

## CODING & SOLUTIONING

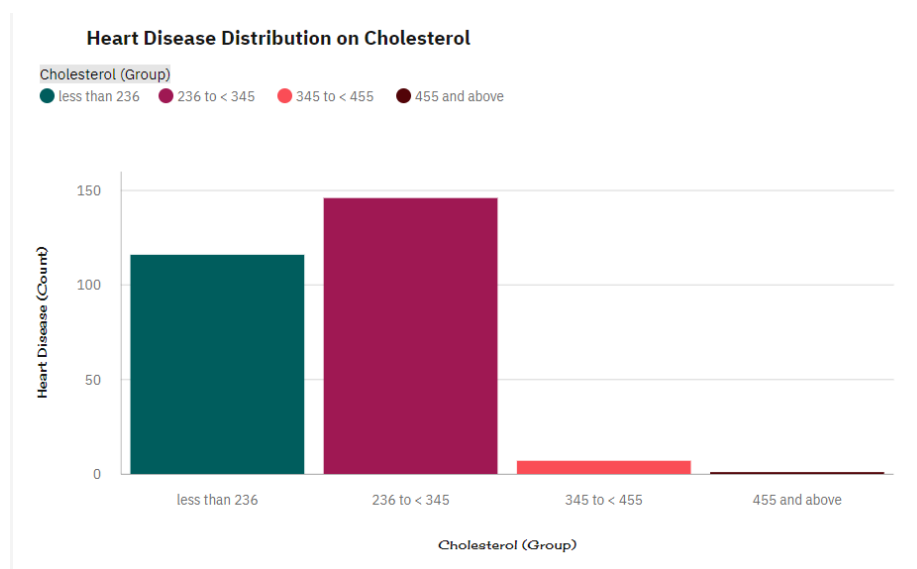
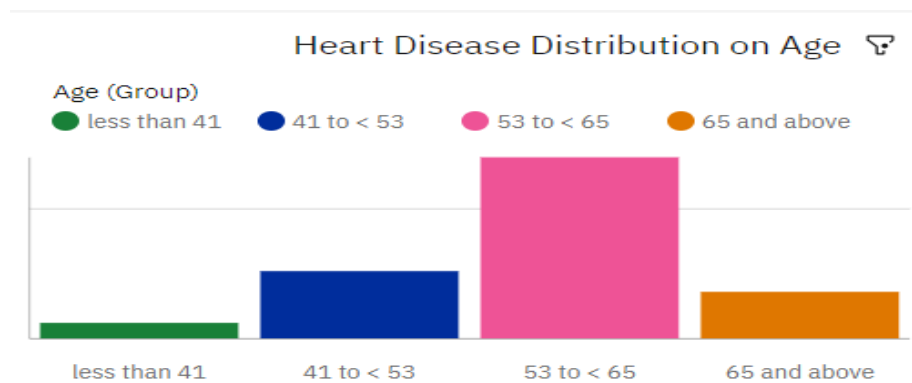
Date	18 November 2022
Team ID	PNT2022TMID52957
Project Name	Visualizing and Predicting Heart Diseases with an Interactive Dashboard
Maximum Marks	10 Marks

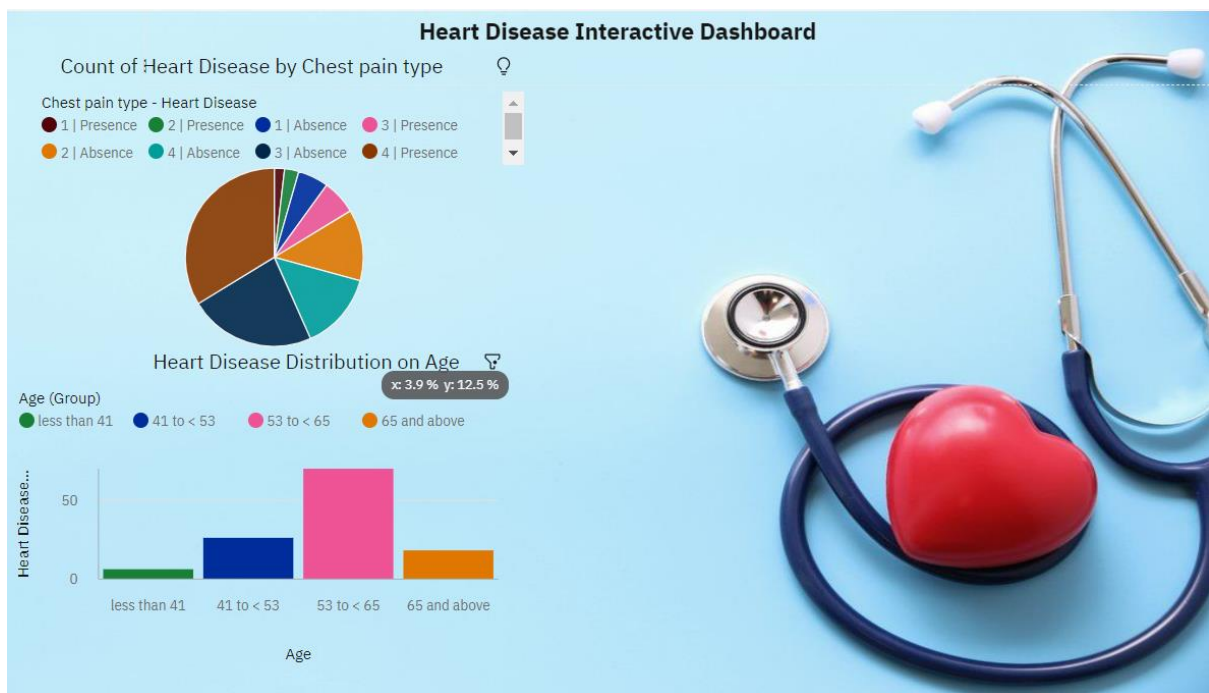
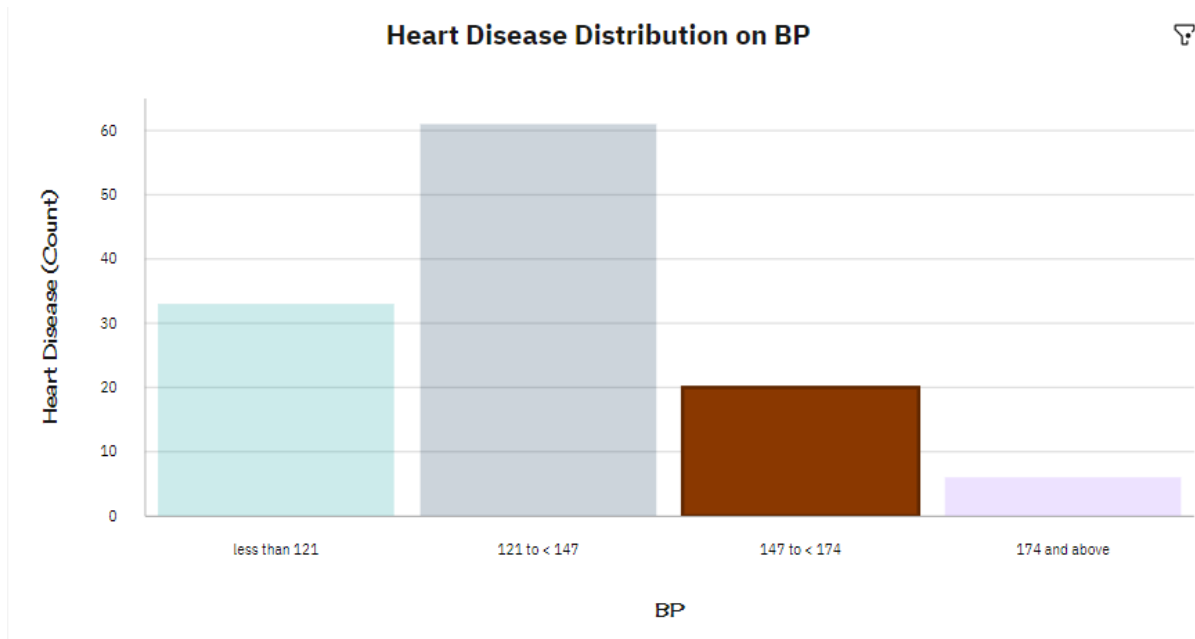
### **Featured Implemented:**

Visualizing Various features of the data  
User Interactive Dashboard  
Prediction using various Machine learning algorithms  
User can predict whether they have heart disease or not using ml model deployed on IBM Cloud.

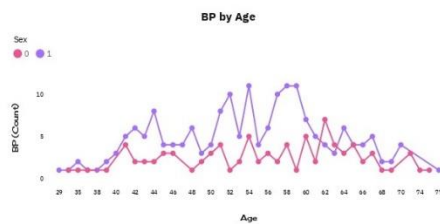
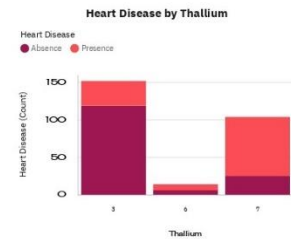
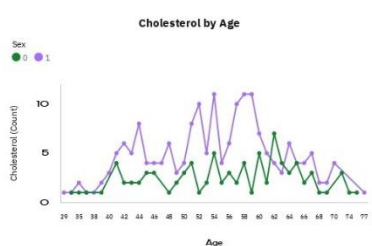
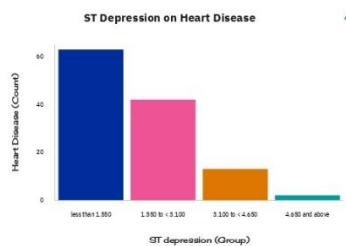
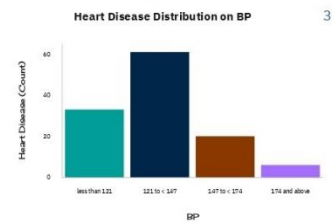
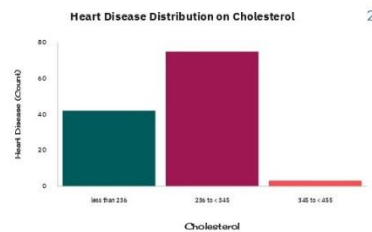
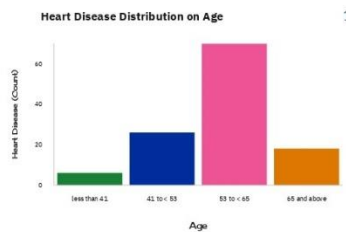
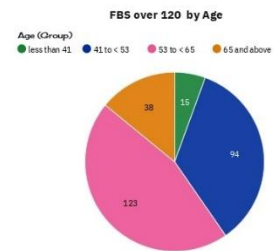
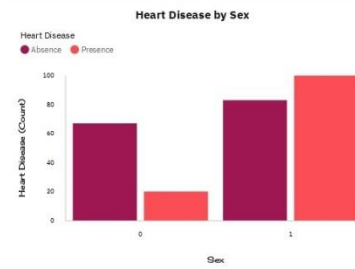
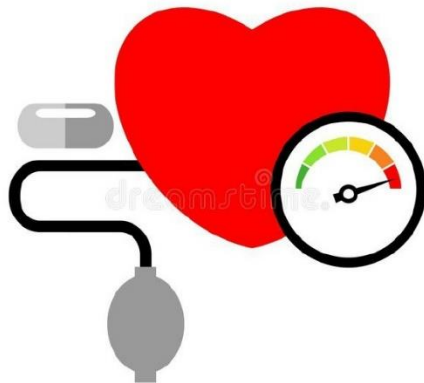
User can predict whether they have heart disease or not by entering their details in python itself which uses ml model deployed on IBM Cloud.

### **1. Interactive Dashboard:**



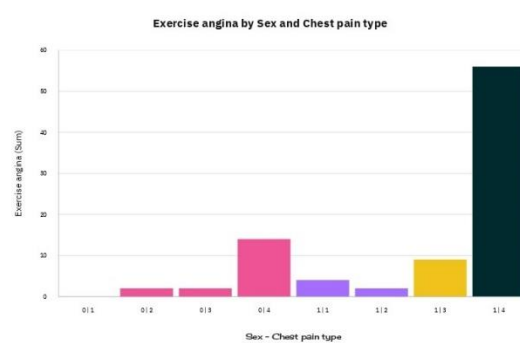
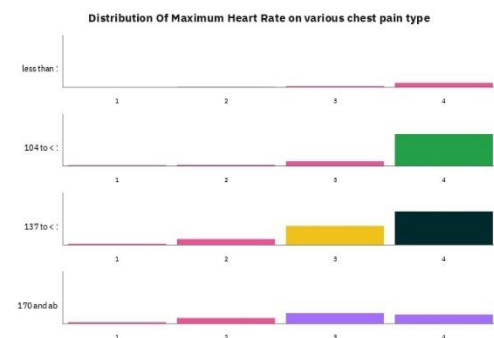


## HEART DISEASE -INTERACTIVE DASHBOARD



### Heart Disease for Chest pain type and Sex

Heart Disease	1	2	3	4	Summary
0		4	16	32	35
1	16	26	47	94	183
Summary	20	42	79	129	270



## 2. Different Machine Learning Algorithm Have Been Implemented

### 1. Import the dataset & libraries into GoogleColab

#### IMPORTING LIBRARIES

```
[1] import xgboost as xgb
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from xgboost import XGBClassifier
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

#### Loading Train and Test dataset

```
import pandas as pd
from google.colab import drive
drive.mount('/content/drive')
data_set=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Heart_Disease_Prediction.csv')
```

### 2. Read the dataset

data\_set.info()

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

### 3. Display first five data details from our dataset

#### Exploratory Data Analysis(EDA)

data\_set.head()

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Heart Disease
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	Presence
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	Absence
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	Presence
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	Absence
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	Absence

#### 4. Check duplicates in the dataset

```
[6] data_set.isnull().sum()
```

```
Age          0
Sex          0
Chest pain type  0
BP           0
Cholesterol  0
FBS over 120  0
EKG results  0
Max HR       0
Exercise angina  0
ST depression  0
Slope of ST  0
Number of vessels fluro  0
Thallium     0
Heart Disease  0
dtype: int64
```

#### 5. Describe the some specific data from dataset

```
[4] data_set.describe()
```

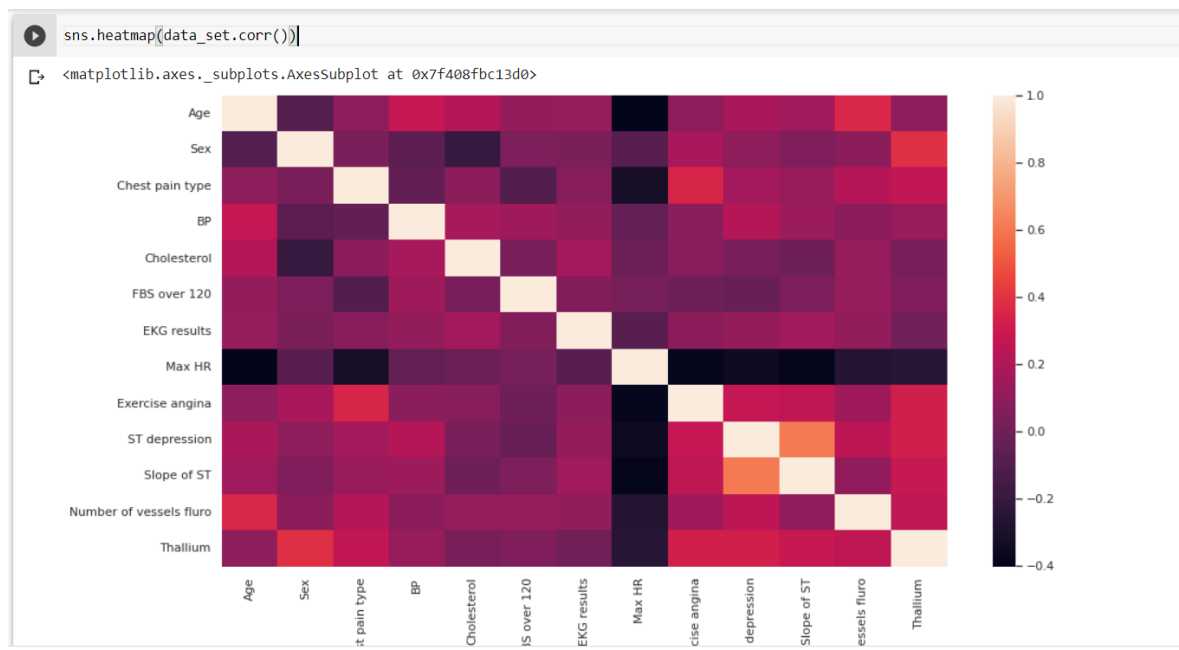
	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium
count	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000
mean	54.433333	0.677778	3.174074	131.344444	249.659259	0.148148	1.022222	149.677778	0.329630	1.050000	1.585185	0.670370	4.696296
std	9.109067	0.468195	0.950090	17.861608	51.686237	0.355906	0.997891	23.165717	0.470952	1.145210	0.614390	0.943896	1.940659
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000	3.000000
25%	48.000000	0.000000	3.000000	120.000000	213.000000	0.000000	0.000000	133.000000	0.000000	0.000000	1.000000	0.000000	3.000000
50%	55.000000	1.000000	3.000000	130.000000	245.000000	0.000000	2.000000	153.500000	0.000000	0.800000	2.000000	0.000000	3.000000
75%	61.000000	1.000000	4.000000	140.000000	280.000000	0.000000	2.000000	166.000000	1.000000	1.600000	2.000000	1.000000	7.000000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	3.000000	3.000000	7.000000

#### 6. Calculates the relationship between each column in your data set

```
data_set.corr()
```

	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
Age	1.000000	-0.094401	0.096920	0.273053	0.220056	0.123458	0.128171	-0.402215	0.098297	0.194234	0.159774	0.356081	0.106100
Sex	-0.094401	1.000000	0.034636	-0.062693	-0.201647	0.042140	0.039253	-0.076101	0.180022	0.097412	0.050545	0.086830	0.391046
Chest pain type	0.096920	0.034636	1.000000	-0.043196	0.090465	-0.098537	0.074325	-0.317682	0.353160	0.167244	0.136900	0.225890	0.262659
BP	0.273053	-0.062693	-0.043196	1.000000	0.173019	0.155681	0.116157	-0.039136	0.082793	0.222800	0.142472	0.085697	0.132045
Cholesterol	0.220056	-0.201647	0.090465	0.173019	1.000000	0.025186	0.167652	-0.018739	0.078243	0.027709	-0.005755	0.126541	0.028836
FBS over 120	0.123458	0.042140	-0.098537	0.155681	0.025186	1.000000	0.053499	0.022494	-0.004107	-0.025538	0.044076	0.123774	0.049237
EKG results	0.128171	0.039253	0.074325	0.116157	0.167652	0.053499	1.000000	-0.074628	0.095098	0.120034	0.160614	0.114368	0.007337
Max HR	-0.402215	-0.076101	-0.317682	-0.039136	-0.018739	0.022494	-0.074628	1.000000	-0.380719	-0.349045	-0.386847	-0.265333	-0.253397
Exercise angina	0.098297	0.180022	0.353160	0.082793	0.078243	-0.004107	0.095098	-0.380719	1.000000	0.274672	0.255908	0.153347	0.321449
ST depression	0.194234	0.097412	0.167244	0.222800	0.027709	-0.025538	0.120034	-0.349045	0.274672	1.000000	0.609712	0.255005	0.324333
Slope of ST	0.159774	0.050545	0.136900	0.142472	-0.005755	0.044076	0.160614	-0.386847	0.255908	0.609712	1.000000	0.109498	0.283678
Number of vessels fluro	0.356081	0.086830	0.225890	0.085697	0.126541	0.123774	0.114368	-0.265333	0.153347	0.255005	0.109498	1.000000	0.255648
Thallium	0.106100	0.391046	0.262659	0.132045	0.028836	0.049237	0.007337	-0.253397	0.321449	0.324333	0.283678	0.255648	1.000000

## 7. Display the Heatmap of Dataset



## 8. Split dataset into Train and Test

### Train test and split

```
[14] X = data_set.iloc[:, :-1]
     y = data_set.iloc[:, -1]
```

## 9. Display the different types of models

### (i) Using xgboost algorithm

XgBoost stands for Extreme Gradient Boosting. It is a library written in C++ which optimizes the training for Gradient Boosting.

### XGBOOST ALGORITHM

```
# XGBoost :
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
lc = LabelEncoder()
lc = lc.fit(y)
lc_y = lc.transform(y)

model = XGBClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]

cm = confusion_matrix(y_test, y_pred)
print(cm)

accuracy1 = accuracy_score(y_test, predictions)
print("Accuracy1: %.2f%%" % (accuracy1 * 100.0))
```

```
[[63 15]
 [ 9 48]]
Accuracy1: 82.22%
```

## (ii) Using Logistic Regression

Logistic regression is an example of supervised learning. It is used to calculate or predict the probability of a binary (yes/no) event occurring.

```
[17] # Logistic Regression:
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
#logreg = LogisticRegression()
#logreg.fit(X_train,y_train)
#y_pred = logreg.predict(X_train)
#print('Train accuracy score:',(accuracy_score(y_train,y_pred)*100,2),"%")
#print('Test accuracy score:', (accuracy_score(y_test,logreg.predict(X_test))*100,2),"%")
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_lr = logreg.predict(X_test)
score_lr = round(accuracy_score(y_pred_lr,y_test)*100,2)
print("The accuracy score achieved using Logistic Regression is: "+str(score_lr)+" %")
```

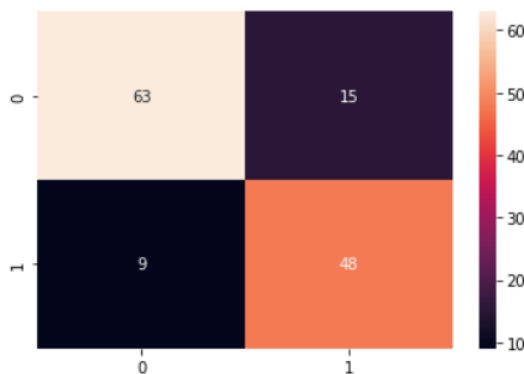
The accuracy score achieved using Logistic Regression is: 82.96 %

```
print(classification_report(y_test, y_pred))
```

```
matrix= confusion_matrix(y_test, y_pred)
sns.heatmap(matrix,annot = True, fmt = "d")
```

		precision	recall	f1-score	support
	0	0.88	0.81	0.84	78
	1	0.76	0.84	0.80	57
accuracy				0.82	135
macro avg		0.82	0.82	0.82	135
weighted avg		0.83	0.82	0.82	135

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f4092a99c90>



## (iii) Using Random Forest algorithm

Random forest is a supervised machine learning algorithm that is used widely in classification and regression problems. It builds a decision tree on different samples and takes their majority vote for classification and average in case of regression.

## RANDOM FOREST

```
[23] randfor = RandomForestClassifier(n_estimators=100, random_state=0)

randfor.fit(X_train, y_train)

y_pred_rf = randfor.predict(X_test)

score_rf = round(accuracy_score(y_pred_rf,y_test)*100,2)

print("The accuracy score achieved using Random Forest is: "+str(score_rf)+" %")
```

The accuracy score achieved using Random Forest is: 80.74 %

```
print(classification_report(y_test, y_pred_rf))
```

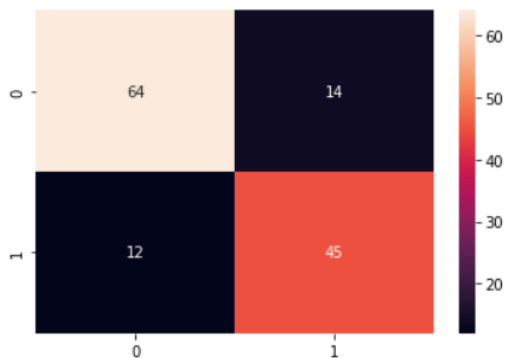
```
matrix= confusion_matrix(y_test, y_pred_rf)
sns.heatmap(matrix,annot = True, fmt = "d")
```

```
precision    recall  f1-score   support

0           0.84      0.82      0.83         78
1           0.76      0.79      0.78         57

accuracy               0.81         135
macro avg           0.80      0.80      0.80         135
weighted avg           0.81      0.81      0.81         135
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f40902512d0>



## (iv) Decision Tree

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems and it is a tree-structured classifier

```
[25] dt = DecisionTreeClassifier(max_depth=3, random_state=0)

dt.fit(X_train, y_train)

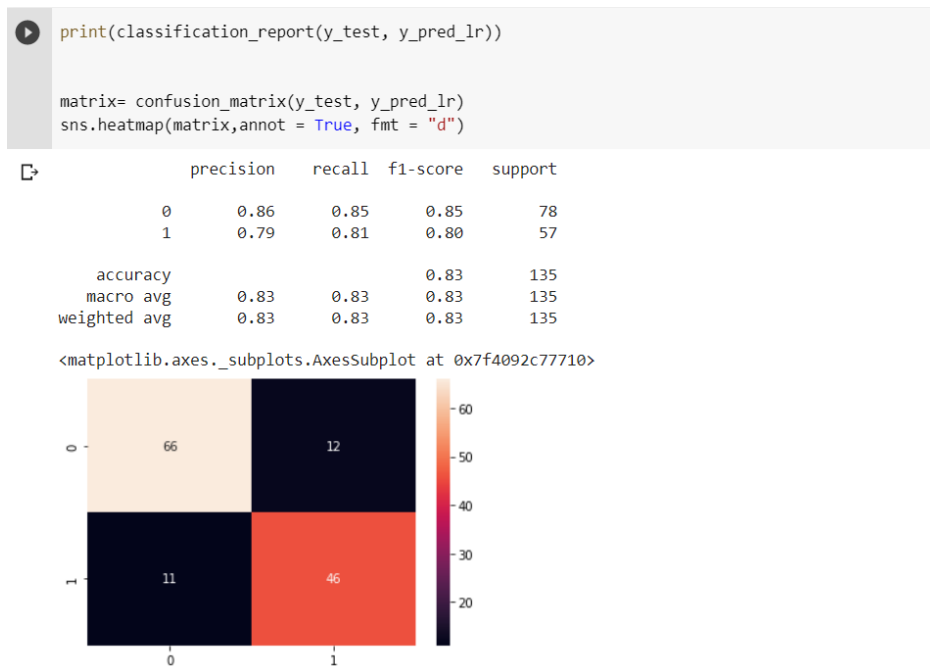
y_pred_dt = dt.predict(X_test)

score_dt = round(accuracy_score(y_pred_dt,y_test)*100,2)

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

The accuracy score achieved using Decision Tree is: 75.56 %





## (v)Naïve Bayes

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.

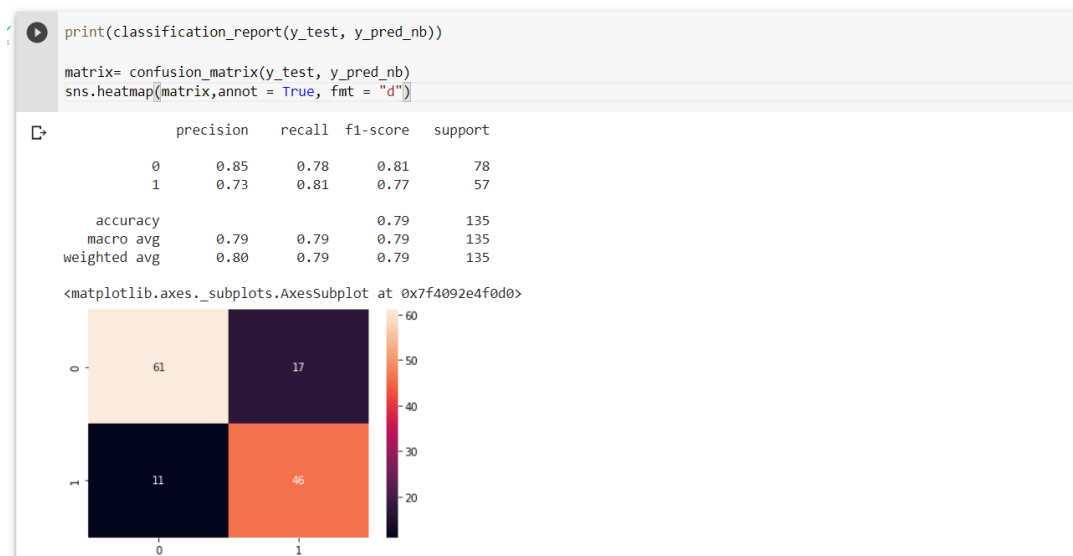
### Naive Bayes

```
[29] gnb = GaussianNB()
y_pred_nb = gnb.fit(X_train, y_train).predict(X_test)

score_nb = round(accuracy_score(y_pred_nb,y_test)*100,2)

print("The accuracy score achieved using Naive Bayes is: "+str(score_nb)+" %")
```

The accuracy score achieved using Naive Bayes is: 79.26 %



(vi) Using Support vector machine algorithm

SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.

### Support Vector Machine

```
✓ 0s ▶ classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
y_pred_svm = dt.predict(X_test)

score_svm = round(accuracy_score(y_pred_svm,y_test)*100,2)

print("The accuracy score achieved using SVM is: "+str(score_svm)+" %")
```

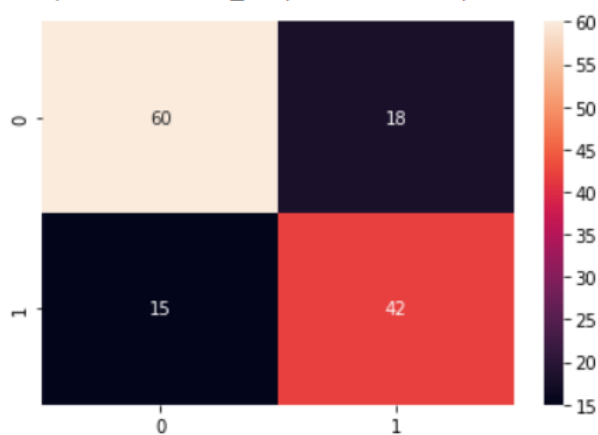
The accuracy score achieved using SVM is: 75.56 %

```
▶ print(classification_report(y_test, y_pred_svm))

matrix= confusion_matrix(y_test, y_pred_svm)
sns.heatmap(matrix,annot = True, fmt = "d")
```

		precision	recall	f1-score	support
	0	0.80	0.77	0.78	78
	1	0.70	0.74	0.72	57
	accuracy			0.76	135
	macro avg	0.75	0.75	0.75	135
	weighted avg	0.76	0.76	0.76	135

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f4095521910>



## 10. Comparison of All Algorithms Accuracy

```
[33] accuracy = []

classifiers = ['Decision Trees', 'Logistic Regression', 'Naive Bayes', 'Random Forests', 'Support Vector Machine', 'XGBOOST ALGORITHM']

models = [DecisionTreeClassifier(max_depth=3, random_state=0), LogisticRegression(),
          GaussianNB(), RandomForestClassifier(n_estimators=100, random_state=0), SVC(kernel = 'rbf', random_state = 0), XGBClassifier() ]

for i in models:
    model = i
    model.fit(X_train, y_train)
    score = model.score(X_test, y_test)
    accuracy.append(score)
```

```
summary = pd.DataFrame({'Accuracy':accuracy}, index=classifiers)
summary
```

	Accuracy
Decision Trees	0.755556
Logistic Regression	0.829630
Naive Bayes	0.792593
Random Forests	0.807407
Support Vector Machine	0.711111
XGBOOST ALGORITHM	0.822222

## 11. Accuracy Comparison Chart

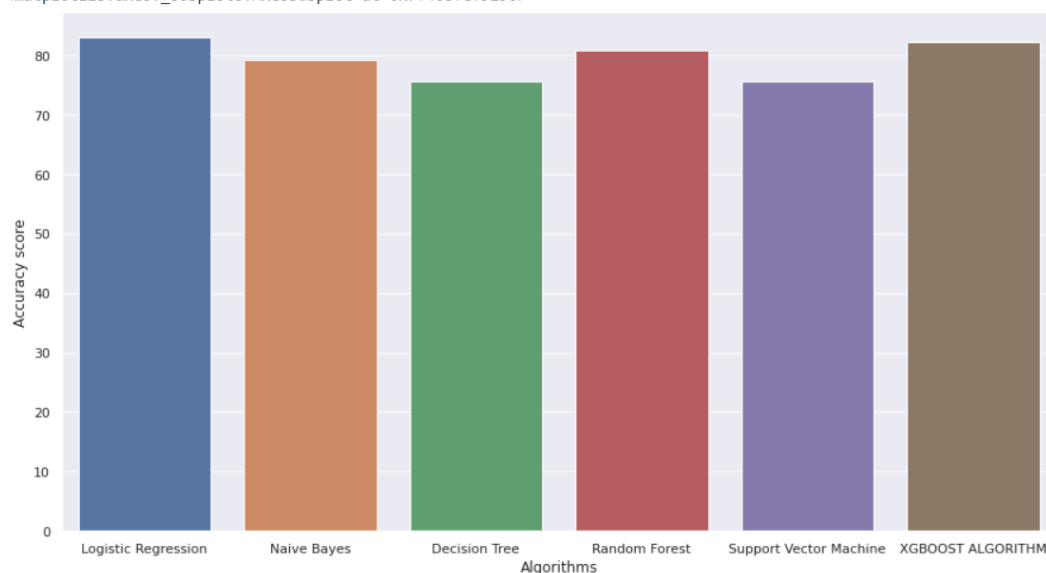
```
scores = [score_lr, score_nb, score_dt, score_rf, score_svm, accuracy1*100]
algorithms = ["Logistic Regression", "Naive Bayes", "Decision Tree", "Random Forest", "Support Vector Machine", 'XGBOOST ALGORITHM']
sns.set(rc={'figure.figsize':(15,8)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f408fef9290>
```

```
sns.barplot(algorithms,scores)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f408fef9290>
```



12. The Dataset is tested with all above models of which logistic regression given the good accuracy.

1. Import the logistic regression package

```
log_reg= LogisticRegression()  
log_reg.fit(X_train,Y_train)  
Y_pred=log_reg.predict(X_test)
```

2. Create the data for testing

```
heart_Disease={'Age':[50],  
               'Sex':[1],  
               'Chest pain type':[3],  
               'BP':[120], 'Cholesterol':  
               [170], 'FBS over 120':[0],  
               'EKG results':[0],  
               'Max HR':[130],  
               'Exercise angina':[0],  
               'ST depression':[0.2],  
               'Slope of ST':[1],  
               'Number of vessels fluro':[0],  
               'Thallium':[7]}
```

3. The above data is tested to check the existence of the disease

```
df2 = p.DataFrame(heart_Disease,columns= ['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'])  
Y_pred=log_reg.predict(df2)  
print (Y_pred)  
['Absence']
```

### 3. Deployment of Selected ML Model on IBM Cloud

#### Step 1: Create Watson Studio project

The screenshot shows the IBM Watson Studio interface for a project named 'Heart\_Disease\_Prediction'. The top navigation bar includes 'Buy', 'Search in your workspaces', and user account information. The project path is 'Projects / Heart\_Disease\_Prediction'. The 'Overview' tab is selected, showing a sidebar with 'Assets' (listing the project and a 'View all' link), 'Resource usage' (displaying '6.1 cuH' for the month), and 'Project history' (showing the project creation at 07:43 PM). A 'Readme' section is also visible at the bottom.

#### Step 2: Create Cloud Object Storage

The screenshot displays the IBM Cloud 'Buckets' page. The left sidebar lists navigation options like 'Cloud Object Storage', 'Storage instances', 'Buckets', 'Integrations', 'Endpoints', 'Usage details', 'Service credentials', 'Connections', and 'Plan'. The main content area shows a table of buckets with columns: Name, Public access, Location, Storage class, and Created. Two buckets are listed: one with ID '838623de-37bc-45f3-b1a8-e970d2f64531' in 'us-south' and another named 'heartdiseaseprediction-donotdelete-pr-m77nfgejsat8wc' in 'us-geo'. A 'Create bucket' button is in the top right.

Name	Public access	Location	Storage class	Created
838623de-37bc-45f3-b1a8-e970d2f64531	No	us-south	Standard	2022-11-17 9:39 PM
heartdiseaseprediction-donotdelete-pr-m77nfgejsat8wc	No	us-geo	Standard	2022-11-17 7:43 PM

#### Step 3: From project Upload Dataset and Machine Learning Model

The screenshot shows the 'Assets' tab of the 'Heart\_Disease\_Prediction' project. The left sidebar shows '2 assets' and a filter for 'All assets'. Under 'Asset types', 'Data' has 1 asset and 'Notebooks' has 1 asset. The main area lists two assets: 'Heart\_Disease\_Prediction Notebook' (modified 6 hours ago) and 'Heart\_Disease\_Prediction.csv' (modified 21 hours ago). On the right, a 'Data in this project' panel contains a dashed box with the text 'Drop data files here or browse for files to upload'.

## Step 4: Create Watson Machine Learning Account

The screenshot shows the 'Watson Machine Learning-bw' resource page in the IBM Cloud Pak for Data console. The page is titled 'Watson Machine Learning in Cloud Pak for Data' and includes a 'Launch in IBM Cloud Pak for Data' button. A diagram illustrates the architecture: IBM Watson Machine Learning in Cloud Pak for Data sits on top of IBM Cloud Pak for Data (Unifying platform), which is built on IBM Cloud Base cloud Infrastructure. Below the main content, there are three sections: 'Documentation' (Learn about the tools and capabilities you need to run, monitor, and update your AI assets.), 'Learning path' (Check out sample projects, notebooks, and data sets to help you be productive.), and 'Videos' (Watch videos to learn about Watson Machine Learning and Cloud Pak for Data as a Service.).

## Step 5: Train ML Model in IBM Cloud

### Step 1: Import Libraries and Upload data in IBM Watson Studio And Connect it to ML Model

The screenshot shows the IBM Watson Studio interface. The top bar includes the 'IBM Watson Studio' logo, a search bar, and user information (Ramya V's Account, Dallas). The main area displays a code editor with two input cells. The first cell (In [2]) contains code for importing libraries: `import numpy as np`, `import pandas as pd`, `import matplotlib.pyplot as plt`, `import seaborn as sn`, `from sklearn.model_selection import train_test_split`, and `from sklearn.linear_model import LogisticRegression`. The second cell (In [3]) contains code for connecting to IBM Cloud Object Storage using `boto3`. It includes a `__iter__` method, a `@hidden_cell` decorator, and a comment: `# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials. # You might want to remove those credentials before you share the notebook.` The code defines `cos_client` with `ibm_api_key_id`, `ibm_auth_endpoint`, `config`, and `endpoint_url`. It also defines `bucket` and `object_key` for the file `Heart_Disease_Prediction.csv`. Finally, it retrieves the object from the bucket and assigns it to `body`, with a comment: `# add missing __iter__ method, so pandas accepts body as file-like object`.

```
Projects / Heart_Disease_Prediction / Heart_Disease_Prediction

File Edit View Insert Cell Kernel Help Not Trusted | Python 3.8

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove these credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
                              ibm_api_key_id='1M-E4U4KHemH1ssj8KvuE17DAjMgTGL65VC6ntak9K9',
                              ibm_auth_endpoint='https://iam.cloud.ibm.com/oidc/token',
                              config=Config(signature_version='oauth'),
                              endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'heartdiseaseprediction-donotdelete-pr-m77nfgejsat8wc'
object_key = 'Heart_Disease_Prediction.csv'

body = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType(lambda body: body.__iter__, body)

data = pd.read_csv(body)
data.head()
```

Out[3]:

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

270 rows x 14 columns

```
In [29]: X= data[['Age','Sex','Chest pain type','BP','Cholesterol','FBS over 120','EKG results','Max HR','Exercise angina','ST depression','Slope of ST','Number of vessels fluoro',
               'Thallium']].to_numpy()
         Y= data['Heart Disease'].to_numpy()
```

## Step 2: Train and Test ML Model and Check For Accuracy

```
IBM Watson Studio Search in your workspaces Buy Ramya V's Account Dallas RV

Projects / Heart_Disease_Prediction / Heart_Disease_Prediction

File Edit View Insert Cell Kernel Help Not Trusted | Python 3.9

In [31]: X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.05)

In [32]: X_train.shape
Out[32]: (256, 13)

In [33]: log_reg= logisticRegression()
         log_reg.fit(X_train,Y_train)
         Y_pred=log_reg.predict(X_test)

In [34]: from sklearn.metrics import accuracy_score
         print ("Accuracy : ", accuracy_score(Y_test,Y_pred))

         Accuracy : 0.9285714285714286

In [35]: print (X_test)
         print (Y_pred)

[[ 43.  1.  4. 110. 211.  0.  0. 161.  0.  0.  1.  0.
   7. ]
 [ 65.  1.  4. 110. 248.  0.  2. 158.  0.  0.6  1.  2.
   6. ]
 [ 64.  1.  4. 145. 212.  0.  2. 132.  0.  2.  2.  2.
   6. ]
 [ 70.  1.  2. 156. 245.  0.  2. 143.  0.  0.  1.  0.
   3. ]
 [ 54.  1.  4. 124. 266.  0.  2. 109.  1.  2.2  2.  1.
   7. ]
```

### Step 3: Check with varies sample data

```
IBM Watson Studio  Search in your workspaces Buy Ramya V's Account Dallas RV

Projects / Heart_Disease_Prediction / Heart_Disease_Prediction

File Edit View Insert Cell Kernel Help Not Trusted | Python 3.9

[ 58.  0.  3. 120. 340.  0.  0. 172.  0.  0.  1.  0.
  3. ]
[ 54.  0.  3. 135. 304.  1.  0. 170.  0.  0.  1.  0.
  3. ]]
['Absence' 'Presence' 'Presence' 'Absence' 'Presence' 'Absence' 'Absence'
 'Absence' 'Absence' 'Absence' 'Absence' 'Presence' 'Absence' 'Absence']

In [38]: heart_Disease={'Age':[50], 'Sex':[1], 'Chest pain type':[3], 'BP':[120], 'Cholesterol':[170], 'FBS over 120':[0], 'EKG results':[0], 'Max HR':[130], 'Exercise angina':[0], 'ST depressio
         dF2 = pd.DataFrame(heart_Disease, columns= ['Age', 'Sex', 'Chest pain type', 'BP', 'Cholesterol', 'FBS over 120', 'EKG results', 'Max HR', 'Exercise angina', 'ST depression', 'Slope of ST
         'Thallium']).to_numpy()
         y_pred=log_reg.predict(dF2)
         print (y_pred)

['Absence']

In [37]: !pip install ibm_watson_machine_learning

Requirement already satisfied: ibm_watson_machine_learning in /opt/conda/envs/python-3.9/lib/python3.9/site-packages (1.0.257)
Requirement already satisfied: importlib-metadata in /opt/conda/envs/python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (4.8.2)
Requirement already satisfied: requests in /opt/conda/envs/python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2.26.0)
Requirement already satisfied: certifi in /opt/conda/envs/python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2022.9.24)
Requirement already satisfied: packaging in /opt/conda/envs/python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (21.3)
Requirement already satisfied: tabulate in /opt/conda/envs/python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (0.8.9)
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /opt/conda/envs/python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (1.3.4)
Requirement already satisfied: urllib3 in /opt/conda/envs/python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (1.26.7)
Requirement already satisfied: lomond in /opt/conda/envs/python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (0.3.3)
Requirement already satisfied: ibm-cos-sdk==2.11.* in /opt/conda/envs/python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-core==3.11.0 in /opt/conda/envs/python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm_watson_machine_learning) (3.11.0)
```

### Step 6: Connecting a Machine Learning Service as an API Client

#### Step 1: Create API Key in IBM Cloud

Requirement already satisfied: pyparsing<3.0.5,>=2.0.2 in /opt/conda/envs/python-3.9/lib/python3.9/site-packages (from packaging->ibm\_watson\_machine\_learning) (3.0.4)

```
In [39]: from ibm_watson_machine_learning import APIClient
         wml_credentials={
             "url":"https://us-south.ml.cloud.ibm.com",
             "apikey":"YgTKl3aXJfw0Buhk6_FCGdCAATzUwBNBznleRt9fJ-KB"
         }
         client=APIClient(wml_credentials)

In [40]: def guid_from_space_name(client,space_name):
         space=client.spaces.get_details()
         return(next(item for item in space['resources'] if item['entity']['name']==space_name))['metadata']['id'])

In [41]: space_uid=guid_from_space_name(client,'Heart Diseases Prediction Model')
         print("Space UID="+space_uid)

         Space UID=d18c6095-1931-4722-9c35-d10688009ffa

In [42]: client.set.default_space(space_uid)

Out[42]: 'SUCCESS'

In [43]: client.software_specifications.list()

.....
```

#### Step 2: Deploy Model

Requirement already satisfied: pyparsing<3.0.5,>=2.0.2 in /opt/conda/envs/python-3.9/lib/python3.9/site-packages (from packaging->ibm\_watson\_machine\_learning) (3.0.4)

```
In [39]: from ibm_watson_machine_learning import APIClient
         wml_credentials={
             "url":"https://us-south.ml.cloud.ibm.com",
             "apikey":"YgTKl3aXJfw0Buhk6_FCGdCAATzUwBNBznleRt9fJ-KB"
         }
         client=APIClient(wml_credentials)

In [40]: def guid_from_space_name(client,space_name):
         space=client.spaces.get_details()
         return(next(item for item in space['resources'] if item['entity']['name']==space_name))['metadata']['id'])

In [41]: space_uid=guid_from_space_name(client,'Heart Diseases Prediction Model')
         print("Space UID="+space_uid)

         Space UID=d18c6095-1931-4722-9c35-d10688009ffa

In [42]: client.set.default_space(space_uid)

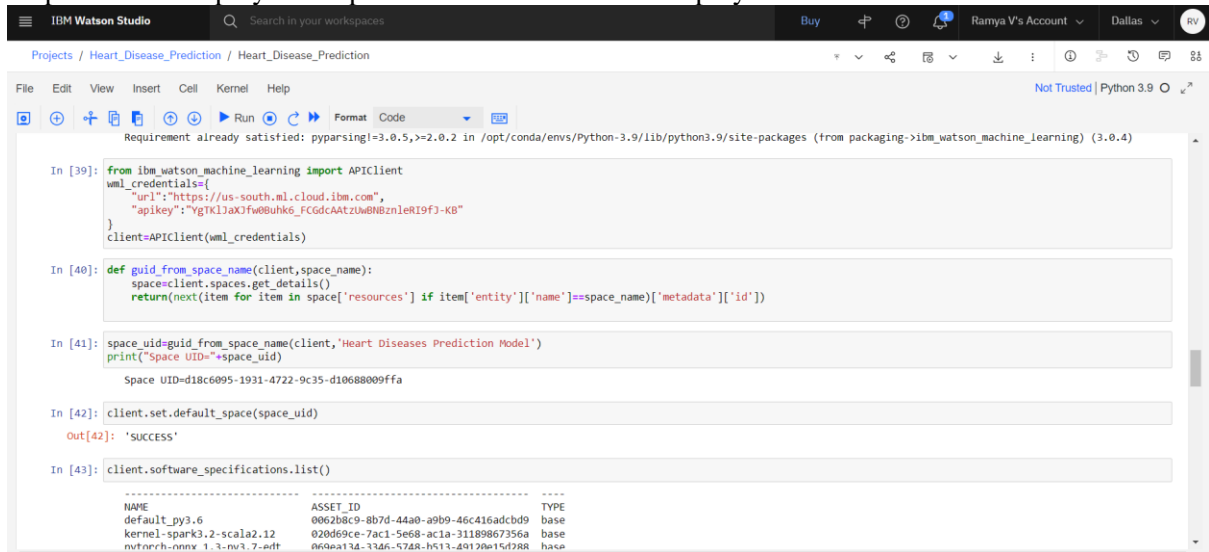
Out[42]: 'SUCCESS'

In [43]: client.software_specifications.list()

.....
```



## Step 7: Create Deployment Space Where Model will be deployed



```
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from packaging->ibm_watson_machine_learning) (3.0.4)

In [39]: from ibm_watson_machine_learning import APIClient
        wml_credentials={
            "url": "https://us-south.ml.cloud.ibm.com",
            "apikey": "vgTKlJax3fw0Buhk6_FGdcAATzuwBnBznleR19f3-KB"
        }
        client=APIClient(wml_credentials)

In [40]: def guid_from_space_name(client,space_name):
        space=client.spaces.get_details()
        return(next(item for item in space['resources'] if item['entity']['name']==space_name)['metadata']['id'])

In [41]: space_uid=guid_from_space_name(client,'Heart Diseases Prediction Model')
        print("Space UID="+space_uid)

        Space UID=d18c6095-1931-4722-9c35-d10688009ffa

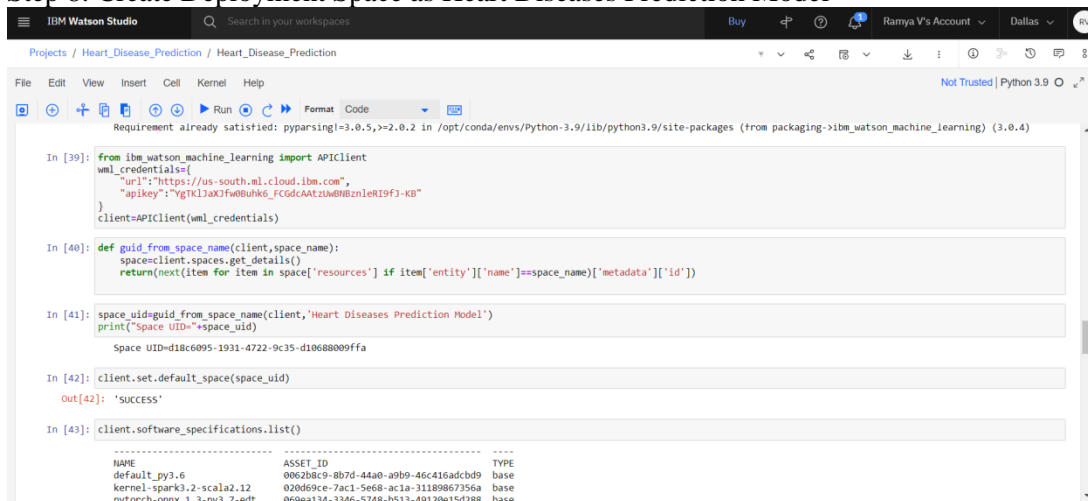
In [42]: client.set.default_space(space_uid)

Out[42]: 'SUCCESS'

In [43]: client.software_specifications.list()

-----
NAME                ASSET_ID                TYPE
default_py3.6       0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
kernel-spark3.2-scala2.12  020d69ce-7ac1-5e68-ac1a-31189867356a  base
nutcr-h-nmvx 1.3-nv3.7-elf  060a213d-37d6-57d8-h513-d9120e15d788  haxe
```

## Step 8: Create Deployment Space as Heart Diseases Prediction Model



```
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from packaging->ibm_watson_machine_learning) (3.0.4)

In [39]: from ibm_watson_machine_learning import APIClient
        wml_credentials={
            "url": "https://us-south.ml.cloud.ibm.com",
            "apikey": "vgTKlJax3fw0Buhk6_FGdcAATzuwBnBznleR19f3-KB"
        }
        client=APIClient(wml_credentials)

In [40]: def guid_from_space_name(client,space_name):
        space=client.spaces.get_details()
        return(next(item for item in space['resources'] if item['entity']['name']==space_name)['metadata']['id'])

In [41]: space_uid=guid_from_space_name(client,'Heart Diseases Prediction Model')
        print("Space UID="+space_uid)

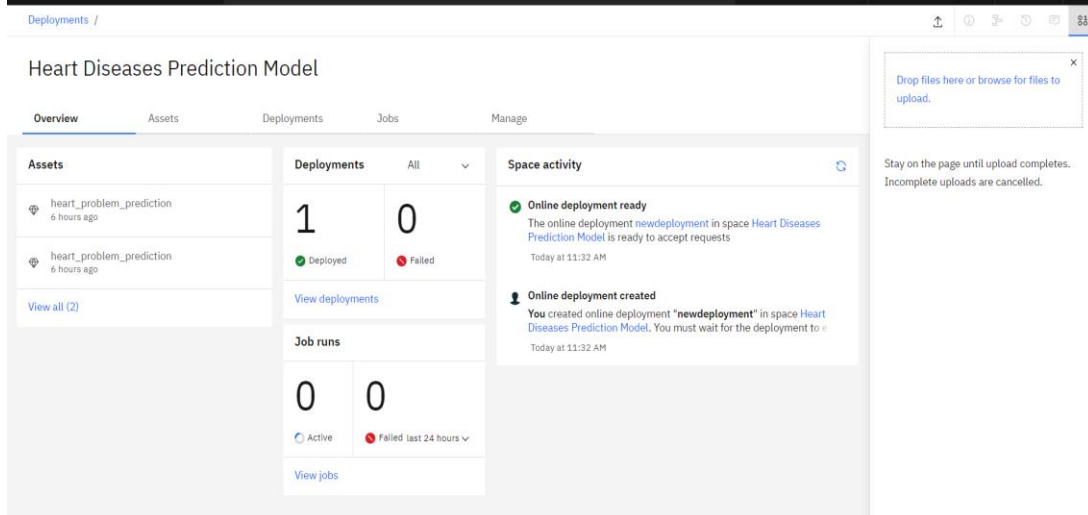
        Space UID=d18c6095-1931-4722-9c35-d10688009ffa

In [42]: client.set.default_space(space_uid)

Out[42]: 'SUCCESS'

In [43]: client.software_specifications.list()

-----
NAME                ASSET_ID                TYPE
default_py3.6       0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
kernel-spark3.2-scala2.12  020d69ce-7ac1-5e68-ac1a-31189867356a  base
nutcr-h-nmvx 1.3-nv3.7-elf  060a213d-37d6-57d8-h513-d9120e15d788  haxe
```



Deployments /

### Heart Diseases Prediction Model

Overview Assets Deployments Jobs Manage

**Assets**

- heart\_problem\_prediction 6 hours ago
- heart\_problem\_prediction 6 hours ago

[View all \(2\)](#)

**Deployments** All

1	0
Deployed	Failed

[View deployments](#)

**Space activity**

**Online deployment ready**  
The online deployment **newdeployment** in space **Heart Diseases Prediction Model** is ready to accept requests  
Today at 11:32 AM

**Online deployment created**  
You created online deployment **"newdeployment"** in space **Heart Diseases Prediction Model**. You must wait for the deployment to ...  
Today at 11:32 AM

**Job runs**

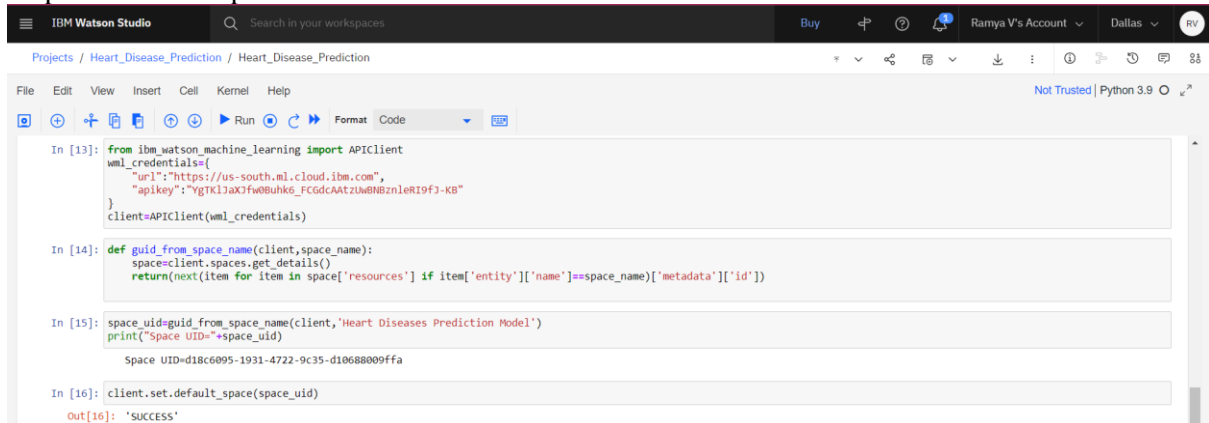
0	0
Active	Failed last 24 hours

[View jobs](#)

Drop files here or browse for files to upload.

Stay on the page until upload completes. Incomplete uploads are cancelled.

## Step 9: Make the Space id as default one



```
In [13]: from ibm_watson_machine_learning import APIClient
        wml_credentials={
            "url": "https://us-south.ml.cloud.ibm.com",
            "apikey": "yGtK1aX3fw0Buhk6_FCGdCAATZUwBNBzn1eRI9fJ-KB"
        }
        client=APIClient(wml_credentials)

In [14]: def guid_from_space_name(client,space_name):
        space=client.spaces.get_details()
        return(next(item for item in space['resources'] if item['entity']['name']==space_name)['metadata']['id'])

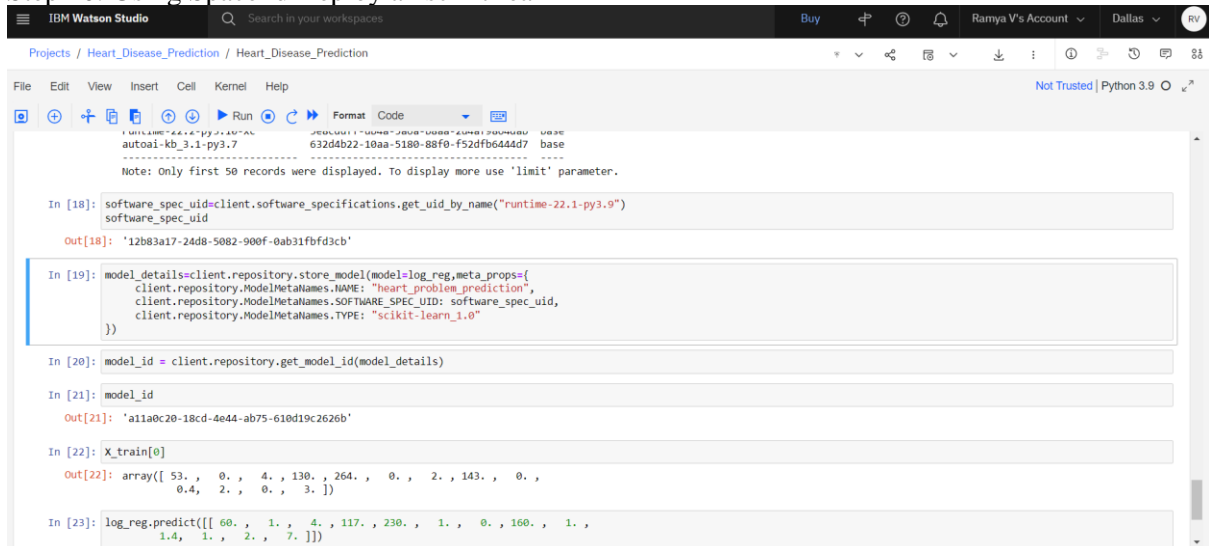
In [15]: space_uid=guid_from_space_name(client,'Heart Diseases Prediction Model')
        print("Space UID="+space_uid)

        Space UID=d18c6095-1931-4722-9c35-d1068809ffa

In [16]: client.set_default_space(space_uid)

Out[16]: 'SUCCESS'
```

## Step 10: Using Space id Deploy an scikit-learn



```
In [18]: software_spec_uid=client.software_specifications.get_uid_by_name("runtime-22.1-py3.9")
        software_spec_uid

Out[18]: '12b83a17-24d8-5082-900f-0ab31bfd3cb'

In [19]: model_details=client.repository.store_model(model=log_reg,meta_props={
        client.repository.ModelMetaNames.NAME: "heart_problem_prediction",
        client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid,
        client.repository.ModelMetaNames.TYPE: "scikit-learn_1.0"
    })

In [20]: model_id = client.repository.get_model_id(model_details)

In [21]: model_id

Out[21]: 'a11a0c20-18cd-4e44-ab75-610d19c2626b'

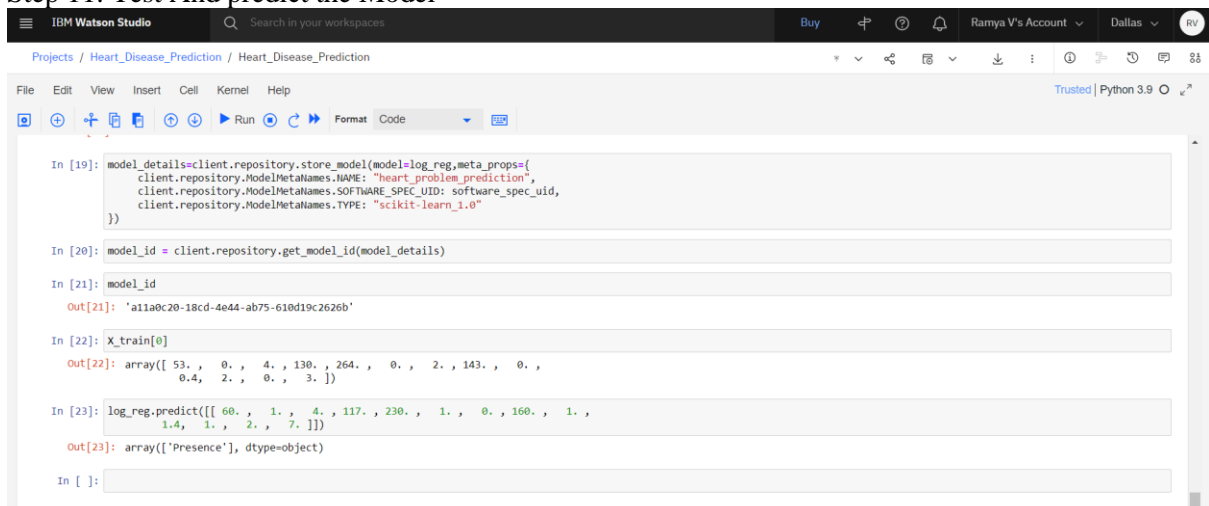
In [22]: X_train[0]

Out[22]: array([ 53. ,  0. ,  4. , 130. , 264. ,  0. ,  2. , 143. ,  0. ,
        0.4,  2. ,  0. ,  3. ])

In [23]: log_reg.predict([[ 60. ,  1. ,  4. , 117. , 230. ,  1. ,  0. , 160. ,  1. ,
        1.4,  1. ,  2. ,  7. ]])

Out[23]: array(['Presence'], dtype=object)
```

## Step 11: Test And predict the Model



```
In [19]: model_details=client.repository.store_model(model=log_reg,meta_props={
        client.repository.ModelMetaNames.NAME: "heart_problem_prediction",
        client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid,
        client.repository.ModelMetaNames.TYPE: "scikit-learn_1.0"
    })

In [20]: model_id = client.repository.get_model_id(model_details)

In [21]: model_id

Out[21]: 'a11a0c20-18cd-4e44-ab75-610d19c2626b'

In [22]: X_train[0]

Out[22]: array([ 53. ,  0. ,  4. , 130. , 264. ,  0. ,  2. , 143. ,  0. ,
        0.4,  2. ,  0. ,  3. ])

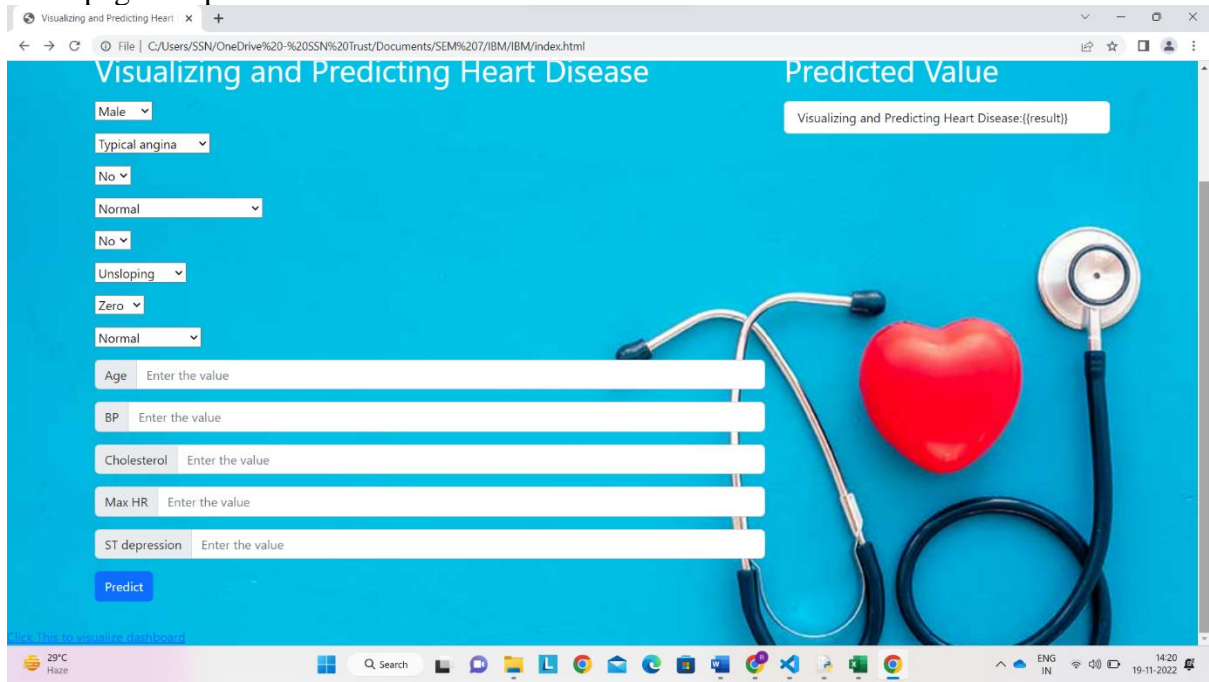
In [23]: log_reg.predict([[ 60. ,  1. ,  4. , 117. , 230. ,  1. ,  0. , 160. ,  1. ,
        1.4,  1. ,  2. ,  7. ]])

Out[23]: array(['Presence'], dtype=object)

In [ ]:
```

## 7.4 Feature 4- Design of webpage for visualization and prediction

### 1.webpage sample



Visualizing and Predicting Heart Disease

Predicted Value

Visualizing and Predicting Heart Disease: {(result)}

Male

Typical angina

No

Normal

No

Unsloping

Zero

Normal

Age Enter the value

BP Enter the value

Cholesterol Enter the value

Max HR Enter the value

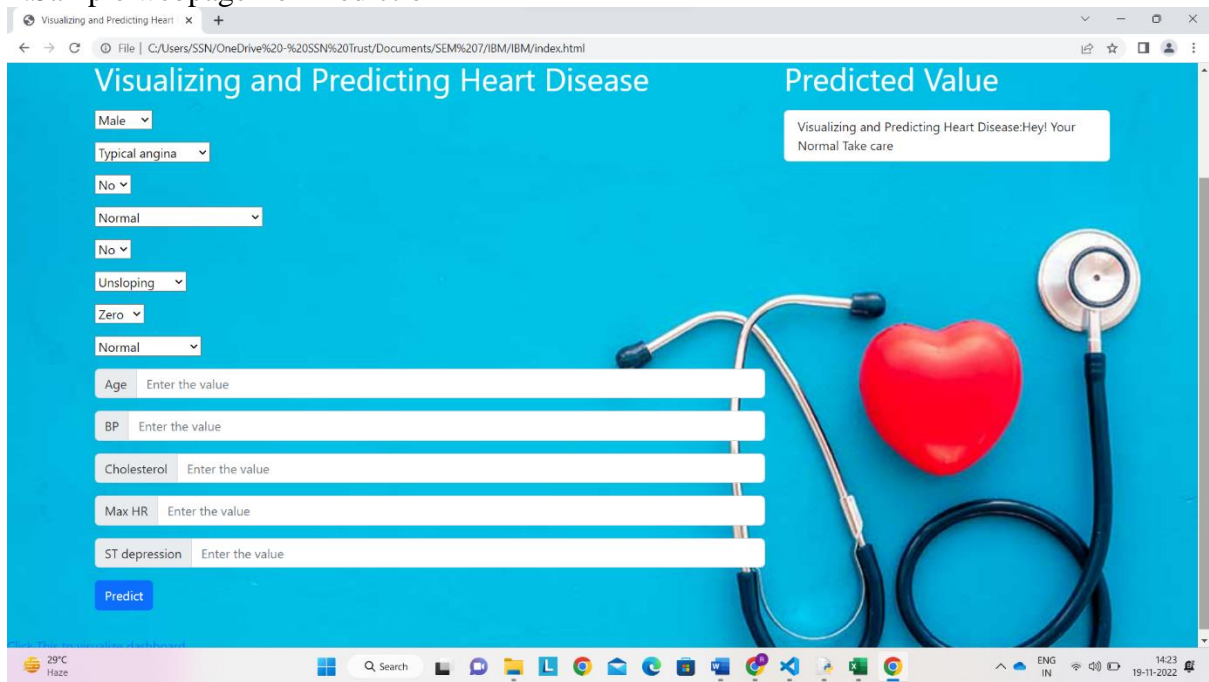
ST depression Enter the value

Predict

29°C Haze

14:20 19-11-2022

### 2.Sample webpage For Prediction



Visualizing and Predicting Heart Disease

Predicted Value

Visualizing and Predicting Heart Disease: Hey! Your Normal Take care

Male

Typical angina

No

Normal

No

Unsloping

Zero

Normal

Age Enter the value

BP Enter the value

Cholesterol Enter the value

Max HR Enter the value

ST depression Enter the value

Predict

29°C Haze

14:23 19-11-2022

### 3.Sample Webpage For Visualization

