DECISION TREE (CLASSIFICATION) ON THE IRIS FLOWER DATASET

USING PYTHON

DR. ALVIN ANG

CONTENTS

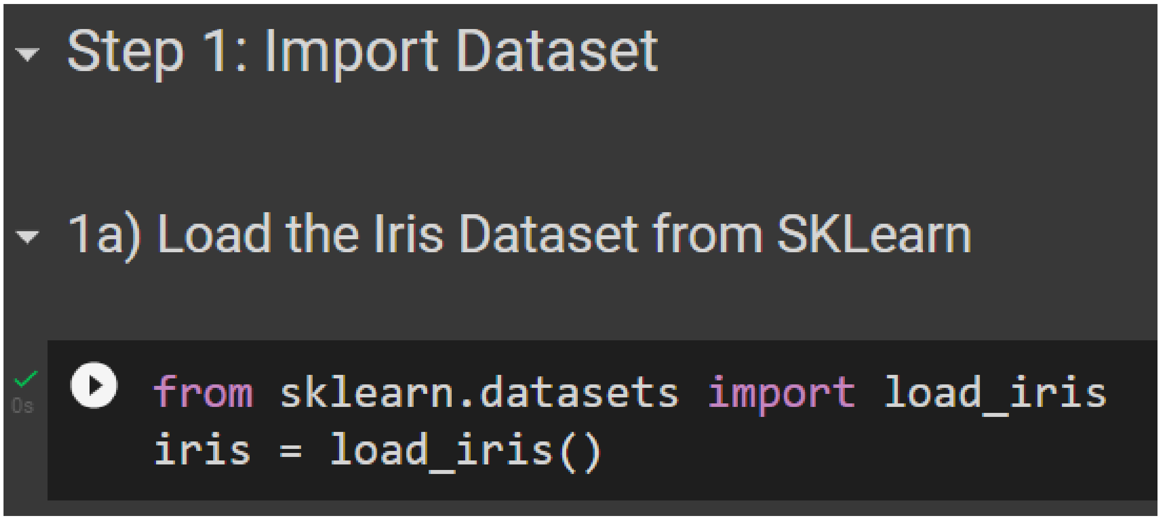
I. STEP 1: IMPORT DATASET

https://www.alvinang.sg/s/Decision-Tree-Classification-using-WEKA-by-Dr-Alvin-Ang.pdf

https://medium.com/x8-the-ai-community/decision-tree-visualisation-quick-ml-tutorial-for-beginners-6d1028542c31

https://www.alvinang.sg/s/Decision_Tree_Classification_on_the_Iris_Flower_Dataset_using_Python_by_Dr_Alvin_Ang.ipynb

A. LOAD THE IRIS DATASET FROM SK LEARN

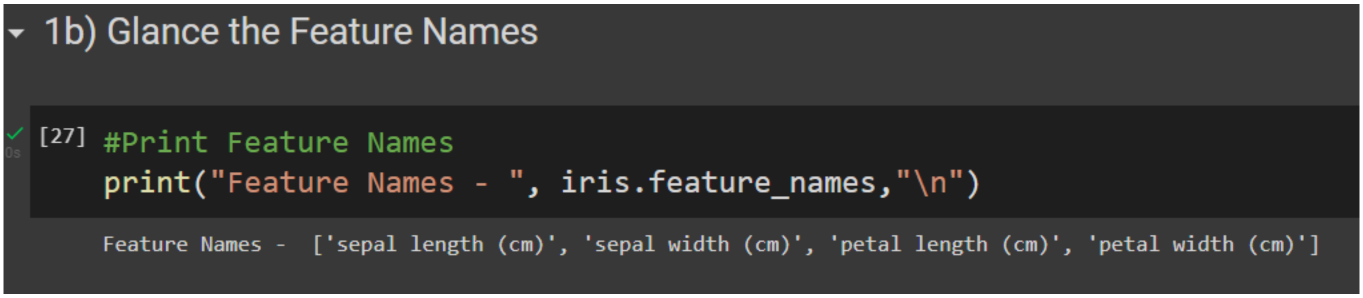


Step 1: Import Dataset

1a) Load the Iris Dataset from SKLearn

```
from sklearn.datasets import load_iris
iris = load_iris()
```

B. GLANCE THE FEATURE NAMES

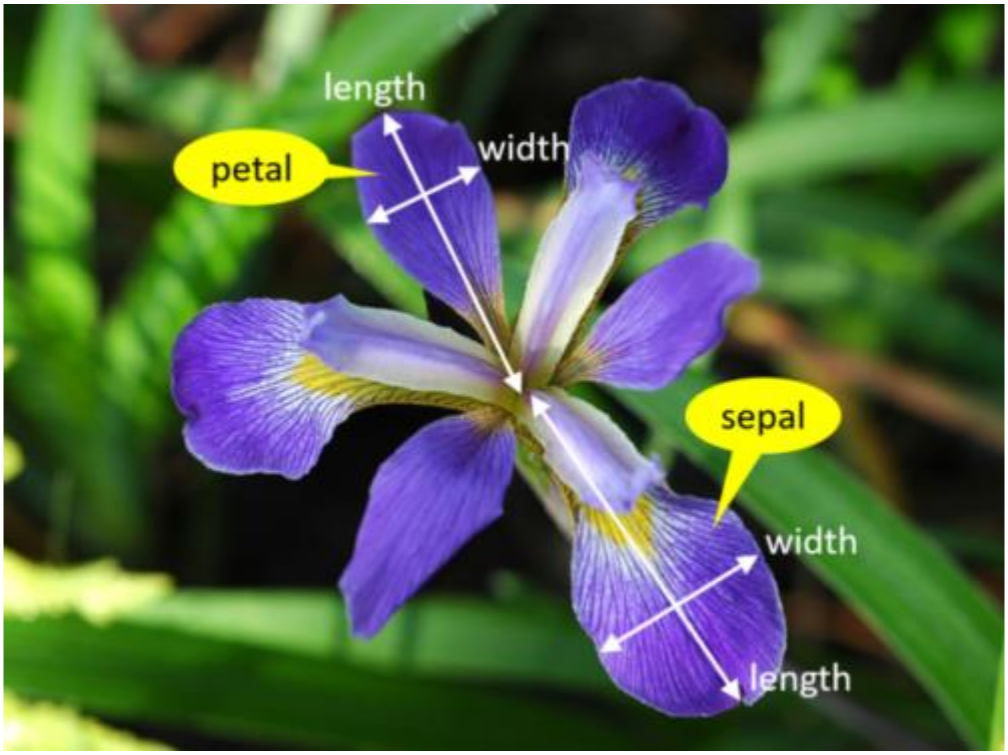


1b) Glance the Feature Names

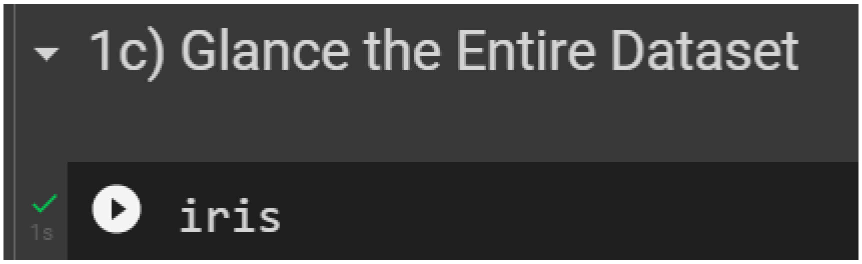
[27]

```
#Print Feature Names
print("Feature Names - ", iris.feature_names,"\n")
```

```
Feature Names -  ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

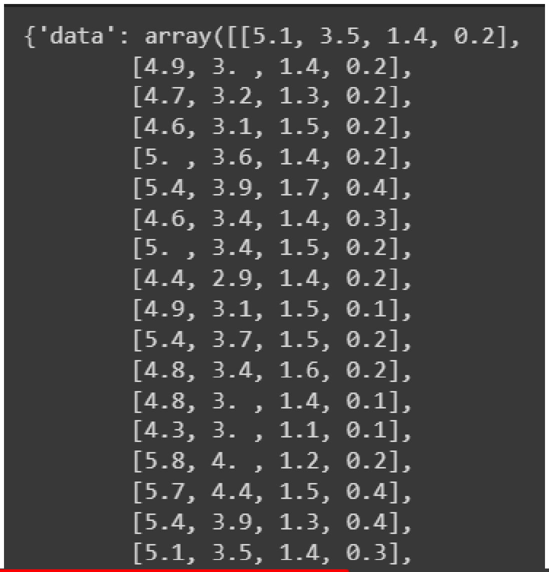


C. GLANCE THE ENTIRE DATASET

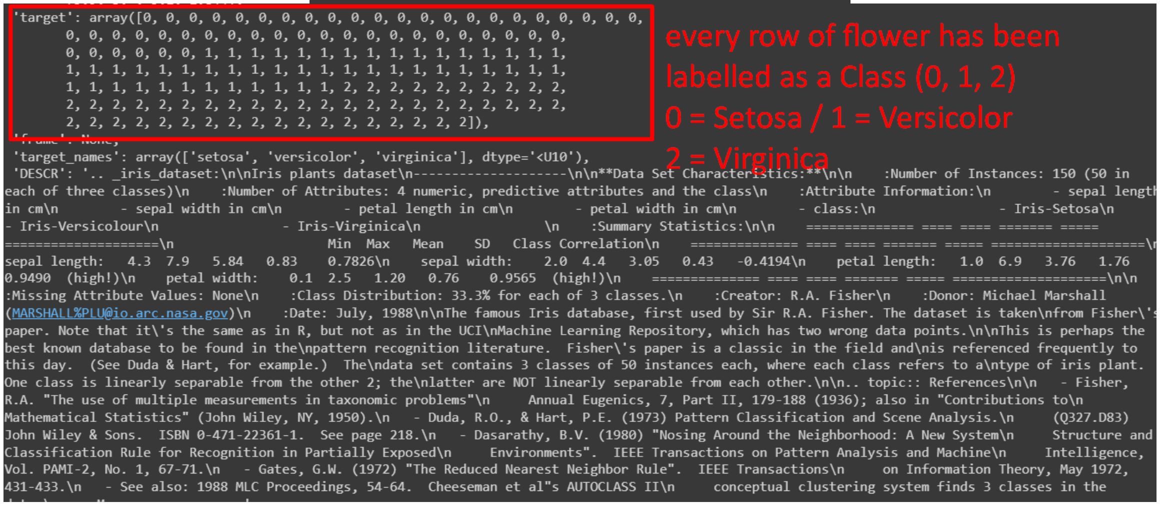


1c) Glance the Entire Dataset

```
iris
```



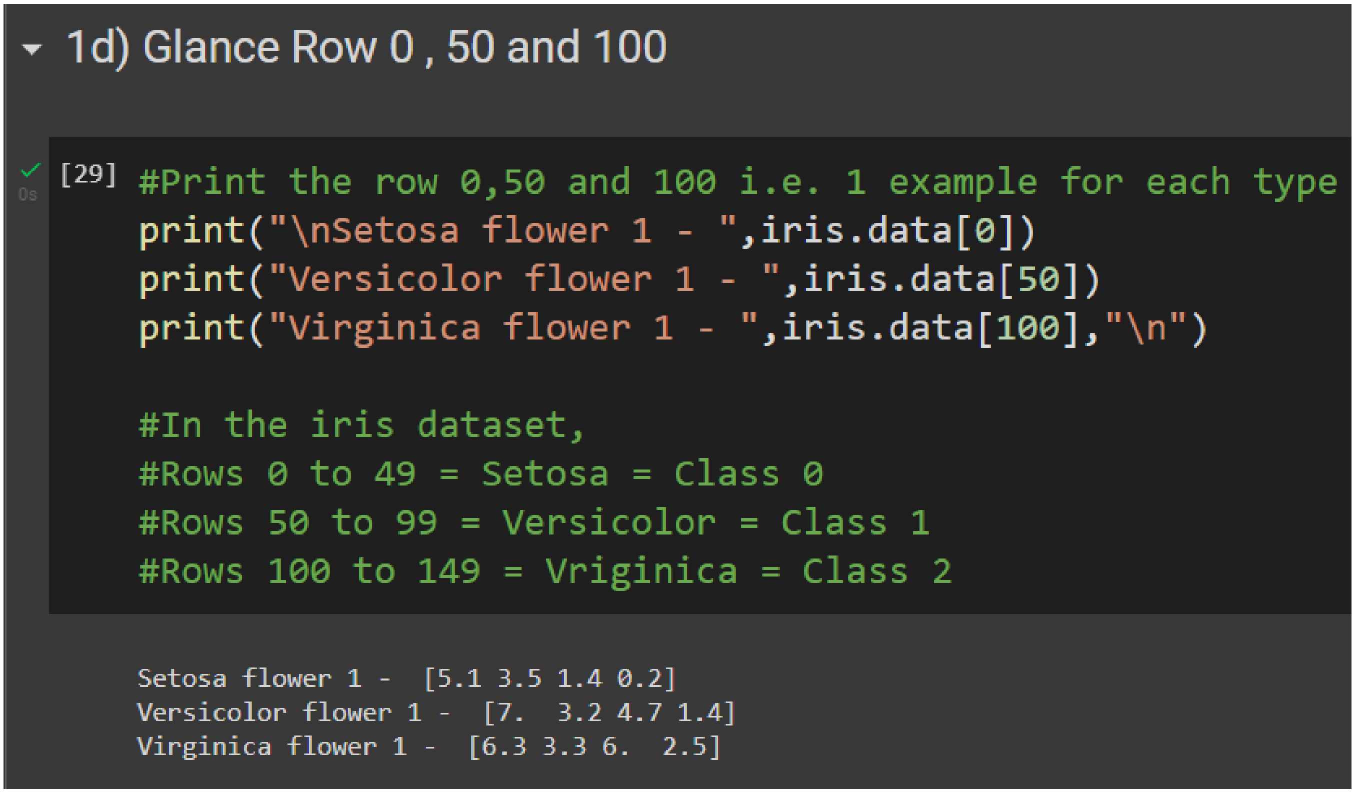
```
{'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5. , 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.4, 3.7, 1.5, 0.2],
        [4.8, 3.4, 1.6, 0.2],
        [4.8, 3. , 1.4, 0.1],
        [4.3, 3. , 1.1, 0.1],
        [5.8, 4. , 1.2, 0.2],
        [5.7, 4.4, 1.5, 0.4],
        [5.4, 3.9, 1.3, 0.4],
        [5.1, 3.5, 1.4, 0.3],
```



```
'target': array([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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
 'frame': None,
 'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
 'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n--------------------\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 150 (50 in
each of three classes)\n    :Number of Attributes: 4 numeric, predictive attributes and the class\n    :Attribute Information:\n        - sepal length
in cm\n        - sepal width in cm\n        - petal length in cm\n        - petal width in cm\n        - class:\n                - Iris-Setosa\n
- Iris-Versicolour\n                - Iris-Virginica\n                \n    :Summary Statistics:\n\n    ============== ==== ==== ======= =====
====================\n                    Min  Max   Mean    SD   Class Correlation\n    ============== ==== ==== ======= ===== ====================\
sepal length:   4.3  7.9   5.84   0.83    0.7826\n    sepal width:    2.0  4.4   3.05   0.43   -0.4194\n    petal length:   1.0  6.9   3.76   1.76
0.9490  (high!)\n    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)\n    ============== ==== ==== ======= ===== ====================\n\n
:Missing Attribute Values: None\n    :Class Distribution: 33.3% for each of 3 classes.\n    :Creator: R.A. Fisher\n    :Donor: Michael Marshall
(MARSHALL%PLU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThe famous Iris database, first used by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s
paper. Note that it\'s the same as in R, but not as in the UCI\nMachine Learning Repository, which has two wrong data points.\n\nThis is perhaps the
best known database to be found in the\npattern recognition literature.  Fisher\'s paper is a classic in the field and\nis referenced frequently to
this day.  (See Duda & Hart, for example.)  The\ndata set contains 3 classes of 50 instances each, where each class refers to a\ntype of iris plant.
One class is linearly separable from the other 2; the\nlatter are NOT linearly separable from each other.\n\n.. topic:: References\n\n   - Fisher,
R.A. "The use of multiple measurements in taxonomic problems"\n     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to\n
Mathematical Statistics" (John Wiley, NY, 1950).\n   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.\n     (Q327.D83)
John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.\n   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System\n     Structure and
Classification Rule for Recognition in Partially Exposed\n     Environments".  IEEE Transactions on Pattern Analysis and Machine\n     Intelligence,
Vol. PAMI-2, No. 1, 67-71.\n   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions\n     on Information Theory, May 1972,
431-433.\n   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II\n     conceptual clustering system finds 3 classes in the
```

every row of flower has been labelled as a Class (0, 1, 2)
0 = Setosa / 1 = Versicolor
2 = Virginica

D. GLANCE ROW 0, 50, 100



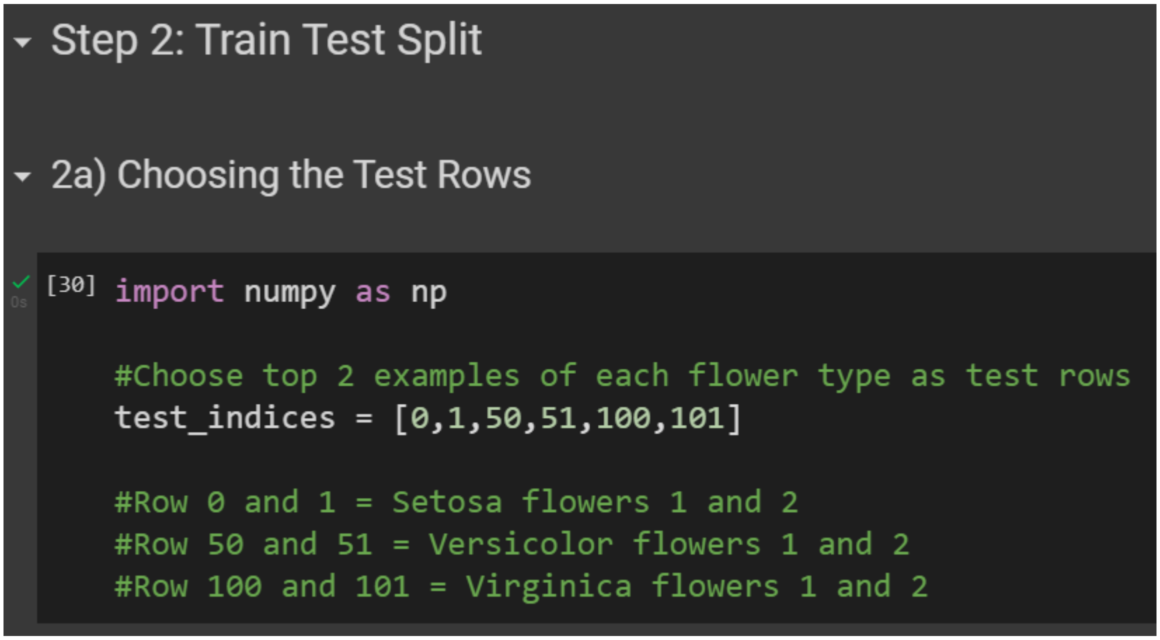
1d) Glance Row 0 , 50 and 100

```
#Print the row 0,50 and 100 i.e. 1 example for each type
print("\nSetosa flower 1 - ",iris.data[0])
print("Versicolor flower 1 - ",iris.data[50])
print("Virginica flower 1 - ",iris.data[100],"\n")

#In the iris dataset,
#Rows 0 to 49 = Setosa = Class 0
#Rows 50 to 99 = Versicolor = Class 1
#Rows 100 to 149 = Vriginica = Class 2
```

```
Setosa flower 1 -  [5.1 3.5 1.4 0.2]
Versicolor flower 1 -  [7.  3.2 4.7 1.4]
Virginica flower 1 -  [6.3 3.3 6.  2.5]
```

II. STEP 2: TRAIN TEST SPLIT

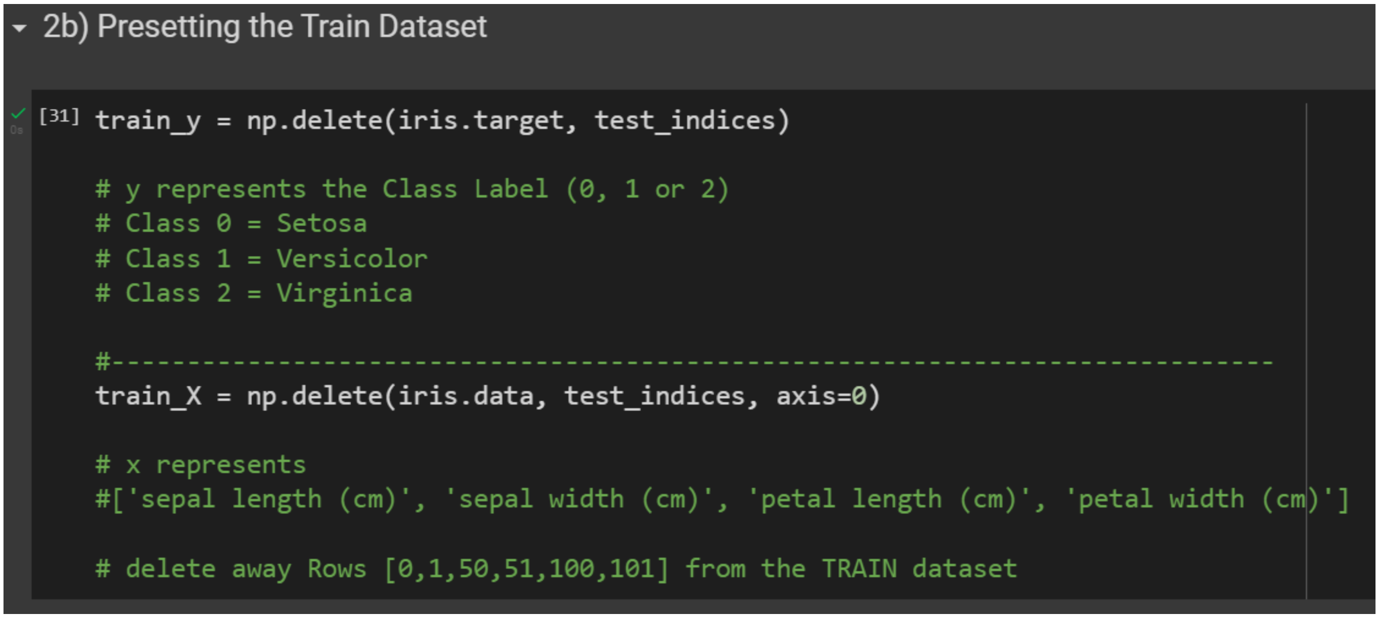
A. CHOOSING THE TEST ROWS


Step 2: Train Test Split

2a) Choosing the Test Rows

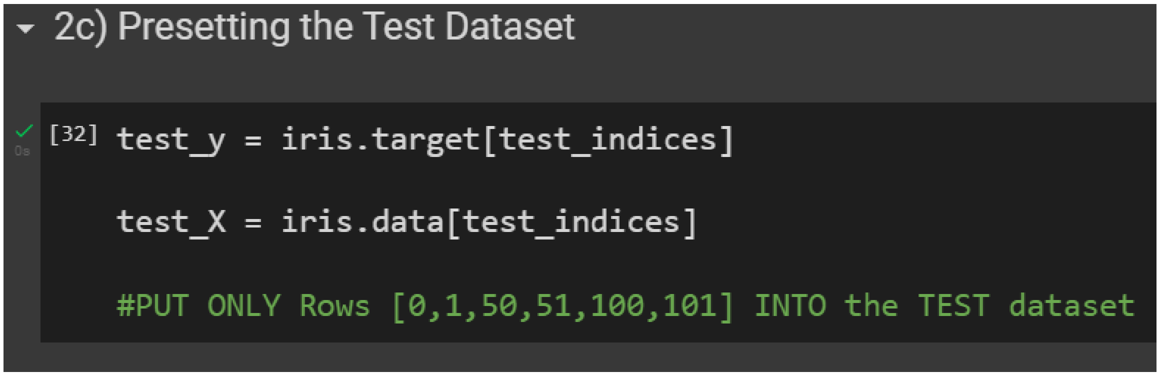
```
import numpy as np

#Choose top 2 examples of each flower type as test rows
test_indices = [0,1,50,51,100,101]

#Row 0 and 1 = Setosa flowers 1 and 2
#Row 50 and 51 = Versicolor flowers 1 and 2
#Row 100 and 101 = Virginica flowers 1 and 2
```

B. PRESETTING THE TRAIN DATASET


2b) Presetting the Train Dataset

```
train_y = np.delete(iris.target, test_indices)

# y represents the Class Label (0, 1 or 2)
# Class 0 = Setosa
# Class 1 = Versicolor
# Class 2 = Virginica

#-----------------------------------------------------------------------------
train_X = np.delete(iris.data, test_indices, axis=0)

# x represents
#['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

# delete away Rows [0,1,50,51,100,101] from the TRAIN dataset
```

C. PRESETTING THE TEST DATASET



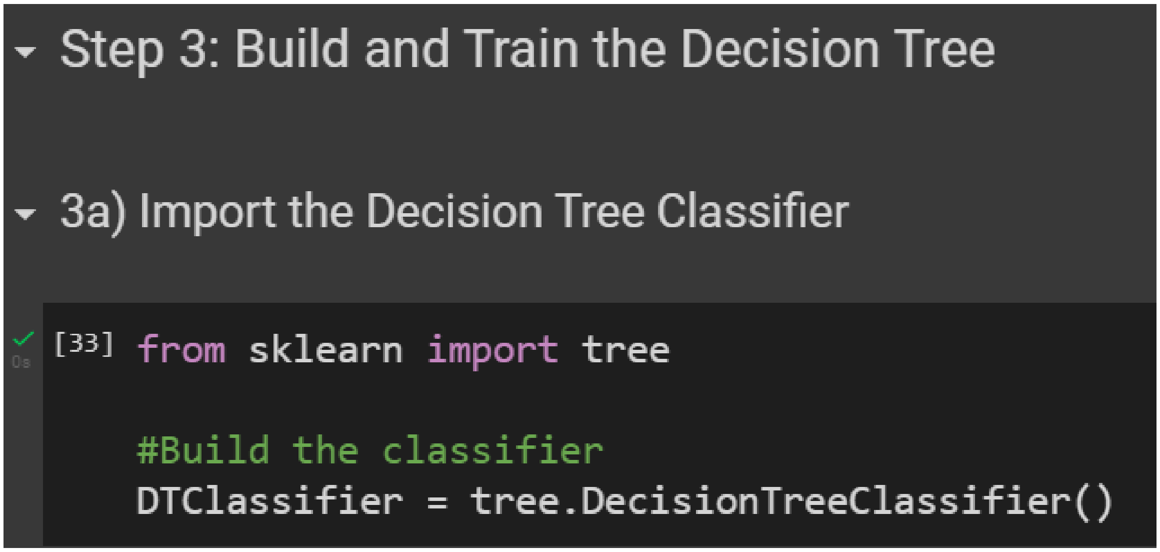
2c) Presetting the Test Dataset

[32]

```
test_y = iris.target[test_indices]

test_X = iris.data[test_indices]

#PUT ONLY Rows [0,1,50,51,100,101] INTO the TEST dataset
```

III. STEP 3: BUILD AND TRAIN THE DECISION TREE

A. IMPORT THE DECISION TREE CLASSIFIER



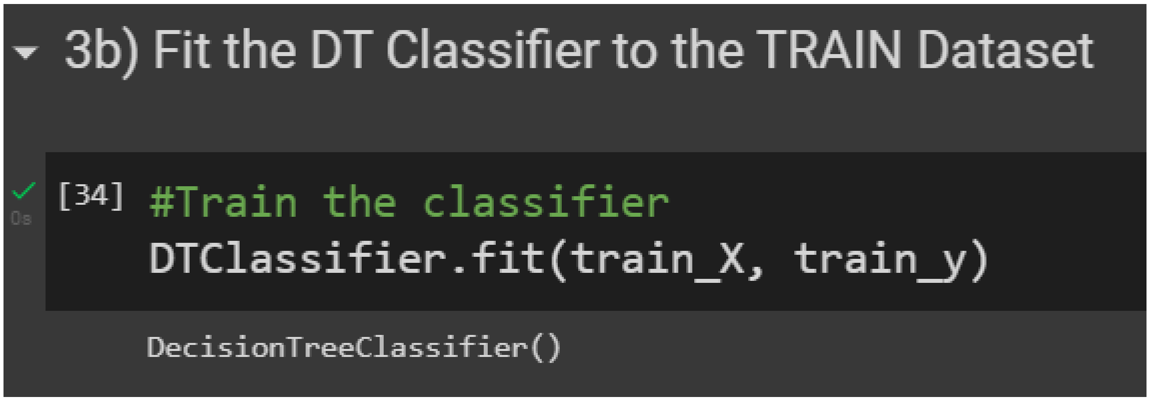
Step 3: Build and Train the Decision Tree

3a) Import the Decision Tree Classifier

[33]

```
from sklearn import tree

#Build the classifier
DTClassifier = tree.DecisionTreeClassifier()
```

B. FIT THE DT CLASSIFIER TO THE TRAIN DATASET



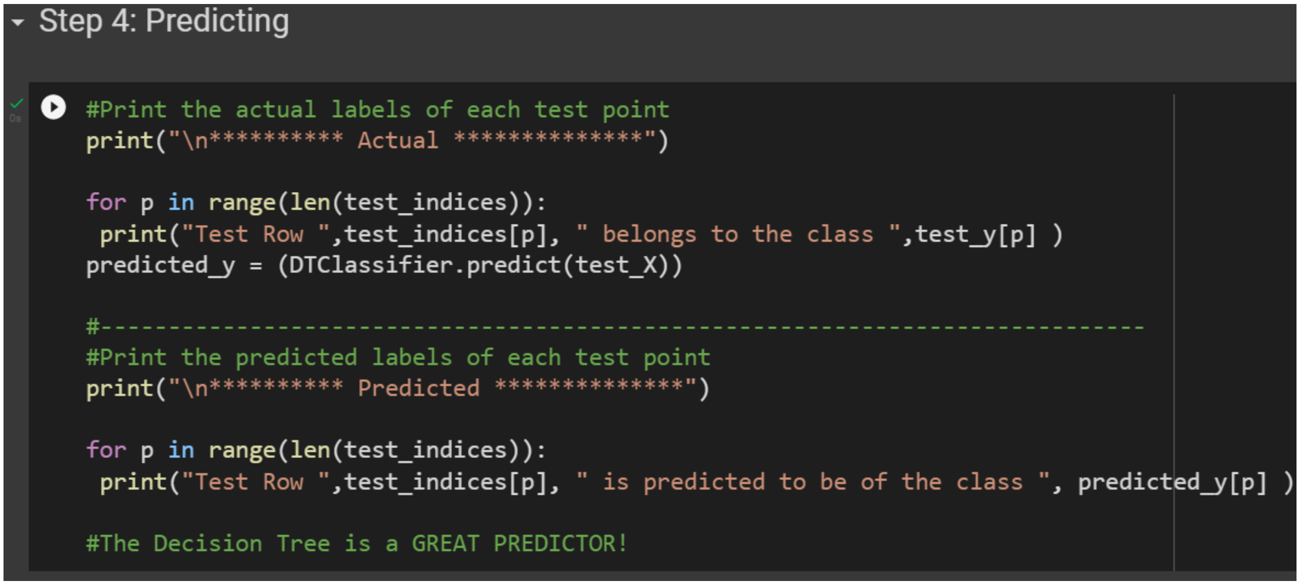
3b) Fit the DT Classifier to the TRAIN Dataset

[34]

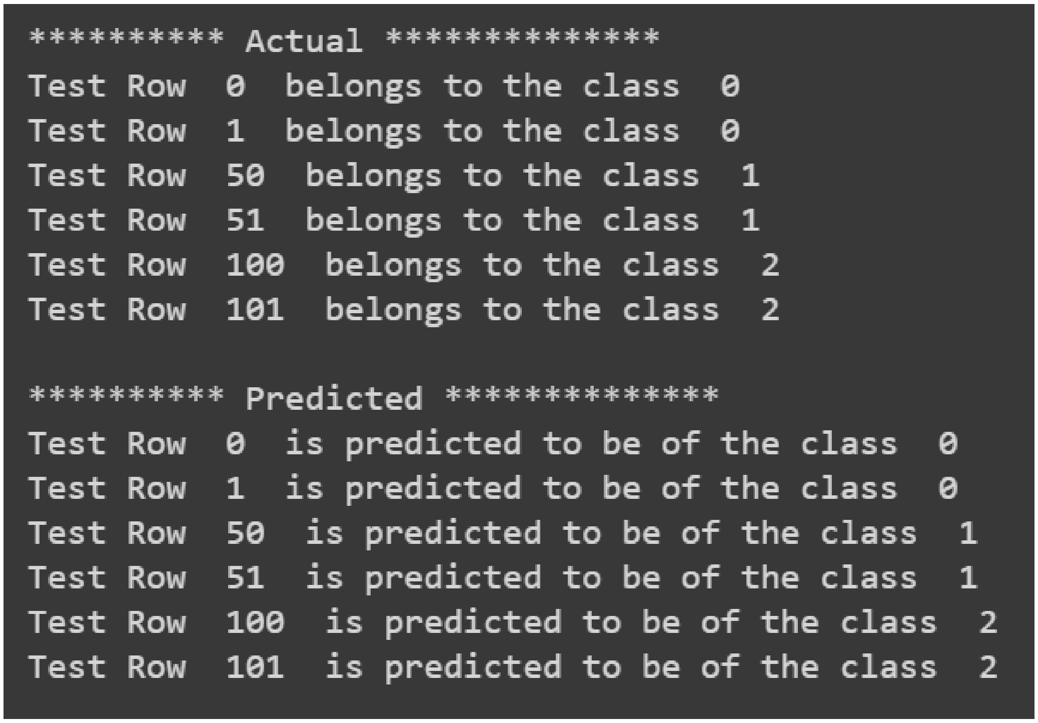
```
#Train the classifier
DTClassifier.fit(train_X, train_y)
```

```
DecisionTreeClassifier()
```

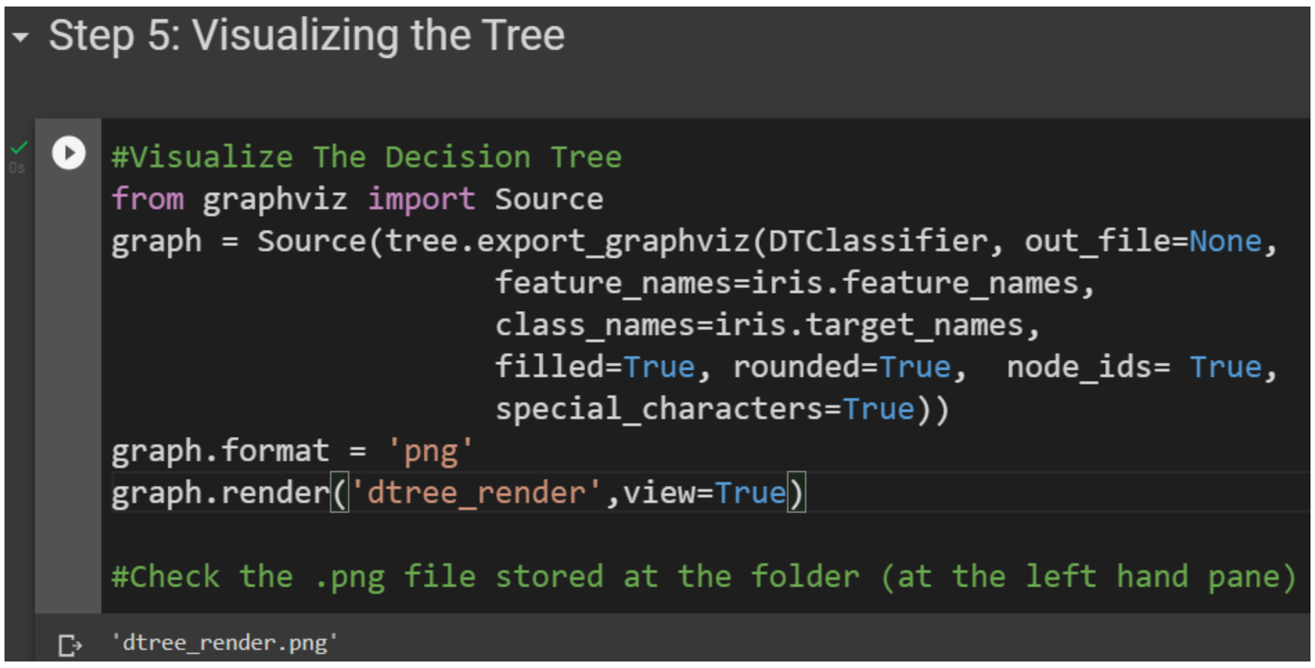
IV. STEP 4: PREDICTING



```
#Print the actual labels of each test point
print("\n********** Actual **************")

for p in range(len(test_indices)):
 print("Test Row ",test_indices[p], " belongs to the class ",test_y[p] )
predicted_y = (DTClassifier.predict(test_X))

#-----------------------------------------------------------------------------
#Print the predicted labels of each test point
print("\n********** Predicted **************")

for p in range(len(test_indices)):
 print("Test Row ",test_indices[p], " is predicted to be of the class ", predicted_y[p] )

#The Decision Tree is a GREAT PREDICTOR!
```



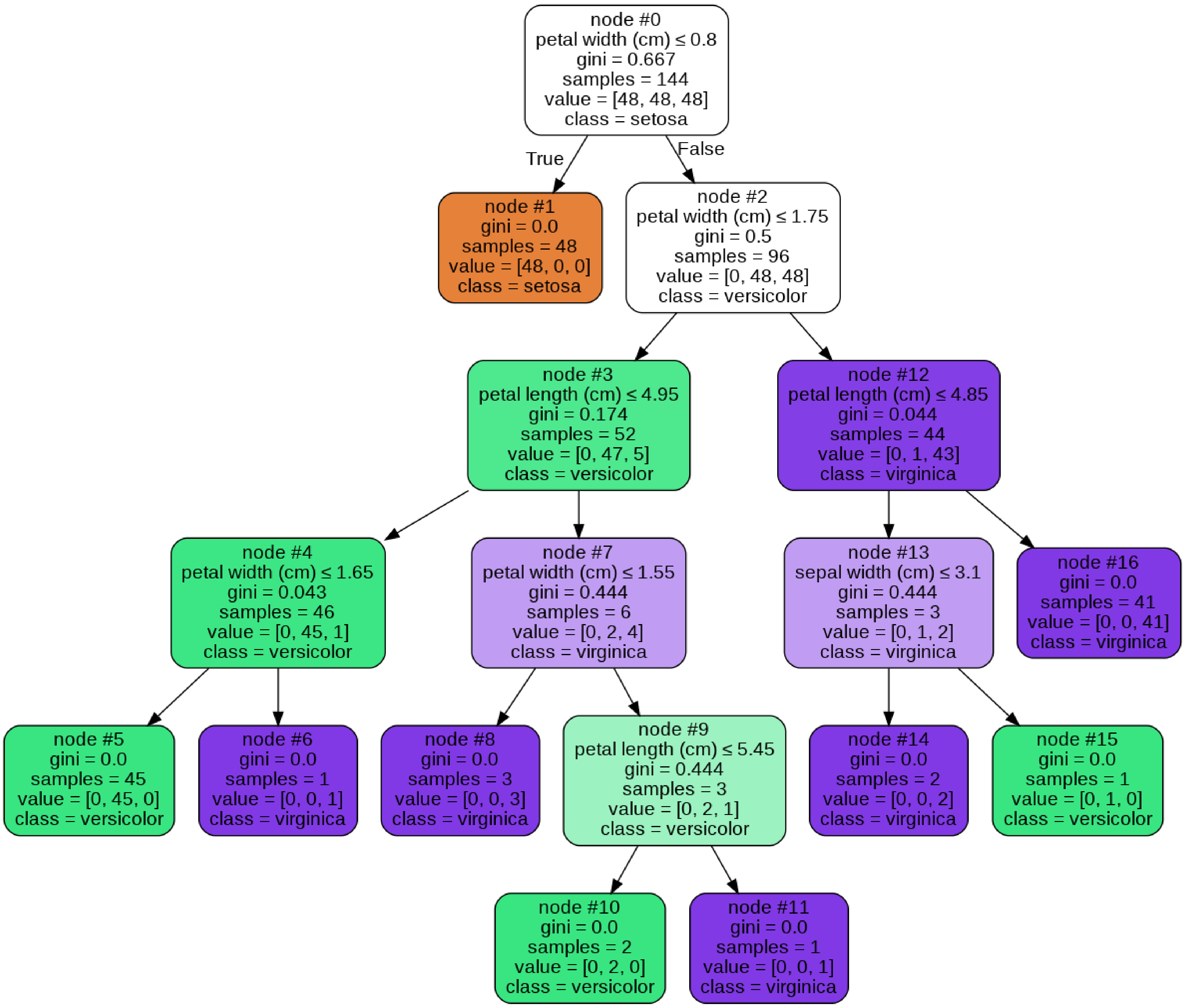
```
********** Actual **************
Test Row  0  belongs to the class  0
Test Row  1  belongs to the class  0
Test Row  50  belongs to the class  1
Test Row  51  belongs to the class  1
Test Row  100  belongs to the class  2
Test Row  101  belongs to the class  2

********** Predicted **************
Test Row  0  is predicted to be of the class  0
Test Row  1  is predicted to be of the class  0
Test Row  50  is predicted to be of the class  1
Test Row  51  is predicted to be of the class  1
Test Row  100  is predicted to be of the class  2
Test Row  101  is predicted to be of the class  2
```



Step 5: Visualizing the Tree

```
#Visualize The Decision Tree
from graphviz import Source
graph = Source(tree.export_graphviz(DTClassifier, out_file=None,
                     feature_names=iris.feature_names,
                     class_names=iris.target_names,
                     filled=True, rounded=True,  node_ids= True,
                     special_characters=True))
graph.format = 'png'
graph.render('dtree_render',view=True)

#Check the .png file stored at the folder (at the left hand pane)
```

```
'dtree_render.png'
```



THE END

ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.