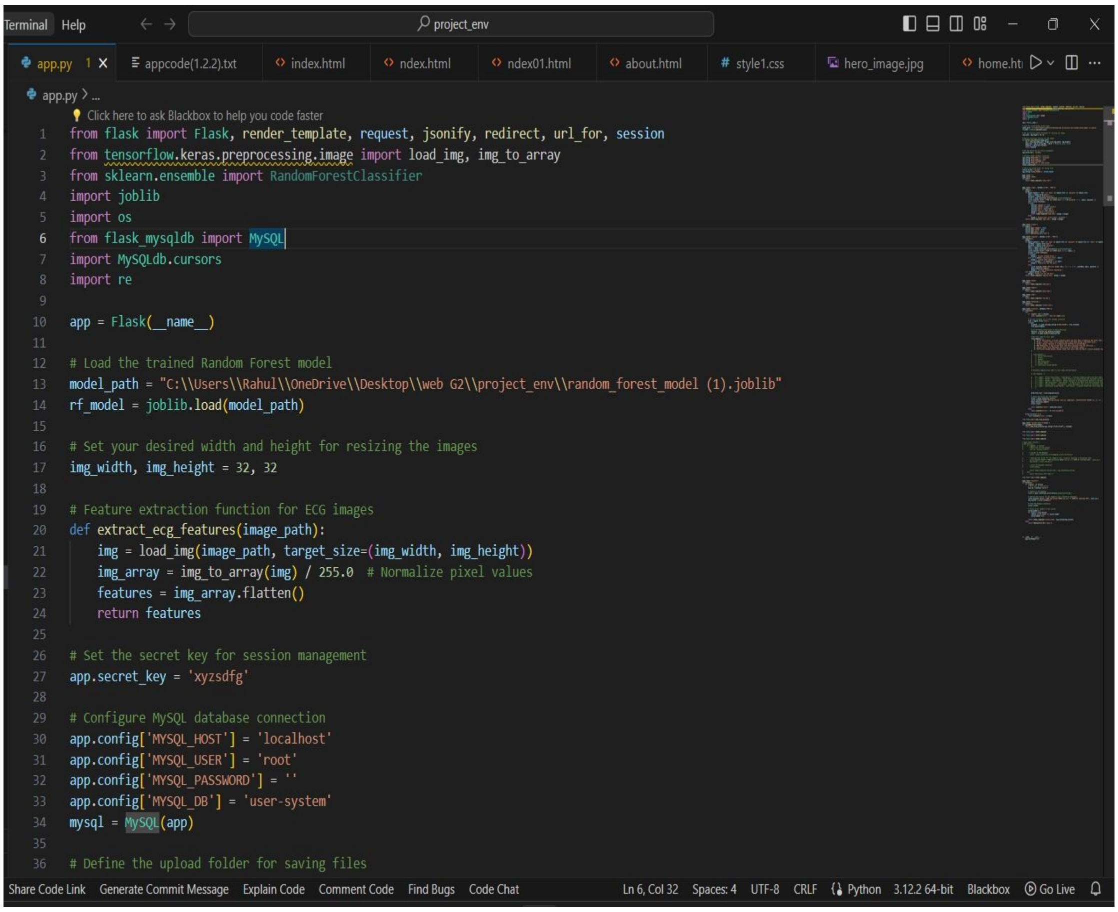


```css
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family:'Open Sans', sans-serif;
}

body{
    height: 100vh;
    background-image: linear-gradient(rgba(0, 0, 0, 0.6), rgba(0, 0, 0, 0.6)), url(./images/hero_image.jpg);
    background-size: cover;
    background-position: center;
    background-attachment: fixed ;
}

.body1{
    height: 100vh;
    /* background-color: black; */
    /* background-image: url(./images/hero_image.jpg); */
    /* background-image: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url(./images/hero_image.jpg);
    background-size: cover;
    background-position: center; */
    /* background-repeat: no-repeat; */
}

li{
    list-style: none;

}
a{
    text-decoration: none;
    color: #fff;
    font-size: 1rem;
}
a:hover{
    color: #fd5e53;
```

Fig 3.3.3.2 CSS

2) To handle the backend functionality, Flask application was implemented. Flask provides a lightweight yet powerful framework for developing web applications in Python.

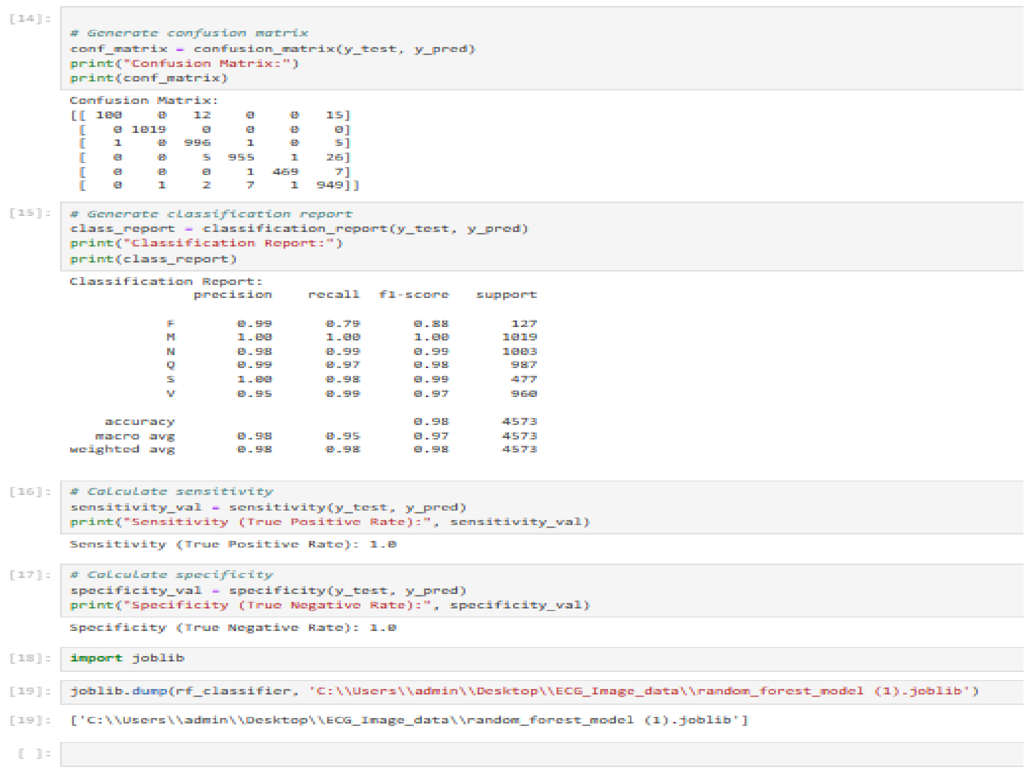
Fig 3.3.3.


Terminal Help
project_env
app.py 1
appcode(1.2.2).txt
index.html
ndex.html
ndex01.html
about.html
style1.css
hero_image.jpg
home.htı
app.py > ...
Click here to ask Blackbox to help you code faster
Share Code Link Generate Commit Message Explain Code Comment Code Find Bugs Code Chat
Ln 6, Col 32 Spaces: 4 UTF-8 CRLF Python 3.12.2 64-bit Blackbox Go Live

```python
from flask import Flask, render_template, request, jsonify, redirect, url_for, session
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from sklearn.ensemble import RandomForestClassifier
import joblib
import os
from flask_mysqldb import MySQL
import MySQLdb.cursors
import re

app = Flask(__name__)

# Load the trained Random Forest model
model_path = "C:\\Users\\Rahul\\OneDrive\\Desktop\\web G2\\project_env\\random_forest_model (1).joblib"
rf_model = joblib.load(model_path)

# Set your desired width and height for resizing the images
img_width, img_height = 32, 32

# Feature extraction function for ECG images
def extract_ecg_features(image_path):
    img = load_img(image_path, target_size=(img_width, img_height))
    img_array = img_to_array(img) / 255.0  # Normalize pixel values
    features = img_array.flatten()
    return features

# Set the secret key for session management
app.secret_key = 'xyzsdfg'

# Configure MySQL database connection
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'user-system'
mysql = MySQL(app)

# Define the upload folder for saving files
```

3
Flask

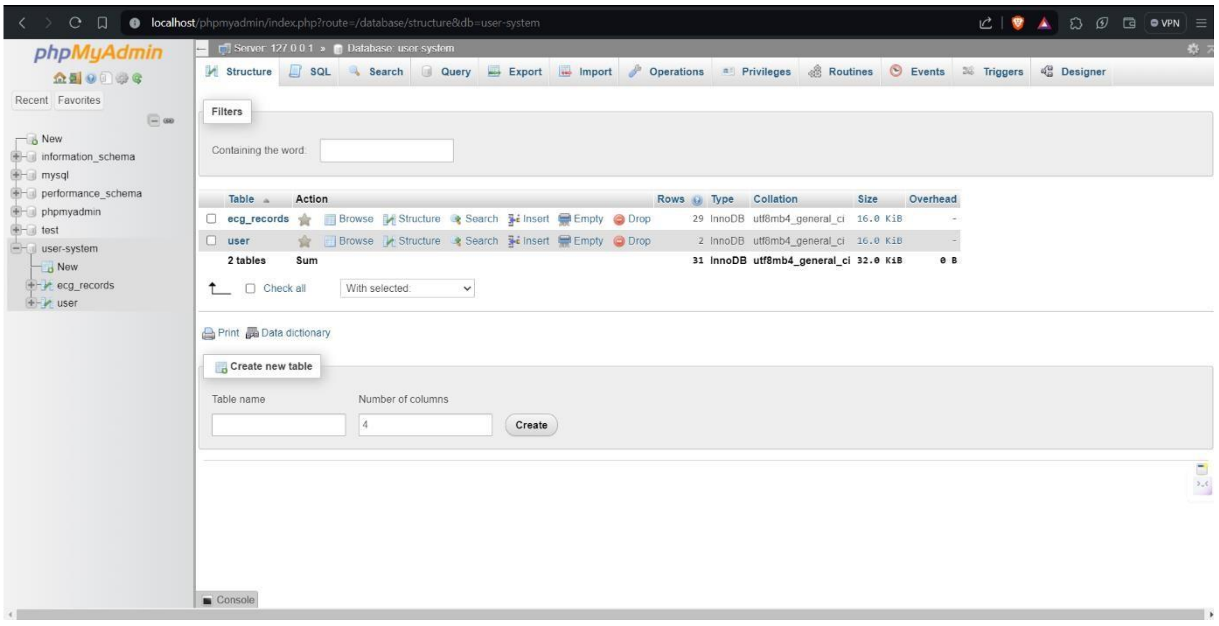
3) One of the core features of web application is its ability to leverage machine learning for ECG classification. A random classifier model was deployed using Python and serialized it into a .joblib file. Through the Flask application, it seamlessly connected this model to the frontend, enabling users to obtain real-time ECG classification results.



```
[14]:
# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

Confusion Matrix:
[[ 100    0   12    0    0   15]
 [    0 1019    0    0    0    0]
 [    1    0  996    1    0    5]
 [    0    0    5  955    1   26]
 [    0    0    0    1  469    7]
 [    0    1    2    7    1  949]]

[15]:
# Generate classification report
class_report = classification_report(y_test, y_pred)
print("Classification Report:")
print(class_report)

Classification Report:
              precision    recall  f1-score   support

           F       0.99      0.79      0.88       127
           M       1.00      1.00      1.00      1019
           N       0.98      0.99      0.99      1003
           Q       0.99      0.97      0.98       987
           S       1.00      0.98      0.99       477
           V       0.95      0.99      0.97       960

    accuracy                           0.98      4573
   macro avg       0.98      0.95      0.97      4573
weighted avg       0.98      0.98      0.98      4573

[16]:
# Calculate sensitivity
sensitivity_val = sensitivity(y_test, y_pred)
print("Sensitivity (True Positive Rate):", sensitivity_val)

Sensitivity (True Positive Rate): 1.0

[17]:
# Calculate specificity
specificity_val = specificity(y_test, y_pred)
print("Specificity (True Negative Rate):", specificity_val)

Specificity (True Negative Rate): 1.0

[18]:
import joblib

[19]:
joblib.dump(rf_classifier, 'C:\\Users\\admin\\Desktop\\ECG_Image_data\\random_forest_model (1).joblib')

[19]:
['C:\\Users\\admin\\Desktop\\ECG_Image_data\\random_forest_model (1).joblib']

[ ]:
```

Fig 3.3.3.4 Executed Code

4) To maintain user accounts and ECG records systematically, a database using XAMPP PHPMyAdmin was implemented. The database consisted of two primary tables: 'user' for storing user account information and 'ecg_record' for storing the historical ECG records of individual patients. This database setup facilitated user registration, login authentication, and the storage of ECG data for future reference.



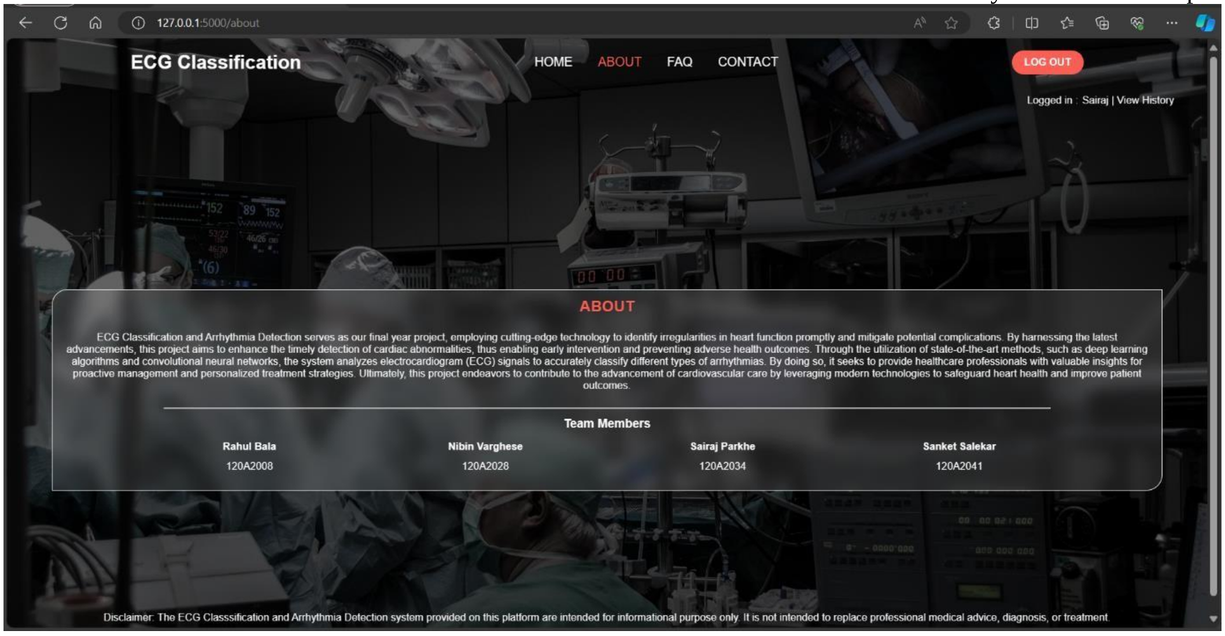
localhost/phpmyadmin/index.php?route=/database/structure&db=user-system

Server: 127.0.0.1 » Database: user-system

| | Table | Action | Rows | Type | Collation | Size | Overhead |
|---|---|---|---|---|---|---|---|
| ☐ | ecg_records | Browse Structure Search Insert Empty Drop | 29 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |
| ☐ | user | Browse Structure Search Insert Empty Drop | 2 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |
| | 2 tables | Sum | 31 | InnoDB | utf8mb4_general_ci | 32.0 KiB | 0 B |

Fig 3.3.3.5 PHPMyAdmin

5) To enhance user experience and provide additional value, features such as an "About" section to provide insights into the project, an FAQ section to address common queries, and a "Contact Us" section were implemented to facilitate communication with users. These additional features not only contribute to usability but also demonstrate the website's commitment to deliver a user-friendly and informative platform.



ABOUT

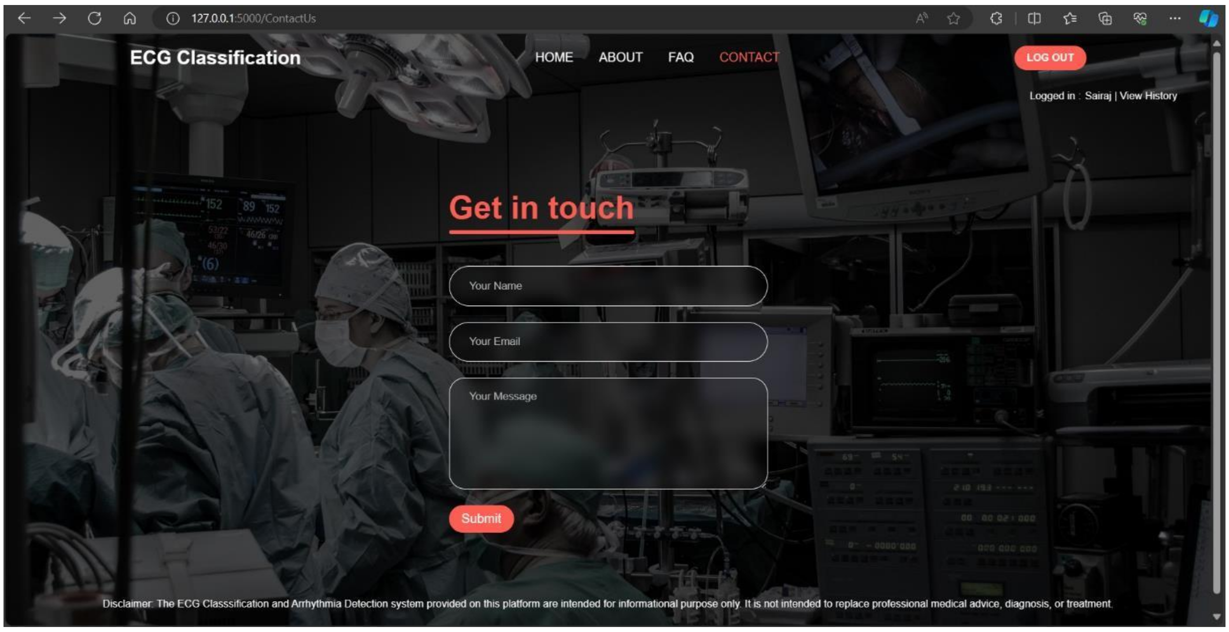
ECG Classification and Arrhythmia Detection serves as our final year project, employing cutting-edge technology to identify irregularities in heart function promptly and mitigate potential complications. By harnessing the latest advancements, this project aims to enhance the timely detection of cardiac abnormalities, thus enabling early intervention and preventing adverse health outcomes. Through the utilization of state-of-the-art methods, such as deep learning algorithms and convolutional neural networks, the system analyzes electrocardiogram (ECG) signals to accurately classify different types of arrhythmias. By doing so, it seeks to provide healthcare professionals with valuable insights for proactive management and personalized treatment strategies. Ultimately, this project endeavors to contribute to the advancement of cardiovascular care by leveraging modern technologies to safeguard heart health and improve patient outcomes.

Team Members

| Rahul Bala | Nibin Varghese | Sairaj Parkhe | Sanket Salekar |
|---|---|---|---|
| 120A2008 | 120A2028 | 120A2034 | 120A2041 |

Disclaimer: The ECG Classsification and Arrhythmia Detection system provided on this platform are intended for informational purpose only. It is not intended to replace professional medical advice, diagnosis, or treatment.

Fig 3.3.3.6 About Section
Fig 3.3.3.7 Contact Section



Disclaimer: The ECG Classsification and Arrhythmia Detection system provided on this platform are intended for informational purpose only. It is not intended to replace professional medical advice, diagnosis, or treatment.

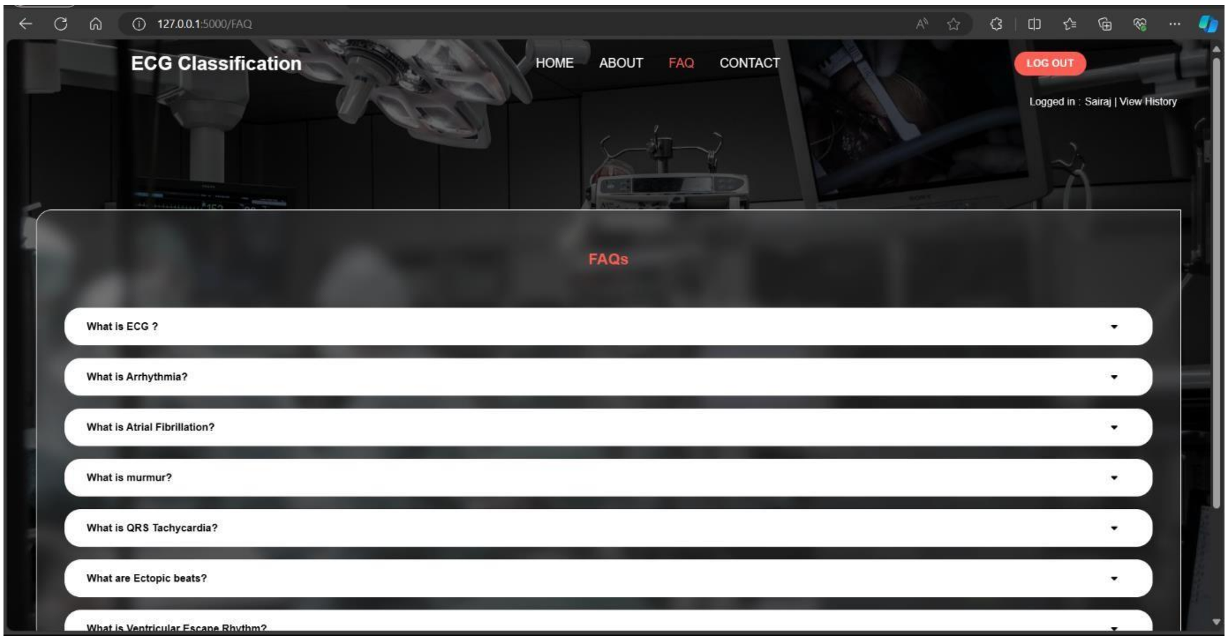


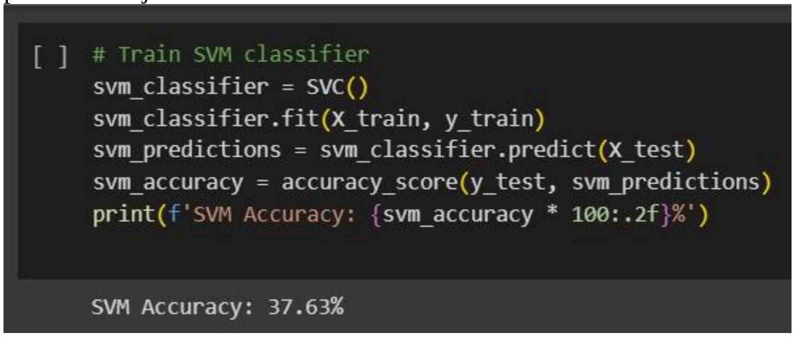
Fig 3.3.3.8 FAQ Section

# Chapter 4 Results And Discussion

The following outcomes were observed while evaluating several machine learning models and algorithms for our project:

## 1. Support Vector Machine (SVM):

SVM produced the lowest accuracy of the models tested. This could be attributable to its linear decision boundary, which may fail to reflect the dataset's non-linear relationships. SVM performance may be limited by factors such as kernel function selection and parameter adjustment.

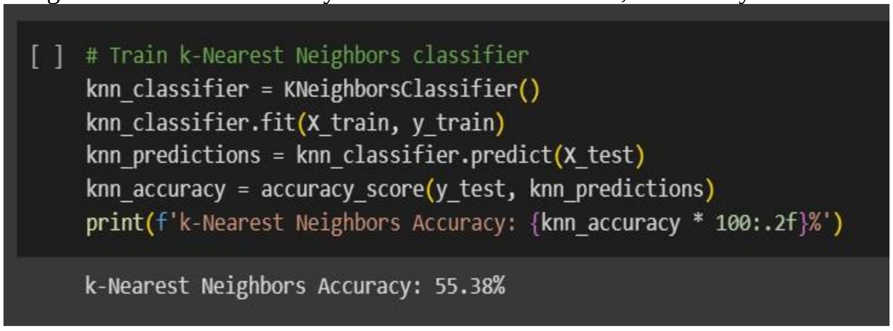


```
[ ]  # Train SVM classifier
     svm_classifier = SVC()
     svm_classifier.fit(X_train, y_train)
     svm_predictions = svm_classifier.predict(X_test)
     svm_accuracy = accuracy_score(y_test, svm_predictions)
     print(f'SVM Accuracy: {svm_accuracy * 100:.2f}%')

     SVM Accuracy: 37.63%
```

Figure 4.1 SVM Classifier

## 2. K Nearest Neighbours (KNN):

KNN outperformed SVM by a moderate margin but fell short of other models. While KNN is a basic and intuitive classification algorithm, it may perform poorly in high-dimensional feature spaces due to the curse of dimensionality. Furthermore, KNN's categorization choice is mainly based on local similarities, which may not accurately reflect the data's global structure.

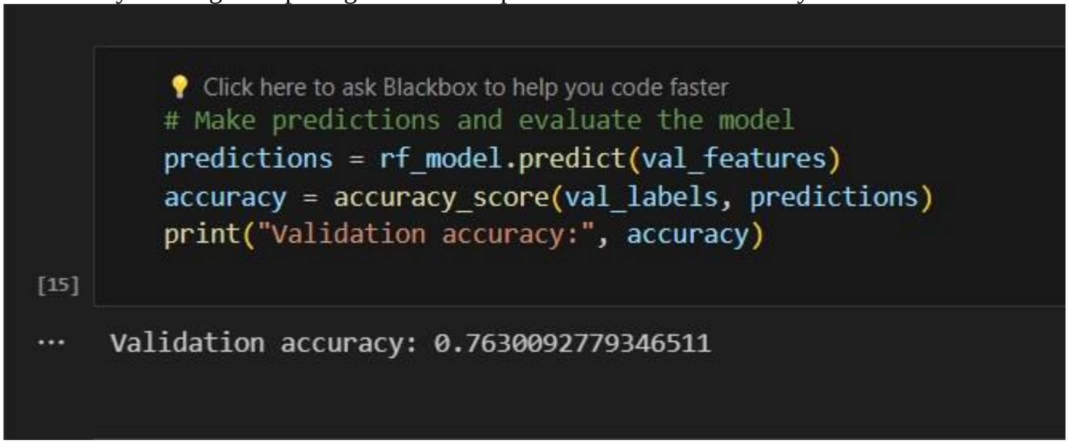


```
[ ]  # Train k-Nearest Neighbors classifier
     knn_classifier = KNeighborsClassifier()
     knn_classifier.fit(X_train, y_train)
     knn_predictions = knn_classifier.predict(X_test)
     knn_accuracy = accuracy_score(y_test, knn_predictions)
     print(f'k-Nearest Neighbors Accuracy: {knn_accuracy * 100:.2f}%')

     k-Nearest Neighbors Accuracy: 55.38%
```

Fig4.2 K-Nearest Neighbors Classifier

## 3. VGG 16 and Random Forest:

This combination takes advantage of the strengths of wavelet transform for feature extraction, VGG 16 for deep feature learning, and random forest for ensemble learning. The wavelet transforms aids in the extraction of both local and global characteristics from data, which are subsequently refined by the deep features learned by VGG 16. The ensemble technique improves performance by integrating numerous decision trees trained on various subsets of data. The large improvement over SVM and KNN demonstrates the efficacy of using multiple algorithms to improve classification accuracy.

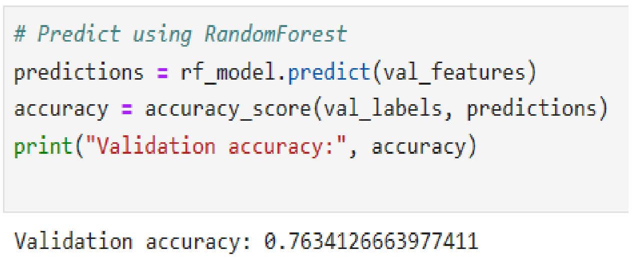


```
# Make predictions and evaluate the model
predictions = rf_model.predict(val_features)
accuracy = accuracy_score(val_labels, predictions)
print("Validation accuracy:", accuracy)
```

```
Validation accuracy: 0.7630092779346511
```

Fig 4.3Wavelet Transform and VGG 16 and Random Forest

Similar to the Wavelet Transform and VGG 16 and Random Forest combo, this model uses VGG 16 features in conjunction with random forest for classification. VGG 16, a deep CNN model, collects hierarchical features from the data, which are subsequently used by random forest to classify. The minor increase in accuracy over the prior combination indicates the strength of VGG 16 features and the efficacy of ensemble learning.



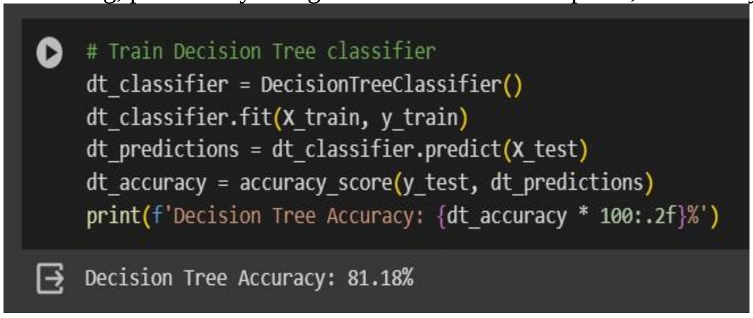
```
# Predict using RandomForest
predictions = rf_model.predict(val_features)
accuracy = accuracy_score(val_labels, predictions)
print("Validation accuracy:", accuracy)
```

```
Validation accuracy: 0.7634126663977411
```

Fig 4.4 VGG 16 and Random Forest

## 5. The Decision Tree:

When compared to earlier models, decision trees performed significantly better in terms of accuracy. Decision trees recursively partition the feature space according to feature values, allowing them to represent complex decision boundaries. The reasonably high accuracy demonstrates the model's ability to capture complicated relationships in the dataset. However, decision trees are prone to overfitting, particularly in high-dimensional feature spaces, which may limit their generalisation capacity.



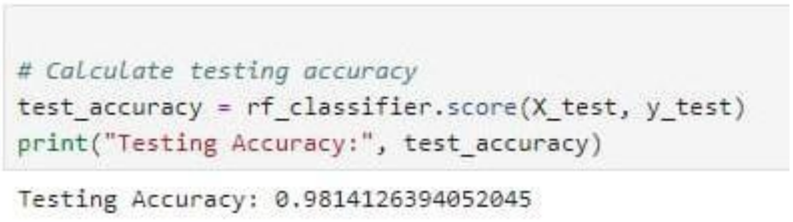
```
# Train Decision Tree classifier
dt_classifier = DecisionTreeClassifier()
dt_classifier.fit(X_train, y_train)
dt_predictions = dt_classifier.predict(X_test)
dt_accuracy = accuracy_score(y_test, dt_predictions)
print(f'Decision Tree Accuracy: {dt_accuracy * 100:.2f}%')
```

```
Decision Tree Accuracy: 81.18%
```

Fig4.5 Decision Tree

## 6. EfficientNet B7 and Random Forest:

EfficientNet B7, a cutting-edge CNN model, scored the highest accuracy of all tested models. Its higher performance demonstrates the effectiveness of deep learning systems in complicated pattern recognition tasks. EfficientNet's scalable architecture strikes a compromise between model size and computational cost, allowing it to efficiently capture and generalise complicated patterns in the dataset. The use of random forest improves performance further through ensemble learning, culminating in excellent accuracy.



```
# Calculate testing accuracy
test_accuracy = rf_classifier.score(X_test, y_test)
print("Testing Accuracy:", test_accuracy)
```

```
Testing Accuracy: 0.9814126394052045
```

Fig 4.6 Actual model implemented (EfficientNetB7 and Random Forest)

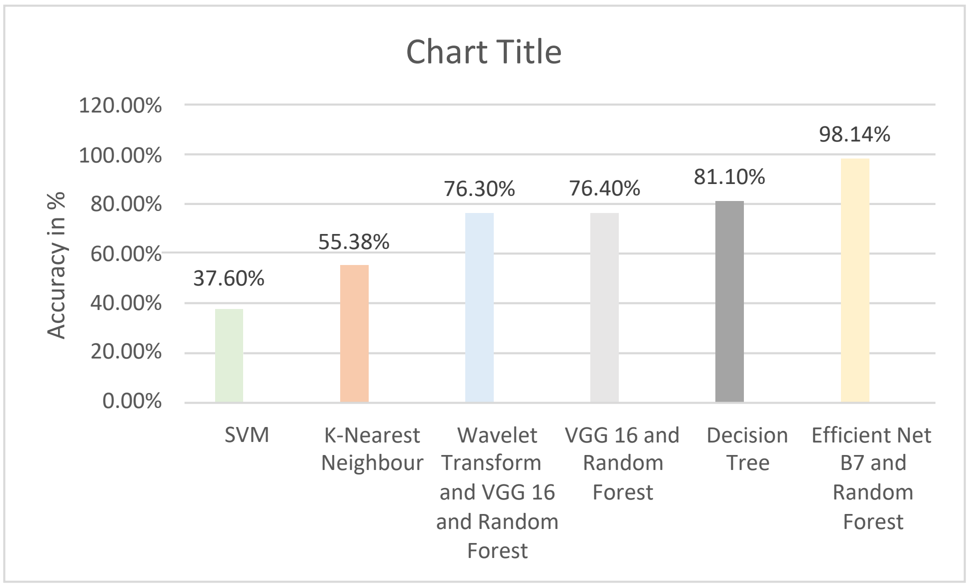
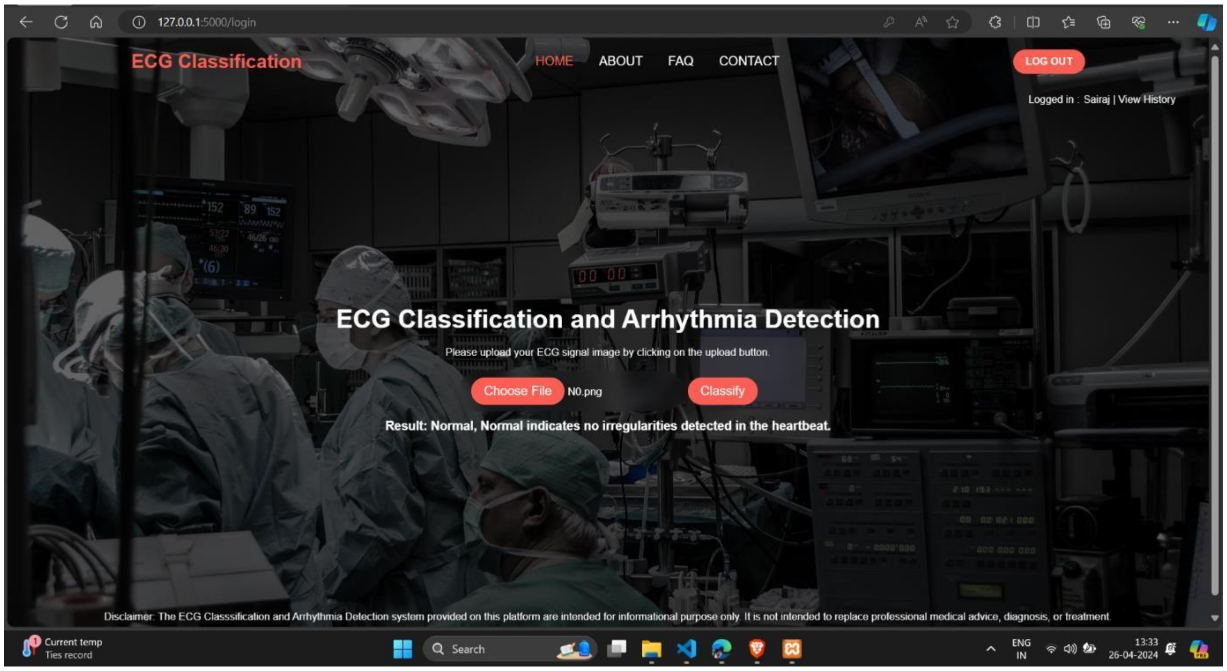


Fig 4.7 Comparative Analysis of Implemented Algorithms

The complete study identifies each model's strengths and weaknesses in handling the categorization problem. Traditional machine learning models like SVM and KNN were less accurate, but ensemble techniques and deep learning architectures showed significant improvement. Top-performing models include decision trees and EfficientNet B7 + Random Forest, highlighting the need of capturing complex patterns for accurate categorization. The findings demonstrate the efficacy of using multiple techniques, such as feature extraction, deep learning, and ensemble learning, to improve classification accuracy in complex datasets.



ECG Classification and Arrhythmia Detection

Please upload your ECG signal image by clicking on the upload button.

Result: Normal, Normal indicates no irregularities detected in the heartbeat.

Disclaimer: The ECG Classsification and Arrhythmia Detection system provided on this platform are intended for informational purpose only. It is not intended to replace professional medical advice, diagnosis, or treatment.

Fig 4.8 Final Result

# Chapter 5 Conclusion And Future Scope

In conclusion, ECG classification and arrhythmia detection are vital applications of machine learning and artificial intelligence in the field of cardiology and healthcare. These technologies have the potential to revolutionize the way we diagnose and manage cardiac conditions. In the field of ECG classification and arrhythmia detection, significant progress has been made in recent years, thanks to advances in machine learning, deep learning, and data analysis techniques. Machine learning algorithms, especially deep neural networks, have shown remarkable potential in accurately classifying ECG signals, distinguishing between various cardiac arrhythmias, and aiding in the diagnosis of heart conditions.

ECG classification and arrhythmia detection represent a crucial frontier in cardiac health monitoring and healthcare technology. The application of machine learning and artificial intelligence to ECG data has the potential to transform the early diagnosis and management of heart-related conditions. As this field continues to evolve, it is essential to ensure the ethical and responsible use of ECG classification technology, protect patient privacy, and maintain high standards of data security. Ongoing research and collaboration between medical professionals, data scientists, and engineers will be critical in harnessing the full potential of ECG classification and arrhythmia detection for the benefit of patients and the healthcare industry.

## References

[1] M. I. Rizqyawan, A. Munandar, M. F. Amri, R. Korio Utoro and A. Pratondo, "Quantized Convolutional Neural Network toward Real-time Arrhythmia Detection in Edge Device," 2020 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET), Tangerang, Indonesia, 2020, pp. 234-239, doi: 10.1109/ICRAMET51080.2020.9298667.

[2] X. Xu and H. Liu, "ECG Heartbeat Classification Using Convolutional Neural Networks," in IEEE Access, vol. 8, pp. 8614-8619, 2020, doi: 10.1109/ACCESS.2020.2964749.keywords:

[3] {Heartbeat;Training;Electrocardiography;Neurons;Convolution;Databases;Convolutional neural networks;Heartbeats;Holter;convolutional neural networks;MIT-BIH arrhythmia database;electrocardiogram signals},

[4] B. N. Sudila and G. Poravi, "Technological Analysis of ECG Classification based on Machine Learning and Deep Learning Techniques," 2020 International Conference on Image Processing and Robotics (ICIP), Negombo, Sri Lanka, 2020, pp. 1-6, doi: 10.1109/ICIP48927.2020.9367365. keywords: {Wavelet transforms;Heart;Machine learning algorithms;Adaptive filters;Electrocardiography;Feature extraction;Classification algorithms;Arrhythmia classification;ECG;pre-processing;feature extraction;optimization;SVM;wavelet transform;CNN;ANN adaptive filter},.

[5] Mohebbanaaz, Y. P. Sai and L. V. R. kumari, "A Review on Arrhythmia Classification Using ECG Signals," 2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS), Bhopal, India, 2020, pp. 1-6, doi: 10.1109/SCEECS48394.2020.9. keywords: {Energy

consumption;Databases;Noise reduction;Electrocardiography;Feature extraction;Real-time systems;Optimization;ECG signal;Denoising;Feature Extraction;Arrhythmia Classification},

[6] H. I. Bulbul, N. Usta and M. Yildiz, "Classification of ECG Arrhythmia with Machine Learning Techniques," 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 2017, pp. 546-549, doi: 10.1109/ICMLA.2017.0-

[7] 104. keywords: {Electrocardiography;Support vector machines;Classification algorithms;Diseases;Discrete wavelet transforms;Discrete cosine transforms;Heart;Machine learning;ECG;SVM;MLP;DWT;DCT;CWT},

[8] J. Huang, B. Chen, B. Yao and W. He, "ECG Arrhythmia Classification Using STFT-Based Spectrogram and Convolutional Neural Network," in IEEE Access, vol. 7, pp. 92871-92880, 2019, doi: 10.1109/ACCESS.2019.2928017.

[9] R. Thilagavathy, R. Srivatsan, S. Sreekarun, D. Sudeshna, P. L. Priya and B. Venkataramani, "Real-Time ECG Signal Feature Extraction and Classification using Support Vector Machine," 2020 International Conference on Contemporary Computing and Applications (IC3A), Lucknow, India, 2020, pp. 44-48, doi: 10.1109/IC3A48958.2020.233266.

[10] S. M. Jadhav, S. L. Nalbalwar and A. Ghatol, "Artificial Neural Network based cardiac arrhythmia classification using ECG signal data," 2010 International Conference on Electronics and Information Engineering, Kyoto, Japan, 2010, pp. V1-228-V1-231, doi: 10.1109/ICEIE.2010.5559887.

[11] H. I. Bulbul, N. Usta and M. Yildiz, "Classification of ECG Arrhythmia with Machine Learning Techniques," 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 2017, pp. 546-549, doi: 10.1109/ICMLA.2017.0- 104.

[12] A. P. Roy, S. Chatterjee, P. Maji and H. K. Mondal, "Classification of ECG Signals for IoT-based Smart Healthcare Applications using WBAN," 2020 International Symposium on Devices, Circuits and Systems (ISDCS), Howrah, India, 2020, pp. 1-4, doi: 10.1109/ISDCS49393.2020.9263011.

## AUTHORS

**First Author** – Rahul Bala, Department of Electronics and Telecommunication Engineering SIES GRADUATE SCHOOL OF TECHNOLOGY, Nerul, Navi Mumbai 400706 University of Mumbai
**Second Author** – Nibin Varghese, Department of Electronics and Telecommunication Engineering SIES GRADUATE SCHOOL OF TECHNOLOGY, Nerul, Navi Mumbai 400706 University of Mumbai
**Third Author** – Sairaj Parkhe, Department of Electronics and Telecommunication Engineering SIES GRADUATE SCHOOL OF TECHNOLOGY, Nerul, Navi Mumbai 400706 University of Mumbai
**Fourth Author** – Sanket Salekar, Department of Electronics and Telecommunication Engineering SIES GRADUATE SCHOOL OF TECHNOLOGY, Nerul, Navi Mumbai 400706 University of Mumbai