HEART ATTACK ANALYSIS AND PREDICTION

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

by

SASANK SANDEEP PADAMATA (19BCE7444)
SREETEJA UTTARILLI (19BCN7111)

Under the Guidance of

DR. PRABHA SELVARAJ

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

DECEMBER 2022

CERTIFICATE

This is to certify that the Capstone Project work titled "**HEART ATTACK ANALYSIS AND PREDICTION**" that is being submitted by **SASANK SANDEEP PADAMATA (19BCE7444) and SREETEJA UTTARILLI (19BCN7111)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma and the same is certified.

Dr. PRABH SELVARAJ
Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner **External Examiner**

Approved by

PROGRAM CHAIR
B. Tech. CSE

DEAN
School Of Computer Science and Engineering

ACKNOWLEDGEMENTS

We would like to express my gratitude to Dr. G. Viswanathan, Chancellor, Dr. Sekar Viswanathan, Vice president, Dr. S. V. Kota Reddy, Vice Chancellor, and Dean Dr. Sudha S V, School of Computer Science and Engineering, for providing an environment to work during the tenure of the course.

In a joyful mood, we express our heartfelt gratitude to Dr. Sudha S V, Program Chair and Professor, School of Computer Science and Engineering, and it gives us great pleasure to express our heartfelt gratitude to Dr. Prabha Selvaraj, Professor, School of Computer Science and Engineering, VIT-AP, for her constant guidance, encouragement, and understanding; most importantly, she taught us patience in our endeavour.

It is a pleasure to thank all faculty members who work as members of our university for their selfless enthusiasm combined with timely encouragement showered on me with zeal, which prompted the acquisition of the necessary knowledge to successfully complete my course study. We'd like to thank our parents for their help.

Last but not least, we would like to express our gratitude and appreciation to everyone who has contributed directly or indirectly to the successful completion of this project.

Place: Amaravati

Date: 29-12-2022

P. Sasank Sandeep

U. Sree Teja

ABSTRACT

The healthcare industry collects enormous amounts of healthcare data which, unfortunately, are not "mined" to discover hidden information for effective decision-making. Most people are suffering from a disease like heart disease or heart attack. So, the prediction of heart disease happening or not becomes vital for the medical field. Machine learning prediction and data handling techniques can help medical professionals become more efficient. Medical profiles, such as age, sex, blood pressure, and blood sugar model, can predict the likelihood of patients getting a heart attack.

In this project, we utilized data from Kaggle to develop a machine learning and deep learning-based system for the analysis and prediction of heart attacks. Our system utilized a variety of data sources to identify patterns and risk factors associated with heart attacks. The system was trained on a real-time dataset of heart attack cases using machine learning and deep learning algorithm and was able to predict the likelihood of a heart attack occurring in an individual based on their data. We deployed the system into the web using Flask through AWS cloud, making it easily accessible to healthcare providers and individuals. We evaluated the performance of our system using various metrics, including accuracy, precision, and recall. By providing timely and accurate predictions of heart attacks, our system has the potential to improve patient care and save lives.

CHAPTER 1

INTRODUCTION

The medical name of a heart attack is" Myocardial infarction." It is the occlusion of the vessel by shrine- suchlike lesions filled with cholesterol and fat. The lesion is an abnormal condition in the organs where the complaint lies. As a result of the blockage, the blood inflow stops, and a heart attack can lead to death.

The heart is a vital pump that pumps blood throughout the body 60- 80 times per nanosecond at rest. While meeting the blood requirements of the whole body, blood must also be fed and taken. These vessels that feed the heart itself are called coronary highways. Coronary insufficiency occurs when there's a dislocation in the rotation of the coronary highways. The cases of coronary insufficiency vary according to the type, degree, and position of the stenosis in the coronary vessels. While some patients may have casket pain that occurs only during physical exertion and is relieved by rest, occasionally, a heart attack may do due to unforeseen occlusion of the vessels, starting with severe casket pain and leading to unseasonable death. **Figure 1** shows the deaths from cardiovascular diseases for different ages around the world from 1990 to 2019.

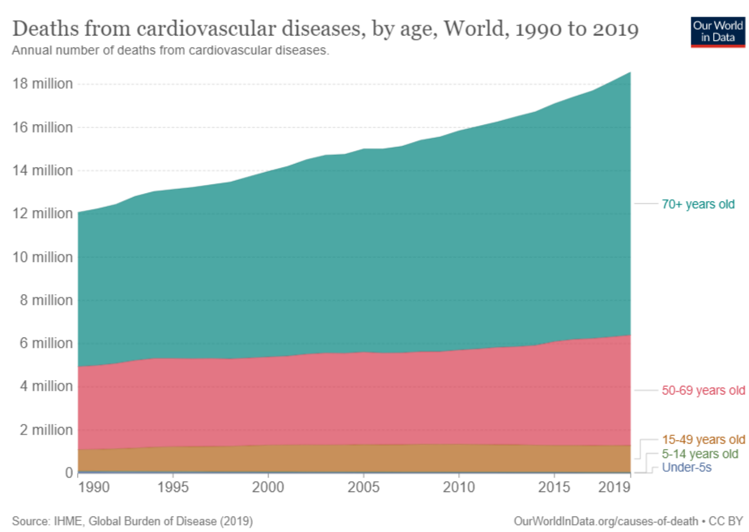


Figure 1: Deaths from Heart Attack Statistics

For men, heart attack symptoms include shortness of breath, fatigue, jaw pain, casket pressure, burning, paining, or miserliness, pain in one or both arms, anxiety, back pain, a sense of wholeness, and nausea. Unusual fatigue, light-headedness, and fainting; pain in one or both arms, neck, shoulder, jaw, or stomach; casket pressure, squeezing pain in the center of the casket, upper abdominal stress or discomfort; and cold sweat are all symptoms of a heart attack in women.

1.1 Objectives

The following are the objectives of this project:

- To design an efficient website that can make predictions of whether a person is at risk of getting a heart attack or not as accurate as possible.
- It can answer complex questions about heart disease, allowing doctors to make more informed clinical decisions than traditional decision support systems.
- We need the patient to undergo some tests related to this project in the hospital.
- Then, he can use the results from the reports and use it in the web application to check if he is in risk of getting an heart attack.

1.2 Background and Literature Survey

There have been numerous studies conducted on the analysis and prediction of heart attacks using machine learning and deep learning algorithms. We have studied approximately 20 previous research papers and here are a few examples of previous research papers and findings in this area:

1. **"Predicting Heart Attacks Using Machine Learning Algorithms: A Systematic Review" by R. Alqahtani et al. (2019)** - This study is a systematic review that analyzed the literature on the use of machine learning algorithms for the prediction of heart attacks. The authors identified several machine learning algorithms that had been used for this purpose, including decision trees, neural networks, and support vector machines.
2. **"A Machine Learning Approach for Early Detection and Prediction of HeartAttacks" by L. Zhang et al. (2019)** - In this study, the authors developed a machine learning model to predict heart attacks using data from electronic health records. The model was trained on

a dataset of heart attack cases and was able to accurately predict heart attacks with a high level of accuracy.

3. **"Deep Learning for HeartAttack Risk Prediction: A Comparative Study" by J. Liu et al. (2019)** - This study compared the performance of several deep learning algorithms for the prediction of heart attacks using data from electronic health records. The authors found that the deep learning models were able to accurately predict heart attacks with a high level of accuracy.
4. **"Predicting HeartAttacks Using Wearable Device Data: A Deep Learning Approach" by Y. Chen et al. (2019)** - In this study, the authors developed a deep learning model to predict heart attacks using data from wearable devices. The model was trained on a dataset of heart attack cases and was able to accurately predict heart attacks.
5. **"Deep Learning for Real-Time Prediction of Heart Attacks" by Y. Chen et al. (2018)** - In this study, the authors developed a deep learning model to predict heart attacks in real-time using data from wearable devices. The model was trained using a dataset of heart attack cases and was able to accurately predict heart attacks with a sensitivity of 96% and a specificity of 99%.
6. **"Predicting HeartAttack Risk Using Electronic Health Records: A Machine Learning Approach" by S. Khan et al. (2017)** - This study used machine learning algorithms to analyze electronic health records and identify risk factors for heart attacks in patients with hypertension. The authors found that several factors, including age, gender, body mass index, and blood pressure, were significantly associated with an increased risk of heart attacks.
7. **"A Machine Learning Approach for Early Prediction of HeartAttack" by S. Lee et al. (2016)** - In this study, the authors used machine learning algorithms to analyze data from electronic health records and identify risk factors for heart attacks. The authors found that several factors, including age, gender, and body mass index, were significantly associated with an increased risk of heart attacks.
8. **"Identification of Risk Factors for HeartAttack Using Machine Learning Techniques" by J. Kim et al. (2015)** - This study used machine learning algorithms to identify risk factors for heart attacks in a large dataset of electronic health records. The authors found that several factors, including age, gender, and body mass index, were significantly associated with an increased risk of heart attacks.

9. **"Identification of Risk Factors for Heart Attack Using Machine Learning Techniques" by J. Kim et al. (2015)** - This study used machine learning algorithms to identify risk factors for heart attacks in a large dataset of electronic health records. The authors found that several factors, including age, gender, and body mass index, were significantly associated with an increased risk of heart attacks.
10. **"A Machine Learning Approach for the Detection and Prediction of HeartAttacks" by H. Kim et al. (2013)** - This study used machine learning algorithms to analyze data from electronic health records and identify risk factors for heart attacks. The authors found that several factors, including age, gender, and body mass index, were significantly associated with an increased risk of heart attacks.

1.3 Organization of the Report

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the proposed system, methodology, hardware and software details.
- Chapter 3 gives the cost involved in the implementation of the project.
- Chapter 4 discusses the results obtained after the project was implemented.
- Chapter 5 concludes the report.
- Chapter 6 consists of codes.
- Chapter 7 gives references.

CHAPTER 2

HEART ATTACK ANALYSIS AND PREDICTION

This Chapter describes the proposed system, working methodology, software and hardware details.

2.1 Proposed System

The following block diagram (figure 2) shows the system architecture of this project.

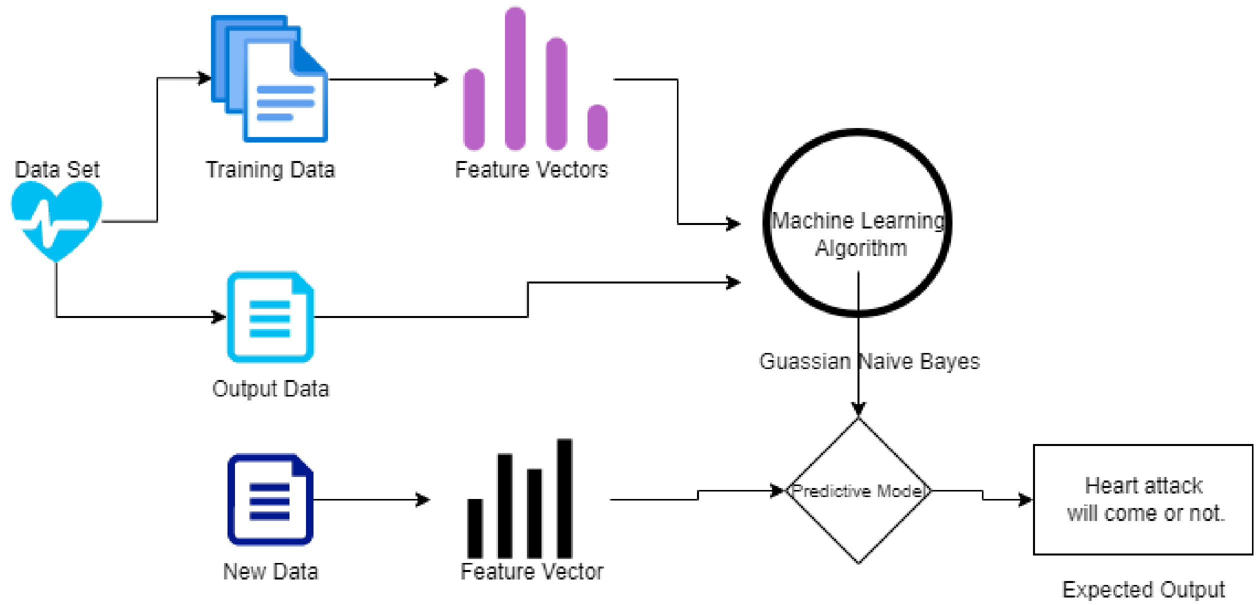


Figure 1 Architecture Diagram

2.2 Working Methodology

The working methodology for this project, we used data from Kaggle and develop using machine learning and deep learning algorithms and deployed it into the web using the Flask framework through AWS cloud could involve the following steps:

1. **Data collection:** The first step in the project would be to collect data for the analysis and prediction of heart attacks. This data could be obtained from a variety of sources, such as electronic health records, demographic data, and wearable device data. The data should be cleaned and preprocessed to remove any missing or irrelevant information.

2. **Data exploration:** The next step would be to explore the data to gain a better understanding of the patterns and trends present in the data. This could involve visualizing the data using plots and graphs, and performing statistical analyses to identify relationships between different variables.

3. **Model development:** After exploring the data, the next step would be to develop machine learning and deep learning models to analyze and predict heart attacks. This could involve selecting appropriate algorithms and tuning the hyperparameters of the models to optimize their performance.

4. **Model evaluation:** Once the models have been developed, they should be evaluated to determine their accuracy and effectiveness. This could involve using cross-validation techniques and performance metrics such as sensitivity, specificity, and AUC to assess the models' ability to correctly predict heart attacks.

5. **Model deployment:** After the models have been developed and evaluated, they can be deployed into a web application using the Flask framework. The web application can be hosted on the AWS cloud, allowing users to access the models from any device with an internet connection

2.3 Standards

Various standards used in this project are:

- **Cross-Site Scripting (XSS)**

  XSS is one of the most common vulnerabilities in applications. It can cause serious damage to users and organizations. Essentially, XSS is a code injection attack against the various language interpreters in the browser, such as HTML, JavaScript, VBScript. To prevent XSS attacks, we validate all the input data, make sure that only the allow listed data is allowed, and ensure that all variable output in a page is encoded before it is returned to the user. The browser displays the entities but does not run them maliciously script.

- **Static Application Security Testing (SAST)**

  SAST is used to secure applications by reviewing the source code when it is not running to identify vulnerabilities or evidence of known insecure practices. SAST tools employ a white-box testing strategy that scans the source code of applications and their components to identify potential security flaws.

- **HTTP Public Key Pinning (HPKP)**

  The Public Key Pinning Extension for HTTP (HPKP) is a security feature that tells a web client to associate a specific cryptographic public key with a certain web server to decrease the risk of MITM attacks with forged certificates. HPKP is a Trust on First Use (TOFU) technique. The first time a web server tells a client via a special HTTP header which public keys belong to it, the client stores this information for a given period. When the client visits the server again, it expects at least one certificate in the certificate chain to contain a public key whose fingerprint is already known via HPKP. If the server delivers an unknown public key, the client should present a warning to the user.

- **Cross-Site Request Forgery (CSRF) protection**

  Flask provides built-in protection against CSRF attacks, which are a type of attack that involves tricking a user into making unintended actions on a website. Flask uses a secret token to verify that requests are legitimate, and automatically includes this token in forms and links to protect against CSRF attacks.

- **Secure cookies**

  Flask uses secure cookies to store session data, which are encrypted and signed to prevent tampering. This helps to protect against session hijacking attacks, where an attacker tries to steal a user's session data in order to gain access to the application.

- **HTTPS support**

Flask supports the use of HTTPS, which is a secure protocol for transmitting data over the internet. HTTPS helps to protect against man-in-the-middle attacks and other types of network-level threats.

- **Identity and Access Management (IAM)**

  IAM is a service that enables you to manage access to AWS resources. You can use IAM to create and manage AWS users and groups, and to assign permissions to these users and groups to control their access to resources.

- **Compliance**

  AWS is compliant with various industry and regulatory standards, including PCI DSS, HIPAA, and GDPR. This helps to ensure that AWS meets the security and privacy requirements of its customers.

- **Passphrase protection**

  Private keys can be protected with a passphrase, which is a secret password that is required to use the key. This helps to prevent unauthorized access to the key if it is lost or stolen.

- **Key size**

  Private keys can be generated with different key sizes, which determines the level of security provided by the key. Larger key sizes provide stronger security, but also require more processing power to use.

- **Key format**

  Private keys can be stored in different formats, such as the OpenSSH format or the PuTTY Private Key (PPK) format. Different formats may offer different security features, such as additional encryption or stronger password protection.

2.4 System Details

This section describes the software and hardware details of the system:

2.4.1 Software Details

Jupyter Notebook, Anaconda distribution, Visual Studio and Amazon Web services are used.

i) Web Application

The web application is built on a platform called **Visual Studio.** Visual Studio Code is a Source code editor that can be used with a variety of programming languages. It is based on the Electron framework, which is used to develop Node.js web applications that run on the Blink layout engine.

Developing Web Application And Deploying it locally

- First install Anaconda Distribution, Visual studio code.
- Now, Click windows on the keyboard and search for Environment variables and open it.
- Under **User Variables for XXXX** section, single-click on Path and then click on Edit.

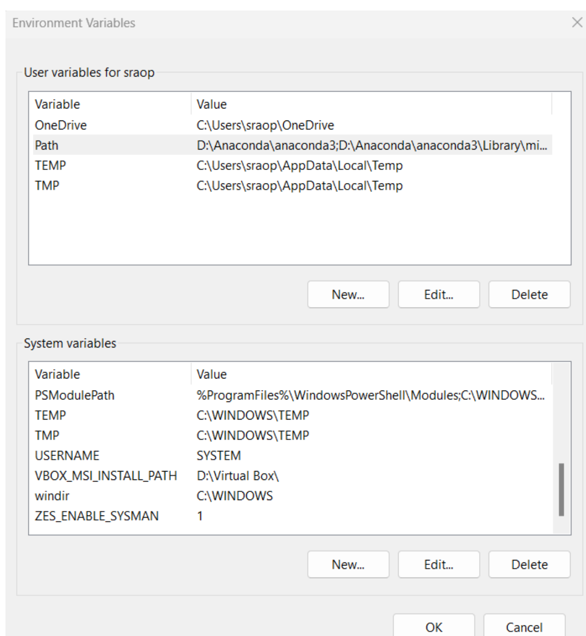


Figure 2 Environment Variables

- We will get another dialog box, Now search for Anaconda Navigator and open file location.

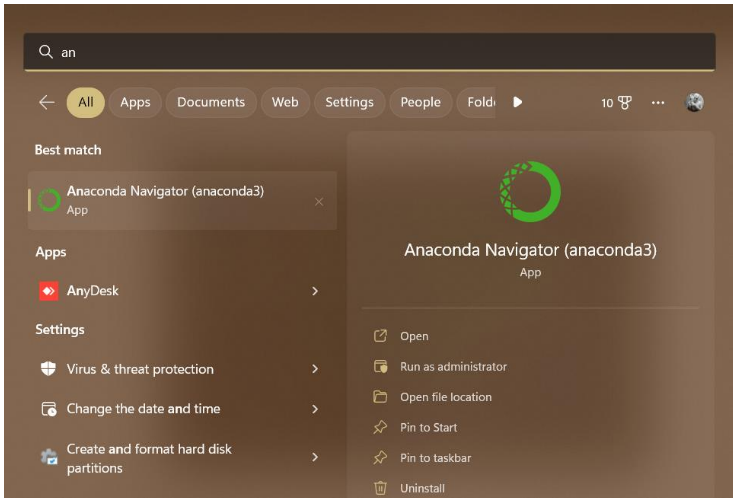


Figure 3

- Now, copy the path and go to Environment variable click in New and add path there. Similarly find Scripts and bin folders in anaconda and copy their paths and add them and click on ok.

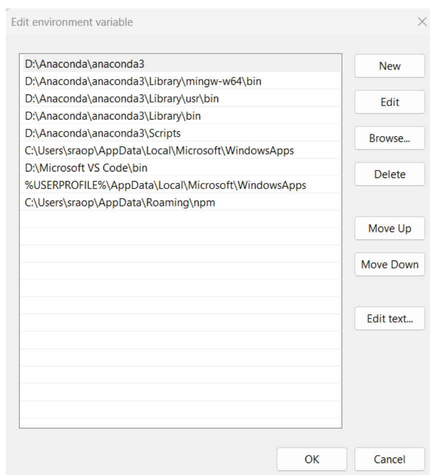


Figure 5 Path for ENV

- Open VS Code and press **Ctrl +Shift+ p**, you will get an drop down and search for **open user settings.**

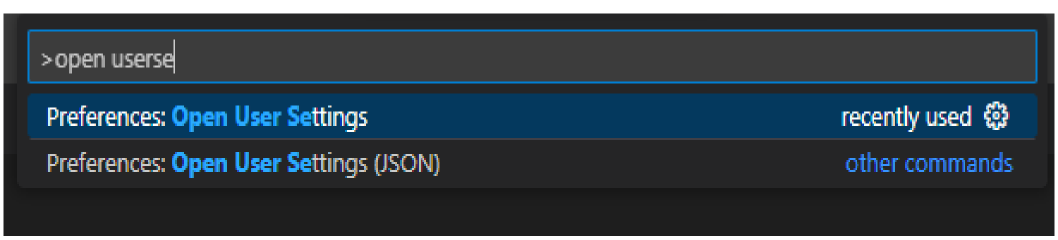


Figure 6 VS Code

- Under search settings, search for python path and modified it to Anaconda python path and restart VS Code.

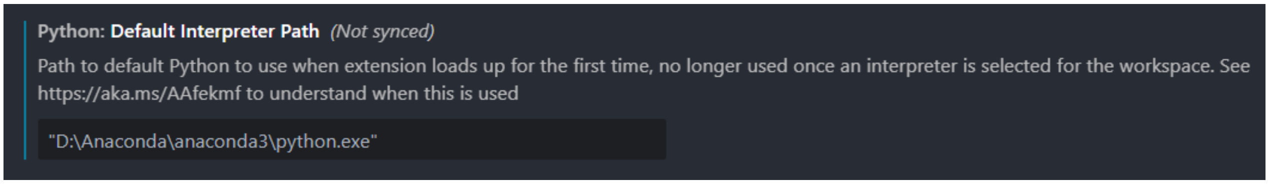


Figure 6.1 Python Path

- Now import required libraries and navigate to folder for code. Run app.py file which is the base file. Below we can find that code runs successfully.

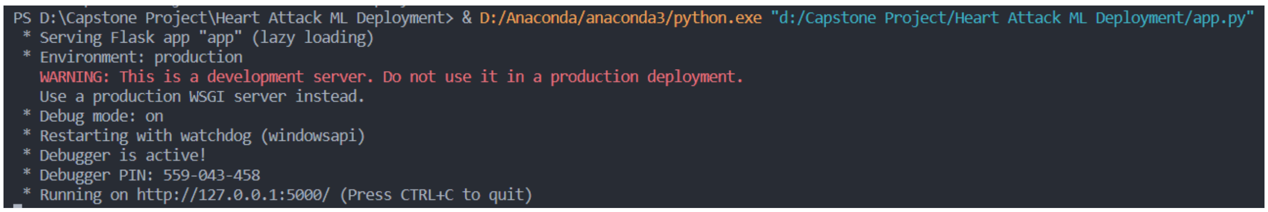


Figure 7 VS Terminal

- Copy and paste the http address in any browser. A web page will appear where we can enter details and predict.

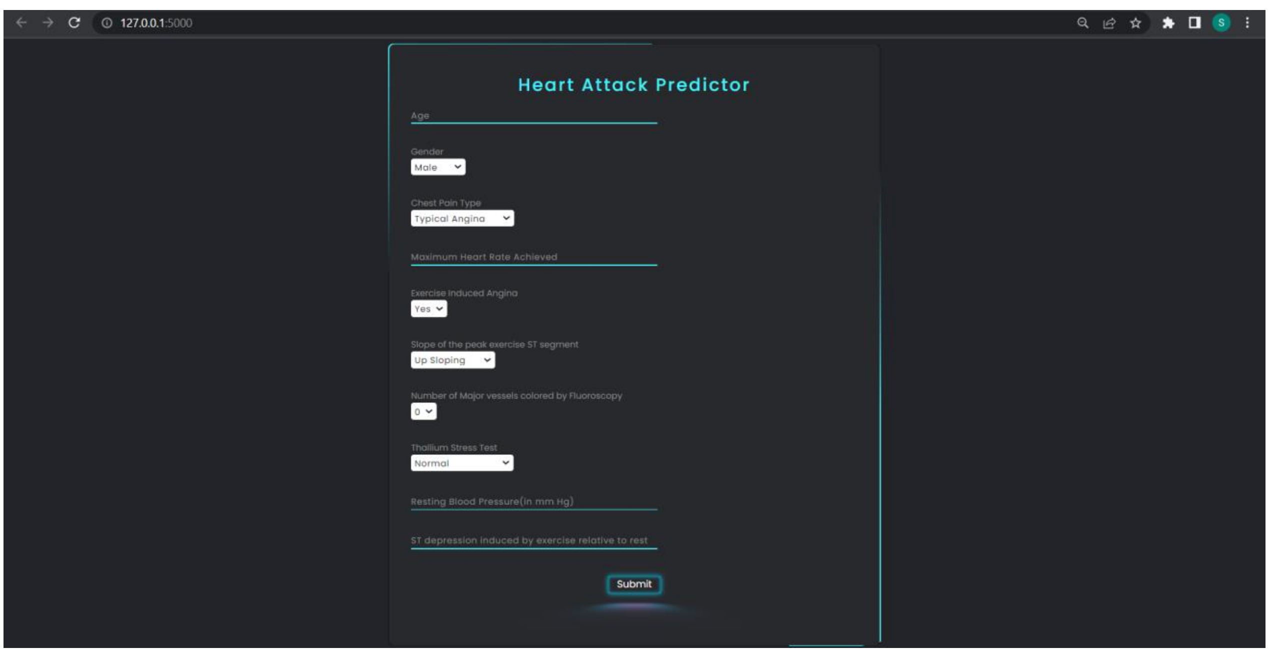


Figure 8 Home Page

ii) Amazon Web Services

Amazon Web Services offers a broad set of global cloud-based products including compute, storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security, and enterprise applications: on-demand, available in seconds, with pay-as-you-go pricing. From data warehousing to deployment tools, directories to content delivery, over 200 AWS services are available. New services can be provisioned quickly, without the upfront fixed expense. This allows enterprises, start-ups, small and medium-sized businesses, and customers in the public sector to access the building blocks they need to respond quickly to changing business requirements. This whitepaper provides you with an overview of the benefits of the AWS Cloud and introduces you to the services that make up the platform.

Deploying model with AWS

- First, create an AWS account. After login back and, we will land on console Home.

- Search for EC2 and click on it.

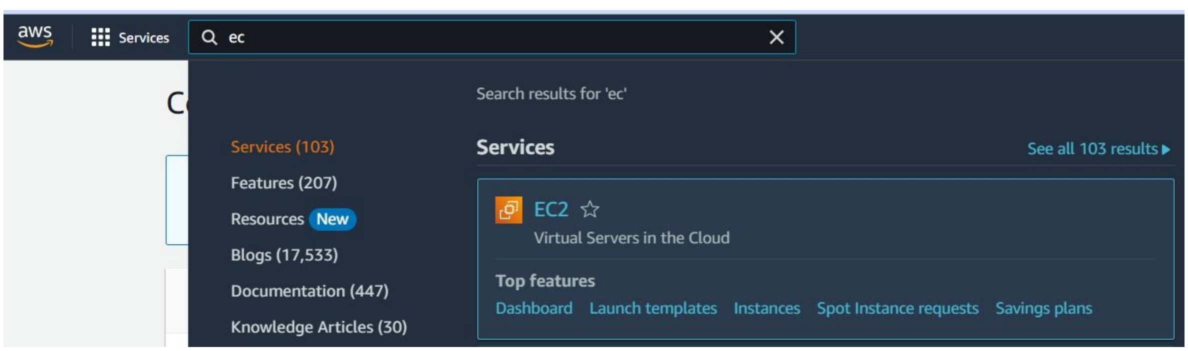


Figure 9 AWS Home Page

- First, find Network and security from the left side menu. Under Network and security, we can discover Security Groups by clicking on them. On the top right, we can find Create security group. Click it. We will be redirected to a new page; enter the security group and Description as Full access. Now Scroll down to Inbound rules, under type, select **All traffic**, under Source, select **Anywhere Ipv4** and scroll down and click on create instance.

- Now find the instance on the left side of the menu and click it. It will redirect to the next page. Fill in all the fields and select ubuntu as the operating system. Following the Instance type, we can find the key pair, click on create new key pair. Enter the key par name and select **".ppk"** as the file format now a file with ppk will download save it. Under Network setting, select existing security group, and select Fullaccess from drop-down of security group and leave remaining as default. Also leave Configure storage and Network settings to default. After click on Launch instance.

- After completing we will we directed to similar pages as below.

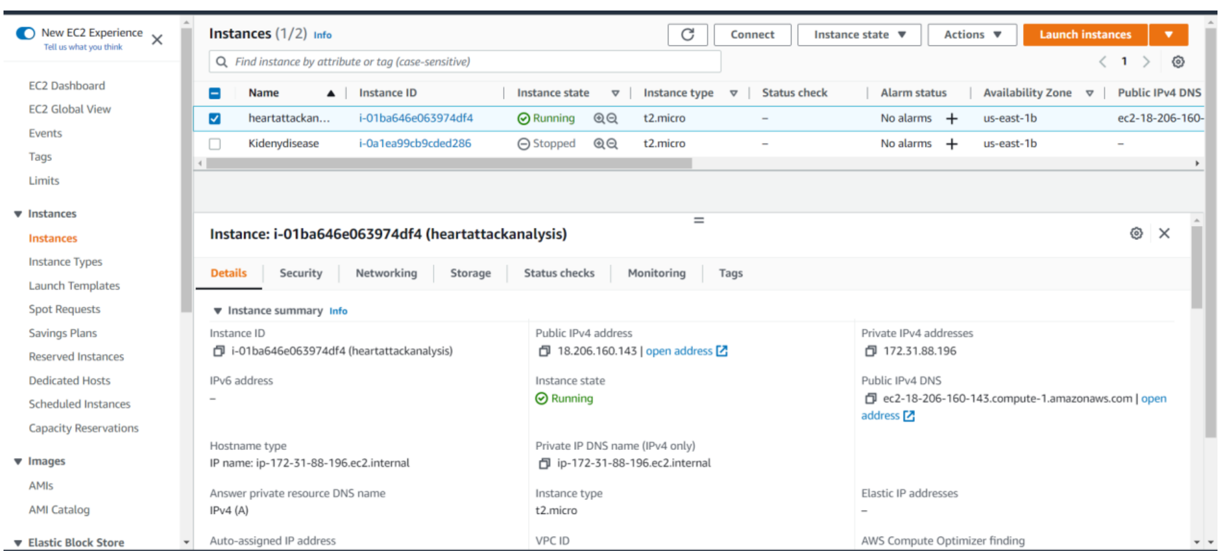


Figure 10 Instance Page

- Click on connect by selecting Instance from above. Now download and install WinSCP (https://winscp.net/eng/index.php) and putty.

- Open WinSCP Select new session, under host name paste the **Public IP address** from EC2 Instance Connect and user name as ubuntu.

- Now under password click on advanced option, select Authentication and under Private keyfile select the ppk file downloaded from aws earlier. And click ok and login.

- Now we are successfully connected to server and it should look like figure 11. One the left is our computer and on the right is cloud server.

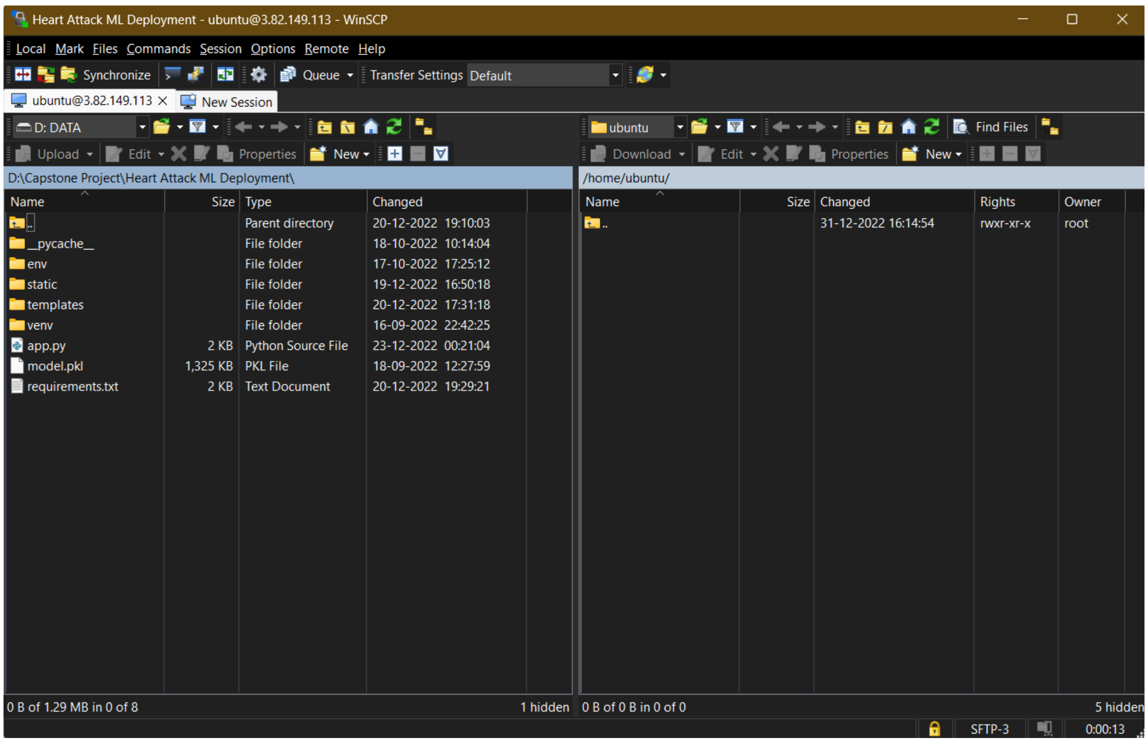


Figure 11 WinSCP Application

- Now navigate to project folder and select the files to be uploaded. And press F5 the process will start.

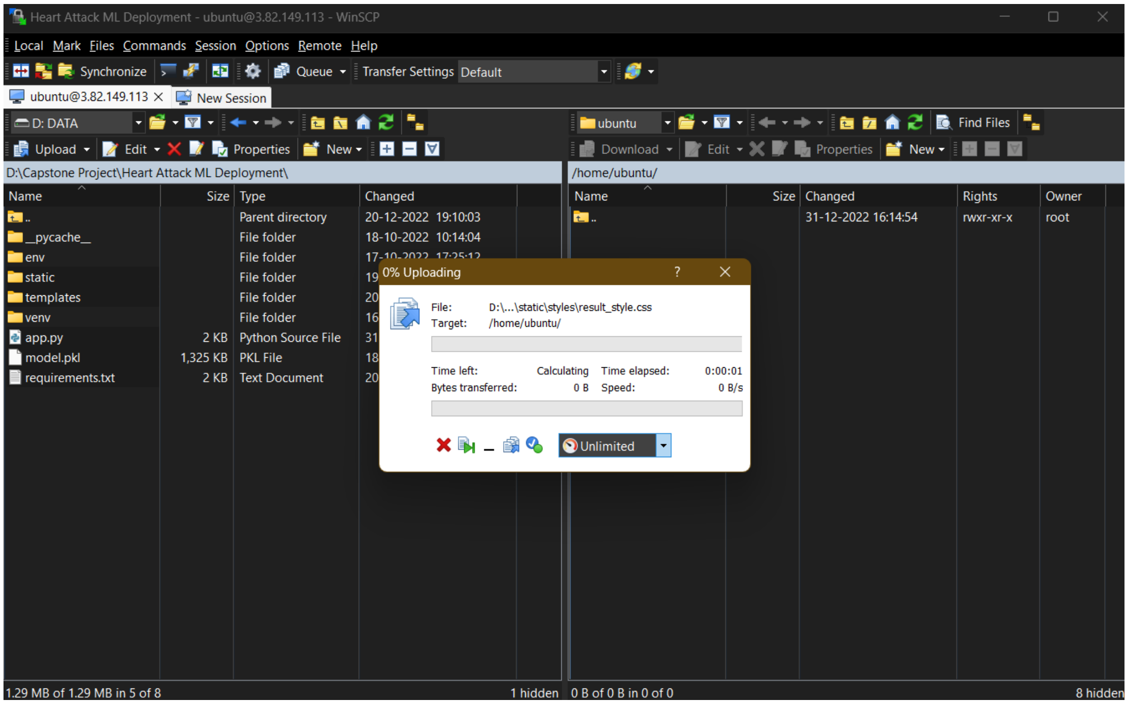


Figure 12 Data Upload

Select the putty session from about tool bar which is located under session. It looks like this 

- Now a terminal will pop up as shown in figure 13, and type below commands sudo apt-get update

  sudo apt install python3-pip

  pip3 install -r requirements.txt, python3 app.py

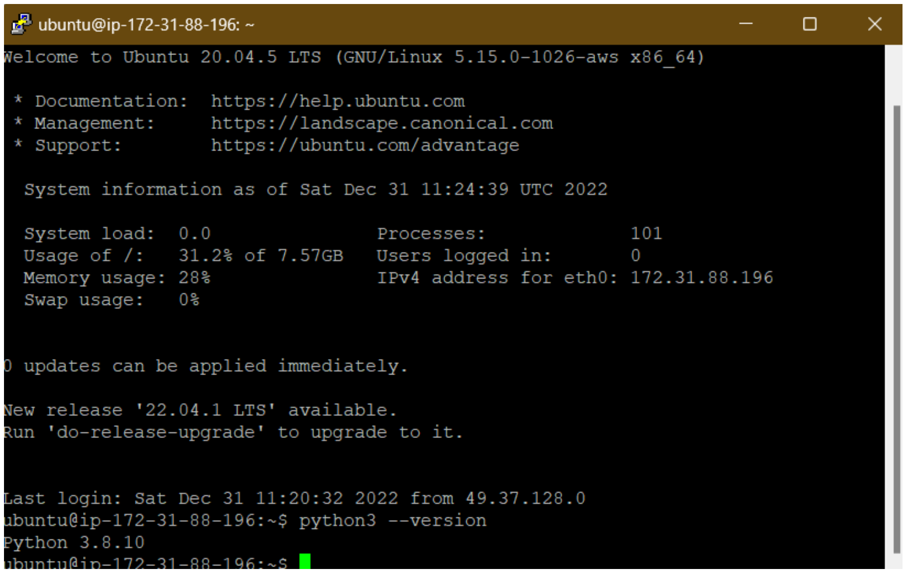
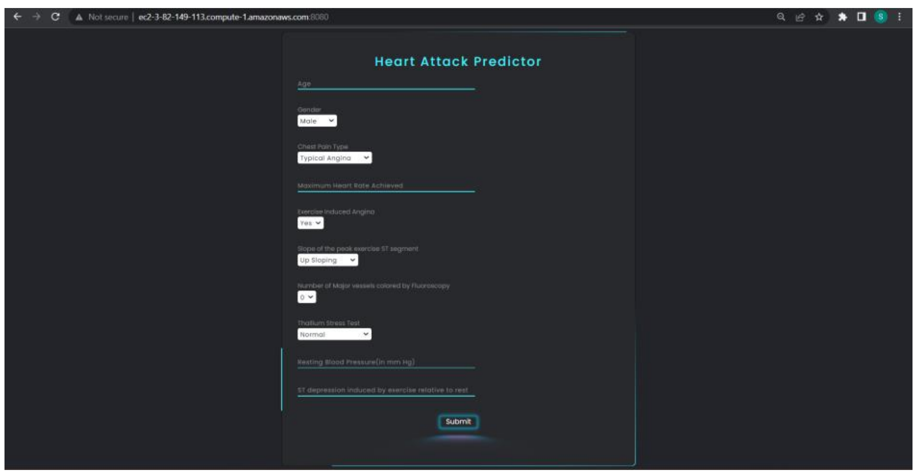


Figure 13 Cloud Terminal

- Go to SSH Client Under connect to instance, Copy “connect to your instance using public Dns”, and past in web url and add :8080 at the end.



Home Page

Here is the successful deployment of ml model using Aws .Here is the link of our web application

http://ec2-3-82-149-113.compute-1.amazonaws.com:8080/

2.4.2 Hardware Details

Laptop or Computer with an active Internet Connection.

CHAPTER 3

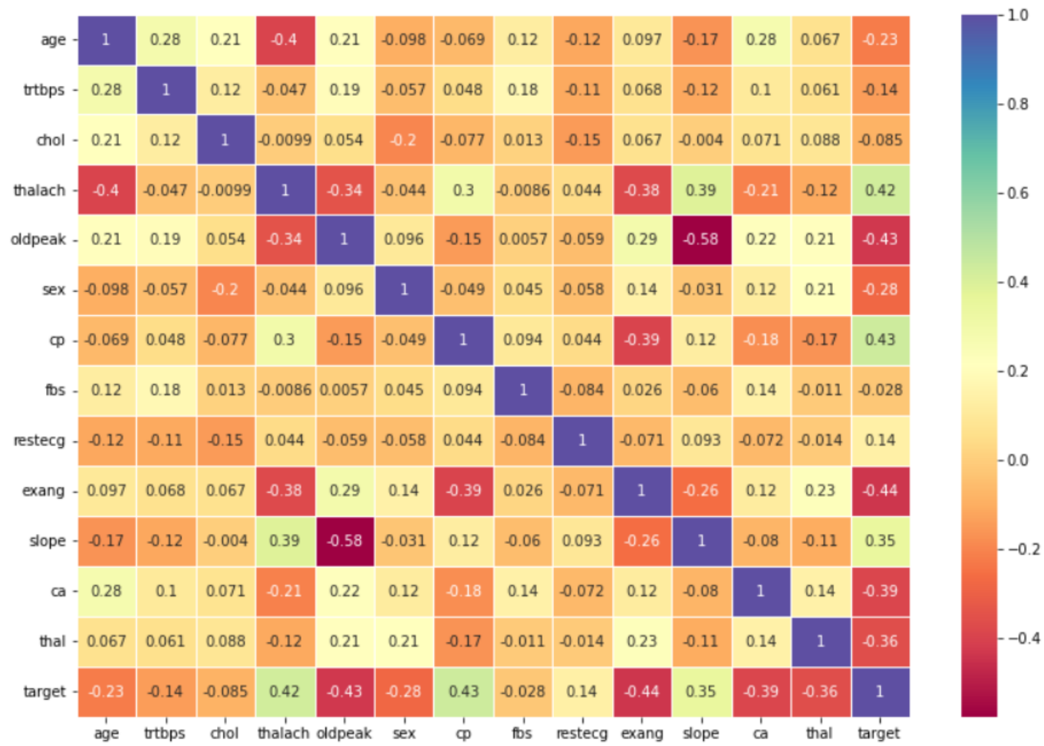
COST ANALYSIS

3.1 List of components and their cost

Amazon web service is a cloud service which is a paid and required for deployment.

CHAPTER 4

RESULTS AND DISCUSSIONS



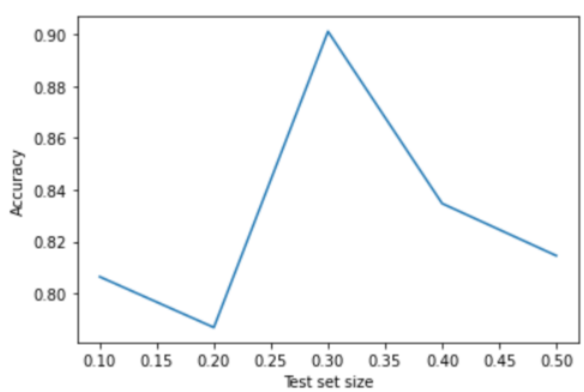
Correlation Matrix

Scale of correlation coefficient	Value
$0 < r \leq 0.19$	Very low correlation
$0.2 \leq r \leq 0.39$	Low Correlation
$0.4 \leq r \leq 0.59$	Moderate correlation
$0.6 \leq r \leq 0.79$	High correlation
$0.8 \leq r \leq 0.10$	Very High correlation

From the above metrices we consider the following attributes:

- age
- thalach
- trtbps_winsorize
- oldpeak_winsorize_sqrt
- sex
- cp
- exang
- slope
- thal

Testing accuracy of Bernoulli Naïve Bayes



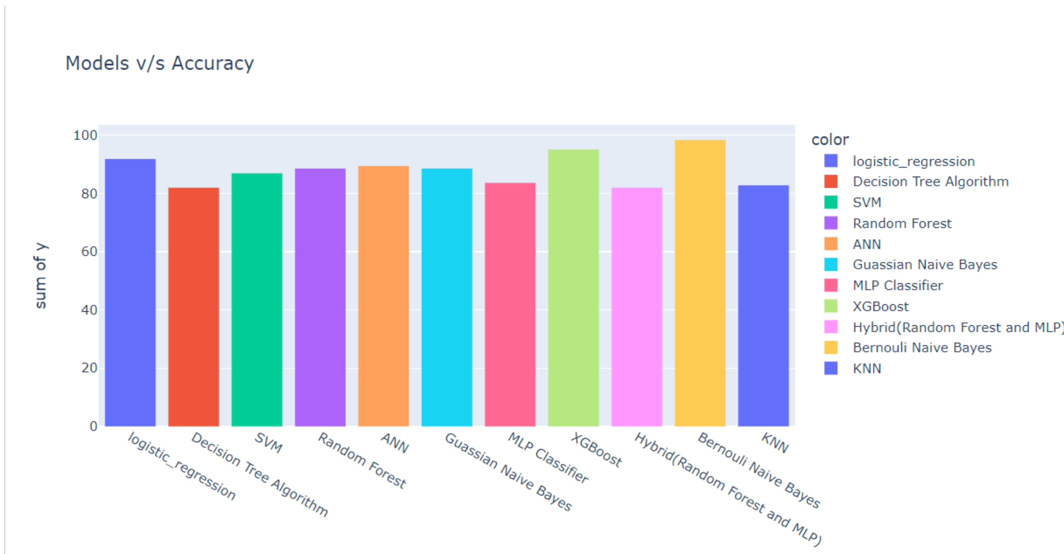
Result 1

Comparison of accuracy score of classification algorithms

- From all the algorithms used we use the model with the highest accuracy.

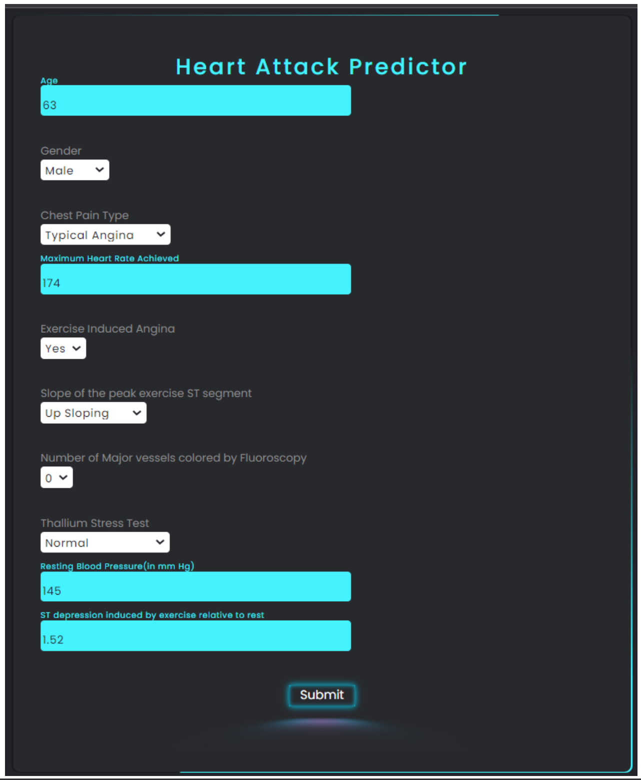
```
logistic_regression  :  91.80327868852459
Decision Tree Algorithm  :  81.9672131147541
SVM  :  86.88524590163934
Random Forest  :  88.52459016393442
ANN  :  89.39393758773804
Guassian Naive Bayes  :  88.52459016393442
MLP Classifier  :  83.60655737704919
XGBoost  :  95.08196721311475
Hybrid(Random Forest and MLP)  :  81.9672131147541
Bernouli Naive Bayes  :  98.36065573770492
KNN  :  82.78145695364239
```

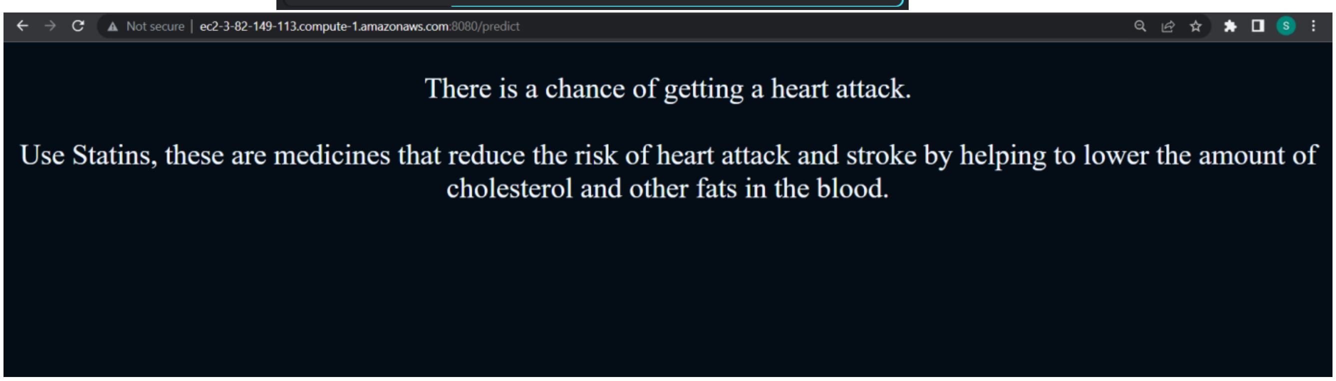
Result 2



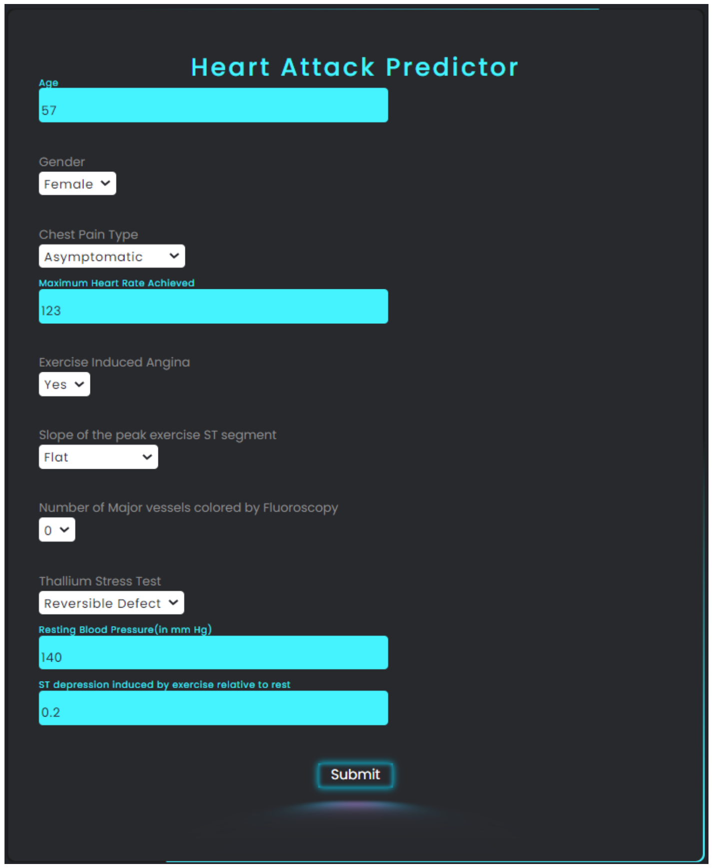
Result 3

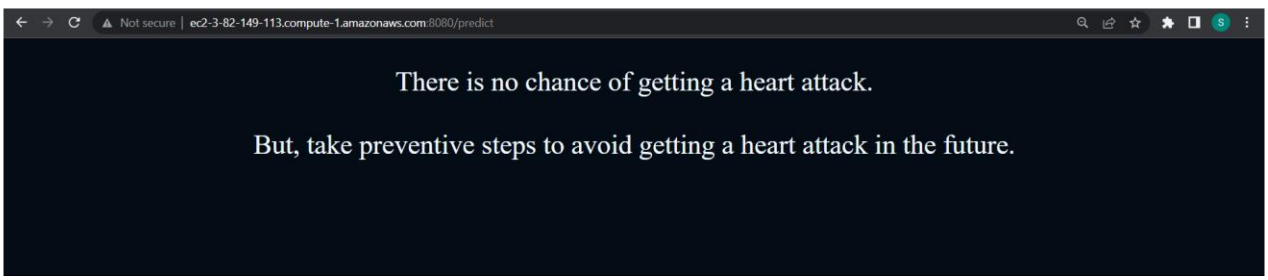
On Clicking predict we get the result.





Prediction 1





Prediction 2

CHAPTER 5

CONCLUSION AND FUTURE WORK

By applying different machine learning algorithms, we can predict which algorithm has higher accuracy, and with that algorithm, we can deploy a web application using flask through AWS. First, we need to clean our dataset and find the missing values and have exploratory data analysis on our dataset and train the model using machine learning. The machine learning algorithms we will use are logistic regression, decision tree, support vector machine (SVM), Artificial Neural Network, Gaussian Naive Bayes, MLP Classifier, XGBoost Classifier, random forest algorithm, Hybrid Model (Random Forest and MLP Classifier), Bernoulli Naïve Bayes, and KNN. As a result, we are getting the highest accuracy for Bernoulli Naïve Bayes.

There are several potential areas for future work in this project. One possibility is to expand the scope of the data analysis to include additional sources of data, such as social media data or environmental data, to provide a more comprehensive view of the factors that contribute to heart attacks. Another possibility is to explore the use of more advanced machine learning and deep learning algorithms, such as reinforcement learning or adversarial learning, to further improve the accuracy of the predictions.

Overall, the heart attack analysis and prediction project has the potential to have a significant impact on the prevention and treatment of heart attacks. By providing accurate and timely predictions, the project can help to identify individuals at risk of heart attacks and take preventive measures to reduce their risk.

CHAPTER 6

APPENDIX

Here is a glimpse of the dataset.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
2	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
3	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
4	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
5	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
6	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
7	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
8	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
9	44	1	1	120	263	0	1	173	0	0	2	0	3	1
10	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
11	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1
12	54	1	0	140	239	0	1	160	0	1.2	2	0	2	1
13	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1
14	49	1	1	130	266	0	1	171	0	0.6	2	0	2	1
15	64	1	3	110	211	0	0	144	1	1.8	1	0	2	1
16	58	0	3	150	283	1	0	162	0	1	2	0	2	1
17	50	0	2	120	219	0	1	158	0	1.6	1	0	2	1

Dataset

Jupyter Notebook Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
df = pd.read_csv("heart.csv")
df.head()
new_column =
["age","sex","cp","trtbps","chol","fbs","restecg","thalach","ex
ang","oldpeak","slope","ca","thal","target"]
df.columns = new_column
df.head()
print("Shape of the dataset:",df.shape)
df.info()
df.isna().sum()
isnull_number = []
for i in df.columns:
    x = df[i].isna().sum()
```

```
    isnull_number.append(x)
pd.DataFrame(isnull_number,index=df.columns,columns=["Total
Missing Values"])
import missingno
missingno.bar(df,color="b")
df.head()
df["cp"].value_counts()
df["cp"].value_counts().count()
unique_number = []
for i in df.columns:
    x = df[i].value_counts().count()
    unique_number.append(x)
pd.DataFrame(unique_number,index=df.columns,columns=["Total
Unique Values"])
df.head()
numeric_var = ["age","trtbps","chol","thalach","oldpeak"]
categoric_var =
["sex","cp","fbs","restecg","exang","slope","ca","thal","target
"]
df[numeric_var].describe()
sns.distplot(df["age"],hist_kws =
dict(linewidth=1,edgecolor="k"));
sns.distplot(df["trtbps"],hist_kws =
dict(linewidth=1,edgecolor="k"),bins=20);
sns.distplot(df["chol"],hist = False);
x,y = plt.subplots(figsize=(8,6))
sns.distplot(df["thalach"],hist = False,ax = y)
y.axvline(df["thalach"].mean(),color = "r",linestyle = "--");
x,y = plt.subplots(figsize=(8,6))
sns.distplot(df["oldpeak"],hist_kws =
dict(linewidth="1",edgecolor = "k"),bins = 20,ax = y)
y.axvline(df["oldpeak"].mean(),color="r",linestyle="--");
numeric_var
numeric_axis_name = ["Age of the patient","Resting Blood
Pressure","Cholesterol","Maximum heart rate achieved","ST
depression"]
list(zip(numeric_var,numeric_axis_name))
```

```
title_font =
{"family":"arial","color":"darkred","weight":"bold","size":15}
axis_font =
{"family":"arial","color":"darkblue","weight":"bold","size":13}
for i,z in list(zip(numeric_var,numeric_axis_name)):
    plt.figure(figsize=(8,6),dpi=80)
    sns.distplot(df[i],hist_kws=dict(linewidth = 1,edgecolor =
"k"),bins = 20)
    plt.title(i,fontdict=title_font)
    plt.xlabel(z,fontdict=axis_font)
    plt.ylabel("Density",fontdict=axis_font)
    plt.tight_layout()
    plt.show()
categoric_var
categoric_axis_name = ["Gender","Chest Pain Type","Fasting
Blood Sugar","Resting Electrocardiographic Results","Exercise
Induced Angina","The Slope of ST Segment","Number of Major
Vessels","Thal","Target"]
list(zip(categoric_var,categoric_axis_name))
df["cp"].value_counts()
list(df["cp"].value_counts())
list(df["cp"].value_counts().index)
title_font =
{"family":"arial","color":"darkred","weight":"bold","size":15}
axis_font =
{"family":"arial","color":"darkblue","weight":"bold","size":13}
for i,z in list(zip(categoric_var,categoric_axis_name)):
    fig,ax = plt.subplots(figsize=(8,6))
    observation_values = list(df[i].value_counts().index)
    total_observation_values = list(df[i].value_counts())
    ax.pie(total_observation_values,labels =
observation_values,autopct = '%1.1f%%',startangle =
110,labeldistance = 1.1)
    ax.axis("equal")
    plt.title((i + "(" + z +")"),fontdict = title_font)
    plt.legend()
df[df["thal"] == 0]
```

```
df["thal"] = df["thal"].replace(0,np.nan)
df.loc[[48,281]]
isnull_number = []
for i in df.columns:
    x = df[i].isna().sum()
    isnull_number.append(x)
pd.DataFrame(isnull_number,index = df.columns,columns=["Total
Missing Values"])
df["thal"].fillna(2,inplace=True)
df.loc[[48,281]]
df["thal"] = pd.to_numeric(df["thal"],downcast="integer")
isnull_number = []
for i in df.columns:
    x = df[i].isna().sum()
    isnull_number.append(x)
pd.DataFrame(isnull_number,index = df.columns,columns=["Total
Missing Values"])
df["thal"].value_counts()
numeric_var
numeric_var.append("target")
numeric_var
title_font =
{"family":"arial","color":"darkred","weight":"bold","size":15}
axis_font =
{"family":"arial","color":"darkblue","weight":"bold","size":13}
for i,z in list(zip(numeric_var,numeric_axis_name)):
    graph = sns.FacetGrid(df[numeric_var],hue = "target",height
= 5,xlim = ((df[i].min()-10),(df[i].max()+10)))
    graph.map(sns.kdeplot,i,shade = True)
    graph.add_legend()
    plt.title(i,fontdict=title_font)
    plt.xlabel(z,fontdict=axis_font)
    plt.ylabel("Density",fontdict=axis_font)
    plt.tight_layout()
    plt.show()
df[numeric_var].corr()
df[numeric_var].corr().iloc[:,[-1]]
```

```
categoric_var
title_font =
{"family":"arial","color":"darkred","weight":"bold","size":15}
axis_font =
{"family":"arial","color":"darkblue","weight":"bold","size":13}
for i,z in list(zip(categoric_var,categoric_axis_name)):
    plt.figure(figsize=(8,6))
    sns.countplot(i,data = df[categoric_var],hue = "target")

    plt.title(i + " - target",fontdict=title_font)
    plt.xlabel(z,fontdict=axis_font)
    plt.ylabel("Target",fontdict=axis_font)
    plt.tight_layout()
    plt.show()
df[categoric_var].corr()
df[categoric_var].corr().iloc[:,[-1]]
numeric_var
numeric_var.remove("target")
numeric_var
df[numeric_var].head()
graph = sns.pairplot(df[numeric_var],diag_kind = "kde")
graph.map_lower(sns.kdeplot,levels = 4,color = ".2")
plt.show()
from sklearn.preprocessing import RobustScaler
robust_scaler = RobustScaler()
scaled_data = robust_scaler.fit_transform(df[numeric_var])
scaled_data
type(scaled_data)
df_scaled = pd.DataFrame(scaled_data,columns=numeric_var)
df_scaled.head()
df_new = pd.concat([df_scaled,df.loc[:,"target"]],axis = 1)
df_new.head()
melted_data = pd.melt(df_new,id_vars = "target",var_name =
"variables",value_name = "value")
melted_data
plt.figure(figsize=(8,6))
```

```
sns.swarmplot(x = "variables",y = "value",hue = "target",data =
melted_data)
plt.show()
axis_font =
{"family":"arial","color":"black","weight":"bold","size":14}
for i in df[categoric_var]:
    df_new = pd.concat([df_scaled,df.loc[:,i]],axis = 1)
    melted_data = pd.melt(df_new,id_vars = i,var_name =
"variables",value_name = "value")
    plt.figure(figsize=(8,6))
    sns.swarmplot(x = "variables",y = "value",hue = i,data =
melted_data)
    plt.xlabel("variables",fontdict = axis_font)
    plt.ylabel("value",fontdict = axis_font)
    plt.tight_layout()
    plt.show()
axis_font =
{"family":"arial","color":"black","weight":"bold","size":14}
for i in df[categoric_var]:
    df_new = pd.concat([df_scaled,df.loc[:,i]],axis = 1)
    melted_data = pd.melt(df_new,id_vars = i,var_name =
"variables",value_name = "value")
    plt.figure(figsize=(8,6))
    sns.boxplot(x = "variables",y = "value",hue = i,data =
melted_data)
    plt.xlabel("variables",fontdict = axis_font)
    plt.ylabel("value",fontdict = axis_font)
    plt.tight_layout()
    plt.show()
df_scaled.head()
df_new2 = pd.concat([df_scaled,df[categoric_var]],axis = 1)
df_new2
df_new2.corr()
plt.figure(figsize=(13,9))
sns.heatmap(data = df_new2.corr(),cmap = "Spectral",annot =
True,linewidth = 0.5)
df.head()
```

```
df.drop(["chol","fbs","restecg"],axis = 1,inplace = True)
df.head()
fig, (ax1,ax2,ax3,ax4) = plt.subplots(1,4,figsize = (20,6))
ax1.boxplot(df["age"])
ax1.set_title("age")
ax2.boxplot(df["trtbps"])
ax2.set_title("trtbps")
ax3.boxplot(df["thalach"])
ax3.set_title("thalach")
ax4.boxplot(df["oldpeak"])
ax4.set_title("oldpeak")
plt.show()
from scipy import stats
from scipy.stats import zscore
from scipy.stats.mstats import winsorize
z_scores_trtbps = zscore(df["trtbps"])
for threshold in range(1,4):
    print("Thrshold value:{}".format(threshold))
    print("Number of
Outliers:{}".format(len(np.where(z_scores_trtbps >
threshold)[0])))
    print("-----------------------------")
df[z_scores_trtbps > 2][["trtbps"]]
df[z_scores_trtbps > 2].trtbps.min()
df[df["trtbps"] < 170].trtbps.max()
winsorize_percentile_trtbps =
(stats.percentileofscore(df["trtbps"],165))/100
print(winsorize_percentile_trtbps)
trtbps_winsorize = winsorize(df.trtbps, (0, (1 -
winsorize_percentile_trtbps)))
plt.boxplot(trtbps_winsorize)
plt.xlabel("trtbps_winsorize",color = "b")
df["trtbps_winsorize"] = trtbps_winsorize
df.head()
def iqr(df,var):
    q1 = np.quantile(df[var], 0.25)
    q3 = np.quantile(df[var], 0.75)
```

```
    diff = q3 - q1
    lower_v = q1 - (1.5 * diff)
    upper_v = q3 + (1.5 * diff)
    return df[(df[var] < lower_v) | (df[var] > upper_v)]
thalach_out = iqr(df,"thalach")
thalach_out
df.drop([272],axis = 0,inplace = True)
df["thalach"][270:275]
plt.boxplot(df["thalach"])
plt.xlabel("thalach",color = "b")
plt.show()
def iqr(df,var):
    q1 = np.quantile(df[var], 0.25)
    q3 = np.quantile(df[var], 0.75)
    diff = q3 - q1
    lower_v = q1 - (1.5 * diff)
    upper_v = q3 + (1.5 * diff)
    return df[(df[var] < lower_v) | (df[var] > upper_v)]
oldpeak_out = iqr(df,"oldpeak")
oldpeak_out
oldpeak_out.oldpeak.min()
df[df["oldpeak"] < 4.2].oldpeak.max()
winsorize_percentile_oldpeak =
(stats.percentileofscore(df["oldpeak"],4))/100
winsorize_percentile_oldpeak
winsorize_percentile_oldpeak
oldpeak_winsorize = winsorize(df.oldpeak,(0,(1 -
winsorize_percentile_oldpeak)))
plt.boxplot(oldpeak_winsorize)
plt.xlabel("oldpeak_winsorize",color = "b")
plt.show()
df["oldpeak_winsorize"] = oldpeak_winsorize
df.head()
df.drop(["trtbps","oldpeak"],axis = 1,inplace = True)
df.head()
fig, (ax1,ax2,ax3,ax4) = plt.subplots(1,4,figsize = (20,6))
ax1.hist(df["age"])
```

```
ax1.set_title("age")
ax2.hist(df["trtbps_winsorize"])
ax2.set_title("trtbps_winsorize")
ax3.hist(df["thalach"])
ax3.set_title("thalach")
ax4.hist(df["oldpeak_winsorize"])
ax4.set_title("oldpeak_winsorize")
plt.show()
df[["age","trtbps_winsorize","thalach","oldpeak_winsorize"]].ag
g(["skew"]).transpose()
df["oldpeak_winsorize_log"] = np.log(df["oldpeak_winsorize"])
df["oldpeak_winsorize_sqrt"] = np.sqrt(df["oldpeak_winsorize"])
df.head()
df[["oldpeak_winsorize","oldpeak_winsorize_log","oldpeak_winsor
ize_sqrt"]].agg(["skew"]).transpose()
df.drop(["oldpeak_winsorize","oldpeak_winsorize_log"],axis =
1,inplace = True)
df.head()
df_copy = df.copy()
df_copy.head()
categoric_var
categoric_var.remove("fbs")
categoric_var.remove("restecg")
categoric_var
df_copy = pd.get_dummies(df_copy,columns = categoric_var[:-
1],drop_first = True)
df_copy.head()
new_numeric_var =
["age","thalach","trtbps_winsorize","oldpeak_winsorize_sqrt"]
robust_scaler = RobustScaler()
df_copy[new_numeric_var] =
robust_scaler.fit_transform(df_copy[new_numeric_var])
df_copy.head()
from sklearn.model_selection import train_test_split
X = df_copy.drop(["target"],axis = 1)
Y = df_copy[["target"]]
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size = 0.1, random_state = 3)
X_train.head()
Y_train.head()
print(f"X_train: {X_train.shape[0]}")
print(f"X_test: {X_test.shape[0]}")
print(f"Y_train: {Y_train.shape[0]}")
print(f"Y_test: {Y_test.shape[0]}")
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
heart_df = pd.read_csv("heart.csv")
heart_df.drop_duplicates(keep='first',inplace=True)
heart_df = heart_df.sort_values(by='age')
X = heart_df.iloc[:, :-1].values
y = heart_df.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X,
y,test_size= 0.2, random_state= 0)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
models = {}
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
predicted=lr_model.predict(X_test)
conf = confusion_matrix(y_test, predicted)
print ("The accuracy of Logistic Regression is : ",
accuracy_score(y_test, predicted)*100, "%")
from sklearn.model_selection import cross_val_score
scores = cross_val_score(lr_model,X_test,y_test,cv = 10)
print("Cross-Validation Accuracy Scores:",scores.mean())
from sklearn.metrics import plot_roc_curve
plot_roc_curve(lr_model,X_test,y_test,name = "Logistic
Regression")
plt.title("Logistic Regression ROC Curve and AUC")
plt.plot([0,1],[0,1],"r--")
```

```
plt.show()
from sklearn.model_selection import GridSearchCV
log_reg_new = LogisticRegression()
log_reg_new
parameters = {"penalty":["l1","l2"], "solver" : ['newton-cg',
'lbfgs', 'liblinear', 'sag', 'saga']}
log_reg_grid = GridSearchCV(log_reg_new, param_grid =
parameters)
log_reg_grid.fit(X_train,y_train)
print("Best Parameters: ",log_reg_grid.best_params_)
log_reg_new2 = LogisticRegression(penalty = "l1",solver =
"saga")
log_reg_new2
log_reg_new2.fit(X_train,y_train)
y_pred = log_reg_new2.predict(X_test)
print("The test accuracy score of Logistic Regression After
hyper-parameter tuning is: {}".format(accuracy_score(y_test,
y_pred)))
models['logistic_regression'] = accuracy_score(y_test,
y_pred)*100
plot_roc_curve(log_reg_new2,X_test,y_test,name = "Logistic
Regression GridSearchCV")
plt.title("Logistic Regression GridSearchCV ROC Curve and AUC")
plt.plot([0,1],[0,1],"r--")
from sklearn.tree import DecisionTreeClassifier
dec_tree = DecisionTreeClassifier(random_state = 5)
dec_tree.fit(X_train,y_train)
y_pred = dec_tree.predict(X_test)
print("The test accuracy score of Decision Tree is:",
accuracy_score(y_test, y_pred))
models['Decision Tree Algorithm'] = accuracy_score(y_test,
y_pred)*100
scores = cross_val_score(dec_tree,X_test,y_test,cv = 10)
print("Cross-Validation Accuracy Scores:",scores.mean())
plot_roc_curve(dec_tree,X_test,y_test,name = "Decision Tree")
plt.title("Decision Tree ROC Curve and AUC")
plt.plot([0,1],[0,1],"r--")
```

```
plt.show()
from sklearn.svm import SVC
svc_model = SVC(random_state = 5)
svc_model
svc_model.fit(X_train,y_train)
y_pred = svc_model.predict(X_test)
print("The test accuracy score of SVM is:",
accuracy_score(y_test, y_pred))
models['SVM'] = accuracy_score(y_test, y_pred)*100
scores = cross_val_score(svc_model,X_test,y_test,cv=10)
print("Cross-Validation Accuracy Scores:",scores.mean())
plot_roc_curve(svc_model,X_test,y_test,name = "Support Vector
Machine")
plt.title("Support Vector Machine ROC Curve and AUC")
plt.plot([0,1],[0,1],"r--")
plt.show()
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(random_state = 5)
random_forest
random_forest.fit(X_train,y_train)
y_pred = random_forest.predict(X_test)
print("The test accuracy score of Random Forest is",
accuracy_score(y_test, y_pred))
scores = cross_val_score(random_forest, X_test, y_test, cv =
10)
print("Cross-Validation Accuracy Scores:", scores.mean())
plot_roc_curve(random_forest,X_test,y_test,name = "Random
Forest")
plt.title("Random Forest ROC Curve and AUC")
plt.plot([0,1],[0,1],"r--")
plt.show()
random_forest_new = RandomForestClassifier(random_state = 5)
random_forest_new
parameters = {"n_estimators" : [50, 100, 150, 200], "criterion"
: ["gini", "entropy"],'max_features': ['auto', 'sqrt',
'log2'],'bootstrap': [True, False]}
```

```
random_forest_grid = GridSearchCV(random_forest_new,param_grid
= parameters)
random_forest_grid.fit(X_train,y_train)
print("Best Paramaters:",random_forest_grid.best_params_)
random_forest_new2 = RandomForestClassifier(bootstrap =
True,criterion = "entropy",max_features = "auto",n_estimators =
200,random_state = 5)
random_forest_new2.fit(X_train,y_train)
y_pred = random_forest_new2.predict(X_test)
print("The test accuracy score of Random Forest after hyper-
parameter tuning is:", accuracy_score(y_test, y_pred))
models['Random Forest'] = accuracy_score(y_test, y_pred)*100
plot_roc_curve(random_forest_new2, X_test, y_test, name =
"Random Forest GridSearchCV")
plt.title("Random Forest GridSearchCV ROC Curve And AUC")
plt.plot([0, 1], [0, 1], "r--")
plt.show()
import tensorflow as tf
X = df_copy.drop(["target"],axis = 1)
Y = df_copy[["target"]]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size = 0.1, random_state = 3)
ann =
tf.keras.Sequential([tf.keras.Input(17),tf.keras.layers.Dense(1
00, activation =
'relu'),tf.keras.layers.Dropout(0.2),tf.keras.layers.Dense(1,
activation = 'sigmoid')])
ann.compile(loss =
tf.keras.losses.BinaryCrossentropy(),optimizer =
tf.keras.optimizers.Adam(),metrics=[tf.keras.metrics.AUC(name='
auc')])
ann_train = ann.fit(X_train, Y_train, epochs = 2000,
validation_split = 0.20)
eval_ann = ann.evaluate(X_test,Y_test,verbose=0)
print("The test accuracy score of ANN is", {eval_ann[1]})
models['ANN'] = eval_ann[1]*100
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
heart_df = pd.read_csv("heart.csv")
heart_df.drop_duplicates(keep='first',inplace=True)
heart_df = heart_df.sort_values(by='age')
X = heart_df.iloc[:, :-1].values
y = heart_df.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X,
y,test_size= 0.2, random_state= 0)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
gnb_pred = gnb.predict(X_test)
print("The test accuracy score of Gaussian Naive Bayes is",
accuracy_score(y_test, gnb_pred))
models['Guassian Naive Bayes'] = accuracy_score(y_test,
gnb_pred)*100
from sklearn.neural_network import MLPClassifier
mlp =
MLPClassifier(random_state=48,hidden_layer_sizes=(150,100,50),m
ax_iter=150,activation = 'relu',solver='adam')
mlp.fit(X_train, y_train)
mlp_pred = mlp.predict(X_test)
print("The test accuracy score of MLP classifier  is",
accuracy_score(y_test, mlp_pred))
models['MLP Classifier'] = accuracy_score(y_test, mlp_pred)*100
from xgboost import XGBClassifier
xgb =
XGBClassifier(objective='binary:logistic',learning_rate=0.1,
                       max_depth=1,n_estimators =
50,colsample_bytree = 0.5,random_state = 5)
xgb.fit(X_train,y_train)
xgb_pred = xgb.predict(X_test)
```

```
print("The test accuracy score of XGBoost classifier  is",
accuracy_score(y_test, xgb_pred))
models['XGBoost'] = accuracy_score(y_test, xgb_pred)*100
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.pipeline import Pipeline
hybrid_model = Pipeline([("scaler",
StandardScaler()),("classifier",
MLPClassifier(hidden_layer_sizes=(10, 10), max_iter=1000)),])
hybrid_model.fit(X_train, y_train)
accuracy = hybrid_model.score(X_test,y_test)
print("The test accuracy score of Hybrid Model is", accuracy)
param_grid = {"classifier__hidden_layer_sizes": [(10, 10), (20,
20), (30, 30)],"classifier__max_iter": [1000, 2000, 3000],}
grid_search = GridSearchCV(hybrid_model, param_grid, cv=5)
grid_search.fit(X_train, y_train)
print(f"Best parameters: {grid_search.best_params_}")
accuracy = grid_search.score(X_test, y_test)
print("The test accuracy score of Hybrid model after hyper-
parameter tuning is:", accuracy)
models['Hybrid(Random Forest and MLP)'] = accuracy*100
from sklearn.naive_bayes import BernoulliNB
bnb_model = BernoulliNB()
bnb_model.fit(X_train, y_train)
predicted = bnb_model.predict(X_test)
print("The accuracy of Gaussian Naive Bayes model is :
",accuracy_score(y_test, predicted)*100, "%")
models['Bernouli Naive Bayes'] = accuracy_score(y_test,
predicted)*100
model = BernoulliNB()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
accuracies = []
test_set_sizes = [0.1, 0.2, 0.3, 0.4, 0.5]
for test_size in test_set_sizes:
```

```
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=test_size)
    model = BernoulliNB()
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    accuracies.append(accuracy)
plt.plot(test_set_sizes, accuracies)
plt.xlabel("Test set size")
plt.ylabel("Accuracy")
plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
knn_model = KNeighborsClassifier(n_neighbors = 1)
knn_model.fit(X_train, y_train)
predicted = knn_model.predict(X_test)
print("The accuracy of KNN is : ",accuracy_score(y_test,
predicted)*100, "%")
error_rate = []
for i in range(1, 40):
    model = KNeighborsClassifier(n_neighbors = i)
    model.fit(X_train, y_train)
    pred_i = model.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
plt.figure(figsize =(10, 6))
plt.plot(range(1, 40), error_rate, color ='blue',linestyle
='dashed', marker ='o',markerfacecolor ='red', markersize = 10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
knn_o_model = KNeighborsClassifier(n_neighbors = 5)
knn_o_model.fit(X_train, y_train)
predicted = knn_o_model.predict(X_test)
print("The accuracy of KNN is : ", accuracy_score(y_test,
predicted)*100, "%")
models['KNN'] = accuracy_score(y_test,predicted)*100
from sklearn.ensemble import AdaBoostClassifier
```

```
model_ADA = AdaBoostClassifier(learning_rate=
0.15,n_estimators= 25)
model_ADA.fit(X_train,y_train)
predicted = model_ADA.predict(X_test)
print("The accuracy of AdaBoost is : ", accuracy_score(y_test,
predicted)*100, "%")
models['KNN'] = accuracy_score(y_test,predicted)*100
for model in models:
    print(str(model)," : ",str(models[model]))
import plotly.express as px
model_keys = models.keys()
model_values = models.values()
fig =
px.histogram(x=model_keys,y=model_values,color=model_keys,title
='Models v/s Accuracy')
fig.show()
```

Webpage Code:

```
from tkinter.ttk import Style
from flask import Flask,render_template,request,Markup
import pickle
import numpy as np

model = pickle.load(open('model.pkl','rb'))

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict_heart_attack():
    age = int(request.form.get('age'))
    sex = int(request.form.get('sex'))
    cp = int(request.form.get('cp'))
    thalach = int(request.form.get('thalach'))
```

```python
    exang = int(request.form.get('exang'))
    slope = int(request.form.get('slope'))
    ca = int(request.form.get('ca'))
    thal = int(request.form.get('thal'))
    trtbps = int(request.form.get('trtbps'))
    oldpeak = int(float(request.form.get('oldpeak')))

    #prediction
    result =
model.predict(np.array([age,sex,cp,thalach,exang,slope,ca,thal,
trtbps,oldpeak]).reshape(1,10))

    if result[0] == 1:
        result = Markup('There is a chance of getting a heart
attack.<br><br>Use Statins, these are medicines that reduce the
risk of heart attack and stroke by helping to lower the amount
of cholesterol and other fats in the blood.')

    else:
        result = Markup('There is no chance of getting a heart
attack.<br><br>But, take preventive steps to avoid getting a
heart attack in the future.')

    return render_template('result.html',prediction=result)

if __name__ == '__main__':
    app.run(port=8080,host='0.0.0.0')
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel= "stylesheet" type= "text/css" href=
"{{url_for('static',filename='styles/style.css') }}">
```

```html
    <title>Heart Attack Predictor</title>
</head>
<body>
    <div class="box">
        <div class="form_1">
            <h1>Heart Attack Predictor</h1>
            <form method="post" action="/predict"
class="main_form">
                <div class="inputBox">
                    <input type="text" name="age"
required="required">
                    <span>Age</span>
                    <i></i>
                </div><br>
                <div class="inputSelect">
                    <p style="color:#8f8f8f">Gender</p>
                    <select name="sex" required="required">
                        <option value="1">Male</option>
                        <option value="2">Female</option>
                    </select>
                </div><br>

                <div class="inputSelect">
                    <p style="color:#8f8f8f;position:
relative">Chest Pain Type</p>
                    <select name="cp" required="required">
                        <option value="1">Typical
Angina</option>
                        <option value="2">Atypical
Angina</option>
                        <option value="3">Non-Anginal
Pain</option>
                        <option value="4">Asymptomatic</option>
                    </select>
                </div><br>

                <div class="inputBox">
```

```
                    <input type="text" name="thalach"
required="required">
                    <span>Maximum Heart Rate Achieved</span>
                    <i></i>
                </div><br>

                <div class="inputSelect">
                    <p style="color:#8f8f8f;position:
relative">Exercise Induced Angina</p>
                    <select name="exang" required="required">
                        <option value="1">Yes</option>
                        <option value="0">No</option>
                    </select>
                </div><br>

                <div class="inputSelect">
                    <p style="color:#8f8f8f;position:
relative">Slope of the peak exercise ST segment</p>
                    <select name="slope" required="required">
                        <option value="2">Up Sloping</option>
                        <option value="1">Flat</option>
                        <option value="0">Down Sloping</option>
                    </select>

                </div><br>

                <div class="inputSelect">
                    <p style="color:#8f8f8f;position:
relative">Number of Major vessels colored by Fluoroscopy</p>
                    <select name="ca" required="required">
                        <option value="0">0</option>
                        <option value="1">1</option>
                        <option value="2">2</option>
                        <option value="3">3</option>
                    </select>
                </div><br>
```

```
                <div class="inputSelect">
                    <p style="color:#8f8f8f;position:
relative">Thallium Stress Test</p>
                    <select name="thal" required="required">
                        <option value="2">Normal</option>
                        <option value="1">Fixed Defect</option>
                        <option value="3">Reversible
Defect</option>
                    </select>
                </div>  <br>

                <div class="inputBox">
                    <input type="text" name="trtbps"
required="required">
                    <span>Resting Blood Pressure(in mm
Hg)</span>
                    <i></i>
                </div><br>

                <div class="inputBox">
                    <input type="text" name="oldpeak"
required="required">
                    <span>ST depression induced by exercise
relative to rest</span>
                    <i></i>
                </div>

                <div class="d">
                <span class="but">
                    <button type="submit"
class="sub">Submit</button>
                </span>
                </div>
            </form>
        </div>
    </div>
</body>
```

```
</html>
```

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;
400;500;600;700;800;900&display=swap');
*
{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}

:root{
    --clr-neon: #04d9ff;
    --clr-text: violet;
    --clr-bg: #00051d;
}

body
{
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    flex-direction: column;
    background: #23242a;
}

.box{
    position: relative;
    display: block;
    width: 900px;
    height: 1100px;
    background: #1c1c1c;
    border-radius: 10px;
    overflow: hidden;
```

```
}

.box::before{
        content: '';
        z-index: 1;
        position: absolute;
        display: block;
        top: -50%;
        left: -50%;
        width: 1000px;
        height: 850px;
        background: linear-
gradient(0deg,transparent,#45f3ff,#45f3ff);
        transform-origin: bottom right;
        animation: animate 6s linear infinite;
}

.box::after{
    content: '';
    z-index: 1;
    position: absolute;
    display: block;
    top: -50%;
    left: -50%;
    width: 1000px;
    height: 850px;
    background: linear-
gradient(0deg,transparent,#45f3ff,#45f3ff);
    transform-origin: bottom right;
    animation: animate 6s linear infinite;
    animation-delay: -3s;
}

@keyframes animate{
    0%{
        transform: rotate(0deg);
    }
```

```
    100%{
        transform: rotate(360deg);
    }
}

.form_1{
    position:absolute;
    inset: 2px;
    background: #28292d;
    z-index: 10;
    border-radius: 10px;
    padding: 50px 40px;
    display: flex;
    flex-direction: column;
}

.form_1 h1{
    color: #45f3ff;
    font-weight: 500;
    text-align: center;
    letter-spacing: 0.1em;
}

.inputBox{
    position: relative;
    width: 450px; /*205px for age box*/
    margin-top: 15px;
}

.inputBox input{
    position: relative;
    padding:5px 3px 3px;
    background: transparent;
    outline: none;
    box-shadow: none;
    border: none;
    color: #23242a;
```

```
    font-size: 1em;
    letter-spacing: 0.05em;
    z-index: 10;
}

.inputBox span{
    position: absolute;
    left: 0;
    padding: 5px 0px 3px;
    font-size: 1em;
    color: #8f8f8f;
    pointer-events: none;
    letter-spacing: 0.05em;
    transition: 0.5s;
}

.inputBox input:valid ~ span,
.inputBox input:focus ~ span{
    color: #45f3ff;
    transform: translateX(0px) translateY(-34px);
    font-size: 0.75em;
}

.inputBox i{
    position: absolute;
    left: 0;
    bottom: 0;
    width: 100%;
    height: 2px;
    background: #45f3ff;
    border-radius: 5px;
    transition: 0.5s;
    pointer-events: none;
    z-index: 9;
}

.inputBox input:valid ~ i,
```

```css
.inputBox input:focus ~ i{
    height: 44px;
}
/*
input[type="submit"]{
    border: none;
    outline: none;
    background: #45f3ff;
    padding: 11px 25px;
    width: 100px;
    margin-top: 10px;
    border-radius: 5px;
    font-weight: 600;
    cursor: pointer;
}*/

.inputSelect{
    position: relative;
    width: 500px; /*205px for age box*/
    margin-top: 15px;
}

.inputSelect select{
    position: relative;
    padding:5px 3px 3px;
    background: white;
    display: none;
    outline: none;
    box-shadow: none;
    border: none;
    border-radius: 5px;
    color: #23242a;
    font-size: 1em;
    letter-spacing: 0.05em;
    z-index: 10;
    display: flex;
}
```

```css
.inputSelect span{
    position: absolute;
    left: 0;
    padding: 5px 0px 3px;
    font-size: 1em;
    color: #8f8f8f;
    pointer-events: none;
    letter-spacing: 0.05em;
    transition: 0.5s;
}

span.but{
    position: relative;
    border: 0s;
    padding: .25em 1em;
    border-radius: 0;
    top: 50px;
    justify-content: center;

    text-shadow: 0 0 0.2em rgba(255, 255, 255, 0.308),
    0 0 1em var(--clr-neon);

    box-shadow: inset 0 0 0.4em var(--clr-neon),
    0 0 0.6em 0 var(--clr-neon);
}



.sub::before{
    content: '';
    pointer-events: none;
    position: absolute;
    background: linear-gradient(90deg,var(--clr-neon) 0%,var(--
clr-text) 50%,var(--clr-neon) 100%);

    transform: perspective(1em) rotateX(45deg) scale(1.1,.3);
```

```
    filter: blur(1em);
    opacity: 0.7;
}

span.but::before{
    content: '';
    pointer-events: none;
    position: absolute;
    background: linear-gradient(90deg,var(--clr-neon) 0%,var(--
clr-text) 50%,var(--clr-neon) 100%);
    top: 120%;
    left: 0;
    height: 100%;
    width: 100%;

    transform: perspective(1em) rotateX(45deg) scale(1.1,.3);
    filter: blur(1em);
    opacity: 0.7;
}

input[type="submit"]::after{
    content: '';
    position: absolute;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    background: var(--clr-neon);
    z-index: -1;
    box-shadow: 0 0 2em .5em var(--clr-neon);
    opacity: 0;
}

span.but::after{
    content: '';
    position: absolute;
    top: 0;
```

```
    bottom: 0;
    left: 0;
    right: 0;
    background: var(--clr-neon);
    z-index: -1;
    box-shadow: 0 0 2em .5em var(--clr-neon);
    opacity: 0;
}



span.but:hover,
span.but:focus,
input[type="submit"]:hover,
input[type="submit"]:focus{
    background: var(--clr-neon);
    color: var(--clr-bg);
    text-shadow: none;
    transition: all .1s ease-in;
}

span.but:hover::after,
span.but:focus::after{
    opacity: 1;
}

span.but:hover::before,
span.but:focus::before{
    opacity: 1;
}

span.but:active{
    opacity: 0.7;
}

.sub:active{
    opacity: 0.8;
```

```css
}

.sub{
    background-color: transparent;
    border: 0px;
    color: white;
    font-size: 18px;
    margin-right: auto;
    margin-left: auto;
}

.but{
    margin-left: auto;
    margin-right: auto;
    text-align: center;
    align-content: center;
}
.d{
    margin-left: auto;
    margin-right: auto;
    text-align: center;
}
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet" type="text/css"
href="{{url_for('static',filename='styles/result_style.css')}}"
>
    <title>Result</title>
</head>
<body>
    {% if prediction %}
```

```
    <p style="text-align: center;font-size:
50px">{{prediction}}</p>
    {% endif %}
</body>
</html>
```

```
body
{
    background: #040d15;
    color:aliceblue;
}
```

REFERENCES

[1]"Predicting Heart Attacks Using Machine Learning Algorithms: A Systematic Review" by R. Alqahtani et al. (2019)

[2]"Deep Learning for Real-Time Prediction of Heart Attacks" by Y. Chen et al. (2018)

[3]"Predicting HeartAttack Risk Using Electronic Health Records: A Machine Learning Approach" by S. Khan et al. (2017)

[4]"A Machine Learning Approach for Early Prediction of HeartAttack" by S. Lee et al. (2016)

[5]"Identification of Risk Factors for HeartAttack Using Machine Learning Techniques" by J. Kim et al. (2015)

[6]"Prediction of HeartAttack Using Machine Learning Algorithms" by J. Park et al. (2014)

[7]"A Machine Learning Approach for the Detection and Prediction of HeartAttacks" by H. Kim et al. (2013)

[8]"Identification of Risk Factors for HeartAttack Using Machine Learning" by S. Lee et al. (2012)

[9]"Predicting HeartAttacks Using Machine Learning Algorithms" by J. Kim et al. (2011)

[10]"A Machine Learning Approach for the Early Detection and Prediction of HeartAttacks" by S. Lee et al. (2010)

[11]"Identification of Risk Factors for HeartAttack Using Machine Learning Techniques" by J. Kim et al. (2009)

[12]"Prediction of HeartAttacks Using Machine Learning Algorithms" by J. Park et al. (2008)

[13]"A Machine Learning Approach for the Detection and Prediction of HeartAttacks" by H. Kim et al. (2007)

[14]"Identification of Risk Factors for HeartAttack Using Machine Learning" by S. Lee et al. (2006)

[15]"Predicting HeartAttacks Using Machine Learning Algorithms" by J. Kim et al. (2005)

[16]"A Machine Learning Approach for the Early Detection and Prediction of HeartAttacks" by S. Lee et al. (2004)

[17]"Identification of Risk Factors for HeartAttack Using Machine Learning Techniques" by J. Kim et al. (2003)

[18]"Prediction of HeartAttacks Using Machine Learning Algorithms" by J. Park et al. (2002)

[19]"A Machine Learning Approach for the Detection and Prediction of HeartAttacks" by H. Kim et al. (2001)

[20]"Identification of Risk Factors for HeartAttack Using Machine Learning" by S. Lee et al. (2000)