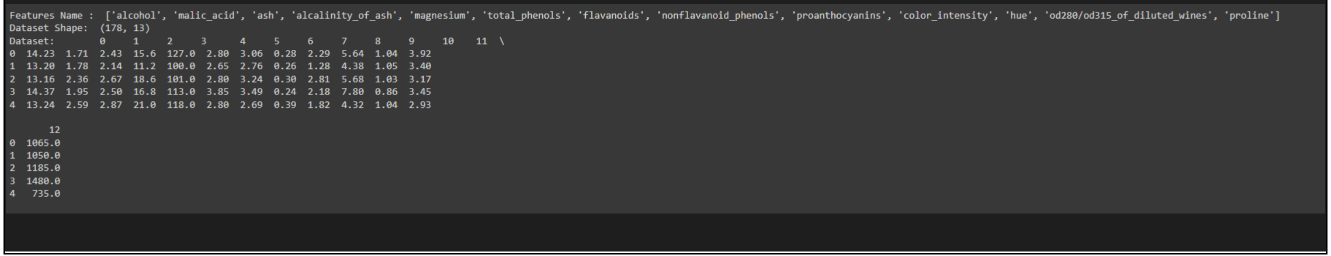
	Lembar Kerja Mahasiswa	Mata Kuliah Bahasan Halaman	: Data Mining : Decision Tree : 1/17

NIM	222410101095
Nama	Muhammad Afif Rohman Muzaky
Kelas	A
Program Studi	Sistem Informasi
Asisten	1. Renata Sayidatul Arikha 212410101057 2. Aprodhita Nanda Eka Wijaya 212410101071

LANGKAH KERJA

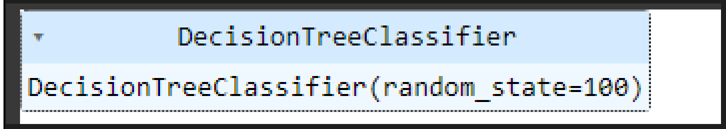
1. Ulangilah tahapan klasifikasi menggunakan dataset Wine. Hitunglah akurasinya berdasar confusion matrix dan classification report dengan persentase data testing 20%, 30% dan 40%

```
import numpy as np
import pandas as pd
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_wine
from matplotlib import pyplot as plt
from sklearn import datasets
from sklearn import tree

#Import dataset
wine_data = load_wine()
wine=pd.DataFrame(wine_data.data)
print ("Features Name : ", wine_data.feature_names)
print ("Dataset Shape: ", wine.shape)
print ("Dataset: ", wine.head())
```



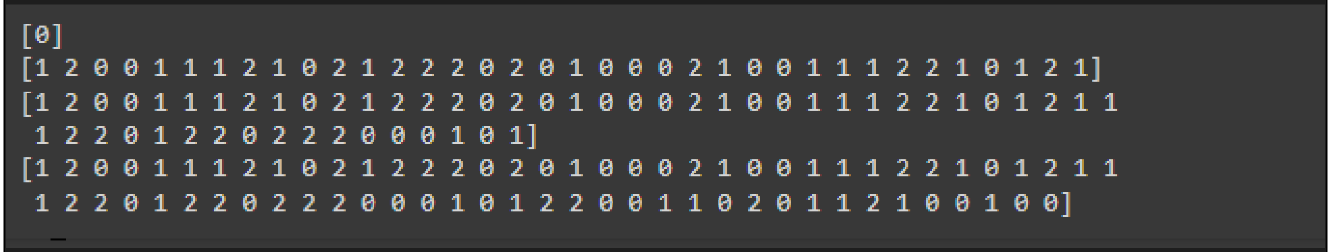
```
Features Name :  ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'proline']
Dataset Shape:  (178, 13)
Dataset:        0     1     2     3      4     5     6     7     8     9     10    11  \
0  14.23  1.71  2.43  15.6  127.0  2.80  3.06  0.28  2.29  5.64  1.04  3.92
1  13.20  1.78  2.14  11.2  100.0  2.65  2.76  0.26  1.28  4.38  1.05  3.40
2  13.16  2.36  2.67  18.6  101.0  2.80  3.24  0.30  2.81  5.68  1.03  3.17
3  14.37  1.95  2.50  16.8  113.0  3.85  3.49  0.24  2.18  7.80  0.86  3.45
4  13.24  2.59  2.87  21.0  118.0  2.80  2.69  0.39  1.82  4.32  1.04  2.93

       12
0  1065.0
1  1050.0
2  1185.0
3  1480.0
4   735.0
```

```
X = wine.values[:, 0:13]
Y = wine_data.target
#Train data 20%
X1_train, X1_test, y1_train, y1_test = train_test_split(X, Y, test_size = 0.2,
random_state = 100)
#Train data 30%
X2_train, X2_test, y2_train, y2_test = train_test_split(X, Y, test_size = 0.3,
random_state = 100)
#Train data 40%
X3_train, X3_test, y3_train, y3_test = train_test_split(X, Y, test_size = 0.4,
random_state = 100)

clf= DecisionTreeClassifier(random_state = 100)
clf.fit(X1_train, y1_train)
clf.fit(X2_train, y2_train)
clf.fit(X3_train, y3_train)
```



```
DecisionTreeClassifier(random_state=100)
```

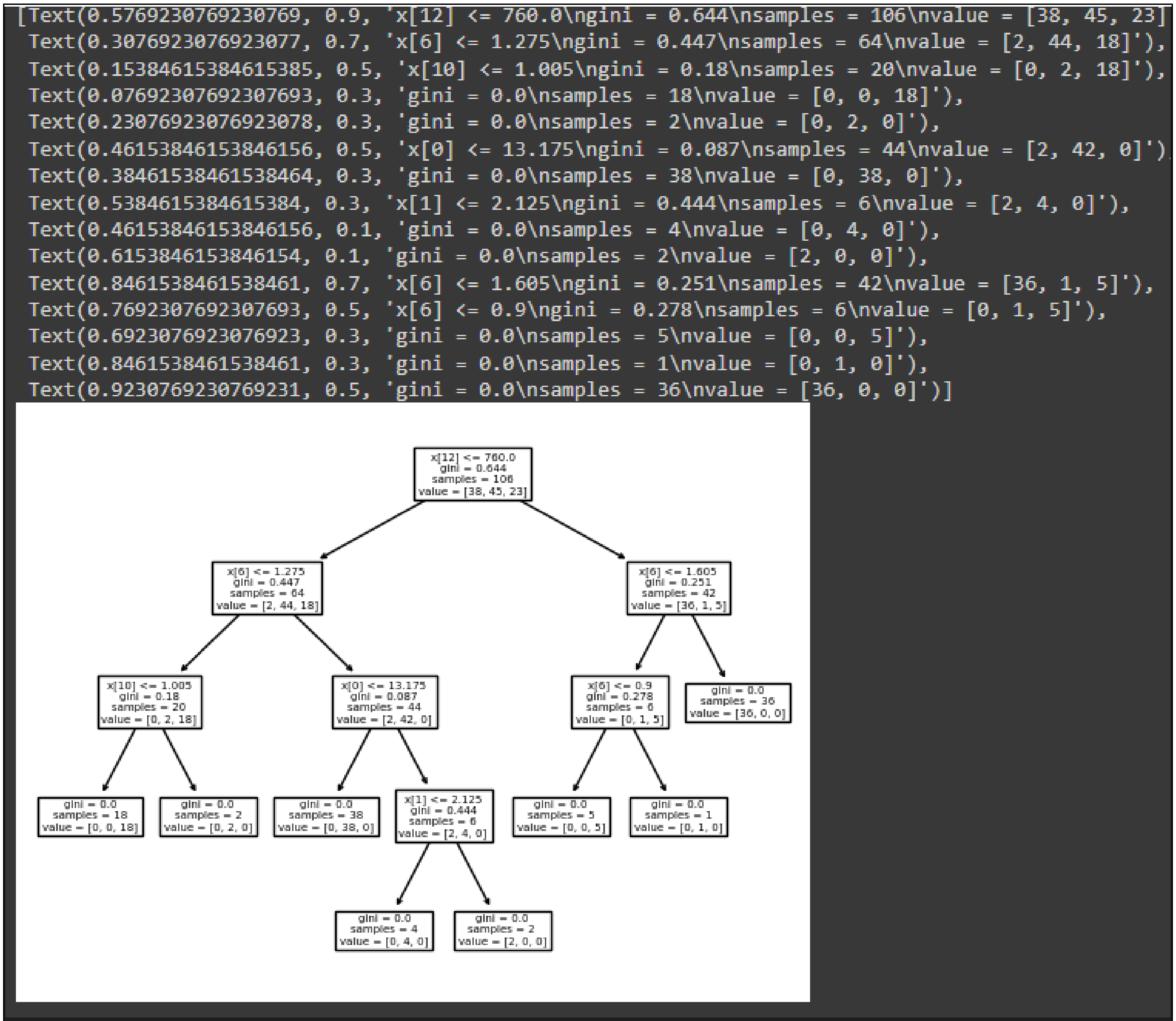
```
X=[[9.8, 1.43 ,2.1 ,11.8 , 107, 3, 2.75, 0.11, 1.2, 3.67, 1.02, 3.16, 900]]
Y_pred=clf.predict(X)
print(Y_pred)
Y_pred=clf.predict(X1_test)
print(Y_pred)
Y_pred=clf.predict(X2_test)
print(Y_pred)
Y_pred=clf.predict(X3_test)
print(Y_pred)
```



```
[0]
[1 2 0 0 1 1 1 2 1 0 2 1 2 2 2 0 2 0 1 0 0 0 2 1 0 0 1 1 1 2 2 1 0 1 2 1]
[1 2 0 0 1 1 1 2 1 0 2 1 2 2 2 0 2 0 1 0 0 0 2 1 0 0 1 1 1 2 2 1 0 1 2 1 1
 1 2 2 0 1 2 2 0 2 2 2 0 0 0 1 0 1]
[1 2 0 0 1 1 1 2 1 0 2 1 2 2 2 0 2 0 1 0 0 0 2 1 0 0 1 1 1 2 2 1 0 1 2 1 1
 1 2 2 0 1 2 2 0 2 2 2 0 0 0 1 0 1 2 2 0 0 1 1 0 2 0 1 1 2 1 0 0 1 0 0]
```

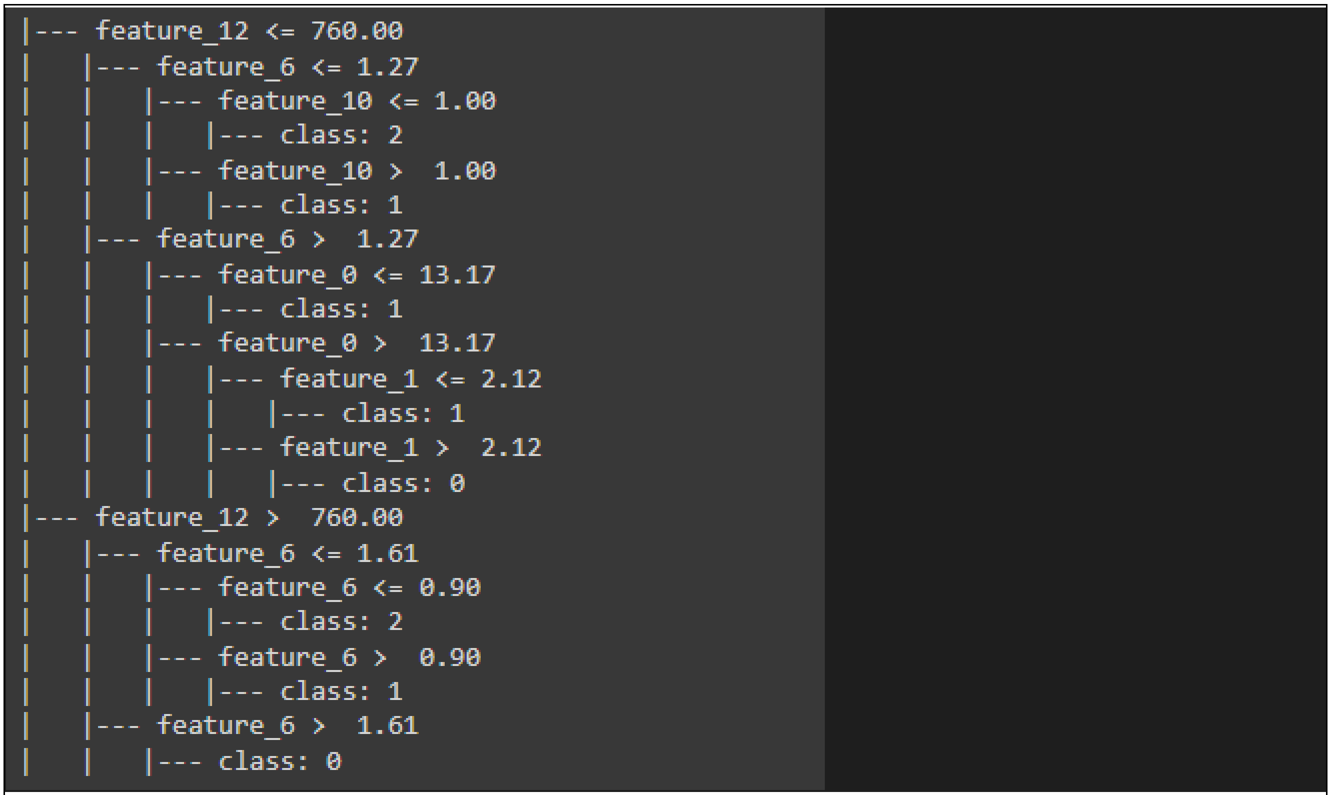
```
#Data 20%
print("Data Test 20% :")
```

```
Y_pred = clf.predict(X1_test)
print("Confusion Matrix : \n", confusion_matrix(y1_test, Y_pred))
print("Hasil Akurasi : {:.2f} \n".format(accuracy_score(y1_test, Y_pred)))

#Data 30%
print("Data Test 30% :")
Y_pred = clf.predict(X2_test)
print("Confusion Matrix : \n", confusion_matrix(y2_test, Y_pred))
print("Hasil Akurasi : {:.2f} \n".format(accuracy_score(y2_test, Y_pred)))

#Data 40%
print("Data Test 40% :")
Y_pred = clf.predict(X3_test)
print("Confusion Matrix : \n", confusion_matrix(y3_test, Y_pred))
print("Hasil Akurasi : {:.2f} \n".format(accuracy_score(y3_test, Y_pred)))






#Visualisasi Data

tree.plot_tree(clf)
```



```
[Text(0.5769230769230769, 0.9, 'x[12] <= 760.0\ngini = 0.644\nsamples = 106\nvalue = [38, 45, 23]
 Text(0.3076923076923077, 0.7, 'x[6] <= 1.275\ngini = 0.447\nsamples = 64\nvalue = [2, 44, 18]'),
 Text(0.15384615384615385, 0.5, 'x[10] <= 1.005\ngini = 0.18\nsamples = 20\nvalue = [0, 2, 18]'),
 Text(0.07692307692307693, 0.3, 'gini = 0.0\nsamples = 18\nvalue = [0, 0, 18]'),
 Text(0.23076923076923078, 0.3, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0]'),
 Text(0.46153846153846156, 0.5, 'x[0] <= 13.175\ngini = 0.087\nsamples = 44\nvalue = [2, 42, 0]')
 Text(0.38461538461538464, 0.3, 'gini = 0.0\nsamples = 38\nvalue = [0, 38, 0]'),
 Text(0.5384615384615384, 0.3, 'x[1] <= 2.125\ngini = 0.444\nsamples = 6\nvalue = [2, 4, 0]'),
 Text(0.46153846153846156, 0.1, 'gini = 0.0\nsamples = 4\nvalue = [0, 4, 0]'),
 Text(0.6153846153846154, 0.1, 'gini = 0.0\nsamples = 2\nvalue = [2, 0, 0]'),
 Text(0.8461538461538461, 0.7, 'x[6] <= 1.605\ngini = 0.251\nsamples = 42\nvalue = [36, 1, 5]'),
 Text(0.7692307692307693, 0.5, 'x[6] <= 0.9\ngini = 0.278\nsamples = 6\nvalue = [0, 1, 5]'),
 Text(0.6923076923076923, 0.3, 'gini = 0.0\nsamples = 5\nvalue = [0, 0, 5]'),
 Text(0.8461538461538461, 0.3, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.9230769230769231, 0.5, 'gini = 0.0\nsamples = 36\nvalue = [36, 0, 0]')]
```

```
text_representation = tree.export_text(clf)
print(text_representation)
```

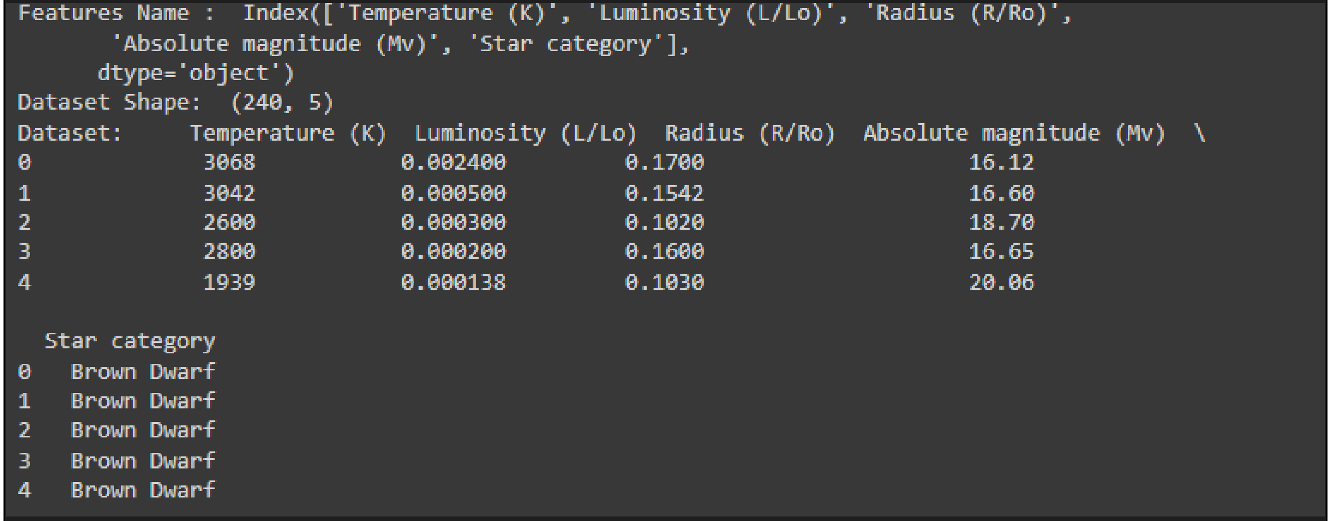


```
|--- feature_12 <= 760.00
|   |--- feature_6 <= 1.27
|   |   |--- feature_10 <= 1.00
|   |   |   |--- class: 2
|   |   |--- feature_10 >  1.00
|   |   |   |--- class: 1
|   |--- feature_6 >  1.27
|   |   |--- feature_0 <= 13.17
|   |   |   |--- class: 1
|   |   |--- feature_0 >  13.17
|   |   |   |--- feature_1 <= 2.12
|   |   |   |   |--- class: 1
|   |   |   |--- feature_1 >  2.12
|   |   |   |   |--- class: 0
|--- feature_12 >  760.00
|   |--- feature_6 <= 1.61
|   |   |--- feature_6 <= 0.90
|   |   |   |--- class: 2
|   |   |--- feature_6 >  0.90
|   |   |   |--- class: 1
|   |--- feature_6 >  1.61
|   |   |--- class: 0
```

2. Ulangilah tahapan klasifikasi menggunakan dataset Stars. Hitunglah akurasinya berdasar confusion matrix dan classification report dengan persentase data testing 20%, 30% dan 40%

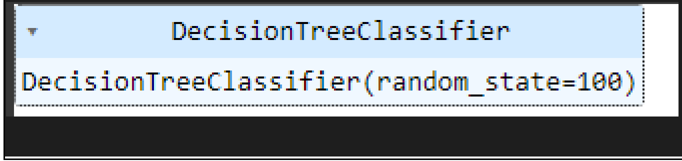
```
import numpy as np
import pandas as pd
import sklearn
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt
from sklearn import datasets
from sklearn import tree

#Import Stars.csv dari drive
stars_data = pd.read_csv('Stars.csv', index_col=0)
stars=pd.DataFrame(stars_data)
```

```
print ("Features Name : ", stars.columns)
print ("Dataset Shape: ", stars.shape)
print ("Dataset: ", stars.head())
```



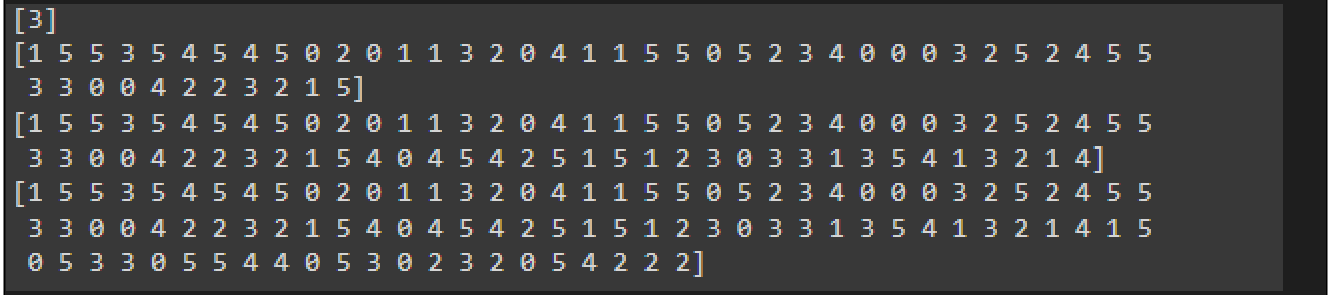
```
Features Name :  Index(['Temperature (K)', 'Luminosity (L/Lo)', 'Radius (R/Ro)',
       'Absolute magnitude (Mv)', 'Star category'],
      dtype='object')
Dataset Shape:  (240, 5)
Dataset:     Temperature (K)  Luminosity (L/Lo)  Radius (R/Ro)  Absolute magnitude (Mv)  \
0             3068           0.002400         0.1700                    16.12   
1             3042           0.000500         0.1542                    16.60   
2             2600           0.000300         0.1020                    18.70   
3             2800           0.000200         0.1600                    16.65   
4             1939           0.000138         0.1030                    20.06   

  Star category  
0   Brown Dwarf  
1   Brown Dwarf  
2   Brown Dwarf  
3   Brown Dwarf  
4   Brown Dwarf  
```

```
stars1 = stars.drop(columns=['Star category'])
X = stars1.values[:, 0:4]
Y = stars['Star category']

label_encoder = LabelEncoder()
Y_encoded = label_encoder.fit_transform(Y)

X1_train, X1_test, y1_train, y1_test = train_test_split(X, Y_encoded,
test_size = 0.2, random_state = 100)
X2_train, X2_test, y2_train, y2_test = train_test_split(X, Y_encoded,
test_size = 0.3, random_state = 100)
X3_train, X3_test, y3_train, y3_test = train_test_split(X, Y_encoded,
test_size = 0.4, random_state = 100)

clf= DecisionTreeClassifier(random_state = 100)
clf.fit(X1_train, y1_train)
clf.fit(X2_train, y2_train)
clf.fit(X3_train, y3_train)
```



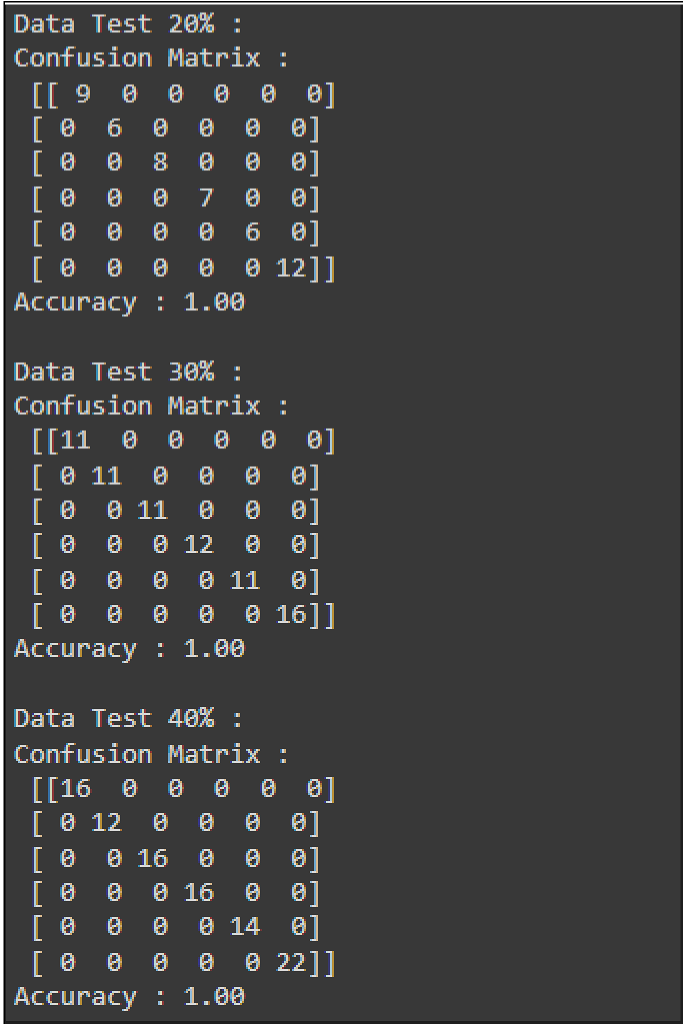
```
DecisionTreeClassifier(random_state=100)
```

```
X=[[1000, 0.045, 0.07, 12]]
Y_pred=clf.predict(X)
print(Y_pred)
Y_pred=clf.predict(X1_test)
print(Y_pred)
Y_pred=clf.predict(X2_test)
print(Y_pred)
Y_pred=clf.predict(X3_test)
print(Y_pred)
```



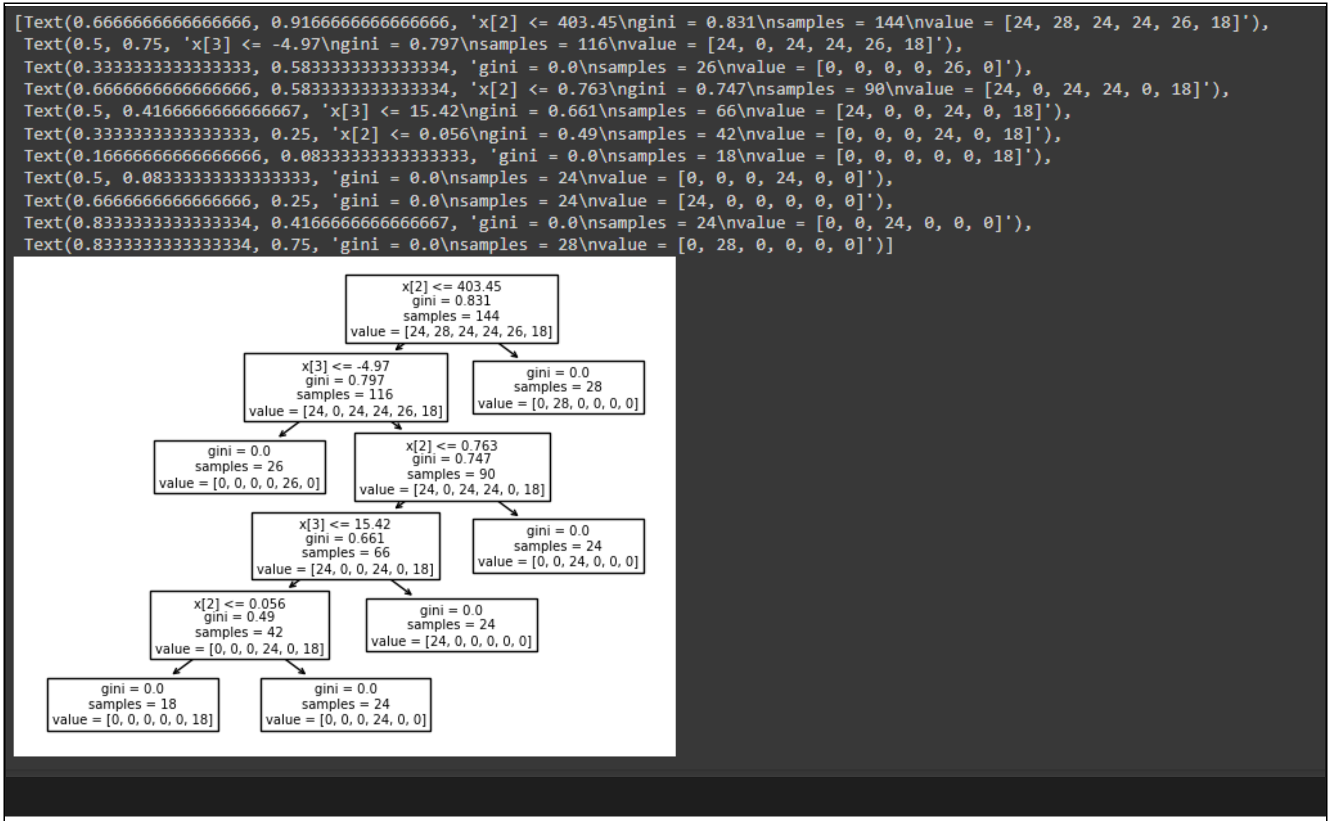
```
[3]
[1 5 5 3 5 4 5 4 5 0 2 0 1 1 3 2 0 4 1 1 5 5 0 5 2 3 4 0 0 0 3 2 5 2 4 5 5
 3 3 0 0 4 2 2 3 2 1 5]
[1 5 5 3 5 4 5 4 5 0 2 0 1 1 3 2 0 4 1 1 5 5 0 5 2 3 4 0 0 0 3 2 5 2 4 5 5
 3 3 0 0 4 2 2 3 2 1 5 4 0 4 5 4 2 5 1 5 1 2 3 0 3 3 1 3 5 4 1 3 2 1 4]
[1 5 5 3 5 4 5 4 5 0 2 0 1 1 3 2 0 4 1 1 5 5 0 5 2 3 4 0 0 0 3 2 5 2 4 5 5
 3 3 0 0 4 2 2 3 2 1 5 4 0 4 5 4 2 5 1 5 1 2 3 0 3 3 1 3 5 4 1 3 2 1 4 1 5
 0 5 3 3 0 5 5 4 4 0 5 3 0 2 3 2 0 5 4 2 2 2]
```

```
#Data 20%
print("Data Test 20% :")
Y_pred = clf.predict(X1_test)
print("Confusion Matrix : \n", confusion_matrix(y1_test, Y_pred))
print("Accuracy : {:.2f} \n".format(accuracy_score(y1_test, Y_pred)))


#Data 30%
print("Data Test 30% :")
Y_pred = clf.predict(X2_test)
print("Confusion Matrix : \n", confusion_matrix(y2_test, Y_pred))
print("Accuracy : {:.2f} \n".format(accuracy_score(y2_test, Y_pred)))


#Data 40%
print("Data Test 40% :")
Y_pred = clf.predict(X3_test)
print("Confusion Matrix : \n", confusion_matrix(y3_test, Y_pred))
print("Accuracy : {:.2f} \n".format(accuracy_score(y3_test, Y_pred)))
```



```
Data Test 20% :
Confusion Matrix :
 [[ 9  0  0  0  0  0]
 [ 0  6  0  0  0  0]
 [ 0  0  8  0  0  0]
 [ 0  0  0  7  0  0]
 [ 0  0  0  0  6  0]
 [ 0  0  0  0  0 12]]
Accuracy : 1.00

Data Test 30% :
Confusion Matrix :
 [[11  0  0  0  0  0]
 [ 0 11  0  0  0  0]
 [ 0  0 11  0  0  0]
 [ 0  0  0 12  0  0]
 [ 0  0  0  0 11  0]
 [ 0  0  0  0  0 16]]
Accuracy : 1.00

Data Test 40% :
Confusion Matrix :
 [[16  0  0  0  0  0]
 [ 0 12  0  0  0  0]
 [ 0  0 16  0  0  0]
 [ 0  0  0 16  0  0]
 [ 0  0  0  0 14  0]
 [ 0  0  0  0  0 22]]
Accuracy : 1.00
```

```
tree.plot_tree(clf)
```

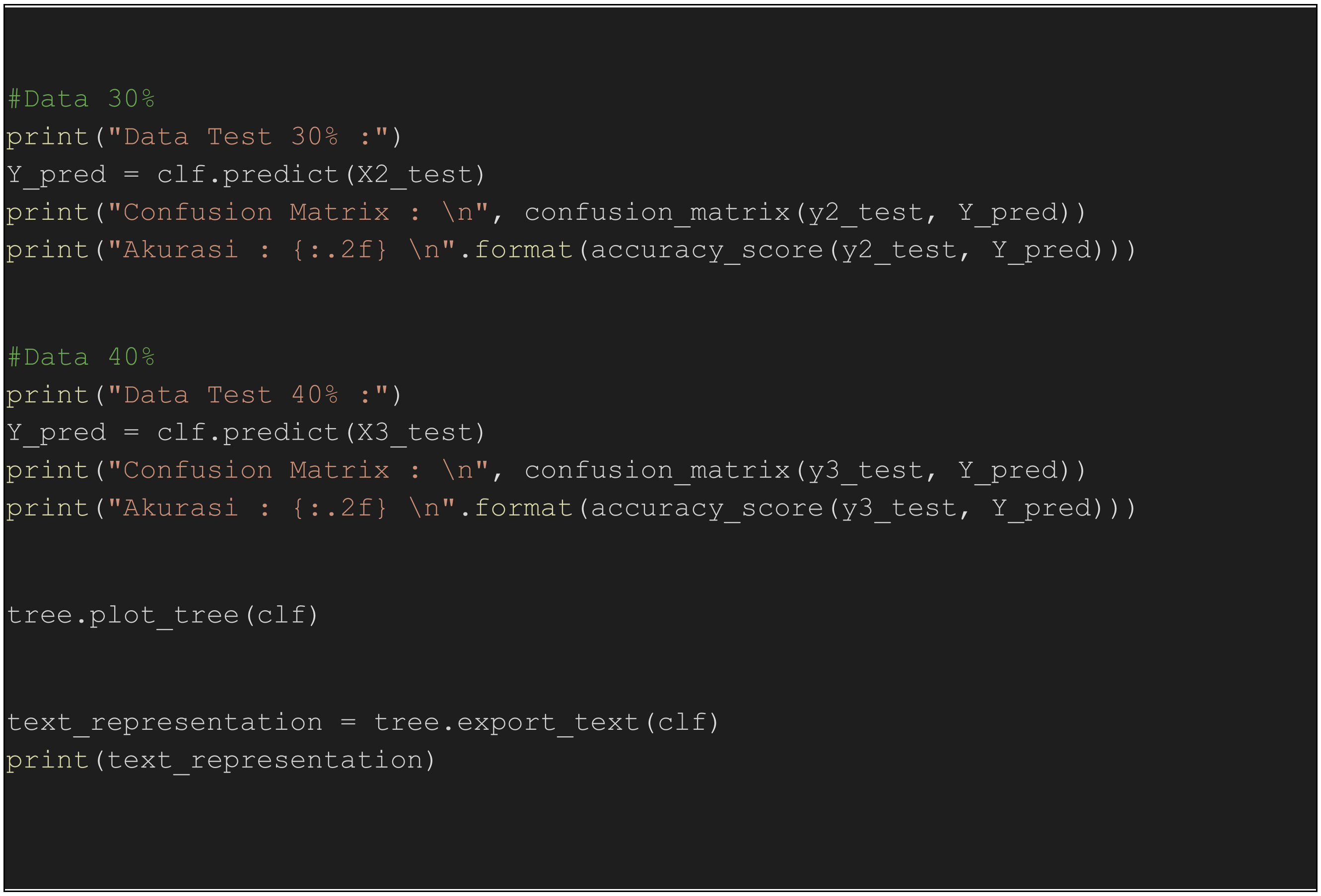


```
[Text(0.6666666666666666, 0.9166666666666666, 'x[2] <= 403.45\ngini = 0.831\nsamples = 144\nvalue = [24, 28, 24, 24, 26, 18]'),
 Text(0.5, 0.75, 'x[3] <= -4.97\ngini = 0.797\nsamples = 116\nvalue = [24, 0, 24, 24, 26, 18]'),
 Text(0.3333333333333333, 0.5833333333333334, 'gini = 0.0\nsamples = 26\nvalue = [0, 0, 0, 0, 26, 0]'),
 Text(0.6666666666666666, 0.5833333333333334, 'x[2] <= 0.763\ngini = 0.747\nsamples = 90\nvalue = [24, 0, 24, 24, 0, 18]'),
 Text(0.5, 0.4166666666666667, 'x[3] <= 15.42\ngini = 0.661\nsamples = 66\nvalue = [24, 0, 0, 24, 0, 18]'),
 Text(0.3333333333333333, 0.25, 'x[2] <= 0.056\ngini = 0.49\nsamples = 42\nvalue = [0, 0, 0, 24, 0, 18]'),
 Text(0.16666666666666666, 0.08333333333333333, 'gini = 0.0\nsamples = 18\nvalue = [0, 0, 0, 0, 0, 18]'),
 Text(0.5, 0.08333333333333333, 'gini = 0.0\nsamples = 24\nvalue = [0, 0, 0, 24, 0, 0]'),
 Text(0.6666666666666666, 0.25, 'gini = 0.0\nsamples = 24\nvalue = [24, 0, 0, 0, 0, 0]'),
 Text(0.8333333333333334, 0.4166666666666667, 'gini = 0.0\nsamples = 24\nvalue = [0, 0, 24, 0, 0, 0]'),
 Text(0.8333333333333334, 0.75, 'gini = 0.0\nsamples = 28\nvalue = [0, 28, 0, 0, 0, 0]')]
```

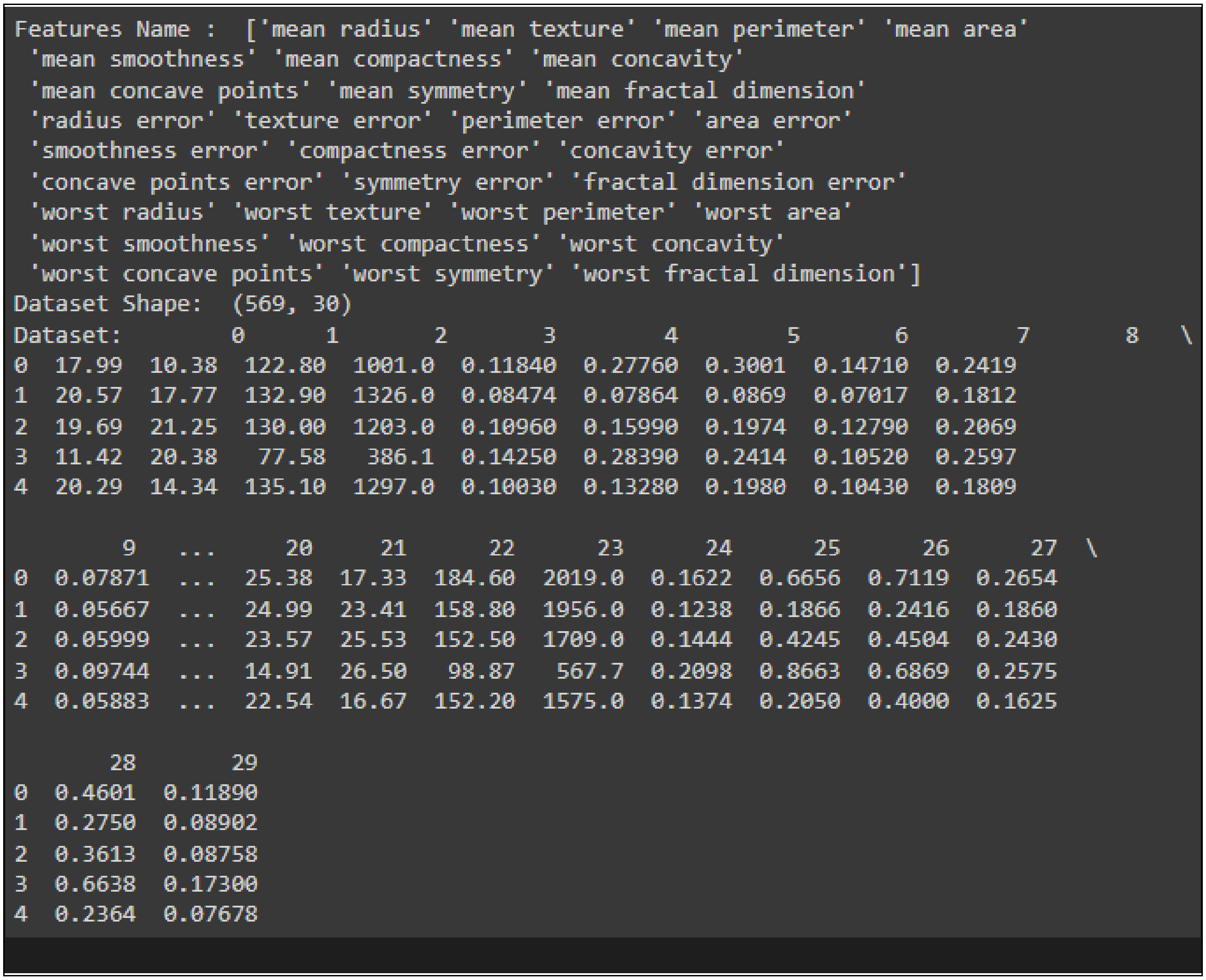
3. Ulangilah tahapan klasifikasi menggunakan dataset Breast Cancer. Hitunglah akurasinya berdasar confusion matrix dan classification report dengan persentase data testing 20%, 30% dan 40%

```
import numpy as np
import pandas as pd
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_breast_cancer
from matplotlib import pyplot as plt
from sklearn import datasets
from sklearn import tree


breast_cancer_data = load_breast_cancer()
breast_cancer=pd.DataFrame(breast_cancer_data.data)
```

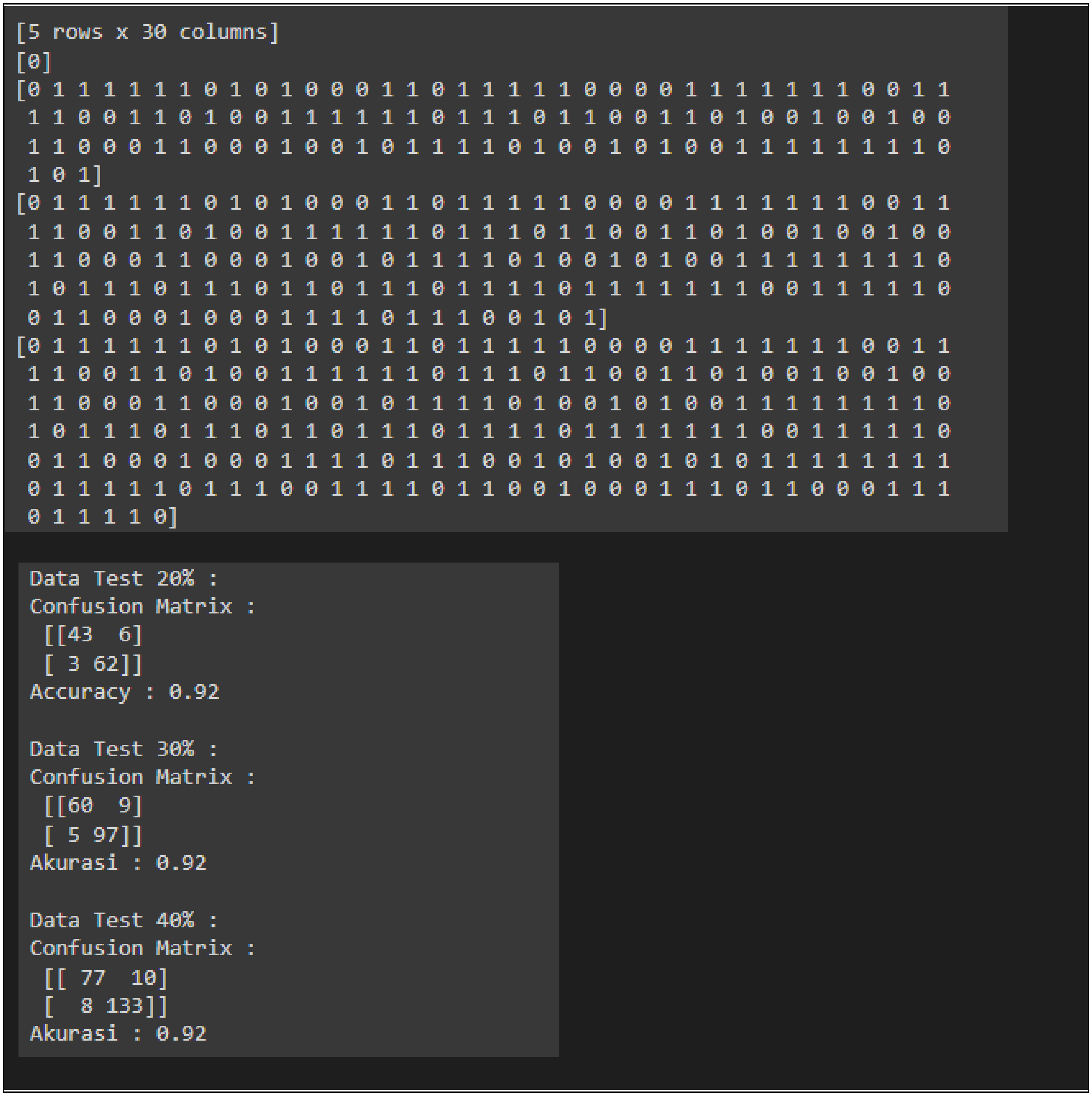
```
print ("Features Name : ", breast_cancer_data.feature_names)
print ("Dataset Shape: ", breast_cancer.shape)
print ("Dataset: ", breast_cancer.head())


X = breast_cancer.values[:, 0:30]
Y = breast_cancer_data.target


X1_train, X1_test, y1_train, y1_test = train_test_split(X, Y, test_size = 0.2,
random_state = 100)
X2_train, X2_test, y2_train, y2_test = train_test_split(X, Y, test_size = 0.3,
random_state = 100)
X3_train, X3_test, y3_train, y3_test = train_test_split(X, Y, test_size = 0.4,
random_state = 100)


clf= DecisionTreeClassifier(random_state = 100)
clf.fit(X1_train, y1_train)
clf.fit(X2_train, y2_train)
clf.fit(X3_train, y3_train)


X=[[180, 180, 180, 180, 180, 180, 180, 180, 180, 180,
    180, 180, 180, 180, 180, 180, 180, 180, 180, 180,
    180, 180, 180, 180, 180, 180, 180 ,180, 180, 180]]
Y_pred=clf.predict(X)
print(Y_pred)
Y_pred=clf.predict(X1_test)
print(Y_pred)
Y_pred=clf.predict(X2_test)
print(Y_pred)
Y_pred=clf.predict(X3_test)
print(Y_pred)


#Data 20%
print("Data Test 20% :")
Y_pred = clf.predict(X1_test)
print("Confusion Matrix : \n", confusion_matrix(y1_test, Y_pred))
print("Accuracy : {:.2f} \n".format(accuracy_score(y1_test, Y_pred)))
```



```
#Data 30%
print("Data Test 30% :")
Y_pred = clf.predict(X2_test)
print("Confusion Matrix : \n", confusion_matrix(y2_test, Y_pred))
print("Akurasi : {:.2f} \n".format(accuracy_score(y2_test, Y_pred)))


#Data 40%
print("Data Test 40% :")
Y_pred = clf.predict(X3_test)
print("Confusion Matrix : \n", confusion_matrix(y3_test, Y_pred))
print("Akurasi : {:.2f} \n".format(accuracy_score(y3_test, Y_pred)))


tree.plot_tree(clf)


text_representation = tree.export_text(clf)
print(text_representation)
```

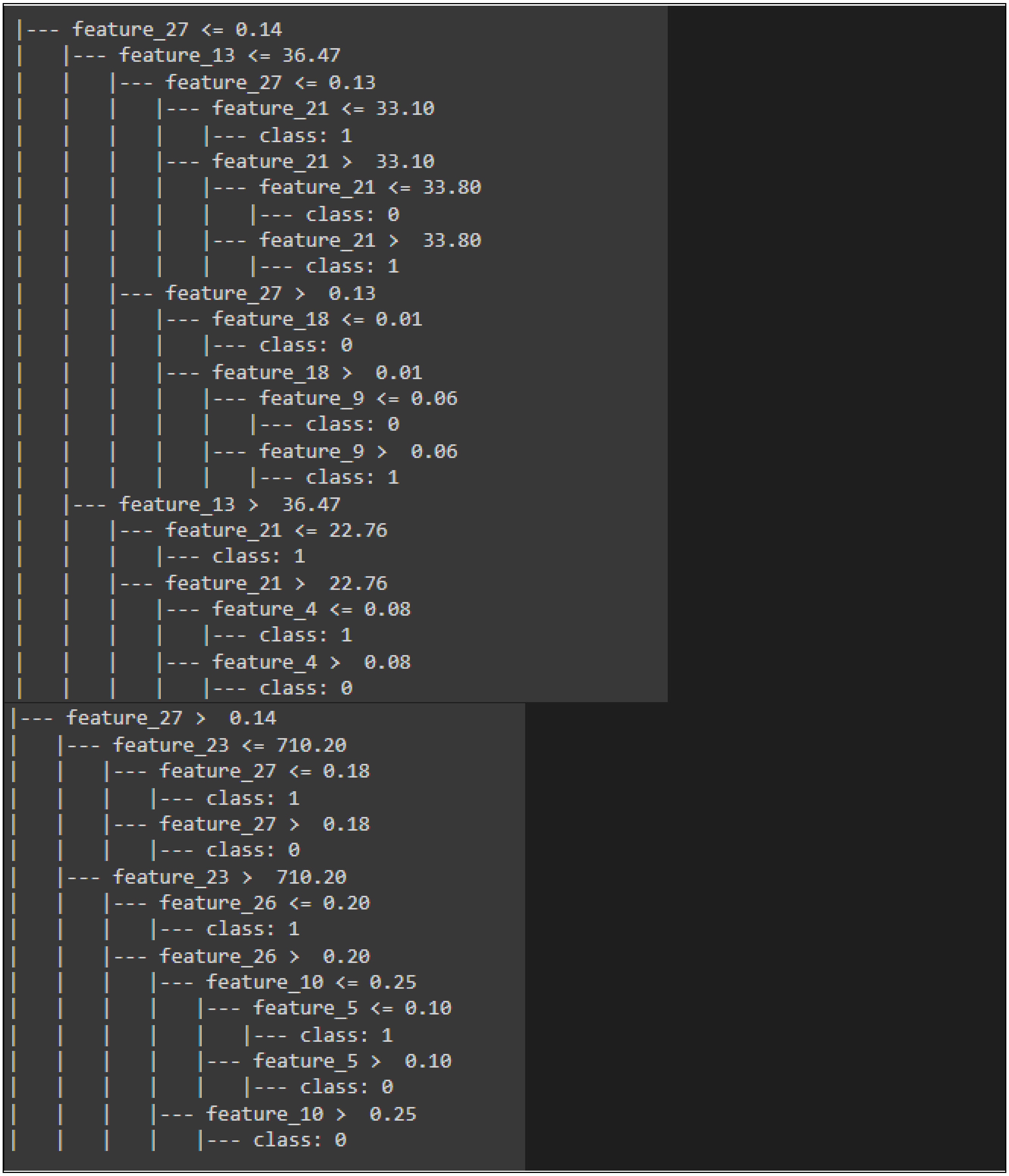


```
Features Name :  ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
Dataset Shape:  (569, 30)
Dataset:        0      1       2       3        4        5       6        7       8   \
0  17.99  10.38  122.80  1001.0  0.11840  0.27760  0.3001  0.14710  0.2419
1  20.57  17.77  132.90  1326.0  0.08474  0.07864  0.0869  0.07017  0.1812
2  19.69  21.25  130.00  1203.0  0.10960  0.15990  0.1974  0.12790  0.2069
3  11.42  20.38   77.58   386.1  0.14250  0.28390  0.2414  0.10520  0.2597
4  20.29  14.34  135.10  1297.0  0.10030  0.13280  0.1980  0.10430  0.1809

         9  ...     20     21      22      23      24      25      26      27  \
0  0.07871  ...  25.38  17.33  184.60  2019.0  0.1622  0.6656  0.7119  0.2654
1  0.05667  ...  24.99  23.41  158.80  1956.0  0.1238  0.1866  0.2416  0.1860
2  0.05999  ...  23.57  25.53  152.50  1709.0  0.1444  0.4245  0.4504  0.2430
3  0.09744  ...  14.91  26.50   98.87   567.7  0.2098  0.8663  0.6869  0.2575
4  0.05883  ...  22.54  16.67  152.20  1575.0  0.1374  0.2050  0.4000  0.1625

       28       29
0  0.4601  0.11890
1  0.2750  0.08902
2  0.3613  0.08758
3  0.6638  0.17300
4  0.2364  0.07678
```

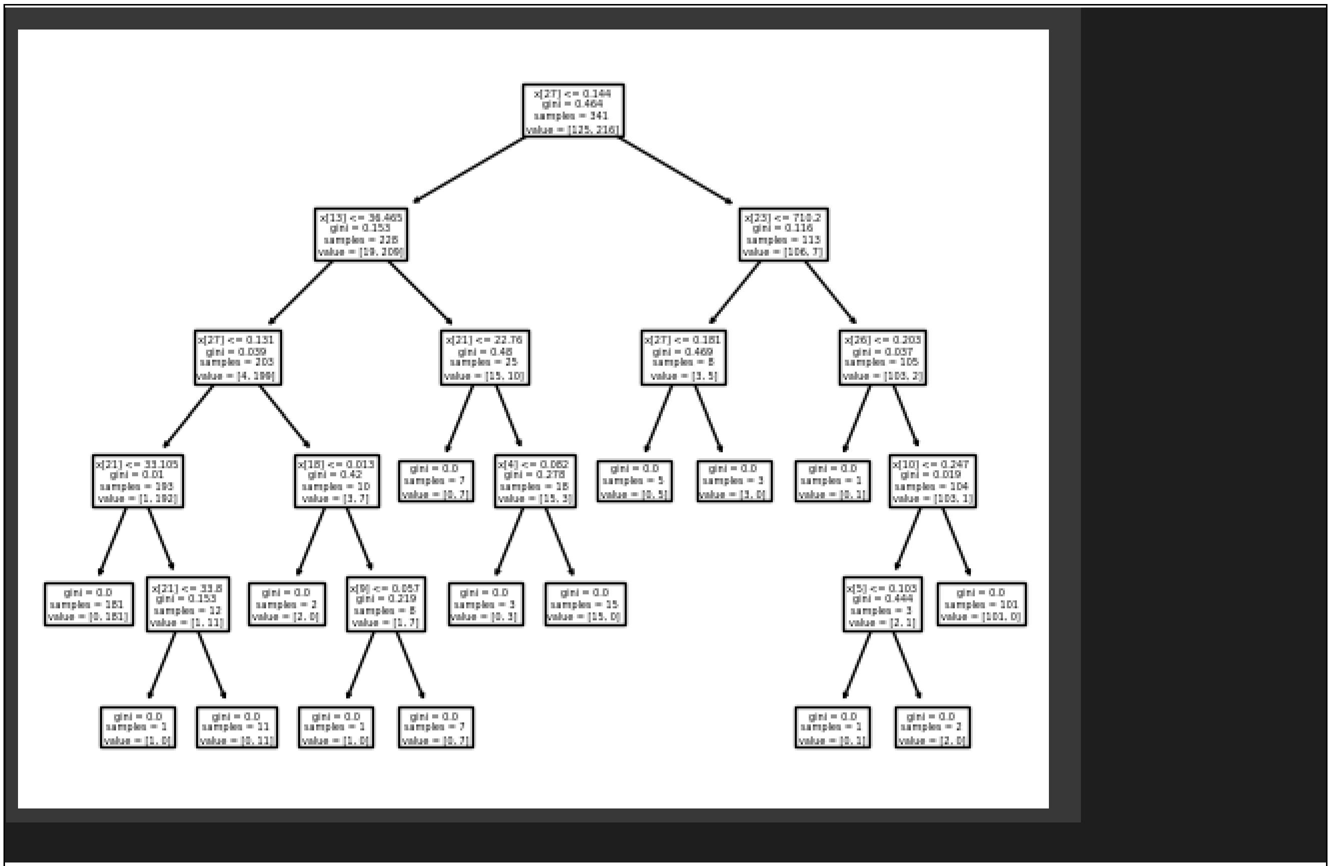


```
[5 rows x 30 columns]
[0]
[0 1 1 1 1 1 1 0 1 0 1 0 0 0 1 1 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1
 1 1 0 0 1 1 0 1 0 0 1 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 1 0 0 1 0 0 1 0 0
 1 1 0 0 0 1 1 0 0 0 1 0 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 1 1 1 1 0
 1 0 1]
[0 1 1 1 1 1 1 0 1 0 1 0 0 0 1 1 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1
 1 1 0 0 1 1 0 1 0 0 1 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 1 0 0 1 0 0 1 0 0
 1 1 0 0 0 1 1 0 0 0 1 0 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 1 1 1 1 0
 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0
 0 1 1 0 0 0 1 0 0 0 1 1 1 1 0 1 1 1 0 0 1 0 1]
[0 1 1 1 1 1 1 0 1 0 1 0 0 0 1 1 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1
 1 1 0 0 1 1 0 1 0 0 1 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 1 0 0 1 0 0 1 0 0
 1 1 0 0 0 1 1 0 0 0 1 0 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 1 1 1 1 0
 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0
 0 1 1 0 0 0 1 0 0 0 1 1 1 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1 0 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 0 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 0 1 1 1
 0 1 1 1 1 0]
```

```
Data Test 20% :
Confusion Matrix :
 [[43  6]
 [ 3 62]]
Accuracy : 0.92

Data Test 30% :
Confusion Matrix :
 [[60  9]
 [ 5 97]]
Akurasi : 0.92

Data Test 40% :
Confusion Matrix :
 [[ 77  10]
 [  8 133]]
Akurasi : 0.92
```



```
|--- feature_27 <= 0.14
|   |--- feature_13 <= 36.47
|   |   |--- feature_27 <= 0.13
|   |   |   |--- feature_21 <= 33.10
|   |   |   |   |--- class: 1
|   |   |   |--- feature_21 >  33.10
|   |   |   |   |--- feature_21 <= 33.80
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_21 >  33.80
|   |   |   |   |   |--- class: 1
|   |   |--- feature_27 >  0.13
|   |   |   |--- feature_18 <= 0.01
|   |   |   |   |--- class: 0
|   |   |   |--- feature_18 >  0.01
|   |   |   |   |--- feature_9 <= 0.06
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_9 >  0.06
|   |   |   |   |   |--- class: 1
|   |--- feature_13 >  36.47
|   |   |--- feature_21 <= 22.76
|   |   |   |--- class: 1
|   |   |--- feature_21 >  22.76
|   |   |   |--- feature_4 <= 0.08
|   |   |   |   |--- class: 1
|   |   |   |--- feature_4 >  0.08
|   |   |   |   |--- class: 0
|--- feature_27 >  0.14
|   |--- feature_23 <= 710.20
|   |   |--- feature_27 <= 0.18
|   |   |   |--- class: 1
|   |   |--- feature_27 >  0.18
|   |   |   |--- class: 0
|   |--- feature_23 >  710.20
|   |   |--- feature_26 <= 0.20
|   |   |   |--- class: 1
|   |   |--- feature_26 >  0.20
|   |   |   |--- feature_10 <= 0.25
|   |   |   |   |--- feature_5 <= 0.10
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_5 >  0.10
|   |   |   |   |   |--- class: 0
|   |   |   |--- feature_10 >  0.25
|   |   |   |   |--- class: 0
```


x[27] <= 0.144 gini = 0.464 samples = 341 value = [125, 216]
x[13] <= 38.465 gini = 0.153 samples = 228 value = [19, 209]
x[23] <= 710.2 gini = 0.116 samples = 113 value = [106, 7]
x[27] <= 0.131 gini = 0.039 samples = 203 value = [4, 199]
x[21] <= 22.76 gini = 0.48 samples = 25 value = [15, 10]
x[27] <= 0.181 gini = 0.469 samples = 8 value = [3, 5]
x[26] <= 0.203 gini = 0.037 samples = 105 value = [103, 2]
x[21] <= 33.105 gini = 0.01 samples = 193 value = [1, 192]
x[18] <= 0.013 gini = 0.42 samples = 10 value = [3, 7]
gini = 0.0 samples = 7 value = [0, 7]
x[4] <= 0.082 gini = 0.278 samples = 18 value = [15, 3]
gini = 0.0 samples = 5 value = [0, 5]
gini = 0.0 samples = 3 value = [3, 0]
gini = 0.0 samples = 1 value = [0, 1]
x[10] <= 0.247 gini = 0.019 samples = 104 value = [103, 1]
gini = 0.0 samples = 181 value = [0, 181]
x[21] <= 33.8 gini = 0.153 samples = 12 value = [1, 11]
gini = 0.0 samples = 2 value = [2, 0]
x[9] <= 0.057 gini = 0.219 samples = 8 value = [1, 7]
gini = 0.0 samples = 3 value = [0, 3]
gini = 0.0 samples = 15 value = [15, 0]
x[5] <= 0.103 gini = 0.444 samples = 3 value = [2, 1]
gini = 0.0 samples = 101 value = [101, 0]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 11 value = [0, 11]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 7 value = [0, 7]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 2 value = [2, 0]

HASIL DAN ANALISIS DATA

Tahap pertama dari pengerjaan setiap nomer dari LKM 4 ini adalah mengimport dataset untuk melakukan Langkah selanjutnya yaitu dataset training yang mana klasikasi yang dilakukan pada dataset ini yaitu dengan presentase 20% 30% 40% ,

Dari training yang dilakukan pada setiap Langkah yang di minta oleh LKM didapatkan data sebagai berikut :

Tingkat Akurasi (Dataset Wine) :

20% : Tingkat Akurasi 78%

30% : Tingkat Akurasi 83%

40% : Tingkat Akurasi 88%

Tingkat Akurasi (Dataset Stars) :

20% ,30%,40% : Tingkat Akurasi 100%

Tingkat Akurasi (Dataset Breast Cancer) :

20% ,30%,40% : Tingkat Akurasi 92%

KESIMPULAN

Jadi pada LKM kali ini kita diminta untuk memahami Bagaimana cara data training metode decision tree bekerja yang mana setelah mengerjakan LKM 4 kali ini saya dapat menyimpulkan sebagai berikut :

Decision tree sendiri adalah sebuah algoritma pembelajaran atau algoritma training yang mana Decision tree sendiri digunakan untuk klasifikasi dan regresi. Pengambilan Keputusan, dan analisis resiko. Setiap fitur akan diwakili dengan nodenya mewakili sebuah fitur dan setiap cabang mewakilikeputusan berdasarkan fitur dan setiap daun mewakili hasil dari prediksi. Seperti yang dijelaskan pada modul yang diberikan pada setiap langkahnya,membentuk pemisahan atau "split" pada setiap node, dan secara rekursif membagi data kedalam sub tree yang semakin homogen. Kemudian decision tree digunakan untuk memprediksi kelas atau nilai dari data baru dengan melewati dari node ke node. Proses ini berlanjut hingga kriteria penghentian terpenuhi.

Decision tree diterapkan untuk melakukan training pada LKM 4 ini dataset yang digunakan adalah :

(Wine,Stars, dan Breast Cancer)

Dengan decision tree juga kita dapatkan akurasi pada setiap trainingnya dengan nilai :

Tingkat Akurasi (Dataset Wine) :

20% : Tingkat Akurasi 78%

30% : Tingkat Akurasi 83%

40% : Tingkat Akurasi 88%

Tingkat Akurasi (Dataset Stars) :

20% ,30%,40% : Tingkat Akurasi 100%

Tingkat Akurasi (Dataset Breast Cancer) : 20% ,30%,40% : Tingkat Akurasi 92%	
Link Google Colab	https://colab.research.google.com/drive/1UZl8oG7NZk59WMKdYPN_bwcKYc6b77tf?usp=sharing
Link Youtube (Unlisted)	https://youtu.be/2m-7CB0dVvg

Jember, ………………………….2024

Mengetahui,
Dosen Datamining

Fajrin Nurman Arifin, S.T., M.Eng
NIP. 198511282015041002

Asisten,

(Nama Jelas)
NIM.