

TABLE OF THE CONTENT

CHAPTER	CONTENTS	PAGE NO
1	INTRODUCTION	4
	1.1 PROJECT OVERVIEW	
	1.2 PURPOSE	
2	LITERATURE SURVEY	5
	2.1 EXISTING PROBLEM	
	2.2 REFERENCES	
	2.3 PROBLEM STATEMENT DEFINITION	
3	IDEATION AND PROPOSED SOLUTION	8
	3.1 EMPATHY MAP CANVAS	
	3.2 IDEATION AND BRAINSTORMING	
	3.3 PROPOSED SOLUTION	
	3.4 PROBLEM SOLUTION FIT	
4	REQUIREMENT ANALYSIS	10
	4.1 FUNCTIONAL REQUIREMENT	
	4.2 NON-FUNCTIONAL REQUIREMENT	
5	PROJECT DESIGN	12
	5.1 DATA FLOW DIAGRAM	
	5.2 SOLUTION AND TECHNICAL ARCHITECTURE	
	5.3 USER STORIES	
6	PROJECT PLANNING AND SCHEDULING	13
	6.1 SPRINT PLANNING AND ESTIMATION	
	6.2 SPRINT DELIVERY SCHEDULE	
	6.3 REPORTS FROM JIRA	

7	CODING AND SOLUTIONING	18
	7.1 FEATURE 1	
	7.2 FEATURE 2	
8	TESTING	23
	8.1 TEST CASES	
	8.2 USER ACCEPTANCE TESTING	
9	RESULTS	27
	9.1 PERFORMANCE METRICS	
10	ADVANTAGES AND DISADVANTAGES	30
11	CONCLUSION	32
12	FUTURE SCOPE	36
13	APPENDIX	38
14	SOURCE CODE	40
15	GITHUB AND PROJECT DEMO LINK	42

CHAPTER-1

INTRODUCTION

1.1 PROJECT OVERVIEW

Machine Learning is a way of Manipulating and extraction of implicit, previously unknown/known and potentially useful information about data. Machine Learning is a very vast and diverse field and its scope and implementation is increasing day by day. Machine learning Incorporates various classifiers of Supervised, Unsupervised and Ensemble Learning which are used to predict and Find the Accuracy of the given dataset. We can use that knowledge in our project of Heart Disease Prediction as it will help a lot of people. Cardiovascular diseases are very common these days, they describe a range of conditions that could affect your heart. The World health organization estimates that 17.9 million global deaths from (Cardiovascular diseases) CVDs . It is the primary reason for deaths in adults. Our project can help predict the people who are likely to be diagnosed with a heart disease by help of their medical history. It recognizes who all are having any symptoms of heart disease such as chest pain or high blood pressure and can help in diagnosing disease with less medical tests and effective treatments, so that they can be cured accordingly. This project focuses on mainly three data mining techniques namely: (1) Logistic regression, (2) KNN and (3) Random Forest Classifier. The accuracy of our project is 87.5% which is better than the previous system where only one data mining technique was used. So, using more data mining techniques increased the HDPS accuracy and efficiency. Logistic regression falls under the category of supervised learning. Only discrete values are used in logistic regression. The objective of this project is to check whether the patient is likely to be diagnosed with any cardiovascular heart diseases based on their medical attributes such as

gender, age, chest pain, fasting sugar level, etc. A dataset is selected from the Kaggle. By using this dataset, we predict whether the patient can have a heart disease or not. To predict this, we use 13 medical attributes of a patient and classify him if the patient is likely to have a heart disease. These medical attributes are trained under three algorithms: K Nearest Neighbour Classifier, Support Vector Classifier, Decision Tree Classifier and Random Forest Classifier. I varied parameters across each model to improve their scores. In the end, K Nearest Neighbors Classifier achieved the highest score of 87% with 8 nearest neighbors.

1.2 PURPOSE

As we all know, the heart is the most important part of our body other than the brain. It pumps blood through the blood vessels of the circulatory system. The circulatory system is extremely important because it transports blood, oxygen and other materials to the different organs of the body. Heart plays the most crucial role in the circulatory system. If the heart does not function properly then it will lead to serious health conditions including death. For having a healthy heart, there are many solutions available in the market. Exercise can also play an important role for maintaining heart health. Apart from medical treatments, technology can also prove to be very useful in treating any heart disease. Any heart disease is predicted beforehand, then curing it would be not much complex. But predicting it would be a tough task. Medical science has made excellent use of technological breakthroughs to raise the standard of healthcare. These technological developments have opened the path for precise illness diagnosis and prognosis. Machine learning might be a great option for you to obtain a high level of accuracy when it comes to forecasting heart illnesses with the help of algorithms.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

Before we did the experiments, we did research on how people explored heart disease prediction so that we can broaden our horizons and learn from them. In 2011, Ujma Ansari[1] made use of the Decision Tree model to predict heart disease and get a high accuracy of 99%, which inspires us to use a better version of Decision Tree and it is Random Forest. Unfortunately, the paper uses a dataset with 3000 instances but does not provide a reference of how they get the data. The UCI website only provides 303 instances of dataset so we doubt where the author gets 3000 instances of dataset. In 2012, Chaitrali S. Dangare [2] made the prediction by using three models and such models are Naïve Bayes, Decision Trees and Neural Network. We are using the same dataset as he did. The difference between his work and ours is that he added 2 more features into the dataset, which means there are 15 features of his work while there are 13 features in our dataset. Though there is no big difference between 13 features and 15 features in his work, what he did on dataset inspires us to make useful changes to our dataset (Try normalization on dataset) to make our results comprehensive. However, during this paper there are only 3 models. More models need to be considered so that the results are comprehensive. In 2017, Kaan Uyar and Ahmet İlhan[3] did the same experiment and used the same dataset as we did for projects. During their analysis, “Class distributions are interpreted as 54% absence and 46% presence of a heart disease”. The dataset we download from Kaggle has 54% 1s and 46% 0s in the target column. From their analysis, we realize 1 indicates absence of heart disease and vice versa. To make it easily

understood, we switched 1s and 0s in the target column so that 1 indicates presence of heart disease to show our confusion matrix[10] in our results. After reviewing paper [4] and [5], we have learned that a neural network has the advantage of fault tolerance and it has the ability to work with inadequate knowledge as human beings. Therefore, in our project we decide to spend some time working on neural network to detect heart diseases

2.2 REFERENCES

https://en.wikipedia.org/wiki/Cardiovascular_disease.

http://www.heart.org/HEARTORG/Conditions/HeartAttack/WarningSignsofaHeartAttack/Warning-Signs-of-aHeartAttack_UCM_002039_Article.jsp#.

WNpKgPl97IU.

www.who.int/cardiovascular_diseases/en/.

<http://food.ndtv.com/health/world-heart-day-2015-heart-disease-in-india-is-agrowing-concern-ansari-1224160>.

2.3 PROBLEM STATEMENT DEFINITION

Visualizing And Predicting Heart Diseases With An Interactive Dashboard

CHAPTER-3

PROBLEM DEFINITION

3.1 PROBLEM STATEMENT

Visualizing And Predicting Heart Diseases With An Interactive Dashboard

3.2 EXISTING SYSTEM

In this system, the input details are obtained from the patient. Then from the user inputs, using ML techniques heart disease is analyzed. Now, the obtained results are compared with the results of existing models within the same domain and found to be improved. The data of heart disease patients collected from the UCI laboratory is used to discover patterns with NN, DT, Support Vector machines SVM, and Naive Bayes. The results are compared for performance and accuracy with these algorithms. The proposed hybrid method returns results of 87% for F-measure, competing with the other existing methods

3.2.1 DISADVANTAGE OF EXISTING SYSTEM

Prediction of cardiovascular disease results is not accurate.

Data mining techniques do not help to provide effective decision making.

Cannot handle enormous datasets for patient records.

3.3 PROPOSED SYSTEM

After evaluating the results from the existing methodologies, we have used python and pandas operations to perform heart disease classification for the data obtained from the UCI repository. It provides an easy-to-use visual representation of the dataset, working environment and building the predictive analytics. ML process starts from a preprocessing data phase followed by feature selection based on data cleaning, classification of modeling performance

evaluation. Random forest technique is used to improve the accuracy of results.

3.3.1 ADVANTAGE OF THE PROPOSED SYSTEM:

Increased accuracy for effective heart disease diagnosis.

Handles roughest(enormous) amount of data using random forest algorithm and feature selection.

Reduce the time complexity of doctors.

Cost effective for patients.

CHAPTER-4

REQUIREMENT

4.1 TOOL REQUIREMENTS

4.1.1 TOOLS USED

FRONT END :

CSS

BACK END :

PYTHON

DEVELOPMENT TOOLS :

VISUAL STUDIO

OPERATING SYSTEM :

Windows 7, Windows 8, Windows 10, Ubuntu

4.1.2 MINIMUM REQUIREMENTS OF THE SYSTEM

Processor : Intel Pentium III

RAM : minimum 4 GB

HDD : 40 GB

Secondary Storage : 1.44 MB FDD, CD-R, CD+RW CD

Monitor : 14" Color Monitor.

4.2 FUNCTIONAL REQUIREMENTS

- Every online application needs to be associated with an account.
- Hospital Users have to register their details such as hospital name, departments available in the hospital in order to make use of this application.
- Function to set and edit the address details, bed availability ,doctor details, contact details, Doctor timing can be done by both the admin and the registered hospital users.

4.3 NON-FUNCTIONAL REQUIREMENTS

4.3.1. Performance

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design an application, which will fit into the required environment. The load for the user interface screens shall take no longer than 2 seconds. The login information shall be verified within 5 seconds. Queries shall return results within 5 seconds.

4.3.2 Safety and Security Requirements

- User Identification:

The system requires the Dashboard user to identify himself/herself as a user

- Login ID:

Any Dashboard user who uses the system shall have a Login.

- Modification:

Any modification (insert, delete (or) update) for the Database shall be

CHAPTER-5

MODULES DESCRIPTION

5.1 USER MODULE

In this module we deal with the searching of disease level based on the constraints provided by the user.

5.2 ADMIN FEATURES

- Admin have their respective user id & password for authenticated login
- They are able to create a new Dashboard User.

5.2 DASHBOARD USER FEATURES

- Hospital user have their respective user id & password for authenticated login
- They are able to view, add and update the hospital details along with the user details.

5.3 USER FEATURES

The user is able to search for the disease by providing the information.

5.3.1 Pre-Condition

- The system displays the menu of available functions that are detailed information.
- This system displays the menu of the admin panel.

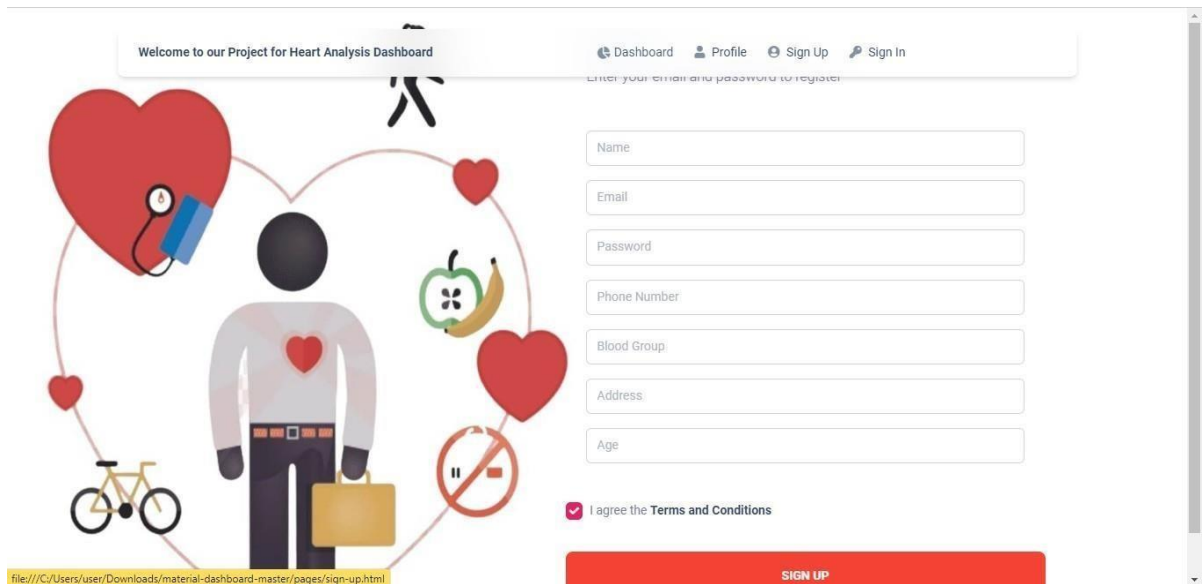
5.3.2 Post-Condition

- Login is successful.

- Whole system is more flexible for users to use.

USER INTERFACE

Registration-Can register for the application by entering my email, password, and confirming my password



Welcome to our Project for Heart Analysis Dashboard

Dashboard Profile Sign Up Sign In

Enter your email and password to register

Name

Email

Password

Phone Number

Blood Group

Address

Age

☒ I agree the Terms and Conditions

SIGN UP

file:///C:/Users/user/Downloads/material-dashboard-master/pages/sign-up.html

Welcome to our Project for Heart Analysis Dashboard

Dashboard Profile Sign Up Sign In

Enter your email and password to register

IBM-TEAM

test@test.com

1234567890

A1b+

Address

Age

☒ I agree the Terms and Conditions

SIGN UP

Login - Can't log into the application by entering email/PhoneNumber & password

Welcome to Our Project for Heart analysis Dashboard

Dashboard Profile Sign Up Sign In

Sign in

f G

Email/PhoneNumber

Password

☐ Remember me

SIGN IN

Don't have an account? [Sign up](#)

Welcome to Our Project for Heart analysis Dashboard

Dashboard Profile Sign Up Sign In

Sign in

f G

test@test.com

☐ Remember me

SIGN IN

Don't have an account? [Sign up](#)

Welcome to Our Project for Heart analysis Dashboard

Dashboard Profile Sign Up Sign In

Sign in

f G

1234567890

☐ Remember me

SIGN IN

Don't have an account? [Sign up](#)

Heart Disease Predictor

A Deep Learning Web App, Built with ANN and Flask.

20
Sex
Chest pain type
10
100
Fasting blood sugar > 120 mg/dl
Resting electrocardiographic results
150
Exercise induced angina
Depression induced by exercise relative to rest eg.1-5
The slope of the peak exercise ST segment
Number of major vessels (0-3) colored by flourosopy
Thalassemia

Predict

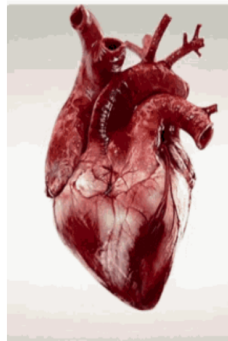


Heart Disease Predictor

A Deep Learning Web App, Built with Ann and Flask.

Oops! Your have Heart condition is SEVERE.

Please consult Doctor.Stay Healthy Mate!



Heart Disease Predictor

A Deep Learning Web App, Built with ANN and Flask.

20

Sex

Chest pain type

10

10

Fasting blood sugar > 120 mg/dl

Resting electrocardiographic results

10

Exercise induced angina


3

The slope of the peak exercise ST segment

Number of major vessels (0-3) colored by flourosopy

Thalassemia

Predict



Heart Disease Predictor

A Deep Learning Web App, Built with Ann and Flask.

Wooh! Your Heart looks NORMAL.

Long Live!



CHAPTER-7

IMPLEMENTATIO

7.1 FRONT END

CSS

BOOTSTRAP

7.1.1 FEATURES

FEATURES OF CSS:

CSS can define color, font, text alignment, size, borders, spacing, layout and many other typographic characteristics, and can do so independently for on-screen and printed views.

CSS also defines non-visual styles, such as reading speed and emphasis for aural text readers.

FEATURES OF BOOTSTRAP:

Bootstrap has a lot of features. These features not only make it stand out, but they also make it more popular even among those web designers who like to take things in a very conventional way.

✓ EASY TO BEGIN WITH

It is pretty easy, to begin with. Being easy to get started with is probably the first quality which makes Bootstrap very appealing.

✓ LESS AS WELL AS CSS FILES

Bootstrap not only offers LESS files but also includes the old CSS files.

✓ EASILY CUSTOMIZABLE

Despite the fact that Bootstrap is designed in responsive 12-column grids, layouts, and components, it is also very easy to customize. Whether you need a fixed grid or a responsive one, it can be made possible by making a few changes. Offsetting and nesting of columns are also easy to do in both CPU-based and mobile-based browser grids.

✓ RESPONSIVE UTILITY CLASSES

Another prominent feature of Bootstrap is its responsive utility classes. Using responsive utility classes, a particular piece of content can be made to appear or hide only on devices depending on the size of the screen being used. This feature is extremely helpful for designers who want to make a mobile and tablet-friendly version of their websites.

✓ COMPONENTS OF BOOTSTRAP

Some of the components that come pre-styled in Bootstrap are:

✓ DROP-DOWNS

Button
Navigation
Badges Alerts
Progress Bar

✓ DROP-DOWN COMPONENT MENU

The drop-down component menu is a responsive additive feature of a website. To include it in a website, a lot of different plugins, mostly Java-based, are tested. But, via Bootstrap and its easy customization options, this can easily be done in a matter of minutes.

✓ BOOTSTRAP TEMPLATES

The readily available templates make it easier for inexperienced users to create a website following a simple tutorial or demo available on the Bootstrap.

7.1.2 FRONT END CODE

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@100;200;300;400;500;600&display=swap');

*{
  font-family: 'Poppins', sans-serif;
  margin:0; padding:0;
  box-sizing: border-box;
  outline: none; border:none;
  text-decoration: none;
}

.container{
  min-height: 100vh;
```

```
display: flex;
align-items: center;
justify-content: center;
padding: 20px;
padding-bottom: 60px;
}
```

```
.container .content{
  text-align: center;
}
```

```
.container .content h3{
  font-size: 30px;
  color: #333;
}
```

```
.container .content h3 span{
  background: crimson;
  color: #fff;
  border-radius: 5px;
  padding: 0 15px;
}
```

```
.container .content h1{
  font-size: 50px;
  color: #333;
}
```

```
.container .content h1 span{
  color: crimson;
}
```

```
.container .content p{
  font-size: 25px;
  margin-bottom: 20px;
}
```

```
.container .content .btn{
  display: inline-block;
  padding: 10px 30px;
  font-size: 20px;
  background: #333;
  color: #fff;
}
```

```
margin:0 5px;
text-transform: capitalize;
}
```

```
.container .content .btn:hover{
background: crimson;
}
```

```
.form-container{
min-height: 100vh;
display: flex;
align-items: center;
justify-content: center;
padding:20px;
padding-bottom: 60px;
background: #eee;
}
```

```
.form-container form{
padding:20px;
border-radius: 5px;
box-shadow: 0 5px 10px rgba(0,0,0,.1);
background: #fff;
text-align: center;
width: 500px;
}
```

```
.form-container form h3{
font-size: 30px;
text-transform: uppercase;
margin-bottom: 10px;
color:#333;
}
```

```
.form-container form input,
.form-container form select{
width: 100%;
padding:10px 15px;
font-size: 17px;
margin:8px 0;
background: #eee;
border-radius: 5px;
}
```



```

else{
    x.type = "password";
}
}

```

```

function validate(){
    var password = document.getElementById("pass");
    var length = document.getElementById("length");

    if(password.value.length >= 8){
        alert("Login Successful");
        window.location.replace("heart .html");
        return false;
    }
    else{
        alert("Login Failed");
    }
}

```

```

function page(){
    window.location.replace("Landingpage.html")
}

```

```

window.watsonAssistantChatOptions = {
    integrationID: "3f5d7446-04cb-4796-8496-1351f8d8acfd", // The ID of this integration.
    region: "au-syd", // The region your integration is hosted in.
    serviceInstanceID: "58af1a9a-a26e-4a48-ae6f-0881b0be4c0d", // The ID of your service
instance.
    onLoad: function(instance) { instance.render(); }
};
setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
});
function myfunction(){
    var x =document.getElementById("pass");

    if(x.type === "password"){

```

```

        x.type = "text";
    }
    else{
        x.type = "password";
    }
}

function validate(){
    var password = document.getElementById("pass");
    var length = document.getElementById("length");

    if(password.value.length >= 8){
        alert("Login Successful");
        window.location.replace("heart .html");
        return false;
    }
    else{
        alert("Login Failed");
    }
}

```

```

function page(){
    window.location.replace("Landingpage.html")
}

```

BACK END CODE:

```

#!/usr/bin/env python
# coding: utf-8

```

```

# In[2]:

```

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

```



```
# In[4]:
```

```
df=pd.read_csv('Downloads/Heart_Disease_Prediction.csv')
```

```
# In[5]:
```

```
df.head()
```

```
# In[6]:
```

```
df.isnull().sum()
```

```
# In[7]:
```

```
print(df.info())
```

```
# In[9]:
```

```
plt.figure(figsize=(20,10))  
sns.heatmap(df.corr(), annot=True, cmap='terrain')
```

```
# In[10]:
```

```
sns.pairplot(data=df)
```

```
# In[11]:
```

```
df.hist(figsize=(10,12), layout=(5,4));
```

```
# In[13]:
```

```
df.plot(kind='box', subplots=True, layout=(6,3), figsize=(10,10))  
plt.show()
```

```
# In[19]:
```

```
sns.catplot(data=df, x='Sex', y='Age', hue='Heart Disease', palette='tab10')
```

```
# In[20]:
```

```
sns.barplot(data=df, x='Sex', y='Cholesterol', hue='Heart Disease', palette='spring')
```

```
# In[21]:
```

```
df['Sex'].value_counts()
```

```
# In[22]:
```

```
df['Chest pain type'].value_counts()
```

```
# In[23]:
```

```
sns.countplot(x='Chest pain type', hue='Heart Disease', data=df, palette='rocket')
```



```
# In[24]:
```

```
gen = pd.crosstab(df['Sex'], df['Heart Disease'])  
print(gen)
```

```
# In[25]:
```

```
gen.plot(kind='bar', stacked='True', color=['green','blue'],grid=False)
```

```
# In[ ]:
```

```
# In[ ]:
```

```
#!/usr/bin/env python  
# coding: utf-8
```

```
# In[1]:
```

```
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import warnings  
warnings.filterwarnings('ignore')
```

```
# In[2]:
```

```
df=pd.read_csv('Downloads/Heart_Disease_Prediction.csv')
```

```
# In[3]:
```

```
df.head()
```

```
# In[4]:
```

```
df.isnull().sum()
```

```
# In[5]:
```

```
print(df.info())
```

```
# In[6]:
```

```
plt.figure(figsize=(20,10))  
sns.heatmap(df.corr(), annot=True, cmap='terrain')
```

```
# In[7]:
```

```
sns.pairplot(data=df)
```

```
# In[8]:
```

```
df.hist(figsize=(10,12), layout=(5,4));
```

```
# In[9]:
```

```
df.plot(kind='box', subplots=True, layout=(6,3), figsize=(10,10))  
plt.show()
```

```
# In[10]:
```

```
sns.catplot(data=df, x='Sex', y='Age', hue='Heart Disease', palette='tab10')
```

```
# In[11]:
```

```
sns.barplot(data=df, x='Sex', y='Cholesterol', hue='Heart Disease', palette='spring')
```

```
# In[12]:
```

```
df['Sex'].value_counts()
```

```
# In[13]:
```

```
df['Chest pain type'].value_counts()
```

```
# In[14]:
```

```
sns.countplot(x='Chest pain type', hue='Heart Disease' , data=df, palette='rocket')
```

```
# In[15]:
```

```
gen = pd.crosstab(df['Sex'], df['Heart Disease'])
print(gen)
```

```
# In[16]:
```

```
gen.plot(kind='bar', stacked='True', color=['green','blue'],grid=False)
```

```
# In[17]:
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
StandardScaler = StandardScaler()
columns_to_scale=['Age', 'EKG results', 'Cholesterol', 'Thallium', 'Number of vessels fluro']
df[columns_to_scale] = StandardScaler.fit_transform(df[columns_to_scale])
```

```
# In[18]:
```

```
df.head()
```

```
# In[19]:
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
StandardScaler = StandardScaler()
columns_to_scale=['Age', 'EKG results', 'Cholesterol', 'Thallium', 'Number of vessels fluro']
df[columns_to_scale] = StandardScaler.fit_transform(df[columns_to_scale])
```

```
# In[20]:
```

```
df.head()
```

```
# In[21]:
```

```
x=df.drop(['Heart Disease'], axis=1)
```

```
y=df['Heart Disease']
```

```
# In[22]:
```

```
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.3, random_state=40)
```

```
# In[23]:
```

```
print('x_train-', x_train.size)
print('x_test-', x_test.size)
print('y_train-', y_train.size)
print('x_test-', x_test.size)
```

```
# In[24]:
```

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
model1=lr.fit(x_train,y_train)
prediction1=model 1.predict(x_test)
```

```
# In[25]:
```

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,prediction1)
cm
```

```
# In[26]:
```

```
sns.heatmap(cm, annot=True,cmap='BuPu')
```

```
# In[27]:
```

```
TP=cm[0][0]
TN=cm[1][1]
FN=cm[1][0]
FP=cm[0][1]
print('Testing Accuracy:', (TP+TN+FN)/(TP+TN+FN+FP))
```

```
# In[28]:
```

```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(y_test,prediction1)
l=accuracy_score(y_test,prediction1)
```

```
# In[29]:
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, prediction1))
```

```
# In[30]:
```

```
import pandas as pd
from sklearn import neighbors,metrics
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
import pickle
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# In[31]:
```

```
from sklearn.metrics import accuracy_score
```

```
# In[32]:
```

```
dataset = pd.read_csv("Downloads/Heart_Disease_Prediction.csv")
```

```
# In[33]:
```

```
KX = dataset[['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']].values
```

```
# In[34]:
```

```
KY = dataset[['Heart Disease']].values
```

```
# In[35]:
```

KX

In[36]:

```
KY = KY.flatten()
print(KY)
```

In[37]:

```
KX_train , KX_test , KY_train , KY_test =
train_test_split(KX,KY,test_size=0.2,random_state=4)
```

In[38]:

```
knn = KNeighborsClassifier(n_neighbors = 20)
knn.fit(KX_train, KY_train)
print(knn.score(KX_test, KY_test))
```

In[39]:

```
pickle.dump(knn,open('heart_knn_model.sav','wb'))
```

In[40]:

```
predict_knn = knn.predict(KX_test)
accuracy_knn = metrics.accuracy_score(KY_test,predict_knn)
```

In[41]:

```
predict_knn
```

In[42]:

```
accuracy_knn
```

In[43]:

```
k=accuracy_knn
```

In[45]:

```
import csv
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix, f1_score, roc_curve, auc
import matplotlib.pyplot as plt
from itertools import cycle
from scipy import interp
```

```
# In[46]:
```

```
df = pd.read_csv('Downloads/Heart_Disease_Prediction.csv', header = None)
```

```
# In[47]:
```

```
training_x=df.iloc[1:df.shape[0],0:13]
```

```
# In[48]:
```

```
training_y=df.iloc[1:df.shape[0],13:14]
```

```
# In[49]:
```

```
nx=np.array(training_x)
ny=np.array(training_y)
```

```
# In[52]:
```

```
for z in range(5):
    print("\nTest Train Split no. ",z+1,"\n")
    nx_train,nx_test,ny_train,ny_test = train_test_split(nx,ny,test_size=0.25,random_state=None)
    # Gaussian function of sklearn
    gnb = GaussianNB()
    gnb.fit(nx_train, ny_train.ravel())
    ny_pred = gnb.predict(nx_test)
```

```
# In[61]:
```

```
print("\n Naive Bayes model accuracy(in %):", metrics.accuracy_score(ny_test, ny_pred))
```

```
# In[62]:
```

```
n=metrics.accuracy_score(ny_test, ny_pred)
```

```
# In[64]:
```

```
import pandas as pd
from sklearn import neighbors,metrics
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
import pickle
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# In[65]:
```

```
from sklearn.metrics import accuracy_score
```

```
# In[67]:
```

```
dataset = pd.read_csv("Downloads/Heart_Disease_Prediction.csv")
```

```
# In[69]:
```

```
DX = dataset[['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']].values
```

```
# In[70]:
```

```
dy = dataset[['Heart Disease']].values
```

```
# In[71]:
```

```
DX
```

```
# In[72]:
```

```
dy = dy.flatten()
```



```
print(dy)
```

```
# In[73]:
```

```
DX_train , DX_test , dy_train , dy_test = train_test_split(DX,dy,test_size=0.2,random_state=4)
```

```
# In[74]:
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
max_accuracy = 0
```

```
# In[75]:
```

```
for x in range(200):  
    dt = DecisionTreeClassifier(random_state=x)  
    dt.fit(DX_train,dy_train)  
    dy_pred_dt = dt.predict(DX_test)  
    current_accuracy = round(accuracy_score(dy_pred_dt,dy_test)*100,2)  
    if(current_accuracy>max_accuracy):  
        max_accuracy = current_accuracy  
        best_x = x
```

```
# In[85]:
```

```
dt = DecisionTreeClassifier(random_state=best_x)  
dt.fit(DX_train,dy_train)  
dy_pred_dt = dt.predict(DX_test)
```

```
# In[88]:
```

```
score_dt = (accuracy_score(dy_pred_dt,dy_test))
```

```
# In[89]:
```

```
print("The accuracy score achieved using Decision Tree is: "+str(score_dt))
```

```
# In[90]:
```

```
d=(accuracy_score(dy_pred_dt,dy_test))
```

```
# In[91]:
```

```
print('Logistic Regression :',l)
print('KNN :',k)
print('Naive Bayes :',n)
print('Decision Tree :',d)
```

```
# In[93]:
```

```
print('Logistic Regression :',l*100,'%')
print('KNN :',k*100,'%')
print('Naive Bayes :',n*100,'%')
print('Decision Tree :',d*100,'%')
```

```
# In[ ]:
```

```
import js2py
import os
from flask import Flask,request,jsonify, json, Response, make_response, render_template
from flask_pymongo import PyMongo
from flask_bcrypt import Bcrypt
from flask_cors import CORS
import db
```

```
js2py.run_file(bot.js)
```

```
app = Flask(__name__)
bcrypt = Bcrypt(app)
CORS(app)
```

```
@app.route('/')

```

```
@app.route("/test")

```

```
def test():
    return "Connected to the database!"
```

```
class UserAuthUtil:
```

```
    @app.route("/", methods=['GET'])
    def hello_world():
        return "Working"
```

```
    @app.route("/login", methods=['POST'])
    def login_user():
        try:
            if request.method == 'POST':
                form_data = request.get_json()
```

```

email = form_data['email']
password = form_data['password']
if(email != "" and password != ""):
    data = list(db.users.find({'email': email}))
    if(len(data) == 0):
        return Response(status=404, response=json.dumps({'message': 'user does not exist'}), mimetype='application/json')
    else:
        data = data[0]
        if(bcrypt.check_password_hash(data['password'], password)):
            #token = jwt.encode({'email': email}, app.config['SECRET_KEY'])
            return make_response(jsonify({'message': 'User logged in successfully'}), 201)
        else:
            return Response(status=402, response=json.dumps({'message': 'Invalid password'}), mimetype='application/json')
    else:
        return Response(status=400, response=json.dumps({'message': 'Bad request'}), mimetype='application/json')
    else:
        return Response(status=401, response=json.dumps({'message': 'invalid request type'}), mimetype='application/json')
except Exception as Ex:
    print("\n\n\n*****")
    print(Ex)
    print("*****\n\n\n")
    return Response(response=json.dumps({'message': "Internal Server error"}), status=500, mimetype="application/json")

```

```

@app.route("/register", methods=['POST'])
def register_user():
    try:
        if request.method == "POST":
            user_details = request.get_json()
            full_name = user_details["fullName"]
            email = user_details["email"]
            password = user_details["password"]
            password_hash = bcrypt.generate_password_hash(password).decode('utf-8')
            if (full_name != "" and email != "" and password_hash != ""):
                db.users.insert_one({'fullName': full_name, 'email': email, 'password': password_hash})
                return Response(response=json.dumps({'message': 'User created successfully'}), status=200, mimetype="application/json")
            else:
                return Response(status=400, response=json.dumps({'message': 'Please enter your details'}), mimetype='application/json')
    except:
        return Response(status=400, response=json.dumps({'message': 'Please enter your details'}), mimetype='application/json')

```

```

else:
    return Response(status=400, response=json.dumps( {'message': 'Bad request'}),
mimetype='application/json')
except Exception as Ex:
    print("\n\n\n*****")
    print(Ex)
    print("*****\n\n\n")
    return Response(response=json.dumps( {'message': "Internal Server Error"}), status=500,
mimetype="application/json")

if __name__ == '__main__':
    app.run(port=8000)

```

7.2 **BACK END**

Python

7.3.2 **FEATURES**

- Self-Contained
- Zero-configuration
- Transactional
- Single Database File

CHAPTER-8

TESTING

8.1 INTRODUCTION

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive. A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

8.2 STRATEGIC APPROACH TO SOFTWARE TESTING

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn. A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Taking another turn outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system 37 elements are tested as a whole

8.3 UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

8.3.1 WHITE BOX TESTING

To follow the concept of white box testing we have tested each form.

We have created independently to verify that Data flow is correct and all conditions are exercised to check their validity.

- All loops are executed on their boundaries. This type of testing ensures that all independent paths have been exercised at least once
- All logical decisions have been exercised on their true and false sides
- All loops are executed at their boundaries and within their operational bounds
- All internal data structures have been exercised to assure their validity paths.

8.3.2 CONDITIONAL TESTING

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested so that each path that may generate on a particular condition is traced to uncover any possible errors.

8.3.3 DATA FLOW TESTING

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variables were declared. The definition-use chain method was used in this type of testing.

LOOP TESTING

In this type of testing all the loops are tested to all the limits possible.

The following exercise was adopted for all loops:

- All the loops were tested at their limits, just above them and just below them. All the loops were skipped at least once.
- For nested loops, test the innermost loop first and then work outwards. For concatenated loops the values of dependent loops were set with the help of connected loops.
- Unstructured loops were resolved into nested loops or concatenated loops and tested as above.
- Each unit has been separately tested by the development team itself and all the inputs have been validated.


8.4 TEST CASE

A test case, in software engineering, is a set of conditions or variables under which a tester will determine whether an application, software system or one of its features is working as it was originally established for it to do. The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle. In some settings, an oracle could be a requirement or use case, while in others it could be a heuristic. It may take many test cases to determine that a software program or system is considered sufficiently scrutinized to be released. Test cases are often referred to as test scripts, particularly when written - when they are usually collected into test suites.

CHAPTER-9

9.1SCREENSHOTS





CHAPTER-10

CONCLUSION AND FUTURE WORK

10.1 CONCLUSION

The proposed Heart Disease Predict System has been implemented in Visual Studio as a web application. The tasks involved in this work are divided into modules. The proposed system is efficient and has a friendly user interface.

10.2 FUTURE WORK

Additional authentication protocols and security tests added to the system would make the system more secure.

REFERENCES

- 1[1] A. S. Abdullah and R. R. Rajalaxmi, “A data mining model for predicting

the coronary heart disease using random forest classifier,” in Proc. Int. Conf. Recent Trends Comput. Methods, Commun. Controls, Apr. 2012, pp. 22–25.

[2] A. H. Alkeshuosh, M. Z. Moghadam, I. Al Mansoori, and M. Abdar, “Using PSO algorithm for producing best rules in diagnosis of heart disease,” in Proc. Int. Conf. Comput. Appl. (ICCA), Sep. 2017, pp. 306–311.

[3] N. Al-milli, “Back Propagation neural network for prediction of heart disease,” J. Theor. Appl. Inf. Technol., vol. 56, no. 1, pp. 131–135, 2013.

[4] C. A. Devi, S. P. Rajamohanan, K. Umamaheswari, R. Kiruba, K. Karunya, and R. Deepika, “Analysis of neural networks based heart disease prediction system,” in Proc. 11th Int. Conf. Hum. Syst. Interact. (HSI), Gdansk, Poland, Jul. 2018, pp. 233–239.

[5] P. K. Anooj, “Clinical decision support system: Risk level prediction of heart disease using weighted fuzzy rules,” J. King Saud Univ.- Comput. Inf. Sci., vol. 24, no. 1, pp. 27–40, Jan. 2012. doi:10.1016/j.jksuci.2011.09.002.

[6] L. Baccour, “Amended fused TOPSIS-VIKOR for classification (ATOVIC) applied to some UCI data sets,” Expert Syst. Appl., vol. 99, pp. 115–125, June. 2018. doi: 10.1016/j.eswa.2018.01.025.

[7] C.-A. Cheng and H.-W. Chiu, “An artificial neural network model for the evaluation of carotid artery stenting prognosis using a national-wide database,” in Proc. 39th Annual. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC), Jul. 2017, pp. 2566–2569.