In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries inst
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will l

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that
# You can also write temporary files to /kaggle/temp/, but they won't be sav
```

```
/kaggle/input/heart-disease-uci/heart.csv
```

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

In [3]:
```
df.head()
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	ta
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	

In [4]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(df.iloc[:,0:-1],df.iloc[:,-
```

In [5]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

In [6]:
```
clf1 = LogisticRegression()
clf2 = DecisionTreeClassifier()
```

In [7]:
```
clf1.fit(X_train,y_train)
clf2.fit(X_train,y_train)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:765:
```

```
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-regres
sion
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[7]:
```
DecisionTreeClassifier()
```

In [8]:
```
y_pred1 = clf1.predict(X_test)
y_pred2 = clf2.predict(X_test)
```

In [9]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
print("Accuracy of Logistic Regression",accuracy_score(y_test,y_pred1))
print("Accuracy of Decision Trees",accuracy_score(y_test,y_pred2))
```

```
Accuracy of Logistic Regression 0.9016393442622951
Accuracy of Decision Trees 0.8360655737704918
```

In [10]:
```
confusion_matrix(y_test,y_pred1)
```

Out[10]:
```
array([[26,  6],
       [ 0, 29]])
```

In [11]:
```
print("Logistic Regression Confusion Matrix\n")
pd.DataFrame(confusion_matrix(y_test,y_pred1),columns=list(range(0,2)))
```

```
Logistic Regression Confusion Matrix
```

Out[11]:

	0	1
0	26	6
1	0	29

In [12]:
```
print("Decision Tree Confusion Matrix\n")
pd.DataFrame(confusion_matrix(y_test,y_pred2),columns=list(range(0,2)))
```

```
Decision Tree Confusion Matrix
```

Out[12]:

	0	1
0	24	8
1	2	27

In [13]:
```
result = pd.DataFrame()
result['Actual Label'] = y_test
result['Logistic Regression Prediction'] = y_pred1
result['Decision Tree Prediction'] = y_pred2
```

In [14]:
```
result.sample(10)
```

Out[14]:

	Actual Label	Logistic Regression Prediction	Decision Tree Prediction
257	0	0	0
65	1	1	1
164	1	1	1
29	1	1	1
296	0	1	0
184	0	0	0
161	1	1	1
292	0	0	0
53	1	1	1
147	1	1	1

In [15]:
```
from sklearn.metrics import recall_score,precision_score,f1_score
```

In [16]:
```
print("For Logistic regression Model")
print("-"*50)
cdf = pd.DataFrame(confusion_matrix(y_test,y_pred1),columns=list(range(0,2))
print(cdf)
print("-"*50)
print("Precision - ",precision_score(y_test,y_pred1))
print("Recall - ",recall_score(y_test,y_pred1))
print("F1 score - ",f1_score(y_test,y_pred1))
```

```
For Logistic regression Model
--------------------------------------------------
    0   1
0  26   6
1   0  29
--------------------------------------------------
Precision -  0.8285714285714286
Recall -  1.0
F1 score -  0.90625
```

In [17]:
```
print("For DT Model")
print("-"*50)
cdf = pd.DataFrame(confusion_matrix(y_test,y_pred2),columns=list(range(0,2))
print(cdf)
print("-"*50)
print("Precision - ",precision_score(y_test,y_pred2))
print("Recall - ",recall_score(y_test,y_pred2))
print("F1 score - ",f1_score(y_test,y_pred2))
```

```
For DT Model
--------------------------------------------------
    0   1
0  24   8
```

```
0  24   8
1   2  27
--------------------------------------------------
Precision -  0.7714285714285715
Recall -  0.9310344827586207
F1 score -  0.8437500000000001
```

In [18]:
```
precision_score(y_test,y_pred1,average=None)
```

Out[18]:
```
array([1.        , 0.82857143])
```

In [19]:
```
precision_score(y_test,y_pred2,average=None)
```

Out[19]:
```
array([0.92307692, 0.77142857])
```

In [21]:
```
recall_score(y_test,y_pred2,average=None)
```

Out[21]:
```
array([0.75      , 0.93103448])
```