SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

ARRHYTHMIA DETECTION USING MACHINE LEARNING TECHNIQUES

CSE340 -MACHINE LEARNING
CA4 PROJECT REPORT

Submitted by

S. Samyuktha - E0321052
J. Janeika - E0321041
Sarah Madhavan- E0321037

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence and Data Analytics)

Sri Ramachandra Faculty of Engineering and Technology

Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116

April, 2024

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

ARRHYTHMIA DETECTION USING MACHINE LEARNING TECHNIQUES

CSE340 -MACHINE LEARNING
CA4 PROJECT REPORT

Submitted by

S. Samyuktha - E0321052
J. Janeika - E0321041
Sarah Madhavan- E0321037

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence and Data Analytics)

Sri Ramachandra Faculty of Engineering and Technology

Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116

April, 2024

ABSTRACT

- Cardiac arrhythmias pose a significant health risk, potentially leading to severe complications such as stroke and heart failure. Early detection of arrhythmias is crucial for preventing these complications. Machine learning (ML) techniques have demonstrated promise in various medical applications, including arrhythmia detection.
- This paper presents a lightweight ML approach for accurately detecting eight different cardiac arrhythmias along with normal rhythm. To harness ML methods, we apply resampling and baseline wander removal techniques to electrocardiogram (ECG) signals.
- In this study, we utilize 500 sample ECG segments as model inputs. Rhythm classification is performed using various ML algorithms, including support vector machines (SVMs) and decision trees, in an end-to-end manner, eliminating the need for manual feature extraction.
- We explore recent ML advances for cardiac arrhythmia detection, discussing the challenges associated with ML for arrhythmia detection and classification, such as the requirement for large datasets and the risk of overfitting.
- To evaluate the proposed technique, we select ECG signals from two PhysioNet databases: the MIT-BIH arrhythmia database and the long-term AF database. Our proposed ML framework demonstrates superior results compared to most state-of-the-art methods.

TABLE OF CONTENTS

TITLE	PAGE NO
ABSTRACT	3
TABLE OF CONTENTS	4
INTRODUCTION	5
LITERATURE REVIEW	8
PROPOSED METHODOLOGY	10
IMPLEMENTATION	11
RESULTS AND DISCUSSIONS	12
CONCLUSION & FUTURE ENHANCEMENT	20
REFERENCES	21

CHAPTER 1

INTRODUCTION

Heart

The human heart, a vital organ, pumps blood through a dual-circuit system. Deoxygenated blood enters the right atrium from the body via the superior and inferior vena cavae. It moves through the tricuspid valve into the right ventricle, then is pumped to the lungs through the pulmonary artery for oxygenation. Oxygen-rich blood returns to the left atrium via the pulmonary veins, flows through the mitral valve into the left ventricle, and is pumped out to the body via the aorta. This one-way flow is ensured by heart valves, maintaining the pressure necessary for efficient circulation.

Electrical system of the heart

The heart's electrical system initiates and coordinates its rhythmic contractions. It begins with an impulse from the sinus node, known as the heart's natural pacemaker. This impulse spreads through the atria, then to the atrioventricular (AV) node, Bundle of His, and Purkinje fibers, ultimately causing ventricular contraction.

A normal heart beats regularly due to organized electrical impulses, typically 60 to 100 times per minute in adults. Electrocardiography (ECG or EKG) records this electrical activity, aiding in the diagnosis of arrhythmias.

What Is an Arrhythmia?

Arrhythmias, or abnormal heartbeats, can arise from various heart regions, leading to irregular, too fast, or too slow rhythms. While some are harmless, others pose a risk of cardiac arrest. Symptoms include palpitations, chest discomfort, dizziness, and fatigue. Arrhythmias may stem from heart disease, electrolyte imbalances, medications, or emotional stress. Diagnosis and treatment, if necessary, are determined by healthcare providers based on symptoms and underlying causes. Monitoring heart rhythm is crucial as the heart supplies the body with vital nutrients and oxygen.

What Are the Symptoms of Arrhythmia?

An arrhythmia can be silent, meaning you don't notice any symptoms. Your doctor may spot an uneven heartbeat during a physical exam.This includes:

- Palpitations (a feeling of skipped heartbeats, fluttering, or "flip-flops")
- Pounding in your chest
- Dizziness or feeling lightheaded
- Fainting
- Shortness of breath
- Chest pain or tightness
- Weakness or fatigue (feeling very tired)
- Anxiety & Blurry vision
- Sweating

What Causes Arrhythmia?

It could happen because of:

- Heart disease
- The wrong balance of electrolytes (such as sodium or potassium) in your blood
- Heart injury or changes such as reduced blood flow or stiff heart tissue
- Healing process after heart surgery
- Infection or fever
- Certain medications
- Problems with the electrical signals in your heart
- Strong emotions, stress, or surprise
- Things in your daily life like alcohol, tobacco, caffeine, or exercise

What Is an Electrocardiogram?

- Cardiovascular disease is amongst the three main global causes of death. As reported by the World Health Organization, nearly 18 million people die each year from heart disease. An electrocardiogram — abbreviated as EKG or ECG — measures the electrical activity of the heartbeat. With each beat, an electrical impulse (or “wave”) travels through the heart. This electrical wave causes the muscle to contract and pump blood from the heart.

- A normal heartbeat on ECG will show the rate and rhythm of the contractions in the upper and lower chambers. A normal heart rhythm consists of three main parts including P waves, T waves and QRS complex.The occurrence of any anomalies in the rhythm and/or heart rate indicates the existence of a disorder and is called arrhythmia. Arrhythmia patients may experience symptoms like insufficient blood pumping, shortness of breath, exhaustion, chest pain, and unconsciousness.

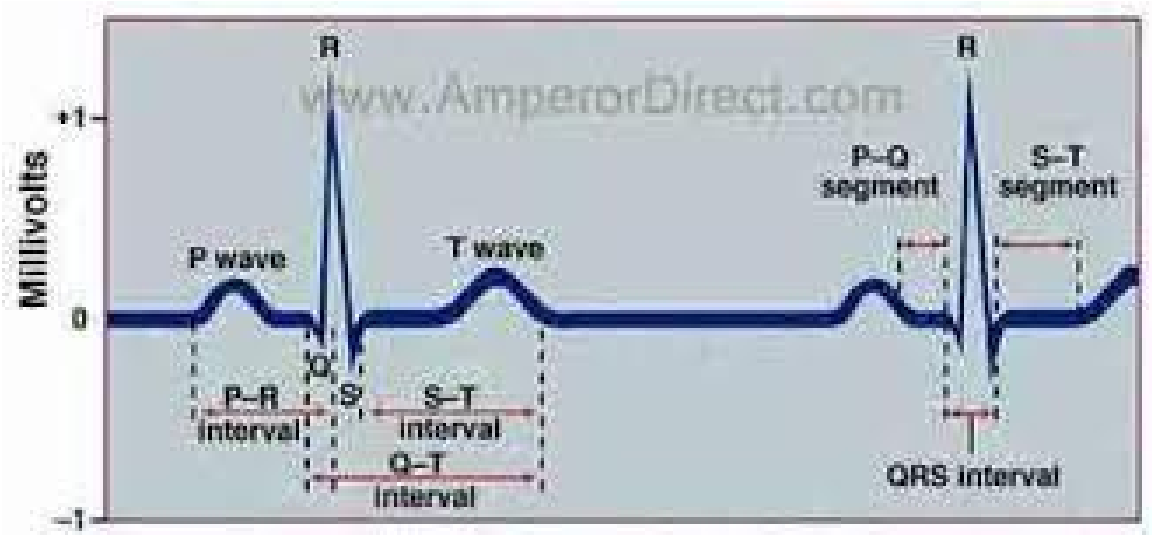

Fig 1.6: ECG

CHAPTER 2
LITERATURE REVIEW

S.NO	TITLE	AUTHORS	ABOUT	ACCURACY
1	Online Automatic Diagnosis System of Cardiac Arrhythmias Based on MIT-BIH ECG Database	Wei Yan and Zhen Zhang	Used wavelet transform and PCA to extract features, SVM to classify arrhythmia into 6 classes: Normal PVC PAB LBBB RBBB ST	Accuracy: 98.05%
2	Cardiac Arrhythmia detection using deep learning	Ali Isina and Selen Ozdalilib	AlexNet pre trained model is used for feature extraction . PCA is used to reduce no.of features . Trained and tested in n- fc6 , n-fc7 ,n-tst .	n fc7 -97.4 % n fc6 96.1 % n tst -90.1 %
3	A Study on Arrhythmia via ECG Signal Classification Using the Convolutional Neural Network	Mengze Wu, Yongdi Lu, Wenli Yang and Shen Yuong Wong	Used 12-layer CNN network to classify arrhythmia into 5 classes: Normal sinus rhythm	Accuracy: 97.21%
4	Automated detection of arrhythmias using different intervals of tachycardia ECG segments with convolutional neural network	U. Rajendra Acharya , Hamido Fujita , Oh Shu Lih, Yuki Hagiwaraa, Jen Hong Tan , Muhammad Adam	Used 11- layer deep CNN to classify arrhythmia into 4 classes: Normal sinus rhytm Atrial Fibrillation Atrial Flutter Ventricular Fibrillation	Accuracy: 92.50%
5	A Study Based on Methods Used in Detection of Cardiac Arrhythmia	C A Aakansha, Durva Dev, Sejal Kore	The study aimed to evaluate the performance of various machine learning algorithms for detecting cardiac arrhythmia. The models used were k-Nearest Neighbor, Decision Tree, Random Forest, Artificial Neural	The highest accuracy of 94.75% was achieved by the Random Forest algorithm.

6	Arrhythmia detection from 12-lead varied-length ECG using Attention-based Time-Incremental CNN	Qihang Yao, Ruxin Wang, Xiaomao Fan, Jikui Liu, Ye Li	The proposed model, Attention-based Time-Incremental CNN, was used for arrhythmia detection	It achieved an accuracy of 81.2% outperforming existing state-of-the-art models.
7	A lightweight hybrid CNN-LSTM model for ECG-based arrhythmia detection	Negin Alamatsaz, Leyla Tabatabaei, Mohammadreza Yazdchi, Hamidreza Payan, Nima Alamatsaz & Fahimeh Nasimi	The study proposed a hybrid CNN-LSTM model for arrhythmia detection using ECG signals.	achieved an accuracy of 98.24%
8	Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation	Amin Ullah ,Syed Muhammad Anwar,Muhammad Bilal,Raja Majid Mehmood	2-D CNN model for the classification of ECG signals into eight classes NOR,VFW, PVC, VEB, RBB, LBB, PAB, and APC.	99.11% average accuracy
9	An Automated System for ECG Arrhythmia Detection Using Machine Learning Techniques	Mohamed Sraitih,Younes Jabraneand Amir Hajjam El Hassani	Automatic classification system using a new comprehensive ECG database inter-patient paradigm separation without performing any features extraction classes classified: NOR,RBBB,LBBB,PAC,PVC	SVM classifier accuracy: 0.83 %
10	A clinical study on Atrial Fibrillation, Premature Ventricular Contraction, and Premature Atrial Contraction screening based on an ECG deep learning model	Jianyuan Hong , Hua-Jung Li , Chung-chi Yang , Chih-Lu Han , Jui-chien Hsieh	ECG interpreter to improve the performance of AF, PVC, and PAC based on clinical ECGs. a deep learning model to delineate ECG features such as P, QRS, and T waves a sliding window with 3-RR intervals in length applied to raw ECG to examine the delineated features in the window	delineator performance on P-wave-sensitivity/specificity of 0.94/0.98 QRS wave-sensitivity/specificity of 1.00/0.99,and T-wave - sensitivity/specificity of 0.97/0.98. F1 measure ,of AF, PVC, and PAC were 0.92, 0.91, and 0.83, respectively

CHAPTER 3

PROPOSED METHODOLOGY

Utilization of Supervised Machine Learning Algorithms:

- Utilize a combination of supervised machine learning algorithms for classification, including Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree, Random Forest (Ensemble), and Naïve Bayes.

Data Preprocessing:

- Perform data preprocessing by splitting the dataset into training and test sets. Use an 80-20 split, where 80% of the data is used for training and 20% for testing.

Implementation of Machine Learning Algorithms:

- Implement each of the selected machine learning algorithms on the training dataset.
- Configure each algorithm with appropriate parameters and settings.

Model Training:

- Train each machine learning model on the training dataset.

Model Evaluation:

- Evaluate the performance of each trained model using confusion matrix and accuracy metrics.
- Confusion matrix: Understand the model's classification performance, providing insights into true positive, true negative, false positive, and false negative predictions.
- Accuracy: Measure the overall correctness of the model's predictions.

Optimization and Tuning:

- Focus on improving each model's predictive power and generalization ability by optimizing algorithm selection and tuning parameters.
- Experiment with different hyperparameters and feature selection techniques to optimize model performance.

Robust Evaluation:

- Ensure robust evaluation by considering multiple evaluation metrics rather than solely relying on accuracy.
- Assess each model's performance comprehensively across the confusion matrix and accuracy metrics.

CHAPTER 4

IMPLEMENTATION

Data Collection:

Data was gathered from reliable sources such as the MIT-BIH Arrhythmia Dataset and the PTB Diagnostic ECG Database, ensuring the collected data's trustworthiness and quality. Each data contain five classes/categories: N (Normal), S (Supraventricular ectopic beat), V (Ventricular ectopic beat), F (Fusion beat), and Q (Unknown beat).

Techniques Involved:

- **Noise Filtering:**Noise filtering techniques, including median filtering, were applied to remove noise and improve signal quality.
- **Resampling:**ECG recordings from different databases may have varying sampling frequencies.Resampling techniques were used to standardize the sampling frequency, ensuring compatibility across datasets.

Feature Extraction using VGG16:

ECG signals were processed using the pre-trained VGG16 model to extract features. Spectrogram images were generated to represent the signals, allowing for the capture of high-level representations of heartbeat patterns.

Normalization of Extracted Features:

The extracted features were normalized for each class separately to ensure consistent scaling and enhance subsequent analysis steps. Normalization techniques were applied using Excel.

Combining Normalized Files:

The normalized feature files were consolidated into a single dataset, preserving the original class labels to maintain classification integrity. The combined dataset was stored in Excel format for further processing.

Model Training and Evaluation:

Multiple machine learning models, including Logistic Regression, Random Forest, KNN, Decision Forest, SVM, and Naive Bayes, were trained on the normalized dataset. The dataset was split into training and testing sets (80% training, 20% testing) for model evaluation.

Model performance was assessed using **accuracy** and **confusion matrix** metrics on the testing set.

CHAPTER 5

RESULTS AND DISCUSSIONS

1. LOGISTIC REGRESSION:

Analyzing the performance measures – accuracy and confusion matrix, we can clearly say that our model is performing really well.and the accuracy was 66%.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
```

```
data=pd.read_excel("/content/drive/MyDrive/data.xlsx")
```

```
#train the model
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state = 0)
classifier.fit(xtrain, ytrain)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/linear_m
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale t
    https://scikit-learn.org/stable/modules/preprocessi
Please also refer to the documentation for alternative
    https://scikit-learn.org/stable/modules/linear_mode
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression(random_state=0)
```

```
y_pred = classifier.predict(xtest)
```

```
#evaluation matrix

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(ytest, y_pred)
print ("Confusion Matrix : \n", cm)
```

```
Confusion Matrix :
 [[ 82  26   0   0  16   2]
 [ 42  13   0   0   5   1]
 [  0   0 116 109   0   0]
 [  0   0 130 111   0   0]
 [ 21   7   0   0  13   1]
 [  0   0   0   0   0 366]]
```

```
from sklearn.metrics import accuracy_score

print ("Accuracy : ", accuracy_score(ytest, y_pred))
```

```
Accuracy :  0.6606974552309143
```

```
# Print the Confusion Matrix and slice it into four pieces
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(ytest, y_pred)
print('Confusion matrix\n\n', cm)
print('\nTrue Positives(TP) = ', cm[0,0])
print('\nTrue Negatives(TN) = ', cm[1,1])
print('\nFalse Positives(FP) = ', cm[0,1])
print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix

 [[ 82  26   0   0  16   2]
 [ 42  13   0   0   5   1]
 [  0   0 116 109   0   0]
 [  0   0 130 111   0   0]
 [ 21   7   0   0  13   1]
 [  0   0   0   0   0 366]]

True Positives(TP) =  82

True Negatives(TN) =  13

False Positives(FP) =  26

False Negatives(FN) =  42
```

```
from sklearn.metrics import classification_report
print(classification_report(ytest, y_pred))

TP = cm[0,0]
TN = cm[1,1]
FP = cm[0,1]
FN = cm[1,0]
```

	precision	recall	f1-score	support
1	0.57	0.65	0.61	126
2	0.28	0.21	0.24	61
3	0.47	0.52	0.49	225
4	0.50	0.46	0.48	241
5	0.38	0.31	0.34	42
6	0.99	1.00	0.99	366
accuracy			0.66	1061
macro avg	0.53	0.52	0.53	1061
weighted avg	0.65	0.66	0.66	1061

2. KNN:

For the KNN classification(N_NEIGHBOURS=5) the accuracy received after standardizing using the standard scaler was 63.9%,meaning from test data the model could properly label 63.9% of the labels correctly after the features are fed into this KNN model.by using grid search cv to find the best value of n_neighbours the accuracy was 66.1%. The accuracy improved when grid search cv was used.

```
#to go with the best n_neighbour count

from sklearn.model_selection import GridSearchCV
# Define the parameter grid to search over
param_grid = {'n_neighbors': [1, 3, 5, 7, 9]}
```

```
# Create a KNN classifier
clf = KNeighborsClassifier()

# Use GridSearchCV to find the best value of n_neighbors
grid_search = GridSearchCV(clf, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)
```


```
# Predict on the test set
y_pred = clf.predict(X_test)
```

```
# Evaluate the performance of the classifier using accuracy score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy score: {accuracy}")
```

```
Accuracy score: 0.6395759717314488
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import recall_score
# y_true: true labels, y_pred: predicted labels
cm = confusion_matrix(y_test,y_pred)
```

```
# Print the confusion matrix
print(cm)
```

```
[[ 84  17   0   0   4   9]
 [ 28   7   0   0   0   5]
 [  0   0 102  91   0   0]
 [  0   0 100  97   0   0]
 [ 22   9   0   0   5   4]
 [ 13   2   0   0   2 248]]
```

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.57      0.74      0.64       114
           2       0.20      0.17      0.19        40
           3       0.50      0.53      0.52       193
           4       0.52      0.49      0.50       197
           5       0.45      0.12      0.20        40
           6       0.93      0.94      0.93       265

    accuracy                           0.64       849
   macro avg       0.53      0.50      0.50       849
weighted avg       0.63      0.64      0.63       849
```

3. NAIVE BAYES:

The accuracy received for Naive Bayes model was 63.36%

```
from sklearn.naive_bayes import GaussianNB
nv = GaussianNB()
nv.fit(X_train,y_train)

from sklearn.metrics import accuracy_score

y_pred = nv.predict(X_test)

print(accuracy_score(y_test,y_pred))
```

```
0.6336866902237926
```

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

print('Confusion matrix\n\n', cm)

print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix

 [[ 38  28   0   0  30  10]
 [ 16   8   0   0  13   3]
 [  0   0 100  81   0   0]
 [  1   0  96  85   1   0]
 [ 10   1   0   0  17   3]
 [  6   7   0   0   5 290]]

True Positives(TP) =  38

True Negatives(TN) =  8

False Positives(FP) =  28

False Negatives(FN) =  16
```

```
# print classification accuracy

classification_accuracy = (TP + TN) / float(TP + TN + FP + FN)

print('Classification accuracy : {0:0.4f}'.format(classification_accuracy))

# print classification error

classification_error = (FP + FN) / float(TP + TN + FP + FN)

print('Classification error : {0:0.4f}'.format(classification_error))

# print precision score

precision = TP / float(TP + FP)


print('Precision : {0:0.4f}'.format(precision))

recall = TP / float(TP + FN)

print('Recall or Sensitivity : {0:0.4f}'.format(recall))
```

```
Classification accuracy : 0.5111
Classification error : 0.4889
Precision : 0.5758
Recall or Sensitivity : 0.7037
```

4. SVM:

Once after the standardization and scaling using standard scaler the data was fed into the SVM model and the accuracy achieved was 66.5%.For the test data the model has recognized 66.5% of the labels correctly.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
clf = svm.SVC(kernel='linear',probability=True)
clf.fit(X_train, y_train)
```

```
SVC(kernel='linear', probability=True)
```

```
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy score: {accuracy}")
```

```
Accuracy score: 0.6654099905749293
```

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.57      0.57      0.57       143
           2       0.21      0.27      0.24        52
           3       0.53      0.56      0.54       235
           4       0.55      0.51      0.53       243
           5       0.38      0.24      0.29        46
           6       0.99      1.00      1.00       342

    accuracy                           0.67      1061
   macro avg       0.54      0.53      0.53      1061
weighted avg       0.67      0.67      0.67      1061
```

5. RANDOM FOREST CLASSIFIER:

The accuracy received for random forest classifier was 62.66%

```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score
```

```
x=data.iloc[:,0:1000].values
y=data.iloc[:, 1000].values
```

```
X_train, X_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

```
rf = RandomForestClassifier(oob_score=True)
```

```
rf.fit(X_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(oob_score=True)
```

```
rf.oob_score_
```

```
0.655081001472754
```

```
y_pred = rf.predict(X_test)
accuracy_score(y_test,y_pred)
```

```
0.6266195524146054
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))

TP = cm[0,0]
TN = cm[1,1]
FP = cm[0,1]
FN = cm[1,0]
```

```
              precision    recall  f1-score   support

           1       0.60      0.65      0.62       114
           2       0.00      0.00      0.00        40
           3       0.49      0.59      0.53       193
           4       0.50      0.41      0.45       197
           5       0.00      0.00      0.00        40
           6       0.80      1.00      0.89       265

    accuracy                           0.63       849
   macro avg       0.40      0.44      0.42       849
weighted avg       0.56      0.63      0.59       849
```

6. DECISION TREE CLASSIFIER:

The accuracy received for the decision tree classifier was 54.47% . The model could properly label just a little more than half of the test data correctly.

```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import
from sklearn.model_selection import train_test_split # Im
from sklearn import metrics #Import scikit-learn metrics
```

```
features = data.iloc[:, :999]

labels = data.iloc[:, 1000]
```

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.3, random_state=1)
```

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()
```

```
# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)
```

```
#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.5447409733124019
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import recall_score
# y_true: true labels, y_pred: predicted labels
cm = confusion_matrix(y_test,y_pred)
```

```
# Print the confusion matrix
print(cm)
```

```
[[ 74  33   2   2  21  37]
 [ 32  12   2   2   7  11]
 [  4   4 147 122   1   8]
 [  6   1 132 125   2  10]
 [ 21  12   1   5   6   2]
 [ 38  23  11  13  15 330]]
```

```
from sklearn.metrics import classification_report
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           1       0.44      0.42      0.43       175
           2       0.18      0.14      0.16        85
           3       0.51      0.50      0.51       295
           4       0.45      0.46      0.46       269
           5       0.13      0.12      0.12        52
           6       0.77      0.83      0.80       398

    accuracy                           0.54      1274
   macro avg       0.41      0.41      0.41      1274
weighted avg       0.53      0.54      0.54      1274
```

```
from sklearn.metrics import classification_report
print(classification_report(y_pred,y_test))
TP = cm[0,0]
TN = cm[1,1]
FP = cm[0,1]
FN = cm[1,0]
```

```
              precision    recall  f1-score   support

           1       0.44      0.42      0.43       175
           2       0.18      0.14      0.16        85
           3       0.51      0.50      0.51       295
           4       0.45      0.46      0.46       269
           5       0.13      0.12      0.12        52
           6       0.77      0.83      0.80       398

    accuracy                           0.54      1274
   macro avg       0.41      0.41      0.41      1274
weighted avg       0.53      0.54      0.54      1274
```

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENT

Artificial intelligence (AI) is a vast field encompassing various techniques that allow machines to exhibit human-like cognitive abilities, such as learning and problem-solving. Machine learning (ML) is a powerful subfield within AI that focuses on algorithms with the capability to learn from data without being explicitly programmed. These algorithms progressively improve their performance as they are exposed to more data.

Deep learning, a specialized form of machine learning, utilizes artificial neural networks inspired by the structure of the human brain. These networks excel at processing unstructured data like images, text, and audio. Deep learning models have a large number of parameters that are learned during training. Additionally, they rely on hyperparameters, which are set before training to control the learning process. Finding the ideal values for these hyperparameters can be a time-consuming but crucial step in optimizing the model's performance.

The core strength of deep learning lies in its ability to discover intricate patterns within data and make independent, proactive predictions. This makes it a valuable tool for various applications within the broader field of AI.

By applying the lessons learned from training multiple machine learning models, the aim is to develop deep learning architectures that can better capture the complexities of ECG data and provide more accurate predictions of cardiac arrhythmias. This transition to deep learning represents a natural progression in our pursuit of more advanced AI techniques to address the challenges in medical diagnostics.

REFERENCES:

1. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8618527/ 2. https://www.mdpi.com/2072-4292/12/10/1685
a. .https://www.ahajournals.org/doi/10.1161/CIRCRESAHA.120.317 872
3. https://www.frontiersin.org/articles/10.3389/fncom.2020.564015/full
4. .https://www.sciencedirect.com/science/article/pii/S2590188520300123 5. .https://www.mdpi.com/1424-8220/22/23/9347
6. https://www.irjet.net/archives/V9/i4/IRJET-V9I4200.pdf
7. https://arxiv.org/pdf/2209.00988.pdf
8. https://www.sciencedirect.com/science/article/pii/S1566253518307632
9. .https://www.sciencedirect.com/science/article/abs/pii/S0022073619304571
10. .https://www.hindawi.com/journals/sp/2021/9926769/?utm_source=google&utm_medium=cpc&utm_campaign=HDW_MRKT_GBL_SUB_ADWO_PAI_DYNA_JOUR_X_PJ_GROUP3&gclid=Cj0KCQiA6fafBhC1ARIsAIJjL8lQ60KsDAwHvJ6o4j4PjkP1aXJnwQ6S85kRxvGGQ_SnmVbbhzrvjlgaAuwYEALw_wcB
11. https://www.sciencedirect.com/science/article/abs/pii/S0950705121009862#preview-section-references
12. .https://nrl.northumbria.ac.uk/id/eprint/44509/1/Transactions_DNN_paper_2_5_2020_1_EH%20%281%29.pdf
13. https://www.mdpi.com/2077-0383/10/22/5450
14. https://www.kaggle.com/datasets/erhmrai/ecg-image-data
15. https://www.mathworks.com/discovery/feature-extraction.html#:~:text=Feature%20
17. https://scikit-learn.org/stable/modules/cross_validation.html