Prediction of Heart Disease Using Machine Learning Algorithms

In this project, I have looked into the heart disease dataset from that dataset we can derive various insights that help us know the weightage of each feature and how they are interrelated to each other but this time my sole aim is to detect whether a person is affected by a heart problem or not. Here, i will show how Machine learning Algorithm (KNN) is used for Heart Disease prediction.

Import python libraries

```
import numpy as numpy
import pandas as pd
```

Create the Data Frame

```
heart=pd.read_csv('/content/heart.csv',sep=",")
heart
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	52	1	0	125	212	0	1	168	0	1.0	2	2
1	53	1	0	140	203	1	0	155	1	3.1	0	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1
4	62	0	0	138	294	1	1	106	0	1.9	1	3
...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0
1021	60	1	0	125	258	0	0	141	1	2.8	1	1
1022	47	1	0	110	275	0	0	118	1	1.0	1	1
1023	50	0	0	110	254	0	0	159	0	0.0	2	0
1024	54	1	0	120	188	0	1	113	0	1.4	1	1

1025 rows × 14 columns

Exploratory Data Analysis

```
heart.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
heart.size
```

```
14350
```

```
heart.shape
```

```
(1025, 14)
```

```
heart.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	52	1	0	125	212	0	1	168	0	1.0	2	2
1	53	1	0	140	203	1	0	155	1	3.1	0	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1
4	62	0	0	138	294	1	1	106	0	1.9	1	3

```
heart.tail()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	c
1020	59	1	1	140	221	0	1	164	1	0.0	2	
1021	60	1	0	125	258	0	0	141	1	2.8	1	
1022	47	1	0	110	275	0	0	118	1	1.0	1	
1023	50	0	0	110	254	0	0	159	0	0.0	2	
1024	54	1	0	120	188	0	1	113	0	1.4	1	

```
# missing values
heart.isna().sum()
```

```
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
```

Seperating X and Y from the Data Frame

```
# seperate x
x=heart.iloc[:,:-1].values
x
```

```
array([[52.,  1.,  0., ...,  2.,  2.,  3.],
       [53.,  1.,  0., ...,  0.,  0.,  3.],
       [70.,  1.,  0., ...,  0.,  0.,  3.],
       ...,
       [47.,  1.,  0., ...,  1.,  1.,  2.],
       [50.,  0.,  0., ...,  2.,  0.,  2.],
       [54.,  1.,  0., ...,  1.,  1.,  3.]])
```

```
#seperate y
y=heart.iloc[:,-1].values
y
```

```
array([0, 0, 0, ..., 0, 1, 0])
```

Training and Testing Data

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
y_test
```

```
array([0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1,
       1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1])
```

Normalization of x_train & x_test

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
x_test
```

```
array([[-1.70591538,  0.65878676, -0.92638308, ..., -0.66566659,
        -0.74267123,  1.09139253],
       [-1.70591538, -1.51794185,  1.01293566, ..., -0.66566659,
        -0.74267123, -0.5056043 ],
       [-0.27803847,  0.65878676,  0.04327629, ...,  0.98013666,
        -0.74267123, -0.5056043 ],
       ...,
       [-1.48624201,  0.65878676,  0.04327629, ...,  0.98013666,
        -0.74267123, -0.5056043 ],
       [-0.0583651 ,  0.65878676, -0.92638308, ..., -0.66566659,
         0.20146932, -0.5056043 ],
       [ 0.16130827,  0.65878676,  0.04327629, ...,  0.98013666,
        -0.74267123,  1.09139253]])
```

Implimentation of knn algorithm

```
from sklearn.neighbors import KNeighborsClassifier
heart_knn=KNeighborsClassifier(n_neighbors=5)
heart_knn.fit(x_train,y_train)
y_pred=heart_knn.predict(x_test)
y_pred
```

```
array([0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0,
       1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1,
       1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0])
```

Performance evaluvation

Confusion Matrix

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
result=confusion_matrix(y_test,y_pred)
result
```

```
array([[118,  37],
       [ 21, 132]])
```

Accuracy score

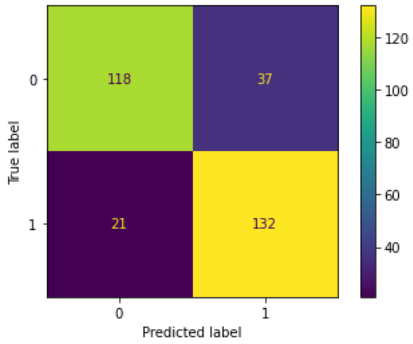
```
score=accuracy_score(y_test,y_pred)
score
```

```
0.8116883116883117
```

Display Confusion Matrix

```
from sklearn.metrics import ConfusionMatrixDisplay
label=['0','1']
cmd=ConfusionMatrixDisplay(result,display_labels=label)
cmd.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f3b491feee0>
```



✓ 0s completed at 11:37 AM