In [3]:
```
# IMPORT LIBRARIES

import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

In [4]:
```
# IMPORT DATASET
```

In [5]:
```
data = pd.read_csv('heart.csv')
data
```

Out[5]:

	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

In [6]:
```
data.head()
```

Out[6]:

	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 [7]:
```
# Total no's of rows and columns
```

In [8]:
```
data.shape
```

Out[8]:
```
(303, 14)
```

In [9]:
```
# Data Describe
```

In [10]:
```
data.describe()
```

Out[10]:

	age	sex	cp	trestbps	chol	fbs	restecg	tha
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.64
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.90
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.00
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.50
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.00
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.00
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.00

In [11]:
```
#
```

In [12]:
```
data['target'].value_counts()
```

Out[12]:
```
1    165
0    138
Name: target, dtype: int64
```

In [13]:
```
# mean of target value
```

In [14]:
```
data.groupby('target').mean()
```

Out[14]:

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	e
0	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449	0.5
1	52.496970	0.563636	1.375758	129.303030	242.230303	0.139394	0.593939	158.466667	0.1

In [15]:
```
# Drop target columns
```

In [16]:
```
X = data.drop(columns='target',axis=1)
Y = data['target']
```

In [17]:
```
print(X) # Feature values
```

```
     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
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
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
0        0   0     1
1        0   0     2
2        2   0     2
3        2   0     2
4        2   0     2
..     ...  ..   ...
298      1   0     3
299      1   0     3
300      1   2     3
301      1   1     3
302      1   1     2

[303 rows x 13 columns]
```

In [18]:
```
print(Y) # target value
```

```
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 [19]:
```
scaler = StandardScaler()
```

In [20]:
```
scaler.fit(X)
```

Out[20]:
```
StandardScaler()
```

In [21]:
```
# Standardization
```

In [22]:
```
standardized_data = scaler.transform(X)
```

In [23]:
```
print(standardized_data)
```

```
[[ 0.9521966   0.68100522  1.97312292 ... -2.27457861 -0.71442887
  -2.14887271]
 [-1.91531289  0.68100522  1.00257707 ... -2.27457861 -0.71442887
  -0.51292188]
 [-1.47415758 -1.46841752  0.03203122 ...  0.97635214 -0.71442887
  -0.51292188]
 ...
 [ 1.50364073  0.68100522 -0.93851463 ... -0.64911323  1.24459328
   1.12302895]
 [ 0.29046364  0.68100522 -0.93851463 ... -0.64911323  0.26508221
   1.12302895]
 [ 0.29046364 -1.46841752  0.03203122 ... -0.64911323  0.26508221
  -0.51292188]]
```

In [24]:
```
X = standardized_data
Y = data['target']
```

In [25]:
```
print(X)
print(Y)
```

```
[[ 0.9521966   0.68100522  1.97312292 ... -2.27457861 -0.71442887
  -2.14887271]
 [-1.91531289  0.68100522  1.00257707 ... -2.27457861 -0.71442887
  -0.51292188]
 [-1.47415758 -1.46841752  0.03203122 ...  0.97635214 -0.71442887
  -0.51292188]
 ...
 [ 1.50364073  0.68100522 -0.93851463 ... -0.64911323  1.24459328
   1.12302895]
 [ 0.29046364  0.68100522 -0.93851463 ... -0.64911323  0.26508221
   1.12302895]
 [ 0.29046364 -1.46841752  0.03203122 ... -0.64911323  0.26508221
  -0.51292188]]
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 [26]:
```
# Split the Dataset
```

In [27]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratif
print(X.shape, X_train.shape, X_test.shape)
```

```
(303, 13) (242, 13) (61, 13)
```

In [28]:
```
 # Build Decision Tree Classifier
```

In [29]:
```
from sklearn import tree
```

In [30]:
```
dt_clf = tree.DecisionTreeClassifier()
```

In [31]:
```
dt_clf.fit(X_train, Y_train)
dt_clf.score(X_test,Y_test)
```

Out[31]:
```
0.7704918032786885
```

In [32]:
```
# Accouracy score of DT
```

In [33]:
```
y_pred = dt_clf.predict(X_test)
dt_clf.score(X_test,Y_test)
```

Out[33]:
```
0.7704918032786885
```

In [34]:
```
# Build RandomForest Classifier
# Accuracy_score
```

In [35]:
```
from sklearn import ensemble
rf_clf = ensemble.RandomForestClassifier(n_estimators=5)
rf_clf.fit(X_train,Y_train)
rf_clf.score(X_test,Y_test)
```

Out[35]:
```
0.7540983606557377
```

In [36]:
```
# Build Naive_Bayes
```

In [37]:
```
from sklearn.naive_bayes import GaussianNB
```

In [38]:
```
nb_clf = GaussianNB()
```

In [39]:
```
# Accuracy_score
```

In [40]:
```
nb_clf.fit(X_train,Y_train)
nb_clf.score(X_test,Y_test)
```

Out[40]:
```
0.819672131147541
```

In [41]:
```
# Build a KNN Classifier
```

In [42]:
```
from sklearn.neighbors import KNeighborsClassifier
kn_clf = KNeighborsClassifier()
```

In [43]:
```
# Accuracy_score
```

In [44]:
```
kn_clf.fit(X_train,Y_train)
kn_clf.score(X_test,Y_test)
```

Out[44]: 0.819672131147541

In [45]:
```
# Build a Logistic Regression Classifier
```

In [46]:
```
from sklearn.linear_model import LogisticRegression
lr_clf = LogisticRegression()
```

In [47]:
```
# Accuracy Score
```

In [48]:
```
lr_clf.fit(X_train,Y_train)
lr_clf.score(X_test,Y_test)
```

Out[48]: 0.7868852459016393

In [49]:
```
# Build an SVM Classifier
```

In [50]:
```
from sklearn.svm import SVC
```

In [51]:
```
sv_clf = svm.SVC(kernel = 'linear')
sv_clf.fit(X_train,Y_train)
sv_clf.score(X_test,Y_test)
```

Out[51]: 0.819672131147541

In [ ]:
```

```

In [ ]:
```

```