Contents:

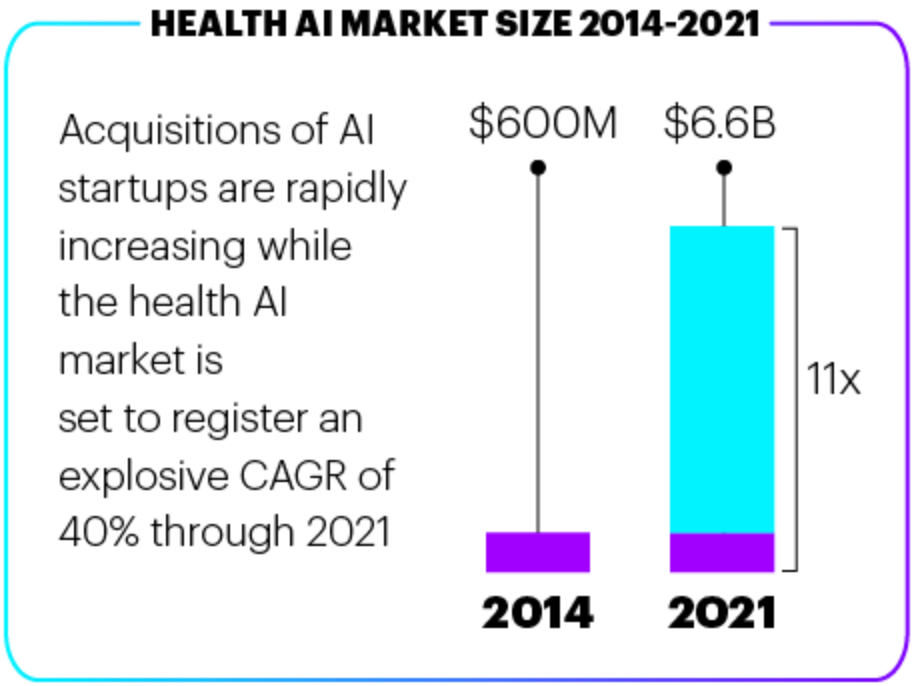
1.Introduction

Heart disease is a serious cause for indian people. It is growing rapidly as study shows. The number of people affected by cardiac diseases has doubled, the study estimated. About 54.5 million people were affected by the disease in 2016 compared to 25.7 million in 1990.

Five are major risk factors: Bad food habits (54.4%), hypertension (56.6%), air pollution (31%), high cholesterol (29.4%) and tobacco usage (18.9%). Among tobacco users, in 83% smoking was seen as a major risk.

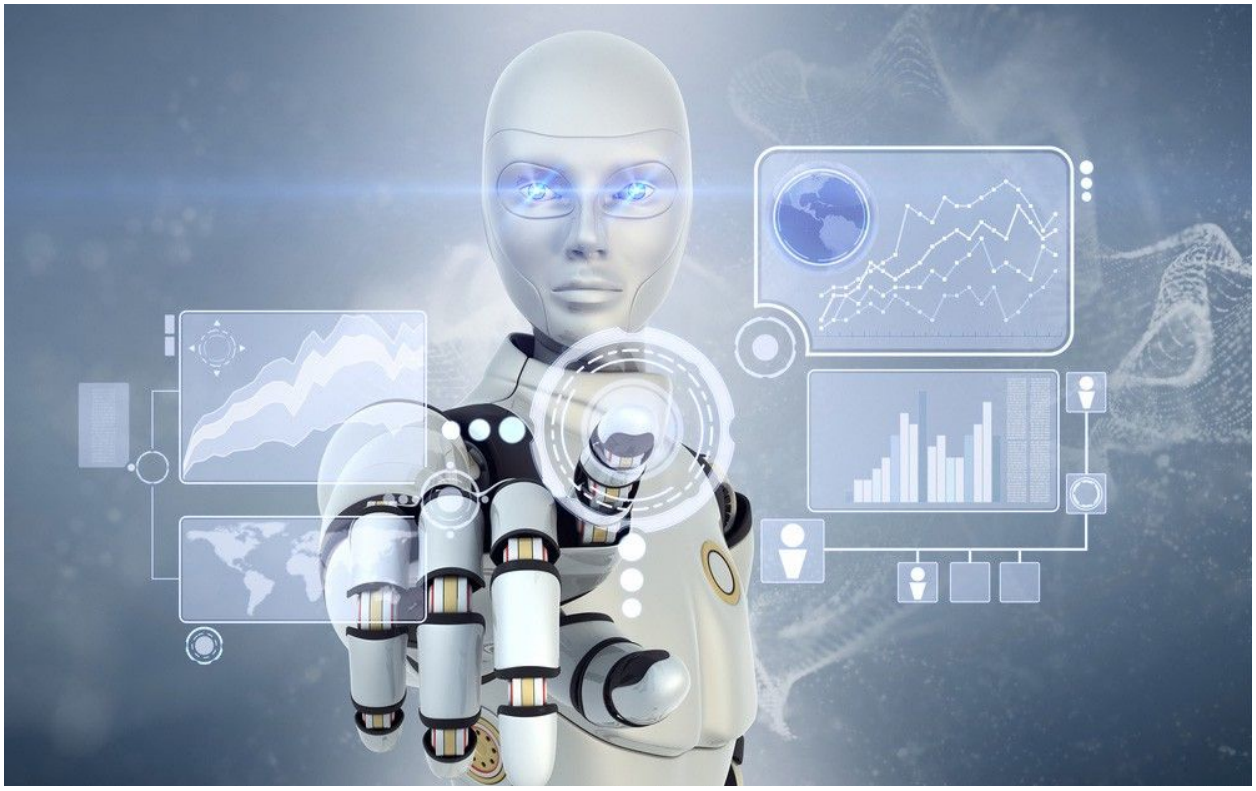
The power of Artificial Intelligence is echoing across many industries. But its impact on healthcare is truly life-changing. With its ability to mimic human cognitive functions, AI is bringing a paradigm shift in the healthcare industry.

The below chart shows the market about health care in AI



Source: Accenture analysis

The ability of AI to use sophisticated algorithms and learn features from a massive amount of data is truly commendable. With the help of these algorithms, insights for assisting clinical practice can be obtained. AI can be equipped with self-correcting and learning abilities which help the system get better accuracy based on the feedback it receives.
Therefore, it gets better with time. These AI systems can help physicians in many ways. Since they are armed with a lot of information, they can assist in clinical decision making. Also, diagnostic errors and therapeutic errors can be minimized.



2.Objective

Recent advancements in AI have fueled discussion of whether AI doctors will replace human doctors in the future. While the idea of replacing human doctors may sound absurd, but AI can help human physicians to make better decisions. In certain areas of healthcare like radiology, it can replace human judgment entirely.

Here we created a model to predict the data of heart disease . Here we take dataset to see which are the components are impacting on the heart disease. Here we use Logistic Regression and DecisionTree to predict our model.

3.Problem Statement

We have a data which classified if patients have heart disease or not according to features in it. We will try to use this data to create a model which tries predict if a patient has this disease or not. We will use logistic regression (classification) and Decision Tree algorithm.

*age — age in years
* sex — (1 = male; 0 = female)
* cp — chest pain type
* trestbps — resting blood pressure (in mm Hg on admission to the hospital)
* chol — serum cholestoral in mg/dl
* fbs — (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
* restecg — resting electrocardiographic results
* thalach — maximum heart rate achieved
* exang — exercise induced angina (1 = yes; 0 = no)
* 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 — 3 = normal; 6 = fixed defect; 7 = reversable defect
* target — have disease or not (1=yes, 0=no)

These are the variables we use to predict the heart disease.

4.Methodology

Packages:

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
import warnings
warnings.filterwarnings('ignore')
```

These are the packages we are gong to use in our model for prediction.

Data Exploration:

Reading the dataset

#reading csv file
data = pd.read_csv('heart_disease_data.csv')
data.head()

	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

df.target.value_counts()

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

df.groupby('target').mean()

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449	0.550725	1.585507	1.166667	1.166667	2.543478
1	52.496970	0.563636	1.375758	129.303030	242.230303	0.139394	0.593939	158.466667	0.139394	0.583030	1.593939	0.363636	2.121212

data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age         303 non-null int64
sex         303 non-null int64
cp          303 non-null int64
trestbps    303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalach     303 non-null int64
exang       303 non-null int64
oldpeak     303 non-null float64
slope       303 non-null int64
ca          303 non-null int64
thal        303 non-null int64
target      303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Missing Value:

Here we have no missing value so we add missing value to the dataset.

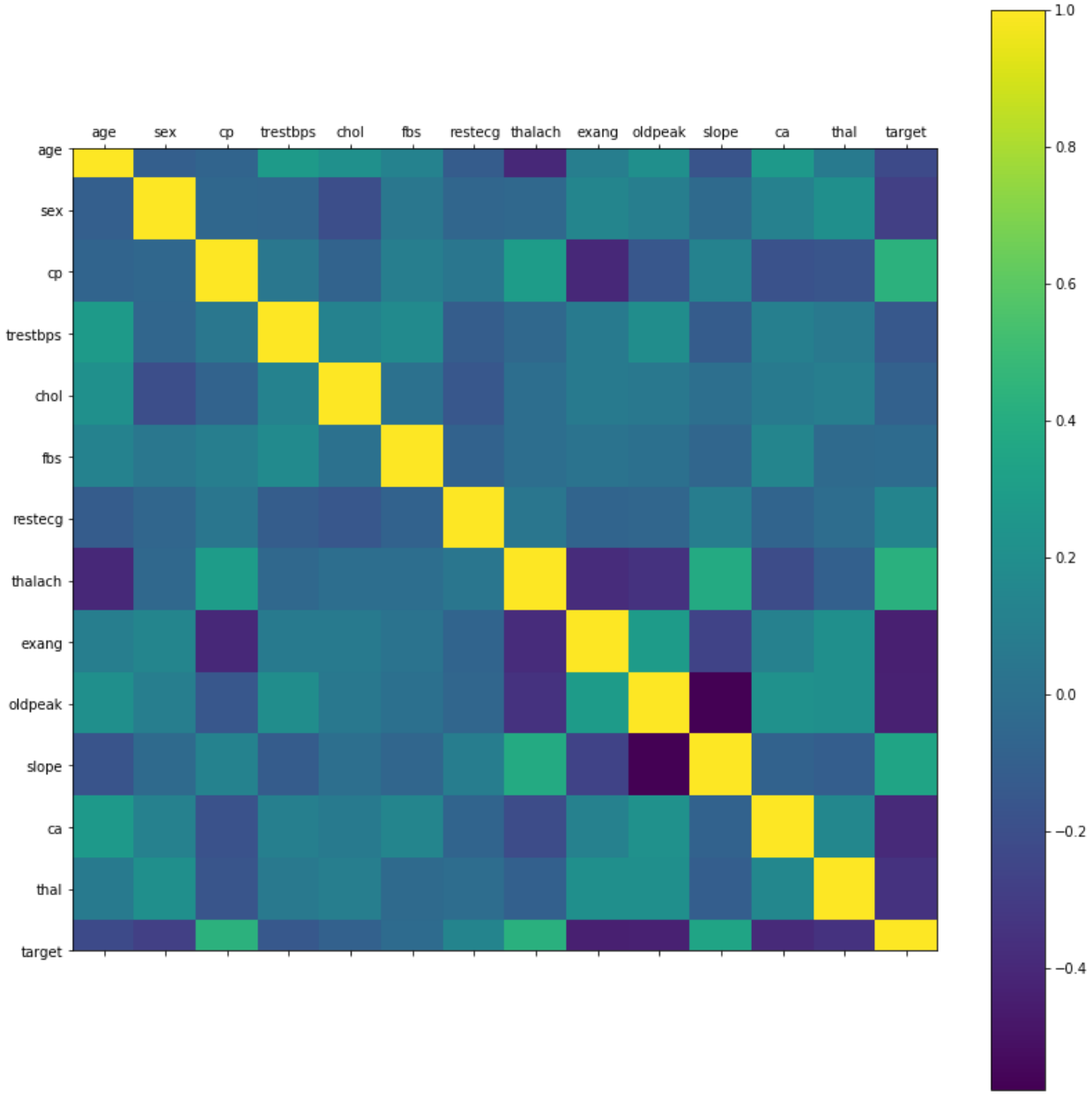
```
#Adding missing value to the dataset
a=X[['age','trestbps','chol','thalach','oldpeak']]
X=X.drop(a,axis=1)
a=a.mask(np.random.random(a.shape)<0.2)
a.isnull().sum()

#merging two dataset
X=pd.concat([X, a.reindex(X.index)], axis=1)

#missing value analysis
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan,strategy="mean")
imputer=imputer.fit(X[['age','trestbps','chol','thalach','oldpeak']])
X[['age','trestbps','chol','thalach','oldpeak']]=imputer.transform(X[['age','trestb
ps','chol','thalach','oldpeak']])
```

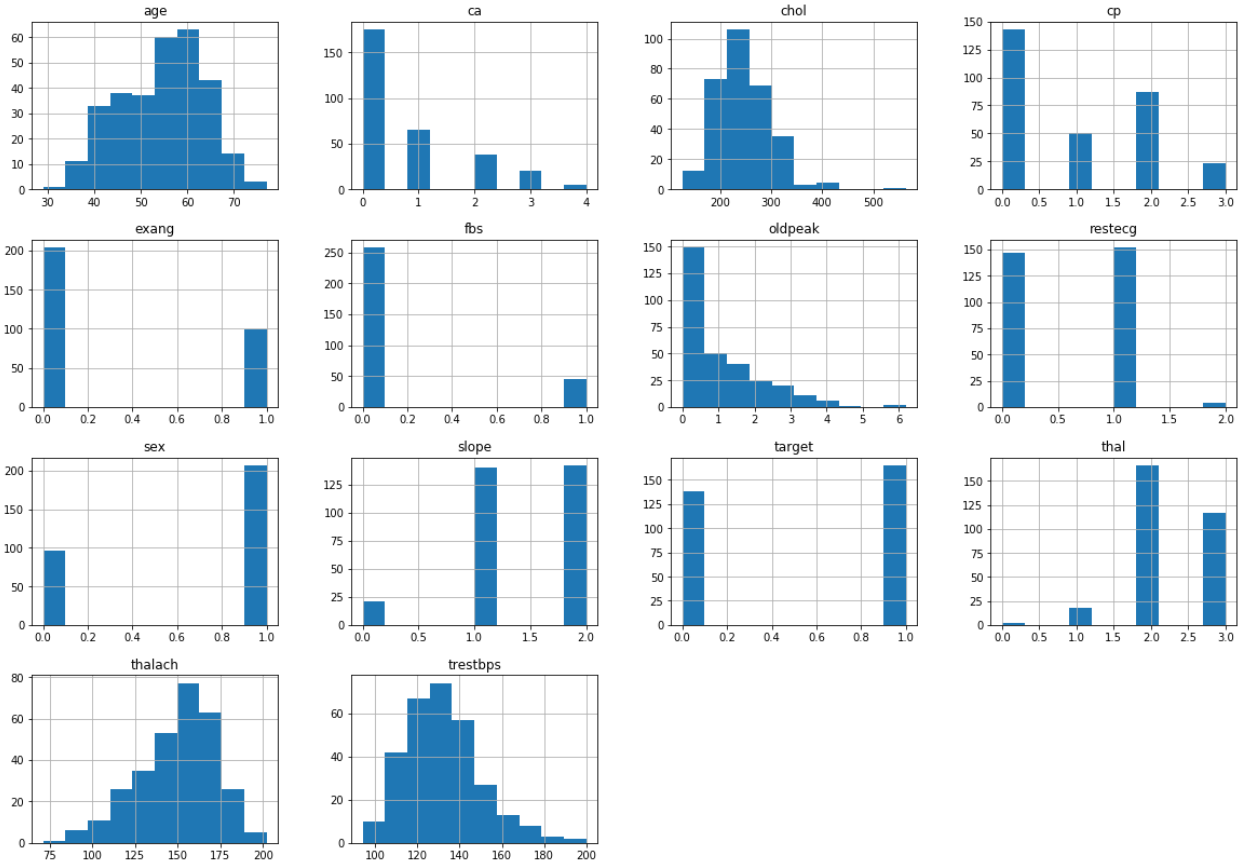
Correlation Matrix:

For better visibility of data and correlation of variables we use correlation matrix .



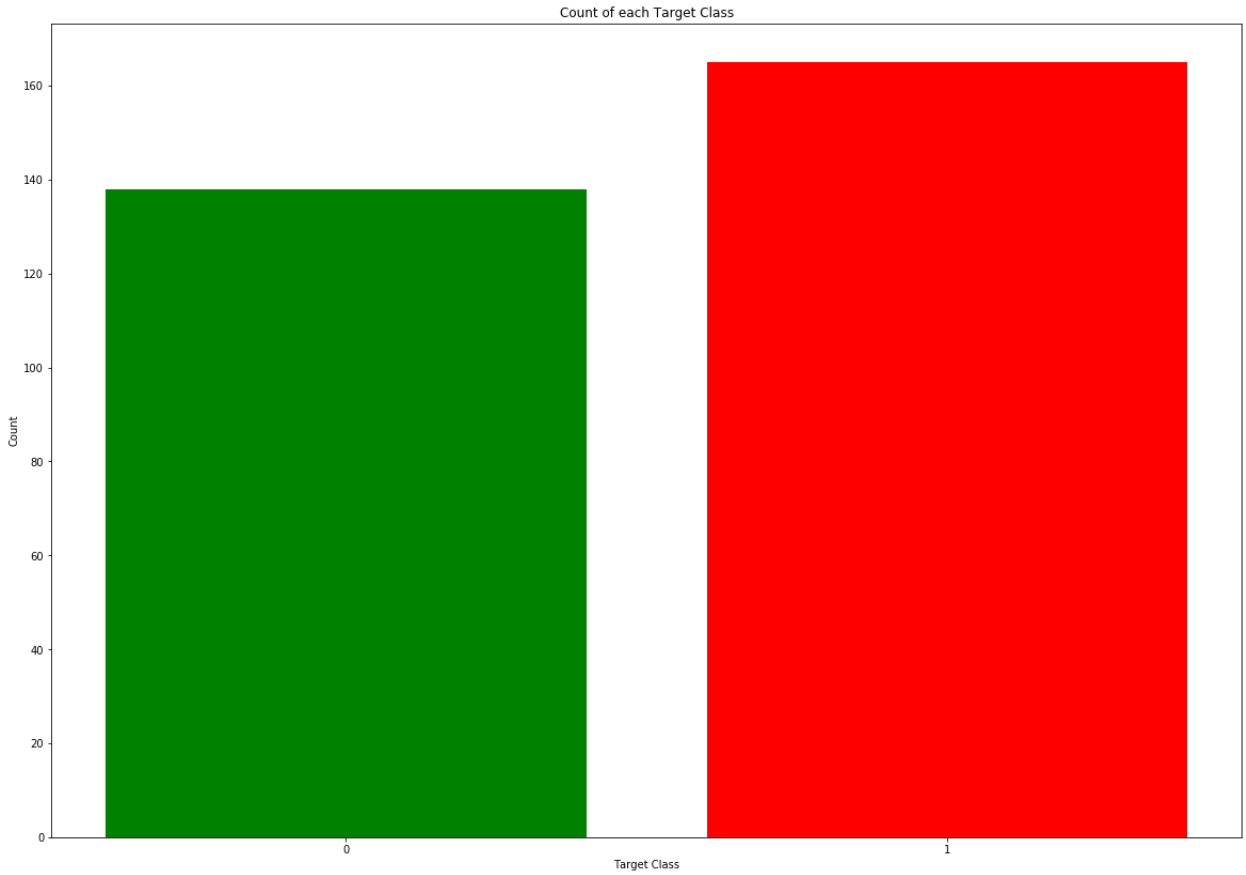
Histogram:

It shows how each feature and label is distributed along different ranges, which further confirms the need for scaling.



Barplot:

It shows the distribution of target variable.



Data Preprocessing:

As we are going to train our model upon categorical variable we need to add dummy variable for the categorical variable. As gender is male and female we take it to 1 and 0.

```
from sklearn.preprocessing import StandardScaler
data = pd.get_dummies(data, columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])
standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
data[columns_to_scale] = standardScaler.fit_transform(data[columns_to_scale])
```

Spliting Dataset:

Here we spliting the dataset to train and test set.

```
X = data.drop(['target'], axis = 1)
y = data['target']
#splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 1)
```

Machine learning Model:

The models we use are:

Decision Tree:

Decision Tree is a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves.

This classifier creates a decision tree based on which, it assigns the class values to each data point. Here, we can vary the maximum number of features to be considered while creating the model. I range features from 1 to 30 (the total features in the dataset after dummy columns were added).

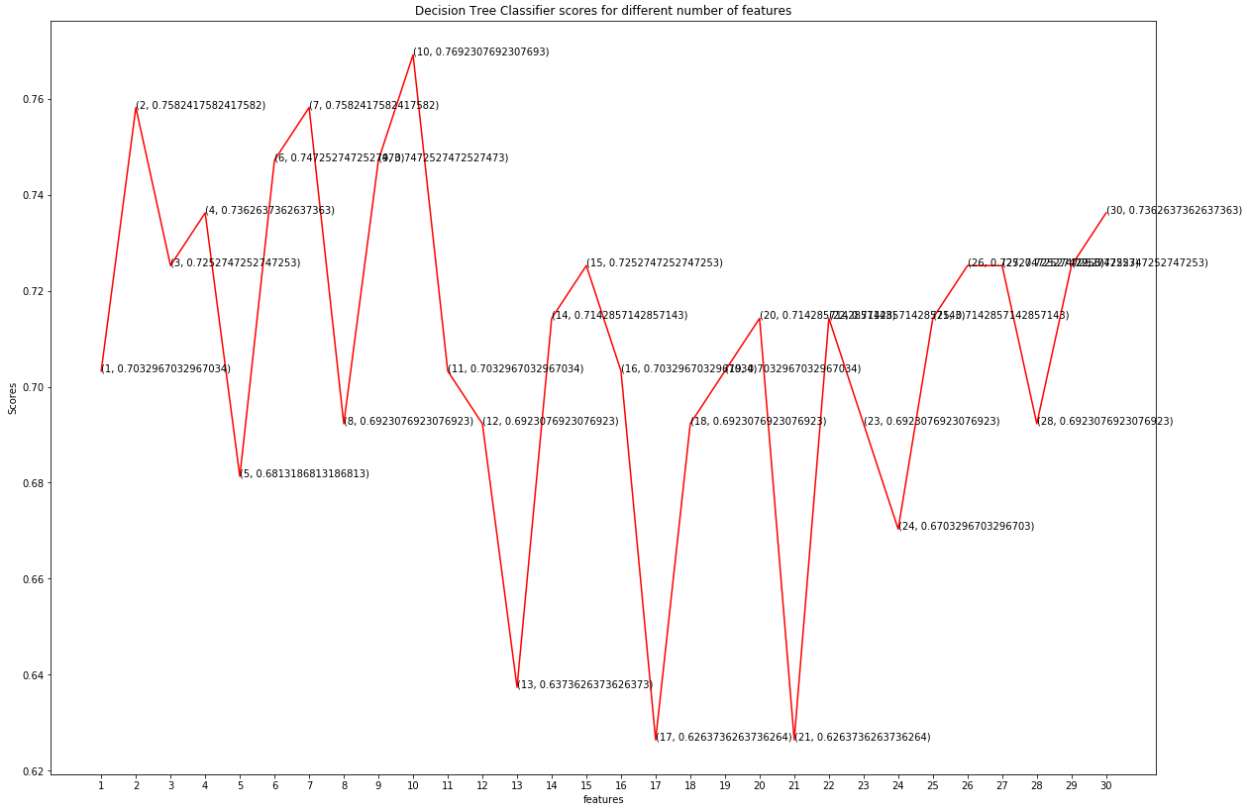
```
dtscores = []
for i in range(1, len(X.columns) + 1):
    dtclassifier = DecisionTreeClassifier(max_features = i, random_state = 0)
    dtclassifier.fit(X_train, y_train)
    dtscores.append(dtclassifier.score(X_test, y_test))
plt.plot([i for i in range(1, len(X.columns) + 1)], dtscores, color = 'red')
for i in range(1, len(X.columns) + 1):
    plt.text(i, dtscores[i-1], (i, dtscores[i-1]))
plt.xticks([i for i in range(1, len(X.columns) + 1)])
plt.xlabel('features')
plt.ylabel('Scores')
plt.title('Decision Tree Classifier scores for different number of features')
```

Confusion matrix:

```
#calculating accuracy
from sklearn.metrics import accuracy_score
dt_pred=dtclassifier.predict(X_test)
print (round(accuracy_score(y_test, dt_pred )*100,2))

#showing confusion matrix
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, dt_pred )
print(cm)
```

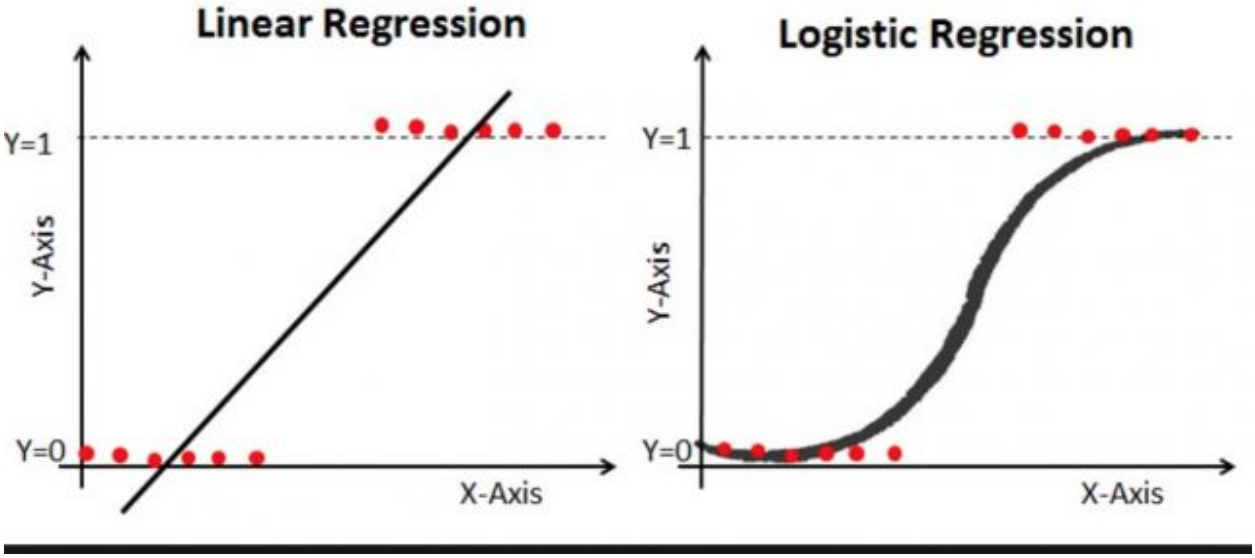


accuracy=73.63

Confusion matrix=[[33 8]
[16 34]]

Logistic Regression:

Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability.



Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Some of the examples of classification problems are Email spam or not spam, Online transactions Fraud or not Fraud, Tumor Malignant or Benign. Logistic regression transforms its output using the logistic sigmoid function to return a probability value.

```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
model_1 =
LogisticRegression(penalty='l1',solver='liblinear',random_state=0,multi_class='ovr'
,C=0.1)
model_1.fit(X_train,y_train)
```

Confusion matrix:

```
print(round(accuracy_score(y_test,y_pred_1)*100))
cm1=confusion_matrix(y_test,y_pred_1)
print(cm1)
```

accuracy=78.0

Confusion matrix=[[30 11]
[9 41]]

Conclusion:

Here I have successfully predicted the risk of heart attack using decision tree and logistic regression.Logistic regression gave highest accuracy of 78%. Here i used only two models for classification problem and there are many more classification algorithms out there for better prediction.