```
#edu data handle madoke
import numpy as np
import pandas as pd

#edu visualization ge
from matplotlib import pyplot as plt

#edu preprocessing madoke
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

#ee models try madtini, yavd better accuracy ede aduna madugtini,
#from sklearn.naive_bayes import GaussianNB
#from sklearn.tree import DecisionTreeClassifier
#from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix

dataset = pd.read_csv("/Users/monish/Downloads/heart.csv")
data_x=np.array(dataset.loc[:,
["age","sex","cp","trestbps","chol","fbs","restecg","thalach","exang",
"oldpeak","slope","ca","thal"]])
data_x

array([[63.,  1.,  3., ...,  0.,  0.,  1.],
       [37.,  1.,  2., ...,  0.,  0.,  2.],
       [41.,  0.,  1., ...,  2.,  0.,  2.],
       ...,
       [68.,  1.,  0., ...,  1.,  2.,  3.],
       [57.,  1.,  0., ...,  1.,  1.,  3.],
       [57.,  0.,  1., ...,  1.,  1.,  2.]])

data_y = dataset.iloc[:,-1].values
data_y

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

```
dataset.isnull().values.any() #empty value ella anta
```

```
False
```

evaga categorical handle mad beku

cp - 3rd column slope - 11th column ca - 12th column thal - 13th column

```
ct = ColumnTransformer(transformers=[("encoder",OneHotEncoder(),
[2])],remainder="passthrough")
data_x=ct.fit_transform(data_x)
ct = ColumnTransformer(transformers=[("encoder",OneHotEncoder(),
[13])],remainder="passthrough")
data_x=ct.fit_transform(data_x)
ct = ColumnTransformer(transformers=[("encoder",OneHotEncoder(),
[16])],remainder="passthrough")
data_x=ct.fit_transform(data_x)
ct = ColumnTransformer(transformers=[("encoder",OneHotEncoder(),
[21])],remainder="passthrough")
data_x=ct.fit_transform(data_x)
print(data_x)
```

```
[[  0.    1.    0.  ... 150.    0.    2.3]
 [  0.    0.    1.  ... 187.    0.    3.5]
 [  0.    0.    1.  ... 172.    0.    1.4]
 ...
 [  0.    0.    0.  ... 141.    0.    3.4]
 [  0.    0.    0.  ... 115.    1.    1.2]
 [  0.    0.    1.  ... 174.    0.    0. ]]
```

```
pd.DataFrame(data_x) #sumne reference ge aste, yallu use madola
```

```
      0    1    2    3    4    5    6    7    8    9  ...   15    16
17  \
0    0.0  1.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  1.0  ...  1.0  63.0
1.0
```

```
1    0.0  0.0  1.0  0.0  1.0  0.0  0.0  0.0  0.0  1.0  ...  0.0  37.0
1.0
2    0.0  0.0  1.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  41.0
0.0
3    0.0  0.0  1.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  56.0
1.0
4    0.0  0.0  1.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  57.0
0.0
..   ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...   ...
...
298  0.0  0.0  0.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  57.0
0.0
299  0.0  0.0  0.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  ...  1.0  45.0
1.0
300  0.0  0.0  0.0  1.0  0.0  0.0  1.0  0.0  0.0  0.0  ...  0.0  68.0
1.0
301  0.0  0.0  0.0  1.0  0.0  1.0  0.0  0.0  0.0  0.0  ...  0.0  57.0
1.0
302  0.0  0.0  1.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  ...  0.0  57.0
0.0

        18     19   20   21     22   23   24
0    145.0  233.0  1.0  0.0  150.0  0.0  2.3
1    130.0  250.0  0.0  1.0  187.0  0.0  3.5
2    130.0  204.0  0.0  0.0  172.0  0.0  1.4
3    120.0  236.0  0.0  1.0  178.0  0.0  0.8
4    120.0  354.0  0.0  1.0  163.0  1.0  0.6
..     ...    ...  ...  ...    ...  ...  ...
298  140.0  241.0  0.0  1.0  123.0  1.0  0.2
299  110.0  264.0  0.0  1.0  132.0  0.0  1.2
300  144.0  193.0  1.0  1.0  141.0  0.0  3.4
301  130.0  131.0  0.0  1.0  115.0  1.0  1.2
302  130.0  236.0  0.0  0.0  174.0  0.0  0.0

[303 rows x 25 columns]
```

```
le = LabelEncoder()
data_y=le.fit_transform(data_y)
data_y
```

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

```
#dataset size small, adike
x_train,x_test,y_train,y_test =
train_test_split(data_x,data_y,test_size=0.2,random_state=1)
x_test,y_test
```

```
(array([[  0. ,   0. ,   0. , ..., 145. ,   0. ,   6.2],
        [  0. ,   0. ,   0. , ..., 163. ,   0. ,   0. ],
        [  0. ,   0. ,   0. , ..., 150. ,   1. ,   0. ],
        ...,
        [  0. ,   0. ,   0. , ..., 182. ,   1. ,   3.8],
        [  0. ,   1. ,   0. , ..., 112. ,   1. ,   0.6],
        [  0. ,   0. ,   1. , ..., 154. ,   1. ,   0. ]]),
 array([0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1,
0,
        1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1,
0,
        0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1]))
```

```
#feature scaling for numerical data

sc = StandardScaler()
x_train[:,16:17] = sc.fit_transform(x_train[:,16:17])
x_test[:,16:17] = sc.transform(x_test[:,16:17])
sc = StandardScaler()
x_train[:,18:20] = sc.fit_transform(x_train[:,18:20])
x_test[:,18:20] = sc.transform(x_test[:,18:20])
sc = StandardScaler()
x_train[:,22:23] = sc.fit_transform(x_train[:,22:23])
x_test[:,22:23] = sc.transform(x_test[:,22:23])
sc = StandardScaler()
x_train[:,24:] = sc.fit_transform(x_train[:,24:])
x_test[:,24:] = sc.transform(x_test[:,24:])
```

```
x_train
```

```
array([[ 0.        ,  1.        ,  0.        , ...,  1.72150023,
         0.        , -0.93352855],
       [ 0.        ,  0.        ,  1.        , ...,  0.95592034,
         0.        , -0.93352855],
       [ 0.        ,  0.        ,  0.        , ...,  0.99845256,
         0.        , -0.75104558],
       ...,
       [ 0.        ,  0.        ,  1.        , ...,  2.23188682,
         0.        , -0.93352855],
       [ 0.        ,  0.        ,  0.        , ...,  0.99845256,
         1.        ,  0.52633516],
       [ 0.        ,  0.        ,  0.        , ...,  0.65819483,
         0.        ,  0.52633516]])
```

```
x_test
```

```
array([[ 0.        ,  0.        ,  0.        , ..., -0.19244949,
         0.        ,  4.72344331],
       [ 0.        ,  0.        ,  0.        , ...,  0.5731304 ,
         0.        , -0.93352855],
       [ 0.        ,  0.        ,  0.        , ...,  0.02021159,
         1.        , -0.93352855],
       ...,
       [ 0.        ,  0.        ,  0.        , ...,  1.3812425 ,
         1.        ,  2.53364775],
       [ 0.        ,  1.        ,  0.        , ..., -1.59601261,
         1.        , -0.38607966],
       [ 0.        ,  0.        ,  1.        , ...,  0.19034045,
         1.        , -0.93352855]])
```

```
#classifier = SVC(kernel="rbf",random_state=0)
#classifier.fit(x_train,y_train)
#classifier = GaussianNB()
#classifier.fit(x_train,y_train)
#classifier =
DecisionTreeClassifier(criterion="entropy",random_state=0)
#classifier.fit(x_train,y_train)
classifier =
RandomForestClassifier(n_estimators=200,random_state=0,criterion="entr
opy")
classifier.fit(x_train,y_train)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=200,
random_state=0)
```

```
ypred = classifier.predict(x_test)
ypred
```

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

```
# validation ge, not necessary
print(accuracy_score(ypred,y_test))
confusion_matrix(ypred,y_test)
```

```
0.7704918032786885
```

```
array([[21,  5],
       [ 9, 26]])
```