To perform and find the accuracy of K means algorithm

In [ ]:
```
Name : Siddhi N. sakharkar
Roll No. : 51
Section : B
```

In [23]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # for data visualization
import seaborn as sns # for statistical data visualization
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
os.getcwd()
```

Out[4]:
```
'C:\\Users\\lenovo'
```

In [5]:
```
os.chdir('C:\\Users\\lenovo\\Desktop')
```

In [6]:
```
df=pd.read_csv('CHD_preprocessed.csv')
```

In [7]:
```
df.head()
```

Out[7]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRat
0	1	39	1	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.
1	0	46	0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.
2	1	48	0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.
3	0	61	1	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.
4	0	46	1	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4133 entries, 0 to 4132
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             4133 non-null   int64  
 1   age              4133 non-null   int64  
 2   education        4133 non-null   int64  
 3   currentSmoker    4133 non-null   int64  
 4   cigsPerDay       4133 non-null   float64
 5   BPMeds           4133 non-null   float64
 6   prevalentStroke  4133 non-null   int64  
 7   prevalentHyp     4133 non-null   int64  
 8   diabetes         4133 non-null   int64  
 9   totChol          4133 non-null   float64
 10  sysBP            4133 non-null   float64
 11  diaBP            4133 non-null   float64
 12  BMI              4133 non-null   float64
 13  heartRate        4133 non-null   float64
 14  glucose          4133 non-null   float64
 15  TenYearCHD       4133 non-null   int64  
dtypes: float64(8), int64(8)
memory usage: 516.8 KB
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4133 entries, 0 to 4132
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
```

```
 0   male             4133 non-null   int64
 1   age              4133 non-null   int64
 2   education        4133 non-null   int64
 3   currentSmoker    4133 non-null   int64
 4   cigsPerDay       4133 non-null   float64
 5   BPMeds           4133 non-null   float64
 6   prevalentStroke  4133 non-null   int64
 7   prevalentHyp     4133 non-null   int64
 8   diabetes         4133 non-null   int64
 9   totChol          4133 non-null   float64
 10  sysBP            4133 non-null   float64
 11  diaBP            4133 non-null   float64
 12  BMI              4133 non-null   float64
 13  heartRate        4133 non-null   float64
 14  glucose          4133 non-null   float64
 15  TenYearCHD       4133 non-null   int64
dtypes: float64(8), int64(8)
memory usage: 516.8 KB
```

In [10]:
```
df.size
```

Out[10]:
```
66128
```

In [12]:
```
df.shape
```

Out[12]:
```
(4133, 16)
```

In [13]:
```
df.describe()
```

Out[13]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totCh
count	4133.000000	4133.000000	4133.000000	4133.000000	4133.000000	4133.000000	4133.000000	4133.000000	4133.000000	4133.00000
mean	0.427293	49.557222	0.280668	0.494798	9.101621	0.034358	0.006049	0.311154	0.025647	236.6644(
std	0.494745	8.561628	0.449380	0.500033	11.918440	0.182168	0.077548	0.463022	0.158100	43.90918
min	0.000000	32.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	107.00000
25%	0.000000	42.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	206.00000
50%	0.000000	49.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	234.00000
75%	1.000000	56.000000	1.000000	1.000000	20.000000	0.000000	0.000000	1.000000	0.000000	262.00000
max	1.000000	70.000000	1.000000	1.000000	70.000000	1.000000	1.000000	1.000000	1.000000	600.00000

In [16]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import adjusted_rand_score
```

In [26]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=0)

kmeans.fit(X)
```

Out[26]:
```
KMeans(n_clusters=2, random_state=0)
```

In [27]:
```
kmeans.cluster_centers_
```

Out[27]:
```
array([[3.91715976e-01, 5.22526627e+01, 2.79881657e-01, 4.64497041e-01,
        8.65029586e+00, 5.20710059e-02, 7.69230769e-03, 4.27810651e-01,
        3.07692308e-02, 2.77700592e+02, 1.39265385e+02, 8.59387574e+01,
        2.63765816e+01, 7.74721893e+01, 8.30378698e+01, 1.88757396e-01],
       [4.51903397e-01, 4.76925911e+01, 2.81211625e-01, 5.15759312e-01,
        9.41383545e+00, 2.21039705e-02, 4.91199345e-03, 2.30454359e-01,
        2.21039705e-02, 2.08276709e+02, 1.27594965e+02, 8.07509210e+01,
        2.53648839e+01, 7.48550962e+01, 8.11915677e+01, 1.26483831e-01]])
```

In [28]:
```
kmeans.inertia_
```

Out[28]: 9283500.938229598

In [ ]: