BANGALORE INSTITUTE OF TECHNOLOGY

K.R.ROAD, V.V.PURAM, BANGALORE - 560004

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

Project Work (21ISP76) on

"CARDIOVASCULAR DISEASE PREDICTION USING ENSEMBLE TECHNIQUE"

Under the guidance of :

Dr. Hema Jagadish
Associate Professor
Department of ISE
B.I.T

Presented By

Manas Chagi	1BI21IS051
Abhishek G	1BI21IS069
Theju T S	1BI21IS124
Bhargava K R	1BI21IS125

CONTENTS

ABSTRACT

The project focuses on predicting cardiovascular disease using ensemble machine learning technique and ECG image processing. The system enhances ECG images by converting them to grayscale, reducing noise with Gaussian smoothing, and extracting waveforms. Principal Component Analysis (PCA) helps reduce dimensionality while preserving key features. A Voting Classifier combines SVM, kNN, Random Forest, Gaussian Naive Bayes, and Logistic Regression, using a soft voting strategy to improve accuracy. The project is Built with scikit- learn and Streamlit, the system provides an easy-to-use interface for medical professionals to upload ECG images and receive predictions, highlighting the potential of AI for non- invasive heart disease diagnosis.

INTRODUCTION

- **Cardiovascular disease** is one of the leading causes of death globally, highlighting the need for early detection and accurate diagnosis to improve patient outcomes.
- Traditional diagnostic methods, particularly those based on **Electrocardiogram** (ECG) signals, often require manual analysis, which can be time-consuming and prone to human error.
- The project Cardiovascular Disease Prediction Using **Ensemble** seeks to automate this process by utilizing advanced image processing and machine learning techniques to predict cardiovascular diseases from ECG images. By employing an ensemble technique, the system combines the strengths of multiple models to enhance prediction **accuracy** and **reliability**. This report outlines the methodologies, data processing steps, and model implementation involved in creating a robust ECG-based prediction system for cardiovascular disease.

LITERATURE SURVEY

[1]: “Digitization of Printed ECG Data Using Image Processing Techniques”

- **Authors:** Tahmida Tabassum and Mohiuddin Ahmad

Key Findings:

1. **High Accuracy Digitization:** The method ensures precise conversion of printed ECG data into digital form, improving storage, analysis, and clinical applications.
2. **Efficient Data Handling:** The digitized ECG reduces data size while maintaining waveform integrity, enabling real-time processing and diagnosis.

Drawbacks:

1. **Sensitivity to Noise:** The accuracy of digitization can be affected by noise and distortions in printed ECG records, requiring additional preprocessing.
2. **Dependency on Image Quality:** The method relies on high-resolution scans and may not perform well on low-quality or degraded ECG strips.

[2]: “Review of Ensemble Learning Techniques for Disease Prediction”

- **Title:** “Review of Ensemble Learning Techniques for Disease Prediction”
- **Authors:** Palak Mahajan et al.

Key Findings:

1. **Stacking Achieves Highest Accuracy:** Stacking outperformed other ensemble methods, particularly in predicting skin disease and diabetes.
2. **Method-Specific Strengths:** Bagging worked best for kidney disease, while boosting excelled in liver disease and diabetes predictions.

Drawbacks:

1. **Performance Variability:** The effectiveness of ensemble methods varied across different diseases and datasets, requiring case-specific model selection.
2. **Complexity in Implementation:** Stacking and other ensemble techniques require extensive computational resources and careful parameter tuning.

[3]: “An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants”

- **Title:** “An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants”
- **Authors:** Bauer and Kohavi

Key Findings:

1. **Bagging Reduces Variance:** Bagging proved effective in minimizing variance, enhancing model stability across different datasets.
2. **Boosting Enhances Accuracy:** Boosting significantly reduced bias, improving overall classification performance.

Drawbacks:

1. **Risk of Overfitting:** Boosting can be sensitive to noisy data, leading to overfitting in certain cases.
2. **Computational Complexity:** Both bagging and boosting require substantial computational resources, making them expensive for large datasets.

[4]: “Effectiveness of Ensemble in Observational Healthcare Data”

- **Title:** “Effectiveness of Ensemble in Observational Healthcare Data”
- **Authors:** Behzad Naderalvojoud and Tina Hernandez-Boussard

Key Findings:

1. **Improved Prediction Accuracy:** The ensemble model enhanced AUROC and AUPRC metrics, improving the classification of patients at risk of prolonged opioid use.
2. **Comprehensive Risk Assessment:** By combining models trained on different covariates, the approach provided deeper insights into risk factors influencing opioid dependency.

Drawbacks:

1. **Complexity in Model Integration:** Combining multiple models increases computational requirements and complexity in implementation.
2. **Data Dependency Issues:** The effectiveness of the ensemble approach relies heavily on the quality and completeness of healthcare datasets.

PROBLEM STATEMENT

The challenge is to develop an automated system for predicting cardiovascular diseases using ECG signals, reducing reliance on manual interpretation, which is time-consuming and error-prone. By leveraging image processing and machine learning, the system aims to enhance detection accuracy, scalability, and early diagnosis.

Objectives

1. **ECG Image Preprocessing:** To enhance the quality of ECG images, apply preprocessing techniques such as grayscale conversion, Gaussian smoothing, and thresholding to remove noise and improve the clarity of important features in the ECG signals.
2. **Lead Division and Signal Extraction:** Extract individual ECG leads (12 standard leads and 1 long lead) from the ECG images to isolate the distinct heart electrical signals.
3. **Feature Extraction and Signal Transformation:** Extract meaningful features from the ECG signals by identifying key signal characteristics. This will involve techniques such as contour extraction, dimensionality reduction using PCA and scaling for normalization.
4. **Ensemble Learning Model Development:** Develop a robust classification model using ensemble techniques.

5. **Model Evaluation and Performance Assessment:** Evaluate the model using metrics such as accuracy, precision, recall, and F1-score to ensure that the system can correctly classify ECG signals into different categories, such as normal heart rhythm, abnormal heartbeats, or indications of myocardial infarction.
6. **Prediction for Early Diagnosis:** To provide an accurate and timely prediction of cardiovascular diseases, enabling early intervention, and supporting clinical decision making for healthcare professionals.

METHODOLOGY

1. ECG Image Collection and Preprocessing

- Collect ECG images in formats such as PNG, JPG, or JPEG.
- Convert images to grayscale to simplify analysis and reduce complexity.

2. Dividing ECG Leads

- Divide ECG images into 13 individual leads (12 standard leads and 1 long lead).
- Treat each lead as a separate region of interest (ROI) for analysis.

3. Signal Extraction and Feature Scaling

- Enhance ECG signals using contour detection to extract features.
- Normalize signal values with Min-Max Scaling for machine learning models.

4. Dimensionality Reduction with PCA

- Use Principal Component Analysis (PCA) to reduce dimensionality.
- Retain the most significant features while removing redundant information.

5. Ensemble Learning for Classification

- Employ Logistic Regression, KNN, Decision Trees, Random Forest, and Naive Bayes.
- Combine predictions of individual models for a robust final diagnosis.

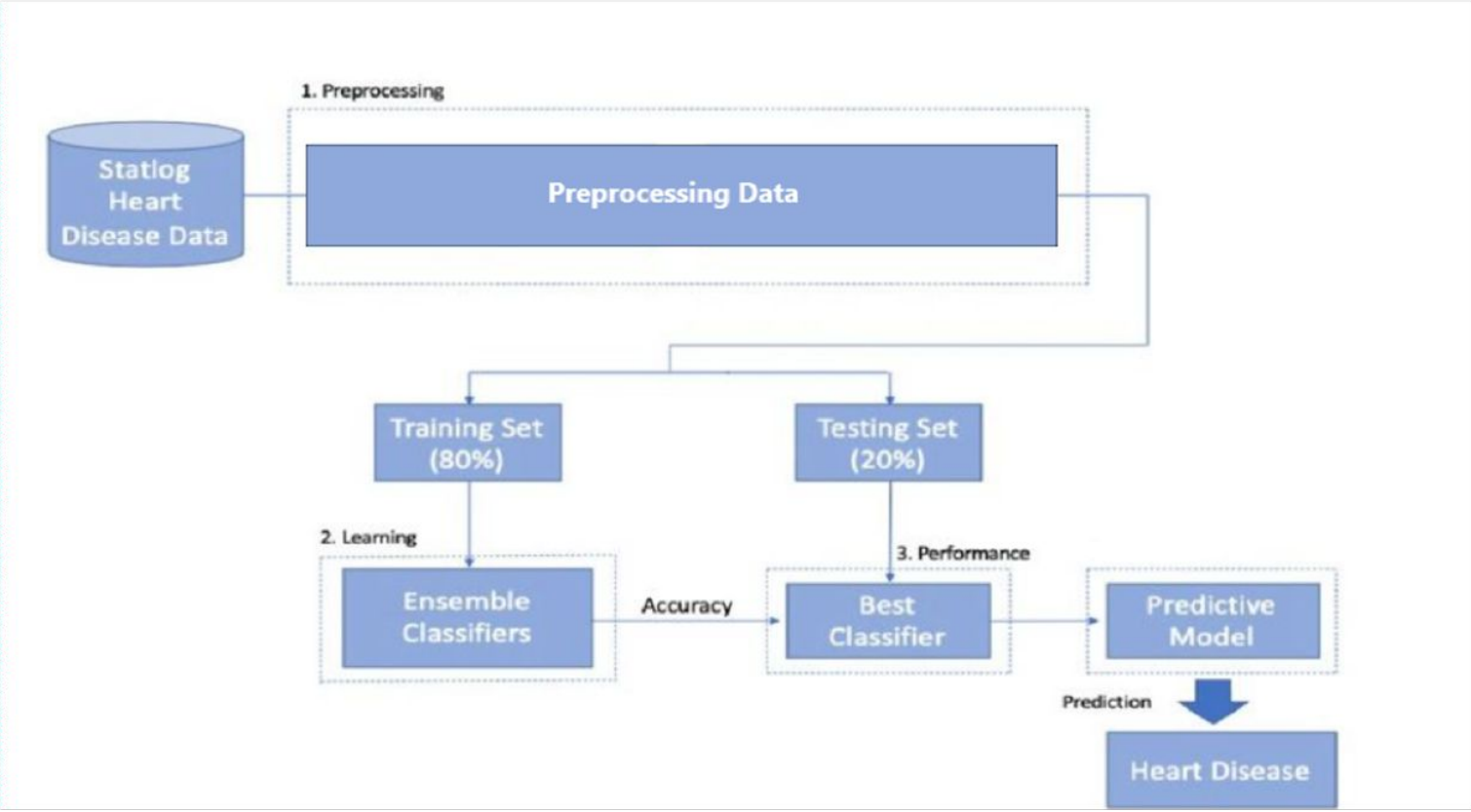
6. Model Training and Evaluation

- Train models on labeled ECG data and apply cross-validation.
- Evaluate performance using metrics like accuracy, precision, recall, and F1-score.
- Select the best-performing model based on evaluation results.

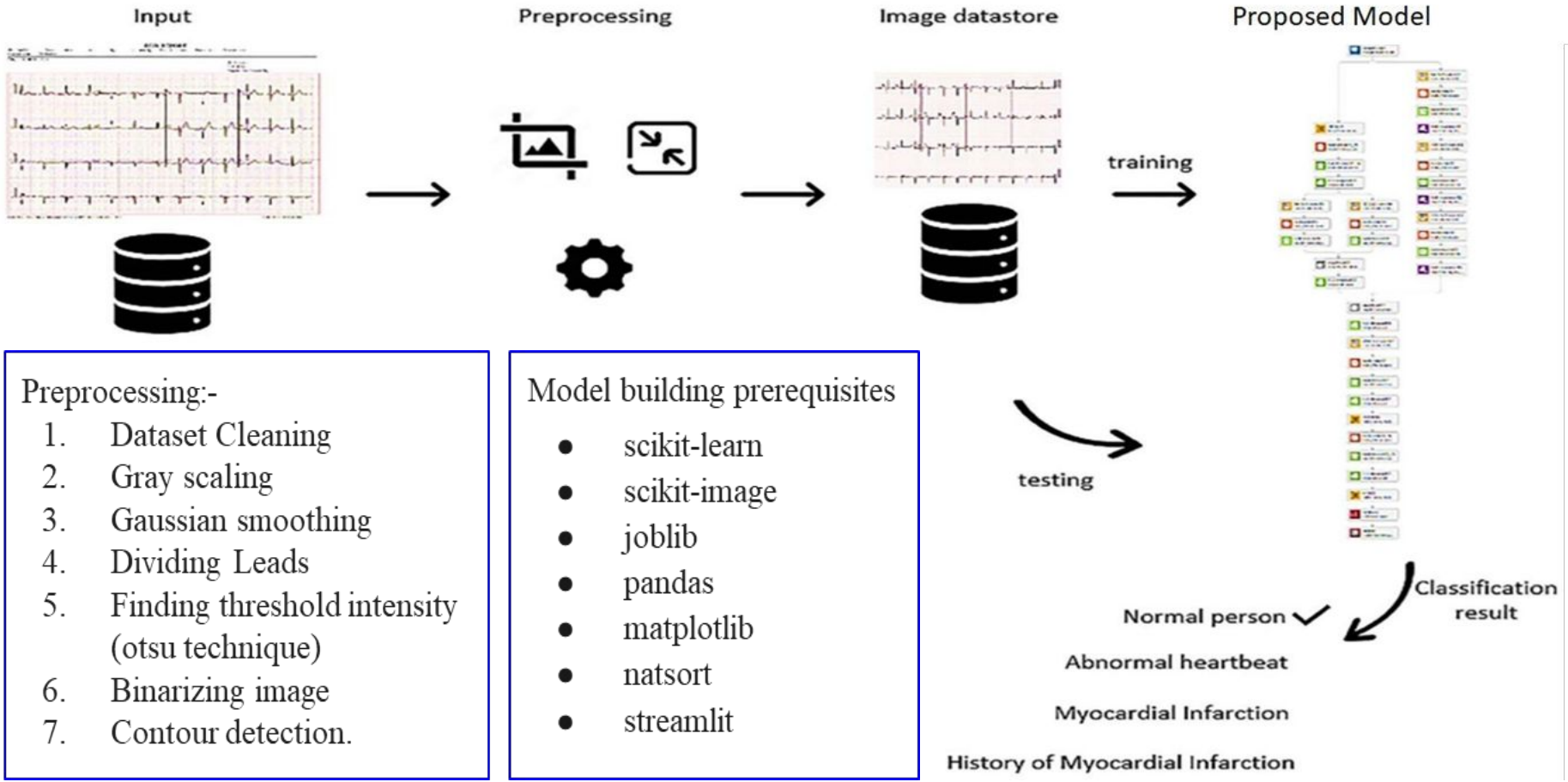
7. Prediction and Results

- Use the trained model to predict cardiovascular diseases, such as myocardial infarction.
- Provide a diagnostic output to assist healthcare professionals in decision-making

ARCHITECTURAL DESIGN



Detailed Design



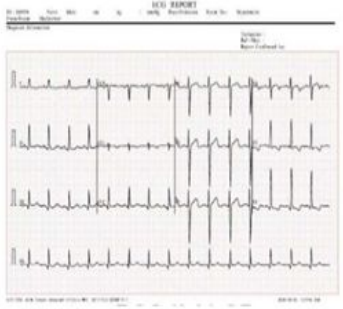
Preprocessing:-

1. Dataset Cleaning
2. Gray scaling
3. Gaussian smoothing
4. Dividing Leads
5. Finding threshold intensity (otsu technique)
6. Binarizing image
7. Contour detection.

Model building prerequisites

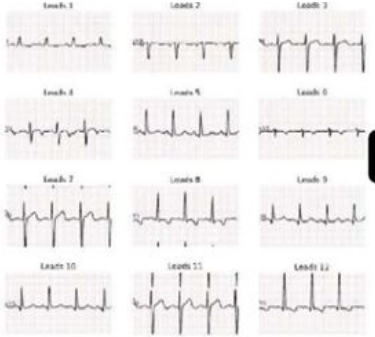
- scikit-learn
- scikit-image
- joblib
- pandas
- matplotlib
- natsort
- streamlit

DATA FLOW DIAGRAMS

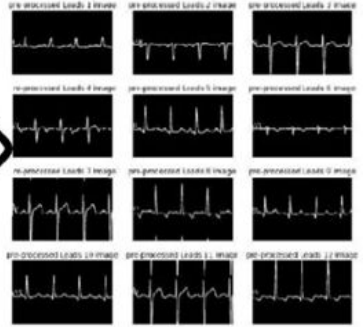


ECG IMAGE

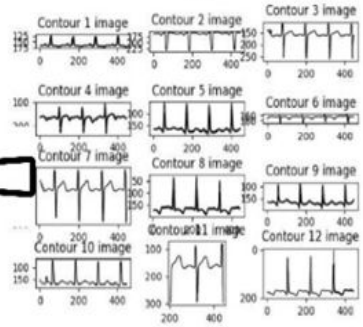
GRAY SCALE CONVERSION



DIVIDING LEADS



PRE-PROCESSING LEADS



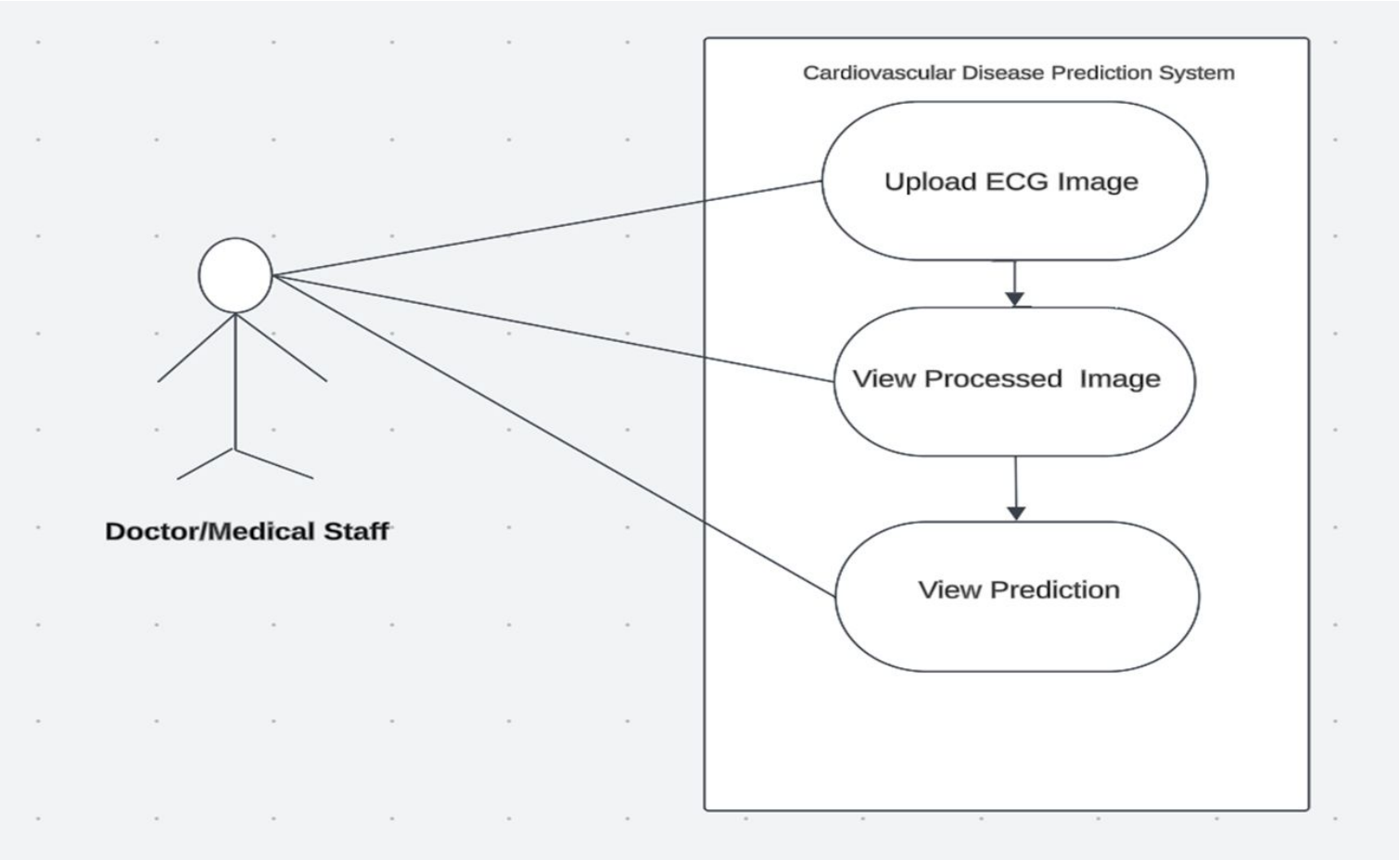
EXTRACTING SIGNALS

1D SIGNALS

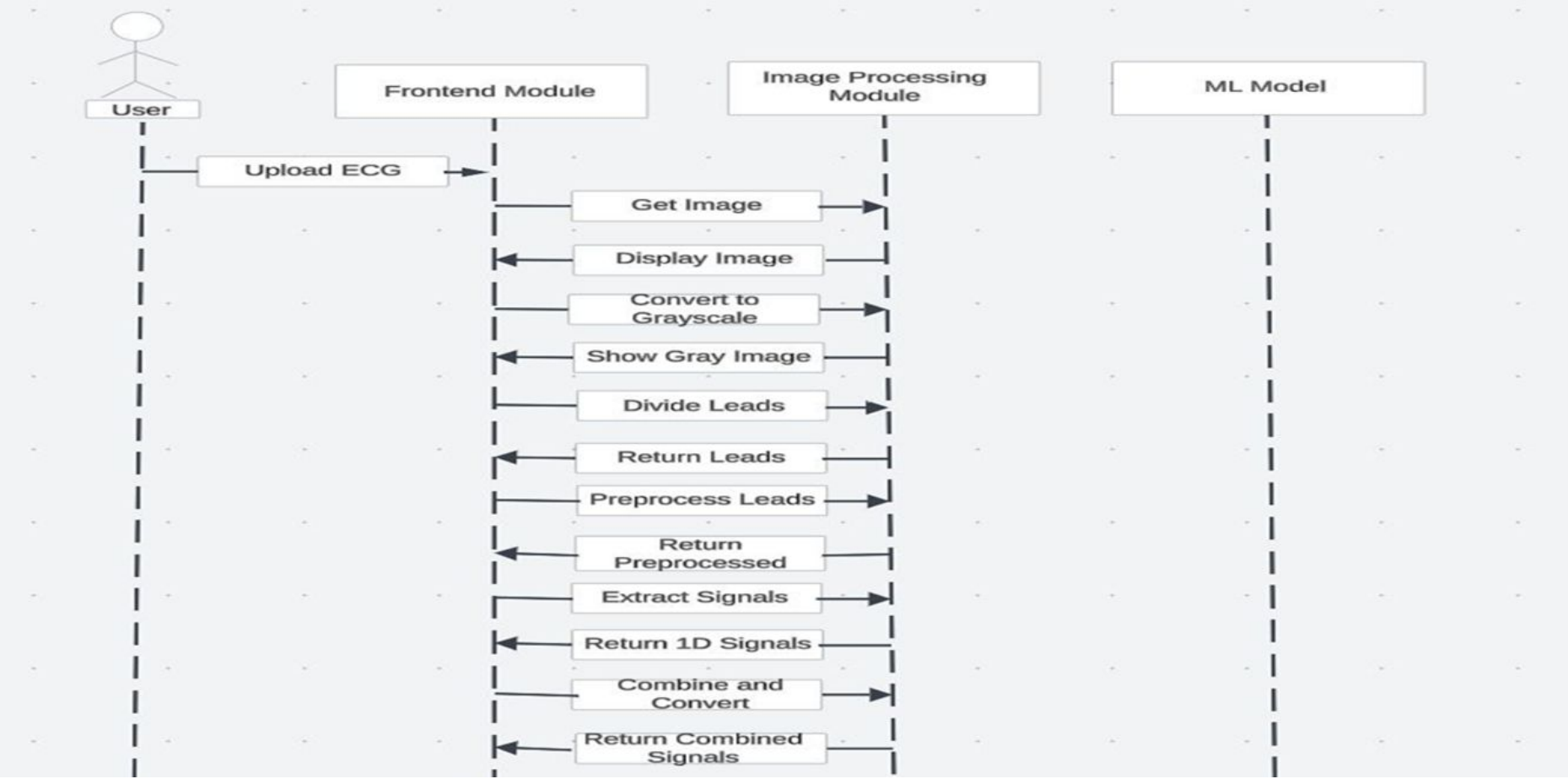
DIMENSIONALITY REDUCTION

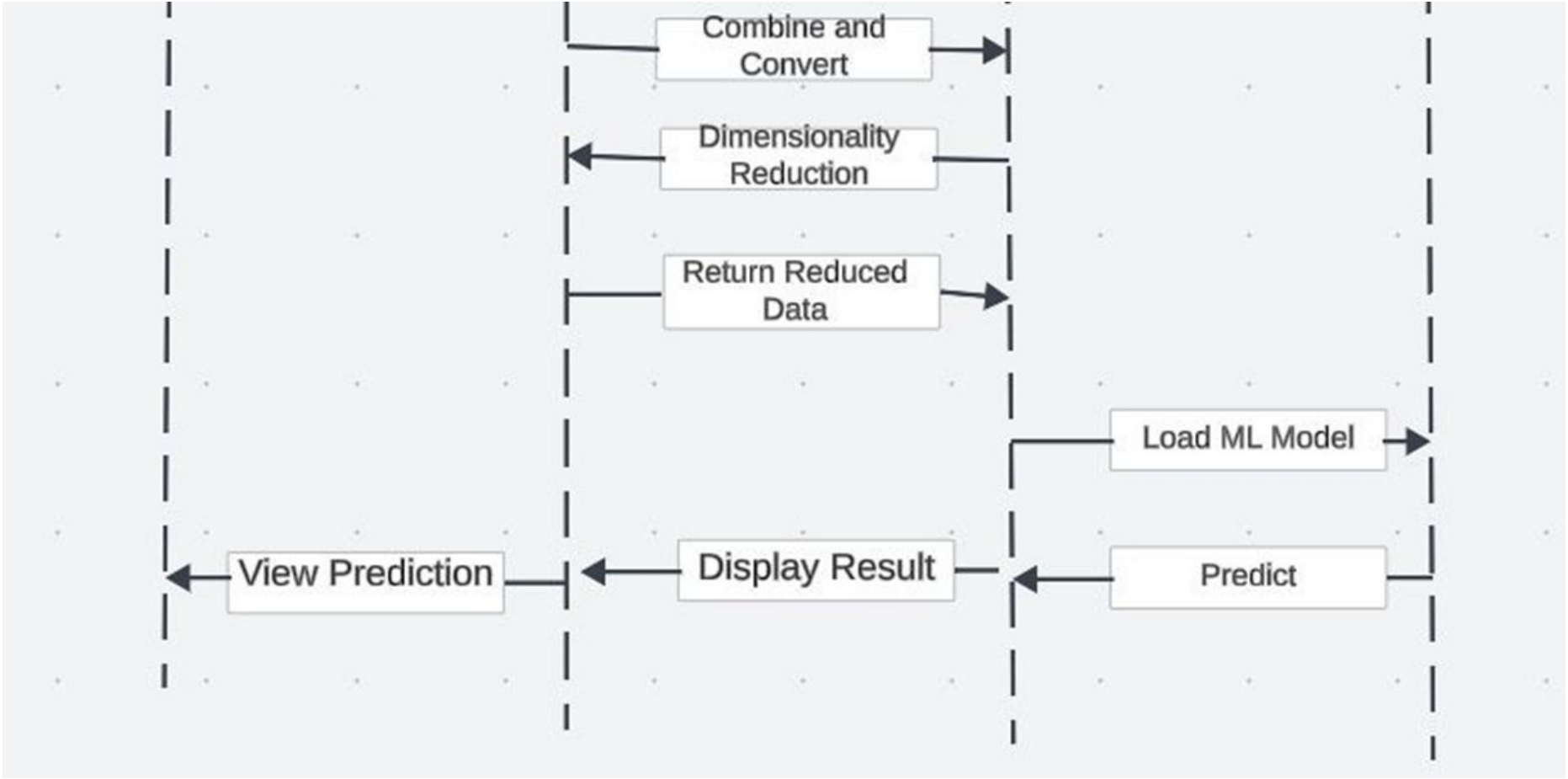
FINAL PREDICTED RESULT

USE CASE DIAGRAM



SEQUENCE DIAGRAM




Combine and Convert
Dimensionality Reduction
Return Reduced Data
Load ML Model
View Prediction
Display Result
Predict

IMPLEMENTATION

Module 1: Image Processing

- **Grayscale Conversion**: Transforming RGB ECG images into grayscale to simplify intensity-based analysis.

```
#converting to gray scale
grayscale = color.rgb2gray(y)
```

- **Gaussian Smoothing**: Applying a Gaussian filter to reduce noise and enhance signal clarity.

```
#smoothing image
blurred_image = gaussian(grayscale, sigma=0.7)
#thresholding to distinguish foreground and background
#using otsu thresholding for getting threshold value
global_thresh = threshold_otsu(blurred_image)

#creating binary image based on threshold
binary_global = blurred_image < global_thresh
#resize image
binary_global = resize(binary_global, (300, 450))
```

- **Contour Extraction**: Identifying and extracting contours to outline and isolate ECG waveforms.

```
#finding contour
contours = measure.find_contours(binary_global,0.9)

contours_shape = sorted([x.shape for x in contours])[::-1][0:1]
print(contours_shape)
```

- **Data Scaling**: Standardizing signal data to ensure consistency

```
#scaling the data and testing
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

fit_transform_data = scaler.fit_transform(df)
Normalized_Scaled=pd.DataFrame(fit_transform_data, columns = ['X','Y'])
```

Module 2: Front-end(Streamlit)

- **Importing the library**

```
import streamlit as st
```

- **Displaying images after GrayScaling**

```
if similarity_score > 0.70:
  st.image(image)
  """#### **GRAY SCALE IMAGE**"""
  my_expander = st.expander(label='Gray SCALE IMAGE')
  with my_expander:
    st.image(image_gray)
```

- **Dividing leads and displaying**

```
my_expander1 = st.expander(label='DIVIDING LEAD')
with my_expander1:
   st.image('Leads_1-12_figure.png')
   st.image('Long_Lead_13_figure.png')
```

- **Displaying Leads after Pre-Processing**

```
my_expander2 = st.expander(label='PREPROCESSED LEAD')
with my_expander2:
   st.image('Preprossed_Leads_1-12_figure.png')
   st.image('Preprossed_Leads_13_figure.png')
```

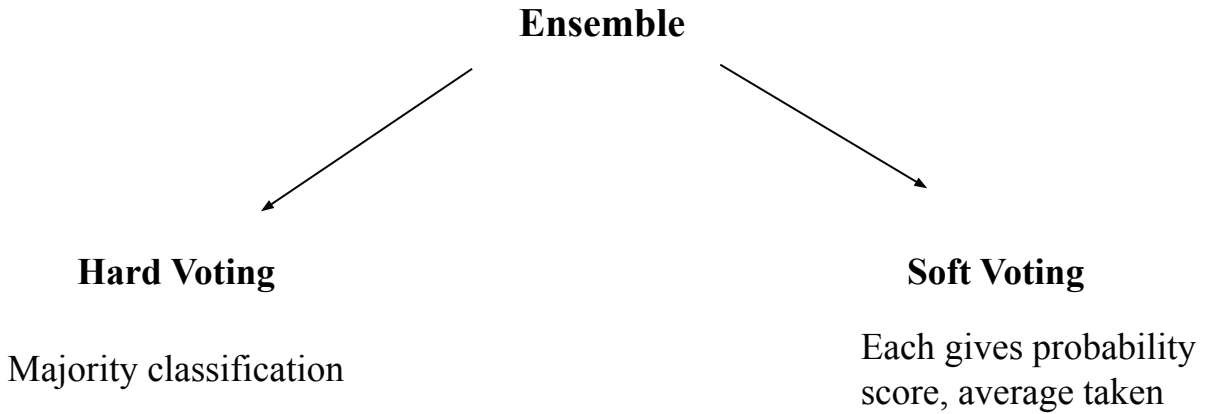
- **Displaying Contours**

```
fig4.savefig('Contour_Leads_1-12_figure.png')
my_expander3 = st.expander(label='CONOTUR LEADS')
with my_expander3:
   st.image('Contour_Leads_1-12_figure.png')
```

Module 3: Prediction Model

Ensemble Modeling for Cardiovascular Disease Prediction:

➔ Ensemble modeling is a technique that combines predictions from multiple base learners to form a robust prediction model, leveraging their collective wisdom to enhance accuracy and reduce variance



Logistic Regression

➔ Logistic regression is a statistical method used for classification that predicts the probability of a binary outcome based on one or more predictor variables by using a logistic function.

```
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report

# Input and target
X = result_df.iloc[:, :-1]
y = result_df.iloc[:, -1]

# Create train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

# Setup pipeline and grid search
pipeline = Pipeline([('lr', LogisticRegression())])
params = {'lr__C': np.logspace(-4, 4, 3), 'lr__penalty': ['l2']}
cv = GridSearchCV(pipeline, params, cv=2).fit(X_train, y_train)

# Predict and print metrics
y_pred = cv.predict(X_test)
print(f"Accuracy: {cv.score(X_test, y_test)}")
print(classification_report(y_test, y_pred))
```

k-Nearest Neighbors algorithm

➔ K-Nearest Neighbors (KNN) is a machine learning algorithm that classifies data points based on the majority class of their **k** closest neighbors in the feature space.

```
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report

# Create the pipeline
pipeline = Pipeline([('knn', KNeighborsClassifier())])

# Hyperparameter grid
k_range = range(1, 30)
parameters = dict(knn__n_neighbors=k_range)

# Input and target
X = final_result_df.iloc[:, :-1]
y = final_result_df.iloc[:, -1]

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

# Grid search with cross-validation
cv = GridSearchCV(pipeline, parameters, cv=2)
cv.fit(X_train, y_train)

# Predictions and evaluation
y_pred = cv.predict(X_test)
print(f"Accuracy: {cv.score(X_test, y_test):.2f}")
print(classification_report(y_test, y_pred))
print(f"Tuned Model Parameters: {cv.best_params_}")
```

Algorithm:

1. **Choose k**: Select the number of neighbors
2. **Calculate Distance:**

$$d(x_i, x) = \sqrt{\sum_{m=1}^{M} (x_{im} - x_m)^2}$$

3.**Find Neighbors**: Identify the k closest points.
4. **Prediction**:

$$\hat{y} = \text{mode}(y_1, \ldots, y_k)$$

Random Forest Algorithm

Random Forest is an ensemble learning method that uses multiple decision trees to improve classification and regression accuracy by averaging their predictions or taking a majority vote.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import joblib

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(final_df, y, test_size=0.3, random_state=42)

# Initialize the Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Fit the model to the training data
rf_classifier.fit(X_train, y_train)

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


# Save the model for future use
joblib.dump(rf_classifier, 'ecg_random_forest_model.pkl')
```

Algorithm

1. **Build Multiple Trees**:Random Forest creates N decision trees on random subsets of data and features.
2. **Bootstrap Sampling**:
Each tree trains on a randomly sampled subset of the data
3. **Random Feature Selection**:
At each node split, choose a random subset of features. For MMM total features, each split considers $\sqrt{M}$.
4. **Prediction (Classification)**:
Take the majority vote across all trees:

$$\hat{y} = \text{mode}(y_1, y_2, \ldots, y_N)$$

Support Vector Machine algorithm

Support Vector Machine (SVM) is a supervised machine learning algorithm that finds the optimal hyperplane to separate data points of different classes in high-dimensional space.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
import joblib

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(final_df, y, test_size=0.3, random_state=42)

# Initialize the SVM classifier
svm_classifier = SVC(kernel='linear', random_state=42)  # You can choose 'linear', 'rbf', etc.

# Fit the model to the training data
svm_classifier.fit(X_train, y_train)

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

# Save the model for future use
joblib.dump(svm_classifier, 'ecg_svm_model.pkl')
```

Algorithm

1. **Goal**: Find a hyperplane that maximizes the margin between classes.

2. **Hyperplane and margin:**

$$w^T x + b = 0 \qquad \text{Margin} = \frac{2}{\|w\|}$$

3. **Optimization**:

$$\min \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1$$

4. **Prediction**: $\hat{y} = \text{sign}(w^T x + b)$

Gaussian Naive Bayes algorithm

Gaussian Naive Bayes is a probabilistic classification algorithm that assumes features are normally distributed and uses Bayes' theorem to predict class probabilities based on the input features.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report


data = pd.read_csv('final12.csv')

# Assuming the last column is the target variable
X = data.iloc[:, :-1]  # Features
y = data.iloc[:, -1]   # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

# Initialize the Gaussian Naive Bayes classifier
gnb = GaussianNB()

# Fit the model on the training data
gnb.fit(X_train, y_train)

# Make predictions on the test data
y_pred = gnb.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
print(classification_report(y_test, y_pred))
```

Algorithm

1. **Calculate Priors**: For each class C , calculate P(C) from the training data

2. **Gaussian Fit**: For each feature Xi in each class C, calculate mean μ and variance σ2 for P(Xi|C)

3. **Predict:** $P(C|X) = P(C) \times P(X_1|C) \times P(X_2|C) \times \ldots$

Choose the class C with the highest probability.

Testing : Frontend and Image Processing Module

TEST CASE	MODULE	INPUT DESCRIPTION	EXPECTED RESULT	OBTAINED RESULT	STATUS
1.	FRONT-END (Streamlit)	Upload valid ECG image	ECG image displays correctly	ECG image displayed correctly	PASS
2.	Model Prediction	Evaluate the ensemble model on test data	Ensemble model generated	Training metrics working	PASS
3.	Image Processing	Provide an ECG image	Image processed and grayscale image returned	Grayscale image displayed	PASS
4.	Image Processing	Upload high-resolution ECG image	Contours detected and highlighted	Contours detected	PASS
5.	Image Processing	Apply binarization to ECG image	Binary image is generated	Binary image generated	PASS

Testing : Model Performance Metrics

Accuracy: 0.543010752688172

	precision	recall	f1-score	support
0	0.36	0.33	0.35	105
1	0.73	0.91	0.81	94
2	0.56	0.58	0.57	112
3	0.38	0.26	0.31	61
accuracy			0.54	372
macro avg	0.51	0.52	0.51	372
weighted avg	0.52	0.54	0.53	372

Tuned Model Parameters: {'lr__C': 10000.0, 'lr__penalty': 'l2'}

Accuracy: 0.853448275862069

	precision	recall	f1-score	support
0	0.79	0.70	0.74	119
1	0.98	1.00	0.99	125
2	0.82	0.87	0.84	140
3	0.80	0.82	0.81	80
accuracy			0.85	464
macro avg	0.85	0.85	0.85	464
weighted avg	0.85	0.85	0.85	464

Accuracy: 0.782258064516129

	precision	recall	f1-score	support
0	0.87	0.63	0.73	105
1	0.91	0.91	0.91	94
2	0.72	0.88	0.79	112
3	0.63	0.67	0.65	61
accuracy			0.78	372
macro avg	0.78	0.77	0.77	372
weighted avg	0.80	0.78	0.78	372

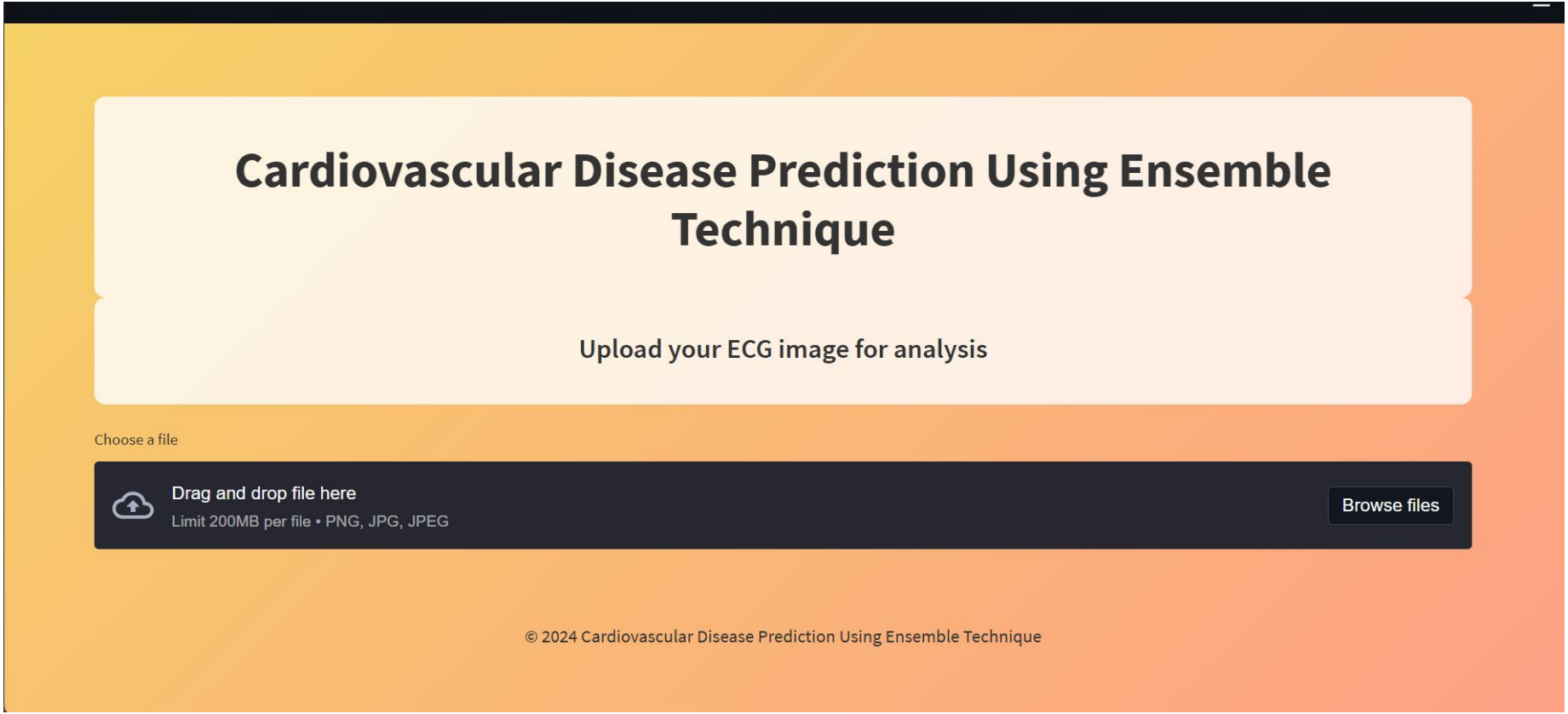
Tuned Model Parameters: {'knn__n_neighbors': 1}

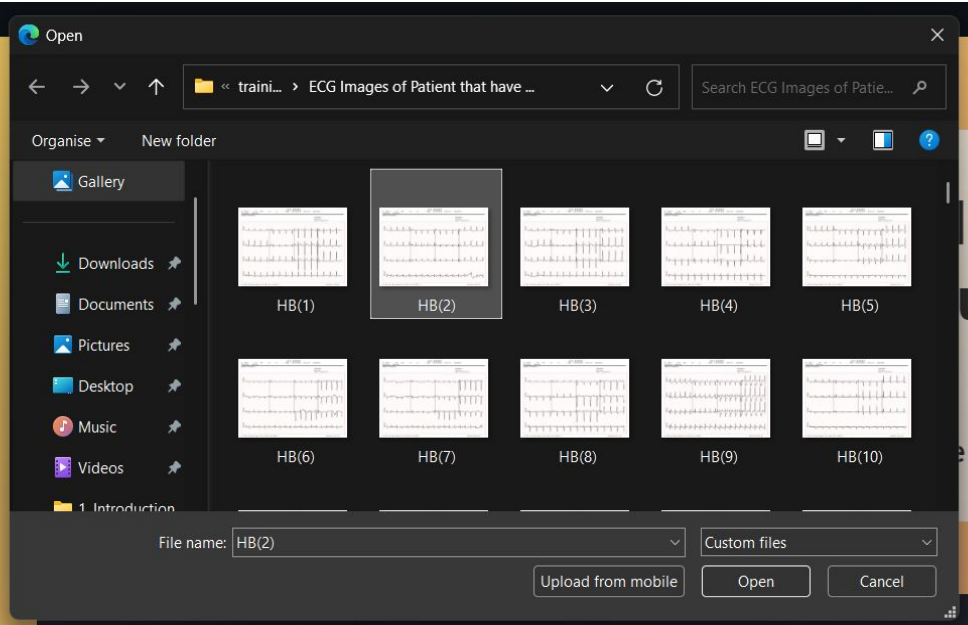
{'SVM__C': 1, 'SVM__gamma': 0.1, 'knn__n_neighbors': 1, 'rf__n_estimators': 300}
Accuracy: 0.9247311827956989

	precision	recall	f1-score	support
0	0.89	0.96	0.92	80
1	1.00	1.00	1.00	72
2	0.92	0.92	0.92	79
3	0.88	0.75	0.81	48
accuracy			0.92	279
macro avg	0.92	0.91	0.91	279
weighted avg	0.92	0.92	0.92	279

{'SVM__C': 1, 'SVM__gamma': 0.1, 'knn__n_neighbors': 1, 'rf__n_estimators': 300}

Results with Snapshots





iction Using Ensemble ue

e for analysis

Drag and drop file here

Limit 200MB per file • PNG, JPG, JPEG

Browse files

Uploaded ECG Image

Gray Scale Image

Dividing Leads

Preprocessed Leads

Extracting Signals

1D Signal Conversion

Dimensionality Reduction

Prediction Results Using Soft Voting method (averaging)

Prediction Results Using Hard Voting Method

References

[1] Tabassum, Tahmida, and Mohiuddin Ahmad. "Numerical data extraction from ECG paper recording using image processing technique." 2020 11th international conference on electrical and computer engineering (ICECE). IEEE, 2020.

[2] McAuliffe, Matthew J., et al. "Medical image processing, analysis and visualization in clinical research." Proceedings 14th IEEE symposium on computer-based medical systems. CBMS 2001. IEEE, 2001.

[3] Mahajan, Palak, et al. "Ensemble learning for disease prediction: A review." Healthcare. Vol. 11. No. 12. MDPI, 2023.

[4] Bauer, Eric, and Ron Kohavi. "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Their Variants." Machine Learning, vol. 36, no. 1-2, 1999, pp. 105–139. Springer, doi:10.1023/A:1007515423169

[5] Naderalvojoud, Behzad, and Tina Hernandez-Boussard. "Improving machine learning with ensemble learning on observational healthcare data." AMIA Annual Symposium Proceedings. Vol. 2023. 2024.


Thank you!