Visualizing and Predicting Heart Diseases with an Interactive Dash Board

Team ID : PNT2022TMID14101

A PROJECT REPORT

Submitted By

CHARULATHA R	**622119202008**
MUKUNTHAN M	**622119202032**
ARIVUSELVAN M	**622119202004**
MANORANJANI S	**622119202027**

TABLE OF THE CONTENT

CHAPTER 1

INTRODUCTION:

12.1 : PROJECT OVERVIEW

The terms "heart disease" and "cardiovascular disease" are frequently used interchangeably. Heart disease is a general term that covers a wide range of heart related medical conditions. The irregular health state that directly affects the heart and all of its components is characterized by these medical conditions. In order to forecast cardiac disease, this study discusses various data mining, big data, and machine learning techniques. Building an important model for the medical system to forecast heart disease or cardiovascular illness requires the use of data mining and machine learning. Our application helps the user in finding out if they have heart disease or not. They can find out by entering details such as their heart rate, cholesterol, blood pressure etc. A dashboard is also attached along with the results for better understanding where they can compare their blood pressure and similar metrics with other users. This project focuses on Random Forest Classifier. The accuracy of our project is 87% for which is better than most other systems in terms of achieving accuracy quickly.

12.2 : PURPOSE

This project's goal is to determine, depending on the patient's medical characteristics—such as gender, age, chest pain, fasting blood sugar level, etc.—whether they are likely to be diagnosed with any cardiovascular heart illnesses. The leading cause of death in the developed world is heart disease. Heart disease cases are rising quickly every day, thus it's crucial and worrisome to predict any potential illnesses in advance. This diagnosis is a challenging task that requires accuracy and efficiency. Therefore, there needs to be work done to help prevent the risks of having a heart attack or stroke. It is the main factor in adult deaths. By using a person's medical history, our initiative can identify

those who are most likely to be diagnosed with a cardiac condition. It can assist in identifying disease with less medical tests and effective therapies, so that patients can be treated appropriately. It can identify anyone who is experiencing any heart disease symptoms, such as chest pain or high blood pressure. Around the world, machine learning is applied in many different fields. There is no exception in the healthcare sector. Machine learning may be crucial in determining whether locomotor disorders, heart illnesses, and other conditions are present or absent. If foreseen well in advance, such information can offer valuable insights to doctors, who can then customize their diagnosis and course of care for each patient.

The EHDPS predicts the likelihood of patients getting heart disease. It enables significant knowledge, eg, relationships between medical factors related to heart disease and patterns, to be established. We have employed the multilayer perceptron neural network with back propagation as the training algorithm

.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

A quiet significant amount of works related to the diagnosis of heart disease using Machine Learning algorithms have been made. An efficient heart disease prediction has been made by using various algorithms some of them include Logistic Regression, KNN, Random Forest Classifier etc. It can be seen in results that each algorithm has its strength to register the defined objectives. The model incorporating IHDPS had the ability to calculate the decision boundary using the previous and new model of machine learning and deep learning. It facilitated the important and the most basic factors/knowledge such as family history connected with any heart disease. But the accuracy that was obtained in such IHDPS model was far more less than the new upcoming model such as detecting coronary heart disease using artificial neural network and other algorithms of machine and deep learning.

2.2 REFERENCES

[1] Ali, Liaqat, et al, "An optimized stacked support vector machines based expert system for the effective prediction of heart failure." IEEE Access 7 (2019): 54007-54014. www.ijcrt.org © 2020 IJCRT | Volume 8, Issue 8 August 2020 | ISSN: 2320-2882 IJCRT2008170 International Journal of Creative Research Thoughts (IJCRT) www.ijcrt.org 1606

[2] Mohan, Senthilkumar, Chandrasegar Thirumalai, and Gautam Srivastava, "Effective heart disease prediction using hybrid machine learning techniques." IEEE Access 7 (2019): 81542-81554.

[3] Purushottam, Kanak Saxena and Richa Sharma, "Efficient heart disease prediction system." Procedia Computer Science 85 (2016): 962-969.

[4] Singh, Yeshvendra K., Nikhil Sinha, and Sanjay K. Singh, "Heart Disease Prediction System Using Random Forest", International Conference on Advances in Computing and Data Sciences. Springer, Singapore, 2016.

[5] Santhana Krishnan. J, Geetha S., "Prediction of Heart Disease Using Machine Learning Algorithms", 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)

[6] Bo Jin ,Chao Che, Zhen Liu, Shulong Zhang, Xiaomeng Yin, And Xiaopeng Wei, "Predicting the Risk of Heart Failure With EHR Sequential Data Modeling" ,IEEE Access 2018.

[7]Aakash Chauhan , Aditya Jain , Purushottam Sharma , Vikas Deep, "Heart Disease Prediction using Evolutionary Rule Learning", "International Conference on "Computational Intelligence and Communication Technology" (CICT 2018).

[8] Boshra Bahrami, Mirsaeid Hosseini Shirvani, "Prediction and Diagnosis of Heart Disease by Data Mining Techniques", Journal of Multidisciplinary Engineering Science and Technology (JMEST) ISSN: 3159-0040 Vol. 2 Issue 2, February–2015.

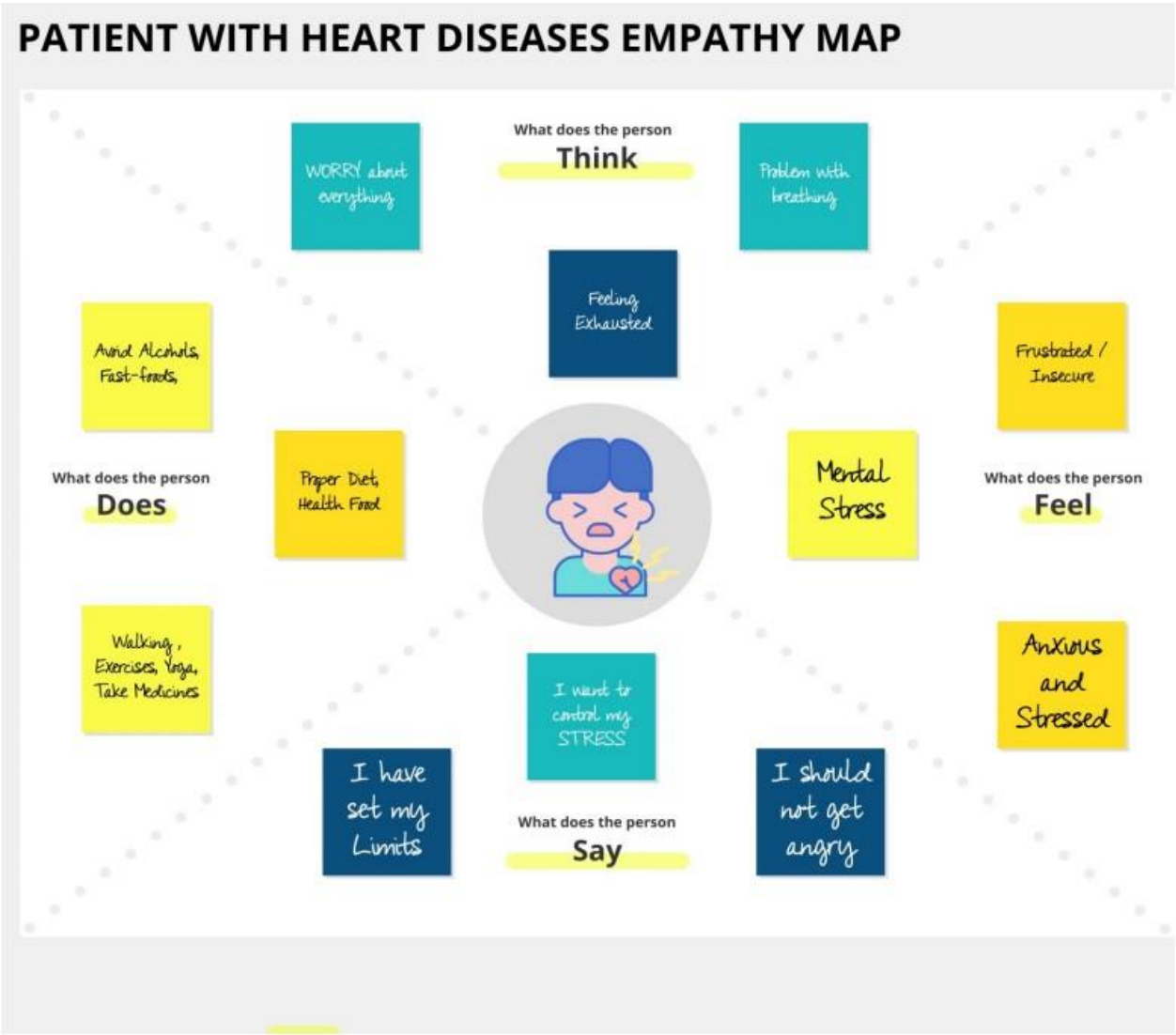
[9] M.Satish, D Sridhar, "Prediction of Heart Disease in Data Mining Technique", International Journal of Computer Trends & Technology (IJCTT), 2015.

[10] Lokanath Sarangi, Mihir Narayan Mohanty, Srikanta Pattnaik, "An Intelligent Decision Support System for Cardiac Disease Detection", IJCTA, International Press 2015.

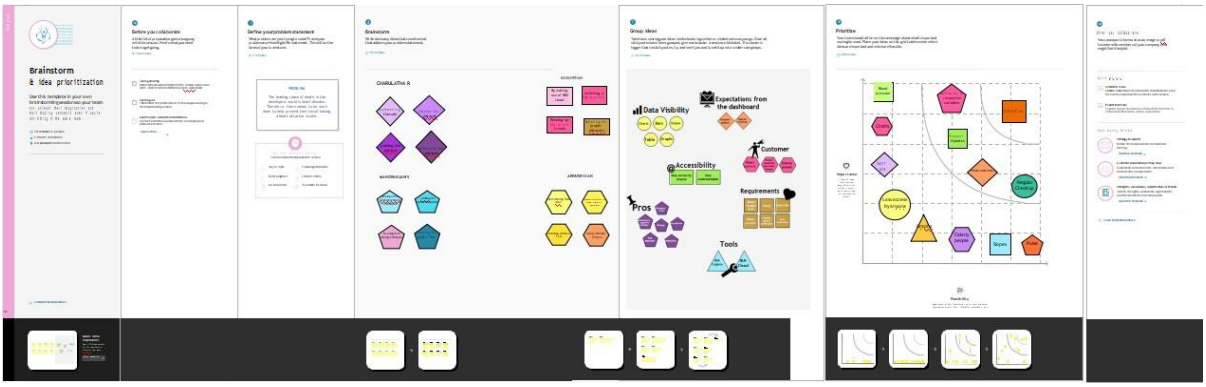
CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING



3.3 PROPOSED SOLUTION

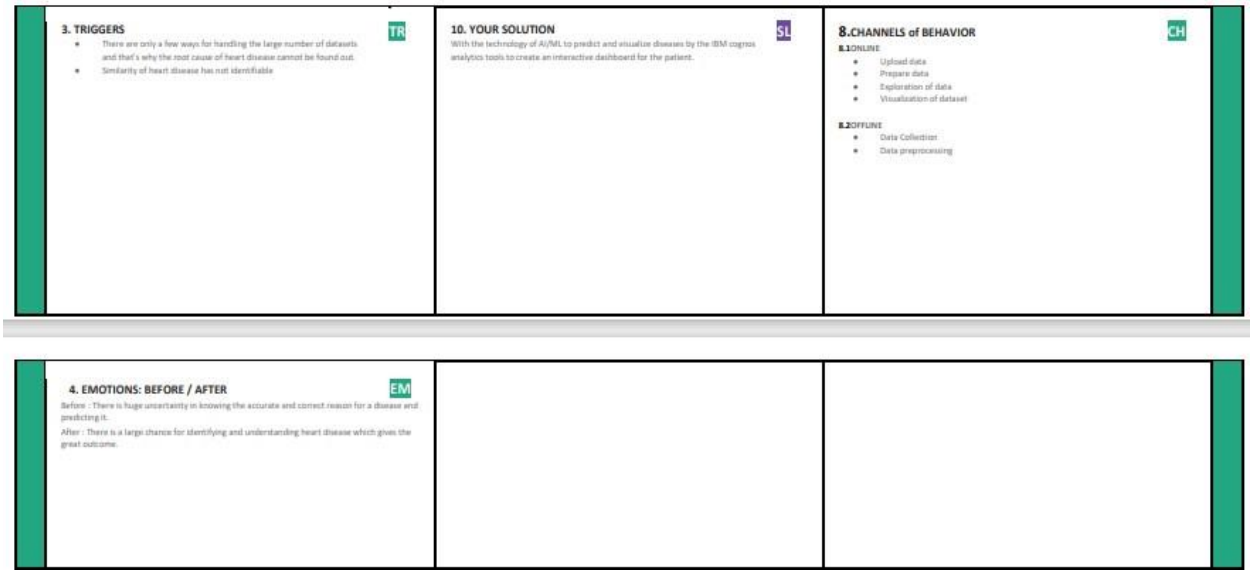
Our application helps the user in finding out if they have heart disease or not. They can find out by entering details such as their heart rate, cholesterol, blood pressure etc. A dashboard is also attached along with the results for better understanding where they can compare their blood pressure and similar metrics with other users. Our application has one of the smoothest user interfaces on the internet making it easy for the user to find their needs quickly and efficiently. And the tool utilizes best machine learning algorithms for better prediction. There are separate sections for viewing treatment options, warning signs of cardiac arrest, risk factors and causes of various types of heart diseases.

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The leading cause of death in the developed world is heart disease. Therefore, there needs to be work done to help prevent the risks of having a heart attack or stroke.
2.	Idea / Solution description	An effective heart disease prediction system (EHDPS) is developed using neural network for predicting the risk level of heart disease. The system uses 15 medical parameters such as age, sex, blood pressure, cholesterol, and obesity for prediction. The EHDPS predicts the likelihood of patients getting heart disease. It enables significant knowledge, eg, relationships between medical factors related to heart disease and patterns, to be established. We have employed the multilayer perceptron neural network with backpropagation as the training algorithm. The obtained results have illustrated that the designed diagnostic system can effectively predict the risk level of heart diseases. **Keywords:** data mining, neural network, multilayer perceptron neural network, backpropagation, disease diagnosis
3.	Novelty / Uniqueness	The predict of heart disease is done in three phases: feature selection process in this process we will automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. The second phase is applying the machine learning algorithms which are AdaBoost, XGBoost and Stacking in which the data will be trained and tested. The third and the last phase is the User interface in which the user will enter his details and then the machine learning models will predict that the user will have heart diseases in future or not.

4.	Social Impact / Customer Satisfaction	Given their flexibility, machine learning approaches may provide an opportunity to incorporate the complex nature of social determinants of health. The limited variety of sources and data in the reviewed studies
		emphasize that there is an opportunity to include more social determinants of health variables, especially environmental ones, that are known to impact cardiovascular disease risk and that recording such data in electronic databases will enable their use.
5.	Business Model (Revenue Model)	In today's digital world, several clinical decision support systems on heart disease prediction have been developed by different scholars to simplify and ensure efficient diagnosis. This paper investigates the state of the art of various clinical decision support systems for heart disease prediction, proposed by various researchers using data mining and machine learning techniques. Classification algorithms such as the Naïve Bayes (NB), Decision Tree (DT), and Artificial Neural Network (ANN) have been widely employed to predict heart diseases, where various accuracies were obtained. Hence, only a marginal success is achieved in the creation of such predictive models for heart disease patients therefore, there is need for more complex models that incorporate multiple geographically diverse data sources to increase the accuracy of predicting the early onset of the disease.
6.	Scalability of the Solution	• Our system response the user request with in few seconds. • Our system response many numbers of request at a time. • So, this will ensure our application will scalable

3.4 PROBLEM SOLUTION FIT

The Problem-Solution Fit simply means that we have found a problem with our customer and that the solution we have realized for it actually solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioural patterns and recognize what would work and why. The purpose is to solve complex problems in a way that fits the state of your customers and succeed faster and increase your solution adoption by tapping into existing mediums and channels of behaviour.



CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

- Users have to register.
- Function to view the homepage by the user.
- Function to display information related to heart diseases on the website.
- Function to provide text boxes to enter medical results.
- Function to predict heart disease using ML model.
- Function to display visualization of the final results.
- Function to provide dashboard to user.

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User verification	Verification through CAPTCHA Verification through I'm not a robot.
FR-4	User Authentication	Recognition of correct person Resending the code in case of forgot password.
FR-5	User validation	Reconfirming the new password Sending a two digit number in (Google account) your Old devices, so that you can enter into a new device By entering the two digit number.
FR-6	User Submission	Submission through Google form Submission through Email.

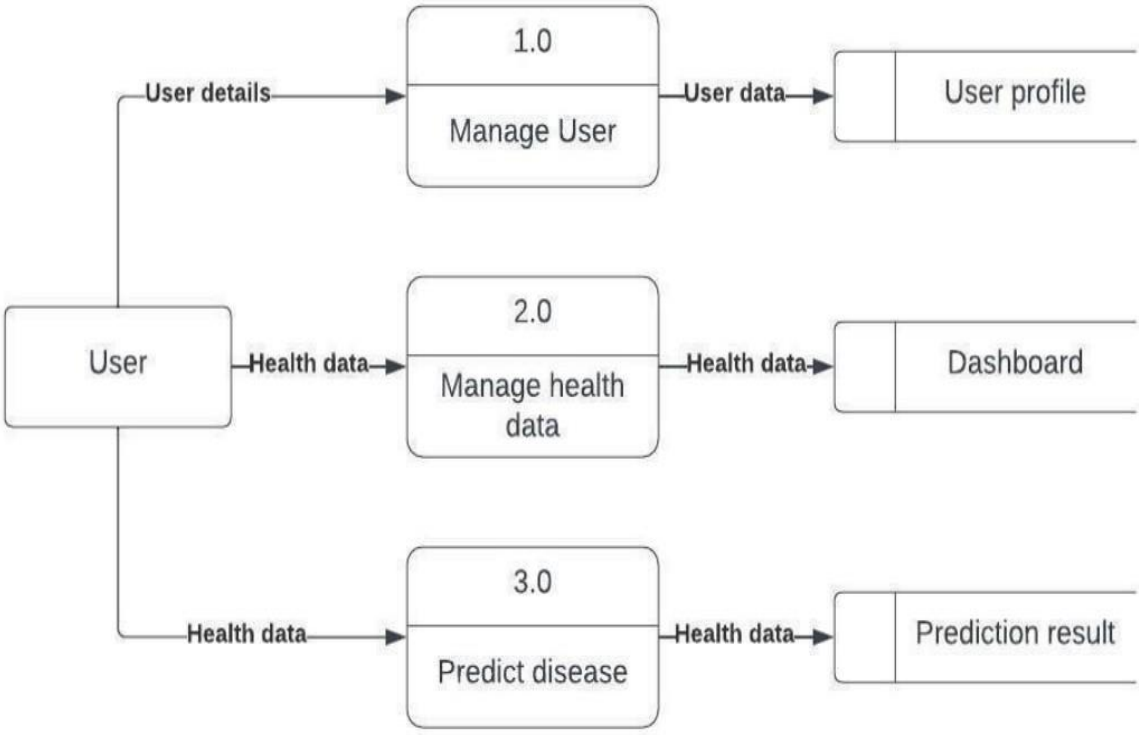
4.2 NON-FUNCTIONAL REQUIREMENTS

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The EHDPS predicts the likelihood of patients getting heart disease. It enables significant knowledge, eg, relationships between medical factors related to heart disease and patterns, to be established
NFR-2	**Security**	When we predict health analysis we provide a correct accuracy and prediction based on these prediction only the seriousness of diseases will be predicted.So the taken data will contains atleast some of true values.
NFR-3	**Reliability**	Support vector machine (SVM), Gaussian Naive Bayes, logistic regression, LightGBM, XGBoost, and random forest algorithm have been employed for developing heart disease risk prediction model and obtained the accuracy as 80.32%, 78.68%, 80.32%, 77.04%, 73.77%, and 88.5%, respectively
NFR-4	**Performance**	This study found that using a heart disease dataset collected from Kaggle threeclassification based decision tree along with accuracy, sensitivity and specificity.
NFR-5	**Availability**	Machine Learning can play an essential role in predicting presence/absence of Locomotor disorders, Heart diseases and more. Such information, if predicted well in advance, can provide important insights to doctors who can then adapt their diagnosis and treatment per patient basis.
NFR-6	**Scalability**	It is depend on the model performance. If the accuracy will not satisfied we will improve the accuracy by boosting method.The high accuracy can achieved through removing duplicates and performing data Cleaning.

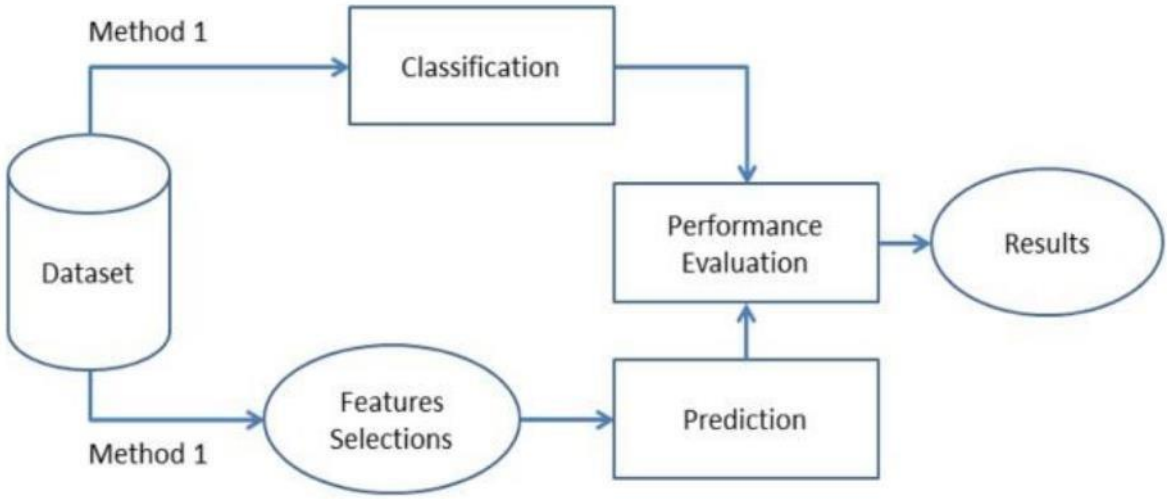
CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS



5.2 SOLUTION & TECHNICAL ARCHITECTURE



5.3 User Stories:

S.No	Component	Description	Technology
1.	Importing data	Data Import lets you upload data from external sources and combine it with data you collect via Analytics	Python,numpy, pandas.
2.	Data Cleaning	Data cleaning is a process by which inaccurate, poorly formatted, or otherwise messy data is organized and corrected	Python
3.	Data Pre-processing	Data pre-processing, a component of data preparation, describes any type of processing performed on raw data to prepare it for another data processing procedure	Python
4.	Training data	Training data is the subset of original data that is used to train the machine learning model,	python
5.	Testing data	Test data is data which has been specifically identified for use in tests, typically of a computer program**.**	python.
6.	Machine learning model	A machine learning model is a file that has been trained to recognize certain types of patterns. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data	python.
7.	Improve model performance	Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right.	python.
8.	Checking accuracy	A data accuracy check, sometimes called a data sanity check, is a set of quality validations that take place before using data.	python.

5.4: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Collection of data	Data collection is the process of gathering, measuring, and analyzing accurate data from a variety of relevant sources to find answers to research problems, answer questions, evaluate outcomes, and forecast trends and probabilities	Python,numpy,pandas
2.	EDA Analysis	Exploratory Data Analysis (EDA) is an approach to analyze the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations	Technology used
3.	Train & Test split of data	The train-test split is used to estimate the performance of machine learning algorithms that are applicable for prediction-based Algorithms/Applications. This method is a fast and easy procedure to perform such that we can compare our own machine learning model results to machine results.	Technology used

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Sprint 1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Charulatha R
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	
		USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard with Gmail Login	Medium	
Sprint 2	Login	USN-5	As a user, I can log into the application by entering email& password	I can register & access the dashboard with Gmail Login	High	Arivuselvan M
Sprint 3	Dashboard	USN-6	Profile - view & update your profile	I can see the profile.	High	Mukunthan M
		USN-7	Change Password - user can change the password	I can able to change the password.	High	
Sprint 4	Classified result	USN-8	Home - Analyse your Heart	I can detect the health condition from where everI want.	High	ManoRanjani S

		USN-9	The user will have to fill in the below 13 fields for the system to predict a disease -Age in Year Gender -Chest Pain Type -Fasting Blood Sugar -Resting Electrographic Results (Restecg) -Exercise Induced	These are the categories available in that applicatio n.	High	

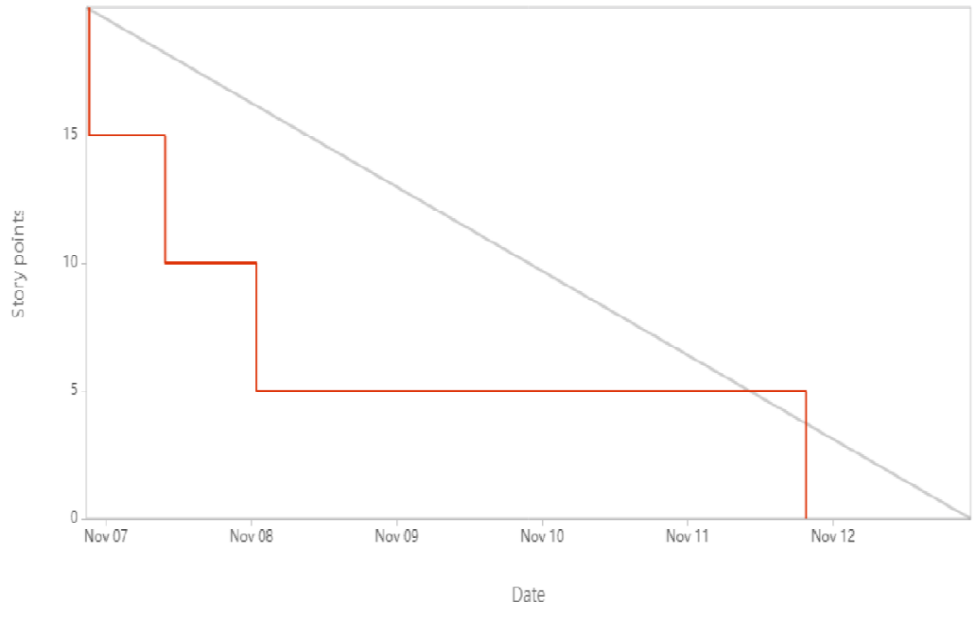
6.2 SPRINT DELIVERY SCHEDULE

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	1	6 Days	20 Nov 2022	26 Nov 2022	20	26 Nov 2022
Sprint-2	1	6 Days	27 Nov 2022	02 Dec 2022	18	02 Dec 2022
Sprint-3	1	6 Days	03 Dec 2022	09 Dec 2022	20	09 Dec 2022
Sprint-4	1	6 Days	10 Dec 2022	16 Dec 2022	19	16 Dec 2022

6.3 REPORTS FROM JIRA

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



CHAPTER 7

CODING & SOLUTIONING

7.1 FEATURE 1

Prediction Model: When applied to a nonlinear data set, the random forest technique performs better than the decision tree. The collection of decision trees known as a random forest was produced by several root nodes. The random forest algorithm can achieve more accuracy quickly and produce expected results.

Algorithm:

Step 1: Input the required details

Step 2: The model processes the input with the help of random forest algorithm

Step 3: The results are displayed

Code:

```
import axios from 'axios'; import React, { useState }
from 'react'; import { useNavigate } from 'react-router-
dom'; import './FormPageCommons.css'; function
PredictionPage() { const [age, setAge] = useState('');
const [sex, setSex] = useState(''); const
[chestPainType, setChestPainType] = useState('');
const [bp, setBp] = useState(''); const [cholesterol,
setCholesterol] = useState(''); const [fbs, setFbs] =
useState('');
```

```
 const [ekg, setEkg] = useState(''); const [maxHr, setMaxHr] = useState(''); const [angina, setAngina] = useState(''); const [stDepression, setStDepression] = useState(''); const [slopeOfSt, setSlopeOfSt] = useState(''); const [fluro, setFluro] = useState(''); const [thallium, setThallium] = useState(''); const navigate = useNavigate(); function predictResult(event) {  event.preventDefault();  if (age && sex && chestPainType && bp && cholesterol && fbs && ekg && maxHr && angina && stDepression && slopeOfSt && fluro && thallium) {
  const predictionParams = {age,sex,chest_pain_type: chestPainType,bp,cholesterol,fbs_over_120: fbs, ekg_results: ekg,max_hr: maxHr,exercise_angina: angina, st_depression: stDepression,slope_of_st: slopeOfSt, number_of_vessels_fluro: fluro, thallium,}; const headers = {
   Authorization: `Bearer ${sessionStorage.getItem('token')}`,
  };
  axios.post('http://127.0.0.1:8000/predict', predictionParams, { headers })
   .then((response) => { predictionParams.prediction = response.data.prediction;
    [predictionParams.date, predictionParams.time] = new Date().toLocaleString().split(', '); sessionStorage.setItem('prediction', JSON.stringify(predictionParams)); navigate('/results');
   }) // eslint-disable-line no-console
   .catch((ex) => console.log(ex)); // eslint-disable-line no-console
 } }
return (
 <div className="main-container">
  <form className="main-form" onSubmit={predictResult}>
```

```
<div className="title">Enter your details</div>
<div className="form-input">
<p>Age</p>
<input type="number" name="Age" id="text-input" value={age} onChange={(e) =>
setAge(e.target.value)} />
</div>
<div className="form-input">
<p>Sex</p> <select
name="sex" id="sel-input"
value={sex} onChange={(e) =>
setSex(e.target.value)}
>
<option value="-1">--Select Value--</option>
<option value="1">Male</option>
<option value="0">Female</option>
</select>
</div>
<div className="form-input">
<p>Chest Pain Type</p>
<select name="chest-pain" id="sel-
input" value={chestPainType}
onChange={(e) => setChestPainType(e.target.value)}
>
<option value="-1">--Select Value--</option>
<option value="1">Typical Angina</option>
```

```
          <option value="2">Atypical Angina</option>
          <option value="3">Non Anginal Pain</option>
          <option value="4">Asymptomatic Pain</option>
        </select>
      </div>
      <div className="form-input">
        <p>Blood Pressure</p>
        <input type="number" className="form_field" name="bp" id="text-input"
value={bp} onChange={(e) => setBp(e.target.value)} />
      </div>
      <div className="form-input">
        <p>Cholesterol</p>
        <input type="number" name="cholesterol" id="text-input"
value={cholesterol} onChange={(e) => setCholesterol(e.target.value)} />
      </div>
      <div className="form-input">
        <p>Fasting Blood Sugar Over 120</p>
        <select name="fbs" id="sel-
input" value={fbs}
onChange={(e) => setFbs(e.target.value)}
        >
          <option value="-1">--Select Value--</option>
          <option value="1">Yes</option>
          <option value="0">No</option>
        </select>
```

```
        </div>
        <div className="form-input">
          <p>EKG Results</p>
          <select name="ekg" id="sel-
input" value={ekg}
onChange={(e) => setEkg(e.target.value)}
          >
            <option value="-1">--Select Value--</option>
            <option value="0">Normal</option>
            <option value="1">Having ST-T wave abnormality (T wave inversions and/or ST
elevation or depression of greater than 0.05 mV)</option>
            <option value="2">Showing probable or definite left ventricular hypertrophy by
Estes`&amp;apos;` criteria</option>
          </select>
        </div>
        <div className="form-input">
          <p>Maximum Heart Rate</p>
          <input type="number" name="mhr" id="text-input" value={maxHr} onChange={(e)
=> setMaxHr(e.target.value)} />
        </div>
        <div className="form-input">
          <p>Exercise Induced Angina</p>
          <select name="angina" id="sel-
input" value={angina}
onChange={(e) => setAngina(e.target.value)}
```

```
          >
            <option value="-1">--Select Value--</option>
            <option value="1">Yes</option>
            <option value="0">No</option>
          </select>
        </div>
        <div className="form-input">
          <p>ST Depression</p>
          <input type="number" name="stdep" id="text-input" value={stDepression}
onChange={(e) => setStDepression(e.target.value)} />
        </div>
        <div className="form-input">
          <p>Slope of ST</p>
          <select name="slope-st" id="sel-
input" value={slopeOfSt}
onChange={(e) => setSlopeOfSt(e.target.value)}
          >
            <option value="-1">--Select Value--</option>
            <option value="1">Upsloping</option>
            <option value="2">Flat</option>
            <option value="3">Downsloping</option>
          </select>
        </div>
        <div className="form-input">
          <p>Number of major vessels colored by Flouroscopy</p>
```

```
        <select name="fluro" id="sel-
input" value={fluro}
onChange={(e) => setFluro(e.target.value)}
>

          <option value="-1">--Select Value--</option>
          <option value="0">0</option>
          <option value="1">1</option>
          <option value="2">2</option>
<option value="3">3</option>
        </select>
      </div>
      <div className="form-input">
        <p>Thallium</p>
        <select name="thallium" id="sel-
input" value={thallium}
onChange={(e) => setThallium(e.target.value)}
        >
          <option value="-1">--Select Value--</option>
          <option value="3">Normal</option>
          <option value="6">Fixed Defect</option>
          <option value="7">Reversable Defect</option>
        </select>
      </div>
      <button type="text" className="submit">Submit</button>
```

```
    </form>
   </div>
  );
} export default
PredictionPage;
```

7.2 FEATURE 2

Dashboard: Our application helps the user in finding out if they have heart disease or not. They can find out by entering details such as their heart rate, cholesterol, blood pressure etc. A dashboard is also attached along with the results for better understanding where they can compare their blood pressure and similar metrics with other users.

Code:

```
import Sidebar from "../../components/sidebar/Sidebar"
import "./home.scss" import "./Cards.css" const
DashboardHome = () => {
 return (
  <div className="home">
   <Sidebar />
   <div className="homeContainer">
    <h1>Welcome to your Dashboard!</h1>
    <br />
    <br />
    <br />
    <h3>Check out your email. Your results have been sent there.</h3>
```

```
<br />

<br />

<h3>Here you can check out different kind of visualizations to get a general idea about the factors increasing causes of getting a cardiac arrest.</h3>

<br />

<div class="row">

 <div class="col-md-4 col-xl-3">

  <div class="card bg-c-blue order-card">

   <div class="card-block">

    <h2  class="text-right"><i  class="fa         fa-cart-plus fleft"></i><span>87%</span></h2>

    <p class="m-b-0">Accuracy</p>

   </div>

  </div>

 </div>

 <div class="col-md-4 col-xl-3">

  <div class="card bg-c-green order-card">

   <div class="card-block">

    <h2 class="text-right"><i class="fa fa-cart-plus f-left"></i><span>5</span></h2>

    <p class="m-b-0">Visualization Types</p>

   </div>

  </div>

 </div>

 <div class="col-md-4 col-xl-3">

  <div class="card bg-c-yellow order-card">
```

```
            <div class="card-block">
              <h2 class="text-right"><i class="fa fa-cart-plus f-left"></i><span>Random Forest Classifier</span></h2>
              <p class="m-b-0">ML Model</p>
            </div>
          </div>
        </div>
      </div>
    </div>
  )
} export default
DashboardHome
```

7.3 FEATURE 3

Login

Algorithm:

1. Input the credentials (email and password).
2. If already logged in user is taken to home page
3. Else, check for validity of credentials
4. If wrong credentials entered , notification is displayed to user and user stays in login page.
5. On correct credentials, user is taken to home page.

Code:

```
import axios from 'axios'; import React, {
useState } from 'react'; import
'./FormPageCommons.css'; import {
useNavigate } from 'react-router-dom'; const
Login = () => { const [email, setEmail] =
useState(''); const [password, setPassword] =
useState(''); const navigate = useNavigate();
function loginUser(event) {
event.preventDefault();     const userDetails = {
email, password,
   };
   // console.log(userDetails); // eslint-disable-line no-console     if
(userDetails && userDetails.email && userDetails.password) {
axios.post('http://127.0.0.1:8000/login', userDetails)
       .then((response) => {
sessionStorage.setItem('token', response.data.token);
navigate('/predict');
       })
       .catch((ex) => {
         // console.log(ex); // eslint-disable-line no-console
         // const error = JSON.parse(ex);
         if (ex.response && ex.response.status && ex.response.status === 404) {
alert('User not found'); // eslint-disable-line no-alert
```

```
        } else {            console.log(ex); // eslint-disable-
line no-console
        }
       });
   } else {       alert('Please enter valid credentials'); // eslint-disable-
line no-alert     } } return (
   <div className="main-container">
     <form className="main-form" onSubmit={loginUser}>
       <div className="form-input">
         <p>e-mail</p>
         <input type="email" name="email" id="email" value={email} onChange={(e) =>
setEmail(e.target.value)} />
       </div>
       <div className="form-input">
         <p>Password</p>
         <input type="password" name="password" id="password" value={password}
onChange={(e) => setPassword(e.target.value)} />
       </div>
       <div className="button-container">
         <input type="submit" />
       </div>
     </form>
   </div>
  ); }; export default
Login;
```

7.3.1 FEATURE 4:

Signup

Algorithm:

1. Input the signup form fields (name , email , password , re-enter password).
2. All credentials are validated.
3. Website checks whether the given email exists in the database.
4. If already registered, notification is displayed. Or else, the user is taken to the login page.

Code:

```
import axios from 'axios'; import React, {
useState } from 'react'; import { useNavigate }
from 'react-router-dom'; import { Link } from
'react-router-dom'; const Register = () => {
const [name, setName] = useState('');

  const [email, setEmail] = useState('');  const
[password, setPassword] = useState(''); const
[confPassword, setConfPassword] = useState('');  const
navigate = useNavigate();    function
registerUser(event) {     event.preventDefault();
const userDetails = {       fullName: name, email,
password,
```

```
    };
    if (userDetails && userDetails.fullName && userDetails.password && userDetails.email)
{
      // console.log('Hi'); // eslint-disable-line no-console
axios.post('http://127.0.0.1:8000/register', userDetails)
        .then(() => {
          // console.log(response.json); // eslint-disable-line no-console
navigate('/login');
        })
        .catch((ex) => console.log(ex)); // eslint-disable-line no-console
    }   }
return (
    <div className="main-container">
      <form className="login-form" onSubmit={registerUser}>
        <div className="form-input">
          <p>Name</p>
          <input type="text" name="text" id="text" value={name} onChange={(e) =>
setName(e.target.value)} />
        </div>
        <div className="form-input">
          <p>e-mail</p>
          <input type="email" name="email" id="email" value={email} onChange={(e) =>
setEmail(e.target.value)} />
        </div>
        <div className="form-input">
```

```
          <p>Password</p>
          <input type="password" name="password" id="password" value={password}
onChange={(e) => setPassword(e.target.value)} />
        </div>
        <div className="form-input">
          <p>Confirm Password</p>
          <input type="password" name="password" id="conf-password" value={confPassword}
onChange={(e) => setConfPassword(e.target.value)} />
        </div>
        <div className="button-container">
          <input type="submit" />
          <p>Already have an account?</p>
          <Link to="/login">
            <button type="button">Login</button>
          </Link>
        </div>
      </form>
    </div>
  ); }; export default
Register;
```

7.3.2 DATABASE SCHEMA

NoSQL databases like MongoDB offer high performance, high availability, and easy scalability. MongoDB is a documentoriented database which stores data in JSON-like documents with dynamic schema. It means you can store your records without worrying about the data structure such as the number of fields or types of fields to store values. MongoDB documents are similar to JSON objects. Details like name, e-mail, password of the registered user are stored so that when the user tries to login, authentication takes place and the user is logged in.

7.4 Training and Testing the Dataset

In [84]:
```
import numpy as np
import pandas as pd
import plotly
import plotly.express as px

import cufflinks as cf
import matplotlib.pyplot as plt
import seaborn as sns

import plotly.offline as pyo
from plotly.offline import init_notebook_mode,plot,iplot
```

In [85]:
```
pyo.init_notebook_mode(connected=True)
cf.go_offline()
```

In [86]:
```
heart
```

Out[86]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Heart Disease
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	1
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	0
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	1
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	0
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
265	52	1	3	172	199	1	0	162	0	0.5	1	0	7	0
266	44	1	2	120	263	0	0	173	0	0.0	1	0	7	0
267	56	0	2	140	294	0	2	153	0	1.3	2	0	3	0
268	57	1	4	140	192	0	0	148	0	0.4	2	0	6	0
269	67	1	4	160	286	0	2	108	1	1.5	2	3	3	1

270 rows × 14 columns

In [87]:
```
heart = pd.read_csv(r'C:\Users\91967\Desktop\data\heart.csv')
```

In [88]:
```
info = ["Age","1: male, 0: female","Chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic","resting blood pressure"," serum cholestoral in mg



for i in range(len(info)):
    print(heart.columns[i]+":\t\t\t"+info[i])
```

```
Age:                    Age
Sex:                    1: male, 0: female
Chest pain type:                Chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic
BP:                     resting blood pressure
Cholesterol:                     serum cholestoral in mg/dl
FBS over 120:                   fasting blood sugar > 120 mg/dl
EKG results:                    resting electrocardiographic results (values 0,1,2)
Max HR:                  maximum heart rate achieved
Exercise angina:                        exercise induced angina
ST depression:                  oldpeak = ST depression induced by exercise relative to rest
Slope of ST:                    the slope of the peak exercise ST segment
Number of vessels fluro:                        number of major vessels (0-3) colored by flourosopy
Thallium:                       thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
```

In [89]:
```
heart['Heart Disease']
```

Out[89]:
```
0      Presence
1       Absence
2      Presence
3       Absence
4       Absence
         ...
265     Absence
266     Absence
267     Absence
268     Absence
269    Presence
Name: Heart Disease, Length: 270, dtype: object
```

In [90]:
```
heart.groupby('Heart Disease').size()
```

Out[90]:
```
Heart Disease
Absence     150
Presence    120
dtype: int64
```

In [91]:
```
heart.groupby('Heart Disease').sum()
```

Out[91]:

Heart Disease	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium
Absence	7906	83	423	19330	36632	23	129	23750	23	93.4	210	43	568
Presence	6791	100	434	16133	30776	17	147	16663	66	190.1	218	138	700

In [92]:
```
heart.shape
```

Out[92]:
```
(270, 14)
```

In [93]:
```
heart.describe()
```

Out[93]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium
count	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.00000	270.000000	270.000000	270.000000
mean	54.433333	0.677778	3.174074	131.344444	249.659259	0.148148	1.022222	149.677778	0.329630	1.05000	1.585185	0.670370	4.696296
std	9.109067	0.468195	0.950090	17.861608	51.686237	0.355906	0.997891	23.165717	0.470952	1.14521	0.614390	0.943896	1.940659
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.00000	1.000000	0.000000	3.000000
25%	48.000000	0.000000	3.000000	120.000000	213.000000	0.000000	0.000000	133.000000	0.000000	0.00000	1.000000	0.000000	3.000000
50%	55.000000	1.000000	3.000000	130.000000	245.000000	0.000000	2.000000	153.500000	0.000000	0.80000	2.000000	0.000000	3.000000
75%	61.000000	1.000000	4.000000	140.000000	280.000000	0.000000	2.000000	166.000000	1.000000	1.60000	2.000000	1.000000	7.000000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.20000	3.000000	3.000000	7.000000

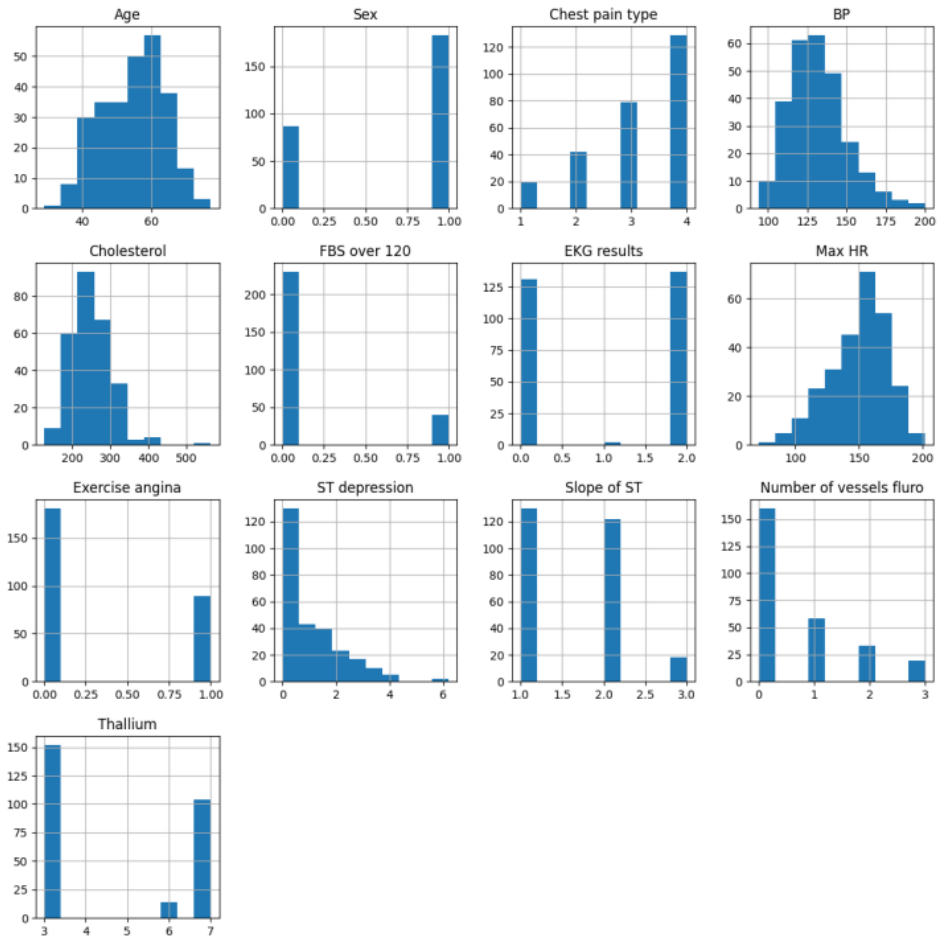
In [94]:
```
heart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       270 non-null    int64
 1   Sex                       270 non-null    int64
 2   Chest pain type           270 non-null    int64
 3   BP                        270 non-null    int64
 4   Cholesterol               270 non-null    int64
 5   FBS over 120              270 non-null    int64
 6   EKG results               270 non-null    int64
 7   Max HR                    270 non-null    int64
 8   Exercise angina           270 non-null    int64
 9   ST depression             270 non-null    float64
 10  Slope of ST               270 non-null    int64
 11  Number of vessels fluro   270 non-null    int64
 12  Thallium                  270 non-null    int64
 13  Heart Disease             270 non-null    object
dtypes: float64(1), int64(12), object(1)
memory usage: 29.7+ KB
```

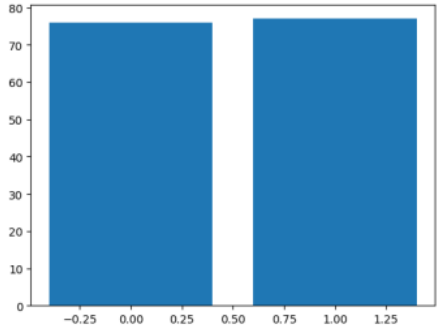
In [95]:
```
heart['Heart Disease'].unique()
```

Out[95]:
```
array(['Presence', 'Absence'], dtype=object)
```

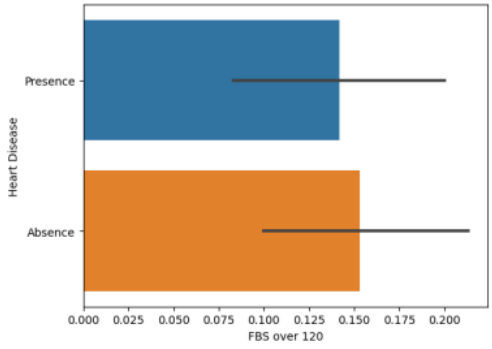
In [96]:
```
heart.hist(figsize=(14,14))
plt.show()
```



In [97]:
```
plt.bar(x=heart['Sex'],height=heart['Age'])
plt.show()
```

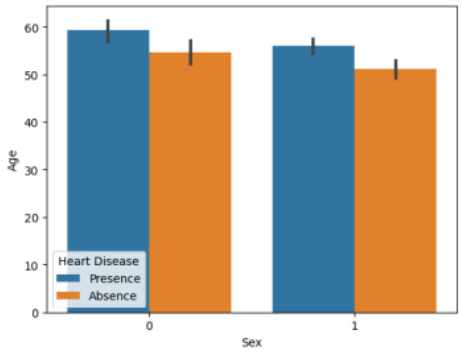


In [98]:
```
sns.barplot(x="FBS over 120", y="Heart Disease", data=heart)
plt.show()
```



In [99]:
```
sns.barplot(x=heart['Sex'],y=heart['Age'],hue=heart['Heart Disease'])
```

Out[99]:
```
<AxesSubplot: xlabel='Sex', ylabel='Age'>
```



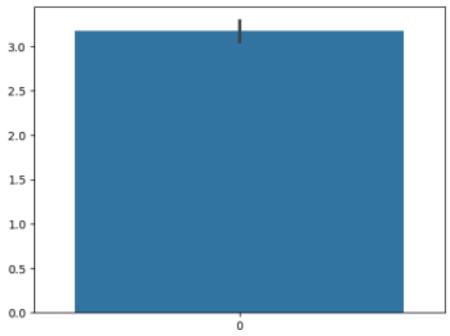
In [102…
```
heart['Heart Disease']=heart['Heart Disease'].replace({'Absence':0,'Presence':1})
```

In [103…
```
heart['Heart Disease']
```

Out[103]:
```
0      1
1      0
2      1
3      0
4      0
      ..
265    0
266    0
267    0
268    0
269    1
Name: Heart Disease, Length: 270, dtype: int64
```

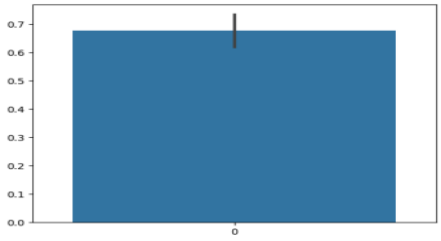
In [104…
```
sns.barplot(heart["Chest pain type"])
```

Out[104]:
```
<AxesSubplot: >
```



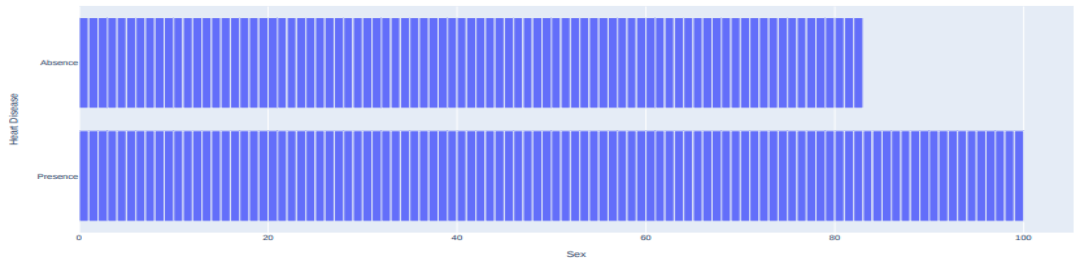
In [105…
```
sns.barplot(heart["Sex"])
```

Out[105]:
```
<AxesSubplot: >
```



In [ ]:
```
```

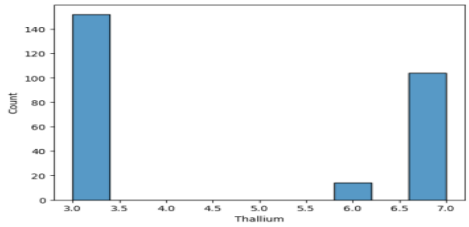
In [69]:
```
px.bar(heart,heart['Sex'],heart['Heart Disease'])
```



Sex

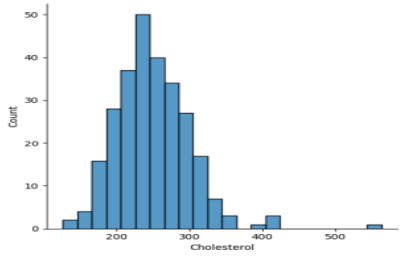
```
In [70]: sns.histplot(heart["Thallium"])
```

```
Out[70]: <AxesSubplot: xlabel='Thallium', ylabel='Count'>
```




```
In [73]: sns.displot(heart["Cholesterol"])
```

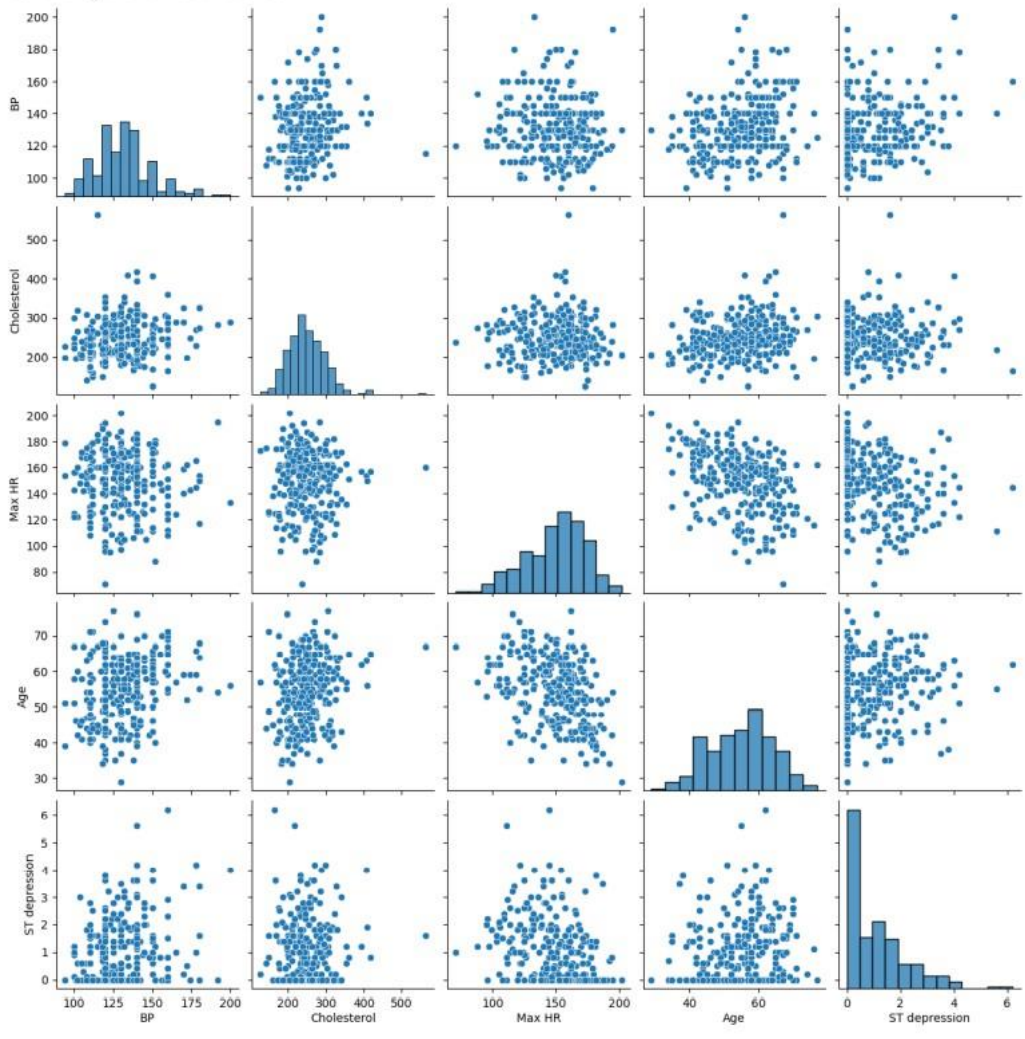
```
Out[73]: <seaborn.axisgrid.FacetGrid at 0x12de4f95690>
```




```
In [76]: numeric_columns=['BP','Cholesterol','Max HR','Age','ST depression']
```

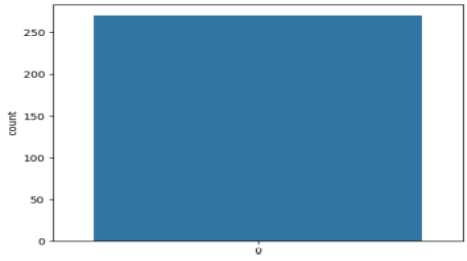
```
In [77]: sns.pairplot(heart[numeric_columns])
```

```
Out[77]: <seaborn.axisgrid.PairGrid at 0x12dce642830>
```

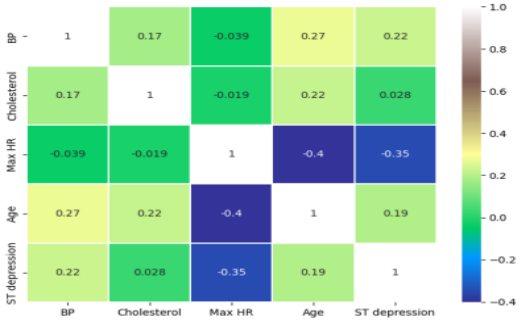


```
In [113]: heart.rename(columns={'Heart Disease':'target'},inplace=True)

In [114]: y = heart["target"]

          sns.countplot(y)

          target_temp = heart.target.value_counts()

          print(target_temp)
```

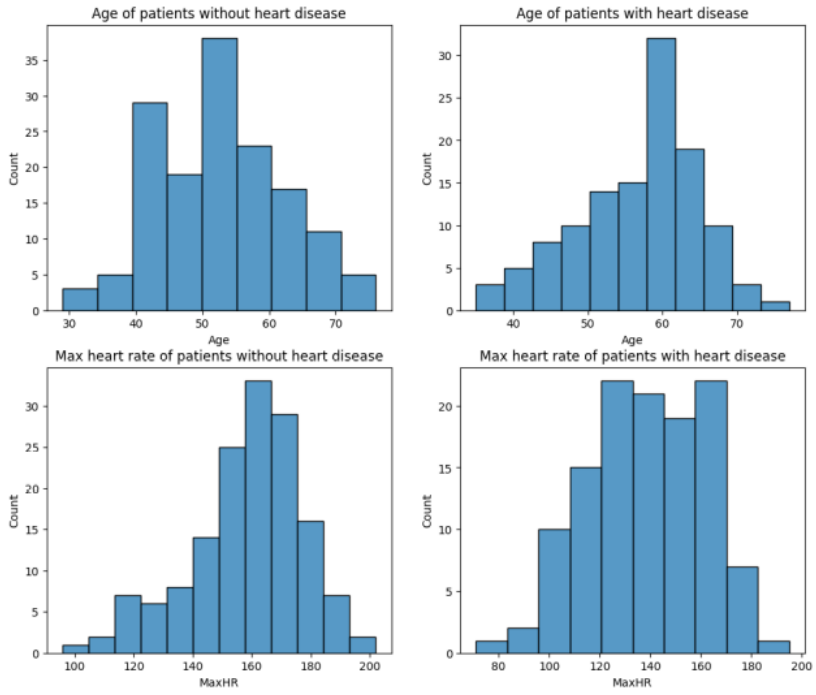
```
0    150
1    120
Name: target, dtype: int64
```



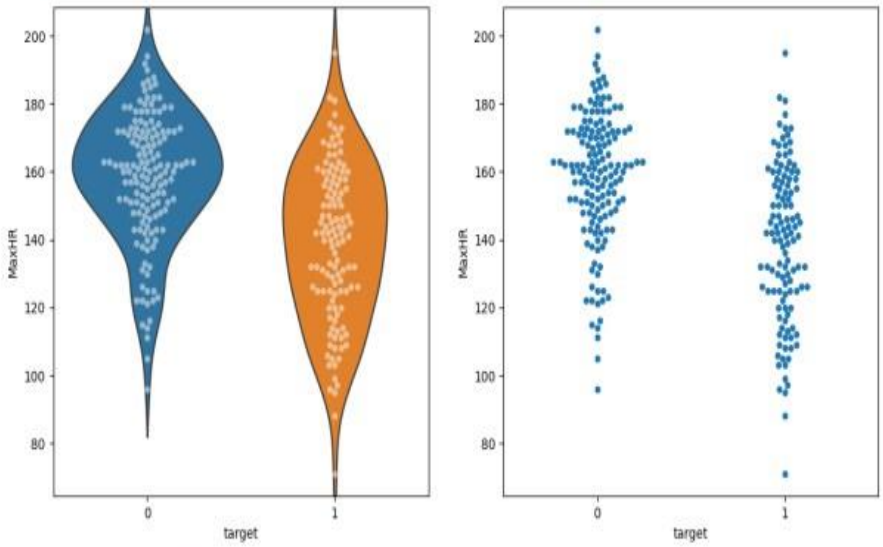

```
In [79]: sns.heatmap(heart[numeric_columns].corr(),annot=True, cmap='terrain', linewidths=0.1)
         fig=plt.gcf()
         fig.set_size_inches(8,6)
         plt.show()
```



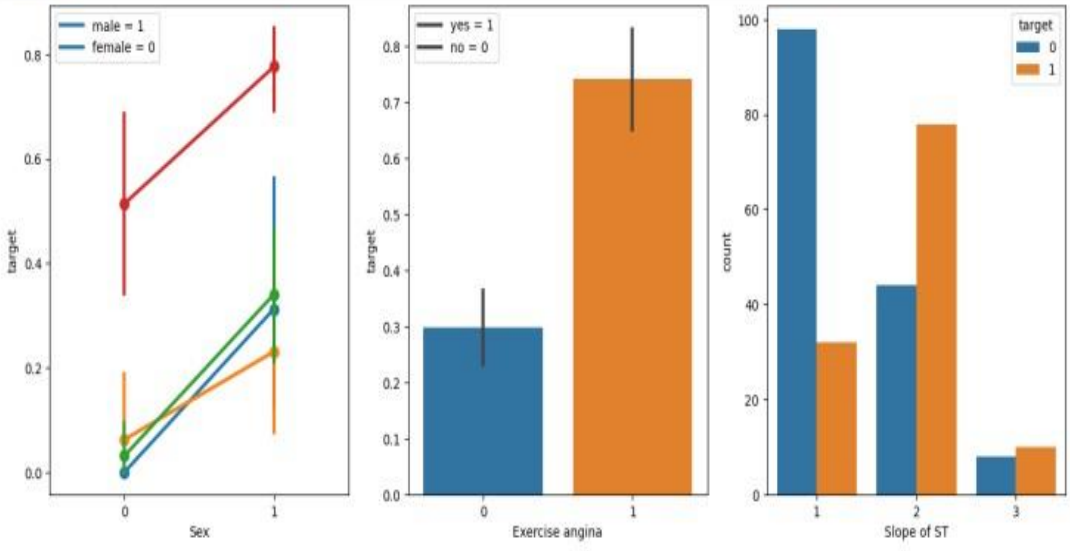

```
In [130]: heart.rename(columns={'Max HR':'MaxHR'},inplace=True)

In [131]: # create four distplots
          plt.figure(figsize=(12,10))
          plt.subplot(221)
          sns.histplot(heart[heart['target']==0].Age)
          plt.title('Age of patients without heart disease')
          plt.subplot(222)
          sns.histplot(heart[heart['target']==1].Age)
          plt.title('Age of patients with heart disease')
          plt.subplot(223)
          sns.histplot(heart[heart['target']==0].MaxHR)
          plt.title('Max heart rate of patients without heart disease')
          plt.subplot(224)
          sns.histplot(heart[heart['target']==1].MaxHR)
          plt.title('Max heart rate of patients with heart disease')
          plt.show()
```



```
In [132...
plt.figure(figsize=(13,6))
plt.subplot(121)
sns.violinplot(x="target", y="MaxHR", data=heart, inner=None)
sns.swarmplot(x="target", y="MaxHR", data=heart, color='w', alpha=0.5)


plt.subplot(122)
sns.swarmplot(x="target", y="MaxHR", data=heart)
plt.show()
```




```
In [133...
# create pairplot and two barplots
plt.figure(figsize=(16,6))
plt.subplot(131)
sns.pointplot(x="Sex", y="target", hue='Chest pain type', data=heart)
plt.legend(['male = 1', 'female = 0'])
plt.subplot(132)
sns.barplot(x="Exercise angina", y="target", data=heart)
plt.legend(['yes = 1', 'no = 0'])
plt.subplot(133)
sns.countplot(x="Slope of ST", hue='target', data=heart)
plt.show()
```



DATA Processing

```
In [134]: heart['target'].value_counts()
```

```
Out[134]: 0    150
          1    120
          Name: target, dtype: int64
```

```
In [135]: heart['target'].isnull()
```

```
Out[135]: 0      False
          1      False
          2      False
          3      False
          4      False
                 ...
          265    False
          266    False
          267    False
          268    False
          269    False
          Name: target, Length: 270, dtype: bool
```

```
In [136]: heart['target'].sum()
```

```
Out[136]: 120
```

```
In [137]: heart['target'].unique()
```

```
Out[137]: array([1, 0], dtype=int64)
```

```
In [138]: heart.isnull().sum()
```

```
Out[138]: Age                      0
          Sex                      0
          Chest pain type          0
          BP                       0
          Cholesterol              0
          FBS over 120             0
          EKG results              0
          MaxHR                    0
          Exercise angina          0
          ST depression            0
          Slope of ST              0
          Number of vessels fluro  0
          Thallium                 0
          target                   0
          dtype: int64
```

Storing in X and y

```
In [139]: X,y=heart,heart.target
```

```
In [140]: X.drop('target',axis=1,inplace=True)
```

```
In [141]: y
```

```
Out[141]: 0      1
          1      0
          2      1
          3      0
          4      0
                ..
          265    0
          266    0
          267    0
          268    0
          269    1
          Name: target, Length: 270, dtype: int64
```

Or X, y = heart.iloc[:, :-1], heart.iloc[:, -1]

```
In [142]: X.shape
```

```
Out[142]: (270, 13)
```

```
In [143]: y.shape
```

```
Out[143]: (270,)
```

```
In [144]: from sklearn.model_selection import train_test_split
          from sklearn.preprocessing import StandardScaler
```

```
In [145]: sc = StandardScaler()
          X = sc.fit_transform(X)
```

```
In [146]: X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=10,test_size=0.3,shuffle=True)

In [147]: X_test

Out[147]: array([[-1.47745975,  0.6894997 , -1.23804513, ..., -0.95423434,
                  -0.71153494, -0.87570581],
                 [ 1.68210896,  0.6894997 , -2.79253153, ...,  0.67641928,
                   0.3498787 , -0.87570581],
                 [-0.37761378,  0.6894997 , -0.18355874, ...,  0.67641928,
                  -0.71153494, -0.87570581],
                 ...,
                 [-0.81755217,  0.6894997 , -0.18355874, ..., -0.95423434,
                  -0.71153494, -0.87570581],
                 [ 0.50226299, -1.45032695,  0.87092765, ...,  0.67641928,
                  -0.71153494, -0.87570581],
                 [-0.70756757,  0.6894997 , -0.18355874, ..., -0.95423434,
                   1.41127648, -0.87570581]])

In [148]: y_test

Out[148]: 111    0
          170    0
          186    0
          185    1
          121    1
                ..
          217    0
          250    1
          69     1
          58     1
          194    0
          Name: target, Length: 81, dtype: int64

In [149]: print ("train_set_x shape: " + str(x_train.shape))
          print ("train_set_y shape: " + str(y_train.shape))
          print ("test_set_x shape: " + str(X_test.shape))
          print ("test_set_y shape: " + str(y_test.shape))

          train_set_x shape: (189, 13)
          train_set_y shape: (189,)
          test_set_x shape: (81, 13)
          test_set_y shape: (81,)
```

Model

```
In [150]: # Decision Tree Classifier
          scores_dict = {}

In [151]: Catagory=['No....but i pray you get Heart Disease or at leaset Corona Virus Soon...','Yes you have Heart Disease....RIP in Advance']

In [155]: from sklearn.tree import DecisionTreeClassifier
          dt=DecisionTreeClassifier()
          dt.fit(X_train,y_train)

Out[155]: DecisionTreeClassifier
          DecisionTreeClassifier()

In [156]: print("Accuracy on training set: {:.3f}".format(dt.score(X_train, y_train)))
          print("Accuracy on test set: {:.3f}".format(dt.score(X_test, y_test)))

          Accuracy on training set: 1.000
          Accuracy on test set: 0.778

In [157]: prediction

Out[157]: array([0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1,
                 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0,
                 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0,
                 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0], dtype=int64)

In [158]: X_DT=np.array([[63 ,1, 3,145,233,1,0,150,0,2.3,0,0,1]])
          X_DT_prediction=dt.predict(X_DT)

In [159]: X_DT_prediction[0]

Out[159]: 1

In [160]: print(Catagory[int(X_DT_prediction[0])])

          Yes you have Heart Disease....RIP in Advance
```

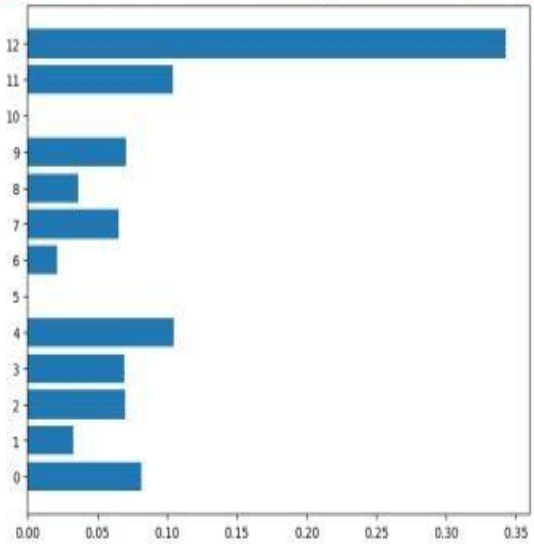
Feature Importance in Decision Trees

In [161]:
```
print("Feature importances:\n{}".format(dt.feature_importances_))
```

```
Feature importances:
[0.08148922 0.03287338 0.07020353 0.06957237 0.10489245 0.
 0.02144372 0.06531475 0.03652597 0.07047152 0.         0.10448896
 0.34272413]
```

In [162]:
```
def plot_feature_importances_diabetes(model):
    plt.figure(figsize=(8,6))
    n_features = 13
    plt.barh(range(n_features), model.feature_importances_, align='center')
    plt.yticks(np.arange(n_features), X)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature")
    plt.ylim(-1, n_features)
plot_feature_importances_diabetes(dt)
plt.savefig('feature_importance')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [162], line 9
      7     plt.ylabel("Feature")
      8     plt.ylim(-1, n_features)
----> 9 plot_feature_importances_diabetes(dt)
     10 plt.savefig('feature_importance')

Cell In [162], line 5, in plot_feature_importances_diabetes(model)
      3 n_features = 13
      4 plt.barh(range(n_features), model.feature_importances_, align='center')
----> 5 plt.yticks(np.arange(n_features), X)
      6 plt.xlabel("Feature importance")
      7 plt.ylabel("Feature")

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\matplotlib\pyplot.py:1887, in yticks(ticks, labels, minor, **kwargs)
   1885         l._internal_update(kwargs)
   1886 else:
-> 1887     labels = ax.set_yticklabels(labels, minor=minor, **kwargs)
   1889 return locs, labels

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\matplotlib\axes\_base.py:73, in _axis_method_wrapper.__set_name__.<locals>.wrapper(self, *args, **kwargs)
     72 def wrapper(self, *args, **kwargs):
---> 73     return get_method(self)(*args, **kwargs)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\matplotlib\axis.py:1968, in Axis._set_ticklabels(self, labels, fontdict, minor, **kwargs)
   1966 if fontdict is not None:
   1967     kwargs.update(fontdict)
-> 1968 return self.set_ticklabels(labels, minor=minor, **kwargs)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\matplotlib\axis.py:1890, in Axis.set_ticklabels(self, ticklabels, minor, **kwargs)
   1886 if isinstance(locator, mticker.FixedLocator):
   1887     # Passing [] as a list of ticklabels is often used as a way to
   1888     # remove all tick labels, so only error for > 0 ticklabels
   1889     if len(locator.locs) != len(ticklabels) and len(ticklabels) != 0:
-> 1890         raise ValueError(
   1891             "The number of FixedLocator locations"
   1892             f" ({len(locator.locs)}), usually from a call to"
   1893             " set_ticks, does not match"
   1894             f" the number of ticklabels ({len(ticklabels)}).")
   1895     tickd = {loc: lab for loc, lab in zip(locator.locs, ticklabels)}
   1896     func = functools.partial(self._format_with_dict, tickd)

ValueError: The number of FixedLocator locations (13), usually from a call to set_ticks, does not match the number of ticklabels (270).
```



KNN

In [4]:
```
from sklearn.neighbors import KNeighborsClassifier
```

CHAPTER 8

TESTING

8.1 TEST CASES

Test case 1: Logging in with registered login details.

Test case 2: Logging in with invalid login details.

Test case 3: Registering with existing user's details.

Test case 4: Entering wrong values while filling medical related details.

Test case 5: Producing visualization for given input.

8.2 USER ACCEPTANCE TESTING

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button	Stable internet connection, Compatible browser, Login credentials	1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Signup popup displayed or not	localhost:3000/login	Login/Signup popup should display	Working as expected	Pass	Login page displayed successfully	N	1	Mukunthan M
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup	Proper code for UI elements, Elements position, Buttons and Textbox response	1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/signup popup with below UI elements: a.email text box b.password text box c.Login button d.New customer? Create account link e.Last password? Recovery password link	localhost:3000/login	Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link e.Last password? Recovery password link	Working as expected	Fail	Elements are displayed successfully but recovery password button is not present	N	2	Mukunthan M
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials	User credentials, Database with credentials of existing users	1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: test2022@gmail.com password: Testing123	User should navigate to user account homepage	Working as expected	Pass	Logged in successfully	N	3	Arivuselvan M
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with Invalid credentials	User credentials, Database with credentials of existing users	1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: test2022@gmail password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	Fail	Login failed due to incorrect login details or user not registered	N	4	Manoranjani S

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with Invalid credentials	User credentials, Database with credentials of existing users	1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: test2022@gmail.com password: Testing123676867867676876	Application should show 'Incorrect email or password ' validation message.	Working as expected	Fail	Login failed due to incorrect login details or user not registered	N	5	Manoranjani S
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with Invalid credentials	User credentials, Database with credentials of existing users	1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: test password: Testing123676867867676876	Application should show 'Incorrect email or password ' validation message.	Working as expected	Fail	Login failed due to incorrect login details or user not registered	N	6	Charulatha R
Dashboard_TC_001	Functional	Dashboard	Verify user is able to check accuracy of the results produced from the data	Dataset of medical reports	1.Enter the website and login 2.Upload the medical reports after logging in 3.Wait for the model to process 4.View accuracy of results in the dashboard 5.View visualisations of the results generated from medical records in visualisations section.	Username: test password: Testing123676867867676876	Application should show possibility of presence or absence of heart disease based on the medical reports given as input	Working as expected	Pass	Dashboard is successfully displayed	N	7	Arivuselvan M
Dashboard_TC_002	Functional	Dashboard	Verify user is able to view the graphs of the results produced from the data	Dataset of medical reports	1.Enter the website and login 2.Upload the medical reports after logging in 3.Wait for the model to process 4.View accuracy of results in the dashboard 5.View visualisations of the results generated from medical records in visualisations section.	Username: test password: Testing123676867867676876	Application should show visualisations of the medical results in the form of graphs in visualisations section	Working as expected	Pass	Visualisations of the results are displayed successfully	N	7	Charulatha R

CHAPTER 9

RESULTS

9.1 PERFORMANCE METRICS

1. Hours worked: 50 hours

2. Stick to Timelines: 100%

3. Stay within budget: 100%

4. Consistency of the product: 85%

5. Efficiency of the product: 85%

6. Quality of the product: 85%

CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- Smooth User Interface
- Accuracy is achieved quickly

DISADVANTAGES:

- Random forest can be used for both classification and regression tasks, but it is not more suitable for Regression tasks.

CHAPTER 11

CONCLUSION

This overview of the project conveys the idea that numerous methods have been investigated for diagnosing cardiovascular disease. Big data, machine learning, and data mining can be used to great success to analyse the prediction model with the highest degree of accuracy. The primary goal of this project is to diagnose cardiovascular disease or heart disease utilizing a variety of techniques and procedures to obtain a prognosis.

CHAPTER 12

FUTURE SCOPE

A future update shall comprise of section for viewing renowned cardiologists and scan centres in their city. The obtained output can be further processed and sent to smart devices to provide necessary assistance. Constant monitoring can provide necessary data to recommend to consult a doctor in case of an emergency.

CHAPTER 13

APPENDIX

PROJECT DEMONSTRATION LINK: https://youtu.be/Rk1Onr8IgWc

GITHUB LINK: https://github.com/IBM-EPBL/IBM-Project-51408-1660979127