University of New Haven
TAGLIATELA COLLEGE OF ENGINEERING
Electrical & Computer Engineering and Computer Science

Electrical & Computer Engineering & Computer Science (ECECS)

CONTENTS

University of New Haven

-Tagliatela college of Engineering

Distributed and Scalable Data

Engineering

Class DSCI-6007-02

Fall-2022

HEART DISEASE PREDICTION

Team Members:

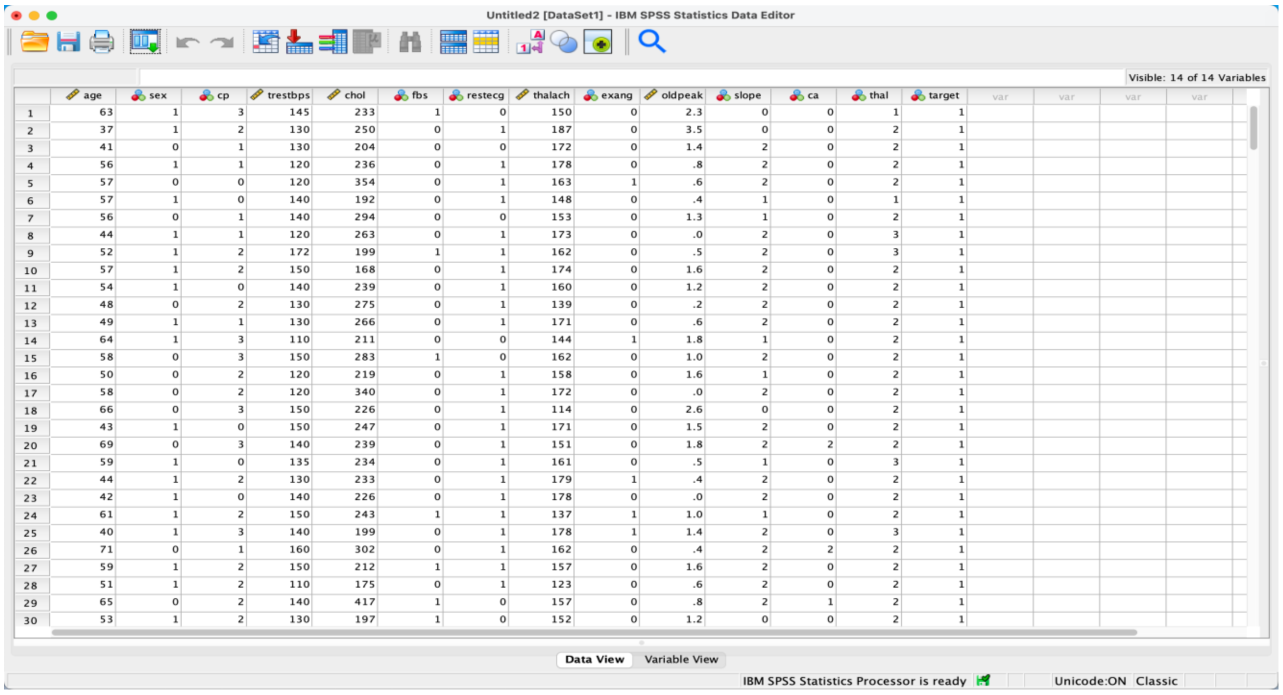
Nikhil Sai Pachipulusu	- 00762868	npach2@unh.newhaven.edu
Prathyusha Beemanaboina	- 00761322	pbeem1@unh.newhaven.edu
Varshitha Dane	- 00760728	vdane1@unh.newhaven.edu
Mohammed Sirajuddin	- 00759969	msira1@unh.newhaven.edu
Mani kumar Busireddy	- 00762386	mbusi1@unh.newhaven.edu

Abstract

The goal of this research is to determine the patient's heart health by employing several detecting techniques. This study tested various models, including decision trees, random forests, and logistic regression, in order to predict the patients' accurate findings. This smart technology makes use of artificial intelligence (AI) techniques to identify the disease most likely to be connected to the patient's data. Based on this method, users can then ask specialists for medical advice. According to certain characteristics in our data, we can determine whether a patient has heart disease or not. In an effort to determine if a patient has this disease or not, we will try to use this data to develop a model. Results indicate thateach algorithm can be used for the detection heart diseases for the patients yet identifies the best model for the detection by comparing the accuracy rate among the four models.

Data Collection

We have collected the data by the process of gathering and measuring information on targeted variables of interest in an organized system, which then allowed us to answer relevant questions and decide future outcomes. We made sure your data is Relevant and Validated. We choose our source site as Kaggle as it provides the quality and valid data.


Untitled2 [DataSet1] - IBM SPSS Statistics Data Editor
Visible: 14 of 14 Variables
Data View
Variable View
IBM SPSS Statistics Processor is ready
Unicode:ON
Classic

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target	var	var	var	var
1	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1				
2	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1				
3	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1				
4	56	1	1	120	236	0	1	178	0	.8	2	0	2	1				
5	57	0	0	120	354	0	1	163	1	.6	2	0	2	1				
6	57	1	0	140	192	0	1	148	0	.4	1	0	1	1				
7	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1				
8	44	1	1	120	263	0	1	173	0	.0	2	0	3	1				
9	52	1	2	172	199	1	1	162	0	.5	2	0	3	1				
10	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1				
11	54	1	0	140	239	0	1	160	0	1.2	2	0	2	1				
12	48	0	2	130	275	0	1	139	0	.2	2	0	2	1				
13	49	1	1	130	266	0	1	171	0	.6	2	0	2	1				
14	64	1	3	110	211	0	0	144	1	1.8	1	0	2	1				
15	58	0	3	150	283	1	0	162	0	1.0	2	0	2	1				
16	50	0	2	120	219	0	1	158	0	1.6	1	0	2	1				
17	58	0	2	120	340	0	1	172	0	.0	2	0	2	1				
18	66	0	3	150	226	0	1	114	0	2.6	0	0	2	1				
19	43	1	0	150	247	0	1	171	0	1.5	2	0	2	1				
20	69	0	3	140	239	0	1	151	0	1.8	2	2	2	1				
21	59	1	0	135	234	0	1	161	0	.5	1	0	3	1				
22	44	1	2	130	233	0	1	179	1	.4	2	0	2	1				
23	42	1	0	140	226	0	1	178	0	.0	2	0	2	1				
24	61	1	2	150	243	1	1	137	1	1.0	1	0	2	1				
25	40	1	3	140	199	0	1	178	1	1.4	2	0	3	1				
26	71	0	1	160	302	0	1	162	0	.4	2	2	2	1				
27	59	1	2	150	212	1	1	157	0	1.6	2	0	2	1				
28	51	1	2	110	175	0	1	123	0	.6	2	0	2	1				
29	65	0	2	140	417	1	0	157	0	.8	2	1	2	1				
30	53	1	2	130	197	1	0	152	0	1.2	0	0	2	1				

Data Cleaning

We used data cleaning to improve the quality of our raw data, which eventually produced more accurate and conclusive results. We took the following steps that produced a cleaner dataset:

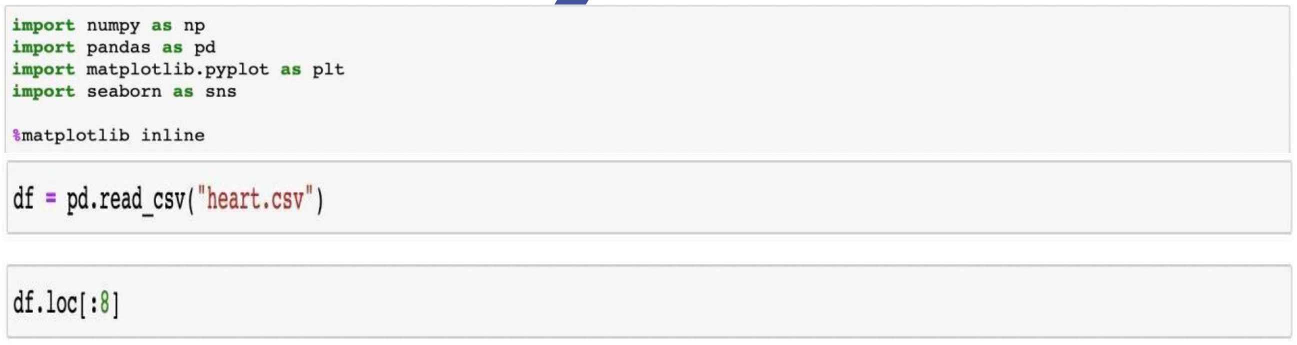
1. We Removed duplicate values as we combined multiple datasets and removed irrelevant observations as they specify the problem that we solved.
2. We addressed missing values by using Imputation techniques, drop features and observations.
3. Reformatted data types e.g. Boolean, numeric, Datetime.
4. Finally, we reformatted the string and validated the data.

Data Understanding

We have imported the data from the data set using different libraries like numpy, pandas, matplotlib and seaborn. After loading the data, we read the dataset. These are the parts of the data loading. Our dataset consists of fourteen columns which are

1. Age: patient's age
2. Sex: Gender of the patient (Male = 1, Female = 0),
3. Chest pain type(CP): Includes pressure, fullness, burning or tightness in the chest .
4. Trestbps: A person's resting blood pressure >120.
5. Chol: cholesterol levels. Good = less than 200, Moderate = 200-239, High = >240.
6. Fasting blood sugar(FBS): Measures the blood sugar after the overnight fast. 0 = normal, 1 = abnormal.
7. Resting electro cartographic results(rest ECG): Normal heart rate beats 60-100 per minute. 0 = normal, 1 = abnormal.
8. Thalach: Persons maximum heart rate achieved.
9. Exercise included angina(exang): The pain or stress that occurs after exercise. 0 = normal, 1 = abnormal.
10. Oldpeak: Exercise relative to rest. Records the slope of the peak exercise whether up, flat or down. Ranges between 0 to 5.
11. Slope: It is ST segment shift relative to the exercise-induced increments in patients heart rate.
12. CA: Coronary artery disease. Patients records of having CA. 0 no records, 1 or 2 had the cardiac attacks in the past.
13. Thal: A blood disorder called thalassemia.
14. Target.

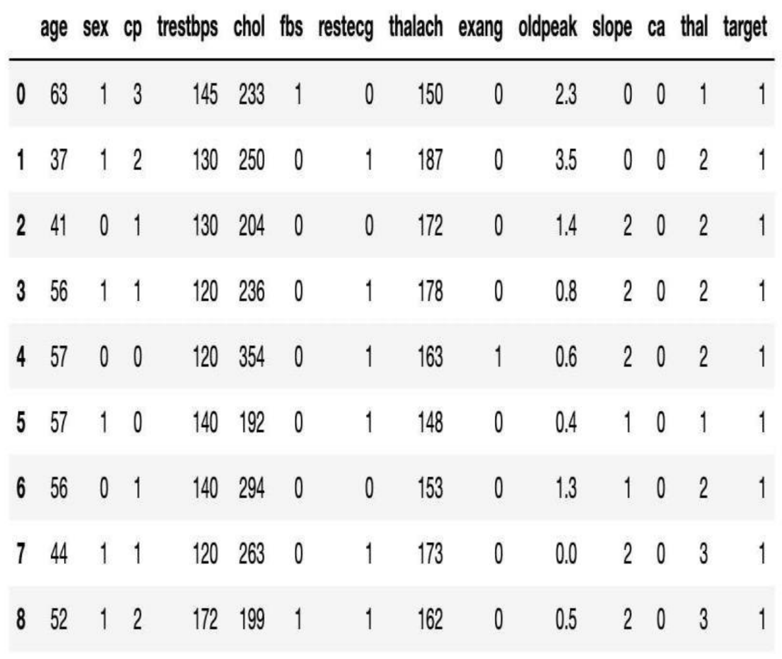
By providing all the fields required we can predict whether the patient has the heart related diseases or not. If our end result is the integer valued 0 then that particular patient has no heart disease and if the result is integer 1 then the patient has heart disease. We used info, dtypes, shape, describe to understand our dataset.



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

%matplotlib inline
```

```
df = pd.read_csv("heart.csv")
```

```
df.loc[:8]
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1

Data Preparation

With the ultimate goal of discovering a model that accurately matches our data, we used EDA to identify outliers, identify underlying patterns in the data, and test our hypotheses.

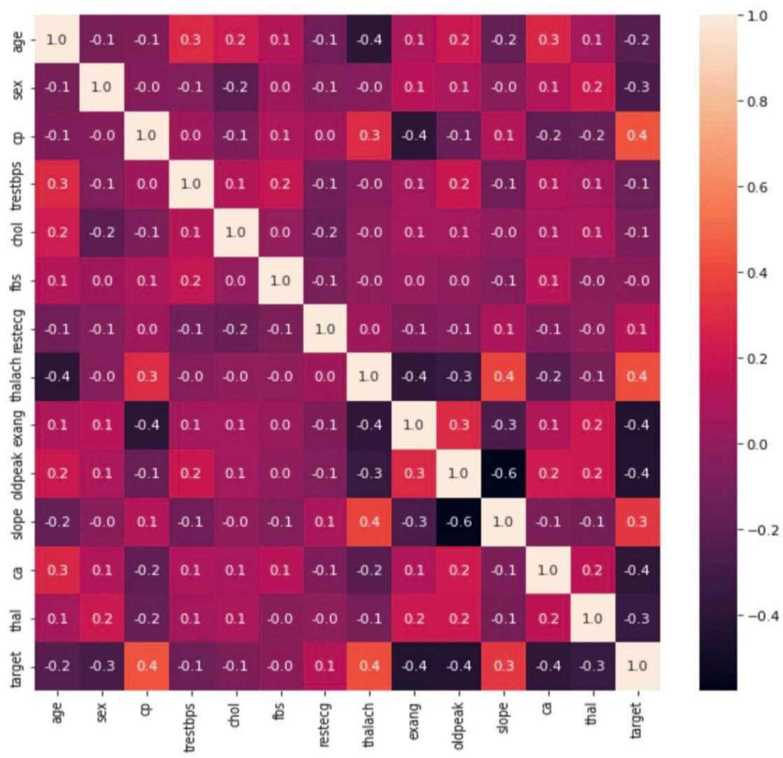
EDA generally falls into one of four categories: multivariate non-graphical, multivariate graphical, univariate non-graphical, and univariate graphical.

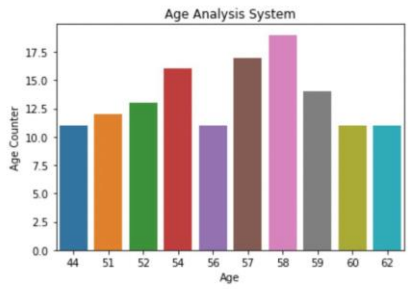
Univariate non-graphical: make observations of the population and understand sample distributions of a single variable. (e.g. the measure of spread, the measure of central tendency, outlier detection)

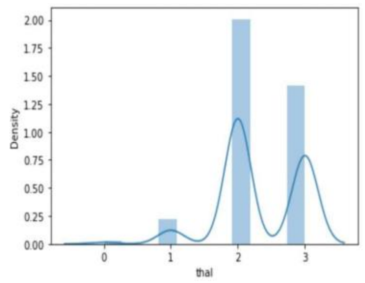
Univariate graphical: graphical analysis on a single variable. (e.g. Histograms, Boxplots, Stem and leaf)

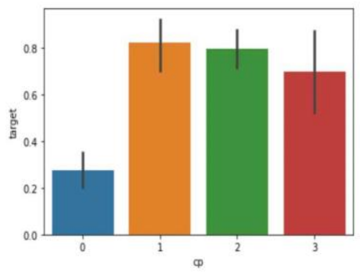
Multivariate non-graphical: techniques which show the relationship between two or more variables. (e.g. covariance, correlations)

Multivariate graphical: graphically show the relationship between two or more variables. (e.g. bar plots, scatterplots)





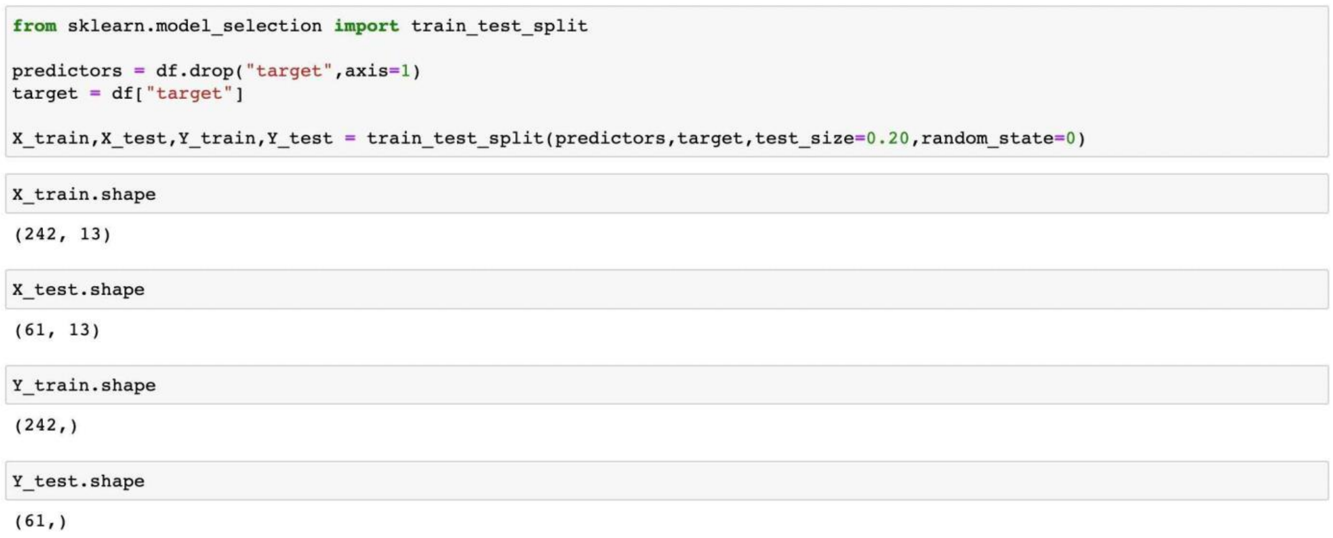




Methodology

In this project we did data modelling to create the most efficient method to find the results by the accuracy rates of our three algorithms which are Random forest classification, Logistic regression and Decision tree. The highest accuracy rate model is chosen to be the best model to predict the patient’s heart disease.

We used train and test methods to measure the accuracy of our model. As we split the data into two sets and 80% of the data is training set and the rest 20% is for testing. So, we trained our model using training set and tested our model using testing set.



```
from sklearn.model_selection import train_test_split

predictors = df.drop("target",axis=1)
target = df["target"]

X_train,X_test,Y_train,Y_test = train_test_split(predictors,target,test_size=0.20,random_state=0)
```

```
X_train.shape
```

```
(242, 13)
```

```
X_test.shape
```

```
(61, 13)
```

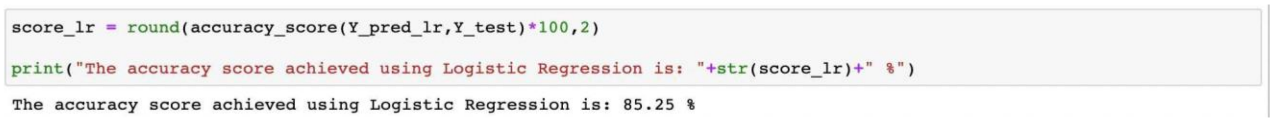
```
Y_train.shape
```

```
(242,)
```

```
Y_test.shape
```

```
(61,)
```

The accuracy rate we achieved using Logistic regression classifier is 85.25%.

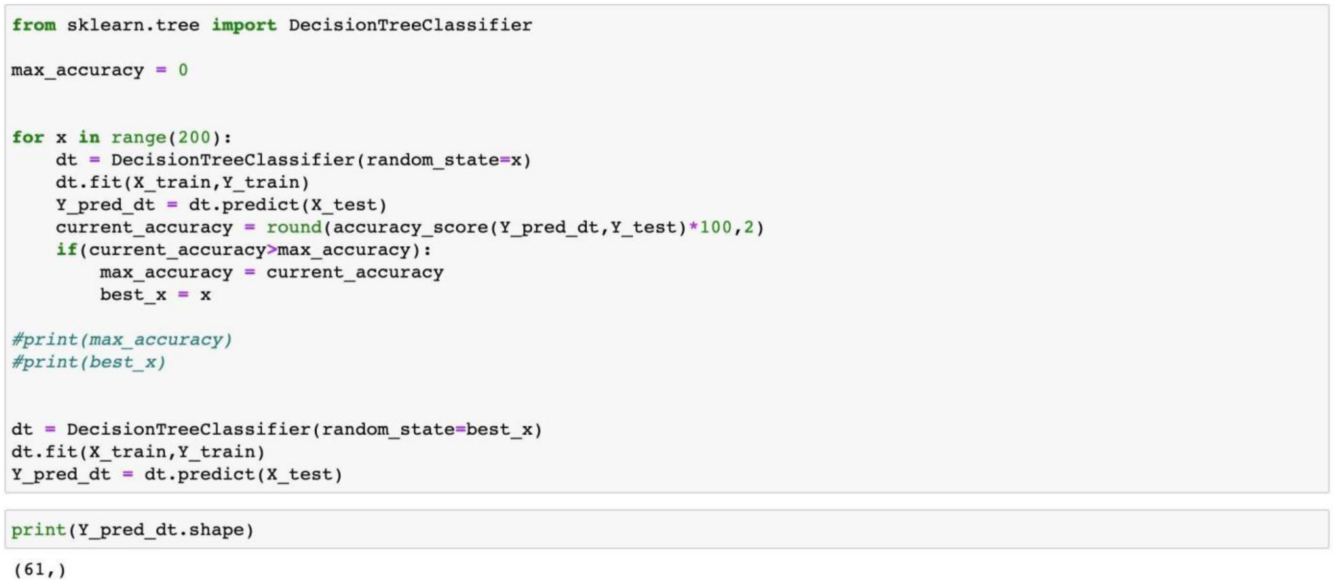


```
score_lr = round(accuracy_score(Y_pred_lr,Y_test)*100,2)

print("The accuracy score achieved using Logistic Regression is: "+str(score_lr)+" %")
```

```
The accuracy score achieved using Logistic Regression is: 85.25 %
```

Decision Tree Classifier

Decision trees, the second classifier we utilized, operate by segmenting our dataset into smaller sections according to certain storing criteria. With each dataset decision, a new subset is produced, and the number of samples in each subset decreases. The data is split up into categories by the network that each include a single data point, and these examples are then categorised using a key that has been assigned..
As the above we imported libraries and read the preprocessed dataset and split the data into training and testing data. Now we have checked whether the model fits on to the data. Then we predicted the values using classifier model which is done to perform an unbiased evaluation and get the accuracy score of the model.
We calculated the accuracies of both training and testing data and found the values are comparable. So, we confirmed the model isn't overfitting. Predictions are made after checking the model fits, we evaluated the classifier and found the accuracy rate of decision tree classifier.

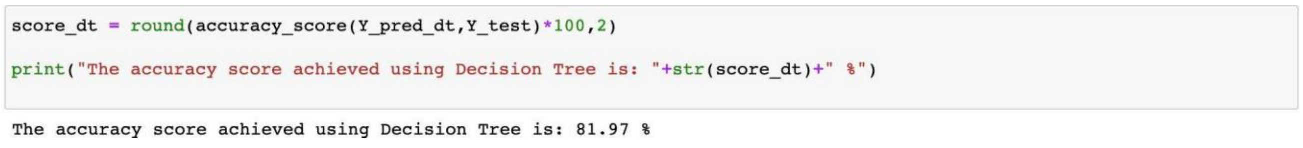


```
from sklearn.tree import DecisionTreeClassifier

max_accuracy = 0


for x in range(200):
    dt = DecisionTreeClassifier(random_state=x)
    dt.fit(X_train,Y_train)
    Y_pred_dt = dt.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_dt,Y_test)*100,2)
    if(current_accuracy>max_accuracy):
        max_accuracy = current_accuracy
        best_x = x

#print(max_accuracy)
#print(best_x)


dt = DecisionTreeClassifier(random_state=best_x)
dt.fit(X_train,Y_train)
Y_pred_dt = dt.predict(X_test)
```

```
print(Y_pred_dt.shape)
```

```
(61,)
```

The accuracy rate we achieved using Decision tree classifier is 81.97%.

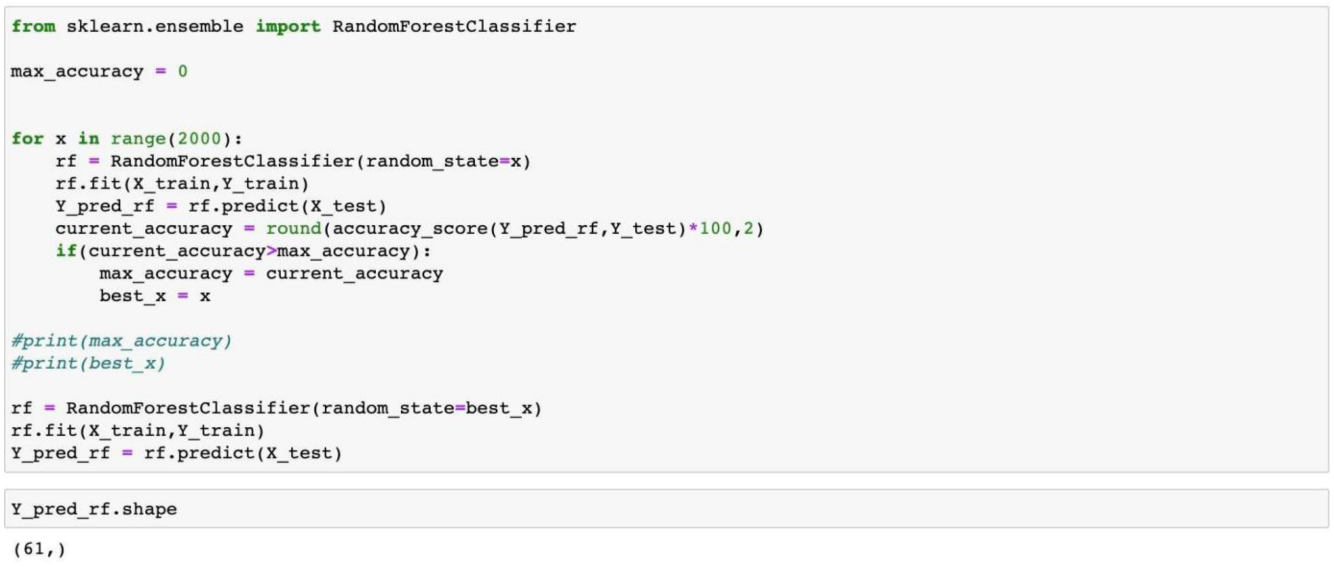


```
score_dt = round(accuracy_score(Y_pred_dt,Y_test)*100,2)

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

```
The accuracy score achieved using Decision Tree is: 81.97 %
```

Random forest classifier

The last classifier we used is Random tree classifier which is the combination of number of decision tree classifier on various sub-samples of the dataset and it uses averaging to improve the predictive accuracy and controls the over-fitting. As the above we considered our data set imported libraries, imported dataset and performed EDA. Split the data into training and testing sets. Now we got maximum accuracy for a binary probabilistic classifier. We compared current accuracy of previous classifiers and maximum accuracy of random forest classifier to choose the highest accuracy among the three. Then we made sure that the model is fitting properly into the dataset without any overfitting.

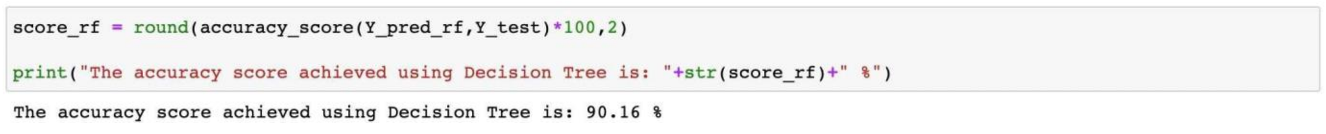


```
from sklearn.ensemble import RandomForestClassifier

max_accuracy = 0


for x in range(2000):
    rf = RandomForestClassifier(random_state=x)
    rf.fit(X_train,Y_train)
    Y_pred_rf = rf.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_rf,Y_test)*100,2)
    if(current_accuracy>max_accuracy):
        max_accuracy = current_accuracy
        best_x = x

#print(max_accuracy)
#print(best_x)

rf = RandomForestClassifier(random_state=best_x)
rf.fit(X_train,Y_train)
Y_pred_rf = rf.predict(X_test)
```

```
Y_pred_rf.shape
```

```
(61,)
```

Lastly, achieved the accuracy rate of this classifier.
The accuracy rate we achieved using Random forest classifier is 90.16%.

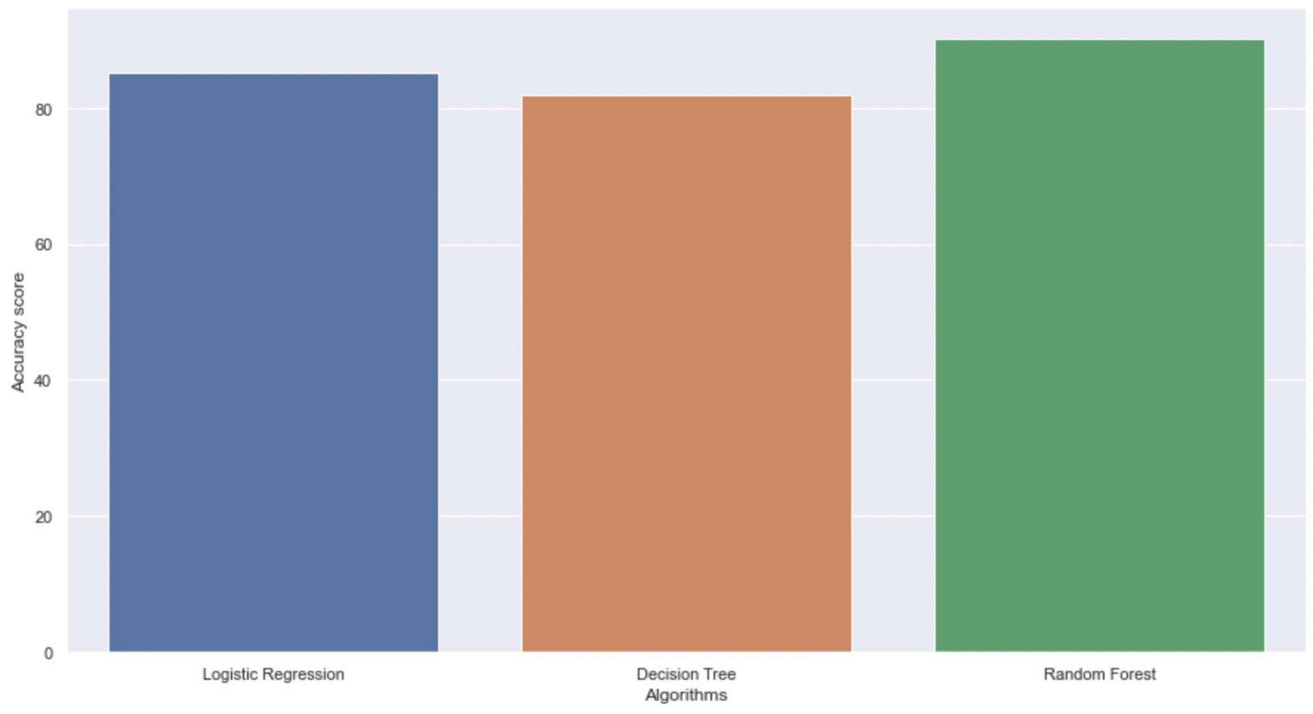


```
score_rf = round(accuracy_score(Y_pred_rf,Y_test)*100,2)

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

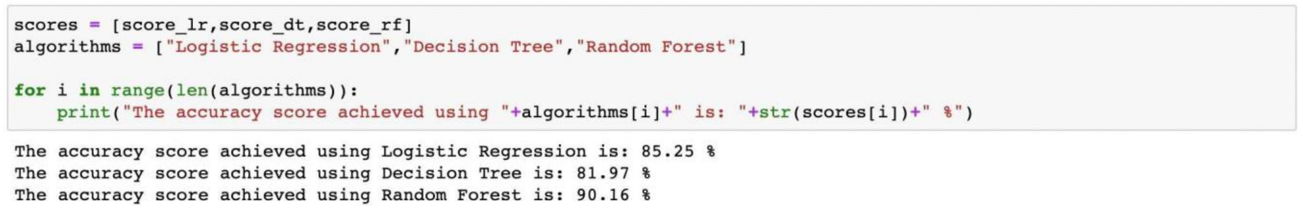
```
The accuracy score achieved using Decision Tree is: 90.16 %
```

Evaluation

After achieving the accuracy scores of all the three classifiers we have printed the results and also plotted using bar plot for the better comparison of the accuracy scores.



The end result is we found the model Random forest classifier to get the highest accuracy score of 90.16% and we considered the best fitted model ad Random forest classifier.



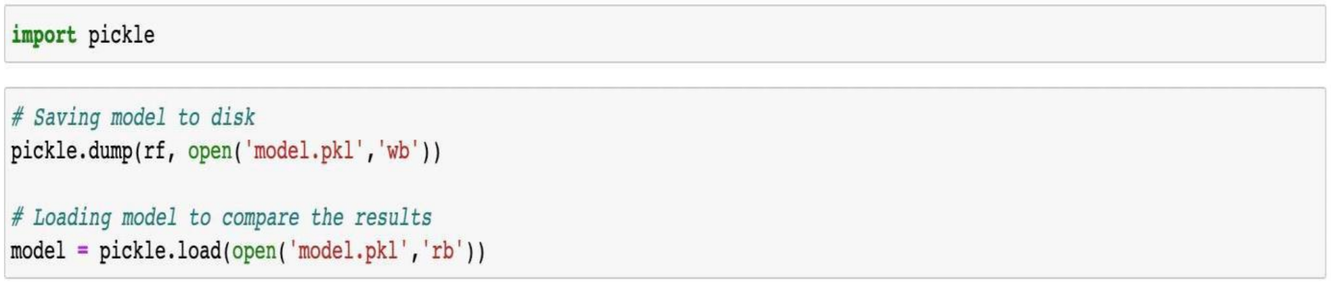
```
scores = [score_lr,score_dt,score_rf]
algorithms = ["Logistic Regression","Decision Tree","Random Forest"]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

```
The accuracy score achieved using Logistic Regression is: 85.25 %
The accuracy score achieved using Decision Tree is: 81.97 %
The accuracy score achieved using Random Forest is: 90.16 %
```

Deployment and saving model

This is the main step for the project to deploy and load the model to the disc to compare the end results. Further can bring the best model into effective action. In general, the environment

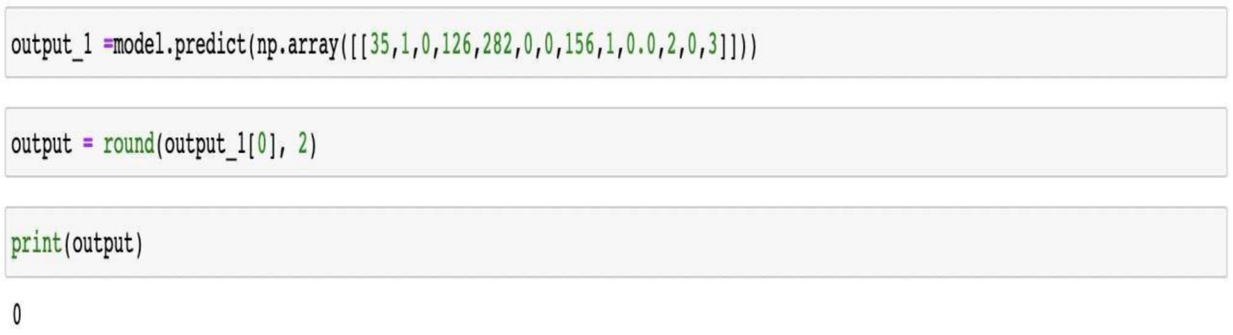
where we deploy the application would be different from where we train them. So, deployment helps us to make sure that our model is persisting effortlessly in all the environments.
In order to save the model into the disc we used pickle module which serializes the object first before writing it into the file. To read the pickle byte stream of the objects we used load function of python pickle.



```
import pickle
```

```
# Saving model to disk
pickle.dump(rf, open('model.pkl','wb'))

# Loading model to compare the results
model = pickle.load(open('model.pkl','rb'))
```

We used predict function to test the data which return the labels of the data passed as argument based upon our training data obtained from the model. At the end printed the respective output.



```
output_1 =model.predict(np.array([[35,1,0,126,282,0,0,156,1,0.0,2,0,3]]))
```

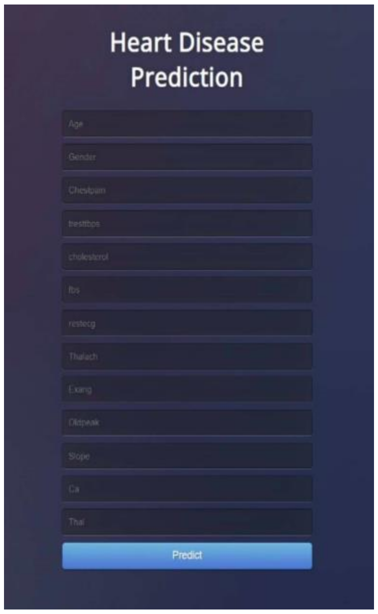
```
output = round(output_1[0], 2)
```

```
print(output)
```

```
0
```

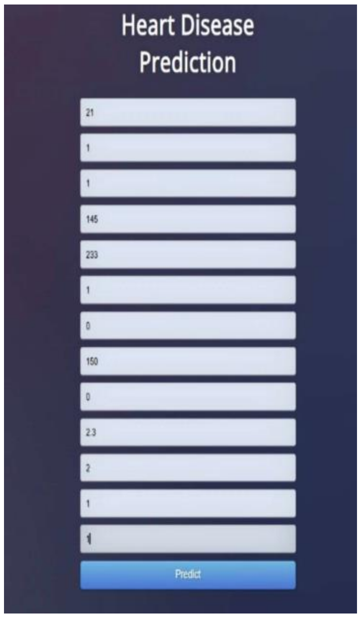
End Results

After deploying and saving our model we have tested the end results in the local web page. We divided into two classifiers 1 the patient has heart disease, 0 meaning the patient has no heart disease. We have all the fourteen fields in the web page, the particular patient has to enter all the required fields. After that the webpage will revert the data to source code and produce the predicated heart conduction class that a user is present. The output we produced from the sample data the provided in the previous stage. The user gets the end result as integer 1 or 0. 1 meaning the patient has the heart disease and 0 meaning the patient has no heart disease.



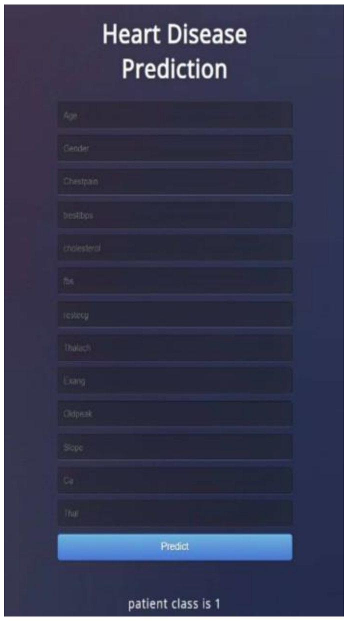
Heart Disease Prediction

Age
Gender
Chestpain
trestbps
cholesterol
fbs
restecg
Thalach
Exang
Oldpeak
Slope
Ca
Thal



Heart Disease Prediction

21
1
1
145
233
1
0
150
0
2.3
2
1
1



Heart Disease Prediction

Age
Gender
Chestpain
trestbps
cholesterol
fbs
restecg
Thalach
Exang
Oldpeak
Slope
Ca
Thal

patient class is 1

For the above given fourteen values our model has predicted that the particular patient with those inputs has generated the patient class as 1 which means the patient has a heart disease.

Conclusion

In this particular project, we compared the patient or user attributes and were able to forecast the likelihood that the patient will develop a particular form of heart disease. This model enables us to compare a patient's characteristics and determine whether they are likely to develop a disease based on factors including age, blood pressure, fasting blood sugar, resting electrocardiogram findings, thalamus, exercise-induced angina (exang), and more.

According to the patient's above-entered values, he is a guy, around the age of 21, with chest pain, a cholesterol level of 233, no resting ECG, and cardiac disease. As a result, patients with similar problems may also be at risk for developing similar heart disorders. With the help of this, medical institutions can warn patients and shield them from future health catastrophes.

Such models can be used in wide range of health conditions, not only for heart or cardiac diseases but also persons mental health conditions, can predict wide range of cancers, diabetes etc.

Contributions/References

- https://www.kaggle.com/kralmachine/analyzing-the-heart-disease
- https://towardsdatascience.com/simple-linear-regression-model-using-python-machine-learning-eab7924d18b4
- https://github.com/jupyterlab/jupyterlab
- https://www.atmosera.com/blog/creating-a-simple-linear-regression-machine-learning-model-with-scikit-learn/
- https://www.ibm.com/support/pages/producing-line-graph-dual-y-axes-spss-statistics
- https://www.open.ac.uk/socialsciences/spsstutorial/files/tutorials/graphs.pdf
- https://jakevdp.github.io/PythonDataScienceHandbook