# HEART DISEASE PREDICTION USING KNN ALGORITHM

## PROBLEM DESCRIPTION

About 610,000 people die of heart disease in the United States every year– that's 1 in every 4 deaths. Heart disease is the leading cause of death for both men and women. More than half of the deaths due to heart disease in 2009 were in men. Coronary Heart Disease (CHD) is the most common type of heart disease, killing over 370,000 people annually. Every year about 735,000 Americans have a heart attack. Of these, 525,000 are a first heart attack and 210,000 happen in people who have already had a heart attack.

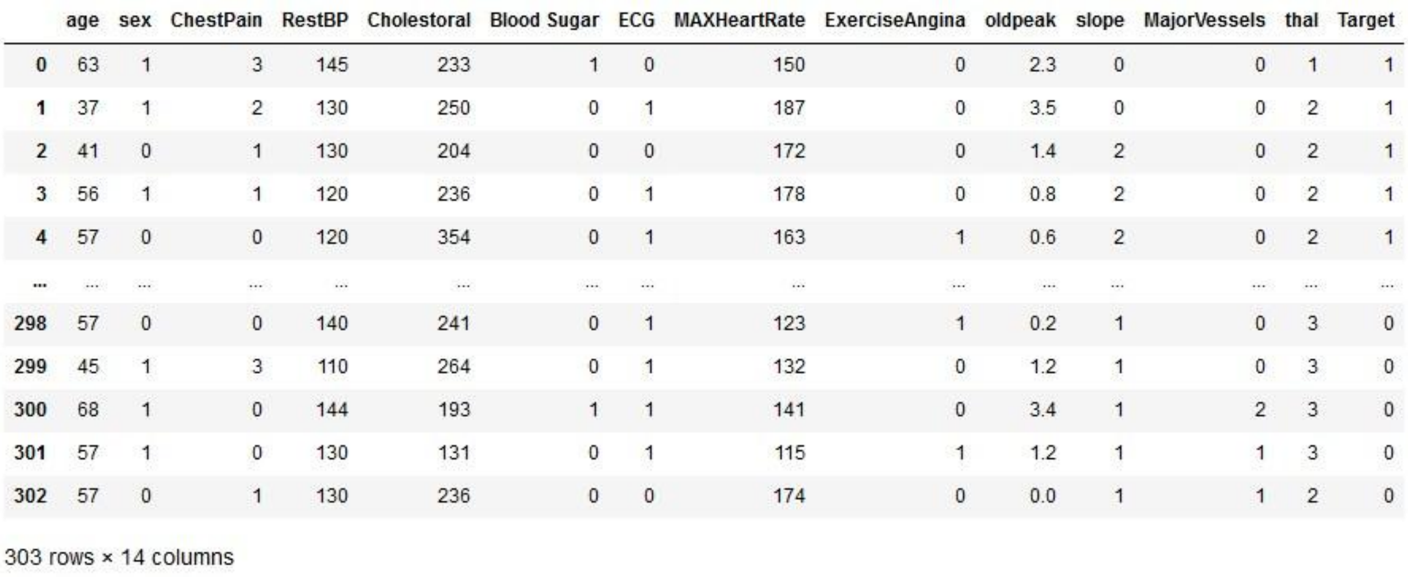
**Heart disease** describes a range of conditions that affect your heart. Diseases under the heart disease umbrella include blood vessel diseases, such as coronary artery disease, heart rhythm problems (arrhythmias) and heart defects you're born with (congenital heart defects), among others. The term "heart disease" is often used interchangeably with the term "cardiovascular disease". Cardiovascular disease generally refers to conditions that involve narrowed or blocked blood vessels that can lead to a heart attack, chest pain (angina) or stroke. Other heart conditions, such as those that affect your heart's muscle, valves or rhythm, also are considered forms of heart disease.

Heart disease is one of the biggest causes of morbidity and mortality among the population of the world. **Prediction of cardiovascular disease is regarded as one of the most important subjects in the section of clinical data analysis.** The amount of data in the healthcare industry is huge. Data mining turns the large collection of raw healthcare data into information that can help to make informed decisions and predictions.

I will be applying Machine Learning algorithm, which proves to be effective for classifying whether a person is suffering from heart disease or not, using one of the most used **Cleveland Heart Disease Dataset from the UCI Repository.**

## THE DATA

The dataset which I took consist of **303 individuals data** related to heart disease. This dataset consist of 13 different features/attributes which I will be using it to predict whether the individual has the heart disease or not.



| | age | sex | ChestPain | RestBP | Cholestoral | Blood Sugar | ECG | MAXHeartRate | ExerciseAngina | oldpeak | slope | MajorVessels | 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

***<u>Dataset Features:</u>***

1. ***Age***: displays the age of the individual.
2. ***Sex***: displays the gender of the individual using the following format :
   1 = male
   0 = female
3. ***Chest-pain type***: displays the type of chest-pain experienced by the individual using the following format :
   1 = typical angina
   2 = atypical angina
   3 = non — angina pain
   4 = asymptotic
4. ***Resting Blood Pressure***: displays the resting blood pressure value of an individual in mmHg (unit)
5. ***Serum Cholesterol***: displays the serum cholesterol in mg/dl (unit)

6. ***Fasting Blood Sugar***: compares the fasting blood sugar value of an individual with 120mg/dl.
   If fasting blood sugar > 120mg/dl then : 1 (true)
   else : 0 (false)
7. ***Resting ECG*** : displays resting electrocardiographic results
   0 = normal
   1 = having ST-T wave abnormality
   2 = left ventricular hypertrophy
8. ***Max heart rate achieved:*** displays the max heart rate achieved by an individual.
9. ***Exercise induced angina*** :
   1 = yes
   0 = no
10. ***ST depression induced by exercise relative to rest***: displays the value which is an integer or float.
11. ***Peak exercise ST segment*** :
    1 = up sloping
    2 = flat
    3 = down sloping
12. ***Number of major vessels (0–3) colored by fluoroscopy***: displays the value as integer or float.
13. ***Thal*** : displays the thalassemia :
    3 = normal
    6 = fixed defect
    7 = reversible defect

**There is an extra column which is the "Target" variable which displays whether the individual is suffering from heart disease or not: (0 = absence and 1 = present)**

## PROGRAMMING LANGUAGE DESCRIPTION

**Python** is one of the most popular high-level languages used by data scientists and software developers alike for data science tasks. It can be used to predict outcomes, automate tasks, streamline processes, and offer business intelligence insights. There are quite a few open-source libraries that make Python data tasks much easier.

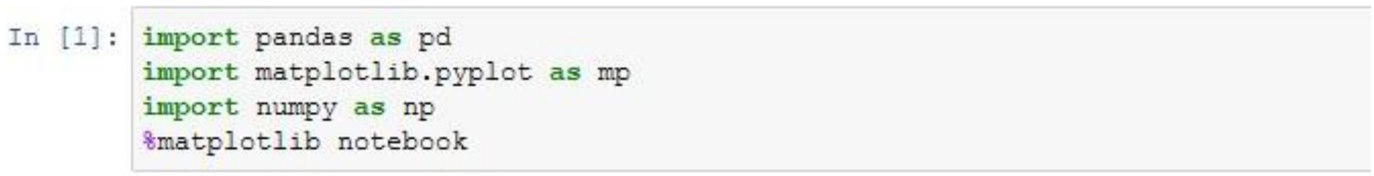
The Python libraries and streamlined processes make data analysis and machine learning to a whole new level. There are so many different libraries in python which helps the user to make their work effective and efficient. The Python libraries are also called as 'Modules'.

For developing the heart disease prediction model, I used Pandas, NumPy, Matplotlib and Scikit-learn modules.

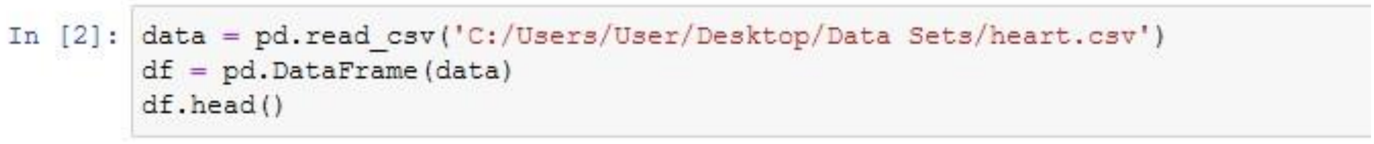
- **Pandas** is an open-source library, specifically developed for data science and analysis. It is built upon the Numpy (to handle numeric data in tabular form) package and has inbuilt data structures to ease up the process of data munging/wrangling, aka data manipulation.
- **NumPy,** which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.
- **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.
- **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.

## DATA ANALYSIS

I used ***Jupyter Notebook*** to do the analysis and develop the ML model. The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Before starting the analysis part we need to import necessary libraries to the notebook. We need to process the data **(Data Pre-processing)** to make it more effective for developing the Machine Learning model. Data pre-processing is an important step in the data mining process. The phrase **"garbage in, garbage out"** is particularly applicable to data mining and machine learning projects.

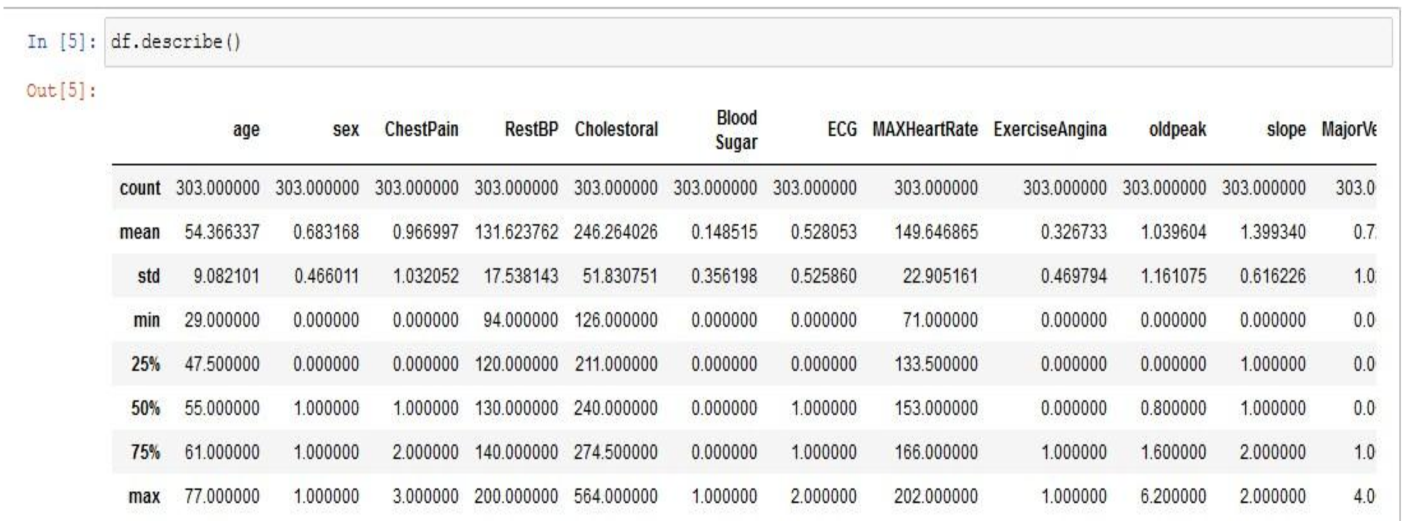


```
In [1]: import pandas as pd
        import matplotlib.pyplot as mp
        import numpy as np
        %matplotlib notebook
```



```
In [2]: data = pd.read_csv('C:/Users/User/Desktop/Data Sets/heart.csv')
        df = pd.DataFrame(data)
        df.head()
```

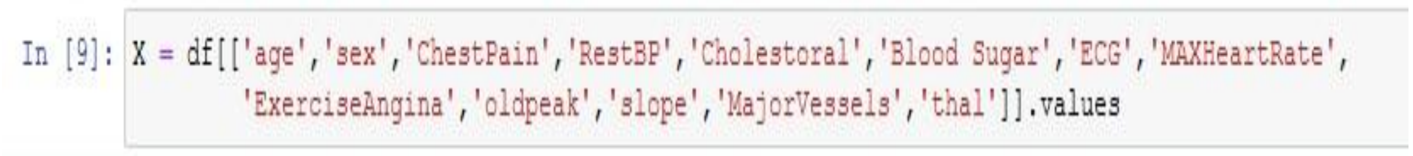
Pandas library is more effective for data manipulation process. Using the **dropna() function**, we can remove/drop the missing values present in the dataset. Using the **describe() function** we can see the statistical information from the dataset.



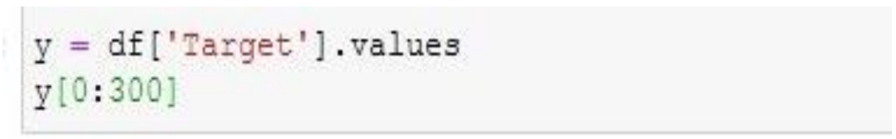
```
In [5]: df.describe()

Out[5]:
              age         sex   ChestPain      RestBP  Cholestoral       Blood         ECG  MAXHeartRate  ExerciseAngina     oldpeak       slope  MajorVe
                                                                         Sugar
count  303.000000  303.000000  303.000000  303.000000   303.000000  303.000000  303.000000    303.000000      303.000000  303.000000  303.000000    303.0
 mean   54.366337    0.683168    0.966997  131.623762   246.264026    0.148515    0.528053    149.646865        0.326733    1.039604    1.399340      0.7
  std    9.082101    0.466011    1.032052   17.538143    51.830751    0.356198    0.525860     22.905161        0.469794    1.161075    0.616226      1.0
  min   29.000000    0.000000    0.000000   94.000000   126.000000    0.000000    0.000000     71.000000        0.000000    0.000000    0.000000      0.0
  25%   47.500000    0.000000    0.000000  120.000000   211.000000    0.000000    0.000000    133.500000        0.000000    0.000000    1.000000      0.0
  50%   55.000000    1.000000    1.000000  130.000000   240.000000    0.000000    1.000000    153.000000        0.000000    0.800000    1.000000      0.0
  75%   61.000000    1.000000    2.000000  140.000000   274.500000    0.000000    1.000000    166.000000        1.000000    1.600000    2.000000      1.0
  max   77.000000    1.000000    3.000000  200.000000   564.000000    1.000000    2.000000    202.000000        1.000000    6.200000    2.000000      4.0
```

**Dependent and independent variables** are variables in Machine Learning modelling. Independent variables are controlled input, which is denoted by 'X'. Dependent variables represent the output or outcome resulting from altering these inputs, which is denoted by 'y'.

- Independent variable (X) – All the features present in the dataset except the 'target' column is my independent variable. These features will be used as the input to the ML algorithm.
- Dependent variables (y) – 'Target' will be the output variable.

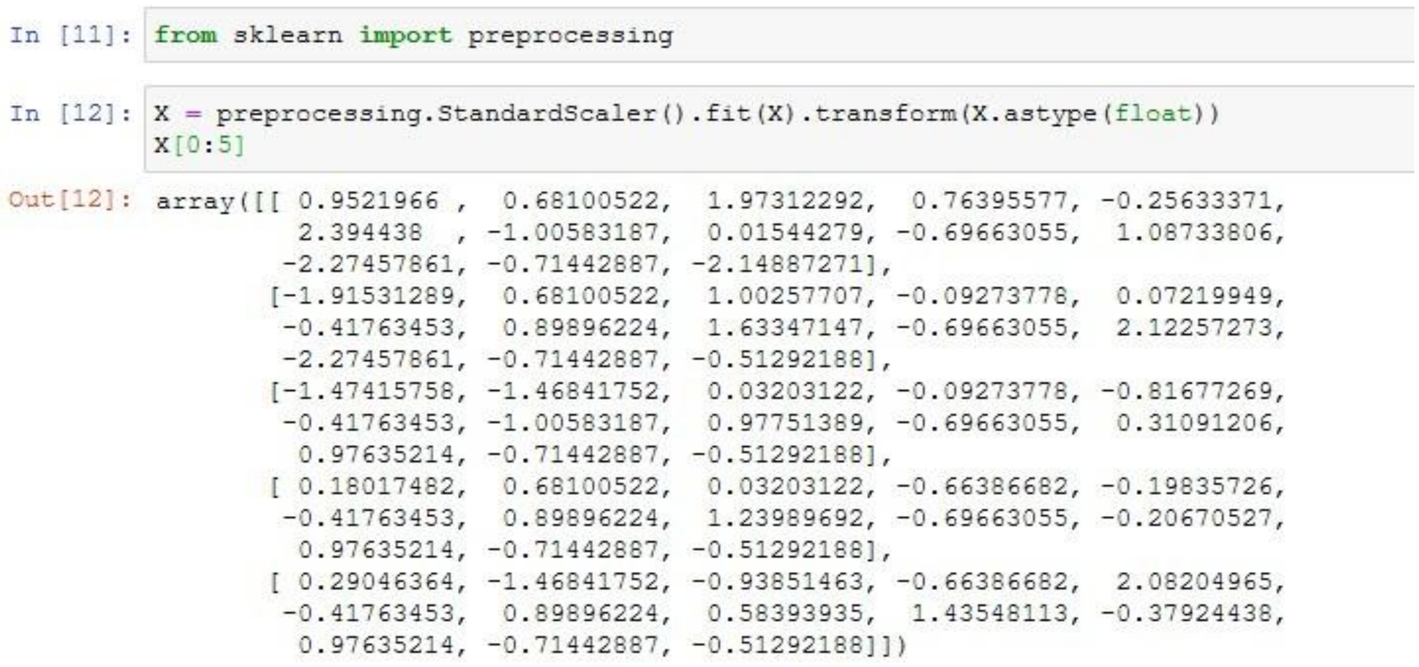


```
In [9]: X = df[['age','sex','ChestPain','RestBP','Cholestoral','Blood Sugar','ECG','MAXHeartRate',
               'ExerciseAngina','oldpeak','slope','MajorVessels','thal']].values
```



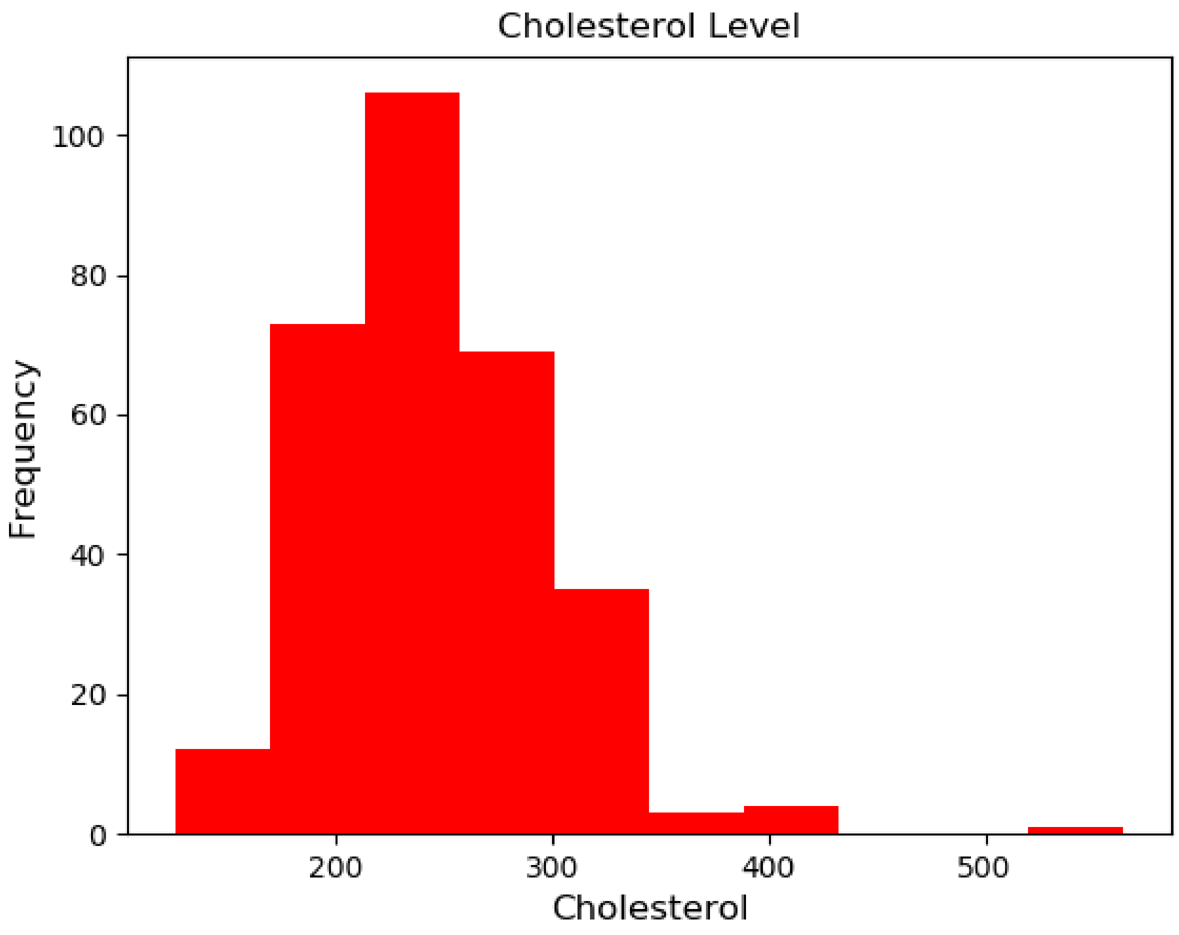
```
y = df['Target'].values
y[0:300]
```

Imported the Scikit-learn module to normalize the independent variable data. **Data Normalization** is the process of structuring a relational data in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity.



```
In [11]: from sklearn import preprocessing

In [12]: X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))
         X[0:5]

Out[12]: array([[ 0.9521966 ,  0.68100522,  1.97312292,  0.76395577, -0.25633371,
                  2.394438  , -1.00583187,  0.01544279, -0.69663055,  1.08733806,
                 -2.27457861, -0.71442887, -2.14887271],
                [-1.91531289,  0.68100522,  1.00257707, -0.09273778,  0.07219949,
                 -0.41763453,  0.89896224,  1.63347147, -0.69663055,  2.12257273,
                 -2.27457861, -0.71442887, -0.51292188],
                [-1.47415758, -1.46841752,  0.03203122, -0.09273778, -0.81677269,
                 -0.41763453, -1.00583187,  0.97751389, -0.69663055,  0.31091206,
                  0.97635214, -0.71442887, -0.51292188],
                [ 0.18017482,  0.68100522,  0.03203122, -0.66386682, -0.19835726,
                 -0.41763453,  0.89896224,  1.23989692, -0.69663055, -0.20670527,
                  0.97635214, -0.71442887, -0.51292188],
                [ 0.29046364, -1.46841752, -0.93851463, -0.66386682,  2.08204965,
                 -0.41763453,  0.89896224,  0.58393935,  1.43548113, -0.37924438,
                  0.97635214, -0.71442887, -0.51292188]])
```

To perform **Data Visualization**, I used the Matplotlib library to plot the histogram and visualize the Cholesterol Level of the individuals.



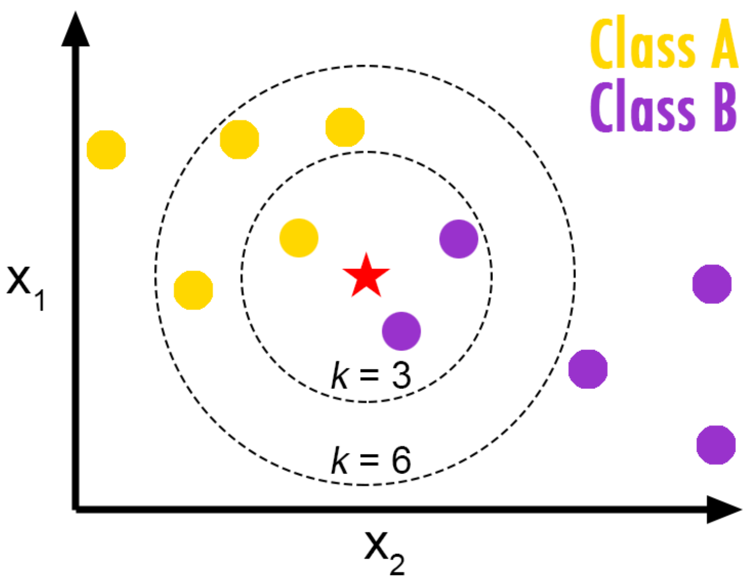
## ALGORITHM (K-NEAREST NEIGHBOR)

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

A supervised machine learning algorithm (as opposed to an unsupervised machine learning algorithm) is one that relies on labeled input data to learn a function that produces an appropriate output when given new unlabeled data.

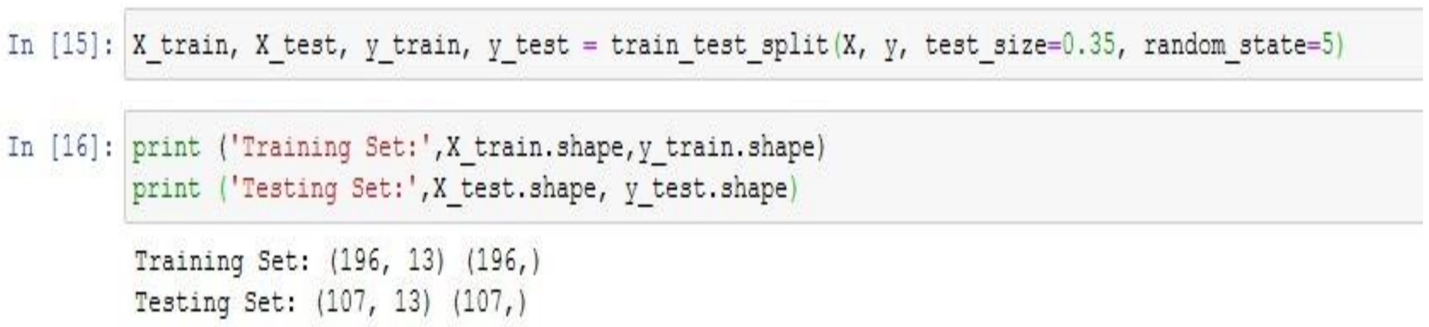
A classification problem has a discrete value as its output. Here, the dependent variable 'Target' is the output. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. To find the distance we use the **Euclidean-distance** formula to calculate the similarity/distance.

Here, the 'k' in k-nearest neighbor is the number of neighbors that our model take into consideration to predict the outcome.



## Train-Test Split

Using the train_test_split function, split the dataset into training and testing set. Training set will be used to train the ML model, whereas the testing set will be used to check how well our model performs in unseen data or also known as real-world data.



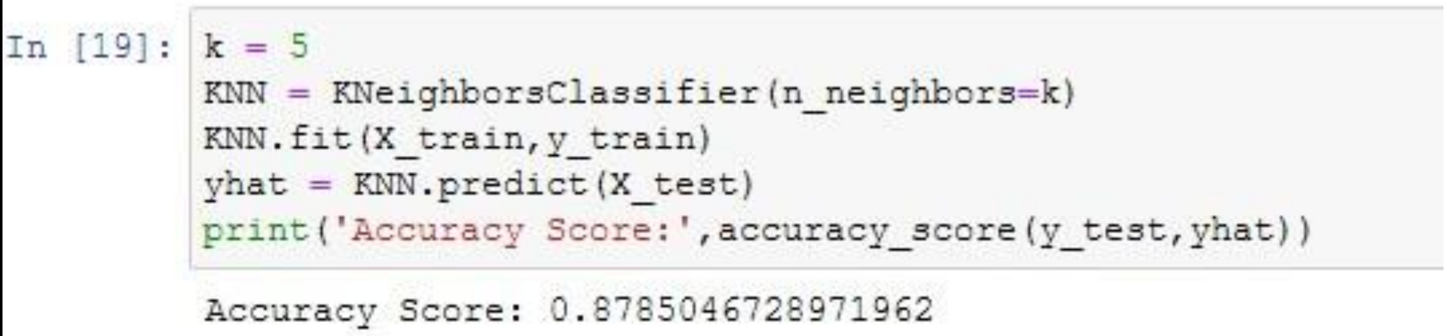
```
In [15]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, random_state=5)
```

```
In [16]: print ('Training Set:',X_train.shape,y_train.shape)
         print ('Testing Set:',X_test.shape, y_test.shape)
```

```
Training Set: (196, 13) (196,)
Testing Set: (107, 13) (107,)
```

**The KNN Algorithm**

- Initialize K to your chosen number of neighbors.
- When the model gets an input real-world data as an query, it will predict whether the individual has the heart disease or not by calculating the distance between this new unseen data and all other records in which our model is trained.
- Add the distance and the index of the example to an ordered collection.
- Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances.
- Pick the first K entries from the sorted collection.
- Get the labels of the selected K entries.
- If regression, return the mean of the K labels.
- If classification, return the mode of the K labels.
- Calculate the model accuracy score, to check how well the model performs in the real world.



```
In [19]: k = 5
         KNN = KNeighborsClassifier(n_neighbors=k)
         KNN.fit(X_train,y_train)
         yhat = KNN.predict(X_test)
         print('Accuracy Score:',accuracy_score(y_test,yhat))

         Accuracy Score: 0.8785046728971962
```

## RESULT

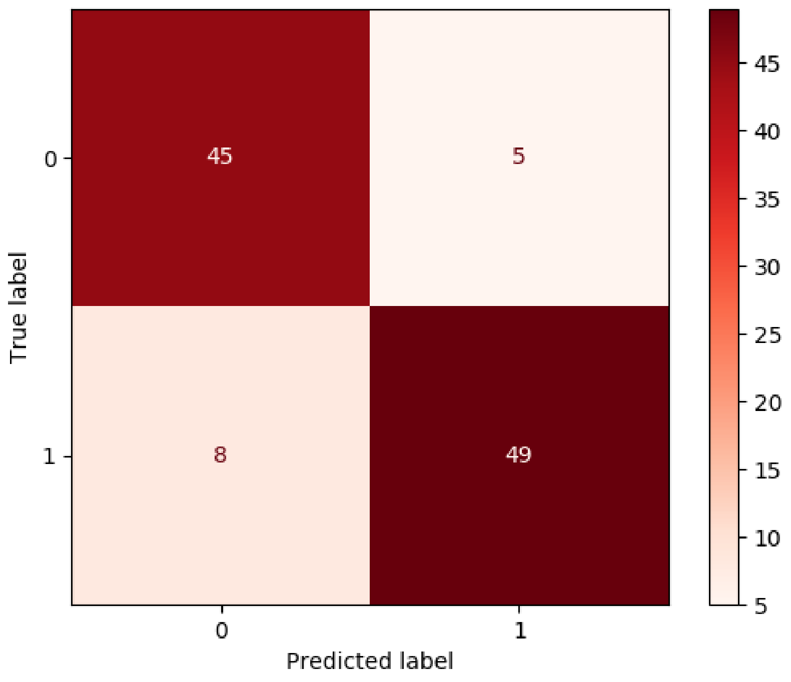
The KNN algorithm successfully predicts the classification problem with **accuracy score of 87.85.** To evaluate the model further, I performed the model evaluation process using the confusion matrix and calculated the F1-Score.

## Model Evaluation

Confusion matrix is the most preferred model evaluation metric for any classification algorithm. The Confusion matrix will show the true positives, true negatives, false positives and false negatives. It is also extremely useful for measuring **Recall, Precision and F1-Score.**

- **True Positive:** You predicted positive and it's true.
- **True Negative:** You predicted negative and it's true.
- **False Positive (Type 1 Error):** You predicted positive and it's false.
- **False Negative (Type 2 Error):** You predicted negative and it's false.

```
In [20]: from sklearn.metrics import classification_report

In [21]: print(classification_report(y_test,yhat))
                  precision    recall  f1-score   support

               0       0.85      0.90      0.87        50
               1       0.91      0.86      0.88        57

        accuracy                           0.88       107
       macro avg       0.88      0.88      0.88       107
    weighted avg       0.88      0.88      0.88       107


In [22]: from sklearn.metrics import plot_confusion_matrix

In [23]: disp = plot_confusion_matrix(KNN, X_test, y_test, cmap='Reds')
```



## CONCLUSION

Heart Disease is one of the major concerns for society today. It is difficult to manually determine the odds of getting heart disease based on risk factors. However, machine learning techniques are useful to predict the output from existing data.