PROFESSIONAL TRAINING REPORT

AT

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

By

NAME: THOTA SAHITHI
(Reg.No: 41111259)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119, TAMILNADU

October 2023

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC | 12B status by UGC |Approved by AICTE

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **THOTA SAHITHI (41111259)** who carried out the project entitled "**HEART DISEASE PREDICTION USING MACHINE LEARNING AND DATA ANALYTICS APPROCH"** under my supervision from JULY 2023 to OCTOBER 2023.

Internal Guide

DR.A.C. SANTHA SHEELA MAM

Head of the Department

Dr. L. Lakshmanan M.E., Ph. D

Submitted for Viva voice Examination held on ____________________

Internal Examiner **External Examiner**

DECLARATION

I THOTA SAHITHI hereby declare that the Project Report entitled HEART DISEASE PREDICTION USING MACHINE LEARNING AND DATA ANALYTICS one by me under the guidance of Dr.A.C. SANTHA SHEELA MAM and (Internal) at ____

(Company name and address) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 07.10.2023 T. SAHITHI

PLACE: CHENNAI **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D.**, **Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide DR.A.C. SANTHA SHEELA MAM or his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE

ORACLE

ORACLE
University

Certificate of Completion

Thota Sahithi
Has Successfully Completed

Machine Learning Cloud
An Oracle University Training

Damien Carey
Senior Vice President,
Oracle University

September 03, 2023
Date

ABSTRACT

In recent times, Machine Learning has played a significant role in the healthcare industry and amongst all of the major diseases, heart disease is one of the significant and most critical diseases to predict. There is a rapid increase in the number of cases each day. It has been observed that in every minute, 4 people between the age group of 30-50 get a stroke, so we are using machine learning algorithms to mitigate this problem. Kaggle used the heart disease dataset used for this project. This paper demonstrates the prediction of heart disease using multiple machine learning classification algorithms such as Naive Bayes, Random Forest, SVM etc., and compares their accuracy scores. Later on, Stacking Ensemble Learning Technique is used to boost our classification models' performance.

The World Health Organization (WHO) estimates that 17.7 million people worldwide die suddenly each year as a result of cardiovascular illnesses. The people may benefit from the ability to foresee the complexity of their health at an early stage thanks to the heart disease prediction system. A doctor's examination or a variety of medical tests, such as an ECG, a stress test, a heart MRI, etc., are the traditional methods for predicting heart disease. There is a vast quantity of hidden information in the health care data that is already available. Making wise decisions benefits from having access to this hidden information. For acceptable findings, computer-based data as well as advanced data mining techniques are used. Existing systems do a good job of accurately predicting the outcome, but more data attributes and the complexity of health parameters form the foundation for the development of novel strategies. In this proposed system we implement Heart Disease Prediction using Artificial Neural Network (ANN). The project "Heart Disease Prediction using Artificial Neural Network (ANN)" aims to develop a predictive model for the early detection of heart disease using ANNs. ANNs are powerful machine learning algorithms that can learn patterns and relationships in data, making them an ideal choice for predicting complex medical conditions like heart disease. The proposed system is implemented using Cleveland Heart Disease dataset available on UCI machine learning repository / Kaggle.

This data is then pre-processed and used to train an ANN model using supervised learning techniques. The model is optimized to achieve high accuracy in predicting the likelihood of heart disease in patients. The trained model is then tested on a separate dataset to evaluate its performance and accuracy. Finally, the project aims to develop a user-friendly interface that allows doctors and healthcare professionals to input patient data and receive the predicted results. The proposed model has the potential to improve the accuracy and speed of heart disease diagnosis, enabling early intervention and better patient outcomes. This project can contribute to the development of AI-powered healthcare solutions and can have a significant impact on public health.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
ANN	Artificial Neural Network
KNN	K-Nearest Neighbour
SVM	Support Vector Machine
RF	Random Forest
ML	Machine Learning
DT	Decision Tree
GA	Genetic Algorithm
ECG	Electro -Cardi-graph

CHAPTER 1

1. INTRODUCTION

It is difficult to identify heart disease because of several contributory risk factors such as diabetes, high blood pressure, high cholesterol, abnormal pulse rate and many other factors. Among various life-threatening diseases, heart disease has garnered a great deal of attention in medical research.

The diagnosis of heart disease is a challenging task, which can offer automated prediction about the heart condition of patient so that further treatment can be made effective. The diagnosis of heart disease is usually based on signs, symptoms of the patient. The severity of the disease is classified based on various methods like K-Nearest Neighbor Algorithm (KNN), Decision Trees (DT), Genetic Algorithm (GA), and Navie Bayes (NB). The Nature of heart disease is complex and hence, the disease must be handled carefully. Not doing so may affect the heart or cause premature death.

In data science research, Machine Learning is a strategy for gaining knowledge from the prior research experience. The conventional methods encounter several issues to solve all the problems pointed by various researchers. The purpose of data assessment is to evaluate genuine models, which necessitates using a dependable, robust, and trustworthy framework, including such Machine Learning methods.

The Machine Learning method likes to work with immediate input from the training sample after learning the foundational patterns in the data. The possible outcome of the whole training phase is an automatic framework. The proposed framework is best for static and dynamic datasets. A prediction of data is an outcome of the training and testing stage. A testing stage utilizes some information collection, datasets for training, dataset for testing, and some specific type of ML models, i.e., classification or clustering models.

CHAPTER 2

2. AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM:

The main aim of this project is to predict whether a person is having heart disease or not using machine learning algorithms.
The supervised learning algorithms used in this project such as Naïve Bayes, Random Forest, Support vector machine, Artificial Neural Network, k Nearest Neighbour, Decision Tree.

2.2 SCOPE:

Here the scope of the project is that integration of clinical decision support with computer-based patient records could reduce medical errors, enhance patient safety, decrease unwanted practice variation, and improve patient outcome. This suggestion is promising as data modelling and analysis tools, e.g., data mining, have the potential to generate a knowledge-rich environment which can help to significantly improve the quality of clinical decisions.

2.3 PROPOSED ARCHITECTURE:

In this system we are implementing effective heart attack prediction system using Naïve Bayes algorithm. We can give the input as in CSV file or manual entry to the system. After taking input the algorithms apply on that input that is Naïve Bayes. After accessing data set the operation is performed and effective heart attack level is produced. The proposed system will add some more parameters significant to heart attack with their weight, age and the priority levels are by consulting expertise doctors and the medical experts. The heart attack prediction system designed to help the identify different risk level of heart attack like normal, low or high and also giving the prescription details with related to the predicted result.

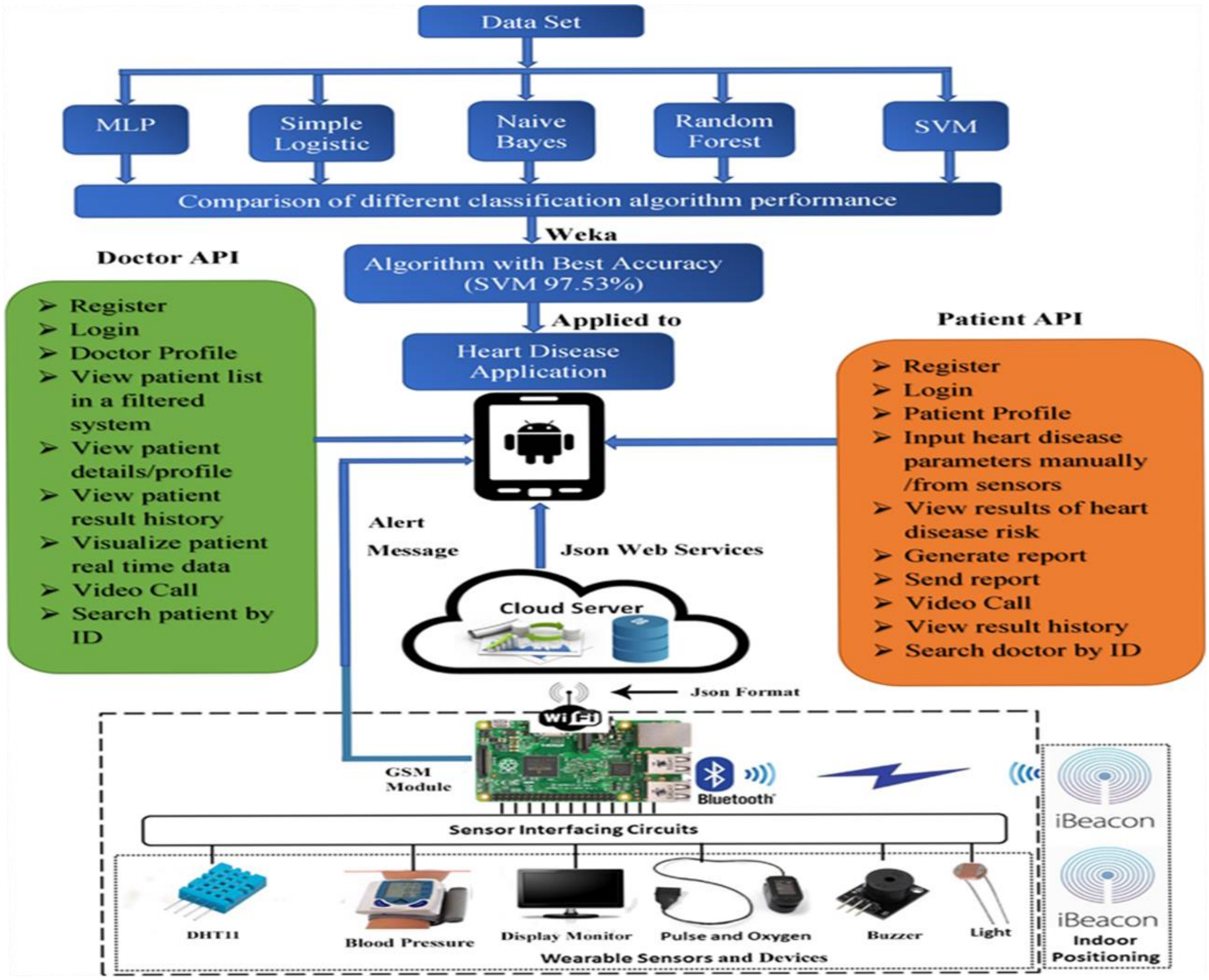


Figure 1

CHAPTER 3

3. EXPERIMENTAL MATERIALS AND METHODOLOGY, ALGORITHM AND SYSTEM IMPLEMENTATION

3.1 MATERIALS:

3.1.1 *SOFTWARE REQUIREMENTS*:

- Operating system: Windows 10.

- Coding Language: Python 3.8
- Web Framework: Flask
- Operating System: Any OS with clients to access the internet
- Network: Wi-Fi Internet or cellular Network
- Visio Studio: Create and design Data Flow and Context Diagram
- GitHub: Versioning Control
- Google Chrome: Medium to find reference to do system testing display and run shin yapp.

3.1.2 *HARDWARE REQUIREMENTS:*

System: Pentium i3 Processor.

- Hard Disk: 500 GB.
- Monitor: 15'' LED
- Input Devices: Keyboard, Mouse
- Ram: 4 GB

3.2 METHODOLOGY:

3.2.1 *DESCRIPITION OF THE DATASET:*

The dataset used for this research purpose was the Public Health Dataset and it is dating from 1988 and consists of four databases: Cleveland, Hungary, Switzerland, and Long Beach V. It contains 76 attributes, including the predicted attribute, but all published experiments refer to using a subset of 14 of them. The "target" field refers to the presence of heart disease in the patient. It is integer-valued 0 = no disease and 1 = disease. The first four rows and all the dataset features are shown in Table 1 without any preprocessing. Now the attributes which are used research purpose is described as follows and for what they are used or resemble

(i)Age—age of patient in years, sex— (1 = male; 0 = female).

(ii)Cp—chest pain type.

(iii)Treetops—resting blood pressure (in mm Hg on admission to the hospital). The normal range is 120/80 (if you have a normal blood pressure reading, it is fine, but if it is a little higher than it Should be, you should try to lower it. Make healthy changes to your lifestyle).

(iv)Chol—serum cholesterol shows the number of triglycerides present. Triglycerides are another lipid that can be measured in the blood. It should be less than 170 mg/dL (may differ in different Labs).

(v)Fbs—fasting blood sugar larger than 120 mg/dl (1 true). Less than 100 mg/dL (5.6 mmol/L) is normal, and 100 to 125 mg/dL (5.6 to 6.9 mmol/L) is considered prediabetes.

(vi)Restecg—resting electrocardiographic results.

(vii)Thalach—maximum heart rate achieved. The maximum heart rate is 220 minus your age.

(viii)Exan g—exercise-induced angina (1 yes). Angina is a type of chest pain caused by reduced blood flow to the heart. Angina is a symptom of coronary artery disease.

(ix)Old peak—ST depression induced by exercise relative to rest.

(x)Slope—the slope of the peak exercise ST segment.

(xi)Ca—number of major vessels (0–3) coloured by fluoroscopy.

(xii)Thal—no explanation provided, but probably thalassemia (3 normal; 6 fixed defects; 7 reversible defects).

(xiii)Target (T)—no disease = 0 and disease = 1, (angiographic disease status).

3.2.2 *PREPROCESSING OF THE DATASET:*

The dataset does not have any null values. But many outliers needed to be handled properly, and also the dataset is not properly distributed. Two approaches were used. One without outliers and feature selection process and directly applying the data to the machine learning algorithms, and the results which were achieved were not promising. But after using the normal distribution of dataset for overcoming the overfitting problem and then applying Isolation Forest for the outlier's detection, the results achieved are quite promising. Various plotting techniques were used for checking the skewness of the data, outlier detection, and the distribution of the data. All these preprocessing techniques play an important role when passing the data for classification or prediction purposes.

3.2.2.1 *checking distribution of the dataset:*

The distribution of the data plays an important role when the prediction or classification of a problem is to be done. We see that the heart disease occurred 54.46% of the time in the dataset, whilst 45.54% was the no heart disease. So, we need to balance the dataset or otherwise it might get overfit. This will help the model to find a pattern in the dataset that contributes to heart disease and which does not as show in Figure **2**.

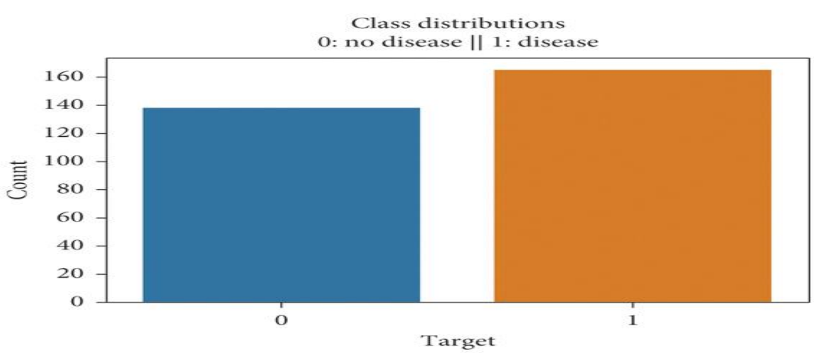
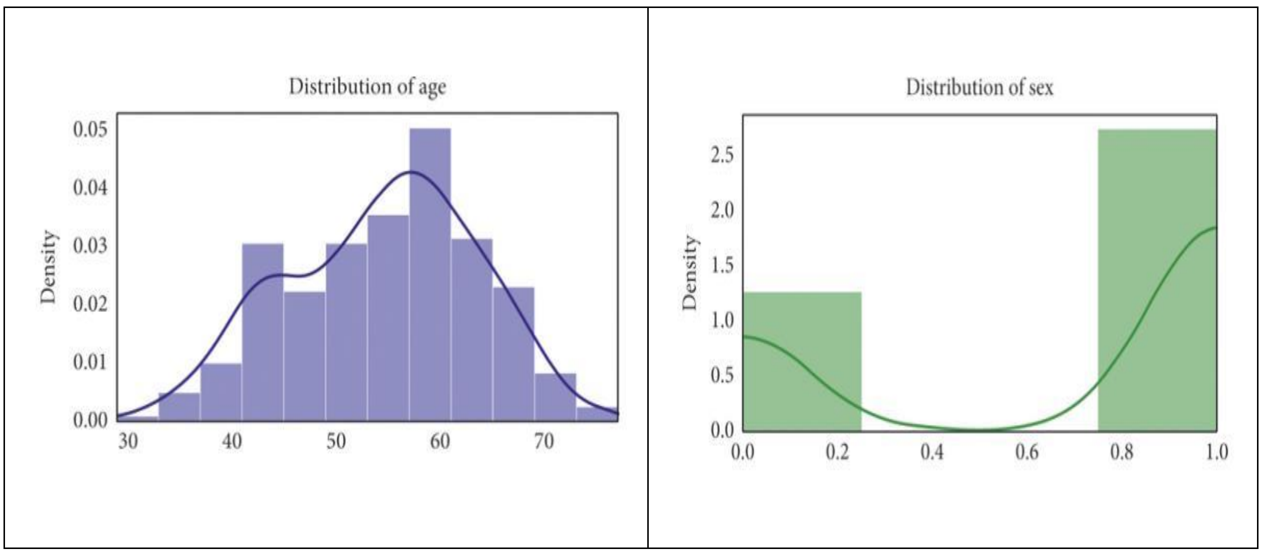


Figure 2

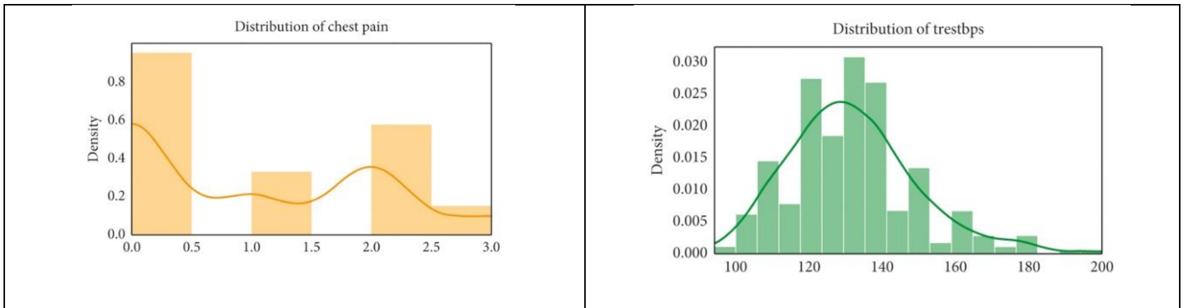
3.2.2.2. Checking the Skewness of the Data:

For checking the attribute values and determining the skewness of the data (the asymmetry of a distribution), many distribution plots are plotted so that some interpretation of the data can be seen. Different plots are shown, so an overview of the data could be analysed. The distribution of age and sex, the distribution of chest pain and trest bps, the distribution of cholesterol and fasting blood, the distribution of ecg resting electrode and thalach, the distribution of exang and old peak, the distribution of slope and ca, and the distribution of that and target all are analysed and the conclusion is drawn as shown in Figures 3 and 4



(a) Figure 3 (b)

By analysing the distribution plots, it is visible that that and fasting blood sugar is not uniformly distributed and they needed to be handled; otherwise, it will result in overfitting or underfitting of the data.

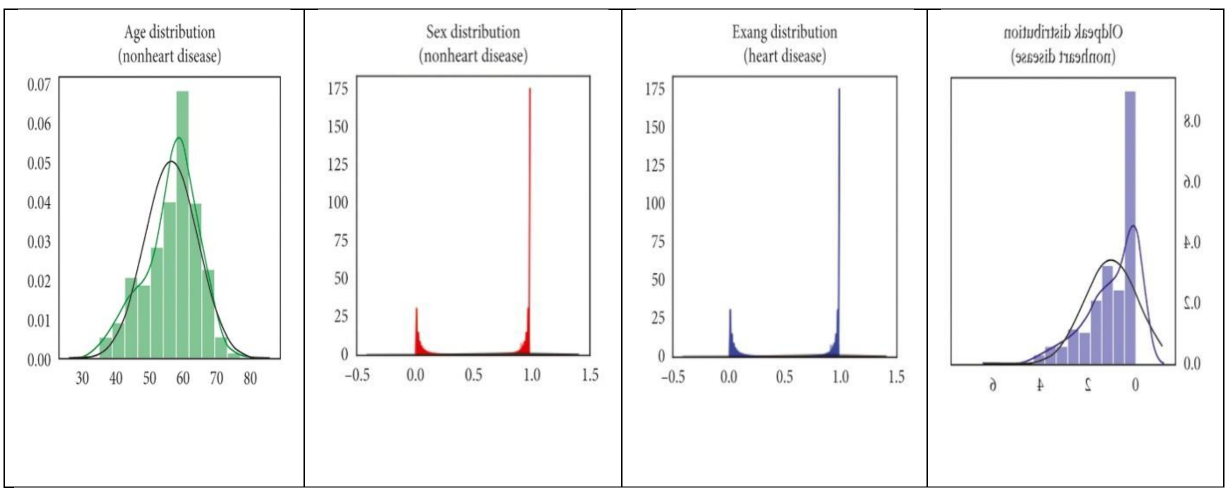


(a) (b)

Figure 4

3.2.2.3 *Checking Stats of the Normal Distribution of Data:*

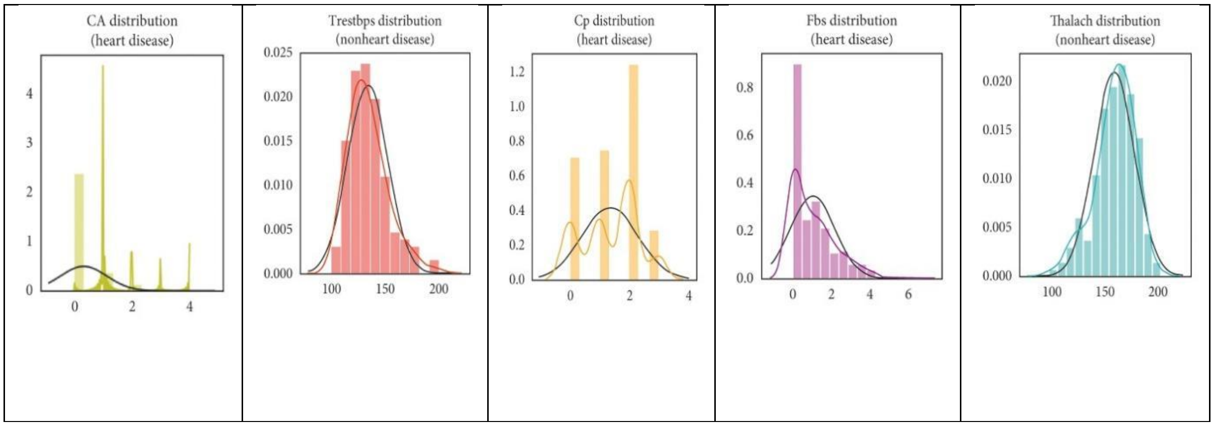
Checking the features which are important for heart disease and not important for heart disease is shown in Figures 5 and 6, respectively. Here the important factors show a different variation which means it is important.



(a) (b) (c) (d)

Figure 5

The conclusion which can be drawn from these statistical figures is that we can see a Gaussian distribution which is important for heart disease and no Gaussian distribution which is playing that much important role in heart disease.



(a) (b) (c) (d) (e)

Figure 6

3.3 ALGORITHM USED:

3.3.1 ARTIFICIAL INTELLEGENCE:

These are used to model/simulate the distribution, functions or mappings among variables as modules of a dynamic system associated with a learning rule or a learning algorithm. The modules here simulate neurons in nervous system and hence ANN collectively refers to the neuron simulators and their synapsthese modules in different layers.

Neural Network is built by stacking together multiple neurons in layers to produce a final output. First layer is the input layer and the last is the output layer. All the layers in between are called hidden layers. Each neuron has an activation function. Some of the popular Activation functions are Sigmoid, ReLU, tanh etc. The parameters of the network are the weights and biases of easuch that the predicted outcome is the same as the ground truth. Back the network parameters.

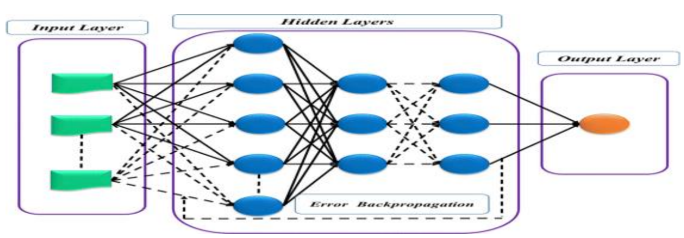


Figure 7

3.3.2 *SUPPORT VECTOR MACHINE(SVM):*

Support vector machines exist in different forms, linear and nonclassifier. What is usual in this context, two different datasets are involved with SVM, training and a test set. In the ideal situation the classes are linearly separable. In such situation a line can be found, which splits the two classes perfectly. However not only one line splits the dataset perfectly, but a whole bunch of lines do. From these lines the best is selected as the "separating line".

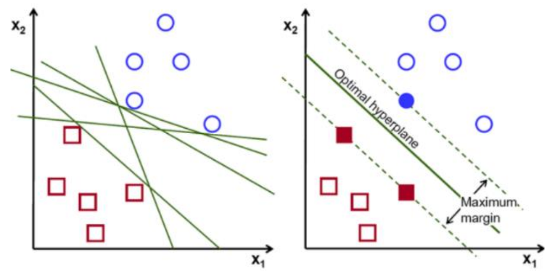


Figure 8

A SVM can make some errors to avoid over Support vector machines classifiers are applied in many applications. They are very popular in recent research. This popularity is due to the good overall empirical performance. Comparing the naïve Bayes and the SVM classifier, the SVM has been applied the most.

3.3.3 *NAVIE BAYES*:

It is a classifier based on supervised learning that solves regression and classification tasks. It is based on Bayesian statistics. The basic concept of the least-squares underpins a binary, i.e., two-class and multi-class classification. The technique is most intuitive for binary classification and also for variable input data The Naive Bayes framework is straightforward and well-suited to large data sets. As compared to other Machine Learning methods, it produces a higher precision. A Bayes theorem calculates the probability of an incident happening based on the possibility of a previous incident.

3.3.*4 K- NEAREST NEIGHBOUR:*

It Is a supervised Machine Learning methodology for the solution of regression as well as classification complications. It is simple to set up and acknowledge, but it does have the disadvantage of being noticeably slower as the volume of data in use expands. As a result of its high level of accuracy, the KNN method can directly compete with precise existing models. If people need high precision and yet do not need a human-readable method, the KNN algorithm is perfect. We can mainly evaluate forecasts based on distance metrics. The best algorithm for the given data set is complicated and depends on several samples, features, and dimensions. Datasets are used in Machine Learning as they need an intelligent analysis. For a search location, the degree of neighbours is demanded.

3.3.5 RANDOM FOREST:

It is a supervised technique-based ML classifier that designs and build models using Decision Tree Classifiers. Trees, in general, learn abnormal behaviour and overfit the trained model with minor differences and bias. It is used to decrease the variance among features in a given dataset. It also helps in classification same training dataset and testing datasets and emerges at the cost of a modest bias increase. Various companies such as banking and online use this method to estimate objectives. An ensemble methodology is used to classify, predict the future, and perform specific activities. If researchers try to classify something, the Random Forest will generate a class that almost all trees have chosen. Random Forests give the effects of K-fold cross-validation.

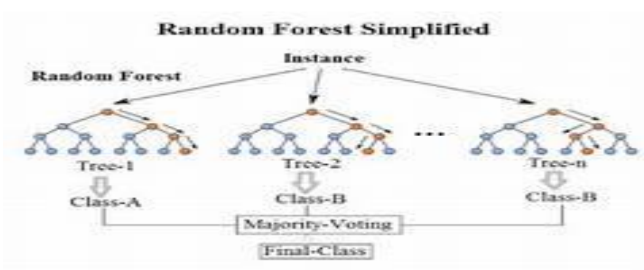


Figure 9

3.3.6 DECISION TREE:

It mainly represents a process incorporating a tree-like growth model of pronouncements and their promising outcomes, including occurrence implications, cost elements, and performance characteristics. It is yet another strategy for demonstrating a preliminary optimization approach. A Decision Tree Classifier is among the most frequent types of classification models. It is just a flowchart-like structure with a network that represents a test on a function. A Decision Tree Classifier, as stated earlier, splits the classifications into sub-sets (i.e., root, left child, right child). In the sample group chosen, it is the most widely used approach.

Figure 10

Iterative Dichotomiser-3 (ID-3) designed by the researcher [31] is among the most prevalent Decision Tree Classifier methods, as it produces all possible intelligent Decision Tree Classifiers and chooses the best. When particularly in comparison to the learning algorithm, the learning time is shorter. The number of items and characteristics in the training data set determines the precise complexity of Decision Tree Classifiers. It is not based on any assumptions about probability distributions. To great accuracy, a Decision Tree Classifier algorithm can manage high-dimensional statistics. Information Gain & Gain Ratio are the essential attribute selection measures (ASM).

SOURCE CODE:

1.*IMPORTING DATASET:*

This dataset contains patient data concerning heart disease diagnosis that was collected at several locations around the world. There are 76 attributes, including age, sex, resting blood pressure, cholesterol levels, echocardiogram data, exercise habits, and many others. To data, all published studies using this data focus on a subset of 14 attributes - so we will do the same. More specifically, we will use the data collected at the Cleveland Clinic Foundation.

To import the necessary data, we will use pandas' built in read_csv() function. Let's get started

Importing libraries:

- Numpy: to work with arrays
- Pandas: to work with csv files and dataframes
- Seaborne:to visualize data
- StandardScalar:To scale all the features
- Train_test_split:to split the dataset into training and testing dataset
- Matplotlib:to create charts using pyplot, define Parameters using rcParams

In [1]:

```
import sys
import pandas as pd
import numpy as np
import sklearn
import matplotlib
import keras

print('Python: {}'.format(sys.version))
print('Pandas: {}'.format(pd.__version__))
print('Numpy: {}'.format(np.__version__))
print('Sklearn: {}'.format(sklearn.__version__))
print('Matplotlib: {}'.format(matplotlib.__version__))
print('Keras: {}'.format(keras.__version__))
```

```
Using TensorFlow backend.

Python: 3.6.6 |Anaconda, Inc.| (default, Oct  9 2018, 12:34:16)
[GCC 7.3.0]
Pandas: 0.23.4
Numpy: 1.16.2
Sklearn: 0.20.3
Matplotlib: 3.0.3
```

Import data:

In [3]:
```python
# read the csv
cleveland = pd.read_csv('../input/heart.csv')
```

In [4]:
```python
# print the shape of the DataFrame, so we can see how many examples we have
print( 'Shape of DataFrame: {}'.format(cleveland.shape))
print (cleveland.loc[1])
```

```
Shape of DataFrame: (303, 14)
age          37.0
sex           1.0
cp            2.0
trestbps    130.0
chol        250.0
fbs           0.0
restecg       1.0
thalach     187.0
exang         0.0
oldpeak       3.5
slope         0.0
ca            0.0
thal          2.0
```

In [5]:
```python
# print the last twenty or so data points
cleveland.loc[280:]
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
280	42	1	0	136	315	0	1	125	1	1.8	1	0	1	0
281	52	1	0	128	204	1	1	156	1	1.0	1	0	0	0
282	59	1	2	126	218	1	1	134	0	2.2	1	1	1	0
283	40	1	0	152	223	0	1	181	0	0.0	2	0	3	0
284	61	1	0	140	207	0	0	138	1	1.9	2	1	3	0
285	46	1	0	140	311	0	1	120	1	1.8	1	2	3	0
286	59	1	3	134	204	0	1	162	0	0.8	2	2	2	0
287	57	1	1	154	232	0	0	164	0	0.0	2	1	2	0
288	57	1	0	110	335	0	1	143	1	3.0	1	1	3	0
289	55	0	0	128	205	0	2	130	1	2.0	1	1	3	0
290	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
291	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
292	58	0	0	170	225	1	0	146	1	2.8	1	2	1	0
293	67	1	2	152	212	0	0	150	0	0.8	1	0	3	0
294	44	1	0	120	169	0	1	144	1	2.8	0	0	1	0
295	63	1	0	140	187	0	0	144	1	4.0	2	2	3	0
296	63	0	0	124	197	0	1	136	1	0.0	1	0	2	0
297	59	1	0	164	176	1	0	90	0	1.0	1	2	1	0

In [6]:
```python
# remove missing data (indicated with a "?")
data = cleveland[~cleveland.isin(['?'])]
data.loc[280:]
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
280	42	1	0	136	315	0	1	125	1	1.8	1	0	1	0
281	52	1	0	128	204	1	1	156	1	1.0	1	0	0	0
282	59	1	2	126	218	1	1	134	0	2.2	1	1	1	0
283	40	1	0	152	223	0	1	181	0	0.0	2	0	3	0
284	61	1	0	140	207	0	0	138	1	1.9	2	1	3	0
285	46	1	0	140	311	0	1	120	1	1.8	1	2	3	0
286	59	1	3	134	204	0	1	162	0	0.8	2	2	2	0
287	57	1	1	154	232	0	0	164	0	0.0	2	1	2	0
288	57	1	0	110	335	0	1	143	1	3.0	1	1	3	0
289	55	0	0	128	205	0	2	130	1	2.0	1	1	3	0
290	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
291	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
292	58	0	0	170	225	1	0	146	1	2.8	1	2	1	0
293	67	1	2	152	212	0	0	150	0	0.8	1	0	3	0
294	44	1	0	120	169	0	1	144	1	2.8	0	0	1	0
295	63	1	0	140	187	0	0	144	1	4.0	2	2	3	0
296	63	0	0	124	197	0	1	136	1	0.0	1	0	2	0

In [7]:

```
# drop rows with NaN values from DataFrame
data = data.dropna(axis=0)
data.loc[280:]
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
280	42	1	0	136	315	0	1	125	1	1.8	1	0	1	0
281	52	1	0	128	204	1	1	156	1	1.0	1	0	0	0
282	59	1	2	126	218	1	1	134	0	2.2	1	1	1	0
283	40	1	0	152	223	0	1	181	0	0.0	2	0	3	0
284	61	1	0	140	207	0	0	138	1	1.9	2	1	3	0
285	46	1	0	140	311	0	1	120	1	1.8	1	2	3	0
286	59	1	3	134	204	0	1	162	0	0.8	2	2	2	0
287	57	1	1	154	232	0	0	164	0	0.0	2	1	2	0
288	57	1	0	110	335	0	1	143	1	3.0	1	1	3	0
289	55	0	0	128	205	0	2	130	1	2.0	1	1	3	0
290	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
291	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
292	58	0	0	170	225	1	0	146	1	2.8	1	2	1	0
293	67	1	2	152	212	0	0	150	0	0.8	1	0	3	0
294	44	1	0	120	169	0	1	144	1	2.8	0	0	1	0
295	63	1	0	140	187	0	0	144	1	4.0	2	2	3	0

In [8]:

```
# print the shape and data type of the dataframe
print(data.shape)
print(data.dtypes)
```

```
(303, 14)
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

In [9]:

```
# transform data to numeric to enable further analysis
data = data.apply(pd.to_numeric)
data.dtypes
```

Out[9]:

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

In [10]:

```
# print data characteristics, usings pandas built-in describe() function
data.describe()
```

Out[10]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000

In [11]:

```
# plot histograms for each variable
data.hist(figsize = (12, 12))
plt.show()
```

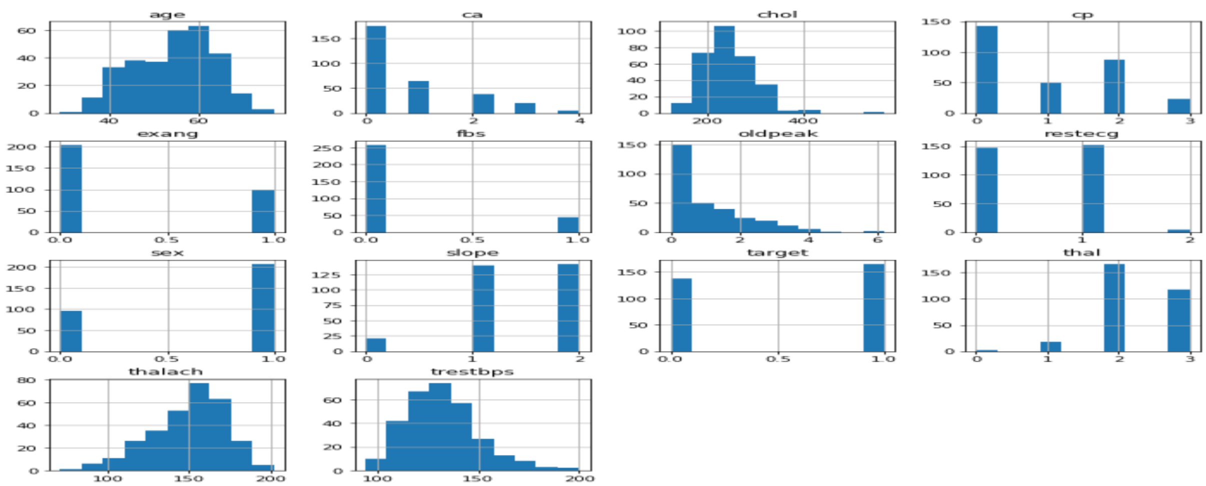


Figure 11

In [12]:

```
pd.crosstab(data.age,data.target).plot(kind="bar",figsize=(20,6))
plt.title('Heart Disease Frequency for Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

In [13]:

```
plt.figure(figsize=(10,10))
sns.heatmap(data.corr(),annot=True,fmt='.1f')
plt.show()
```

```
In [14]:
age_unique=sorted(data.age.unique())
age_thalach_values=data.groupby('age')['thalach'].count().values
mean_thalach=[]
for i,age in enumerate(age_unique):
    mean_thalach.append(sum(data[data['age']==age].thalach)/age_thalach_values[i])

plt.figure(figsize=(10,5))
sns.pointplot(x=age_unique,y=mean_thalach,color='red',alpha=0.8)
plt.xlabel('Age',fontsize = 15,color='blue')
plt.xticks(rotation=45)
plt.ylabel('Thalach',fontsize = 15,color='blue')
plt.title('Age vs Thalach',fontsize = 15,color='blue')
plt.grid()
plt.show()
```

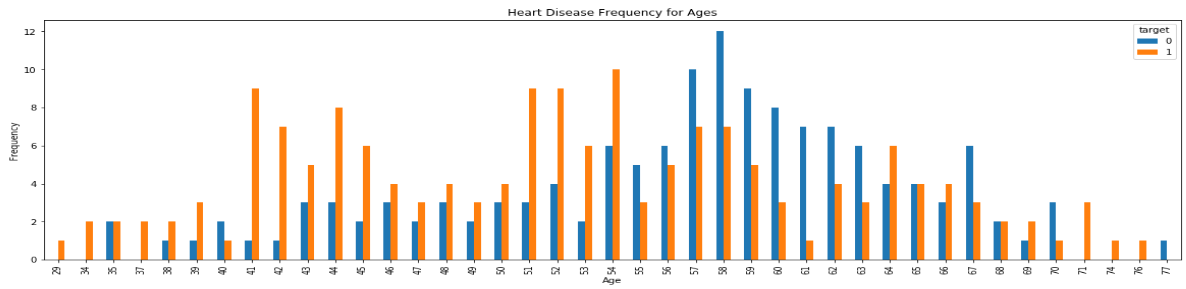
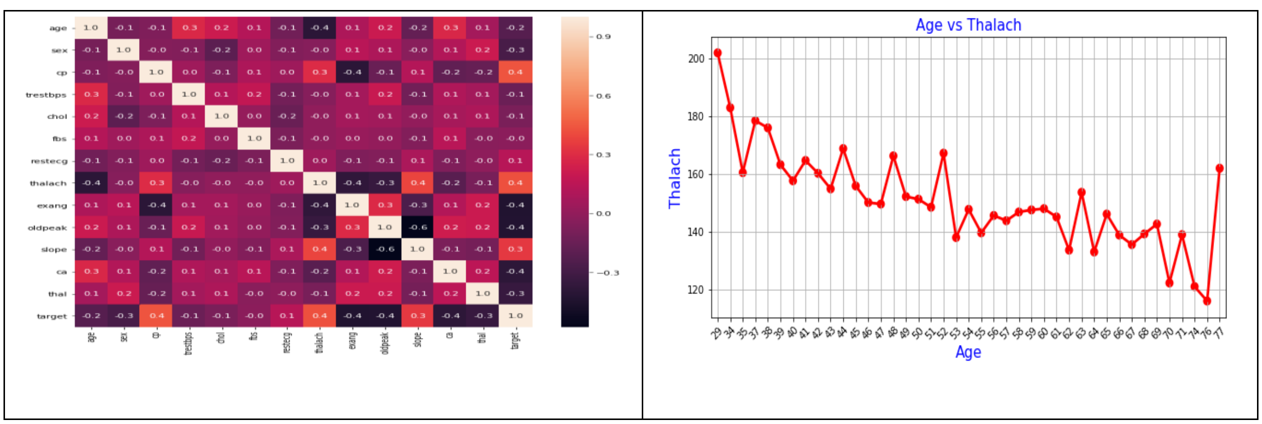


Figure 12



(a) (b)

Figure 13

2.CREATE TRAINING AND TEST DATASETS:

Now that we have pre-processed the data appropriately, we can split it into training and testing's datasets. We will use Sklearn's train_test_split() function to generate a training dataset (80 percent of the total data) and testing dataset (20 percent of the total data).

```
In [15]:
X = np.array(data.drop(['target'], 1))
y = np.array(data['target'])
```

```
In [16]:
X[0]
```

```
Out[16]:
array([ 63. ,    1. ,    3. , 145. , 233. ,    1. ,    0. , 150. ,    0. ,
          2.3,    0. ,    0. ,    1. ])
```

```
In [17]:
mean = X.mean(axis=0)
X -= mean
std = X.std(axis=0)
X /= std
```

```
In [18]:
X[0]
```

```
Out[18]:
array([ 0.9521966 ,  0.68100522,  1.97312292,  0.76395577, -0.25633371,
        2.394438  , -1.00583187,  0.01544279, -0.69663055,  1.08733806,
       -2.27457861, -0.71442887, -2.14887271])
```

```
In [19]:
# create X and Y datasets for training
from sklearn import model_selection

X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, stratify=y, random_sta
te=42, test_size = 0.2)
```

```
In [20]:
# convert the data to categorical labels
from keras.utils.np_utils import to_categorical

Y_train = to_categorical(y_train, num_classes=None)
Y_test = to_categorical(y_test, num_classes=None)
print (Y_train.shape)
print (Y_train[:10])
```

3.BUILDING AND TRAINING THE NEURAL NETWORK:

Now that we have our data fully processed and split into training and testing datasets, we can begin building a neural network to solve this classification problem. Using keras, we will define a simple neural network with one hidden layer. Since this is a categorical classification problem, we will use a SoftMax activation function in the final layer of our network and a categorical_crossentropy loss during phase.

```
In [22]:
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
from keras.layers import Dropout
from keras import regularizers

# define a function to build the keras model
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(16, input_dim=13, kernel_initializer='normal', kernel_regularizer=regularize
rs.l2(0.001), activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(8, kernel_initializer='normal', kernel_regularizer=regularizers.l2(0.001), a
ctivation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(2, activation='softmax'))

    # compile model
    adam = Adam(lr=0.001)
    model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
    return model
```

```
WARNING:tensorflow:From /opt/conda/lib/python3.6/site-packages/tensorflow/python/framework/op
_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and w
ill be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /opt/conda/lib/python3.6/site-packages/keras/backend/tensorflow_backe
nd.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated
and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 16)                224
_________________________________________________________________
dropout_1 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 136
_________________________________________________________________
dropout_2 (Dropout)          (None, 8)                 0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 18
=================================================================
```

```
In [23]:
# fit the model to the training data
history=model.fit(X_train, Y_train, validation_data=(X_test, Y_test),epochs=50, batch_size=10)
```

```
WARNING:tensorflow:From /opt/conda/lib/python3.6/site-packages/tensorflow/python/ops/math_op
s.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed i
n a future version.
Instructions for updating:
Use tf.cast instead.
Train on 242 samples, validate on 61 samples
Epoch 1/50
242/242 [==============================] - 3s 14ms/step - loss: 0.6749 - acc: 0.6694 - val_lo
ss: 0.6578 - val_acc: 0.7869
Epoch 2/50
242/242 [==============================] - 0s 316us/step - loss: 0.6345 - acc: 0.7521 - val_l
oss: 0.6204 - val_acc: 0.7705
Epoch 3/50
242/242 [==============================] - 0s 322us/step - loss: 0.5859 - acc: 0.7934 - val_l
oss: 0.5785 - val_acc: 0.7705
Epoch 4/50
242/242 [==============================] - 0s 321us/step - loss: 0.5331 - acc: 0.8099 - val_l
oss: 0.5398 - val_acc: 0.7869
Epoch 5/50
```

```
242/242 [==============================] - 0s 318us/step - loss: 0.5023 - acc: 0.8223 - val_l
oss: 0.5068 - val_acc: 0.7869
Epoch 6/50
242/242 [==============================] - 0s 325us/step - loss: 0.4604 - acc: 0.8306 - val_l
oss: 0.4796 - val_acc: 0.7869
Epoch 7/50
242/242 [==============================] - 0s 337us/step - loss: 0.4352 - acc: 0.8471 - val_l
oss: 0.4566 - val_acc: 0.8033
Epoch 8/50
242/242 [==============================] - 0s 338us/step - loss: 0.4203 - acc: 0.8471 - val_l
oss: 0.4404 - val_acc: 0.7869
Epoch 9/50
242/242 [==============================] - 0s 325us/step - loss: 0.4134 - acc: 0.8264 - val_l
oss: 0.4322 - val_acc: 0.7869
Epoch 10/50
242/242 [==============================] - 0s 322us/step - loss: 0.3854 - acc: 0.8512 - val_l
oss: 0.4218 - val_acc: 0.7869
Epoch 11/50
242/242 [==============================] - 0s 318us/step - loss: 0.3906 - acc: 0.8430 - val_l
oss: 0.4166 - val_acc: 0.7869
Epoch 12/50
242/242 [==============================] - 0s 323us/step - loss: 0.3795 - acc: 0.8471 - val_l
oss: 0.4144 - val_acc: 0.7869
Epoch 13/50
```

```
242/242 [==============================] - 0s 316us/step - loss: 0.3595 - acc: 0.8430 - val_l
oss: 0.4126 - val_acc: 0.7869
Epoch 14/50
242/242 [==============================] - 0s 316us/step - loss: 0.3558 - acc: 0.8595 - val_l
oss: 0.4186 - val_acc: 0.7869
Epoch 15/50
242/242 [==============================] - 0s 313us/step - loss: 0.3359 - acc: 0.8678 - val_l
oss: 0.4198 - val_acc: 0.7869
Epoch 16/50
242/242 [==============================] - 0s 312us/step - loss: 0.3645 - acc: 0.8512 - val_l
oss: 0.4130 - val_acc: 0.7869
Epoch 17/50
242/242 [==============================] - 0s 374us/step - loss: 0.3489 - acc: 0.8678 - val_l
oss: 0.4114 - val_acc: 0.7869
Epoch 18/50
242/242 [==============================] - 0s 323us/step - loss: 0.3320 - acc: 0.8595 - val_l
oss: 0.4114 - val_acc: 0.7869
Epoch 19/50
242/242 [==============================] - 0s 328us/step - loss: 0.3863 - acc: 0.8595 - val_l
oss: 0.4097 - val_acc: 0.7869
Epoch 20/50
242/242 [==============================] - 0s 318us/step - loss: 0.3418 - acc: 0.8471 - val_l
oss: 0.4060 - val_acc: 0.7869
Epoch 21/50
```

```
242/242 [==============================] - 0s 331us/step - loss: 0.3530 - acc: 0.8595 - val_l
oss: 0.4060 - val_acc: 0.7869
Epoch 22/50
242/242 [==============================] - 0s 312us/step - loss: 0.3572 - acc: 0.8512 - val_l
oss: 0.4040 - val_acc: 0.8033
Epoch 23/50
242/242 [==============================] - 0s 315us/step - loss: 0.3627 - acc: 0.8595 - val_l
oss: 0.4075 - val_acc: 0.8033
Epoch 24/50
242/242 [==============================] - 0s 309us/step - loss: 0.3678 - acc: 0.8388 - val_l
oss: 0.4097 - val_acc: 0.8033
Epoch 25/50
242/242 [==============================] - 0s 312us/step - loss: 0.3267 - acc: 0.8595 - val_l
oss: 0.4146 - val_acc: 0.8033
Epoch 26/50
242/242 [==============================] - 0s 315us/step - loss: 0.3622 - acc: 0.8512 - val_l
oss: 0.4128 - val_acc: 0.8033
Epoch 27/50
242/242 [==============================] - 0s 313us/step - loss: 0.3698 - acc: 0.8554 - val_l
oss: 0.4041 - val_acc: 0.8033
Epoch 28/50
242/242 [==============================] - 0s 311us/step - loss: 0.3336 - acc: 0.8760 - val_l
oss: 0.4043 - val_acc: 0.8033
Epoch 29/50
```

```
242/242 [==============================] - 0s 316us/step - loss: 0.3281 - acc: 0.8554 - val_l
oss: 0.4060 - val_acc: 0.8033
Epoch 30/50
242/242 [==============================] - 0s 315us/step - loss: 0.3621 - acc: 0.8760 - val_l
oss: 0.4106 - val_acc: 0.8033
Epoch 31/50
242/242 [==============================] - 0s 319us/step - loss: 0.3425 - acc: 0.8678 - val_l
oss: 0.4098 - val_acc: 0.8033
Epoch 32/50
242/242 [==============================] - 0s 321us/step - loss: 0.3339 - acc: 0.8636 - val_l
oss: 0.4133 - val_acc: 0.8033
Epoch 33/50
242/242 [==============================] - 0s 315us/step - loss: 0.3308 - acc: 0.8678 - val_l
oss: 0.4141 - val_acc: 0.8033
Epoch 34/50
242/242 [==============================] - 0s 340us/step - loss: 0.3182 - acc: 0.8884 - val_l
oss: 0.4113 - val_acc: 0.8033
Epoch 35/50
242/242 [==============================] - 0s 314us/step - loss: 0.3327 - acc: 0.8678 - val_l
oss: 0.4099 - val_acc: 0.8033
Epoch 36/50
242/242 [==============================] - 0s 321us/step - loss: 0.3240 - acc: 0.8760 - val_l
oss: 0.4080 - val_acc: 0.8033
Epoch 37/50
```

```
242/242 [==============================] - 0s 316us/step - loss: 0.3407 - acc: 0.8843 - val_l
oss: 0.4085 - val_acc: 0.8033
Epoch 38/50
242/242 [==============================] - 0s 325us/step - loss: 0.3440 - acc: 0.8554 - val_l
oss: 0.4037 - val_acc: 0.8033
Epoch 39/50
242/242 [==============================] - 0s 319us/step - loss: 0.3265 - acc: 0.8554 - val_l
oss: 0.4037 - val_acc: 0.8033
Epoch 40/50
242/242 [==============================] - 0s 315us/step - loss: 0.3354 - acc: 0.8802 - val_l
oss: 0.4042 - val_acc: 0.8033
Epoch 41/50
242/242 [==============================] - 0s 331us/step - loss: 0.3392 - acc: 0.8636 - val_l
oss: 0.4065 - val_acc: 0.8033
Epoch 42/50
242/242 [==============================] - 0s 342us/step - loss: 0.3099 - acc: 0.8926 - val_l
oss: 0.4051 - val_acc: 0.8033
Epoch 43/50
242/242 [==============================] - 0s 323us/step - loss: 0.3339 - acc: 0.8760 - val_l
oss: 0.4073 - val_acc: 0.8033
Epoch 44/50
242/242 [==============================] - 0s 342us/step - loss: 0.3324 - acc: 0.8802 - val_l
oss: 0.4079 - val_acc: 0.8033
Epoch 45/50
```

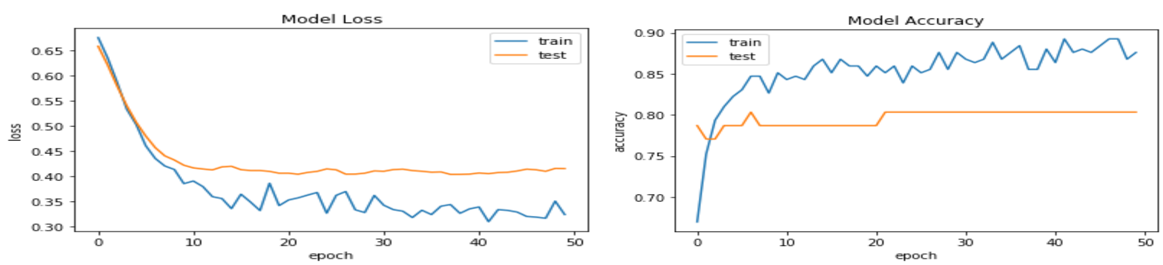
```
242/242 [==============================] - 0s 327us/step - loss: 0.3291 - acc: 0.8760 - val_l
oss: 0.4104 - val_acc: 0.8033
Epoch 46/50
242/242 [==============================] - 0s 327us/step - loss: 0.3204 - acc: 0.8843 - val_l
oss: 0.4141 - val_acc: 0.8033
Epoch 47/50
242/242 [==============================] - 0s 313us/step - loss: 0.3190 - acc: 0.8926 - val_l
oss: 0.4128 - val_acc: 0.8033
Epoch 48/50
242/242 [==============================] - 0s 313us/step - loss: 0.3168 - acc: 0.8926 - val_l
oss: 0.4098 - val_acc: 0.8033
Epoch 49/50
242/242 [==============================] - 0s 321us/step - loss: 0.3508 - acc: 0.8678 - val_l
oss: 0.4156 - val_acc: 0.8033
Epoch 50/50
242/242 [==============================] - 0s 324us/step - loss: 0.3240 - acc: 0.8760 - val_l
oss: 0.4153 - val_acc: 0.8033
```

In [24]:

```
import matplotlib.pyplot as plt
%matplotlib inline
# Model accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()
```

In [25]:

```
# Model Losss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()
```



(a) (b)

Figure 14

4. IMPROVING RESULTS- A BINARY CLASSIFICATION PROBLEM:

Although we achieved promising results, we still have a fairly large error. This could be because it is very difficult to distinguish between the different severity levels of heart disease (classes 1 - 4). Let's simplify the problem by converting the data to a binary classification problem - heart disease or no heart diseases.

In [26]:

```
# convert into binary classification problem - heart disease or no heart disease
Y_train_binary = y_train.copy()
Y_test_binary = y_test.copy()

Y_train_binary[Y_train_binary > 0] = 1
Y_test_binary[Y_test_binary > 0] = 1

print(Y_train_binary[:20])
```

```
[1 0 0 0 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 1]
```

In [27]:

```
# define a new keras model for binary classification
def create_binary_model():
    # create model
    model = Sequential()
    model.add(Dense(16, input_dim=13, kernel_initializer='normal',  kernel_regularizer=regulariz
ers.l2(0.001),activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(8, kernel_initializer='normal',  kernel_regularizer=regularizers.l2(0.001),a
ctivation='relu'))
    model.add(Dropout(0.25))
```

```
    # Compile model
    adam = Adam(lr=0.001)
    model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
    return model

binary_model = create_binary_model()

print(binary_model.summary())
```

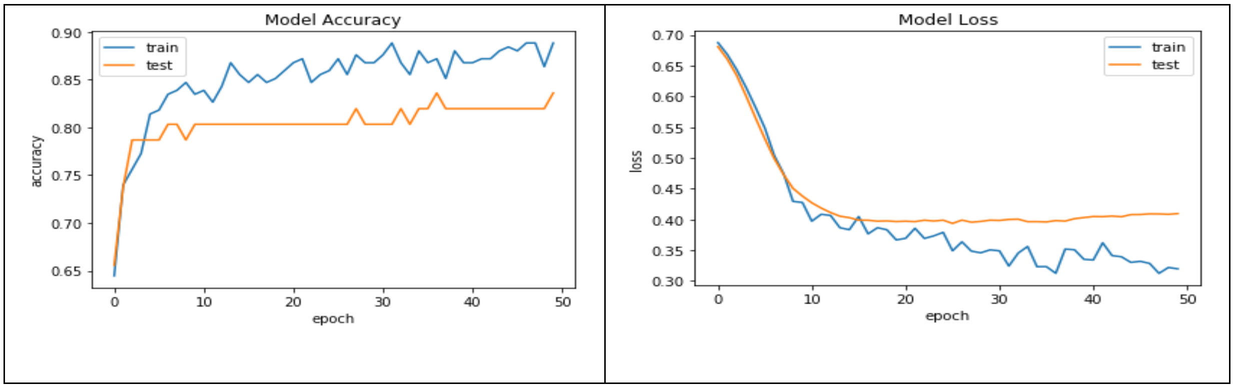
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 16)                224
_________________________________________________________________
dropout_3 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_5 (Dense)              (None, 8)                 136
_________________________________________________________________
dropout_4 (Dropout)          (None, 8)                 0
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 9
=================================================================
Total params: 369
```

In [29]:

```
import matplotlib.pyplot as plt
%matplotlib inline
# Model accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()
```

In [30]:

```
# Model Losss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()
```



(a) *(b)*

Figure 15

5.RESULTS AND METRICS:

The accuracy results we have been seeing are for the training data, but what about the testing dataset? If our models cannot generalize to data that wasn't used to train them, they won't provide any utility.

Let's test the performance of both our categorical model and binary model. To do this, we will make predictions on the training dataset and calculate performance metrics using Sklearn.

In [31]:
```
# generate classification report using predictions for categorical model
from sklearn.metrics import classification_report, accuracy_score

categorical_pred = np.argmax(model.predict(X_test), axis=1)

print('Results for Categorical Model')
print(accuracy_score(y_test, categorical_pred))
print(classification_report(y_test, categorical_pred))
```

```
Results for Categorical Model
0.8032786885245902
              precision    recall  f1-score   support

           0       0.86      0.68      0.76        28
           1       0.77      0.91      0.83        33

   micro avg       0.80      0.80      0.80        61
   macro avg       0.82      0.79      0.80        61
weighted avg       0.81      0.80      0.80        61
```

```
In [32]:
# generate classification report using predictions for binary model
from sklearn.metrics import classification_report, accuracy_score
# generate classification report using predictions for binary model
binary_pred = np.round(binary_model.predict(X_test)).astype(int)

print('Results for Binary Model')
print(accuracy_score(Y_test_binary, binary_pred))
print(classification_report(Y_test_binary, binary_pred))
```

```
Results for Binary Model
0.8360655737704918
              precision    recall  f1-score   support

           0       0.88      0.75      0.81        28
           1       0.81      0.91      0.86        33

   micro avg       0.84      0.84      0.84        61
   macro avg       0.84      0.83      0.83        61
weighted avg       0.84      0.84      0.83        61
```

CHAPTER 4

4.RESULTS AND DISCUSSION, PERFORMANCE ANALIYSIS

4.1 RESULTS:

The final prediction rate (heart disease prediction) results for all the Machine Learning classifier techniques
are presented. The SVM method shows 83.25%, Decision Tree Classifier 83.89 %, KNN 86.45, Random Forest 88.35, Logistic Regression 84.22% and Naive Bayes 84.69% prediction score. The experimental result demonstrates that the Random Forest classifier technique has a better prediction rate for detecting heart disease than other models.

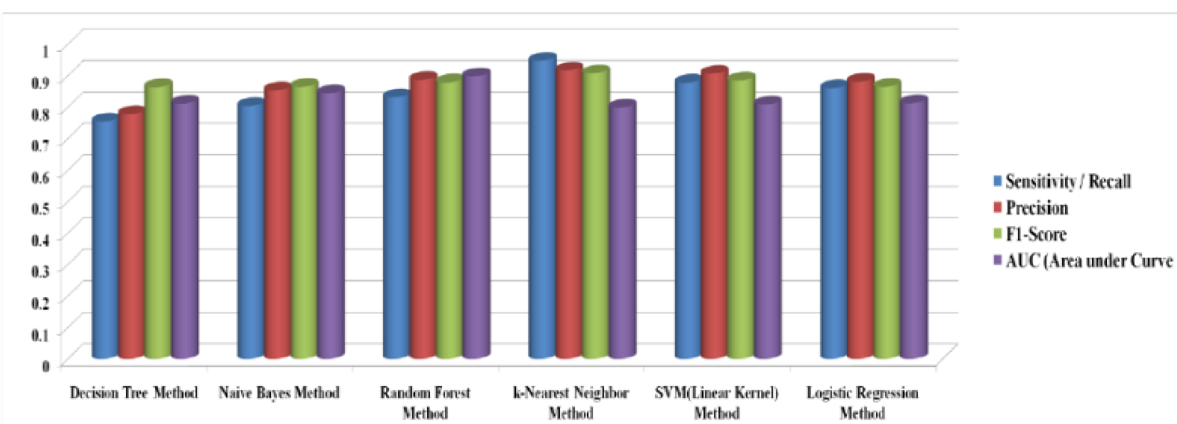


Figure 16

4.2 ANALYSIS:

Age("age") Analysis

Here we will be checking the 10 ages and their counts.

```
plt.figure(figsize=(25,12))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=data.age.value_counts()[:10].index,y=data.age.value_counts()[:10].values)
plt.tight_layout()
```

Output:

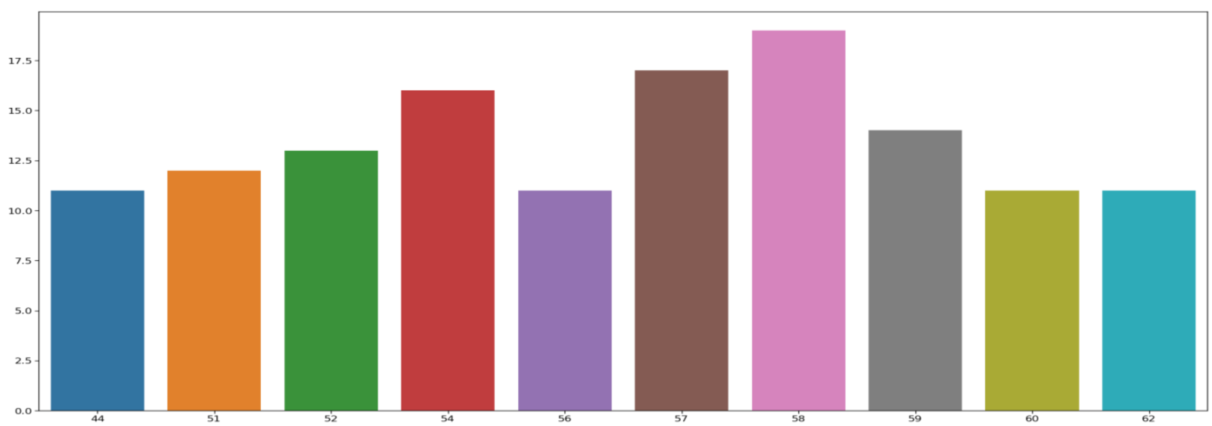


Figure 16

Sex("sex") Feature Analysis

```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(data['sex'])
plt.tight_layout()
```

Output:

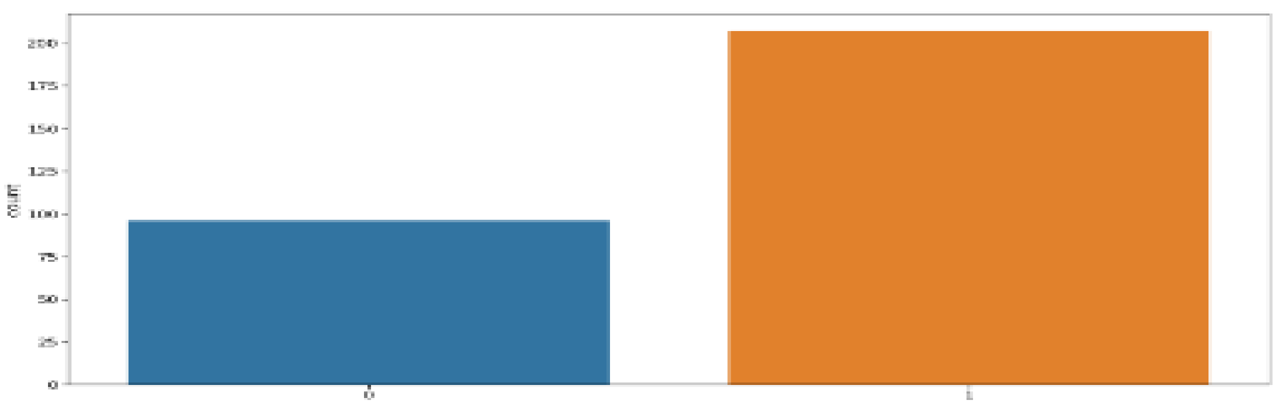


Figure 17

Chest Pain Type("cp") Analysis

```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(data['cp'])
plt.tight_layout()
```

Output:

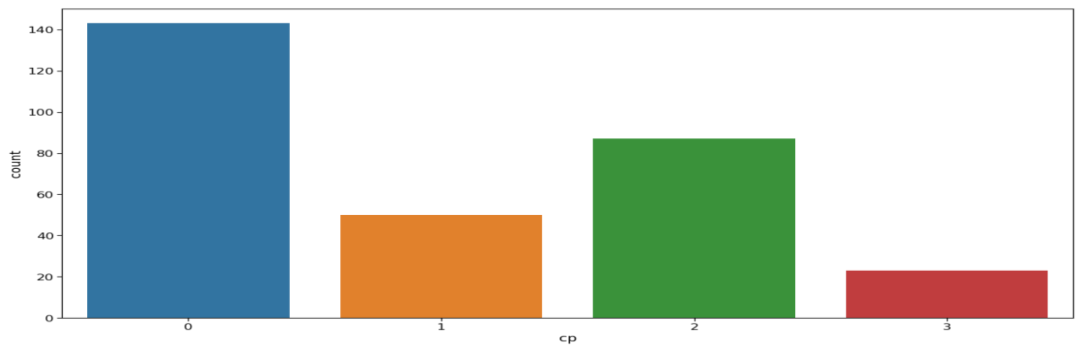


Figure 18

CHAPTER 5

5. SUMMARY AND CONCLUSION

5.1 SUMMARY:

The proposed system is GUI-based, user-friendly, scalable, reliable and an expandable system. The proposed working model can also help in reducing treatment costs by providing Initial diagnostics in time. The model can also serve the purpose of training tool for medical students and will be a soft diagnostic tool available for physician and cardiologist. General physicians can utilize this tool for initial diagnosis of cardio-patients. There are many possible improvements that could be explored to improve the scalability and accuracy of this prediction system. As we have developed a generalized system, in future we can use this system for the analysis of different data sets. The performance of the health's diagnosis can be improved significantly by handling numerous class labels in the prediction process, and it can be another positive direction of research.

In DM warehouse, generally, the dimensionality of the heart database is high, so identification and selection of significant attributes for better diagnosis of heart disease are very challenging tasks for future research.

5.2 CONCLUSION:

This project provides the deep insight into machine learning techniques for classification of heart diseases. The role of classifier is crucial in healthcare industry so that the results can be used for predicting the treatment which can be provided to patients. The existing techniques are studied and compared for finding the efficient and accurate systems. Machine learning techniques significantly improves accuracy of cardiovascular risk prediction through which patients can be identified during an early stage of disease and can be benefitted by preventive treatment. It can be concluded that there is a huge scope or machine Learning algorithms in predicting cardiovascular diseases or heart related diseases. Each of the above-mentioned algorithms have performed extremely well in some cases but poorly in some other cases

REFERENCES

1. K. Vembandasamyp, R. R. Sasipriyap and E. Deepap, "Heart Diseases Detection Using Naive Bayes Algorithm", *IJISET-International J. Innov. Sci. Eng. Technol*, vol. 2, no. 9, 2015, [online] Available: www.ijiset.com.

Show in Context Google Scholar

2. D. Shah, S. Patel, Santosh and K. Bharti, "Heart Disease Prediction using Machine Learning Techniques", vol. 1, pp. 345, 2020.

Show in Context CrossRef Google Scholar

3. S. Das, R. Sharma, M. K. Gourisaria, S. S. Rautaray, and M. Pandey, "Heart disease detection using core machine learning and deep learning techniques: A comparative study," International Journal on Emerging Technologies, vol. 11, no. 3, pp. 531–538, 2020.

4. M. T., D. Mukherji, N. Padalia, and A. Naidu, "A heart disease prediction model using SVM-decision trees-logistic regression (SDL)," International Journal of Computer Applications, vol. 68, no. 16, pp. 11–15, Apr. 2013, doi: 10.5120/11662-7250.

5. M. M. A. Rahhal, Y. Bazi, H. Alhichri, N. Alajlan, F. Melgani, and R. R. Yager, "Deep learning approach for active classification of electrocardiogram signals," *Information Sciences*, vol. 345, pp. 340–354, 2016.

View at: Publisher Site | Google Scholar

6. M.-S. Yang and Y. Nataliani, "A feature-reduction fuzzy clustering algorithm based on feature-weighted entropy," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 2, pp. 817–835, 2018.

View at: Publisher Site | Google Scholar

*******THANK YOU******