1) Program to load the data set.

```
#PROGRAM TO LOAD DATASET
print('PROGRAM TO LOAD DATASET')
from sklearn import datasets
import pandas as pd
iris=datasets.load_iris()

dir(iris)
print(iris.DESCR)
```

output

```
>>>
PROGRAM TO LOAD DATASET
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
```

```
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
>>>
```

2) PROGRAM TO LOAD DATASET AND DISPLAY 6-Samples

```
#PROGRAM TO LOAD DATASET AND DISPLAY 6-Samples
print('PROGRAM TO LOAD DATASET AND DISPLAY 6-SAMPLES')
from sklearn import datasets
import pandas as pd
iris=datasets.load_iris()

dir(iris)
print(iris.DESCR)

df=pd.DataFrame(
    iris.data,
    columns=iris.feature_names
    )
```

```
print(df)

df['target']=pd.Series(
   iris.target
   )
print(df)

pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
#to print only 6-samples of Data set
print('To display 6-Samples of a Dataset')
xyz=df.sample(n=6)

print(xyz)
print('To display sample and its corresponding classes')
x=df['target_names']=df['target'].apply(lambda y:iris.target_names[y])
print(x)
```

Output:

```
PROGRAM TO LOAD DATASET AND DISPLAY 6-SAMPLES
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

   :Number of Instances: 150 (50 in each of three classes)
   :Number of Attributes: 4 numeric, predictive attributes and the class
   :Attribute Information:
      - sepal length in cm
      - sepal width in cm
      - petal length in cm
      - petal width in cm
      - class:
            - Iris-Setosa
            - Iris-Versicolour
            - Iris-Virginica

   :Summary Statistics:

   ============== ==== ==== ======= ===== ====================
                  Min  Max   Mean    SD   Class Correlation
   ============== ==== ==== ======= ===== ====================
   sepal length:   4.3  7.9   5.84   0.83    0.7826
   sepal width:    2.0  4.4   3.05   0.43   -0.4194
   petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
```

petal width: 0.1 2.5   1.20   0.76   0.9565 (high!)
============== ==== ==== ======= ===== ====================

:Missing Attribute Values: None
:Class Distribution: 33.3% for each of 3 classes.
:Creator: R.A. Fisher
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
:Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken from Fisher's paper. Note that it's the same as in R, but not as in the UCI Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

.. topic:: References

- Fisher, R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to Mathematical Statistics" (John Wiley, NY, 1950).
- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis. (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.
- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System Structure and Classification Rule for Recognition in Partially Exposed Environments". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 1, 67-71.
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions on Information Theory, May 1972, 431-433.
- See also: 1988 MLC Proceedings, 54-64. Cheeseman et al"s AUTOCLASS II conceptual clustering system finds 3 classes in the data.
- Many, many more ...

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
..	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

```
[150 rows x 4 columns]
    sepal length (cm)  sepal width (cm)  ...  petal width (cm)  target
0                  5.1               3.5  ...               0.2       0
1                  4.9               3.0  ...               0.2       0
2                  4.7               3.2  ...               0.2       0
3                  4.6               3.1  ...               0.2       0
4                  5.0               3.6  ...               0.2       0
..                 ...               ...  ...               ...     ...
145                6.7               3.0  ...               2.3       2
146                6.3               2.5  ...               1.9       2
147                6.5               3.0  ...               2.0       2
148                6.2               3.4  ...               2.3       2
149                5.9               3.0  ...               1.8       2

[150 rows x 5 columns]
To display 6-Samples of a Dataset
    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
38                 4.4               3.0                1.3               0.2
101                5.8               2.7                5.1               1.9
88                 5.6               3.0                4.1               1.3
49                 5.0               3.3                1.4               0.2
20                 5.4               3.4                1.7               0.2
86                 6.7               3.1                4.7               1.5

    target
38       0
101      2
88       1
49       0
20       0
86       1
>>>
```

3) PROGRAM TO SHOW TRAIN TEST SPLIT

```
#PROGRAM TO SHOW TRAIN TEST SPLIT

print('PROGRAM TO SHOW TRAIN TEST SPLIT')

from sklearn import datasets

import pandas as pd

iris=datasets.load_iris()
```

```
df=pd.DataFrame(
  iris.data,
  columns=iris.feature_names
  )


from sklearn.model_selection import train_test_split
df_train,df_test=train_test_split(df,test_size=0.3)

print('Number of Training samples')
print(df_train.shape[0])

print('Number of Testing samples')
print(df_test.shape[0])
```

Output

PROGRAM TO SHOW TRAIN TEST SPLIT

Number of Training samples

105

Number of Testing samples

45

4) PROGRAM TO CONSTRUCT A TREE AND DISPLAY ITS ARCHITECHTURE

```
#PROGRAM TO CONSTRUCT A TREE AND DISPLAY IT
print('PROGRAM TO CONSTRUCT A TREE AND DISPLAY IT')
from sklearn import datasets
import pandas as pd
iris=datasets.load_iris()

# df will fold dataset as a table
```

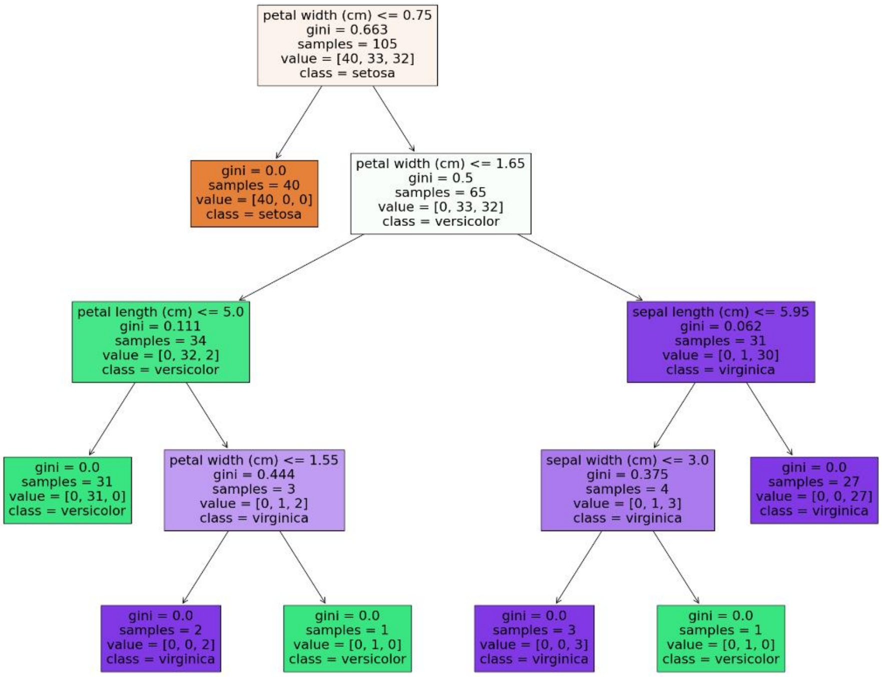
```
df=pd.DataFrame(
  iris.data,
  columns=iris.feature_names
  )
#labels are assigned to df[target] table or array
df['target']=pd.Series(
   iris.target
  )

from sklearn.model_selection import train_test_split
# Train Test Split Ratio
df_train,df_test=train_test_split(df,test_size=0.3)

df['target_names']=df['target'].apply(lambda y:iris.target_names[y])

print('Number of Training samples')
print(df_train.shape[0])

print('Number of Testing samples')
print(df_test.shape[0])

#Importing Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier()

x_train=df_train[iris.feature_names]
x_test=df_test[iris.feature_names]

y_train=df_train['target']
y_test=df_test['target']
```

```
#Training Decision Tree Classifier
clf.fit(x_train,y_train)

#Testing the data
y_test_pred=clf.predict(x_test)

print('Class of Testing Samples')
print(y_test_pred)

#To display the decision tree in command shell
from sklearn.tree import export_text
from sklearn import tree
from matplotlib import pyplot as plt

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

#print(
#         export_text(clf,feature_names=iris.feature_names,spacing=3,decimals=1)
#         )

#To display the decision tree as a tree  in .png file
with open("decistion_tree.log", "w") as fout:
    fout.write(text_representation)

fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clf,
             feature_names=iris.feature_names,
             class_names=iris.target_names,
           filled=True)
```

fig.savefig("decistion_tree.png")

Output

```
PROGRAM TO CONSTRUCT A TREE AND DISPLAY IT
Number of Training samples
105
Number of Testing samples
45
Class of Testing Samples
[0 1 0 2 0 0 0 2 1 2 2 2 2 2 1 2 0 1 1 2 2 1 1 2 2 1 1 1 1 1 2 2 1 1 1 1 2
 1 0 2 0 0 1 2 0]
|--- feature_3 <= 0.75
|   |--- class: 0
|--- feature_3 >  0.75
|   |--- feature_3 <= 1.65
|   |   |--- feature_2 <= 5.00
|   |   |   |--- class: 1
|   |   |--- feature_2 >  5.00
|   |   |   |--- feature_3 <= 1.55
|   |   |   |   |--- class: 2
|   |   |   |--- feature_3 >  1.55
|   |   |   |   |--- class: 1
|   |--- feature_3 >  1.65
|   |   |--- feature_0 <= 5.95
|   |   |   |--- feature_1 <= 3.00
|   |   |   |   |--- class: 2
|   |   |   |--- feature_1 >  3.00
|   |   |   |   |--- class: 1
|   |   |--- feature_0 >  5.95
|   |   |   |--- class: 2
```



5) Program to evaluate the predicted output

```
#PROGRAM TO CONSTRUCT A TREE AND EVALUATE THE PREDICTOR
print('PROGRAM TO CONSTRUCT A TREE AND EVALUATE THE PREDICTOR')

from sklearn import datasets
import pandas as pd
iris=datasets.load_iris()

# df will fold dataset as a table
df=pd.DataFrame(
  iris.data,
  columns=iris.feature_names
  )
```

```
#labels are assigned to df[target] table or array
df['target']=pd.Series(
    iris.target
   )

from sklearn.model_selection import train_test_split
# Train Test Split Ratio
df_train,df_test=train_test_split(df,test_size=0.3)

df['target_names']=df['target'].apply(lambda y:iris.target_names[y])

print('Number of Training samples')
print(df_train.shape[0])

print('Number of Testing samples')
print(df_test.shape[0])

#Importing Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier()

x_train=df_train[iris.feature_names]
x_test=df_test[iris.feature_names]

y_train=df_train['target']
y_test=df_test['target']

#Training Decision Tree Classifier
clf.fit(x_train,y_train)

#Testing the data
```

```
y_test_pred=clf.predict(x_test)

print('Class of Testing Samples')
print(y_test_pred)

#To display the decision tree in command shell
from sklearn.tree import export_text
from sklearn import tree
from matplotlib import pyplot as plt

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

#print(
#          export_text(clf,feature_names=iris.feature_names,spacing=3,decimals=1)
#          )

#To display the decision tree as a tree  in .png file
with open("decistion_tree.log", "w") as fout:
   fout.write(text_representation)

fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clf,
             feature_names=iris.feature_names,
             class_names=iris.target_names,
          filled=True)

fig.savefig("decistion_tree.png")



from sklearn.metrics import accuracy_score
```

x=accuracy_score(y_test,y_test_pred)

print('Accuracy')

print(x)

Output

```
PROGRAM TO CONSTRUCT A TREE AND EVALUATE THE PREDICTOR
Number of Training samples
105
Number of Testing samples
45
Class of Testing Samples
[0 2 1 2 2 1 1 2 1 1 1 0 2 2 1 2 1 1 2 1 2 2 2 2 1 1 0 2 1 2 0 1 2 1 0 2 0
 2 1 0 0 0 2 2 2]
|--- feature_2 <= 2.60
|   |--- class: 0
|--- feature_2 >  2.60
|   |--- feature_2 <= 4.95
|   |   |--- feature_3 <= 1.65
|   |   |   |--- class: 1
|   |   |--- feature_3 >  1.65
|   |   |   |--- feature_1 <= 3.00
|   |   |   |   |--- class: 2
|   |   |   |--- feature_1 >  3.00
|   |   |   |   |--- class: 1
|   |--- feature_2 >  4.95
|   |   |--- feature_2 <= 5.05
|   |   |   |--- feature_1 <= 2.75
|   |   |   |   |--- class: 2
|   |   |   |--- feature_1 >  2.75
|   |   |   |   |--- class: 1
|   |   |--- feature_2 >  5.05
```

```
|   |   |   |--- class: 2
```

Accuracy

0.9777777777777777

6) **PROGRAM TO SHOW IMPORTANT FEATURES IN A DECISION TREE**

```
#PROGRAM TO SHOW IMPORTANT FEATURES IN A DECISION TREE
print('PROGRAM TO SHOW IMPORTANT FEATURES IN A DECISION TREE')

from sklearn import datasets

import pandas as pd
iris=datasets.load_iris()

# df will fold dataset as a table
df=pd.DataFrame(
  iris.data,
  columns=iris.feature_names
  )
#labels are assigned to df[target] table or array
df['target']=pd.Series(
   iris.target
  )

from sklearn.model_selection import train_test_split

# Train Test Split Ratio
df_train,df_test=train_test_split(df,test_size=0.3)

df['target_names']=df['target'].apply(lambda y:iris.target_names[y])
```

```
print('Number of Training samples')
print(df_train.shape[0])

print('Number of Testing samples')
print(df_test.shape[0])

#Importing Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier()

x_train=df_train[iris.feature_names]
x_test=df_test[iris.feature_names]

y_train=df_train['target']
y_test=df_test['target']

#Training Decision Tree Classifier
clf.fit(x_train,y_train)

#Testing the data
y_test_pred=clf.predict(x_test)

print('Class of Testing Samples')
print(y_test_pred)

#To display the decision tree in command shell
from sklearn.tree import export_text
from sklearn import tree
from matplotlib import pyplot as plt

text_representation = tree.export_text(clf)
```

```
print(text_representation)

#print(
#           export_text(clf,feature_names=iris.feature_names,spacing=3,decimals=1)
#           )

#To display the decision tree as a tree  in .png file
with open("decistion_tree.log", "w") as fout:
   fout.write(text_representation)

fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clf,
            feature_names=iris.feature_names,
            class_names=iris.target_names,
          filled=True)

fig.savefig("decistion_tree.png")

## To show important features
print('\nTo show Important Features in a Decision Tree')
x=pd.DataFrame(
   {
      'feature_names':iris.feature_names,
      'feature_importances':clf.feature_importances_
      }
   ).sort_values(
      'feature_importances',ascending=False
      ).set_index('feature_names')
print(x)
```

Output

PROGRAM TO SHOW IMPORTANT FEATURES IN A DECISION TREE

Number of Training samples

105

Number of Testing samples

45

Class of Testing Samples

```
[0 0 1 1 1 0 1 0 1 0 1 0 2 2 2 0 2 1 0 0 1 1 1 2 0 1 0 2 1 0 2 1 2 1 0 0 1
 2 0 2 1 2 0 0 0]
|--- feature_2 <= 2.60
|   |--- class: 0
|--- feature_2 >  2.60
|   |--- feature_3 <= 1.75
|   |   |--- feature_2 <= 4.95
|   |   |   |--- feature_3 <= 1.60
|   |   |   |   |--- class: 1
|   |   |   |--- feature_3 >  1.60
|   |   |   |   |--- class: 2
|   |   |--- feature_2 >  4.95
|   |   |   |--- feature_3 <= 1.55
|   |   |   |   |--- class: 2
|   |   |   |--- feature_3 >  1.55
|   |   |   |   |--- feature_0 <= 6.95
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_0 >  6.95
|   |   |   |   |   |--- class: 2
|   |--- feature_3 >  1.75
|   |   |--- class: 2
```

To show Important Features in a Decision Tree

	feature_importances
feature_names	
petal length (cm)	0.539270
petal width (cm)	0.441583
sepal length (cm)	0.019147
sepal width (cm)	0.000000

7) PROGRAM TO DISPLAY INTERNAL DECISION TREES WITH FEATURE NAMES DECIMAL PLACES AND SPACING

```
#PROGRAM TO DISPLAY INTERNAL DECISION TREES WITH FEATURE NAMES DECIMAL PLACES AND SPACING
print('PROGRAM TO DISPLAY INTERNAL DECISION TREES WITH FEATURE NAMES DECIMAL PLACES AND SPACING')

from sklearn import datasets
import pandas as pd
iris=datasets.load_iris()

# df will fold dataset as a table
df=pd.DataFrame(
  iris.data,
  columns=iris.feature_names
  )
#labels are assigned to df[target] table or array
df['target']=pd.Series(
   iris.target
  )

from sklearn.model_selection import train_test_split
# Train Test Split Ratio
```

```
df_train,df_test=train_test_split(df,test_size=0.3)

df['target_names']=df['target'].apply(lambda y:iris.target_names[y])

print('Number of Training samples')
print(df_train.shape[0])

print('Number of Testing samples')
print(df_test.shape[0])

#Importing Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier()

x_train=df_train[iris.feature_names]
x_test=df_test[iris.feature_names]

y_train=df_train['target']
y_test=df_test['target']

#Training Decision Tree Classifier
clf.fit(x_train,y_train)

#Testing the data
y_test_pred=clf.predict(x_test)

print('Class of Testing Samples')
print(y_test_pred)

#To display the decision tree in command shell
from sklearn.tree import export_text
```

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

text_representation = tree.export_text(clf)


#To display the decision tree as a tree  in .png file
with open("decistion_tree.log", "w") as fout:
   fout.write(text_representation)

fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clf,
           feature_names=iris.feature_names,
           class_names=iris.target_names,
         filled=True)

fig.savefig("decistion_tree.png")

x=pd.DataFrame(
   {
      'feature_names':iris.feature_names,
      'feature_importances':clf.feature_importances_
      }
   ).sort_values(
      'feature_importances',ascending=False
      ).set_index('feature_names')

print(x)

print('\n\nDecision Tree with Feature Names and with three decimal places')
print(
```

```
        export_text(clf,feature_names=iris.feature_names,spacing=3,decimals=3)
        )

print('\n Following is with decimal two spaces')
print(tree.export_text(clf))
```

Output

PROGRAM TO DISPLAY INTERNAL DECISION TREES WITH FEATURE NAMES DECIMAL PLACES AND SPACING

Number of Training samples

105

Number of Testing samples

45

Class of Testing Samples

[0 2 0 0 0 2 2 2 1 1 2 0 2 0 1 2 1 2 1 1 1 0 2 2 0 2 0 2 2 1 0 0 0 2 1 1 1

0 0 0 1 2 1 1 1]

```
                    feature_importances
feature_names
petal length (cm)            0.937983
sepal length (cm)            0.048571
petal width (cm)             0.013445
sepal width (cm)             0.000000
```

Decision Tree with Feature Names and with three decimal places

```
|--- petal length (cm) <= 2.600
|   |--- class: 0
|--- petal length (cm) >  2.600
|   |--- petal length (cm) <= 4.850
|   |   |--- petal width (cm) <= 1.700
|   |   |   |--- class: 1
|   |   |--- petal width (cm) >  1.700
```

```
|   |   |   |--- sepal length (cm) <= 6.050
|   |   |   |   |--- class: 1
|   |   |   |--- sepal length (cm) >  6.050
|   |   |   |   |--- class: 2
|   |--- petal length (cm) >  4.850
|   |   |--- petal length (cm) <= 5.050
|   |   |   |--- sepal length (cm) <= 6.500
|   |   |   |   |--- class: 2
|   |   |   |--- sepal length (cm) >  6.500
|   |   |   |   |--- class: 1
|   |   |--- petal length (cm) >  5.050
|   |   |   |--- class: 2
```

Following is with decimal two spaces

```
|--- feature_2 <= 2.60
|   |--- class: 0
|--- feature_2 >  2.60
|   |--- feature_2 <= 4.85
|   |   |--- feature_3 <= 1.70
|   |   |   |--- class: 1
|   |   |--- feature_3 >  1.70
|   |   |   |--- feature_0 <= 6.05
|   |   |   |   |--- class: 1
|   |   |   |--- feature_0 >  6.05
|   |   |   |   |--- class: 2
|   |--- feature_2 >  4.85
|   |   |--- feature_2 <= 5.05
|   |   |   |--- feature_0 <= 6.50
|   |   |   |   |--- class: 2
|   |   |   |--- feature_0 >  6.50
```

```
|   |   |   |   |--- class: 1
|   |   |--- feature_2 >  5.05
|   |   |   |--- class: 2

>>>
```

8) **PROGRAM TO GET SCORE FROM 100 ITERATIONS**

```
#PROGRAM TO GET SCORE FROM 100 ITERATIONS
print('PROGRAM TO GET SCORE FROM 100 ITERATIONS')

from sklearn import datasets
import pandas as pd
iris=datasets.load_iris()

# df will fold dataset as a table
df=pd.DataFrame(
   iris.data,
   columns=iris.feature_names
   )
#labels are assigned to df[target] table or array
df['target']=pd.Series(
    iris.target
   )

from sklearn.model_selection import train_test_split

df_train,df_test=train_test_split(df,test_size=0.3)



from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.metrics import accuracy_score



accuracy_scores=[]
for _ in range(5):
    # At each iteration there is a fresh split

    df_train,df_test=train_test_split(df,test_size=0.3)

    x_train=df_train[iris.feature_names]
    x_test=df_test[iris.feature_names]
    y_train=df_train['target']
    y_test=df_test['target']
    clf=DecisionTreeClassifier()

    clf.fit(x_train,y_train)
    y_pred=clf.predict(x_test)
    accuracy_scores.append(round(accuracy_score(y_test,y_pred),3))

accuracy_scores=pd.Series(accuracy_scores)
print('Accuracy Scores for 100 iterations are as follows')
print(accuracy_scores)
```

Output

PROGRAM TO GET SCORE FROM 100 ITERATIONS

Accuracy Scores for 100 iterations are as follows

```
0    0.956
1    0.978
2    0.933
3    0.978
```

```
4    0.978
     ...
95   0.956
96   0.911
97   0.956
98   0.933
99   0.956
Length: 100, dtype: float64
```

9) PROGRAM TO SHOW SHUFFLE SPLIT

```
#PROGRAM TO SHOW SHUFFLE SPLIT
print('PROGRAM TO show shuffle split')

from sklearn import datasets
import pandas as pd

iris=datasets.load_iris()

# df will fold dataset as a table
df=pd.DataFrame(
 iris.data,
 columns=iris.feature_names
  )
#labels are assigned to df[target] table or array
df['target']=pd.Series(
   iris.target
  )

from sklearn.model_selection import train_test_split
```

```
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score
from sklearn.model_selection import ShuffleSplit

accuracy_scores=[]

rs=ShuffleSplit(n_splits=100,test_size=0.3)

for train_index,test_index in rs.split(df):
    x_train=df.loc[train_index,iris.feature_names]
    x_test=df.loc[test_index,iris.feature_names]

    y_train=df.loc[train_index,'target']
    y_test=df.loc[test_index,'target']

    clf=DecisionTreeClassifier()

    clf.fit(x_train,y_train)
    y_pred=clf.predict(x_test)

    accuracy_scores.append(round(accuracy_score(y_test,y_pred),3))

accuracy_scores=pd.Series(accuracy_scores)
print('Accuracy Scores for 100 iterations are as follows')
print(accuracy_scores)
```

Output

PROGRAM TO show shuffle split

Accuracy Scores for 100 iterations are as follows

```
0     0.956
1     0.933
2     1.000
3     0.933
4     0.933
      ...
95    0.978
96    0.933
97    0.933
98    0.933
99    0.933
Length: 100, dtype: float64
```