4.2SAMPLE CODE

```
#import the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score
from sklearn.pipeline import make_pipeline, Pipeline from sklearn.model_selection
import GridSearchCV

from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.externals import joblib
from sklearn.metrics import make_scorer, f1_score, recall_score, precision_score
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.metrics import log_loss
import warnings
warnings.simplefilter(action = 'ignore', category= FutureWarning)


from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
import numpy as np
from flask import Flask,request,jsonify, render_template
import pickle
app=Flask(_name_,template_folder='template')
app._static_folder = 'static'
model1=pickle.load(open('model1.pkl','rb'))
model2=pickle.load(open('model2.pkl','rb'))
@app.route('/home')
def homepage():
  return render_template('index.html')
@app.route('/precautions')
def precautions():
  return render_template('precautions.html')
@app.route('/advancedpage')
def advancedpage():
  return render_template('index.html')
@app.route('/quick',methods=['POST'])
def quick():
      def bmi(height,weight):
            bmi=int(weight)/((int(height)/100)**2)
            return bmi
      int_features1 = [float(x) for x in request.form.values()]
```

```
age=int_features1[1]
cigs=int_features1[3]
height=int_features1[8]
weight=int_features1[9]
hrv=int_features1[10]
int_features1.pop(8)
int_features1.pop(9)
bmi=round(bmi(height,weight),2)
int_features1.insert(8,bmi)

if int(int_features1[0])==1.0:
        sex="Male"
else:
        sex="Female"
if int(int_features1[2])==1.0:
        smoking="Yes"
else:
        smoking="No"
if int(int_features1[4])==1.0:
        stroke="Yes"
else:
        stroke="No"

if int(int_features1[5])==1.0:
        hyp="Yes"
else:
        hyp="No"
if int(int_features1[7])==1.0:
        dia="Yes"
else:
        dia="No"
if int(int_features1[6])==1.0:
        bpmeds="Yes"
```

```
    else:
        bpmeds="No"



    final_feature1=[np.array(int_features1)]
    prediction1= model1.predict(final_feature1)
    result=prediction1[0]

    if result==0:
        result="No need to worry"
    else:
        result="You are detected with heart problems. You need to consult a doctor immediately"
    return render_template('quick_report.html',prediction_text1= result,gender=sex,age=age,smoking=smoking,cigs=cigs,stroke=stroke,hyp=hyp,dia=dia,bpmeds=bpmeds,bmi=bmi,hrv=hrv)


@app.route('/quickpage')
def quickpage():
    return render_template('index1.html')


@app.route('/customersupport')
def customersupport():
    return render_template('customercare.html')
@app.route('/Doctorconsult')
def Doctorconsult():
    return render_template('Doctorconsult.html')


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


@app.route('/advanced',methods=['POST'])
def advanced():
        int_features2 = [int(x) for x in request.form.values()]

        final2_feature=[np.array(int_features2)] prediction2=

        model2.predict(final2_feature) result=prediction2[0]

 age=int_features2[0] trestbps=int_features2[3]

chol=int_features2[4] oldspeak=int_features2[7]

thalach=int_features2[7] ca=int_features2[10]

if int(int_features2[1])==1:sex="Male"

        else:

            sex="Female"


        if int(int_features2[2])==1:

            cp="Typical angina"

         elif int(int_features2[2])==2:

cp="Atypical angina"

elif int(int_features2[2])==3:

cp="Non-angina pain"

        else:

            cp="Asymtomatic"



        if int(int_features2[5])==1:

            fbs="Yes"

        else:

            fbs="No"


        if int(int_features2[6])==1:
            restecg="ST-T wave abnormality"
        elif int(int_features2[6])==2:
            restecg="showing probable or definite left ventricular hypertrophy by
    Estes"


        else:
            restecg="Normal"


        if int(int_features2[8])==1:
            exang="Yes"
        else:
            exang="No"


        if int(int_features2[9])==1:
            slope="upsloping"
        elif int(int_features2[9])==2:
            slope="flat"
        else:
            slope="downsloping"


        if int(int_features2[11])==3:
            thal="Normal"
        elif int(int_features2[11])==6:
            thal="Fixed defect"
        else:
            thal=" reversable defect"

if result==0:
            result="No need to worry"
        else:
            result="You are detected with heart problems. You need to consult
    a doctor immediately"
        return render_template('advance_report.html',prediction_text2=
    result,age=age,sex=sex,cp=cp,trestbps=trestbps,chol=chol,fbs=fbs,restecg=restecg,old
    peak=oldspeak,exang=exang,slope=slope,ca=ca,thal=thal)
```

```
if___name__=="__main__":
        app.run(debug=True)

        #read the csv dataset
        data = pd.read_csv("heart.csv", encoding='ANSI')
        data.columns
        data.head()

        #Total number of rows and columns
        data.shape

        # Plot a line graph for Age V/s heart
        disease plt.subplots(figsize =(8,5))
        classifiers = ['<=40', '41-50', '51-60','61 and Above']
        heart_disease = [13, 53, 64, 35] no_heart_disease =
        [6, 23, 65, 44]

        l1 = plt.plot(classifiers, heart_disease , color='g', marker='o', linestyle ='dashed',
        markerfacecolor='y', markersize=10)
        l2 = plt.plot(classifiers, no_heart_disease, color='r',marker='o', linestyle ='dashed',
        markerfacecolor='y', markersize=10 )

        plt.xlabel('Age')
        plt.ylabel('Number of patients')
        plt.title('Age V/s Heart disease')
        plt.legend((l1[0], l2[0]), ('heart_disease', 'no_heart_disease'))
        plt.show()

        # Plot a bar graph for Gender V/s target
        N = 2
        ind = np.arange(N)
        width = 0.1
        fig, ax = plt.subplots(figsize =(8,4))
```

```
heart_disease = [93, 72]
rects1 = ax.bar(ind, heart_disease, width, color='g')
no_heart_disease = [114, 24]
rects2 = ax.bar(ind+width, no_heart_disease, width, color='y')

ax.set_ylabel('Scores')
ax.set_title('Gender V/s target')
ax.set_xticks(ind)
ax.set_xticklabels(('Male','Female'))
ax.legend((rects1[0], rects2[0]), ('heart disease', 'no heart disease'))

plt.show()

#Pie charts for thal:Thalassemla
# Having heart disease
labels= 'Normal', 'Fixed defect', 'Reversable defect'
sizes=[6, 130, 28]
colors=['red', 'orange', 'green']

plt.pie(sizes, labels=labels, colors=colors, autopct='%.1f%%',
shadow=True, startangle=140)

plt.axis('equal')
plt.title('Thalassemla blood disorder status of patients having heart disease')
plt.show()

# Not having heart disease
labels= 'Normal', 'Fixed defect', 'Reversable defect'
sizes=[12, 36, 89]
colors=['red', 'orange', 'green']
```

```
plt.pie(sizes, labels=labels, colors=colors,
autopct='%.1f%%',shadow=True, startangle=140)

plt.axis('equal')
plt.title('Thalassemla blood disorder status of patients who do not have heart
disease')plt.show()

## Feature selection
#get correlation of each feature in dataset

corrmat = data.corr()
top_corr_features =
corrmat.index
plt.figure(figsize=(13,13)
)

#plot heat map
g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")

data=data.drop(['sex', 'fbs', 'restecg', 'slope', 'chol', 'age', 'trestbps'], axis=1)

target=data['target']
data =
data.drop(['target'],axis=
1)data.head()

# We split the data into training and testing set:
x_train, x_test, y_train, y_test = train_test_split(data, target,
test_size=0.3,random_state=10)

## Base
```

Learners

```
clfs = []
kfolds = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
np.random.seed(1)
```