Name : K.Nitheesh

RollNo : 21X05A6728

To predict the Heart Attack Disease for Organization WHO (World Health Organization) , using ml rate of heart attack where increasing or decreasing manner

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data=pd.read_csv("framingham.csv")
data
```

```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
0        1   39          4              0           0       0
1        0   46          2              0           0       0
2        1   48          1              1          20       0
3        0   61          3              1          30       0
4        0   46          3              1          23       0
...    ...  ...        ...            ...         ...     ...
4233     1   50          1              1           1       0
4234     1   51          3              1          43       0
4235     0   48          2              1          20       0
4236     0   44          1              1          15       0
4237     0   52          2              0           0       0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP
BMI  \
0                   0             0         0      195  106.0   70.0
26.97
1                   0             0         0      250  121.0   81.0
28.73
2                   0             0         0      245  127.5   80.0
25.34
3                   0             1         0      225  150.0   95.0
28.58
4                   0             0         0      285  130.0   84.0
23.10
...               ...           ...       ...      ...    ...    ...
...
4233                0             1         0      313  179.0   92.0
25.97
4234                0             0         0      207  126.5   80.0
19.71
4235                0             0         0      248  131.0   72.0
22.00
4236                0             0         0      210  126.5   87.0
19.16
4237                0             0         0      269  133.5   83.0
```

```
21.47

      heartRate  glucose  TenYearCHD
0            80       77           0
1            95       76           0
2            75       70           0
3            65      103           1
4            85       85           0
...         ...      ...         ...
4233         66       86           1
4234         65       68           0
4235         84       86           0
4236         86        0           0
4237         80      107           0

[4238 rows x 16 columns]
```

```
X= data [["age"]]              #input variable : age
y= data ["currentSmoker"]      #predicted : currentsmoker

from sklearn.model_selection import train_test_split
X_train ,X_test ,y_train,y_test = train_test_split(X,y , test_size=0.4
, random_state=0 )

print(X_train)
```

```
      age
3218   42
590    60
3880   41
1548   59
2601   55
...   ...
1033   44
3264   51
1653   39
2607   57
2732   40

[2542 rows x 1 columns]
```

```
print(X_test)
```

```
      age
1669   47
156    58
87     61
685    45
666    57
...   ...
2790   53
```

```
1855   66
700    60
2060   38
2348   48

[1696 rows x 1 columns]
```

```
print(y_train)
```

```
3218    1
590     1
3880    0
1548    0
2601    1
       ..
1033    0
3264    1
1653    1
2607    0
2732    1
Name: currentSmoker, Length: 2542, dtype: int64
```

```
print(y_test)
```

```
1669    0
156     0
87      1
685     0
666     0
       ..
2790    0
1855    0
700     0
2060    0
2348    1
Name: currentSmoker, Length: 1696, dtype: int64
```

Feature Scalling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
print(X_train)
```

```
[[-0.89361628]
 [ 1.21446304]
 [-1.0107318 ]
 ...
```

```
 [-1.24496283]
 [ 0.86311649]
 [-1.12784731]]
```

```
print(X_test)
```

```
[[-0.30803869]
 [ 0.980232  ]
 [ 1.33157856]
 ...
 [ 1.21446304]
 [-1.36207835]
 [-0.19092317]]
```

Training the Logestic Regression on Testing data

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
LogisticRegression(random_state=0)
```

Predict a New Result

```
y_pred = classifier.predict(X_test)
```

Making the confusion Matrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[503 371]
 [303 519]]
```

```
0.6025943396226415
```