In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.read_csv('heart-disease.csv')
```

In [49]:
```
df.head()
```

Out[49]:

	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 [3]:
```
df.shape
```

Out[3]:
```
(303, 14)
```

In [4]:
```
df.isna().sum()
```

Out[4]:
```
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
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
```

```
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [23]:
```
df.head()
```

Out[23]:

	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 [77]:
```
df.tail()
```

Out[77]:

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

In [24]:
```
df.describe()
```

Out[24]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach
count	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
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000

	age	sex	cp	trestbps	chol	fbs	restecg	thalach
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000

In [7]:
```
X = df.drop(columns='target',axis=1)
y = df['target']
```

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

Out[8]:
```
(303, 13)
```

In [9]:
```
y.shape
```

Out[9]:
```
(303,)
```

In [10]:
```
df['target'].value_counts()
```

Out[10]:
```
1    165
0    138
Name: target, dtype: int64
```

In [11]:
```
from sklearn.model_selection import train_test_split
```

In [12]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [13]:
```
X_train.shape
```

Out[13]:
```
(242, 13)
```

In [14]:
```
X_test.shape
```

Out[14]:
```
(61, 13)
```

In [15]:
```
y_train.shape
```

Out[15]:
```
(242,)
```

In [16]:
```
y_test.shape
```

Out[16]:
```
(61,)
```

Create a model

In [50]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

In [51]:
```
models = {'Logistic Reg':LogisticRegression(),
          'Linear Reg':LogisticRegression(),
          'KNN': KNeighborsClassifier(n_neighbors=5,algorithm='auto',leaf_size=30,metri
          'SVM': SVC(kernel='rbf',random_state=15,C=1.0),
          'RF': RandomForestClassifier(n_estimators=150,max_depth = 9,min_samples_leaf
         'Ada': AdaBoostClassifier(base_estimator=LogisticRegression(),n_estimators=100
```

In [52]:
```
for name, model in models.items():
    model.fit(X_train,y_train)
    print(name, model.score(X_test,y_test))
```

```
C:\Users\Life Computer\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Life Computer\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Logistic Reg 0.9016393442622951
Linear Reg 0.9016393442622951
KNN 0.819672131147541
SVM 0.6721311475409836
RF 0.9016393442622951
C:\Users\Life Computer\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Life Computer\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Life Computer\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Life Computer\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Ada 0.8852459016393442
```

In [65]:
```
rf_model = RandomForestClassifier(random_state=2)
rf_model.fit(X_train,y_train)
rf_model.score(X_train,y_train), model.score(X_test,y_test)
```

Out[65]:
```
(1.0, 0.8688524590163934)
```

In [61]:
```
lo_model = LogisticRegression(random_state=2)
model.fit(X_train,y_train)
model.score(X_train,y_train), model.score(X_test,y_test)
```

Out[61]:
```
(1.0, 0.9016393442622951)
```

In [62]:
```
ada_model = AdaBoostClassifier(random_state=2)
model.fit(X_train,y_train)
model.score(X_train,y_train), model.score(X_test,y_test)
```

Out[62]:
```
(1.0, 0.8852459016393442)
```

In [63]:
```
KNN_model = KNeighborsClassifier(n_neighbors=5,algorithm='auto',leaf_size=30,metric='mi
model.fit(X_train,y_train)
model.score(X_train,y_train), model.score(X_test,y_test)
```

Out[63]:
```
(1.0, 0.8688524590163934)
```

Final model select as Random Forest

In [66]:
```
rf_model = RandomForestClassifier()
```

```
rf_model.fit(X_train,y_train)
rf_model.score(X_train,y_train), rf_model.score(X_test,y_test)
```

Out[66]:
```
(1.0, 0.9016393442622951)
```

In [71]:
```
X_train_pred =rf_model.predict(X_train)
traing_data_accuracy = accuracy_score(X_train_pred,y_train)
```

In [72]:
```
traing_data_accuracy
```

Out[72]:
```
1.0
```

In [73]:
```
X_test_pred =rf_model.predict(X_test)
testing_data_accuracy = accuracy_score(X_test_pred,y_test)
```

In [74]:
```
testing_data_accuracy
```

Out[74]:
```
0.9016393442622951
```

In [75]:
```
input_data = (70,1,2,160,269,0,1,112,1,2.9,1,1,3)

num_array = np.asarray(input_data)

num_array_reshape = num_array.reshape(1,-1)

prediction = model.predict(num_array_reshape)

print(prediction)
```

```
[0]
```

In [76]:
```
import pickle

pickle.dump(rf_model , open('model1.pkl','wb'))

heart_disease_predict_model = pickle.load(open('model1.pkl','rb'))
```

In [ ]:
```

```