Importing Datasets and Libraries

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data Collection and Processing

In [2]:
```
# loading the csv data to a Pandas DataFrame
heartDisease_data = pd.read_csv(r"C:\Users\user\Desktop\Machine learning pratice\heart disease prediction project\Heart Disease P
```

In [3]:
```
# print first 5 rows of the dataset
heartDisease_data.head()
```

Out[3]:

	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 [4]:
```
# print last 5 rows of the dataset
heartDisease_data.tail()
```

In [7]:
```
# checking for missing values
heartDisease_data.isnull().sum()
```

Out[7]:
```
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
```

Data Exploration

In [8]:
```
# statistical measures about the data
heartDisease_data.describe()
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
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
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
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
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
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
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
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
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

```
     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 [12]:
```
print(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
```

```
acc_knn = round(knn.score(X_train, Y_train)*100, 2)
print(str(acc_knn)+' Percentage')
```

```
79.34 Percentage
```

4. Decision Tree Classifier

In [18]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, Y_train)
acc_dt = round(dt.score(X_train, Y_train)*100, 2)
print(str(acc_dt)+' Percentage')
```

```
100.0 Percentage
```

5. Random Forest Classifier

In [19]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, Y_train)
acc_rf = round(rf.score(X_train, Y_train)*100, 2)
print(str(acc_rf)+' Percentage')
```

```
100.0 Percentage
```

6. Naive Bayes

In [20]:
```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, Y_train)
acc_nb = round(nb.score(X_train, Y_train)*100, 2)
print(str(acc_nb)+' Percentage')
```

```
84.71 Percentage
```

Comparing Modela

Let's compare the accuracy score of all the models used above

In [23]:
```
test_lr = round(accuracy_score(lr_pred, Y_test)*100,2)
test_svm = round(accuracy_score(svm_pred, Y_test)*100,2)
test_knn = round(accuracy_score(knn_pred, Y_test)*100,2)
test_dt = round(accuracy_score(dt_pred, Y_test)*100,2)
test_rf = round(accuracy_score(rf_pred, Y_test)*100,2)
test_nb = round(accuracy_score(nb_pred, Y_test)*100,2)

test_models = pd.DataFrame({
    'Models':['Logistic Regression', 'Support Vector', 'KNN', 'Decision Tree', 'Random Forest', 'Naive Bayes'],
    'Score(Test Data)':[test_lr, test_svm, test_knn, test_dt, test_rf, test_nb]
})

test_models.sort_values(by='Score(Test Data)', ascending=False)
```

Out[23]:

	Models	Score(Test Data)
0	Logistic Regression	80.33
4	Random Forest	80.33
5	Naive Bayes	80.33
3	Decision Tree	75.41
1	Support Vector	62.30
2	KNN	62.30

From Above two tables, we can see that Logistic Regression has close score on both test and train scores. So will evaluate our model on Logistic Regression.

Building a Predictive System

In [26]:
```
input_data = (56,1,1,120,236,0,1,178,0,0.8,2,0,2)

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
```