

*A*

*Project Report On*

**Heart Disease Prediction**

*Project Report submitted to*

**Knowledge Solution India**

*Submitted by*

**Mr. Rishabh Das**

*Under the guidance of*

**Sachin Lohar**

**Knowledge Solutions India**

**Bharati Vidyapeeth’s**

**Institute of Management and Information Technology**

**Navi Mumbai**.

**BATCH: 2020-22**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Table of Content | Page No. |
| 1. | Abstract | 4 |
| 2. | Introduction | 5 |
| 3. | Software-libraries used | 6 |
| 4. | Dataset | 8 |
| 5. | Methods And Algorithms Used | 9 |
| 6. | Experiments | 10 |
| 7. | Evaluation Metrics | 14 |
| 8. | Conclusion | 16 |
| 9. | Complete Code | 17 |

**LIST OF FIGURES:**

Figure 1: Original Dataset

Figure 2: Checking Duplicate value

Figure 3: Removing Outliers from dataset

Figure 4: Correlation Matrix

Figure 5: Correlation with target

Figure 6: Separating the dataset into people with and without heart disease

Figure 7: Train Test Split

Figure 8: Decision Tree Accuracy

Figure 9: Logistic Regression Accuracy

Figure 10: Decision Tree Prediction

Figure 11: Logistic Regression Prediction

Figure 12: Decision Tree vs Logistic Regression

**ABSTRACT**

This report represents the project Heart Disease Prediction given by the Knowledge Solutions India. Cardiovascular diseases are the most common cause of death worldwide over the last few decades in the developed as well as underdeveloped and developing countries. Early detection of cardiac diseases and continuous supervision of clinicians can reduce the mortality rate. However, it is not possible to monitor patients every day in all cases accurately and consultation of a patient for 24 hours by a doctor is not available since it requires more sapience, time and expertise. In this project, we have developed and researched about models for heart disease prediction through the various heart attributes of patient and detect impending heart disease using Machine learning techniques like logistic regression on the dataset available publicly in Kaggle Website. The early prognosis of cardiovascular diseases can aid in making decisions on lifestyle changes in high risk patients and in turn reduce the complications, which can be a great milestone in the field of medicine

**INTRODUCTION**

According to the World Health Organization, every year 12 million deaths occur worldwide due to Heart Disease. The load of cardiovascular disease is rapidly increasing all over the world from the past few years. Many researches have been conducted in attempt to pinpoint the most influential factors of heart disease as well as accurately predict the overall risk. Heart Disease iseven highlighted as a silent killer which leads to the death of the person without obvious symptoms. The early diagnosis of heart disease plays a vital role in making decisions on lifestyle changes inhigh-risk patients and in turn reduce the complications. This project aims to predict future Heart Disease by analysing data of patients which classifies whether they have heart disease or not using machine-learning algorithms.

The main objective of developing this project are:

1.To develop machine learning model to predict future possibility of heart disease by implementing Logistic Regression.

2.To determine significant risk factors based on medical dataset which may lead to heart disease.

3.To analyse feature selection methods and understand their working principle.

**SOFTWARE-LIBRARIES USED**

* **NumPy:**

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

* **Pandas:**

Pandas is mainly used for data analysis. Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL, Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.

* **Matplotlib:**

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

* **Scikit-learn:**

Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction.

Components of scikit-learn

* Supervised learning algorithms
* Cross-validation
* Unsupervised learning algorithms
* Various toy datasets
* Feature extraction
* **Seaborn:**

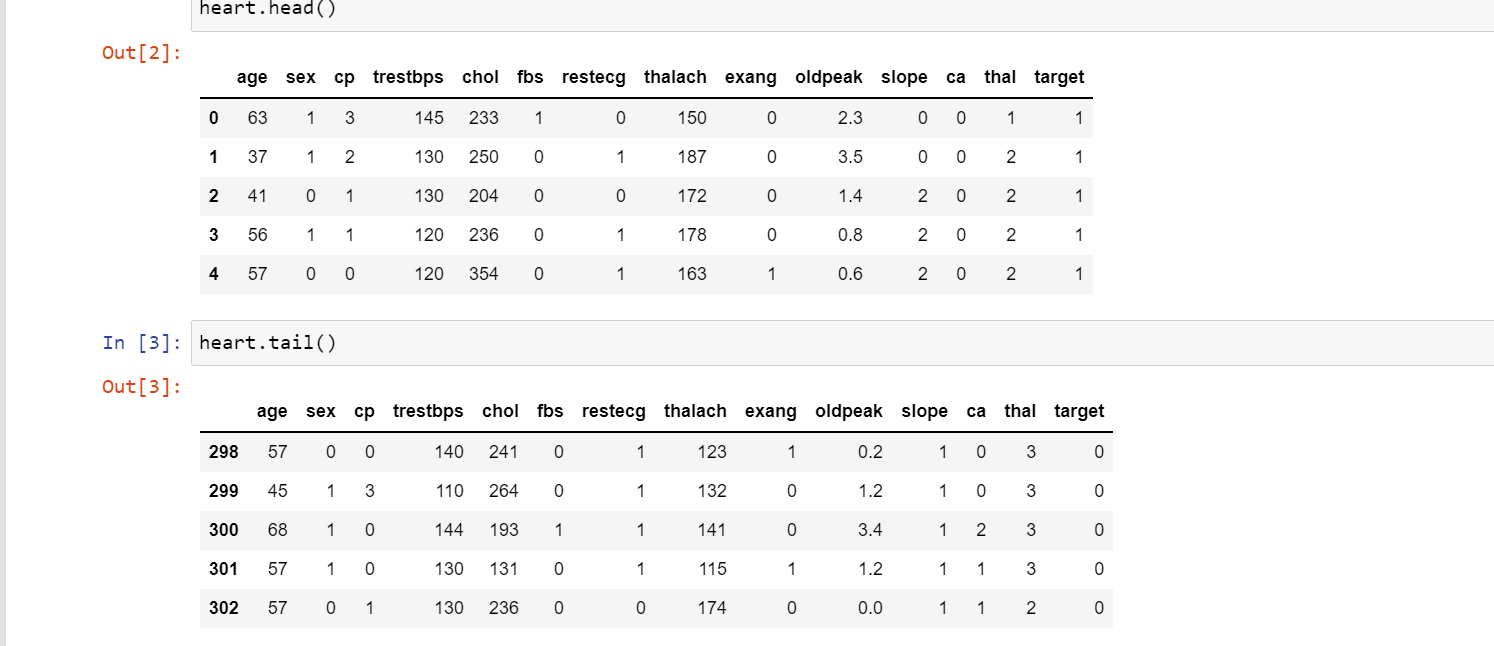
Seaborn is an amazing data visualization library for statistical graphics plotting in Python. It provides beautiful default styles and colour palettes to make statistical plots more attractive. It is built on the top of the Matplotlib library and also closely integrated to the data structures from pandas.

* **OS:**

The os module is a part of the standard library, or stdlib, within Python 3. This means that it comes with your Python installation, but you still must import it. Sample code using os: import os. All of the following code assumes you have os imported. Because it is not a built-in function, you must always import it.

**DATASET**

The dataset is publicly available on the Kaggle Website at which is from an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts. It provides patient information which includes over 4000 records and 14 attributes. The attributes include: age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting, sugar blood, resting electrocardiographic results, maximum heart rate, exercise induced angina, ST depression induced by exercise, slope of the peak exercise, number of major vessels, and target ranging from 0 to 2,where 0 is absence of heart disease. The data set is in csv (Comma Separated Value) format which is further prepared to data frame as supported by pandas library in python.



*Figure 1: Original Dataset*

**METHODS AND ALGORITHMS USED**

The main purpose of designing this system is to predict the ten-year risk of future heart disease. We have used Logistic regression as a machine-learning algorithm to train our system. These algorithms are discussed below in detail

**Logistic Regression**

Logistic Regression is a supervised classification algorithm. It is a predictive analysis algorithm based on the concept of probability. It measures the relationship between the dependent variable and the one or more independent variables (risk factors) by estimating probabilities using underlying logistic function (sigmoid function). Sigmoid function is used as a cost function to limit the hypothesis of logistic regression between 0 and 1 (squashing) i.e. 0 ≤ hθ (x) ≤ 1.

**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. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node. For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.

**EXPERIMENTS**

**Data Preparation**

Since the dataset consists of 383 observations with 0 missing data and 165 observations to be risked for heart disease, two different experiments were performed for data preparation. First, we checked the duplicate rows in data. After observing and removing the duplicate rows in data we have 382 observations.

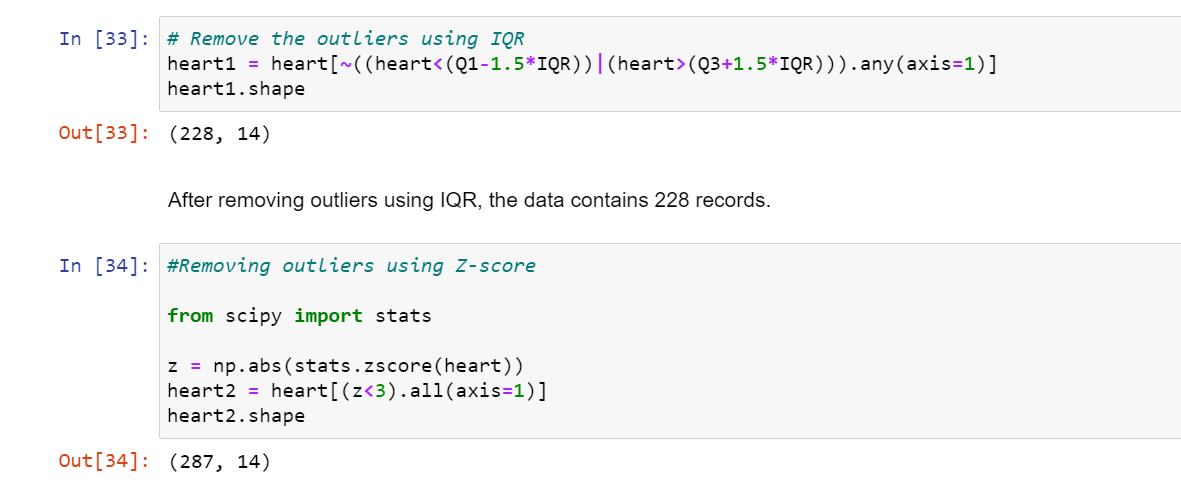


*Figure 2: Checking duplicate value*

After removing the duplicate values we checked for the outliers in the dataset. We found quite a few outliers in different categories of the dataset.

An outlier is an object that deviates significantly from the rest of the objects. They can be caused by measurement or execution error.

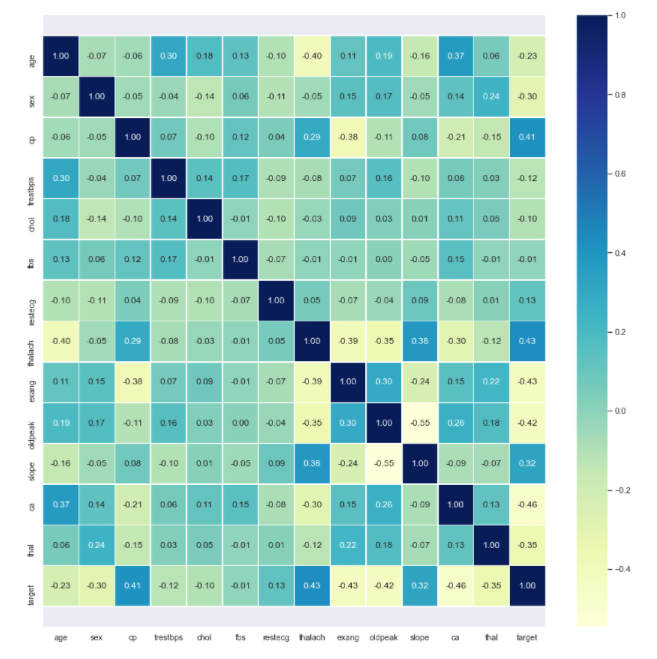
To prevent the unknown factors from affecting the Analysis we removed these outliers. Two methods are used for removing Inter-Quartile Range and Z-score. As the number of records available after removing outliers is higher after Z-score, we proceeded with Z-score.



*Figure 3: Removing Outliers from dataset*

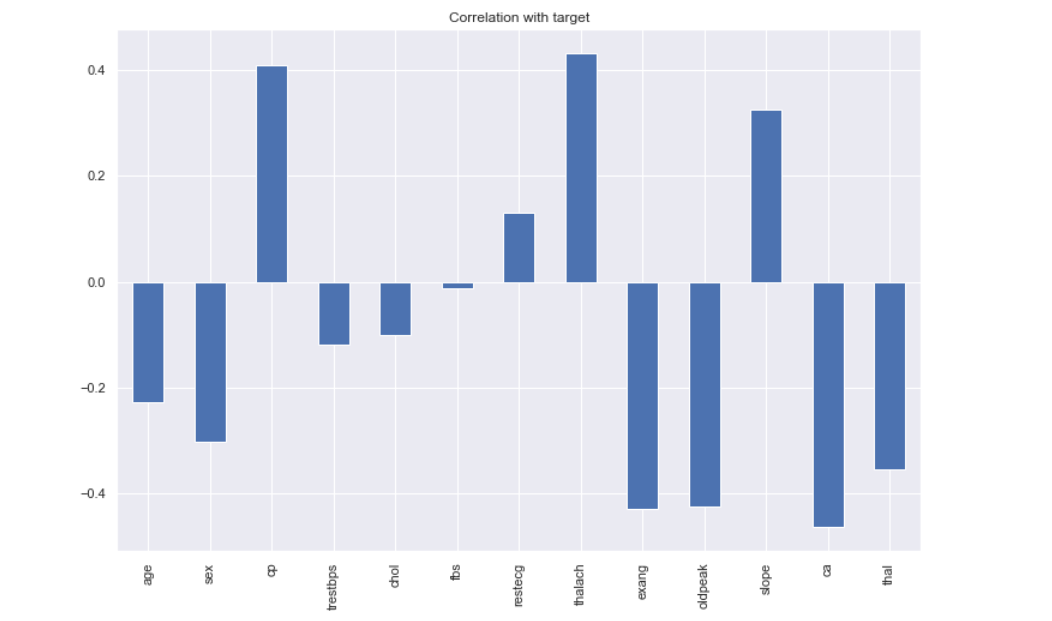
**Exploratory Analysis:**

Correlation Matrix visualization



*Figure 4: Correlation Matrix*

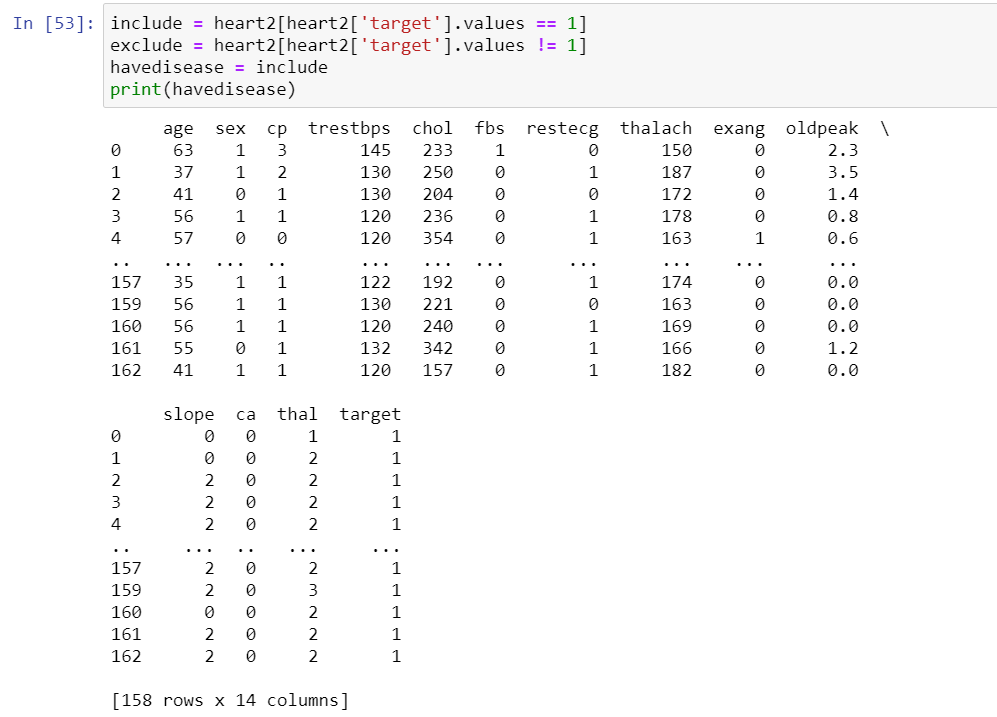
Correlation with target



*Figure 5: Correlation with target*

**Data Pre-processing**

We have separated the data of people who have heart disease and people who do not.



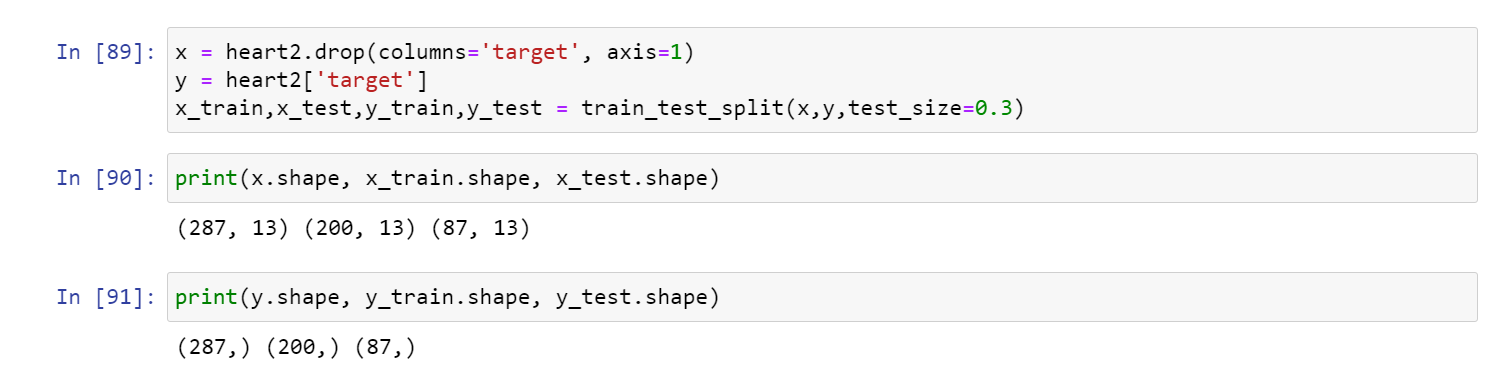
*Figure 6: Separating the dataset into people with and without heart disease*

After that we performed analysis on the people with heart disease.

* I was able to analyze their state of health by observing some of the variables in the data frame. I started by looking at the age of the individuals and found the highest age was 76, the lowest age was 29, and the median age was 52.
* A normal blood pressure level is less than 120/80 mmHg. I analyzed the systolic blood pressure of those individuals with heart disease and found the highest reading was 180, the lowest reading was 94, and the median reading was 130. The systolic reading should not go above 140, which is an indicator of high blood pressure.
* I looked at the blood sugar reading being greater than 120 and found that a small proportion of people had high blood sugar, which is indicative of diabetes.
* Only a small percentage of people had exercise-induced angina.
* I have analyse that majority people with heart disease have non-atypical angina type chest pain. And people with asymptomatic type chest pain are very less compaired to the other types.
* There is not much difference between the ratio of male and female that have heart disease.
* I analysed the Total Cholesterol of those individuals with heart disease and found the highest reading was 394, the lowest reading was 126, and the median reading was 234.
* I analysed that people with heart disease having resting electrocardiographic results(restecg) value = 2 are very less compared with the other two.
* I analysed ST depression induced by exercise relative to rest of people that have heart disease and found that most people have 0 ST depression induced by exercise relative to rest

**Training and testing**

Finally, this resulting data split into 70% train and 30% test data, which was further passed to the Logistic Regression and Decision Tree model to fit, predict and score the model.

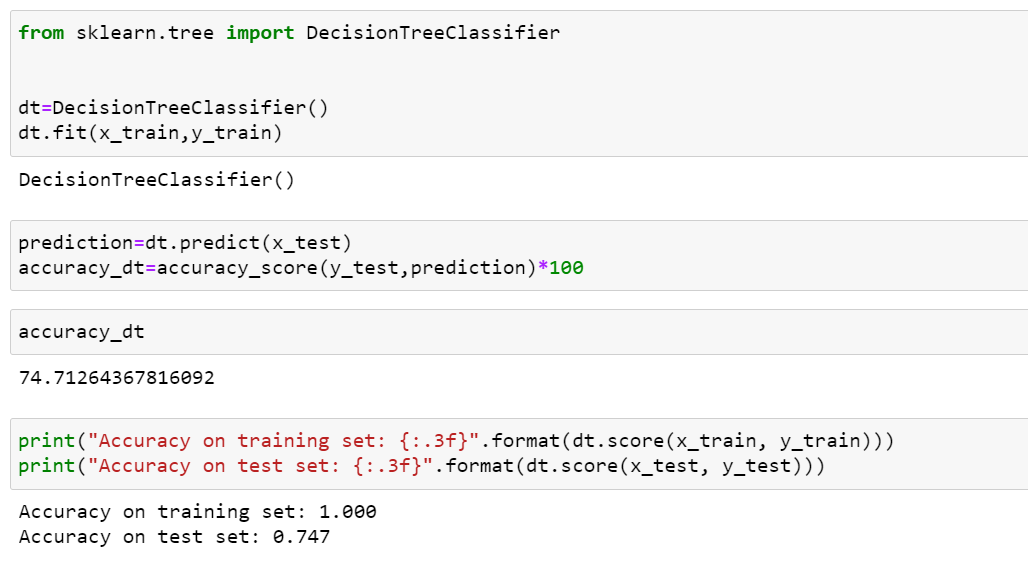


*Figure 7: Train Test Split*

**EVALUATION METRICS**

**Accuracy**

The obtained accuracy during training the data using Decision Tree was 77% .



*Figure 8: Decision Tree Accuracy*

The obtained accuracy during training the data using logistic Regression was 80%.



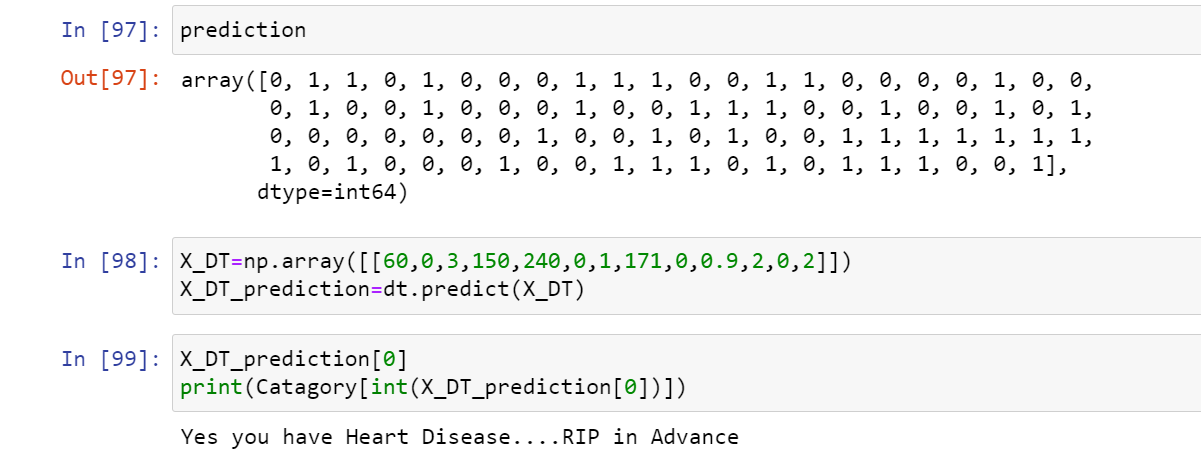
*Figure 9: Logistic Regression Accuracy*

**Prediction**

Predictive modelling is a statistical technique using machine learning and data mining to predict and forecast likely future outcomes with the aid of historical and existing data. It works by analyzing current and historical data and projecting what it learns on a model generated to forecast likely outcomes

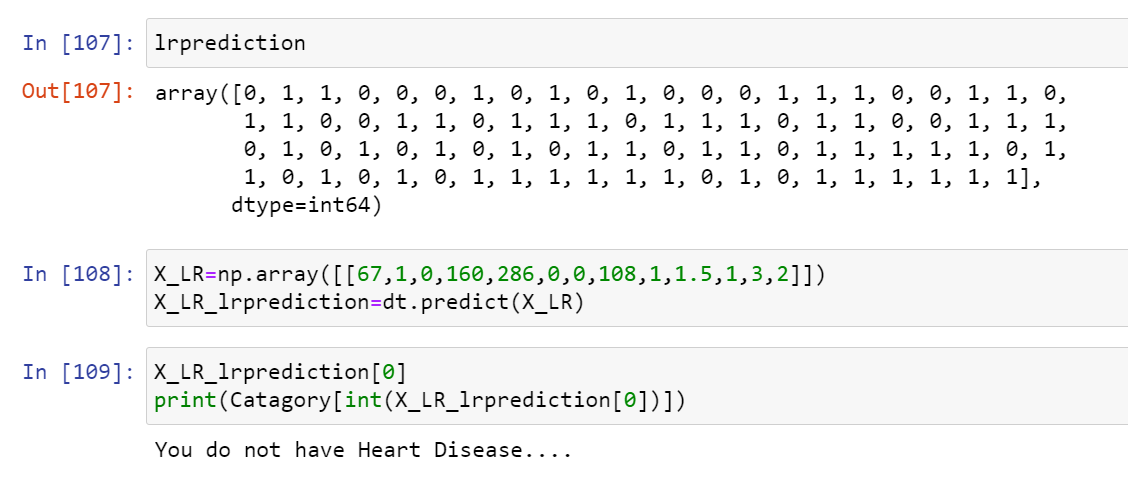
After the accuracy test we made prediction using the made models we have made.

Decision Tree prediction



*Figure 10: Decision Tree Prediction*

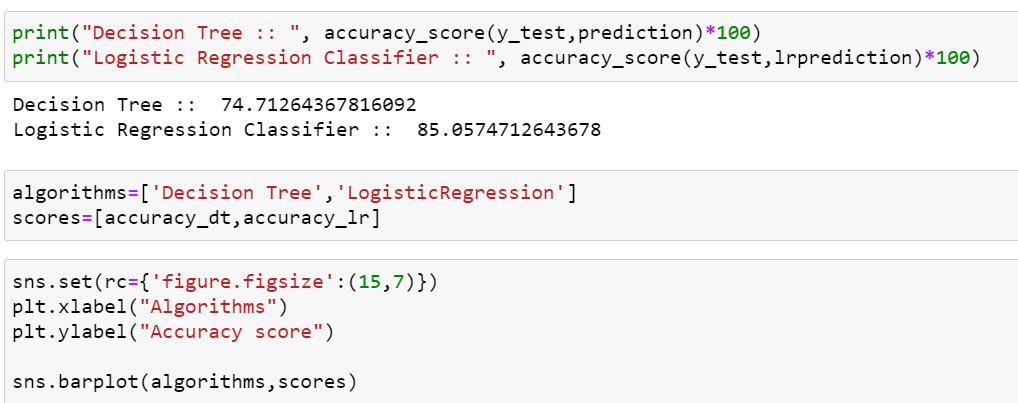
Logistic Regression Prediction



*Figure 11: Logistic Regression Prediction*

**CONCLUSION**

When performing various methods of testing it was found that Logistic Regression gave us the best results among others.



![](data:image/png;base64;base64,)

*Figure 12: Decision Tree vs Logistic Regression*

**COMPLETE CODE**

HeartDisease Prediction Project

# Heart Disease Prediction[¶](#Heart-Disease-Prediction)

In [1]:

#Importing libraries  
  
import numpy as np  
import pandas as pd  
import os  
import matplotlib.pyplot as plt  
from sklearn import preprocessing  
from sklearn.preprocessing import StandardScaler  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import accuracy\_score  
import seaborn as sns  
sns.set(style='white')  
sns.set(style='whitegrid', color\_codes=True)

In [2]:

heart = pd.read\_csv("heart.csv")  
heart.head()

Out[2]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 |

In [3]:

heart.tail()

Out[3]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
| 298 | 57 | 0 | 0 | 140 | 241 | 0 | 1 | 123 | 1 | 0.2 | 1 | 0 | 3 | 0 |
| 299 | 45 | 1 | 3 | 110 | 264 | 0 | 1 | 132 | 0 | 1.2 | 1 | 0 | 3 | 0 |
| 300 | 68 | 1 | 0 | 144 | 193 | 1 | 1 | 141 | 0 | 3.4 | 1 | 2 | 3 | 0 |
| 301 | 57 | 1 | 0 | 130 | 131 | 0 | 1 | 115 | 1 | 1.2 | 1 | 1 | 3 | 0 |
| 302 | 57 | 0 | 1 | 130 | 236 | 0 | 0 | 174 | 0 | 0.0 | 1 | 1 | 2 | 0 |

In [4]:

heart.shape

Out[4]:

(303, 14)

In [5]:

heart.describe()

Out[5]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
| count | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 |
| mean | 54.366337 | 0.683168 | 0.966997 | 131.623762 | 246.264026 | 0.148515 | 0.528053 | 149.646865 | 0.326733 | 1.039604 | 1.399340 | 0.729373 | 2.313531 | 0.544554 |
| std | 9.082101 | 0.466011 | 1.032052 | 17.538143 | 51.830751 | 0.356198 | 0.525860 | 22.905161 | 0.469794 | 1.161075 | 0.616226 | 1.022606 | 0.612277 | 0.498835 |
| min | 29.000000 | 0.000000 | 0.000000 | 94.000000 | 126.000000 | 0.000000 | 0.000000 | 71.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 47.500000 | 0.000000 | 0.000000 | 120.000000 | 211.000000 | 0.000000 | 0.000000 | 133.500000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 2.000000 | 0.000000 |
| 50% | 55.000000 | 1.000000 | 1.000000 | 130.000000 | 240.000000 | 0.000000 | 1.000000 | 153.000000 | 0.000000 | 0.800000 | 1.000000 | 0.000000 | 2.000000 | 1.000000 |
| 75% | 61.000000 | 1.000000 | 2.000000 | 140.000000 | 274.500000 | 0.000000 | 1.000000 | 166.000000 | 1.000000 | 1.600000 | 2.000000 | 1.000000 | 3.000000 | 1.000000 |
| max | 77.000000 | 1.000000 | 3.000000 | 200.000000 | 564.000000 | 1.000000 | 2.000000 | 202.000000 | 1.000000 | 6.200000 | 2.000000 | 4.000000 | 3.000000 | 1.000000 |

In [6]:

heart.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 303 entries, 0 to 302  
Data columns (total 14 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 age 303 non-null int64   
 1 sex 303 non-null int64   
 2 cp 303 non-null int64   
 3 trestbps 303 non-null int64   
 4 chol 303 non-null int64   
 5 fbs 303 non-null int64   
 6 restecg 303 non-null int64   
 7 thalach 303 non-null int64   
 8 exang 303 non-null int64   
 9 oldpeak 303 non-null float64  
 10 slope 303 non-null int64   
 11 ca 303 non-null int64   
 12 thal 303 non-null int64   
 13 target 303 non-null int64   
dtypes: float64(1), int64(13)  
memory usage: 33.3 KB

In [7]:

heart.isnull().sum()

Out[7]:

age 0  
sex 0  
cp 0  
trestbps 0  
chol 0  
fbs 0  
restecg 0  
thalach 0  
exang 0  
oldpeak 0  
slope 0  
ca 0  
thal 0  
target 0  
dtype: int64

In [8]:

#No missing value

In [9]:

info = ["Age of the Person","Gender:- 1: male, 0: female","chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic","resting blood pressure(in mm Hg)","cholestoral in mg/dl","fasting blood sugar > 120 mg/dl","resting electrocardiographic results (values 0,1,2)"," maximum heart rate achieved","exercise induced angina (1 = yes; 0 = no)","oldpeak = ST depression induced by exercise relative to rest","the slope of the peak exercise ST segment","number of major vessels (0-3) colored by flourosopy","thal: 3 = normal; 6 = fixed defect; 7 = reversable defect"]  
  
for i in range(len(info)):  
 print(heart.columns[i]+":\t\t\t"+info[i])

age: Age of the Person  
sex: Gender:- 1: male, 0: female  
cp: chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic  
trestbps: resting blood pressure(in mm Hg)  
chol: cholestoral in mg/dl  
fbs: fasting blood sugar > 120 mg/dl  
restecg: resting electrocardiographic results (values 0,1,2)  
thalach: maximum heart rate achieved  
exang: exercise induced angina (1 = yes; 0 = no)  
oldpeak: oldpeak = ST depression induced by exercise relative to rest  
slope: the slope of the peak exercise ST segment  
ca: number of major vessels (0-3) colored by flourosopy  
thal: thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

In [10]:

# checking the distribution of target variable  
  
heart['target'].value\_counts()

Out[10]:

1 165  
0 138  
Name: target, dtype: int64

# Data Cleaning/ Data preprocessing[¶](#Data-Cleaning/-Data-preprocessing)

In [11]:

#Check duplicate rows in data  
duplicate\_rows = heart[heart.duplicated()]  
print("Number of duplicate rows :: ", duplicate\_rows.shape)

Number of duplicate rows :: (1, 14)

In [12]:

##we have one duplicate row.

In [13]:

#Removing the duplicate row  
heart = heart.drop\_duplicates()  
duplicate\_rows = heart[heart.duplicated()]  
print("Number of duplicate rows :: ", duplicate\_rows.shape)  
#Number of duplicate rows after dropping one duplicate row

Number of duplicate rows :: (0, 14)

In [14]:

heart.shape

Out[14]:

(302, 14)

In [15]:

#Looking for null values  
print("Null values :: ")  
print(heart.isnull() .sum())

Null values ::   
age 0  
sex 0  
cp 0  
trestbps 0  
chol 0  
fbs 0  
restecg 0  
thalach 0  
exang 0  
oldpeak 0  
slope 0  
ca 0  
thal 0  
target 0  
dtype: int64

In [16]:

#No Null value found

In [17]:

#Detecting Outliers using box plot

Detecting Outliers using IQR (InterQuartile Range)

In [18]:

sns.boxplot(x=heart['age'])

Out[18]:

<AxesSubplot:xlabel='age'>

![](data:image/png;base64;base64,)

# No Outliers observed in 'age'[¶](#No-Outliers-observed-in-'age')

In [19]:

sns.boxplot(x=heart['sex'])

Out[19]:

<AxesSubplot:xlabel='sex'>

![](data:image/png;base64;base64,)

# No outliers observed in sex data[¶](#No-outliers-observed-in-sex-data)

In [20]:

sns.boxplot(x=heart['cp'])

Out[20]:

<AxesSubplot:xlabel='cp'>

![](data:image/png;base64;base64,)

# No outliers in 'cp'[¶](#No-outliers-in-'cp')

In [21]:

sns.boxplot(x=heart['trestbps'])

Out[21]:

<AxesSubplot:xlabel='trestbps'>

![](data:image/png;base64;base64,)

# Some outliers are observed in 'trestbps'. They will be removed later[¶](#X4932385ad85ab309cbdf034d3e7560d3c3dbd96)

In [22]:

sns.boxplot(x=heart['chol'])

Out[22]:

<AxesSubplot:xlabel='chol'>

![](data:image/png;base64;base64,)

# Some outliers are observed in 'chol'.[¶](#Some-outliers-are-observed-in-'chol'.)

In [23]:

sns.boxplot(x=heart['fbs'])

Out[23]:

<AxesSubplot:xlabel='fbs'>

![](data:image/png;base64;base64,)

In [24]:

sns.boxplot(x=heart['restecg'])

Out[24]:

<AxesSubplot:xlabel='restecg'>

![](data:image/png;base64;base64,)

In [25]:

sns.boxplot(x=heart['thalach'])

Out[25]:

<AxesSubplot:xlabel='thalach'>

![](data:image/png;base64;base64,)

# Outliers are present in 'thalach'[¶](#Outliers-are-present-in-'thalach')

In [26]:

sns.boxplot(x=heart['exang'])

Out[26]:

<AxesSubplot:xlabel='exang'>

![](data:image/png;base64;base64,)

In [27]:

sns.boxplot(x=heart['oldpeak'])

Out[27]:

<AxesSubplot:xlabel='oldpeak'>

![](data:image/png;base64;base64,)

# Outliers are present in 'OldPeak'[¶](#Outliers-are-present-in-'OldPeak')

In [28]:

sns.boxplot(x=heart['slope'])

Out[28]:

<AxesSubplot:xlabel='slope'>

![](data:image/png;base64;base64,)

In [29]:

sns.boxplot(x=heart['ca'])

Out[29]:

<AxesSubplot:xlabel='ca'>

![](data:image/png;base64;base64,)

# Outliers are present in 'ca'[¶](#Outliers-are-present-in-'ca')

In [30]:

sns.boxplot(x=heart['thal'])

Out[30]:

<AxesSubplot:xlabel='thal'>

![](data:image/png;base64;base64,)

# Outliers are present in 'thal'[¶](#Outliers-are-present-in-'thal')

# Removing Outliers[¶](#Removing-Outliers)

In [31]:

#Removing Outlier using Inter-Quartile Range

In [32]:

#Find the InterQuartile Range  
Q1 = heart.quantile(0.25)  
Q3 = heart.quantile(0.75)  
IQR = Q3-Q1  
print('\*\*\*\*\*\*\*\*\*\*\* InterQuartile Range \*\*\*\*\*\*\*\*\*\*\*')  
print(IQR)

\*\*\*\*\*\*\*\*\*\*\* InterQuartile Range \*\*\*\*\*\*\*\*\*\*\*  
age 13.00  
sex 1.00  
cp 2.00  
trestbps 20.00  
chol 63.75  
fbs 0.00  
restecg 1.00  
thalach 32.75  
exang 1.00  
oldpeak 1.60  
slope 1.00  
ca 1.00  
thal 1.00  
target 1.00  
dtype: float64

In [33]:

# Remove the outliers using IQR  
heart1 = heart[~((heart<(Q1-1.5\*IQR))|(heart>(Q3+1.5\*IQR))).any(axis=1)]  
heart1.shape

Out[33]:

(228, 14)

After removing outliers using IQR, the data contains 228 records.

In [34]:

#Removing outliers using Z-score  
  
from scipy import stats  
  
z = np.abs(stats.zscore(heart))  
heart2 = heart[(z<3).all(axis=1)]  
heart2.shape

Out[34]:

(287, 14)

After using Z-score to detect and remove outliers, the number of records in the dataset is 287.

As the number of records available is higher after Z-score, we will proceed with ‘heart2’

# EDA[¶](#EDA)

In [35]:

heart2.hist(figsize=(14,14))  
plt.show()

![](data:image/png;base64;base64,)

In [36]:

plt.bar(x=heart2['sex'],height=heart2['age'])  
plt.show()

![](data:image/png;base64;base64,)

In [37]:

sns.barplot(x="fbs", y="target", data=heart2)  
plt.show()

![](data:image/png;base64;base64,)

In [38]:

sns.barplot(x=heart2['sex'],y=heart2['age'],hue=heart2['target'])

Out[38]:

<AxesSubplot:xlabel='sex', ylabel='age'>

![](data:image/png;base64;base64,)

In [39]:

sns.barplot(heart2["cp"],heart2['target'])

F:\annaconda\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.  
 warnings.warn(

Out[39]:

<AxesSubplot:xlabel='cp', ylabel='target'>

![](data:image/png;base64;base64,)

In [40]:

sns.barplot(heart2["sex"],heart2['target'])

F:\annaconda\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.  
 warnings.warn(

Out[40]:

<AxesSubplot:xlabel='sex', ylabel='target'>

![](data:image/png;base64;base64,)

In [41]:

sns.distplot(heart2["thal"])

F:\annaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)

Out[41]:

<AxesSubplot:xlabel='thal', ylabel='Density'>

![](data:image/png;base64;base64,)

In [42]:

sns.distplot(heart2["chol"])

F:\annaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)

Out[42]:

<AxesSubplot:xlabel='chol', ylabel='Density'>

![](data:image/png;base64;base64,)

In [43]:

sns.pairplot(heart2,hue='target')

Out[43]:

<seaborn.axisgrid.PairGrid at 0x2327470d160>

![](data:image/png;base64;base64,)

In [44]:

# checking the distribution of target variable  
  
heart2['target'].value\_counts()

Out[44]:

1 158  
0 129  
Name: target, dtype: int64

In [45]:

y = heart2["target"]  
  
sns.countplot(y)  
  
  
target\_temp = heart2.target.value\_counts()  
  
print(target\_temp)

1 158  
0 129  
Name: target, dtype: int64

F:\annaconda\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.  
 warnings.warn(

![](data:image/png;base64;base64,)

1 = Heart Disease 0 = No Heart Disease

158 people in this data set have heart disease and 129 don't.

In [46]:

numeric\_columns=['trestbps','chol','thalach','age','oldpeak']

In [47]:

sns.pairplot(heart2[numeric\_columns])

Out[47]:

<seaborn.axisgrid.PairGrid at 0x2327c391250>

![](data:image/png;base64;base64,)

In [48]:

# create a correlation heatmap  
sns.heatmap(heart2[numeric\_columns].corr(),annot=True, cmap='terrain', linewidths=0.1)  
fig=plt.gcf()  
fig.set\_size\_inches(8,6)  
plt.show()

![](data:image/png;base64;base64,)

In [49]:

# create four distplots  
plt.figure(figsize=(12,10))  
plt.subplot(221)  
sns.distplot(heart2[heart2['target']==0].age)  
plt.title('Age of patients without heart disease')  
plt.subplot(222)  
sns.distplot(heart2[heart2['target']==1].age)  
plt.title('Age of patients with heart disease')  
plt.subplot(223)  
sns.distplot(heart2[heart2['target']==0].thalach )  
plt.title('Max heart rate of patients without heart disease')  
plt.subplot(224)  
sns.distplot(heart2[heart2['target']==1].thalach )  
plt.title('Max heart rate of patients with heart disease')  
plt.show()

F:\annaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)  
F:\annaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)  
F:\annaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)  
F:\annaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)

![](data:image/png;base64;base64,)

In [50]:

plt.figure(figsize=(13,6))  
plt.subplot(121)  
sns.violinplot(x="target", y="thalach", data=heart2, inner=None)  
sns.swarmplot(x="target", y="thalach", data=heart2, color='w', alpha=0.5)  
  
  
plt.subplot(122)  
sns.swarmplot(x="target", y="thalach", data=heart2)  
plt.show()

![](data:image/png;base64;base64,)

In [51]:

# create pairplot and two barplots  
plt.figure(figsize=(16,6))  
plt.subplot(131)  
sns.pointplot(x="sex", y="target", hue='cp', data=heart2)  
plt.legend(['male = 1', 'female = 0'])  
plt.subplot(132)  
sns.barplot(x="exang", y="target", data=heart2)  
plt.legend(['yes = 1', 'no = 0'])  
plt.subplot(133)  
sns.countplot(x="slope", hue='target', data=heart2)  
plt.show()

![](data:image/png;base64;base64,)

In [52]:

#Correlation Matrix

In [53]:

corr\_matrix = heart2.corr()  
fig, ax = plt.subplots(figsize=(15, 15))  
ax = sns.heatmap(corr\_matrix,  
 annot=True,  
 linewidths=0.5,  
 fmt=".2f",  
 cmap="YlGnBu");  
bottom, top = ax.get\_ylim()  
ax.set\_ylim(bottom + 0.5, top - 0.5)

Out[53]:

(14.5, -0.5)

![](data:image/png;base64;base64,)

In [54]:

heart2.drop('target', axis=1).corrwith(heart2.target).plot(kind='bar', grid=True, figsize=(12, 8),   
title="Correlation with target")

Out[54]:

<AxesSubplot:title={'center':'Correlation with target'}>

![](data:image/png;base64;base64,)

fbs and chol are the lowest correlated with the target variable. All other variables have a significant correlation with the target variable.

# DATA Preprocessing¶[¶](#DATA-Preprocessing¶)

In [55]:

#Seperating people having heart disease and people that do not have heart disease

In [56]:

include = heart2[heart2['target'].values == 1]  
exclude = heart2[heart2['target'].values != 1]  
havedisease = include  
print(havedisease)

age sex cp trestbps chol fbs restecg thalach exang oldpeak \  
0 63 1 3 145 233 1 0 150 0 2.3   
1 37 1 2 130 250 0 1 187 0 3.5   
2 41 0 1 130 204 0 0 172 0 1.4   
3 56 1 1 120 236 0 1 178 0 0.8   
4 57 0 0 120 354 0 1 163 1 0.6   
.. ... ... .. ... ... ... ... ... ... ...   
157 35 1 1 122 192 0 1 174 0 0.0   
159 56 1 1 130 221 0 0 163 0 0.0   
160 56 1 1 120 240 0 1 169 0 0.0   
161 55 0 1 132 342 0 1 166 0 1.2   
162 41 1 1 120 157 0 1 182 0 0.0   
  
 slope ca thal target   
0 0 0 1 1   
1 0 0 2 1   
2 2 0 2 1   
3 2 0 2 1   
4 2 0 2 1   
.. ... .. ... ...   
157 2 0 2 1   
159 2 0 3 1   
160 0 0 2 1   
161 2 0 2 1   
162 2 0 2 1   
  
[158 rows x 14 columns]

In [57]:

havedisease.hist(figsize=(14,14))  
plt.show()

![](data:image/png;base64;base64,)

In [58]:

nothavedisease = exclude  
print(nothavedisease)

age sex cp trestbps chol fbs restecg thalach exang oldpeak \  
165 67 1 0 160 286 0 0 108 1 1.5   
166 67 1 0 120 229 0 0 129 1 2.6   
167 62 0 0 140 268 0 0 160 0 3.6   
168 63 1 0 130 254 0 0 147 0 1.4   
169 53 1 0 140 203 1 0 155 1 3.1   
.. ... ... .. ... ... ... ... ... ... ...   
298 57 0 0 140 241 0 1 123 1 0.2   
299 45 1 3 110 264 0 1 132 0 1.2   
300 68 1 0 144 193 1 1 141 0 3.4   
301 57 1 0 130 131 0 1 115 1 1.2   
302 57 0 1 130 236 0 0 174 0 0.0   
  
 slope ca thal target   
165 1 3 2 0   
166 1 2 3 0   
167 0 2 2 0   
168 1 1 3 0   
169 0 0 3 0   
.. ... .. ... ...   
298 1 0 3 0   
299 1 0 3 0   
300 1 2 3 0   
301 1 1 3 0   
302 1 1 2 0   
  
[129 rows x 14 columns]

In [59]:

# Analysising People with Heart Disease

In [60]:

#Analysising Age

In [61]:

age\_max = havedisease.age.max()  
print(age\_max)

76

In [62]:

age\_min = havedisease.age.min()  
print(age\_min)

29

In [63]:

age\_med = havedisease.age.median()  
print(age\_med)

52.0

I was able to analyze their state of health by observing some of the variables in the dataframe. I started by looking at the age of the individuals and found the highest age was 76, the lowest age was 29, and the median age was 52.

In [64]:

# Analysing Resting Blood Preasure

In [65]:

bp\_max = havedisease.trestbps.max()  
print(bp\_max)

180

In [66]:

bp\_min = havedisease.trestbps.min()  
print(bp\_min)

94

In [67]:

bp\_med = havedisease.trestbps.median()  
print(bp\_med)

130.0

A normal blood pressure level is less than 120/80 mmHg. I analyzed the systolic blood pressure of those individuals with heart disease and found the highest reading was 180, the lowest reading was 94, and the median reading was 130. The systolic reading should not go above 140, which is an indicator of high blood pressure.

In [68]:

# Analysing fbs

In [69]:

havedisease.groupby('fbs').target.count().plot.bar(ylim=0)  
plt.show()

![](data:image/png;base64;base64,)

I looked at the blood sugar reading being greater than 120 and found that a small proportion of people had high blood sugar, which is indicative of diabetes.

In [70]:

# Analysing exang

In [71]:

havedisease.groupby('exang').target.count().plot.bar(ylim=0)  
plt.show()

![](data:image/png;base64;base64,)

Only a small percentage of people had exercise-induced angina

In [72]:

# Analysing cp

In [73]:

havedisease.groupby('cp').target.count().plot.bar(ylim=0)  
plt.show()

![](data:image/png;base64;base64,)

I have analyse that majority people with heart disease have non-atypical angina type chest pain. And people with asymptomatic type chest pain are very less compaired to the other types.

In [74]:

# Analysing sex

In [75]:

havedisease.groupby('sex').target.count().plot.bar(ylim=0)  
plt.show()

![](data:image/png;base64;base64,)

There is not much difference between the ratio of male and female that have heart disease.

In [76]:

# Analysing chol

In [77]:

chol\_max = havedisease.chol.max()  
print(chol\_max)

394

In [78]:

chol\_min = havedisease.chol.min()  
print(chol\_min)

126

In [79]:

chol\_med = havedisease.chol.median()  
print(chol\_med)

235.0

I analyzed the Total Cholesterol of those individuals with heart disease and found the highest reading was 394, the lowest reading was 126, and the median reading was 234.

In [80]:

# Analysing restecg

In [81]:

havedisease.groupby('restecg').target.count().plot.bar(ylim=0)  
plt.show()

![](data:image/png;base64;base64,)

I analyzed that people with heart disease having resting electrocardiographic results(restecg) value = 2 are very less compaired with the other two.

In [82]:

# Analysing thalach (maximum heart rate achieved)

In [83]:

mhr\_max = havedisease.thalach.max()  
print(mhr\_max)

202

In [84]:

mhr\_min = havedisease.thalach.min()  
print(mhr\_min)

96

In [85]:

mhr\_med = havedisease.thalach.median()  
print(mhr\_med)

161.5

I analyzed the maximum heart rate achieved of those individuals with heart disease and found the highest reading was 202, the lowest reading was 96, and the median reading was 161.5

In [86]:

# Analysing oldpeak

In [87]:

havedisease.groupby('oldpeak').target.count().plot.bar(ylim=0)  
plt.show()

![](data:image/png;base64;base64,)

I analysed ST depression induced by exercise relative to rest of people that have heart disease and found that most people have 0 ST depression induced by exercise relative to rest

In [88]:

# Analysing slope

In [89]:

havedisease.groupby('slope').target.count().plot.bar(ylim=0)  
plt.show()

![](data:image/png;base64;base64,)

I analysed the slope of the peak exercise ST segment of people that have heart disease and found that most people have value = 2 and least people have value =0

# Train Test split[¶](#Train-Test-split)

In [90]:

x = heart2.drop(columns='target', axis=1)  
y = heart2['target']  
x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.3)

In [91]:

print(x.shape, x\_train.shape, x\_test.shape)

(287, 13) (200, 13) (87, 13)

In [92]:

print(y.shape, y\_train.shape, y\_test.shape)

(287,) (200,) (87,)

In [93]:

Catagory=['You do not have Heart Disease....','Yes you have Heart Disease....RIP in Advance']

# DecisionTreeClassifier[¶](#DecisionTreeClassifier)

In [94]:

from sklearn.tree import DecisionTreeClassifier  
  
  
dt=DecisionTreeClassifier()  
dt.fit(x\_train,y\_train)

Out[94]:

DecisionTreeClassifier()

In [95]:

prediction=dt.predict(x\_test)  
accuracy\_dt=accuracy\_score(y\_test,prediction)\*100

In [96]:

accuracy\_dt

Out[96]:

74.71264367816092

In [97]:

print("Accuracy on training set: {:.3f}".format(dt.score(x\_train, y\_train)))  
print("Accuracy on test set: {:.3f}".format(dt.score(x\_test, y\_test)))

Accuracy on training set: 1.000  
Accuracy on test set: 0.747

In [98]:

prediction

Out[98]:

array([1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1,  
 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0,  
 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1,  
 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1],  
 dtype=int64)

In [99]:

X\_DT=np.array([[60,0,3,150,240,0,1,171,0,0.9,2,0,2]])  
X\_DT\_prediction=dt.predict(X\_DT)

In [100]:

X\_DT\_prediction[0]  
print(Catagory[int(X\_DT\_prediction[0])])

Yes you have Heart Disease....RIP in Advance

In [101]:

#feature importance

In [102]:

print("Feature importances:\n{}".format(dt.feature\_importances\_))

Feature importances:  
[0.04931271 0.01506485 0.2914709 0.08879156 0.04783965 0.  
 0. 0.12151475 0.02667734 0.07844299 0.0120048 0.12502022  
 0.14386024]

In [103]:

def plot\_feature\_importances\_diabetes(model):  
 plt.figure(figsize=(8,6))  
 n\_features = 13  
 plt.barh(range(n\_features), model.feature\_importances\_, align='center')  
 plt.yticks(np.arange(n\_features), x)  
 plt.xlabel("Feature importance")  
 plt.ylabel("Feature")  
 plt.ylim(-1, n\_features)  
plot\_feature\_importances\_diabetes(dt)  
plt.savefig('feature\_importance')

![](data:image/png;base64;base64,)

# LogisticRegressionClassifier[¶](#LogisticRegressionClassifier)

In [104]:

lr=LogisticRegression()  
lr.fit(x\_train,y\_train)

F:\annaconda\lib\site-packages\sklearn\linear\_model\\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.  
  
Increase the number of iterations (max\_iter) or scale the data as shown in:  
 https://scikit-learn.org/stable/modules/preprocessing.html  
Please also refer to the documentation for alternative solver options:  
 https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression  
 n\_iter\_i = \_check\_optimize\_result(

Out[104]:

LogisticRegression()

In [105]:

lrprediction=lr.predict(x\_test)  
accuracy\_lr=accuracy\_score(y\_test,lrprediction)\*100

In [106]:

accuracy\_lr

Out[106]:

85.0574712643678

In [107]:

print("Accuracy on training set: {:.4f}".format(lr.score(x\_train, y\_train)))  
print("Accuracy on test set: {:.4f}".format(lr.score(x\_test, y\_test)))

Accuracy on training set: 0.8850  
Accuracy on test set: 0.8506

In [108]:

lrprediction

Out[108]:

array([1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1,  
 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,  
 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1,  
 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1],  
 dtype=int64)

In [109]:

X\_LR=np.array([[67,1,0,160,286,0,0,108,1,1.5,1,3,2]])  
X\_LR\_lrprediction=dt.predict(X\_LR)

In [110]:

X\_LR\_lrprediction[0]  
print(Catagory[int(X\_LR\_lrprediction[0])])

You do not have Heart Disease....

# Conclusion[¶](#Conclusion)

In [111]:

#Models and their accuracy

In [112]:

print("Decision Tree :: ", accuracy\_score(y\_test,prediction)\*100)  
print("Logistic Regression Classifier :: ", accuracy\_score(y\_test,lrprediction)\*100)

Decision Tree :: 74.71264367816092  
Logistic Regression Classifier :: 85.0574712643678

In [113]:

algorithms=['Decision Tree','LogisticRegression']  
scores=[accuracy\_dt,accuracy\_lr]

In [114]:

sns.set(rc={'figure.figsize':(15,7)})  
plt.xlabel("Algorithms")  
plt.ylabel("Accuracy score")  
  
sns.barplot(algorithms,scores)

F:\annaconda\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.  
 warnings.warn(

Out[114]:

<AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>

![](data:image/png;base64;base64,)

In [ ]: