Data Science Project on Heart Disease

Presented by: -

Donthi Venkata Aravind

- **TABLE OF CONTENT**

- **TABLE OF FIGURES/GRAPHS**

INDEX

Abstract

The health care industries collect huge amounts of data that contain some hidden information, which is useful for making effective decisions. For providing appropriate results and making effective decisions on data, some advanced data mining techniques are used. Nowadays, in healthcare industry, data analysis can save lives by improving the medical diagnosis. And with the huge development in software engineering, different data mining tools are available for researchers, and used to conduct studies and experiments.

Experiments with some python libraries can be used in prediction of a heart diseased person using trained model. A comparative analytical approach will be done for improving prediction accuracy in heart disease and to show its utility to predict disease at an early stage. The results of the study indicate that ensemble techniques, such as bagging and boosting, are effective in improving the prediction accuracy of weak classifiers, and exhibit satisfactory performance in identifying risk of heart disease.

The performance of the process will be further enhanced with a feature selection implementation, and the results significant improvement will be verified with prediction accuracy. + +

Chapter-1

Datasets used in this Analysis

I have used open-source data-set which is available online it is not so huge data-set but it is very convenient for the beginners to analyze data and learn new things in this data set we have used some of the shortcuts. To get to know about those I am displaying them here.

age:	age
sex:	1: male,
	0: female
cp:	chest pain type,
	1: typical angina,
	2: atypical angina,
	3: non- anginal pain,
	4: asymptomatic

trestbps:	resting blood pressure
chol:	serum 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
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

Chapter-2: - Conditions applied on dataset for identifying disease

In this project we have used two important libraries and functions to determine whether the person will suffer from heart disease or not using :-

- Sklearn Model
- Decision Tree Classifier

Sklearn is the library used in this project to train and test the module and to create a module we have used "Decision Tree Classifier".

Example: - (Before training the model)

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 13 columns

Example: - (After training the model)

```
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=10,test_size=0.3,shuffle=False)
```

```
X_train
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
207	60	0	0	150	258	0	0	157	0	2.6	1	2	3
208	49	1	2	120	188	0	1	139	0	2.0	1	3	3
209	59	1	0	140	177	0	1	162	1	0.0	2	1	3
210	57	1	2	128	229	0	0	150	0	0.4	1	1	3
211	61	1	0	120	260	0	1	140	1	3.6	1	1	3

212 rows × 13 columns

This is the major part in this project we need to analyze the data and we have to predict whether the person will suffer from Heart Disease or not

According to my analysis from past testing and on my observations "If our accuracy on test is in between 70%-92% our test will be successful."

Chapter-3: - Creating and training a model

To create a model, we have to import some functions from sklearn

- We use "Decision Tree Classifier" to create a model
- We train and test the data with help of respective functions used in the project

Here's a shot how to create a module

```
from sklearn.tree import DecisionTreeClassifier
```

```
dt=DecisionTreeClassifier()
dt.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

To train the model we have to use some more functions which will be discussed in later chapters

Decision Trees are a class of very powerful Machine Learning model cable of achieving high accuracy in many tasks while being highly interpretable. What makes decision trees special in the realm of ML models is really their clarity of information representation. The "knowledge" learned by a decision tree through training is directly formulated into a hierarchical structure. This structure holds and displays the knowledge in such a way that it can easily be understood, even by non-experts.

Chapter-4

Importing Data Visualization libraries in python

```
In [1]: import sklearn
        import pandas as pd
        import numpy as np
        import plotly as plot
        import plotly.express as px
        import plotly.graph_objs as go
        import cufflinks as cf
        import matplotlib.pyplot as plt
        import seaborn as sns
        import os
        from sklearn.metrics import accuracy_score
        import plotly.offline as pyo
        from plotly.offline import init_notebook_mode,plot,iplot
```

- import pandas as pd

Pandas is a library that we all have to install in our machine locally during python installation or we have to use **pip install pandas.** Simply import the library the current namespace, but rather than using name 'pandas' it can be instructed as pd for our convenience.

- import numpy as np

NumPy is a general Python package that stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object. It is a general purpose array-processing package.

- import matplotlib.pyplot as plt

matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

- import sklearn

sklearn 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 modeling including classification, regression, clustering and dimensionality reduction. Scikit-learn comes loaded with a lot of features such as Supervised learning algorithms, Cross-validation, Unsupervised learning algorithms, Various toy datasets, Feature extraction

- import plotly as plot

Plotly is known for developing and providing online analytics, statistics and graphing tools for individuals or companies. It also develops/provides scientific graphing libraries for Arduino, Julia, MATLAB, Perl, Python, R and REST.

- import plotly.express as px & import plotly.graph_objs as go
- import plotly.offline as pyo
- from plotly.offline import init_notebook_mode,plot,iplot

Plotly Express is a new high-level Python visualization library: it's a wrapper for Plotly.py that exposes a simple syntax for complex charts. It was specifically designed to have a terse, consistent and easy-to-learn API: with just a single import, you can make richly interactive plots in just a single function call, including faceting, maps, animations, and trendlines. It comes with on-board datasets, color scales and themes, and just like Plotly.py, Plotly Express is totally free: with its permissive open-source MIT license, you can use it however you like. Best of all, Plotly Express is fully compatible with the rest of Plotly ecosystem: use it in your Dash apps, export your figures to almost any file format using Orca, or edit them in a GUI with the Jupyter Lab Chart Editor!

- import cufflinks as cf

Cufflinks is another library that connects the Pandas data frame with Plotly enabling users to create visualizations directly from Pandas. The library binds the power of Plotly with the flexibility of Pandas for easy plotting.

- import seaborn as sns

Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. The main idea of Seaborn is that it provides high-level commands to create a variety of plot types useful for statistical data exploration, and even some statistical model fitting.

- import os

Since the os module provides access to system features, the actual options that are available will depend on the underlying operating system. Many of the os options are available for all operating systems, but there are several which are available for UNIX (and most UNIX-based systems, including Linux and the Mac OS), but not for Windows, and some which are available only for certain flavors of UNIX. Since the available commands are system-dependent, when you create a Python program that uses os commands, you will probably want to include a check to find out which os module you're dealing with, using the os.name command

All the procedure will be discussed in next chapter.

Chapter -5 Getting Familiar with the procedure

```
heart=pd.read_csv(r'E:\Data_Science\Heart\heart.csv')
```

We are declaring a variable as “heart” and into that heart variable we are trying to read the csv file which is taken from the open source

In my PC I have saved my file in Local Disk: E and in Data Science folder in Heart subfolder I have saved my CSV file

I have used ‘r’ at the beginning because in any programming languages ‘\’ will be considered as **Escape Characters** so as to avoid this we will use r to avoid the escape character.

```
heart
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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

303 rows × 14 columns

To check whether the data is imported into the variable or not

From this We can say that the data which is available in our CSV file is copied to heart variable.

```
heart['target']
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

In this cell we have displayed that:

Already we have imported all the data from the CSV file into heart and in that dataset, we have a column called “target”. We are trying to display all the data which corresponds to target column.

```
heart.groupby('target').size()
```

```
target
0    138
1    165
dtype: int64
```

We are taking same column name “target” in this code we are trying to group the content based on the value like 0’s-138 1’s-165

Size () is used to get those values it means that count of 0’s and 1’s

```
heart.groupby('target').sum()
```

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	7811	114	66	18547	34650	22	62	19196	76	218.8	161	161	351
1	8662	93	227	21335	39968	23	98	26147	23	96.2	263	60	350

Same as previous this cell’s code looks similar, we group the target column and make then as a sum as we see in its output. Its main thing to add all the values based on target values

```
heart.shape
```

```
(303, 14)
```

Shape-key word

This will help us to understand that how many rows and columns present in the heart variable.

```
heart.size
```

```
4242
```

Size-key word

This might be something little-bit tricky in previous we used to determine the number of unique valued-values as a total count.

But as per this size-key word we can say that it gives "How many cells are used in the CSV file used to enter the data"

By this we can say that Python have huge set of libraries and just we have to why ad where we have use then we can do wonders in any branch of analytics used by python script.

```
heart.describe()
```

	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

Describe ():

This very useful function to get an overview of the data which we have imported to the variable. It gives us all the data of total count, mean, standard deviation, minimum value in that column, Q1, Q2, Q3 of the data, at last we can have maximum of the column in that total info we can get the total of the respective column only.

```
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
```

Info ():

First of all we can have from which library we can have this info () based on the output we can say that it's from pandas library. It gives us number of rows and columns are used to enter the data into the file.

Mainly it gives us title of the columns and Data-type used in that column.

```
heart['target'].unique()
```

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

Unique ():

It is used to check how many different values are present in the particular column. It returns as an array with the data-type of that column.

```
heart.hist(figsize=(15,15))
plt.show()
```

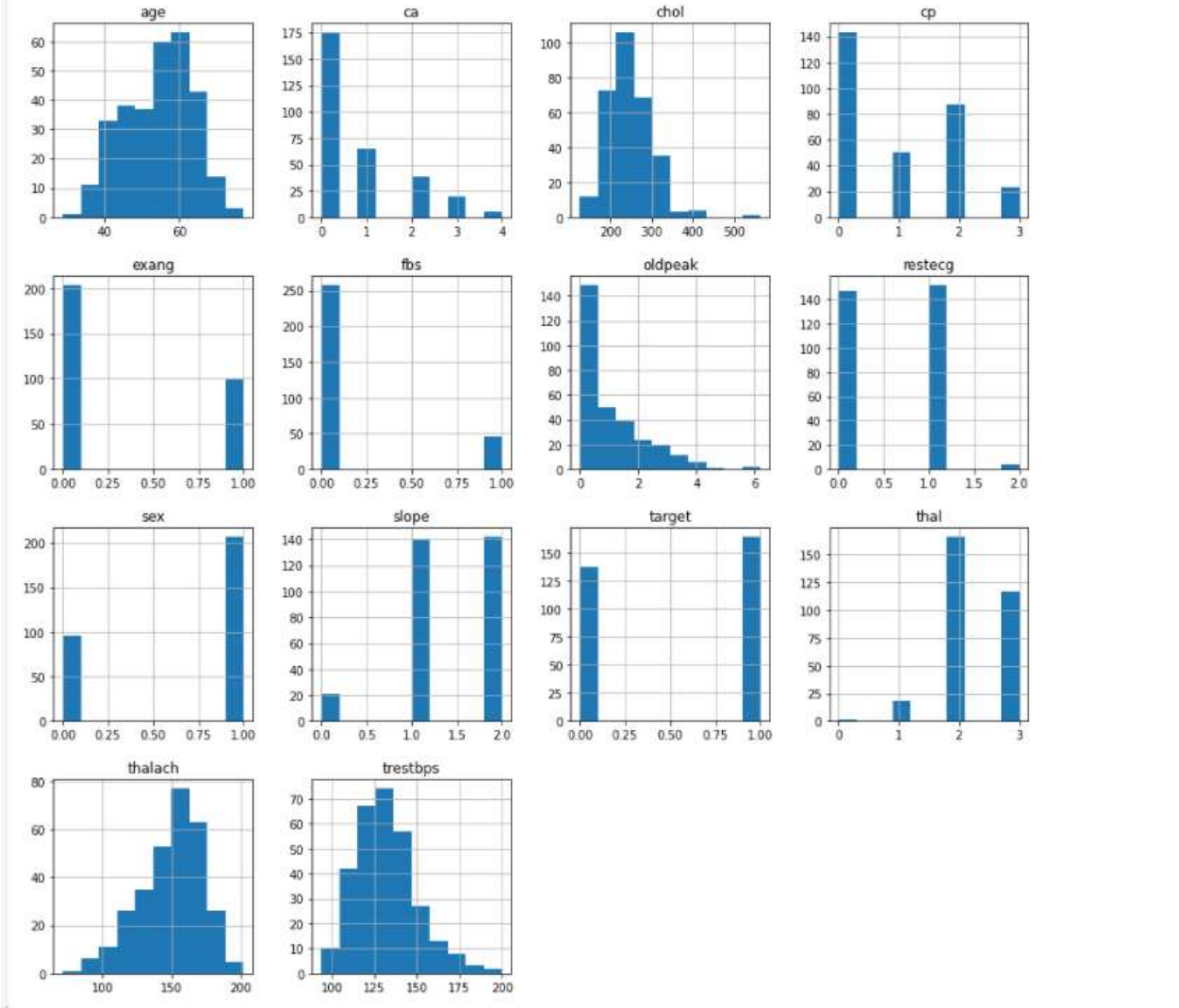

Figure-1

Hist ():

It is used to determine the written data which is in the numerical converted into the graphical format with respect to no of values entered. Best way to represent the data for quick picture about the data in the file.

All the columns in the y-axis and count on x-axis on this basis this is done automatically by the library in Python.

Fig-size: - It gives each graph size to be represented.

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

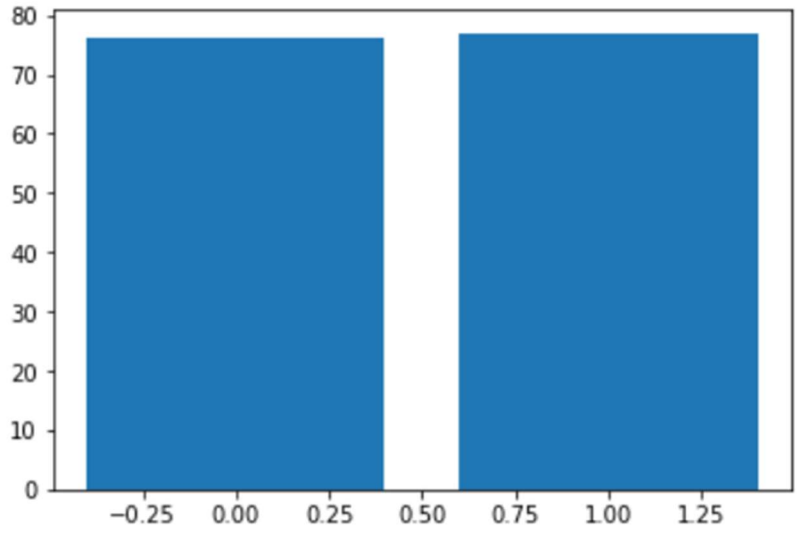

Figure-2

In this cell we have performed by using matplotlib library. We have decided to draw a bar graph with x-axis as Gender and y-axis as age which are from the data given. Plt show () is used to display in the format of graph.

```
px.bar(heart,heart['sex'],heart['target'])
```

Figure-3

Px function is from the plotly.express library. It is drawn across gender and target from the data given in the heart variable. There are 3 arguments to be passed in this function

- ✓ 1->data_variable
- ✓ 2&3-> what we want to display from the data.

```
sns.distplot(heart['thal'])
<matplotlib.axes._subplots.AxesSubplot at 0x20cc5a9e308>
```

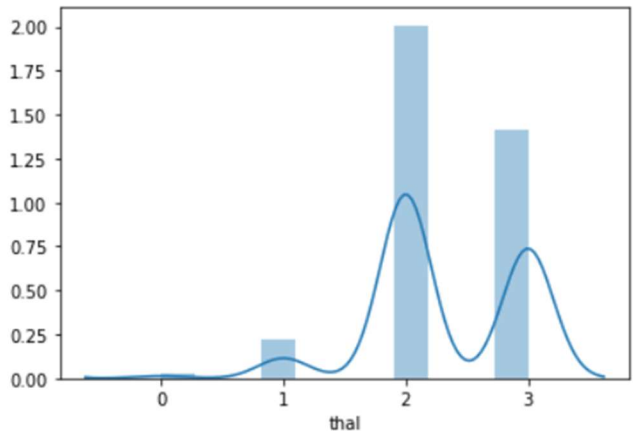

Figure-4

It is imported from the sea born library it is huge library and it contain “iris” data set also in it. From this now we take distplot to draw the graph of the thal. We can pass one or two arguments in this function.

```
sns.distplot(heart['chol'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20cc5715408>
```

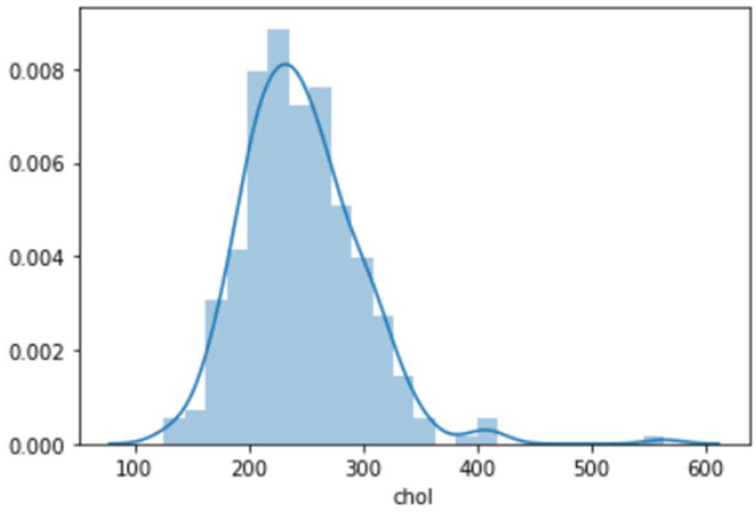

Figure-5

Same as above we have just changed the argument from thal to chol. Both are just the same.

```
numeric_columns=['trestbps','chol','thalach','age','oldpeak']
```

We have declared another variable as numeric_columns in that we have given an array of 5 column names and that will be used later in upcoming cells of code.

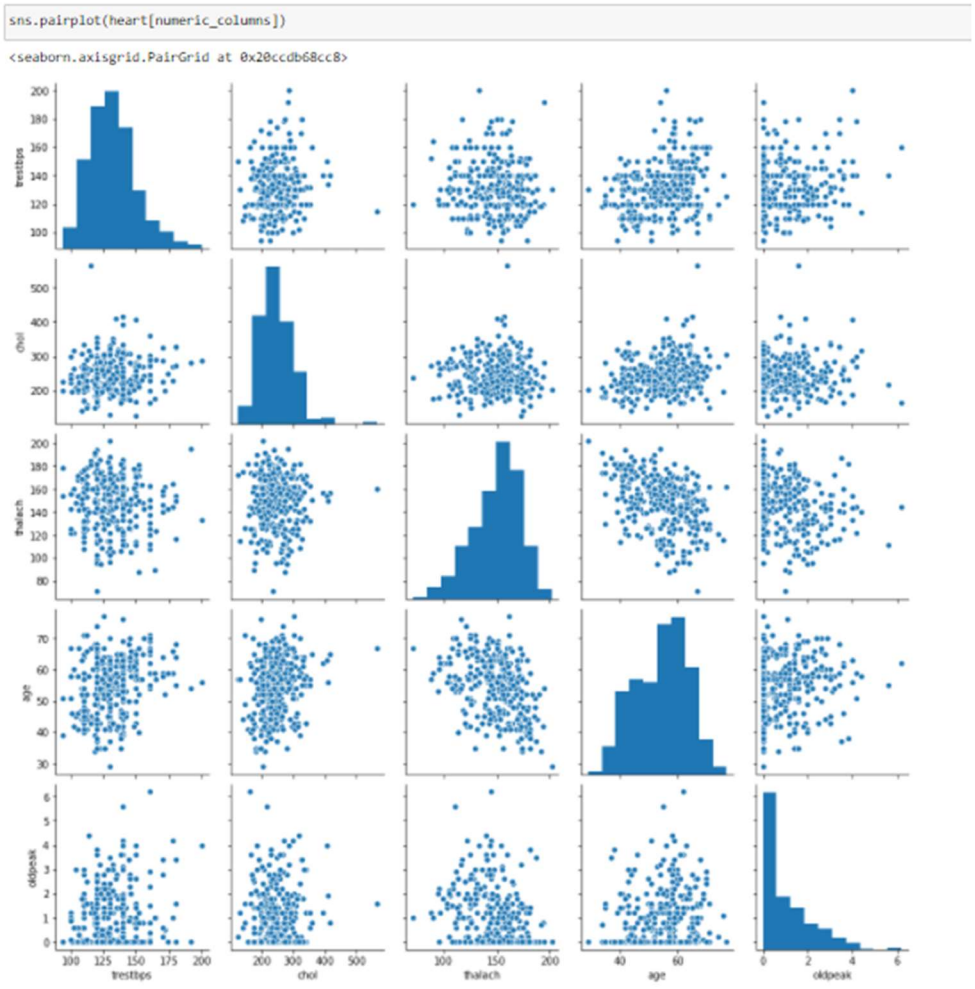

Figure-6

Sns pair plot of numeric columns:

It shows us the all the graphs of the columns present in that variable. Mainly it shows the graphs and by the other graphs we can analyze the relation between the columns.

```
heart['target']
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

Here we display the target values with respect to the column values.

```
y = heart["target"]
```

We have stored heart[‘target’] data in the variable “y”.

```
sns.countplot(y)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20cd0abcc88>
```

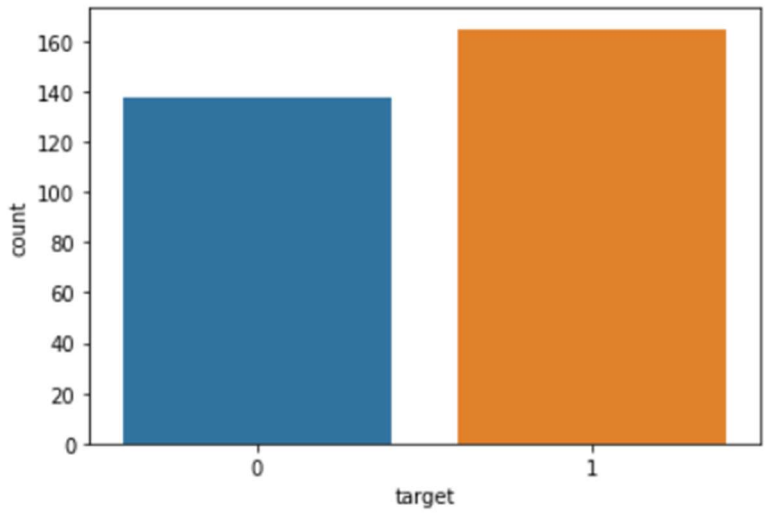

Figure-7

Count is used to count the how many numbers of 0’s and 1’s present in the variable ‘y’. Plot is used to display then in terms of graphs.

```
target_temp = heart.target.value_counts()
```

```
print(target_temp)
```

```
1    165
0    138
Name: target, dtype: int64
```

In previous slide we have plotted in terms of graphs now we are getting in terms of numerical and individual elements.

```
# creation of colletion heatmap
sns.heatmap(heart[numeric_columns].corr(),annot=True,cmap='terrain',linewidths=0.1)
fig=plt.gcf()
fig.set_size_inches(8,6)
plt.show()
```

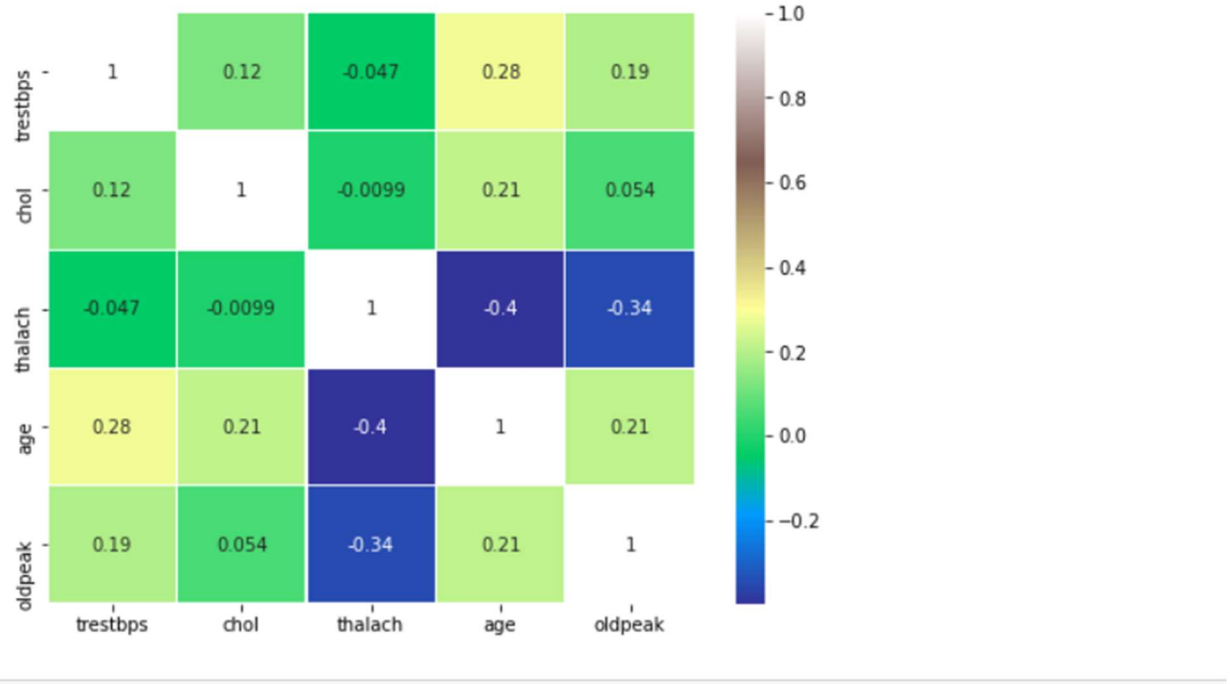

Figure-8

Creating a heat map using python we have choose the attributes whether it is terrain or anything to describe be the color of the hear map corr() is used to get the co-relation between the numeric columns.

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

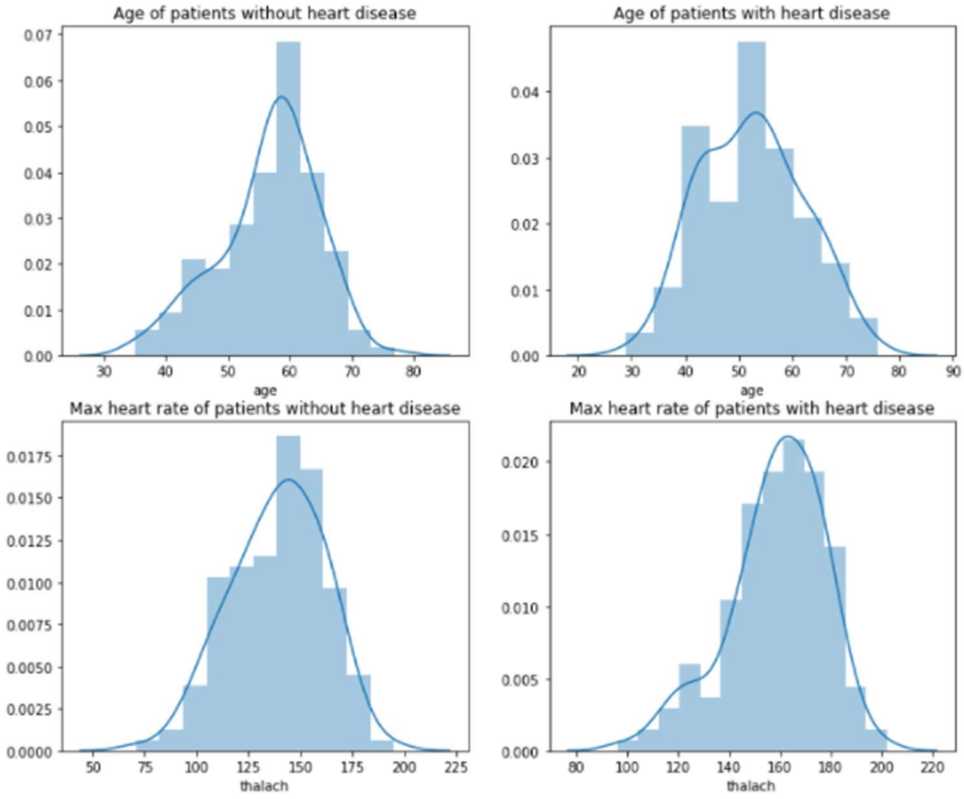

Figure-9

Subplot is used to get number of graphs can be fit into one row,

For example: - subplot (221) It means that there must be 2 rows and 2 columns and that present graph must be at the first position in the row.

Title() is used to Determine the name of the graph or what we will get by reading text of line for better understanding of the graph.

```
plt.figure(figsize=(12,6))
plt.subplot(121)
sns.violinplot(x="target", y="thalach", data=heart, inner=None)
sns.swarmplot(x="target", y="thalach", data=heart, color='w', alpha=0.5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20cd1797788>
```

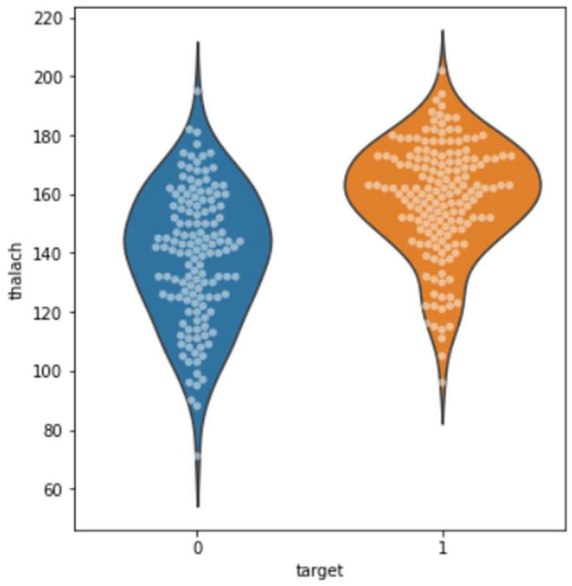

Figure-10

Fig size determines the size of the graph to be displayed in the output.

Violin plot is used to display the data in the shape of the violins. Swarmplot is used to display the data in terms of dots.

```
plt.subplot(122)
sns.swarmplot(x="target", y="thalach", data=heart)
plt.show()
```

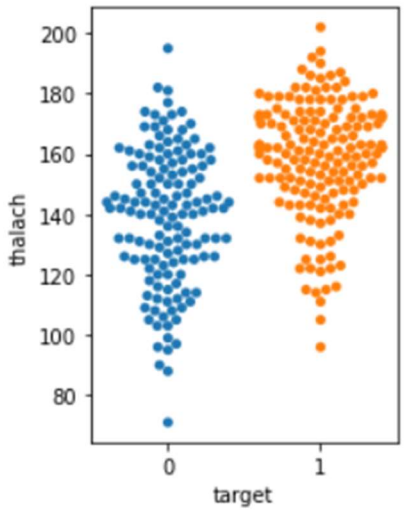

Figure-11

In previous image we have plotted the violin graph with the violin plot but here we have used only swarmplot in this so it looks like a dotted structure or simply a small irregular shape of honey comb.

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

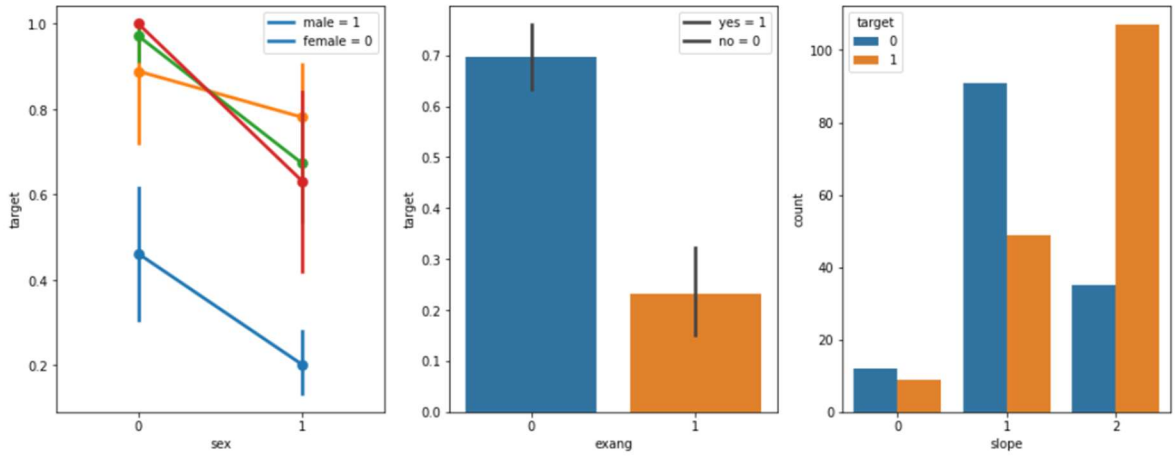

Figure-12

Here in this sub plot we have (131), that indicates 1 row 3 columns and it must be the 1st graph of that row. In this point plot we have seen that chest pain for the females will be more than men

2nd graph is the bar plot with the coordinates (exang, target), 3rd graph has a legend we can determine with the legend () It may be very useful for better analysis of the user.

```
heart['target'].isnull()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
298    False
299    False
300    False
301    False
302    False
Name: target, Length: 303, dtype: bool
```

isNull () is used to determine the NULL value is present in the given column.

```
heart['target'].sum()
```

```
165
```

Sum () is used to do total summation of the column which is in int datatype

```
heart['target'].unique()
```

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

Unique () is used to tell us how many different values present in the column.

```
heart.isnull().sum()
```

```
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
```

Is null and sum ()'s is used to determine if there are any NULL values in the column and it also determine the value if there is any NULL values in the column

```
#Storing in X and y
X,y=heart.loc[:,:'thal'],heart.loc[:,'target']
```

Here we are X as storing variable the values of columns up to thal column.

```
X
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 13 columns

```
y
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

Y stores the vale after thal to target column.

```
X.shape
```

```
(303, 13)
```

```
y.shape
```

```
(303,)
```

Gives us the number of rows and columns present in the variables x has 13 columns and y has only 1 column.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X=heart.drop(['target'],axis=1)
```

From this code we will import a machine learning library sklearn which is very useful for prediction of the data.

From this we will remove the target value from the data we have.

```
X
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 13 columns

```
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=10,test_size=0.3,shuffle=True)
```

```
X_train
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
49	53	0	0	138	234	0	0	160	0	0.0	2	0	2
171	48	1	1	110	229	0	1	168	0	1.0	0	0	3
223	56	0	0	200	288	1	0	133	1	4.0	0	2	3
58	34	1	3	118	182	0	0	174	0	0.0	2	0	2
154	39	0	2	138	220	0	1	152	0	0.0	1	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
156	47	1	2	130	253	0	1	179	0	0.0	2	0	2
123	54	0	2	108	267	0	0	167	0	0.0	2	0	2
15	50	0	2	120	219	0	1	158	0	1.6	1	0	2
125	34	0	1	118	210	0	1	192	0	0.7	2	0	2
265	66	1	0	112	212	0	0	132	1	0.1	2	1	2

212 rows × 13 columns

Here we are trying to train the model and we shuffling the to train the model, test size that we should take only 70% of the data.

```
X_test
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
246	56	0	0	134	409	0	0	150	1	1.9	1	2	3
183	58	1	2	112	230	0	0	165	0	2.5	1	1	3
229	64	1	2	125	309	0	1	131	1	1.8	1	0	3
126	47	1	0	112	204	0	1	143	0	0.1	2	0	2
184	50	1	0	150	243	0	0	128	0	2.6	1	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
69	62	0	0	124	209	0	1	163	0	0.0	2	0	2
21	44	1	2	130	233	0	1	179	1	0.4	2	0	2
210	57	1	2	128	229	0	0	150	0	0.4	1	1	3
78	52	1	1	128	205	1	1	184	0	0.0	2	0	2
174	60	1	0	130	206	0	0	132	1	2.4	1	2	3

91 rows × 13 columns

Here we are displaying the testing model which we have tested in the above case.

```
y_test
```

```
246    0
183    0
229    0
126    1
184    0
      ..
69     1
21     1
210    0
78     1
174    0
Name: target, Length: 91, dtype: int64
```

We are displaying the values stored in the y_test.

```
print ("train_set_x shape: " + str(X_train.shape))
print ("train_set_y shape: " + str(y_train.shape))
print ("test_set_x shape: " + str(X_test.shape))
print ("test_set_y shape: " + str(y_test.shape))
```

```
train_set_x shape: (212, 13)
train_set_y shape: (212,)
test_set_x shape: (91, 13)
test_set_y shape: (91,)
```

Here we are printing number of rows and columns in the respective test and train modules.

```
#Model
#Decision Tree Classifier
Catagory=['No....but i pray you will not get Heart Disease or at leaset Corona Virus Soon...','Yes you have Heart Disease....RIP in Advance']
```

We are creating a model with the Decision tree Classifier and we have taken an statement in the form of array so that it may predict data.

```
from sklearn.tree import DecisionTreeClassifier
```

```
dt=DecisionTreeClassifier()
dt.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

To create a model we have to import Decision tree Classifier from sklearn.tree library.

Chapter-6 Test Cases

```
prediction=dt.predict(X_test)
accuracy_dt=accuracy_score(y_test,prediction)*100
accuracy_dt
```

```
74.72527472527473
```

We have dt as Decision tree Classifier and that data we are predicting the data in the X_test data.

Accuracy score will predict us the data in correct manner. We re printing the accuracy in terms of percentage we have got 74%. It is very good while predicting the data.

```
print("Accuracy on training set: {:.3f}".format(dt.score(X_train, y_train)))
```

```
Accuracy on training set: 1.000
```

```
print("Accuracy on test set: {:.3f}".format(dt.score(X_test, y_test)))
```

```
Accuracy on test set: 0.747
```

Here we are printing the score of x train, y train and x test and y test.

```
y_test
```

```
246    0
183    0
229    0
126    1
184    0
      ..
69     1
21     1
210    0
78     1
174    0
Name: target, Length: 91, dtype: int64
```

We are printing the y test data stored in that variable.

```
prediction
```

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

Prediction values of each in terms of array which are taken from the dt.predict of x_test.

```
X_DT=np.array([[63 ,1, 3,145,233,1,0,150,0,2.3,0,0,1]])
X_DT_prediction=dt.predict(X_DT)
```

```
X_DT_prediction[0]
```

```
1
```

We are taking the data of the 1st row in np.array

In the prediction we are predicting the data which is stored int X_DT variable.

```
X_DT_prediction[0]
```

```
1
```

```
print(Catagory[int(X_DT_prediction[0])])
```

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

We are trying to print the prediction value in the 1st row of code and in the catagory array we have stored and we are showing the array value in it will display in the content present in the catagory array.

```
#Feature Importance in Decision Trees
print("Feature importances:\n{}".format(dt.feature_importances_))
```

```
Feature importances:
[0.03817042 0.03461456 0.33832546 0.094626   0.09938805 0.
 0.02266618 0.08250434 0.04724994 0.05318457 0.06606553 0.11770256
 0.00550238]
```

Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples

```
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')
```

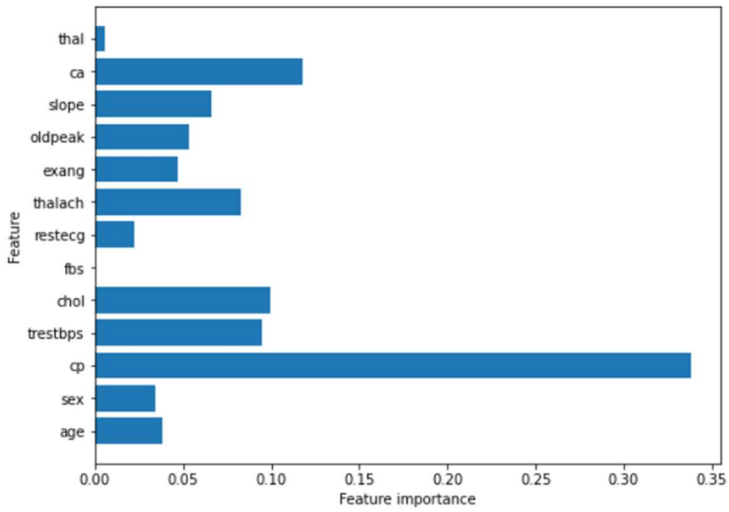

Figure-13

We are creating a function with the name plot_feature_importances_diabetes, it will draw the graph Feature and Feature importance.

```
# KNN

sc=StandardScaler().fit(X_train)
X_train_std=sc.transform(X_train)
X_test_std=sc.transform(X_test)
```

Here we are assigning the variables for different type of operations to transform to standard scale.

```
X_test_std
```

```
array([[ 0.18111199, -1.35154233, -0.97043553, ..., -0.6067969 ,
         1.33369489,  1.22676132],
       [ 0.39865161,  0.73989544,  0.97963397, ..., -0.6067969 ,
         0.33105902,  1.22676132],
       [ 1.05127045,  0.73989544,  0.97963397, ..., -0.6067969 ,
        -0.67157686,  1.22676132],
       ...,
       [ 0.2898818 ,  0.73989544,  0.97963397, ..., -0.6067969 ,
         0.33105902,  1.22676132],
       [-0.25396724,  0.73989544,  0.00459922, ...,  0.98136289,
        -0.67157686, -0.41927286],
       [ 0.61619122,  0.73989544, -0.97043553, ..., -0.6067969 ,
         1.33369489,  1.22676132]])
```

All the data which is in the X_test have been normalized into a standard format.

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn=KNeighborsClassifier(n_neighbors=4)
knn.fit(X_train_std,y_train)
```

```
KNeighborsClassifier(n_neighbors=4)
```

We have declared neighbor classifiers as 4 and we have done fit () to fit X_train_std and y_train.

```
prediction_knn=knn.predict(X_test_std)
accuracy_knn=accuracy_score(y_test,prediction_knn)*100
```

```
accuracy_knn
```

```
84.61538461538461
```

The prediction of knn score is damme high. Its result is very nice compared to the previous comparision.

```
print("Accuracy on training set: {:.3f}".format(knn.score(X_train, y_train)))
```

```
Accuracy on training set: 0.373
```

```
print("Accuracy on test set: {:.3f}".format(knn.score(X_test, y_test)))
```

```
Accuracy on test set: 0.516
```

Here we are printing the score of x train, y train and x test and y test of knn

```
k_range=range(1,26)
scores={}
scores_list=[]
```

```
for k in k_range:
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_std,y_train)
    prediction_knn=knn.predict(X_test_std)
    scores[k]=accuracy_score(y_test,prediction_knn)
    scores_list.append(accuracy_score(y_test,prediction_knn))
```

This code is taking the data and represented to form as a graph.

```
plt.plot(k_range,scores_list)
```

```
[<matplotlib.lines.Line2D at 0x21d90092c88>]
```

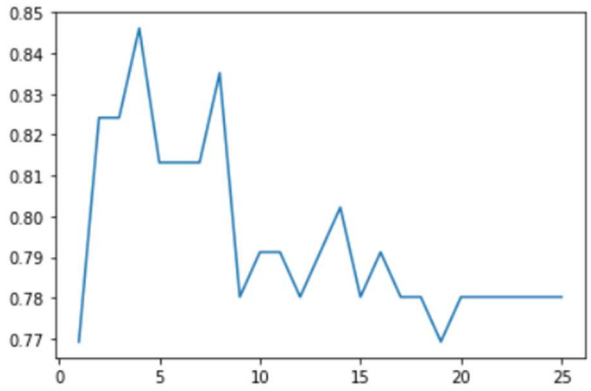

Figure-14

Data is representing in the form of graph.

```
px.line(x=k_range,y=scores_list)
```

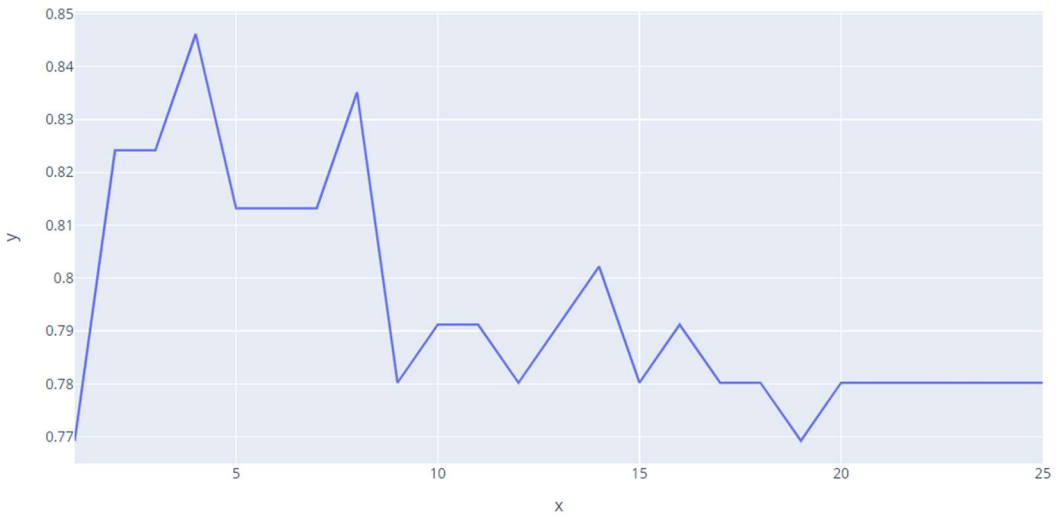

Figure-15

Graph is drawn between the k_range and scores_list.

```
X_knn=np.array([[63 ,1, 3,145,233,1,0,150,0,2.3,0,0,1]])
X_knn_std=sc.transform(X_knn)
X_knn_prediction=dt.predict(X_knn)
```

```
X_knn_std
```

```
array([[ 0.94250064,  0.73989544,  1.95466871,  0.75961822, -0.30064937,
         2.37170825, -0.9841849 ,  0.01848325, -0.6723502 ,  1.10653103,
        -2.1949567 , -0.67157686, -2.06530703]])
```

Prediction values of each in terms of array which are taken from the dt.predict of x_test from knn.

```
(X_knn_prediction[0])
```

```
1
```

```
print(Catagory[int(X_knn_prediction[0])])
```

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

We are trying to print the prediction value in the 1st row of code and in the catagory array we have stored and we are showing the array value in it will display in the content present in the catagory array from knn.

```
sns.set(rc={'figure.figsize':(15,7)})
```

```
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")
sns.barplot(algorithms,scores)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x21d900d0f88>
```

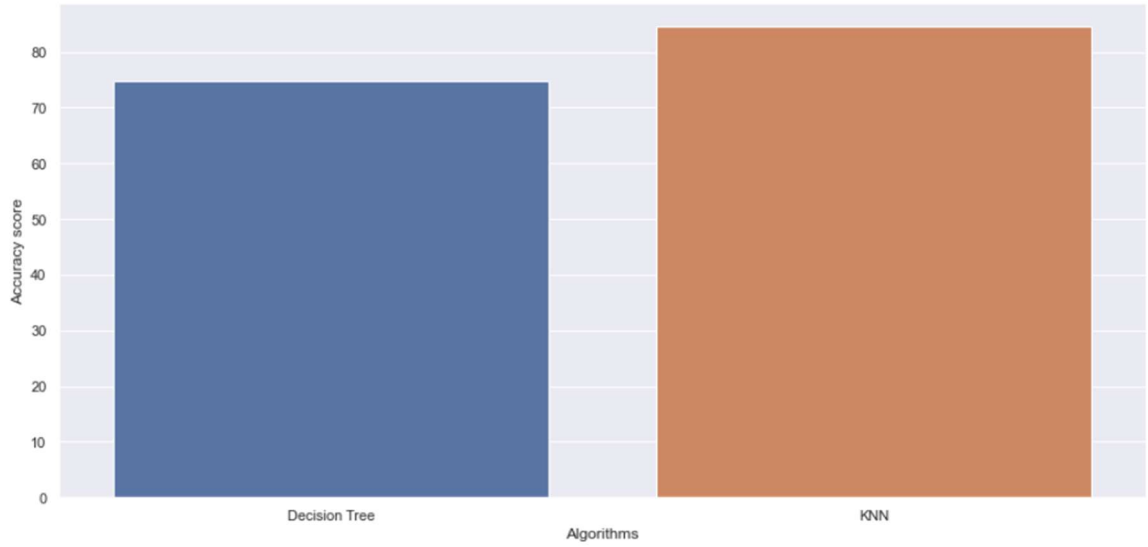

Figure-16

We have given figure size as 15 x 7 and we have drawn graph between the algorithms and accuracy. We have more accuracy in KNN when compares to Decision Tree.

Conclusion

The project performed the different operations to get the maximum accuracy of disease identification from given data-set. The performance of the process has further enhanced with a feature selection implementation, and the results presented in the thesis and it is observed that there is significant improvement in prediction of the disease with high accuracy.