# Information File - ECG Signal Classification for Arrhythmia Detection

**Link For Video**

YouTube: <https://youtu.be/GY6xp_KjZ8Y>

**Link for Blogs:**

Medium Blog Link: <https://medium.com/@s11208621/mm466-blog-8f43aa6381ba>​

GitHub Link: <https://github.com/Mrun-mayi/ecg-ml-blog>​

​

## 📌 Project Overview

The objective of this project was to classify ECG signals using machine learning techniques to detect arrhythmias. The MIT-BIH Arrhythmia Database served as the primary data source. The feature-extracted dataset was split using two approaches: SMOTE for balancing and K-Fold cross-validation for robust model evaluation.  
  
The dataset was divided into:  
- 70% Training  
- 15% Validation  
- 15% Testing  
  
Several classification models were trained using MATLAB's Classification Learner App and tested against the reserved test set to evaluate performance.

## ⚙️ Tools and Technologies

- MATLAB  
- Classification Learner App  
- Signal Processing Toolbox  
- Statistics and Machine Learning Toolbox

## 🧠 Models Trained

- Decision Tree  
- Random Forest  
- K-Nearest Neighbors (KNN)  
- Neural Networks:  
- Narrow  
- Medium  
- Wide  
- Bi-layered  
- Tri-layered

## 📊 Dataset Information

Source: MIT-BIH Arrhythmia Database  
Format: Pre-extracted feature dataset from raw ECG signals  
Classes: Includes various heartbeat types such as Normal, SVEB, VEB, Fusion, and Unknown

## 🧪 Evaluation Methods

- Accuracy, Precision, Recall, F1-Score  
- Confusion Matrix Analysis  
- K-Fold Cross-Validation  
- SMOTE-based oversampling

## 📈 Key Results

Random Forest (K-Fold) achieved the highest accuracy: 99.36%  
Neural networks also showed high performance, especially Wide and Bi-layered models.  
Confusion matrix analysis showed minimal misclassifications between morphologically similar classes.

## 🚀 How to Run

1. `Cleaning\_data.mlx`  
 - Combines all input datasets into one.  
 - Fills missing data.  
 - Normalizes the combined dataset using Z-score standardization.  
  
2. `EDA\_of\_cleaned\_data.mlx`  
 - Performs exploratory data analysis.  
 - Displays histograms, bar plots, and 3D scatter plots.  
  
3. `SMOTE2.mlx`  
 - Performs oversampling using the custom function `customSMOTE\_to\_target\_all.m`.  
 - Balances the dataset by oversampling F, SVEB, Q and selecting 51,669 samples from N.  
  
4. `cvpartitionsmote.mlx`  
 - Splits the SMOTE-balanced dataset into training, validation, and testing sets.  
  
5. `K\_Fold.mlx`  
 - Performs 42-fold cross-validation on the full dataset.

6. Open ClassificationLeanerSessioncomplete and export the models as trainedModel to trainedModel8 and open ClassificationLearnerSessionKFold and export the three models as trainedModel9 to trainedModel11  
  
7. `TrainedModels.mlx`  
 - Displays confusion matrices for trained models.  
 - Outputs accuracy scores in table format.  
  
8. `KeyPerformanceIndicators.mlx`  
 - Calculates and presents Accuracy, Precision, Recall, and F1-Score for each model.

9. Open ‘PredictionGUI.m’ file in the command window to show the GUI which allows the user to select which model to use to test for accuracy. It also lets you choose a sample and let it predict.

## 👨‍💻 Author

Venayak Pillay & Mrunmayi Badgujar  
Bachelor of Engineering in Mechanical Engineering  
University of the South Pacific (USP)

## 📂 License

This project is for academic and research purposes only. Please cite appropriately if used.