JSS MAHAVIDYAPEETHA

JSS Science and Technology University

"Prediction of Cardiac Arrhythmia using Artificial Neural Network"

A technical project report submitted in partial fulfillment of the award of the degree of

MASTER OF COMPUTER APPLICATIONS

IN

DEPARTMENT OF COMPUTER APPLICATIONS

BY

SHUKRASHREE H J

01JST20PMC049

UNDER THE GUIDANCE OF

Prof. CHAITHRA C S

Assistant Professor

Department of Computer Application

JSS STU, Mysuru-06

2021-2022

Department of Computer Applications

JSS MAHAVIDYAPEETHA

JSS Science and Technology University

Certificate

This is to certify that the work entitled

"**Prediction of Cardiac Arrhythmia using Artificial Neural Network**" is a Bonafide work carried out by **Shukrashree H J (01JST20PMC049)** in partial fulfilment of the award of the degree of Master of Computer Applications in Department of Computer Applications for the award of Master of Computer Applications by JSS Science and Technology University, Mysuru, during the year 2021-2022. The project report has been approved as it satisfies the academic requirements in respect to project work prescribed for the Master of Computer Applications in Department of Computer Applications.

Under the guidance of

Prof. Chaithra C S
Assistant Professor
Dept of Computer Applications
JSS STU, Mysuru-06

Head of the department

Dr. V.N. Manjunath Aradhya
Associate Professor and HOD,
Dept of Computer Applications
JSS STU, Mysuru-06

Examiners:

1……

2……

Certificate of Plagiarism.

RE-2022-39817-plag-report

ORIGINALITY REPORT

11%	7%	7%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	medium.com Internet Source	4%
2	Zhi Li, Harm Derksen, Jonathan Gryak, Cheng Jiang, Zijun Gao, Winston Zhang, Hamid Ghanbari, Pujitha Gunaratne, Kayvan Najarian. "Prediction of cardiac arrhythmia using deterministic probabilistic finite-state automata", Biomedical Signal Processing and Control, 2021 Publication	1%
3	Submitted to universityofexeter Student Paper	1%
4	www.tandfonline.com Internet Source	1%
5	Sangeeta Saha, Neema Singh, Bhawana Rudra. "Chapter 1 Detection of Denial of Service Attack Using Deep Learning and Genetic Algorithm", Springer Science and Business Media LLC, 2022 Publication	1%

6	Siddhant Bagga, Anish Goyal, Namita Gupta, Arvind Goyal. "Credit Card Fraud Detection using Pipeling and Ensemble Learning", Procedia Computer Science, 2020 Publication	<1 %
7	Manoj Athreya A*, Avani H S, Pooja, Madhu S, Dr. Paramesha K. "Detection of Cardiac Arrhythmia using Machine Learning Algorithms", International Journal of Recent Technology and Engineering (IJRTE), 2019 Publication	<1 %
8	Submitted to University of Ulster Student Paper	<1 %
9	Submitted to Thadomal Shahani Engineering College Student Paper	<1 %
10	Submitted to Coventry University Student Paper	<1 %
11	"ICICCT 2019 – System Reliability, Quality Control, Safety, Maintenance and Management", Springer Science and Business Media LLC, 2020 Publication	<1 %
12	"International Conference on Innovative Computing and Communications", Springer Science and Business Media LLC, 2021 Publication	<1 %

DECLARATION

I do hereby declare that the project titled **"Prediction of Cardiac Arrhythmia using Artificial Neural Network"** is carried out by me, under the guidance of **Prof. Chaithra C S**, **Assistant Professor**, **Department of Computer Applications** JSS Science and Technology University, Mysuru, in partial fulfilment of requirement for the award of Master of Computer Application by JSS Science and Technology University, Mysore, during the year 2021-2022.

I also declare that I have not submitted this dissertation to any other university for the award of any degree or diploma courses.

Date: **SHUKRASHREE HJ**

Place: Mysore **01JST20PMC049**

ACKNOWLEDGMENT

The success of any endeavor depends a lot on the goals set at the onset as well as the constant guidance and motivation received throughout. It's my duty to acknowledge and thank the individuals who has contributed in the successful completion of the project.

I express my deep sense of gratitude and sincere thanks to my respected guide **Prof. Chaithra C S,** Assistant Professor**,** Department of Computer Applications, sustaining interest and dynamic guidance shown in aiding me to complete this project immaculately and impeccably and being the source of my strength and confidence.

I feel immense pleasure to thank **Dr. V N Manjunath Aradhya,** Associate Professor and Head, Department of Computer Applications, for his encouragement and support throughout the project.

I express my heartfelt thanks to our principal **Dr. S B Kivade** at the esteemed institution **JSS Science and Technology University, Mysuru** for providing me an opportunity to reach my goal.

I sincerely express our thanks and gratitude to our institution **JSS Science and Technology University, Mysuru - 570006** for providing me an opportunity to fulfill our most cherished desire of reaching my goal and thus helping me to make a bright career.

I would like to thank all the **Teaching and Non-Teaching Staff** of Department of MCA for their kind Co-operation during the course of the work. The support provided by the **Departmental library** is gratefully acknowledged.

This successful completion of my project would not have been possible without **my parents Sacrifice, guidance and prayers**. I take this opportunity to thank very much for their continuous encouragement. I convey my thankfulness to all **my friends who were with me to share my happiness and agony**. They gave valuable suggestion which was the solution that helped me to a great extent to complete the project successfully.

SHUKRASHREE H J

List of Contents

CHAPTER 6: Discussion and Results.

CHAPTER 7: Conclusion and Future Enhancements.

CHAPTER 8: References

ABSTRACT

.

Classification of Arrhythmia with high accuracy is an important and challenging task. Arrhythmia which is considered as a life-threatening disease must be accurately predicted and multi classified so that the life span can be increased. The dataset is accessed from the UCI database. Pre-processing and normalization steps have been done before prediction and classification of cardiac arrhythmia. Important features are selected from the co-relation matrix along with pca. The data is normalized by using a standard scalar and cleaning of data is carried out by imputing the mean values replacing the missing values. Neural network model MLP (multilayer perceptron) is used for classification and prediction of Arrhythmia. The 70% of the dataset are used to train the MLP neural network. The proposed algorithm is predicted with next 30% of the datasets.

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

People today suffer from a variety of chronic illnesses. One condition that affects a lot of people is heart disease. Another factor in many heart attacks is stress. By early detection and prompt treatment of arrhythmia, which will lessen the risk of heart attacks in people and also minimise the loss of life, these unpleasant heart attacks and unexpected death can be avoided. The most often used device or tool for assessing cardiac capacity is an ECG. This is captured when cathodes are placed on the body and the heart's electrical drive is demonstrated. P waves, QRS waves, and T waves make up ECG signals. For the purpose of comprehending the heart, it is necessary to connect the P waves, QRS waves, T waves, and RR interims over time and in a particular shape. Arrhythmia is a category of anomalies in heartbeat where the heart beats too slowly or too quickly, leading to cardiac diseases. Ai algorithms can be linked together to increase the accuracy of ordering cardiac arrhythmias from ECG readings.

The setting of usage and information investigation requirements of the selected patient are key factors in classifying heart arrhythmias and determining the best course of action. We provide a useful paradigm for categorising ECG signals into groups that distinguish between the existence but also absence of arrhythmia in this article. The location where the dataset has been segregated is the UCI AI storage. The data are divided into 16 different classes using multiclass characterisation, with normal being at the top of the list and cardiovascular arrhythmias at the bottom.

When that happens, pre-processing is finished for the selected highlight to ensure uniformity in the information's delivery. To increase the precision and predictability of cardiovascular arrhythmia, multilayer perceptron (MLP) is now being connected, developed, and evaluated on the standard dataset. SVM strategy, for example, one-against-one, one-against-all, blunder code and other arrangement calculation, for example, Random Forest, Logistic Regression, Gradient Boosting and Ensemble technique are also being connected, prepared,

and tested. Compared to other grouping calculations, gathering is thought to perform admirably and outperform them in terms of accuracy in forecasting and treating heart arrhythmia.

1.2 PROBLEM STATEMENT

Most cardiac conditions result in irregular heartbeats. Arrhythmia is the term used to describe these erratic patterns in heartbeat rhythm. Clinical professionals most frequently utilise electrocardiograms (ECGs) to record heartbeats. ECG is recognised for being affordable, simple to use, and noninvasive to human anatomy. The use of machine learning models as well as techniques to detect and predict the kind of arrhythmia depending on Electrocardiogram (ECG) instrument will be efficient and reliable.

1.3 AIM

The suggested framework aims to demonstrate a cardiac arrhythmia classification and prediction method based on artificial neural networks. Using a variety of machine learning techniques and neural networks, the study attempts to predict and classify arrhythmia into a number of categories.

1.4 OBJECTIVES

The system's goals are as follows:

- Our goal is to use an individual patient's ECG readings to assign him to one of the 16 Arrhythmia classes, which will help us better understand how machine learning is used in the medical field.
- Objective To reduce the number of future heart disease-related fatalities.
- To create a system capable of accurately detecting an arrhythmia.

- To create a technique for categorising an ECG trace stably into one of 13 broad arrhythmia groups.

- Comparative analysis of the applicable algorithms and determination of the top algorithm for cardiac arrhythmia prediction.

CHAPTER 2
LITERATURE SURVEY

Understanding and evaluating become crucial if you want to make a contribution to this area of development. the approaches that were previously in use. As a result, the literature review for the research project was completed, and the contributions of many authors were examined.

2.1 LITERATURE SURVEY:

1. Deep learning for the identification of cardiac arrhythmias:

By classifying patient ECGs into the relevant cardiac disorders with the use of a deep learning framework that has been trained or educated on a general image data set, automated ECG arrhythmia diagnosis being carried out in this study. The characteristics gathered by a really deep convolutional neural network are sent to a straightforward back propagating neural network in order to execute the classification model (specifically, Alex Net).

2. Heart arrhythmia type prediction using clustering and regression methodology:

In the sort of disease caused by cardiac arrhythmias, this research suggests a diagnosis or prediction strategy. Regression and a clustering strategy are both used. DBSCAN is the clustering method used, and multi - class classification logistic regression is used for regression. The DBSCAN clustering technique divides the entire dataset into several groups. The clusters that are discovered to have fewer instances are then taken into account. The multiclass logistic regression method is applied to these clusters. This is due to the unsupervised nature of the clustering approach.

3. Finite-state automata with deterministic probabilistic behaviour for cardiac arrhythmia prediction

This paper proposes a novel method for categorising and predicting the incidence of cardiac arrhythmias using a specific class of finite state automata (DPFA). The proposed method constructs the fundamental state space of the DPFA model, including the transition probabilities related to the input data. Supraventricular tachycardia (SVT) and atrial high-rate episodes were the two separate cardiac occurrences that the algorithm's efficiency was matched to five other well-known approaches for (AHRE).

4. Cardiac Arrhythmia Prediction and Classification:

This study tries to identify and categorise 14 different types of arrhythmia. Feature selection, Naive Bayes, Support Vector Machine, Random Forests, and Neural Networks are some of the popular techniques from recent literature that were used. Additionally, a novel strategy combining Random Forests and SVM classifiers was put into practise.

5. Cardiac Arrhythmia Detection Using Neural Networks from PCG Signals:

Cardiac arrhythmias, a type of heart condition, enable the heart to pulse perhaps too slowly or too fast. It takes more time for doctors to diagnose and treat this critical cardiac condition, therefore it needs to be found as soon as possible. There are several types of arrhythmias; bradycardia is characterised by a slow heartbeat, whereas tachycardia is characterised by a rapid heartbeat.

2.2 TABLE OF LITERATURE SURVEY:

SL No	Title	Author	Methodology	Conclusion
01	Predict cardiac arrhythmia using deterministic probabilistic finite state automata (DPFA)	ZhiLi Horm Derksen 2020	A brand-new technique using DPFA for categorising and forecasting cardiac arrhythmia. 181 ECG signals from cardiac patients at Michigan Medicine make up the dataset.	Over 80% accuracy was achieved by the suggested strategy. The quantity of cases limits the performance.
02	Deep Learning for the diagnosis of Cardiac Arrhythmias	Ali Isin selenozodalili 2017	In order to extract features, AlexNet is employed. The collected features are used in a straightforward back propagation algorithm to categorise among three different rhythms. MIT-BIH database ECG data.	98.51 percent recognition rate and 92 percent testing accuracy were attained. inadequate data
03	Cardiac Arrhythmia Type Prediction Utilizing Clustering and Regression Method.	Prathibhamol CP 2021	Uses the clustering (DBSCAN) strategy and the multicast logistic regression regression methodology to forecast the kind of anomaly.	Overall, it reaches an accuracy of 80%. less effective

04	Heart rate time series are used to forecast cardiac arrhythmias using a radial basis function network.	J P kelwade S S Salankar 2016	Using the MIT-BIH Arrhythmia database, predict 8 heart arrhythmias.	96.33 percent of predictions were correct overall.
05	Arrhythmia Classification Using ECG Signals and a Deep Convolutional Neural Network with Optimization.	Dinesh kumar atal 2020	Proposes the use of the Bat-Rider optimization method for automatic rhythmia classification..	Provides 93.19 percent total accuracy.
06	Based on morphological and time frequency features, arrhythmia prediction	Elham Zeraatkar Saeed kermani 2011	Convolutional neural network (CNN)	The proposed methodology has a 92.14 percent accuracy rate for classifying the morphological arrhythmia.
07	categorization of cardiac arrhythmias based on ECG signal segments	EJS luz 2016	Collective learning, evolutionary computation, and deep learning	The method has a 99.37 percentage accuracy rate in classifying ECG among 17 classes as well as 15 types of arrhythmias.
08	classification of arrhythmias into atrial fibrillation (AF), supraventricular	Erkan Kuraly 2004	Maximum Mutual Information Estimation (MMIE) theory in conjunction with	The suggested strategy successfully categorises the

	arrhythmia (S), premature ventricular contraction (V), normal (N), and supraventricular arrhythmia (S)		Hidden Markov Models (HMM)	ECG into the appropriate arrhythmia categories.
09	Identification and classification of four heart conditions	DH kim 2018	The Pan Tompkins algorithm for feature extraction	Long-term atrial fibrillation (AF), supraventricular arrhythmia, as well as sleep apnea arrhythmia were the four cardiac disorders that were categorised into these three groups by the proposed technique.
10	Heartbeat classification for the purpose of identifying cardiac arrhythmias	W ullah 2021	Spiking neural network (SNN)	The proposed approach has a 95.7 percent accuracy rate for detecting arrhythmia and categorises ECG heartbeats into one of 23 types.

2.3 OVERALL REVIEW OF LITERATURE SURVEY

- Case volume is capped.
- Not enough feature extraction.
- Takes longer to process and is less reliable.
- Less Effective.

2.4 PROPOSED SYSTEM

The standard repository of machine learning data, known as the UCI repository, is where the necessary data of prediction is gathered. Following data collection, the most pertinent features are extracted using a co-relation matrix and principal component analysis. Gradient boosting, Random Forest, Logistic Regression, and support vector machine (SVM) invariants like one-against-one, one-against-rest, error code, and multilayer perceptron are just a few of the machine learning classifiers that this study recommends for the prediction of cardiac arrhythmia. The accuracy of these algorithms' classification and prediction of cardiac arrhythmias is then compared. so that the most precise machine learning classifier for arrhythmia prediction may be determined.

CHAPTER 3
DATASET

3.1 DATASET

The dataset for the research was obtained from the UCI Machine Learning Repository at https://archive.ics.uci.edu/ml/datasets/Arrhythmia. There are (452) rows, each of which represents a patient's medical file. There are 279 attributes, including information about the patient's ECG, weight, and age.

The data set has 16 different classifications assigned to it. Classes 2 through 15 represent various types of arrhythmia, and class 16 denotes an unnamed patient. Class 1 corresponds to a normal ECG with no arrhythmia.

Title: Cardiac Arrhythmia Database

Sources: UCI Machine Learning Repository

Number of Instances: 452

Number of Attributes: 279

In [4]: data

Out[4]:

	age	sex	height	weight	qrs_duration	p-r_interval	q-t_interval	t_interval	p_interval	qrs	...	KY	KZ	LA	LB	LC	LD	LE	LF	LG	diagnosis
0	75	0	190	80	91	193	371	174	121	-16	...	0.0	9.0	-0.9	0.0	0	0.9	2.9	23.3	49.4	8
1	56	1	165	64	81	174	401	149	39	25	...	0.0	8.5	0.0	0.0	0	0.2	2.1	20.4	38.8	6
2	54	0	172	95	138	163	386	185	102	96	...	0.0	9.5	-2.4	0.0	0	0.3	3.4	12.3	49.0	10
3	55	0	175	94	100	202	380	179	143	28	...	0.0	12.2	-2.2	0.0	0	0.4	2.6	34.6	61.6	1
4	75	0	190	80	88	181	360	177	103	-16	...	0.0	13.1	-3.6	0.0	0	-0.1	3.9	25.4	62.8	7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
447	53	1	160	70	80	199	382	154	117	-37	...	0.0	4.3	-5.0	0.0	0	0.7	0.6	-4.4	-0.5	1
448	37	0	190	85	100	137	361	201	73	86	...	0.0	15.6	-1.6	0.0	0	0.4	2.4	38.0	62.4	10
449	36	0	166	68	108	176	365	194	116	-85	...	0.0	16.3	-28.6	0.0	0	1.5	1.0	-44.2	-33.2	2
450	32	1	155	55	93	106	386	218	63	54	...	-0.4	12.0	-0.7	0.0	0	0.5	2.4	25.0	46.6	1

Figure 3.1.1 Dataset

3.2 ATTRIBUTE INFORMATION

Complete attribute documentation:

1 Age: Age in years , linear
2 Sex: Sex (0 = male; 1 = female) , nominal
3 Height: Height in centimeters , linear
4 Weight: Weight in kilograms , linear
5 QRS duration: Average of QRS duration in msec., linear
6 P-R interval: Average duration between onset of P and Q waves in msec., linear
7 Q-T interval: Average duration between onset of Q and offset of T waves in msec., linear
8 T interval: Average duration of T wave in msec., linear
9 P interval: Average duration of P wave in msec., linear
Vector angles in degrees on front plane of:, linear
10 QRS
11 T
12 P
13 QRST
14 J
15 Heart rate: Number of heart beats per minute ,linear

Of channel DI:
Average width, in msec., of: linear
16 Q wave
17 R wave
18 S wave
19 R' wave, small peak just after R
20 S' wave
21 Number of intrinsic deflections, linear
22 Existence of ragged R wave, nominal
23 Existence of diphasic derivation of R wave, nominal
24 Existence of ragged P wave, nominal
25 Existence of diphasic derivation of P wave, nominal
26 Existence of ragged T wave, nominal
27 Existence of diphasic derivation of T wave, nominal

Of channel DII:
28 .. 39 (similar to 16 .. 27 of channel DI)
Of channels DIII:
40 .. 51
Of channel AVR:
52 .. 63
Of channel AVL:
64 .. 75
Of channel AVF:
76 .. 87

Of channel V1:
88 .. 99
Of channel V2:
100 .. 111
Of channel V3:
112 .. 123
Of channel V4:
124 .. 135
Of channel V5:
136 .. 147
Of channel V6:
148 .. 159

Of channel DI:
Amplitude , * 0.1 milivolt, of
160 JJ wave, linear
161 Q wave, linear
162 R wave, linear
163 S wave, linear
164 R' wave, linear
165 S' wave, linear
166 P wave, linear
167 T wave, linear

168 QRSA , Sum of areas of all segments divided by 10, (Area= width * height / 2), linear
169 QRSTA = QRSA + 0.5 * width of T wave * 0.1 * height of T wave. (If T is diphasic then the bigger segment is considered), linear

Of channel DII:
170 .. 179
Of channel DIII:
180 .. 189
Of channel AVR:
190 .. 199
Of channel AVL:
200 .. 209
Of channel AVF:
210 .. 219
Of channel V1:
220 .. 229

Of channel V2:
230 .. 239

Of channel V3:
240 .. 249
Of channel V4:
250 .. 259
Of channel V5:
260 .. 269
Of channel V6:
270 .. 279

CHAPTER 4
DESIGN

After the data has been gathered, processing it is a subsequent stage. The main challenges in analyzing this data set are the tiny proportion of training instances to features, the bias strongly in favour of a normal ECG condition, the absent values of a features, as well as the feature values that fall into all of continuous and categorical categories.

4.1 DATA PREPROCESSING

The source document as from UCI machine learning repository includes columns that have both incomplete values and single values that are the same for each patient record. The related columns of the dataset were eliminated. An enormous number of dimensions in the data, especially in issues involving many classes, contribute to poor classification accuracy. Therefore, dimensionality reduction techniques are used to decrease the dimension and remove duplicate data from the dataset.

4.2 FEATURE SELECTION

Prior to classification, it is necessary to decrease data features since they are crucial to the categorization of arrhythmias. In this, the most pertinent characteristics are chosen and the remaining attributes are filtered out using a Co-Relation Matrix and Principal Component Analysis.

Correlation is a method for determining the linear relation among two or more variables. Through correlation, we can predict some variable based on another. Because the desirable variables have a strong correlation with the target, correlation can be used to select features. Variables should also be uncorrelated among themselves while being correlated with the objective.

If two variables are associated, we can predict one variable from another. Thus, if two factors are connected, the model only genuinely needs one of them because the other does not add any new

knowledge. We'll use the Pearson Correlation in this situation.

We must establish an absolute value, let's say 0.5, as the cutoff for choosing the variables. If indeed the predictor variables are considered to be connected with one another, we can eliminate the predictor variable with the lowest correlation to the target variable. To determine whether more than two variables are associated to one another, we also can compute multiple correlation coefficients. Multicollinearity is the term for this phenomenon.

Q3

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	44	age	sex	height	weight	qrs_durati	p-r_interv	q-t_interv	t_interval	p_interval	qrs	q_wave	r_wave	s_wave	diagnosis
2	0	75	0	190	80	91	193	371	174	121	-16	0	52	44	8
3	1	56	1	165	64	81	174	401	149	39	25	0	48	0	6
4	2	54	0	172	95	138	163	386	185	102	96	0	40	80	10
5	3	55	0	175	94	100	202	380	179	143	28	0	72	20	1
6	4	75	0	190	80	88	181	360	177	103	-16	0	48	40	7
7	5	13	0	169	51	100	167	321	174	91	107	0	36	48	14
8	6	40	1	160	52	77	129	377	133	77	77	0	44	0	1
9	7	49	1	162	54	78	0	376	157	70	67	0	44	36	1
10	8	44	0	168	56	84	118	354	160	63	61	0	40	0	1
11	9	50	1	167	67	89	130	383	156	73	85	0	44	40	10
12	10	62	0	170	72	102	135	401	156	83	72	20	36	48	3
13	11	45	1	165	86	77	143	373	150	65	12	0	40	28	1
14	12	54	1	172	58	78	155	382	163	81	-24	0	72	0	10
15	13	30	0	170	73	91	180	355	157	104	68	0	92	0	6
16	14	44	1	160	88	77	158	399	163	94	46	0	80	0	1
17	15	47	1	150	48	75	132	350	169	65	36	0	48	0	1
18	16	47	0	171	59	82	145	347	169	61	77	0	48	0	10

Figure 4.2.1 Dataset after feature extraction

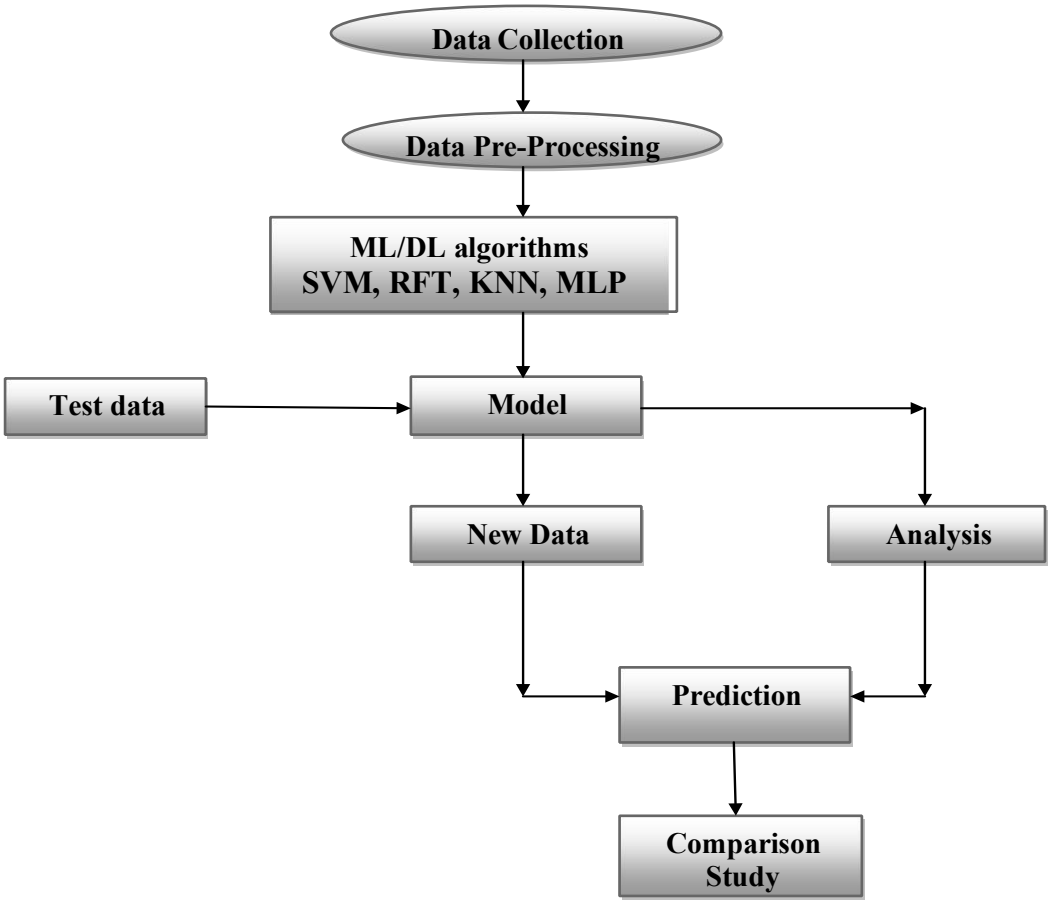


Figure 4.2.2 System Archietecture

4.3 MACHINE LEARNING OVERVIEW

Machine learning is a type of artificial intelligence (AI) that application providers can understand without explicit programming. The basic objective of machine learning is to create computer programmes that really can change in response to new data. This post will cover the foundations of machine learning as well as the Py implementation of a simple machine learning algorithm.

In machine learning, a computer is educated using a predetermined data set, and it utilizes this training to anticipate the characteristics of a predetermined new data set. For instance, we could train a computer by displaying it with 1000 images of cat and 1000 more images that do not feature cats, and inquiring it each moment if the image depicts a cat or not. If we send this new

image to the computer and train it as outlined previously, it should be willing to inform whether or not something is a cat.The training as well as prediction processes involve the employment of specialised algorithms. After receiving the training data, an algorithm uses it to create predictions about brand-new test data.

4.4 ALGORITHMS USED

1. K-NEAREST NEIGHBOR

KNN is another another sluggish algorithm. This indicates that generalisation cannot be made using training data. There is, to put it another way, essentially no conscious training period. This implies that the training process also passes quickly. KNN retains all training data because generalisation is not possible. The testing phase must employ every training data in order to achieve higher accuracy.

Depending on feature similarity, the KNN algorithm We classify a particular piece of data based on how closely out-of-sample attributes match our training set:

KNN can be applied to classification jobs with the goal of determining class membership. The classification of an object is decided by a majority vote among its neighbours, and the object is then given to the class with the greatest number of members among its k closest neighbours. It can also be used for regression, with the outcome being the value of the object (predicts continuous values). The values of its k closest neighbours were averaged to create this value (or median).

Input: uploading datasets

begin

1. Review the data (storage servers). retrieval of the necessary data from servers, such as databases, the cloud, excel sheets, etc. for mining.
2. Establish K = the number of closest neighbours.

3. Determine the separation between each training sample and the query instance. Although there are many different distance functions, Euclidean is the most widely used one.

4. Sort the distance to find the neighbours closest to you using the K-th minimal distance.

5. Call together the closest neighbours in category X.

6. Use the simple majority of the nearest neighbours as the query instance's prediction value.

The pseudocode for the Knn algorithm is as follows:

Let (X_i, C_i) where i = 1, 2……., n be data points. X_i denotes feature values & C_i denotes labels for X_i foreach

Let x be a point for which label is not known, and we would like to find the label class using k-nearest neighbor algorithms.

1. Calculate "d(x, x_i)" i =1, 2, ….., **n**; where **d** denotes the Euclidean_distance between the points.
2. def euclidean_distance(x,y):
3. return sqrt(sum(pow(a-b,2) for a, b in zip(x, y)))
4. Arrange the calculated **n** Euclidean distances in non-decreasing order.
5. Let **k** be a +ve integer, take the first **k** distances from this sorted list.
6. Find those **k**-points corresponding to these **k**-distances.
7. Let $\mathbf{k}_i$ denotes the number of points belonging to the ith class among **k** points i.e. $k \geq 0$
8. If $k_i > k_j$ $\forall$ $i \neq j$ then put x in class i.

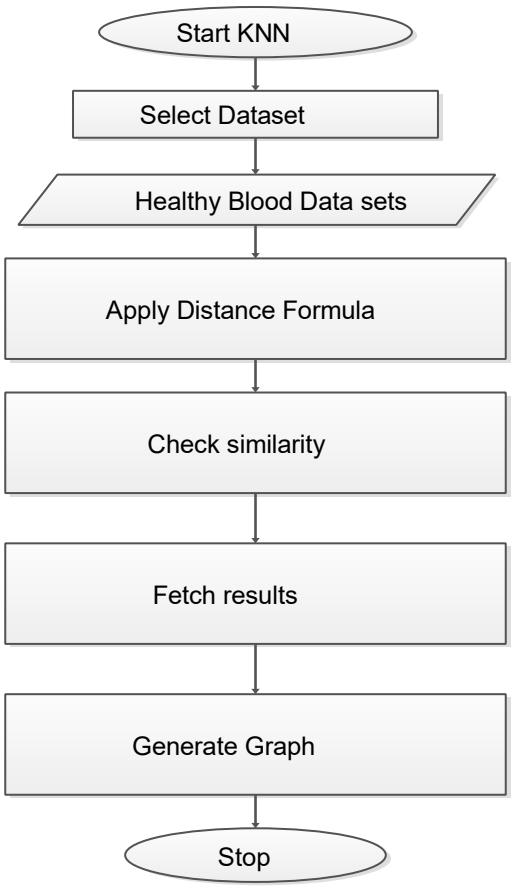


Figure 4.4.1 KnnFlow Diagram

2. NAÏVE BAYES:

Naive Bayes is the name of a statistical classification technique built on the Bayes Theorem. This is one of the simplest supervised learning techniques. A naive Bayes classifier is a reliable, quick, and accurate method. Naive Bayes classifiers function rapidly and precisely on large datasets.

A Naive Bayes classifier works under the premise that a feature's influence on a class is independent of the influence of other attributes. For instance, the eligibility of a loan applicant is influenced by their geography, age, record of loans and activities, and income. Despite the fact that these attributes are related, they are nonetheless considered independently. This assumption is regarded as naive because it makes calculation easier. The term "class conditional independence" refers to this presumption.

- Step 1: Determine the likelihood probability for each characteristic for each class in.
- Step 2. Determine the prior probability for the provided class labels in.
- Step 3. Enter these values into the Bayes formula and estimate the posterior probability.
- Step 4: Determine which class, given that the input corresponds to a higher probability class, has a greater likelihood.

Naïve Bayes Steps:

- Derivation:
- D : Set of tuples
- Each Tuple is an 'n' dimensional attribute vector
- X : (x1,x2,x3,…. xn)
- Let there be 'm' Classes : C1,C2,C3…Cm
- Naïve Bayes classifier predicts X belongs to Class Ci iff
- P (Ci/X) > P(Cj/X) for 1<= j <= m , j <> i Maximum Posteriori Hypothesis
- P(Ci/X) = P(X/Ci) P(Ci) / P(X)
- Maximize P(X/Ci) P(Ci) as P(X) is constant With many attributes, it is computationally expensive to evaluate P(X/Ci). Naïve Assumption of "class conditional independence"
- ∏= = n k P X Ci P xk Ci 1 (xk./ Ci)
- P(X/Ci) = P(x1/Ci) * P(x2/Ci) *…* P(xn/ Ci)

P(A|B) = Fraction of worlds in which B is true that also have A true

P(A ^ B) P(A|B) = ---------- P(B)

Corollary:

P(A ^ B) = P(A|B) P(B) P(A|B)+P(¬A|B) = 1

3. DECISION TREE:

Decision tree analysis is a powerful predictive modelling approach with numerous applications. Decision trees are frequently constructed using an algorithmic technique that looks for ways to segment a data collection based on a number of criteria. It is one of the most well-liked and effective methods for supervised learning. Both classification or regression applications use decision trees, a non-parametric supervised learning method. To create a model that predicts the target variable, the goal is to learn simple decision rules derived from the data attributes.

Building Decision Trees

The root node of the tree represents the complete training dataset and is constructed top-down, recursively, and divide-and-conquer style.

1. The node will be a leaf and will be labelled with that class if the training lists produce the same results.
2. If not, the tree divides the set according to the attribute with the most information and labels the node with its name.
3. Repeat the procedure and terminate until all samples belong to the same class, there are no more samples, or there are new attributes for the section.
4. Tree Ends.

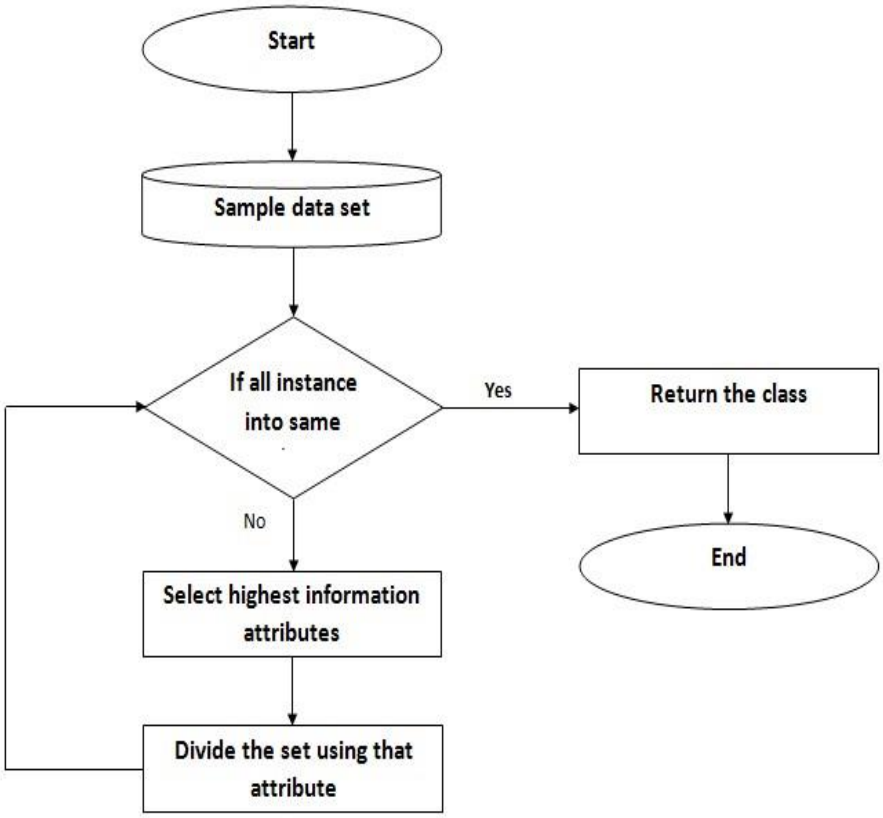


Figure 4.4.2 Decision Tree Flow Chart

Decision Tree Code

```
infoGain(examples, attribute, entropyOfSet)
  gain = entropyOfSet
  for value in attributeValues(examples, attribute):
     sub = subset(examples, attribute, value)
     gain -=  (number in sub)/(total number of examples) * entropy(sub)
  return gain
```

Entropy

```
entropy(examples)
  '''log2(x) = log(x)/log(2) '''
  result = 0
  # handle target attributes with arbitrary labels
```

```
dictionary = summarizeExamples(examples, targetAttribute)
for key in dictionary:
    proportion = dictionary[key]/total number of examples
    result -= proportion * log2(proportion)
return result
```

4. MULTI LAYER PERCEPTRON (MLP)

An MLP is utilised in this work under close monitoring. The weights are altered using the backpropagation method. An Artificial Neural Network (ANNarchitecture)'s and the weight values used are representations of the knowledge of the domain experts in ANNs. It is therefore exceedingly difficult to explain to a specific domain how an ANN generated its results. To solve this issue and offer an explanation again for network's output, we employ an IF/THEN-style rules extraction technique. It is significant to note that experts are more likely to accept these concepts because they mirror human logic.

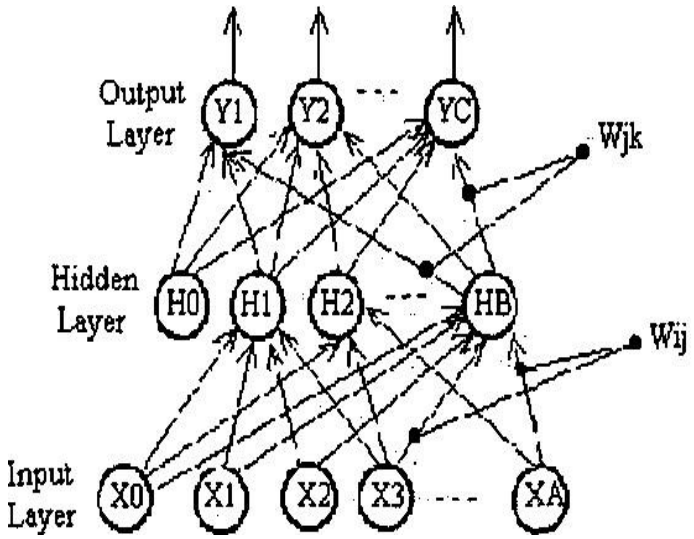


Figure 4.4.3 General MLP Architecture

When we discover a classification error or miss-classification, we adjust the weight.

The multilayer perceptron may contain much more than linear layers. In the simple case of a three-layer network, the top layer is the input layer, the bottom layer is the output layer, and the middle layer is known to as the hidden layer. Our input data is received by the input layer, and our output is received by the output layer. We are free to increase the hidden layers more than we want to make the building more complex in order to achieve our goals.

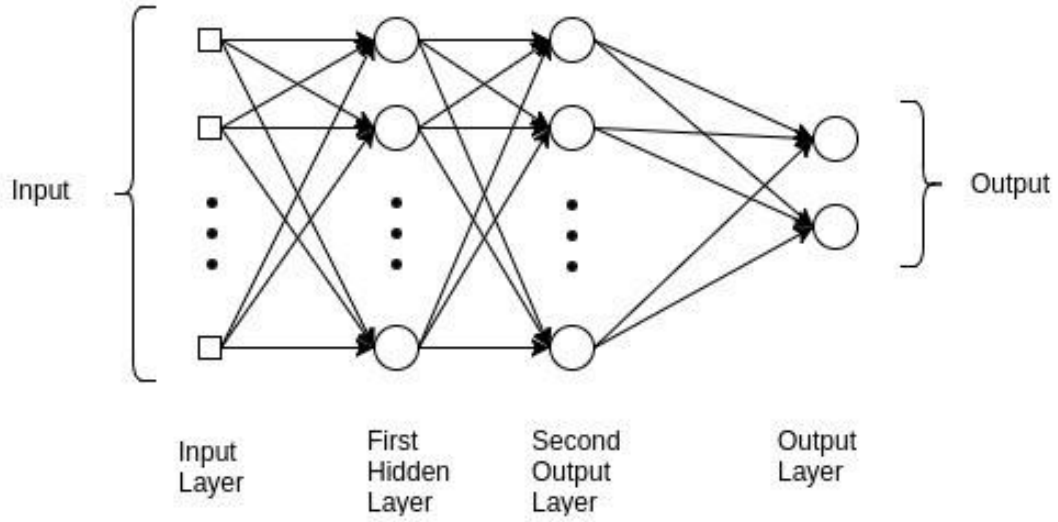


Figure 4.4.4.Proposed MLP Architecture

The feed-forward network is the most often used neural network model. Its goal is to simulate some function f. (). The MLP can figure out the best approach to a classification, such as y = f(x), that translates an input x to an outputs class y by creating a mapping, y = f(x), and learning the optimal parameters for it. The many linked together functions that make up the MLP networks. The formula for a multiple or three-layer network is f(x) = f(3)(f(2)(f(1)(x))). Units that translate a quadratic summing of data into a convolution make up each of these levels. The equation y = f(WxT + b) designates each layer. Where W is the collection of variables, or weights, in the layer, x is the input picture (which can also be the output of the layer before it), f is the perceptron (explained below), and b is the bias vector. An MLP contains numerous fully connected layers because each unit in a layer is connected to any unit in the layer before everything. Since each component inside a fully linked layer has separate attributes from other units in the layer, each

unit will have its own set of weights.

weight = weight + learning_rate * (expected - predicted) * x

Inside a supervised classification scheme, the class label is either provided with the data and then each input vector does have a label, or ground truth, indicating its class. Each input receives a category score, or predictions, as the network's output. The loss function is established in order to evaluate the classifier's performance. If the anticipated class does not match the actual class, there will be a significant loss; otherwise, there will be little loss. When training the model, the overfitting and underfitting issues can occasionally arise. Our model works exceptionally well on training examples in this situation, and not on testing data. We need a loss function and an optimization to perform the optimization method that will train the network.

Iteratively modifying the weights to obtain a lower loss involves initialising them with random values. This refinement is carried out by changes in the direction suggested by the slope of a loss function. The learning rate, which indicates how much more the algorithm improves with each iteration, must also be provided.

Activation function**:**

Activation functions, also referred to as non-linearity, define input-output linkages. This gives the model additional flexibility to express any type of relationship. These popular activation techniques include TanH, Relu, and Sigmoid. I'll talk about these in a future blog post.

Training the Model-

The model training process consists essentially of three parts.

1. Forward pass
2. Calculate error or loss
3. Backward pass

1. Forward pass

In this stage of training the model, we merely provide the input, combine it with weights, impose bias at every layer, and finally retrieve the predicted output of the model.

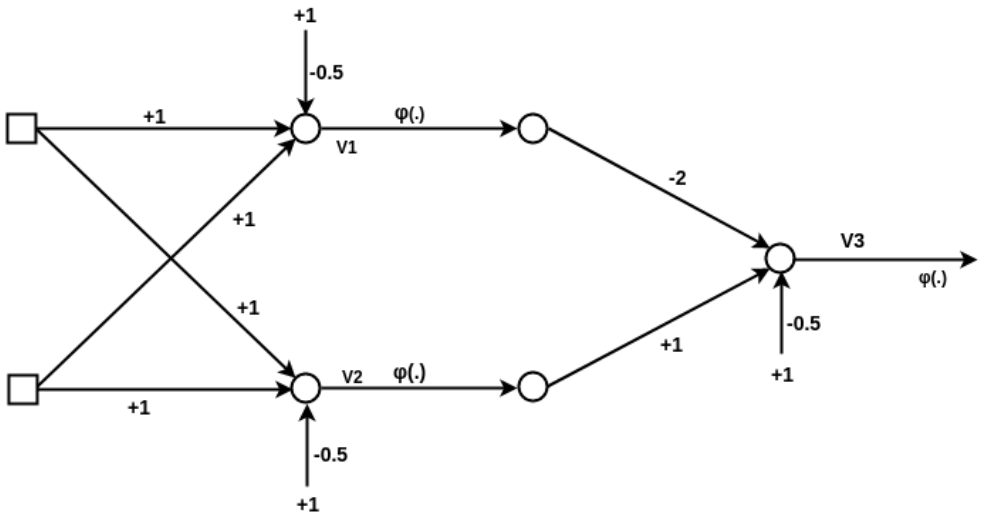


Figure 4.4.5 Forward Pass in MLP

2. Loss Calculate

If we pass the set of data , we will obtain any output first from model known as anticipated output (pred out), since we have the label both with data that signals actual output or expected output . Based on these two, we calculate the damage we must backpropagate. We use several loss functions depending on our outputs and requirements.

3. Backward Pass

After determining the loss, we use gradient to back propagating the damage and update the model's weights. The primary stage in training the model is this one. Weights will be adjusted in this phase to reflect the slope movement in that direction.

4.4.5 PREDICTIONS:

01	Normal
02	Ischemic changes (Coronary Artery Disease)
03	Old Anterior Myocardial Infarction
04	Old Inferior Myocardial Infarction
05	Sinus tachycardy
06	Sinus bradycardy
07	Ventricular Premature Contraction (PVC)
08	Supraventricular Premature Contraction
09	Left bundle branch block
10	Right bundle branch block
14	Left ventricule hypertrophy
15	Atrial Fibrillation or Flutter
16	Others

CHAPTER 5
IMPLEMENTATION

5.1 INTRODUCTION

Python, which is both an object-oriented and a procedure-oriented programming language, is used to carry out the project. By constructing partitioned memory areas of the both data and function which may be used as a template for building additional copies of such modules as needed, object-oriented programming is a method that offers a technique to modularize programmes.

This project is being executed using the Python programming language. Python has dynamic typing and garbage collection. All programming paradigms, including procedural and object-oriented ones, are supported. Python is frequently described as being "batteries contained" due to its large standard library. Machine learning techniques are used in this research.

5.2 CODE

```
#import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import ssl
import math
import operator
from collections import defaultdict
from sklearn import preprocessing
from sklearn.model_selection import train_test_split, KFold
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import normalize, StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import LinearSVC, SVC
```

```
 from sklearn.metrics import confusion_matrix, classification_report, f1_score,accuracy_score
 from sklearn.feature_selection import SelectFromModel
 from sklearn.ensemble import RandomForestClassifier

data=pd.read_csv(r"E:\data_arrhythmia.csv", delimiter=';')
data
data.isnull().sum()
data["diagnosis"].value_cou

#list of well known features with missing data
familiar_features = ['age','sex','height','weight','heart_rate']

#function that creates an histogram for a feature
def print_hist(df,feature,nbins):
    print("Histogram for " + feature + ":")
    column = df[feature]
    plt.hist(column,bins=nbins)
    plt.show()
for feature in familiar_features: print_hist(data,feature,30)
  X = data.drop(columns = [data.columns[-1]])
  y = data[data.columns[-1]]

# Splitting into training and testing data
        X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.3, shuffle = True,
  stratify = y, random_state=43)
# Splitting into training and validation data
#X_train, X_val, Y_train, Y_val = train_test_split(X_trainval,
Y_trainval,test_size=0.2,shuffle=True, stratify = Y_trainval, random_state=43)
print(X_train.shape, Y_train.shape, X_test.shape, Y_test.shape)

from sklearn.utils import class_weight
```

```
class_wt = class_weight.compute_class_weight('balanced', np.unique(Y_train), Y_train)
class_weights = dict(zip([1,2,3,4,5,6,7,8,9,10,14,15,16], class_wt))
#class_weights[0] = 0
#class_weights[11] = 0
#class_weights[12] = 0
#class_weights[13] = 0
print(class_wt.sum())
print(class_weights)

print(np.bincount(Y_train))
print(np.bincount(Y_test))

class Logistic_Regression():

  # declaring learning rate & number of iterations (Hyperparametes)
  def __init__(self, learning_rate, no_of_iterations):
    self.learning_rate = learning_rate
    self.no_of_iterations = no_of_iterations

  # fit function to train the model with dataset
  def fit(self, X, Y):

    # number of data points in the dataset (number of rows)  -->  m
    # number of input features in the dataset (number of columns)  --> n
    self.m, self.n = X.shape

    #initiating weight & bias value
    self.w = np.zeros(self.n)
    self.b = 0
    self.X = X
    self.Y = Y
```

```
  # implementing Gradient Descent for Optimization
    for i in range(self.no_of_iterations):
      self.update_weights()

  def update_weights(self):
    # Y_hat formula (sigmoid function)
    Y_hat = 1 / (1 + np.exp( - (self.X.dot(self.w) + self.b ) ))
    # derivaties
    dw = (1/self.m)*np.dot(self.X.T, (Y_hat - self.Y))
    db = (1/self.m)*np.sum(Y_hat - self.Y)

    # updating the weights & bias using gradient descent
    self.w = self.w - self.learning_rate * dw
    self.b = self.b - self.learning_rate * db

  # Sigmoid Equation & Decision Boundary
  def predict(self, X):
    Y_pred = 1 / (1 + np.exp( - (X.dot(self.w) + self.b ) ))
    Y_pred = np.where( Y_pred > 0.5, 1, 0)
    return Y_pred
   logistic.fit(X_train, Y_train)
  X_test_prediction = logistic.predict(X_test)
   training_data_accuracy = accuracy_score( Y_test, X_test_prediction)
from sklearn.metrics import classification_report
from sklearn import metrics
print('Precision: %.3f' %metrics.recall_score(Y_test, X_test_prediction,
labels=[1,2,3,4,5,6,7,8,9,10,14,15,16], average='micro'))
print('Recall: %.3f' %metrics.precision_score(Y_test, X_test_prediction,
labels=[1,2,3,4,5,6,7,8,9,10,14,15,16], average='macro'))
print('F1_Score: %.3f' %metrics.f1_score(Y_test, X_test_prediction, average='weighted'))
 print(classification_report(Y_test,X_test_prediction))
```

```
#DecisionTreeClassifier

from sklearn.tree import DecisionTreeClassifier
DecisionTree = DecisionTreeClassifier(criterion="entropy",random_state=2,max_depth=5)
DecisionTree.fit(X_train,Y_train)
predicted_values = DecisionTree.predict(X_test)
x = metrics.accuracy_score(Y_test, predicted_values)
x = metrics.accuracy_score(Y_test, predicted_values)
print('Precision: %.3f' %metrics.recall_score(Y_test, predicted_values,
labels=[1,2,3,4,5,6,7,8,9,10,14,15,16], average='micro'))
print('Recall: %.3f' %metrics.precision_score(Y_test, predicted_values,
labels=[1,2,3,4,5,6,7,8,9,10,14,15,16], average='macro'))
print('F1_Score: %.3f' %metrics.f1_score(Y_test, predicted_values, average='weighted'))
print(classification_report(Y_test,predicted_values))

# Hyperaeter tuning on regularization parameter and kernal for SVM

c_list = [0.001, 0.01, 0.1, 1, 10, 100, 1000]
kernals = ['linear','rbf','poly','sigmoid']
pca_accuracy = {'linear':[], 'rbf':[], 'poly':[], 'sigmoid':[]}
#pca_f1_score = {'linear':[], 'rbf':[], 'poly':[], 'sigmoid':[]}
  for kernal in kernals:
  for cval in c_list:
    clf = SVC(max_iter=100000, kernel=kernal, C=cval)
    clf.fit(X_train_pca, Y_train)
    pca_accuracy[kernal].append(clf.score(X_train_pca, Y_train))
    yPred = clf.predict(X_train_pca)
    #pca_f1_score[kernal].append(f1_score(Y_test, yPred, average='weighted'))
    del clf
    del yPred

print('SVM Accuracies - PCA: ')
```

```
print('Linear kernal has maximum accuracy - '+ str(round(max(pca_accuracy['linear']),4)) + ' for
critical factor ' + str(c_list[pca_accuracy['linear'].index(max(pca_accuracy['linear']))]) )
print('\nRadial Basis Function Kernel SVM accuracy - '+ str(round(max(pca_accuracy['rbf']),4))
+ ' for critical factor ' + str(c_list[pca_accuracy['rbf'].index(max(pca_accuracy['rbf']))]) )
print('\nPolynomial Kernel SVM has maximum accuracy - ' +
str(round(max(pca_accuracy['poly']),4)) + ' for critical factor ' +
str(c_list[pca_accuracy['poly'].index(max(pca_accuracy['poly']))]))
print('\nSigmoid Kernel SVM has maximum accuracy - ' +
str(round(max(pca_accuracy['sigmoid']),4)) + ' for critical factor ' +
str(c_list[pca_accuracy['sigmoid'].index(max(pca_accuracy['sigmoid']))]) +'\n\n')

# Plot the Accuracy with C values
fig = plt.figure(1)
fig.patch.set_facecolor('white')
plt.xscale('log')
plt.title('Kernel SVM (Principle Component Analysis)')
plt.xlabel('Critical Factor')
plt.ylabel('Model Accuracy')
plt.plot(c_list, pca_accuracy['linear'], 'r', label = 'Linear')
plt.plot(c_list, pca_accuracy['rbf'], 'g', label = 'Radial Basis')
plt.plot(c_list, pca_accuracy['poly'], 'b', label = 'Polynomial')
plt.plot(c_list, pca_accuracy['sigmoid'], 'y', label = 'Sigmoid')
plt.legend(bbox_to_anchor = (0., 1.02, 1., .202), loc = 10, ncol=4, borderaxespad = 0)
fig.show()

#multi-layer perceptron model
from tensorflow.python.keras import models
from tensorflow.python.keras.layers import Dense
from tensorflow.python.keras.layers import Dropout
import tensorflow as tf
def mlp_model(layers, units, dropout_rate, input_shape, num_classes):
```

```
    """Creates an instance of a multi-layer perceptron model.
    # Arguments
        layers: int, number of `Dense` layers in the model.
        units: int, output dimension of the layers.
        dropout_rate: float, percentage of input to drop at Dropout layers.
        input_shape: tuple, shape of input to the model.
        num_classes: int, number of output classes.
    # Returns
        An MLP model instance.
    model = models.Sequential()
    model.add(Dropout(rate=dropout_rate, input_shape=input_shape))

    for _ in range(layers-1):
        model.add(Dense(units=units, activation='relu'))
        model.add(Dropout(rate=dropout_rate))

    model.add(Dense(units=num_classes, activation='softmax'))
    return model

def train_ngram_model(X_train,Y_train,
                      X_test,
                      Y_test,
                      learning_rate=1e-3,
                      epochs=1000,
                      batch_size=128,
                      layers=2,
                      units=64,
                      dropout_rate=0.2)
    model = mlp_model(layers=layers,
                      units=units,
                      dropout_rate=dropout_rate,
                      input_shape=X_train.shape[1:],
```

```
                num_classes=1)

    # Compile model with learning parameters.
    optimizer = tf.keras.optimizers.Adam(lr=learning_rate)
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])

    # Create callback for early stopping on validation loss. If the loss does
    # not decrease in two consecutive tries, stop training.
    callbacks = [tf.keras.callbacks.EarlyStopping(
        monitor='val_loss', patience=2)]

    # Train and validate model.
    history = model.fit(
            X_train,
            Y_train,
            epochs=epochs,
            callbacks=callbacks,

validation_data=(X_test, Y_test),
            verbose=2,  # Logs once per epoch.
            batch_size=batch_size)
    model.save('saved models\Cardiac.h5')
    return model, history
```

CHAPTER 6

DISCUSSSION AND RESULT

6.1 DISCUSSSION AND RESULT

Because predicting the type of arrhythmia is the goal of the research. Understanding the dataset and locating an effective machine learning algorithm are prerequisites for developing the system. The machine learning system should be capable of effectively learning, predicting, and categorising the type of arrhythmia.

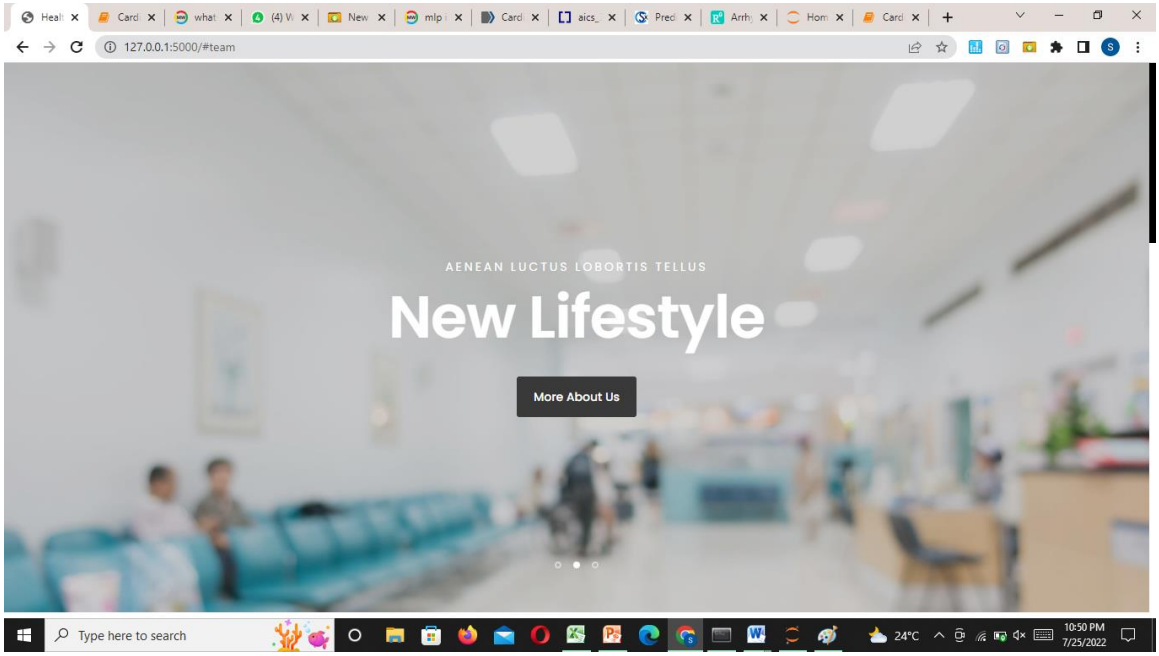
The system we developed for this project provides an efficiency of 86.48 percent and operates as expected. The Multiplayer Perceptron Model is the most appropriate for our purpose because it uses majority voting instead of any other classification or prediction techniques. The model could reach this accuracy score when 70% of the data was provided for training and the rest 30% for testing.

Algorithm	**Train Size (in %)**	**Test Size (in %)**	**Accuracy (in %)**
MLP Classifier	70	30	86.48
SVM	70	30	66.00
Naive_bayes	70	30	27.83
Decision Tree Classifier	70	30	64.00
K-Neighbors Classifier	70	30	48.19

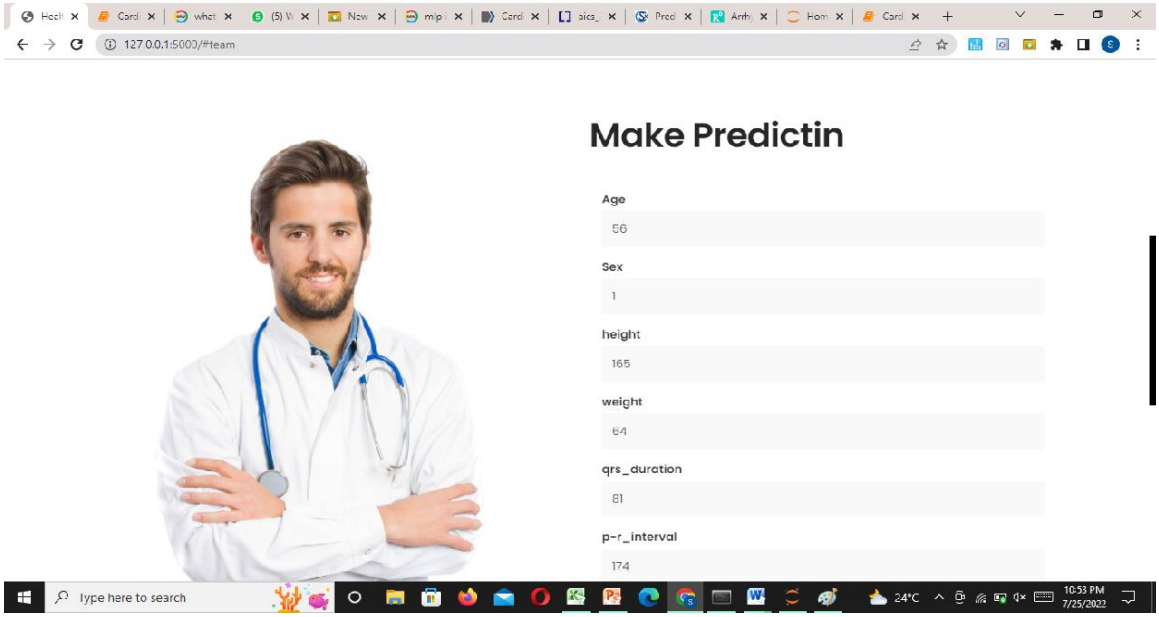
Table 6.1.1 Accuracy Score of each Algorithm.

We may state that in this case and K-Neighbors Classifier Algorithms are the least effective for this project. SVM performs more effectively than the decision tree classifier technique. The MLP Classifier is superior to all others and stands out.

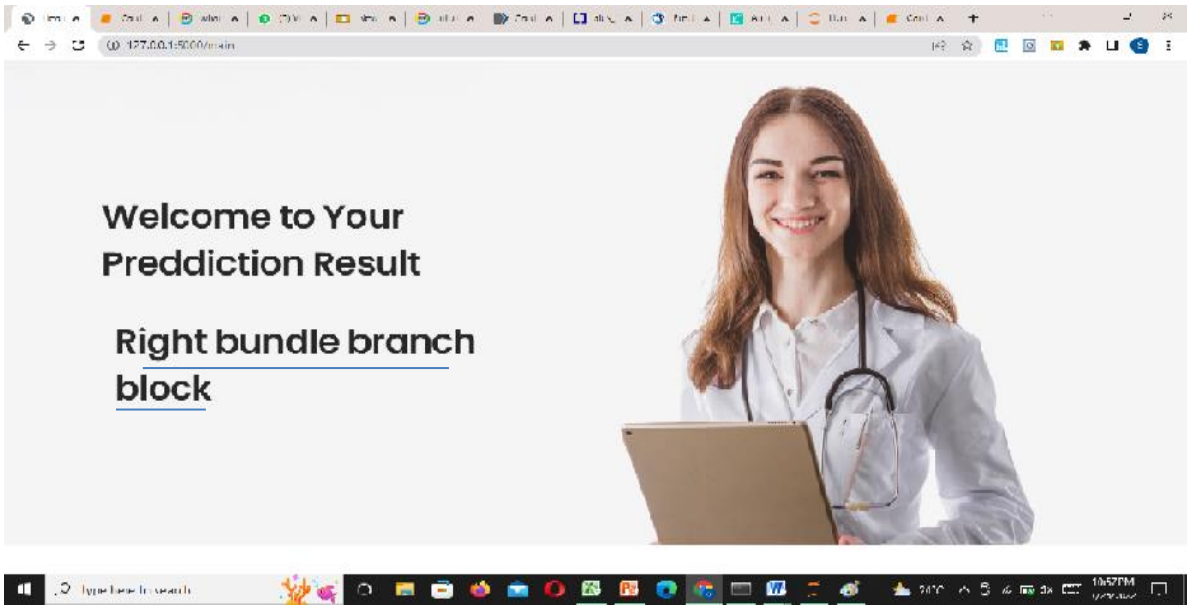
6.2 SCREENSHOTS



Screenshot 6.2.1 Home Page



Screenshot 6.2.2 Data Entry Page



Screenshot 6.2.3 Result Page

CHAPTERE 7
CONCLUSION AND FUTURE ENHANCEMENTS

7.1 CONCLUSION

In the present study, we propose a method for categorising arrhythmias using ECG data plus Multilayer Perceptron methods. To minimise the data's dimension, the most important characteristic is chosen using a correlation matrix and principal component analysis. In order to prevent conflicts brought on by the existence of binary values, the data is additionally standardised. The MLP approach is used to determine whether a disease is present or absent and to categorise the results through one of the sixteen categories that are provided. A combination of the feature selection, preprocessing, and classification algorithms has been created that shows promise for disease categorization. According to the classification results, the mlp approach works well for classifying the ECG dataset retrieved from the UCI repository. Other predictors are also used, and the findings demonstrate that the suggested method works better than other cutting-edge techniques used to categorise arrhythmia using a comparable dataset. The mlp approach has the potential to be improved to be used to additional illness datasets.

7.2 FUTURE ENHANCEMENTS

In this study, we have employed machine learning methods to estimate the severity of the cardiac condition using ECG data. Regarding the creation of this project, we employed 13 classes and roughly 500 data points, which was insufficient. In order to make accurate predictions in the future, we will need to take into account a larger number of datasets. To achieve this, we can employ deep learning models like RNN and CNN.

CHAPTER 8
REFERENCES

8.1 REFERENCES:

1. A real time ECG signal processing application for arrhythmia detection on portable devices - Scientific Figure on ResearchGate. Available from: https://www. researchgate.net/figure/ECG-intervals-and-segments_fig1_321455361, accessed: 05-2019

2. Acharya, U.R., Oh, S.L., Hagiwara, Y., Tan, J.H., Adam, M., Gertych, A., San Tan, R.: A deep convolutional neural network model to classify heartbeats. Computers in biology and medicine 89, 389–396 (2017)

3. Alexakis, C., Nyongesa, H., Saatchi, R., Harris, N., Davies, C., Emery, C., Ireland, R., Heller, S.: Feature extraction and classification of electrocardiogram (ecg) signals related to hypoglycaemia. In: Computers in Cardiology, 2003. pp. 537–540. IEEE (2003)

4. Chollet, F., et al.: Keras (2015)

5. Dastjerdi, A.E., Kachuee, M., Shabany, M.: Non-invasive blood pressure estimation using phonocardiogram. In: 2017 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 1–4. IEEE (2017)

6. Esmaili, A., Kachuee, M., Shabany, M.: Nonlinear cuffless blood pressure estimation of healthy subjects using pulse transit time and arrival time. IEEE Transactions on Instrumentation and Measurement 66(12), 3299–3308 (2017)

7.Fazeli, S.: ECG Heartbeat Categorization Dataset. https://www.kaggle.com/ shayanfazeli/heartbeat, accessed: 05-2019

8.Goldberger AL, Amaral LAN, G.L.H.J.I.P.M.R.M.J.M.G.P.C.K.S.H.: Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic

signals. IEEE Engineering in Medicine and Biology Magazine 101(23), 215–220 (2003)

9. Jadhav, S.M., Nalbalwar, S.L., Ghatol, A.A.: Modular neural network based arrhythmia classification system using ecg signal data. International Journal of Information Technology and Knowledge Management 4(1), 205–209 (2011)

10. Kachuee, M., Fazeli, S., Sarrafzadeh, M.: Ecg heartbeat classification: A deep transferable representation. In: 2018 IEEE International Conference on Healthcare Informatics (ICHI). pp. 443–444. IEEE (2018)

11. Kim, J., Shin, H.S., Shin, K., Lee, M.: Robust algorithm for arrhythmia classification in ecg using extreme learning machine. Biomedical engineering online 8(1), 31 (2009)

12. Martis, R.J., Acharya, U.R., Lim, C.M., Mandana, K., Ray, A.K., Chakraborty, C.: Application of higher order cumulant features for cardiac health diagnosis using ecg signals. International journal of neural systems 23(04), 1350014 (2013)

13. for the Advancement of Medical Instrumentation, A.,et al.: Testing and reporting performance results of cardiac rhythm and st segment measurement algorithms. ANSI/AAMI EC38 1998 (1998)

14. Moody, G.B., Mark, R.G.: The impact of the mit-bih arrhythmia database. IEEE Engineering in Medicine and Biology Magazine 20(3), 45–50 (2001)

15. Roopa, C., Harish, B.: A survey on various machine learning approaches for ecg analysis. International Journal of Computer Applications 163(9), 25–33 (2017)