```
import pandas as pd
data = pd.read_csv('zoo_data(For Decision Tree Program)(1).csv')

data.head()
```

```
   1  0  0.1  1.1  0.2  0.3  1.2  1.3  1.4  1.5  0.4  0.5  4  0.6  0.7
1.6  \
0  1  0    0    1    0    0    0    1    1    1    0    0  4    1    0
1
1  0  0    1    0    0    1    1    1    1    0    0    1  0    1    0
0
2  1  0    0    1    0    0    1    1    1    1    0    0  4    0    0
1
3  1  0    0    1    0    0    1    1    1    1    0    0  4    1    0
1
4  1  0    0    1    0    0    0    1    1    1    0    0  4    1    0
1


   1.7
0    1
1    4
2    1
3    1
4    1
```

```
y = data["1.7"]

import numpy as np
y = np.array(y)
print(y)
```

```
[1 4 1 1 1 1 4 4 1 1 2 4 7 7 7 2 1 4 1 2 2 1 2 6 5 5 1 1 1 6 1 1 2 4 1
1 2
 4 6 6 2 6 2 1 1 7 1 1 1 1 6 5 7 1 1 2 2 2 2 4 4 3 1 1 1 1 1 1 1 1 2 7
4 1
 1 3 7 2 2 3 7 4 2 1 7 4 2 6 5 3 3 4 1 1 2 1 6 1 7 2]
```

```
x = data.drop(columns=["1.7"])

print(x)
```

```
    1  0  0.1  1.1  0.2  0.3  1.2  1.3  1.4  1.5  0.4  0.5  4  0.6
0.7  1.6
0   1  0    0    1    0    0    0    1    1    1    0    0  4    1
0    1
1   0  0    1    0    0    1    1    1    1    0    0    1  0    1
0    0
2   1  0    0    1    0    0    1    1    1    1    0    0  4    0
0    1
3   1  0    0    1    0    0    1    1    1    1    0    0  4    1
0    1
4   1  0    0    1    0    0    0    1    1    1    0    0  4    1
```

```
0    1
.. .. ..  ...  ...  ...  ...  ...  ...  ...  ...  ...  ... ..  ...  ..
.  ...
95  1  0    0    1    0    0    0    1    1    1    0    0  2    1
0    1
96  1  0    1    0    1    0    0    0    0    1    1    0  6    0
0    0
97  1  0    0    1    0    0    1    1    1    1    0    0  4    1
0    1
98  0  0    1    0    0    0    0    0    0    1    0    0  0    0
0    0
99  0  1    1    0    1    0    0    0    1    1    0    0  2    1
0    0

[100 rows x 16 columns]
```

```
x
```

```
    1  0  0.1  1.1  0.2  0.3  1.2  1.3  1.4  1.5  0.4  0.5  4  0.6
0.7  1.6
0   1  0    0    1    0    0    0    1    1    1    0    0  4    1
0    1
1   0  0    1    0    0    1    1    1    1    0    0    1  0    1
0    0
2   1  0    0    1    0    0    1    1    1    1    0    0  4    0
0    1
3   1  0    0    1    0    0    1    1    1    1    0    0  4    1
0    1
4   1  0    0    1    0    0    0    1    1    1    0    0  4    1
0    1
.. .. ..  ...  ...  ...  ...  ...  ...  ...  ...  ...  ... ..  ...  ..
.  ...
95  1  0    0    1    0    0    0    1    1    1    0    0  2    1
0    1
96  1  0    1    0    1    0    0    0    0    1    1    0  6    0
0    0
97  1  0    0    1    0    0    1    1    1    1    0    0  4    1
0    1
98  0  0    1    0    0    0    0    0    0    1    0    0  0    0
0    0
99  0  1    1    0    1    0    0    0    1    1    0    0  2    1
0    0

[100 rows x 16 columns]
```

```
x = np.array(x)
```

```
x
```

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

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

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

```
X_train,X_test,Y_train,Y_test = train_test_split(x,y,test_size=0.3)
```

```
X_train
```

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

```
X_test
```

```
array([[1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 6, 0, 1, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 6, 0, 0, 0],
       [1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 4, 1, 0, 1],
       [0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1],
       [0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 4, 1, 0, 1],
       [0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 5, 0, 0, 0],
       [0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 2, 1, 0, 1],
       [0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 2, 1, 0, 1],
       [1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 4, 1, 0, 1],
       [1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 4, 1, 0, 0],
       [1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 4, 1, 0, 1],
       [0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 2, 0, 0, 1],
       [0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 2, 1, 1, 0],
       [0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0],
       [1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 4, 1, 0, 1],
       [1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 4, 1, 0, 1],
       [0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 2, 1, 1, 0],
       [1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 4, 1, 0, 1],
       [1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 4, 1, 0, 0],
       [0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0],
       [1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 2, 1, 0, 1],
       [0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 2, 1, 0, 1],
       [0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0],
       [1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 4, 0, 1, 0],
       [1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 2, 1, 0, 1],
       [0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 4, 1, 0, 1],
```

```
       [0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0],
       [0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 2, 1, 0, 0]])
```

```
Y_train
```

```
array([2, 4, 1, 2, 1, 3, 1, 1, 7, 1, 4, 7, 5, 1, 3, 7, 4, 1, 2, 1, 1,
4,
       2, 2, 1, 2, 6, 2, 6, 1, 4, 1, 7, 1, 2, 7, 7, 1, 6, 7, 7, 1, 4,
4,
       1, 2, 1, 4, 6, 1, 1, 6, 7, 1, 1, 2, 1, 5, 4, 1, 5, 5, 2, 2, 2,
1,
       1, 6, 1, 2])
```

```
Y_test
```

```
array([6, 6, 1, 4, 1, 1, 7, 2, 2, 1, 1, 1, 1, 2, 4, 1, 1, 2, 1, 1, 4,
1,
       2, 3, 1, 1, 4, 3, 3, 2])
```

```
inst = DecisionTreeClassifier()

inst = inst.fit(X_train,Y_train)

Y_pred = inst.predict(X_test)
```

```
Y_pred
```

```
array([6, 6, 1, 4, 1, 1, 7, 2, 2, 1, 1, 1, 1, 2, 4, 1, 1, 2, 1, 1, 4,
1,
       2, 5, 1, 1, 4, 3, 3, 2])
```

```
from sklearn import metrics
from sklearn.metrics import precision_recall_fscore_support

acc = metrics.accuracy_score(Y_test,Y_pred)
print(acc)
```

```
0.9666666666666667
```

```
print("Precision,recall,F score
is",precision_recall_fscore_support(Y_test,Y_pred,average='macro'))
```

```
Precision,recall,F score is (0.8571428571428571, 0.8095238095238094,
0.8285714285714285, None)
```

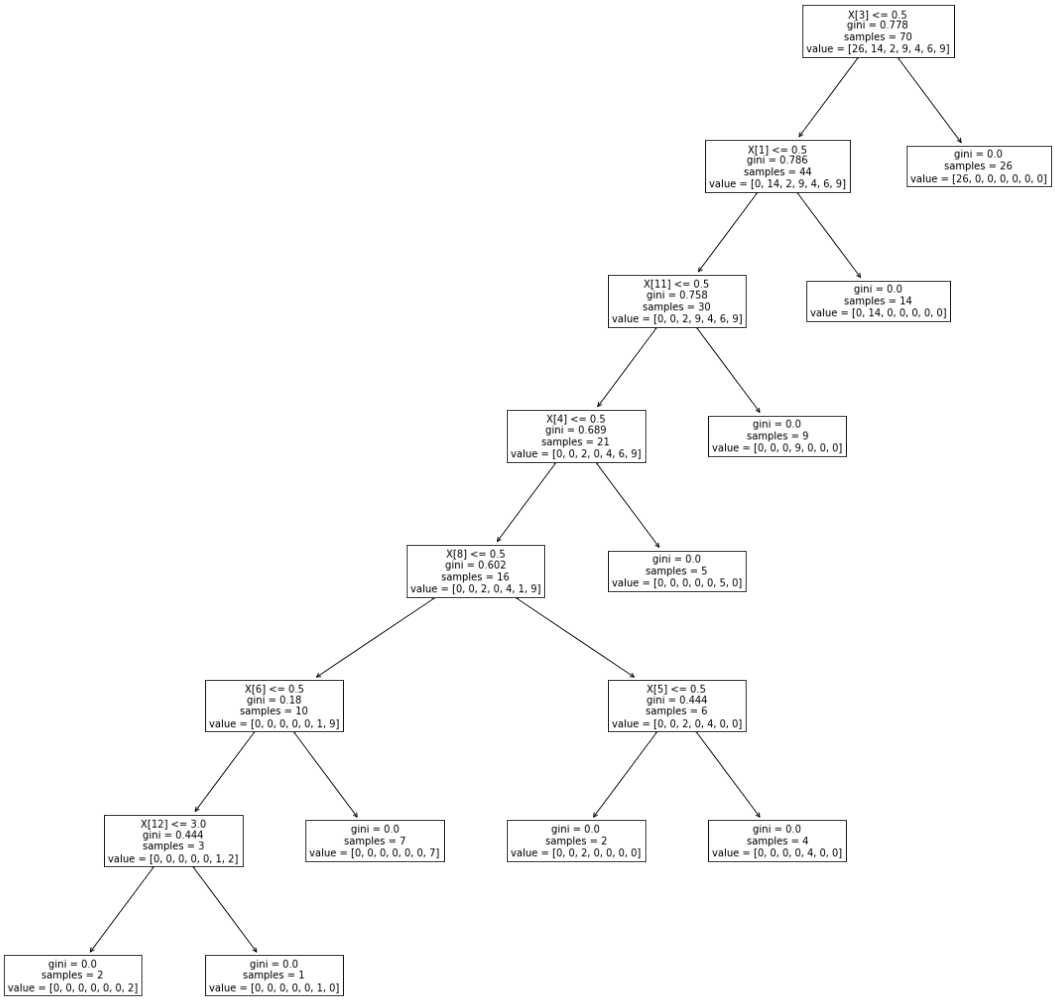
```
/home/ignis/anaconda3/lib/python3.7/site-packages/sklearn/metrics/
_classification.py:1272: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
import graphviz
```

```
import matplotlib.pyplot as plt
from sklearn import tree


plt.figure(figsize=(20,20))
tree.plot_tree(inst,fontsize=10)
```

```
[Text(913.0909090909091, 1019.25, 'X[3] <= 0.5\ngini = 0.778\nsamples
= 70\nvalue = [26, 14, 2, 9, 4, 6, 9]'),
 Text(811.6363636363636, 883.35, 'X[1] <= 0.5\ngini = 0.786\nsamples =
44\nvalue = [0, 14, 2, 9, 4, 6, 9]'),
 Text(710.1818181818181, 747.45, 'X[11] <= 0.5\ngini = 0.758\nsamples
= 30\nvalue = [0, 0, 2, 9, 4, 6, 9]'),
 Text(608.7272727272727, 611.55, 'X[4] <= 0.5\ngini = 0.689\nsamples =
21\nvalue = [0, 0, 2, 0, 4, 6, 9]'),
 Text(507.27272727272725, 475.65, 'X[8] <= 0.5\ngini = 0.602\nsamples
= 16\nvalue = [0, 0, 2, 0, 4, 1, 9]'),
 Text(304.3636363636364, 339.75, 'X[6] <= 0.5\ngini = 0.18\nsamples =
10\nvalue = [0, 0, 0, 0, 0, 1, 9]'),
 Text(202.9090909090909, 203.85000000000002, 'X[12] <= 3.0\ngini =
0.444\nsamples = 3\nvalue = [0, 0, 0, 0, 0, 1, 2]'),
 Text(101.45454545454545, 67.95000000000005, 'gini = 0.0\nsamples = 2\
nvalue = [0, 0, 0, 0, 0, 0, 2]'),
 Text(304.3636363636364, 67.95000000000005, 'gini = 0.0\nsamples = 1\
nvalue = [0, 0, 0, 0, 0, 1, 0]'),
 Text(405.8181818181818, 203.85000000000002, 'gini = 0.0\nsamples = 7\
nvalue = [0, 0, 0, 0, 0, 0, 7]'),
 Text(710.1818181818181, 339.75, 'X[5] <= 0.5\ngini = 0.444\nsamples =
6\nvalue = [0, 0, 2, 0, 4, 0, 0]'),
 Text(608.7272727272727, 203.85000000000002, 'gini = 0.0\nsamples = 2\
nvalue = [0, 0, 2, 0, 0, 0, 0]'),
 Text(811.6363636363636, 203.85000000000002, 'gini = 0.0\nsamples = 4\
nvalue = [0, 0, 0, 0, 4, 0, 0]'),
 Text(710.1818181818181, 475.65, 'gini = 0.0\nsamples = 5\nvalue = [0,
0, 0, 0, 0, 5, 0]'),
 Text(811.6363636363636, 611.55, 'gini = 0.0\nsamples = 9\nvalue = [0,
0, 0, 9, 0, 0, 0]'),
 Text(913.0909090909091, 747.45, 'gini = 0.0\nsamples = 14\nvalue =
[0, 14, 0, 0, 0, 0, 0]'),
 Text(1014.5454545454545, 883.35, 'gini = 0.0\nsamples = 26\nvalue =
[26, 0, 0, 0, 0, 0, 0]')]
```




```
from sklearn.metrics import confusion_matrix

confusion_matrix(Y_test,Y_pred)

array([[14,  0,  0,  0,  0,  0,  0],
       [ 0,  6,  0,  0,  0,  0,  0],
       [ 0,  0,  2,  0,  1,  0,  0],
       [ 0,  0,  0,  4,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  2,  0],
       [ 0,  0,  0,  0,  0,  0,  1]])

from sklearn.metrics import classification_report

print(classification_report(Y_test, Y_pred))

              precision    recall  f1-score   support
```

```
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00         6
           3       1.00      0.67      0.80         3
           4       1.00      1.00      1.00         4
           5       0.00      0.00      0.00         0
           6       1.00      1.00      1.00         2
           7       1.00      1.00      1.00         1

    accuracy                           0.97        30
   macro avg       0.86      0.81      0.83        30
weighted avg       1.00      0.97      0.98        30


/home/ignis/anaconda3/lib/python3.7/site-packages/sklearn/metrics/
_classification.py:1272: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
dot_data = tree.export_graphviz(inst,
                    out_file="tree.dot",
                    filled = True)
```