Name :	**Dipangshu Roy**
Roll No :	**001811001014**
Class :	**IT 4th Year 1st Sem**
Subject :	**Machine Learning Lab**

Assignment 1

Construct a machine learning based model for classification using Python for the following UCI datasets:

UCI datasets (can be loaded from the package itself):

1. Iris plants dataset : https://archive.ics.uci.edu/ml/datasets/Iris/
2. Diabetes dataset : https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html
3. Wisconsin Breast Cancer Dataset : https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

2. Use Decision Tree classifier for all the three datasets and show classification results (Accuracy, Precision, Recall, F-score, confusion matrix) with and without parameter tuning. Generate the decision tree images for all cases highlighting information like Gini and Entropy.

Classification: Decision Tree

Without parameter tuning :

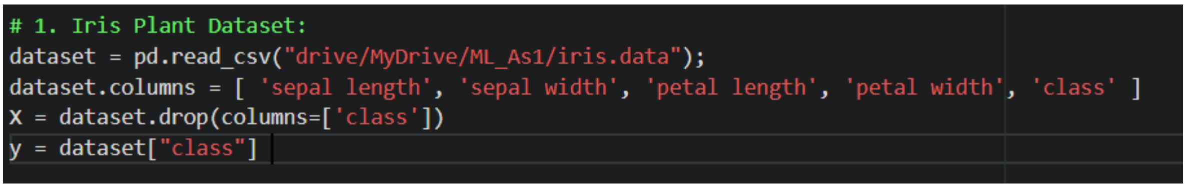
The following code contains the python program to Use Decision Tree classifier for all the three datasets.

To run this program on each of the three dataset, we have to remove the comment on the respective dataset and comment out all the remaining dataset.

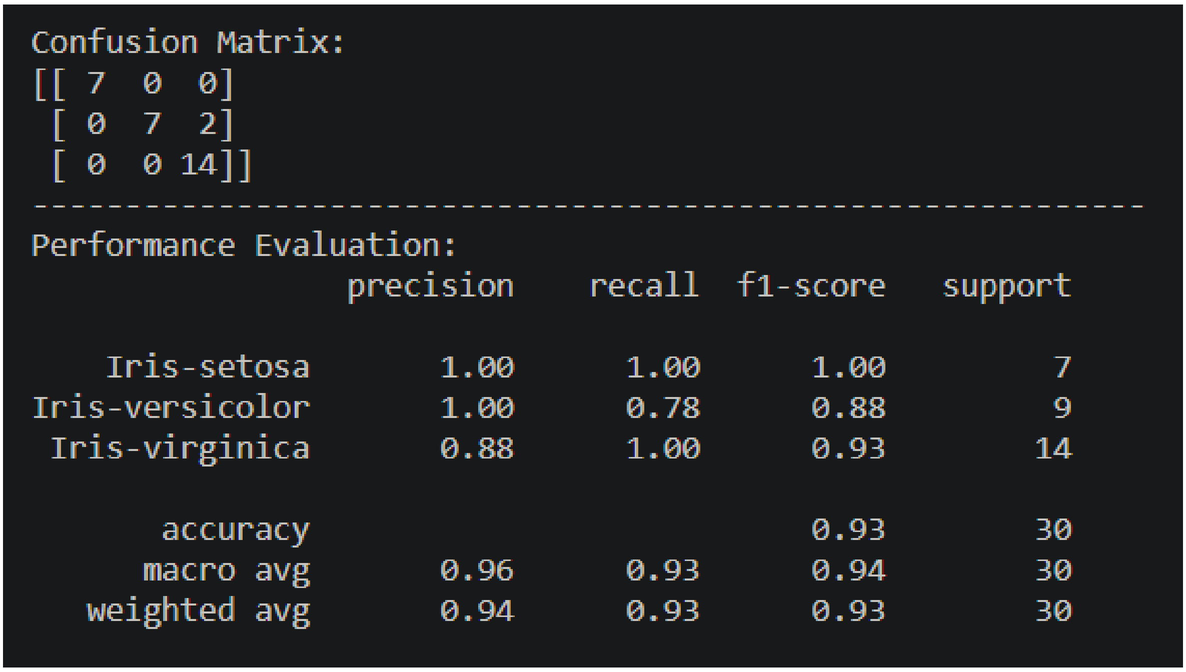
```
# Decision Tree for Classification

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Dataset Preparation

# 1. Iris Plants Dataset:
dataset = pd.read_csv("drive/MyDrive/ML_As1/iris.data");
dataset.columns = [ 'sepal length', 'sepal width', 'petal length', 'petal width',
'class' ]
X = dataset.drop(columns=['class'])
y = dataset["class"]
```

```
# 2. Diabetes Dataset :
# dataset = pd.read_csv("drive/MyDrive/ML_As1/diabetes.tab.txt",sep='\t')
# X = dataset.drop(columns=['SEX'])
# y = dataset["SEX"]

# 3. Wisconsin Breast Cancer Dataset :
# dataset = pd.read_csv("drive/MyDrive/ML_As1/breast-cancer-wisconsin.data");
# dataset.columns = [ 'Sample code number',
#                     'Clump Thickness',
#                     'Uniformity of Cell Size',
#                     'Uniformity of Cell Shape',
#                     'Marginal Adhesion',
#                     'Single Epithelial Cell Size',
#                     'Bare Nuclei',
#                     'Bland Chromatin',
#                     'Normal Nucleoli',
#                     'Mitoses',
#                     'class' ]
# dataset.drop(dataset[dataset['Bare Nuclei'] == '?'].index, inplace=True) # droping rows with missing attribute value denoted by "?"
# X = dataset.drop(columns=[ 'Sample code number', 'class'])
# y = dataset["class"]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

# Classification

from sklearn.tree import DecisionTreeClassifier

# Without Parameter Tuning
classifier = DecisionTreeClassifier()

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

# Evaluation of Classifier Performance

from sklearn.metrics import classification_report, confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("------------------------------------------------------------")

print("Performance Evaluation:")
print(classification_report(y_test, y_pred))
```

Classification results of Iris Plants Dataset :

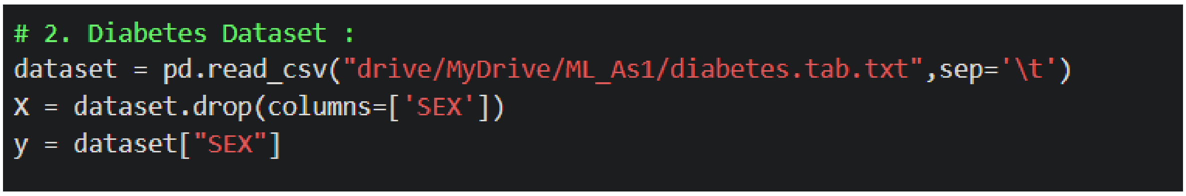


```
# 1. Iris Plant Dataset:
dataset = pd.read_csv("drive/MyDrive/ML_As1/iris.data");
dataset.columns = [ 'sepal length', 'sepal width', 'petal length', 'petal width', 'class' ]
X = dataset.drop(columns=['class'])
y = dataset["class"]
```

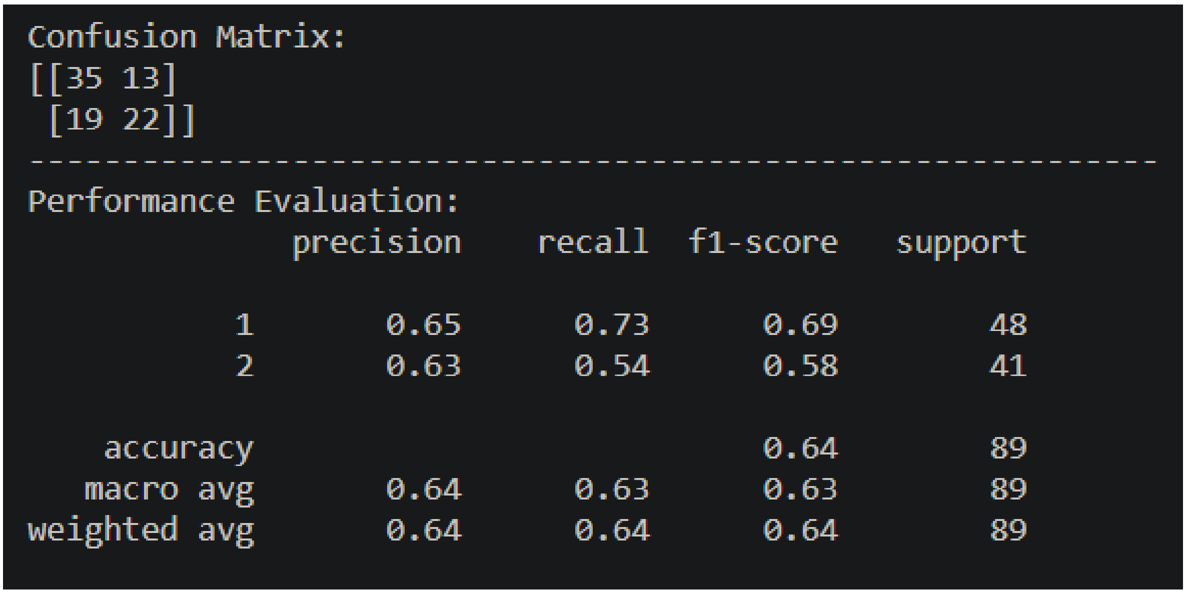


```
Confusion Matrix:
[[ 7  0  0]
 [ 0  7  2]
 [ 0  0 14]]
------------------------------------------------------------
Performance Evaluation:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       1.00      0.78      0.88         9
 Iris-virginica       0.88      1.00      0.93        14

       accuracy                           0.93        30
      macro avg       0.96      0.93      0.94        30
   weighted avg       0.94      0.93      0.93        30
```

Classification results of Diabetes Dataset :

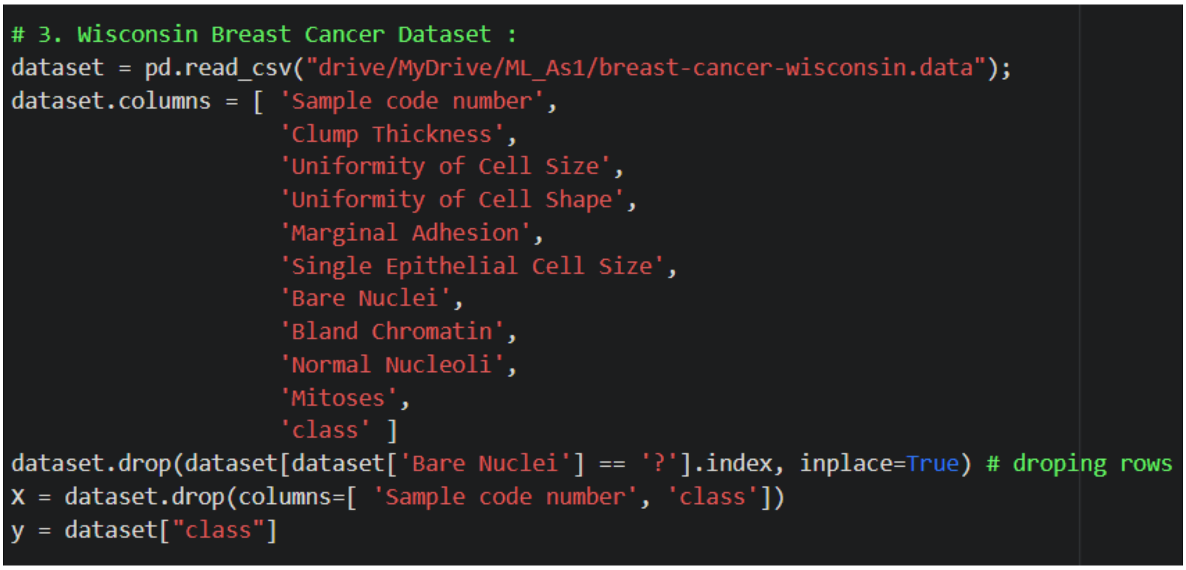


```
# 2. Diabetes Dataset :
dataset = pd.read_csv("drive/MyDrive/ML_As1/diabetes.tab.txt",sep='\t')
X = dataset.drop(columns=['SEX'])
y = dataset["SEX"]
```

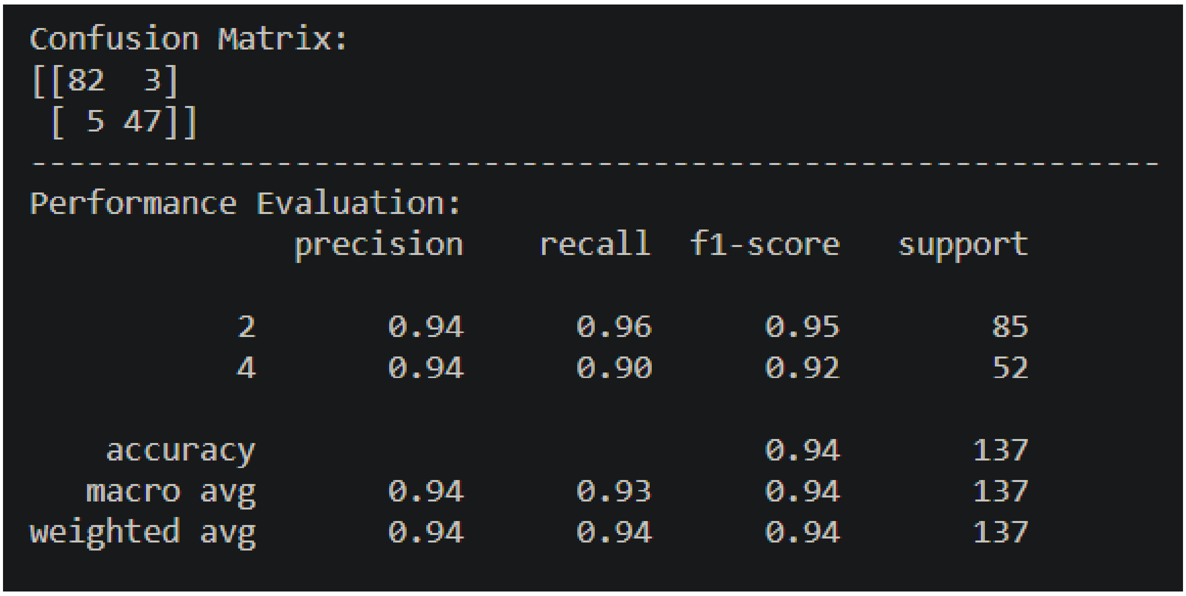


```
Confusion Matrix:
[[35 13]
 [19 22]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           1       0.65      0.73      0.69        48
           2       0.63      0.54      0.58        41

    accuracy                           0.64        89
   macro avg       0.64      0.63      0.63        89
weighted avg       0.64      0.64      0.64        89
```

Classification results of Wisconsin Breast Cancer Dataset :



```
# 3. Wisconsin Breast Cancer Dataset :
dataset = pd.read_csv("drive/MyDrive/ML_As1/breast-cancer-wisconsin.data");
dataset.columns = [ 'Sample code number',
                    'Clump Thickness',
                    'Uniformity of Cell Size',
                    'Uniformity of Cell Shape',
                    'Marginal Adhesion',
                    'Single Epithelial Cell Size',
                    'Bare Nuclei',
                    'Bland Chromatin',
                    'Normal Nucleoli',
                    'Mitoses',
                    'class' ]
dataset.drop(dataset[dataset['Bare Nuclei'] == '?'].index, inplace=True) # droping rows
X = dataset.drop(columns=[ 'Sample code number', 'class'])
y = dataset["class"]
```



```
Confusion Matrix:
[[82  3]
 [ 5 47]]
----------------------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           2       0.94      0.96      0.95        85
           4       0.94      0.90      0.92        52

    accuracy                           0.94       137
   macro avg       0.94      0.93      0.94       137
weighted avg       0.94      0.94      0.94       137
```

With parameter tuning :

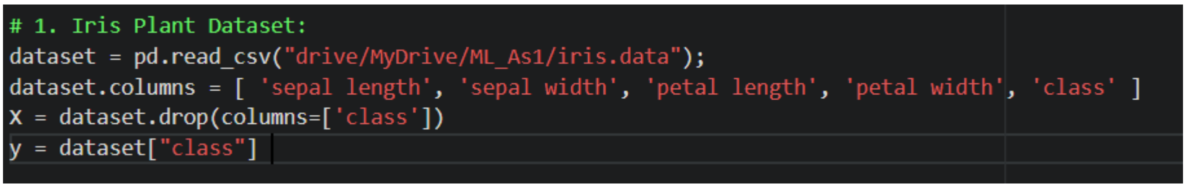
The following code contains the python program to Use Decision Tree classifier for all the three datasets.

To run this program on each of the three dataset, we have to remove the comment on the respective dataset and comment out all the remaining dataset.

```
#Decision Tree for Classification

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Dataset Preparation

# 1. Iris Plants Dataset:
dataset = pd.read_csv("drive/MyDrive/ML_As1/iris.data");
dataset.columns = [ 'sepal length', 'sepal width', 'petal length', 'petal width',
'class' ]
X = dataset.drop(columns=['class'])
y = dataset["class"]

# 2. Diabetes Dataset :
```

```
# dataset = pd.read_csv("drive/MyDrive/ML_As1/diabetes.tab.txt",sep='\t')
# X = dataset.drop(columns=['SEX'])
# y = dataset["SEX"]

# 3. Wisconsin Breast Cancer Dataset :
# dataset = pd.read_csv("drive/MyDrive/ML_As1/breast-cancer-wisconsin.data");
# dataset.columns = [ 'Sample code number',
#                     'Clump Thickness',
#                     'Uniformity of Cell Size',
#                     'Uniformity of Cell Shape',
#                     'Marginal Adhesion',
#                     'Single Epithelial Cell Size',
#                     'Bare Nuclei',
#                     'Bland Chromatin',
#                     'Normal Nucleoli',
#                     'Mitoses',
#                     'class' ]
# dataset.drop(dataset[dataset['Bare Nuclei'] == '?'].index, inplace=True) # droping rows with missing attribute value denoted by "?"
# X = dataset.drop(columns=[ 'Sample code number', 'class'])
# y = dataset["class"]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

# Classification

from sklearn.tree import DecisionTreeClassifier

# With Parameter Tuning
classifier = DecisionTreeClassifier(criterion="entropy", max_depth=3);

# classifier = DecisionTreeClassifier(criterion="entropy", max_depth=10);

# classifier = DecisionTreeClassifier(criterion="gini", max_depth=10);

# classifier = DecisionTreeClassifier(criterion="gini", max_depth=15);

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

# Evaluation of Classifier Performance

from sklearn.metrics import classification_report, confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("------------------------------------------------------------")

print("Performance Evaluation:")
print(classification_report(y_test, y_pred))
```

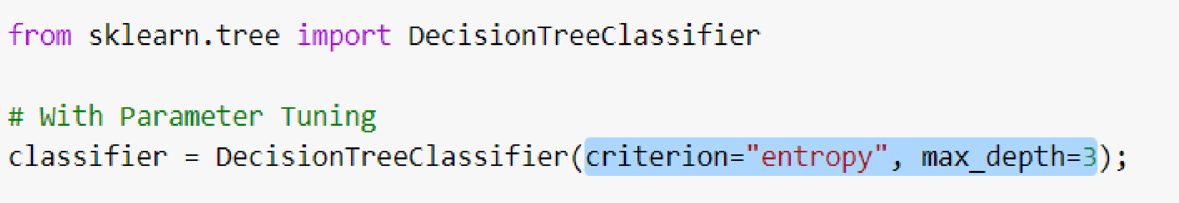
Iris Plants Dataset :



```
# 1. Iris Plant Dataset:
dataset = pd.read_csv("drive/MyDrive/ML_As1/iris.data");
dataset.columns = [ 'sepal length', 'sepal width', 'petal length', 'petal width', 'class' ]
X = dataset.drop(columns=['class'])
y = dataset["class"]
```

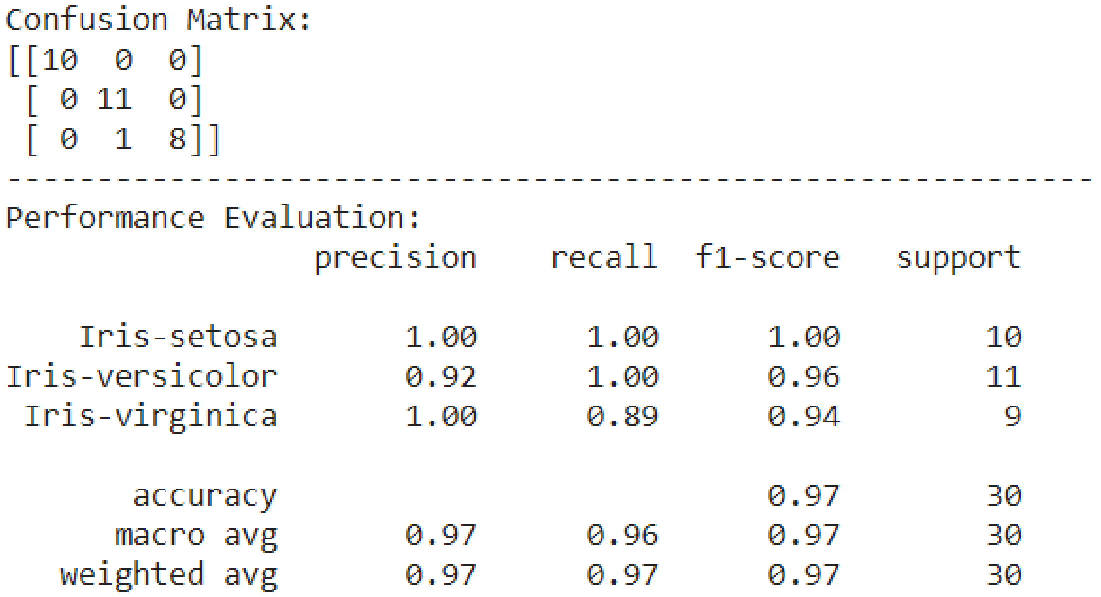
With “entropy” criterion :

> **Passing Parameters : criterion="entropy", max_depth=3**



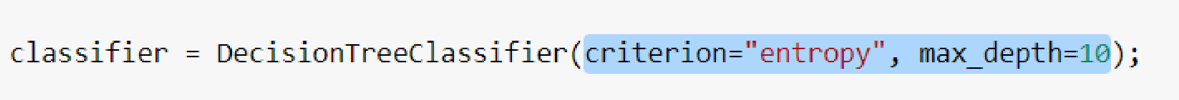
```
from sklearn.tree import DecisionTreeClassifier

# With Parameter Tuning
classifier = DecisionTreeClassifier(criterion="entropy", max_depth=3);
```

Output :



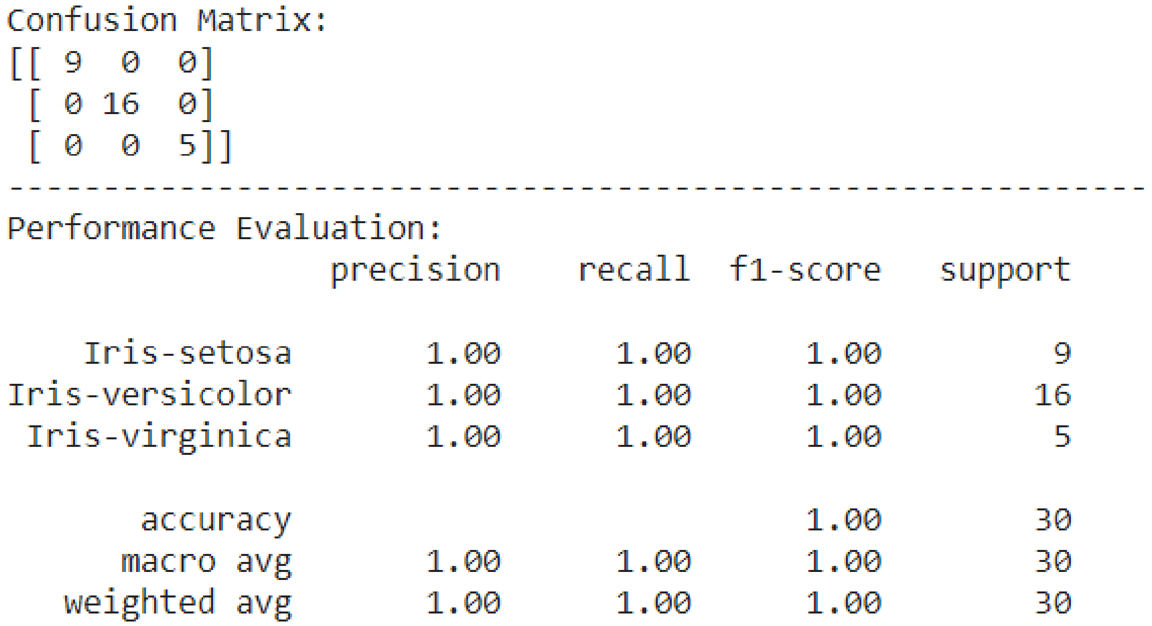
```
Confusion Matrix:
[[10  0  0]
 [ 0 11  0]
 [ 0  1  8]]
------------------------------------------------------------
Performance Evaluation:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       0.92      1.00      0.96        11
 Iris-virginica       1.00      0.89      0.94         9

       accuracy                           0.97        30
      macro avg       0.97      0.96      0.97        30
   weighted avg       0.97      0.97      0.97        30
```

> **criterion="entropy", max_depth=10**



```
classifier = DecisionTreeClassifier(criterion="entropy", max_depth=10);
```

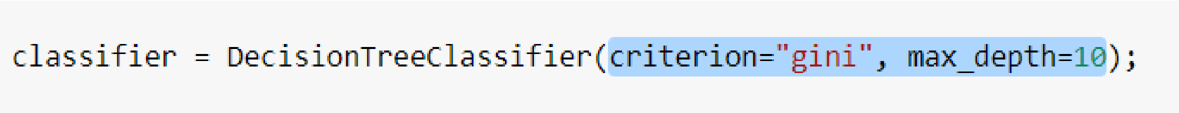
Output :



```
Confusion Matrix:
[[ 9  0  0]
 [ 0 16  0]
 [ 0  0  5]]
------------------------------------------------------------
Performance Evaluation:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         9
Iris-versicolor       1.00      1.00      1.00        16
 Iris-virginica       1.00      1.00      1.00         5

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

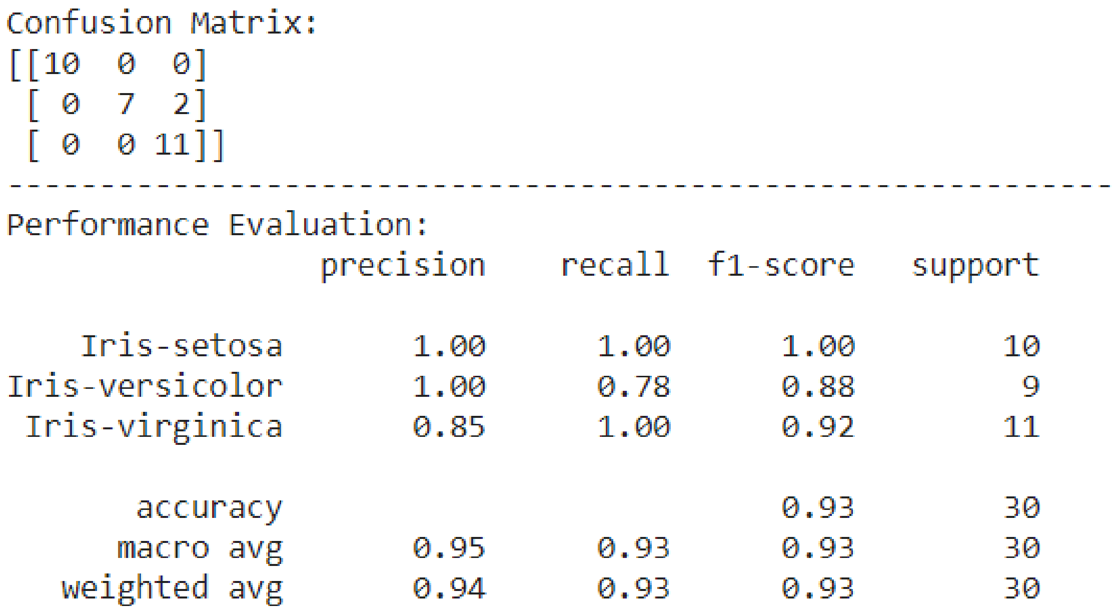
With “gini” criterion :

> **criterion="gini", max_depth=10**



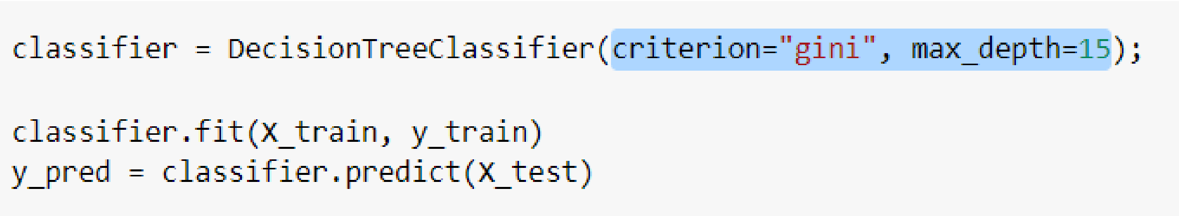
```
classifier = DecisionTreeClassifier(criterion="gini", max_depth=10);
```

Output :



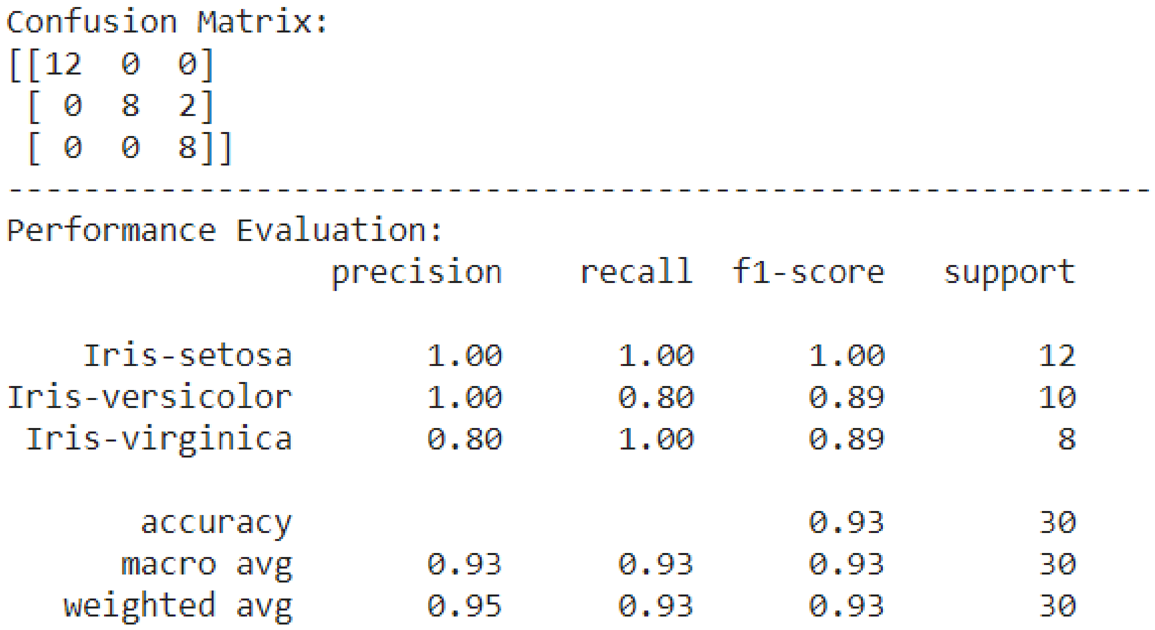
```
Confusion Matrix:
[[10  0  0]
 [ 0  7  2]
 [ 0  0 11]]
------------------------------------------------------------
Performance Evaluation:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      0.78      0.88         9
 Iris-virginica       0.85      1.00      0.92        11

       accuracy                           0.93        30
      macro avg       0.95      0.93      0.93        30
   weighted avg       0.94      0.93      0.93        30
```

> **criterion="gini", max_depth=15**



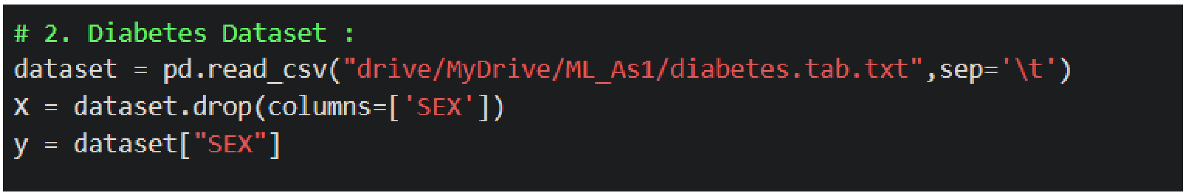
```
classifier = DecisionTreeClassifier(criterion="gini", max_depth=15);

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

Output :



```
Confusion Matrix:
[[12  0  0]
 [ 0  8  2]
 [ 0  0  8]]
------------------------------------------------------------
Performance Evaluation:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        12
Iris-versicolor       1.00      0.80      0.89        10
 Iris-virginica       0.80      1.00      0.89         8

       accuracy                           0.93        30
      macro avg       0.93      0.93      0.93        30
   weighted avg       0.95      0.93      0.93        30
```

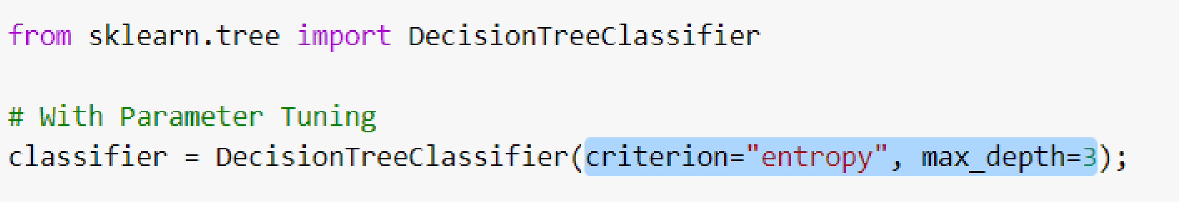
Diabetes Dataset :



```
# 2. Diabetes Dataset :
dataset = pd.read_csv("drive/MyDrive/ML_As1/diabetes.tab.txt",sep='\t')
X = dataset.drop(columns=['SEX'])
y = dataset["SEX"]
```

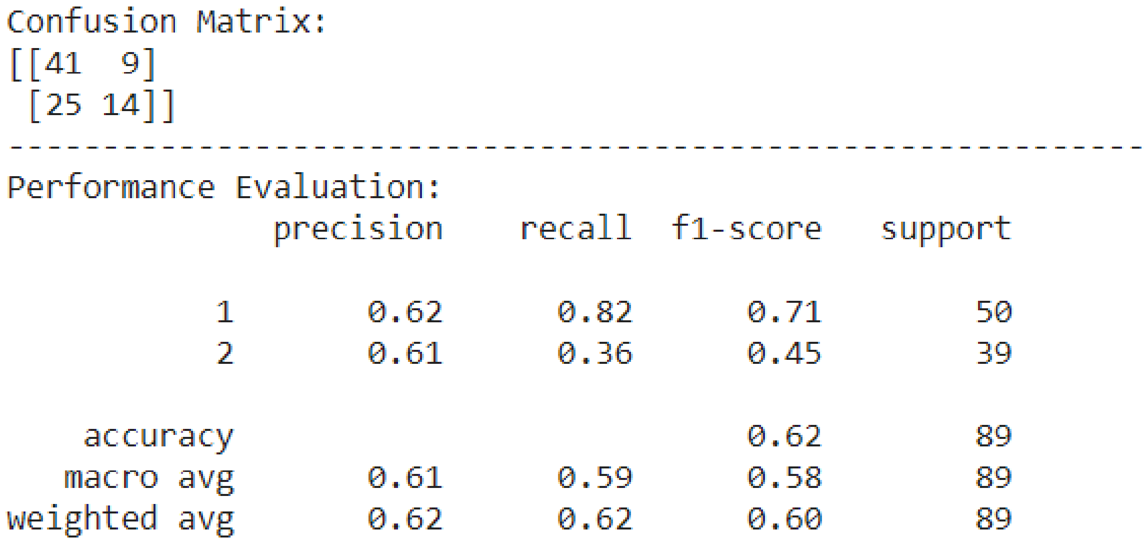
With “entropy” criterion :

> **criterion="entropy", max_depth=3**



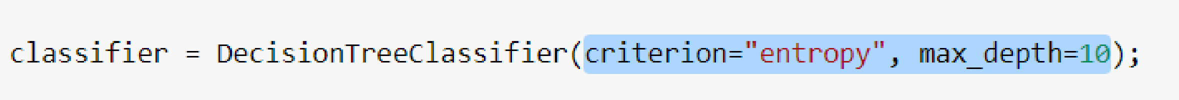
```
from sklearn.tree import DecisionTreeClassifier

# With Parameter Tuning
classifier = DecisionTreeClassifier(criterion="entropy", max_depth=3);
```

<u>Output</u> :



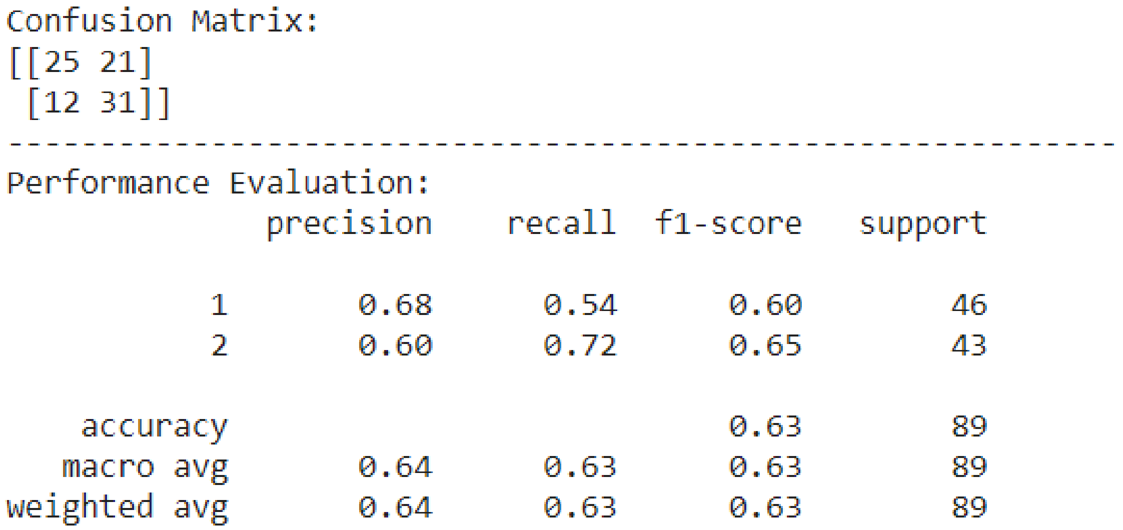
```
Confusion Matrix:
[[41  9]
 [25 14]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           1       0.62      0.82      0.71        50
           2       0.61      0.36      0.45        39

    accuracy                           0.62        89
   macro avg       0.61      0.59      0.58        89
weighted avg       0.62      0.62      0.60        89
```

> **criterion="entropy", max_depth=10**



```
classifier = DecisionTreeClassifier(criterion="entropy", max_depth=10);
```

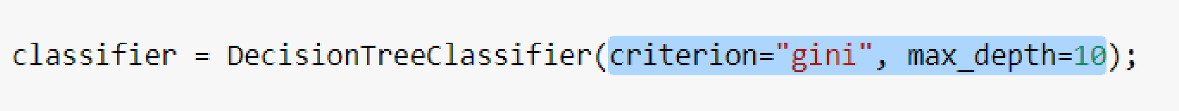
<u>Output</u> :



```
Confusion Matrix:
[[25 21]
 [12 31]]
-----------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           1       0.68      0.54      0.60        46
           2       0.60      0.72      0.65        43

    accuracy                           0.63        89
   macro avg       0.64      0.63      0.63        89
weighted avg       0.64      0.63      0.63        89
```

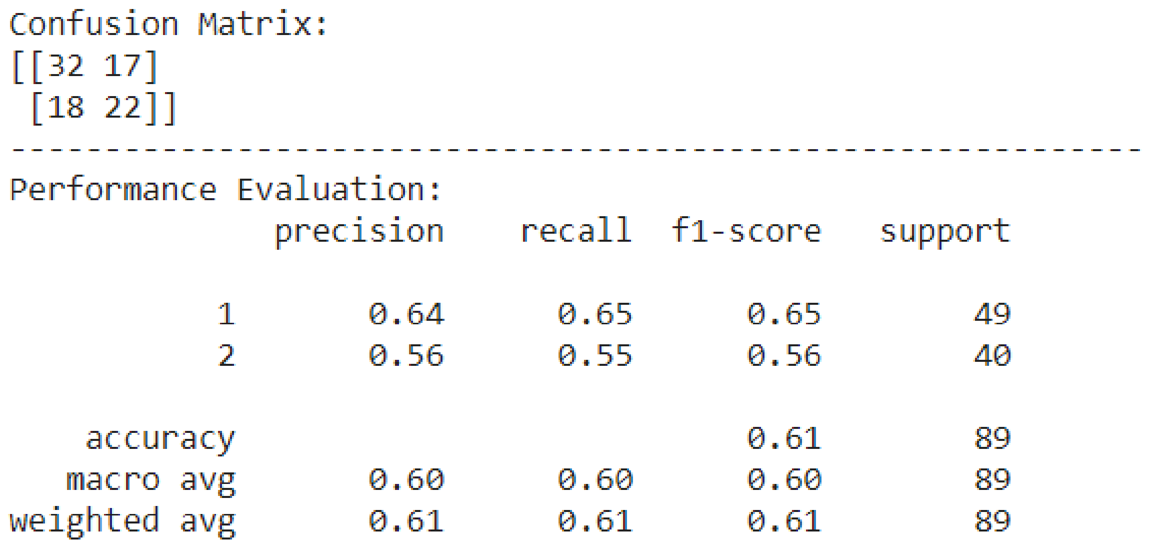
With “gini” criterion :

> **criterion="gini", max_depth=10**



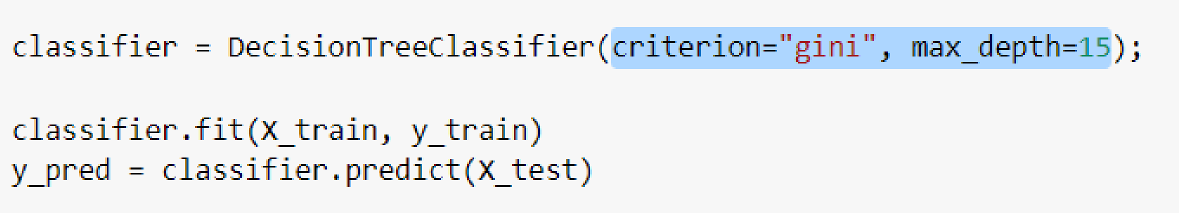
```
classifier = DecisionTreeClassifier(criterion="gini", max_depth=10);
```

Output :



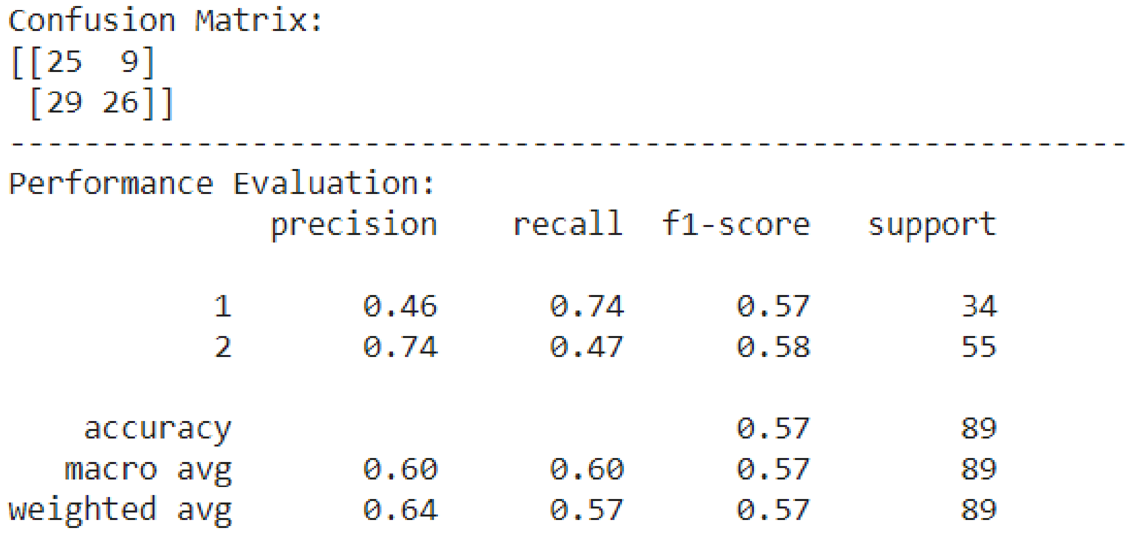
```
Confusion Matrix:
[[32 17]
 [18 22]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           1       0.64      0.65      0.65        49
           2       0.56      0.55      0.56        40

    accuracy                           0.61        89
   macro avg       0.60      0.60      0.60        89
weighted avg       0.61      0.61      0.61        89
```

> **criterion="gini", max_depth=15**



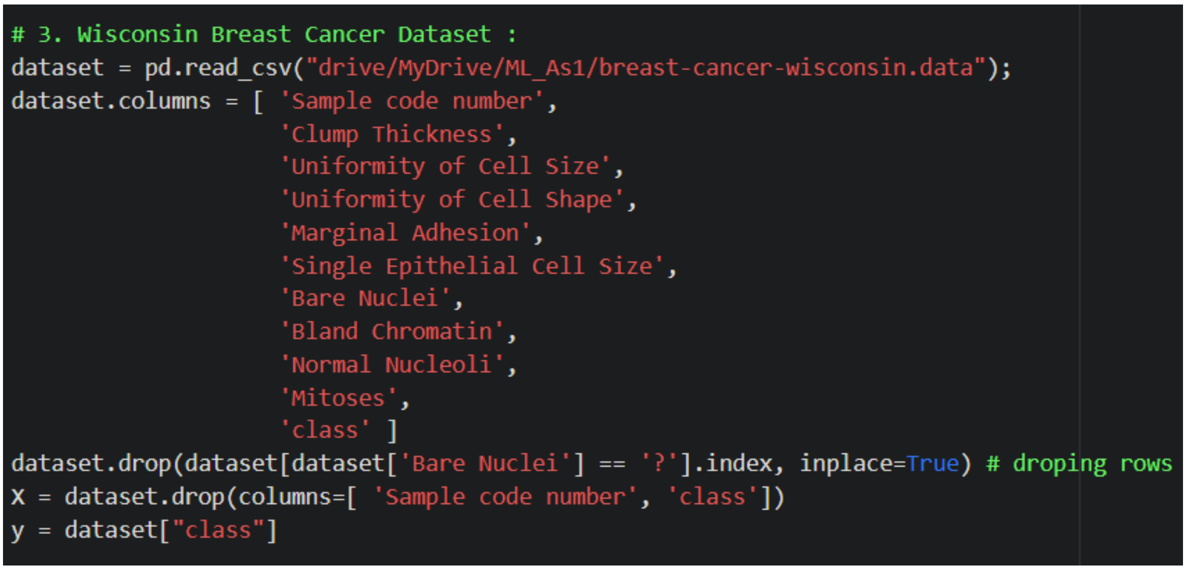
```
classifier = DecisionTreeClassifier(criterion="gini", max_depth=15);

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

Output :



```
Confusion Matrix:
[[25  9]
 [29 26]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           1       0.46      0.74      0.57        34
           2       0.74      0.47      0.58        55

    accuracy                           0.57        89
   macro avg       0.60      0.60      0.57        89
weighted avg       0.64      0.57      0.57        89
```

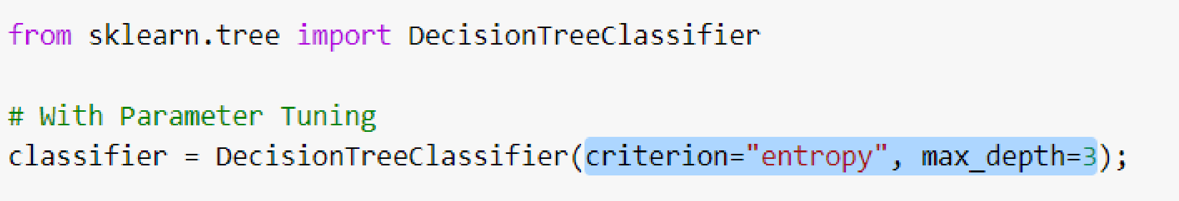
Wisconsin Breast Cancer Dataset :



```
# 3. Wisconsin Breast Cancer Dataset :
dataset = pd.read_csv("drive/MyDrive/ML_As1/breast-cancer-wisconsin.data");
dataset.columns = [ 'Sample code number',
                    'Clump Thickness',
                    'Uniformity of Cell Size',
                    'Uniformity of Cell Shape',
                    'Marginal Adhesion',
                    'Single Epithelial Cell Size',
                    'Bare Nuclei',
                    'Bland Chromatin',
                    'Normal Nucleoli',
                    'Mitoses',
                    'class' ]
dataset.drop(dataset[dataset['Bare Nuclei'] == '?'].index, inplace=True) # droping rows
X = dataset.drop(columns=[ 'Sample code number', 'class'])
y = dataset["class"]
```

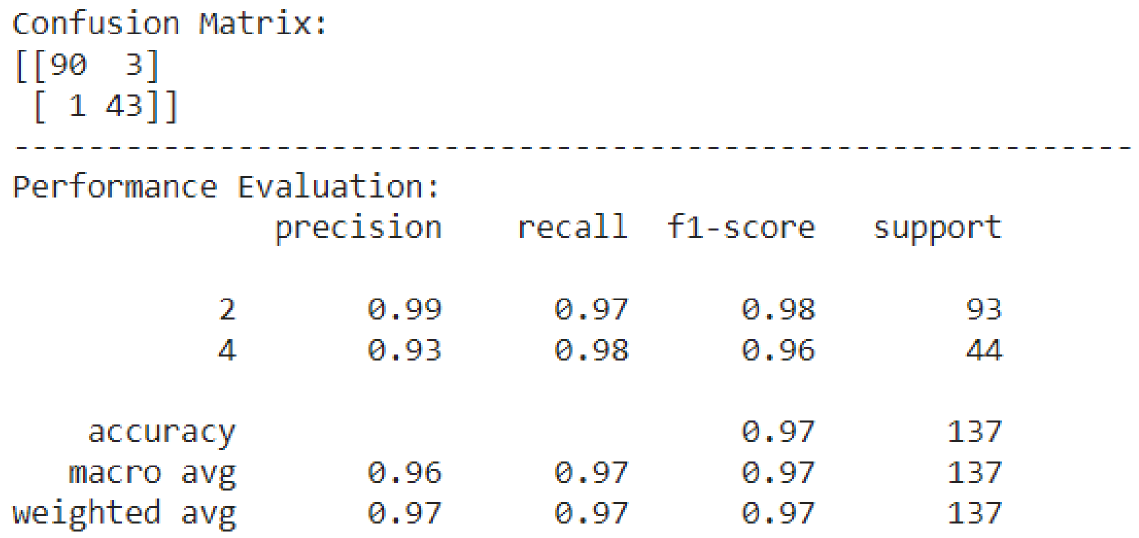
With “entropy” criterion :

> **criterion="entropy", max_depth=3**



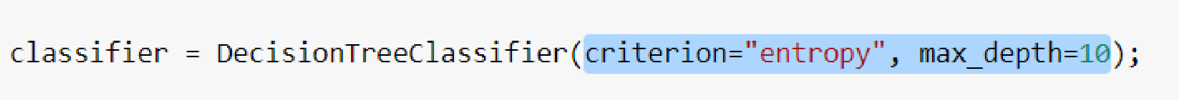
```
from sklearn.tree import DecisionTreeClassifier

# With Parameter Tuning
classifier = DecisionTreeClassifier(criterion="entropy", max_depth=3);
```

Output :



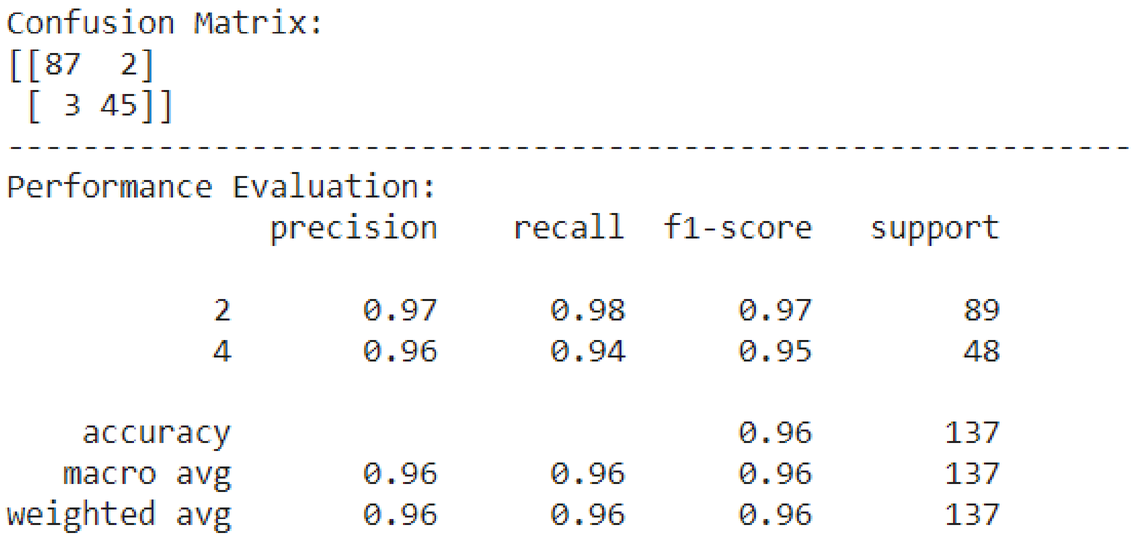
```
Confusion Matrix:
[[90  3]
 [ 1 43]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           2       0.99      0.97      0.98        93
           4       0.93      0.98      0.96        44

    accuracy                           0.97       137
   macro avg       0.96      0.97      0.97       137
weighted avg       0.97      0.97      0.97       137
```

> **criterion="entropy", max_depth=10**



```
classifier = DecisionTreeClassifier(criterion="entropy", max_depth=10);
```

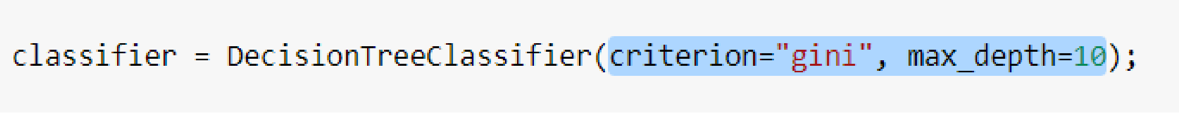
Output :



```
Confusion Matrix:
[[87  2]
 [ 3 45]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           2       0.97      0.98      0.97        89
           4       0.96      0.94      0.95        48

    accuracy                           0.96       137
   macro avg       0.96      0.96      0.96       137
weighted avg       0.96      0.96      0.96       137
```

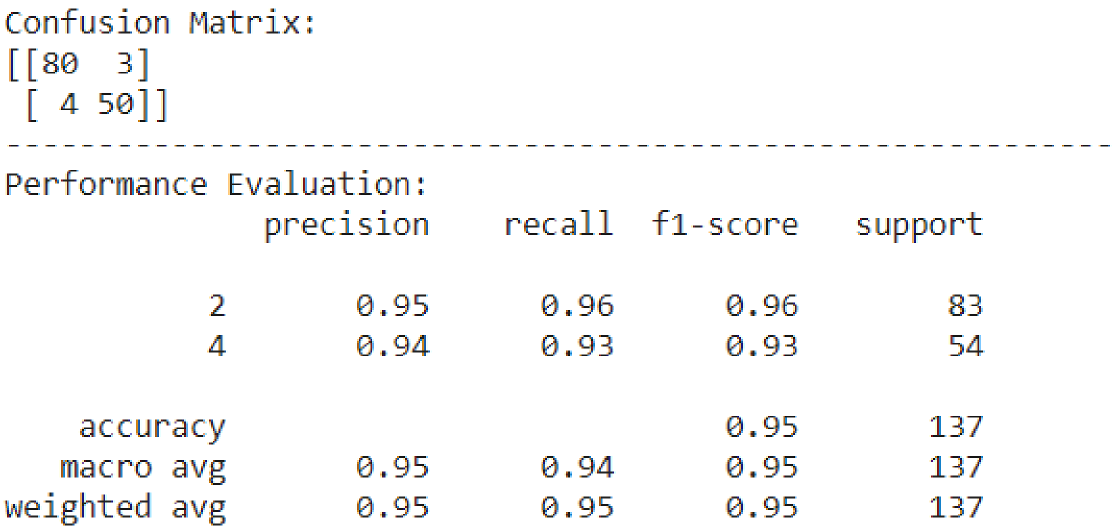
With "gini" criterion :

> **criterion="gini", max_depth=10**



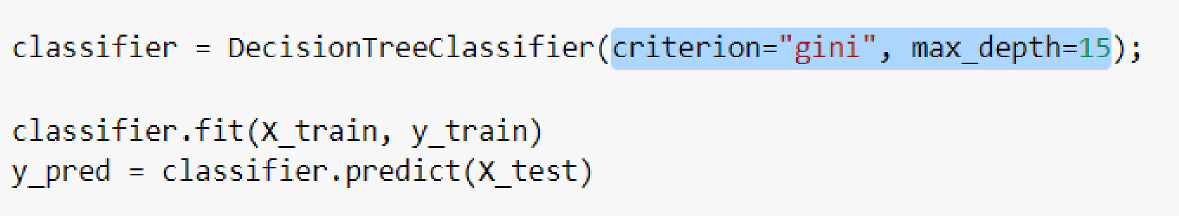
```
classifier = DecisionTreeClassifier(criterion="gini", max_depth=10);
```

Output :



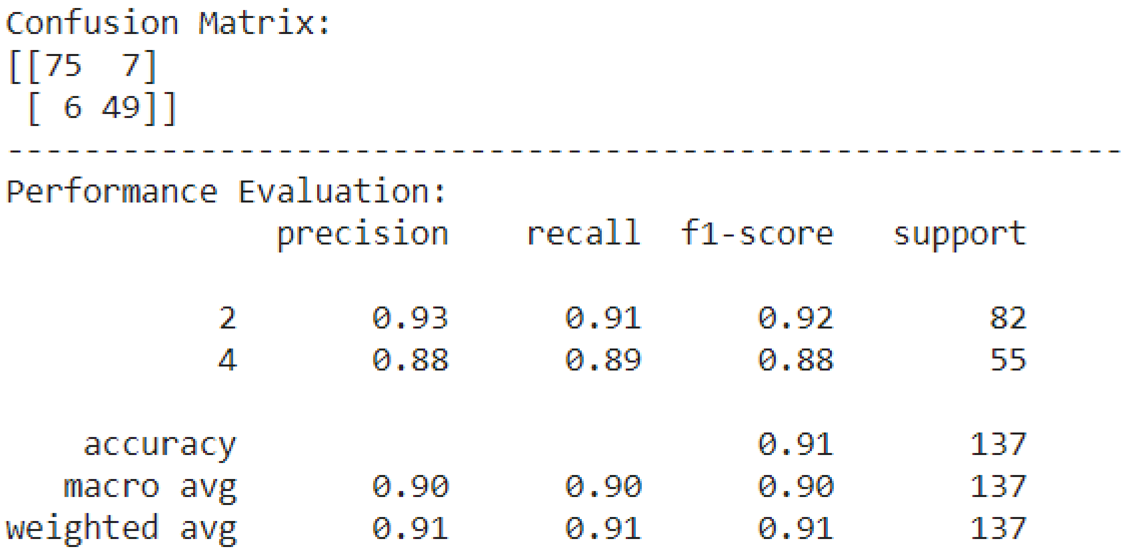
```
Confusion Matrix:
[[80  3]
 [ 4 50]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           2       0.95      0.96      0.96        83
           4       0.94      0.93      0.93        54

    accuracy                           0.95       137
   macro avg       0.95      0.94      0.95       137
weighted avg       0.95      0.95      0.95       137
```

> **criterion="gini", max_depth=15**



```
classifier = DecisionTreeClassifier(criterion="gini", max_depth=15);

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

Output :

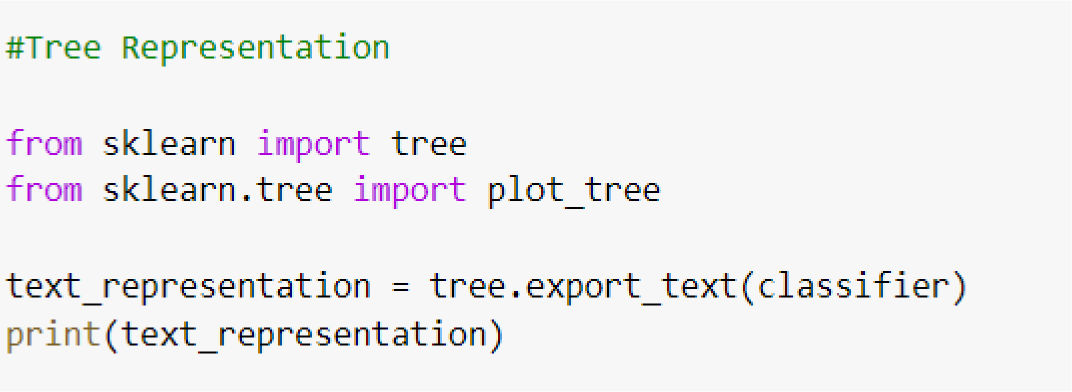


```
Confusion Matrix:
[[75  7]
 [ 6 49]]
-----------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           2       0.93      0.91      0.92        82
           4       0.88      0.89      0.88        55

    accuracy                           0.91       137
   macro avg       0.90      0.90      0.90       137
weighted avg       0.91      0.91      0.91       137
```

Decision Tree Visualization :

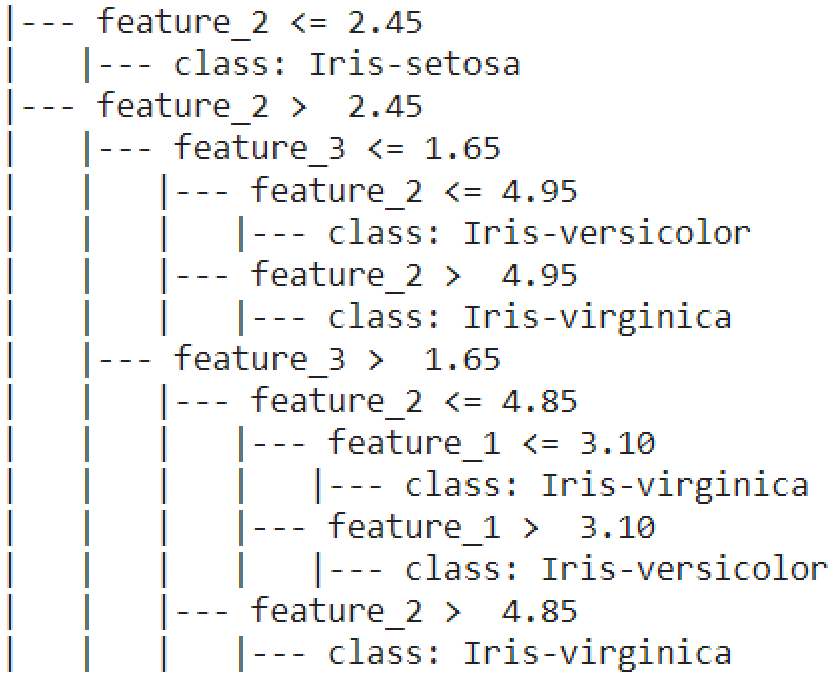
> **Tree Representation :**

Code :



```
#Tree Representation

from sklearn import tree
from sklearn.tree import plot_tree

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

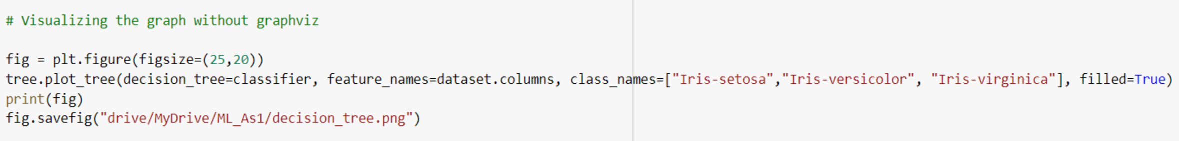
Output :



```
|--- feature_2 <= 2.45
|   |--- class: Iris-setosa
|--- feature_2 >  2.45
|   |--- feature_3 <= 1.65
|   |   |--- feature_2 <= 4.95
|   |   |   |--- class: Iris-versicolor
|   |   |--- feature_2 >  4.95
|   |   |   |--- class: Iris-virginica
|   |--- feature_3 >  1.65
|   |   |--- feature_2 <= 4.85
|   |   |   |--- feature_1 <= 3.10
|   |   |   |   |--- class: Iris-virginica
|   |   |   |--- feature_1 >  3.10
|   |   |   |   |--- class: Iris-versicolor
|   |   |--- feature_2 >  4.85
|   |   |   |--- class: Iris-virginica
```

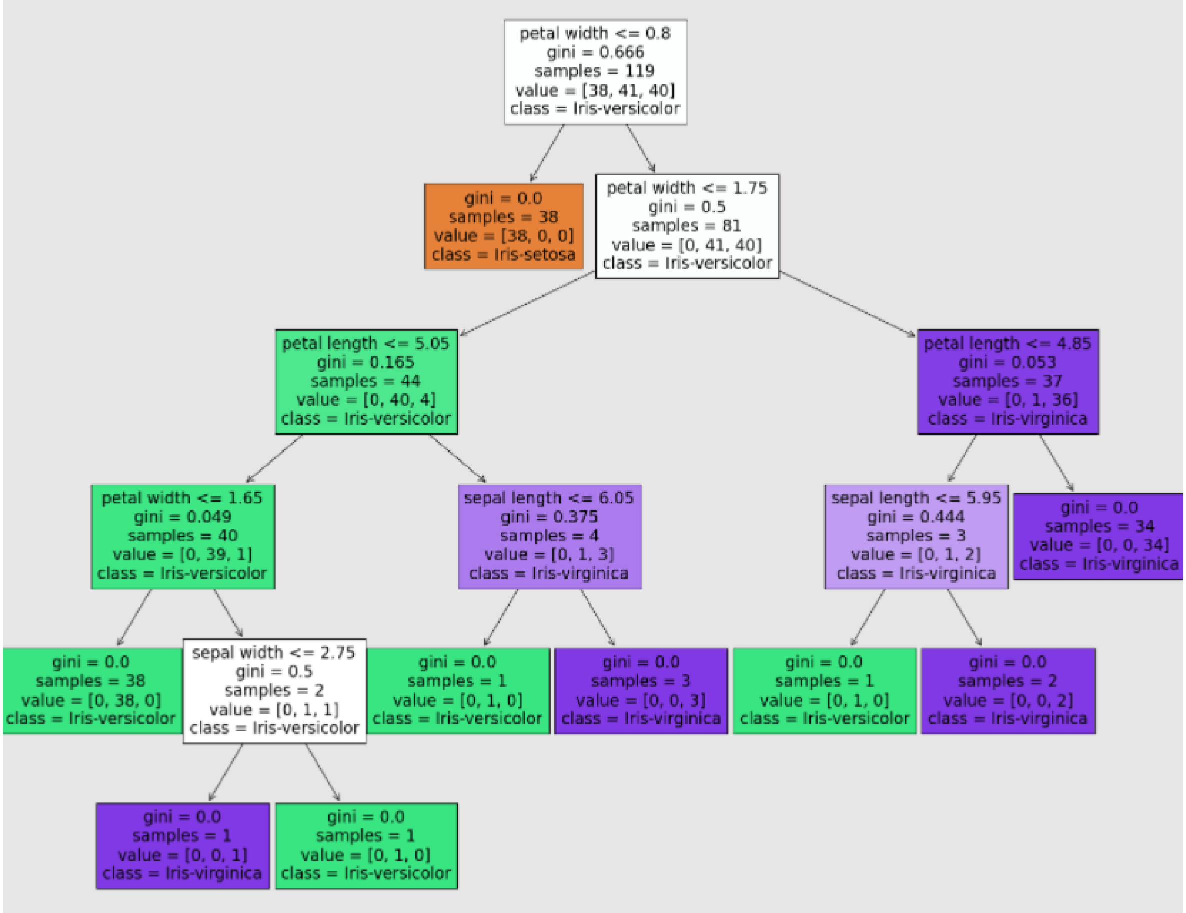
> **Visualizing Graph :**

Code :



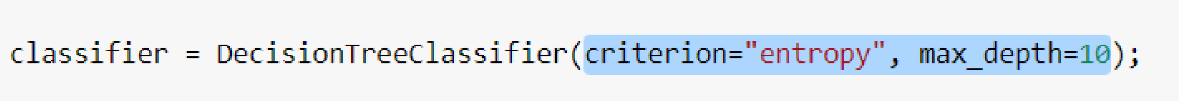
```
# Visualizing the graph without graphviz

fig = plt.figure(figsize=(25,20))
tree.plot_tree(decision_tree=classifier, feature_names=dataset.columns, class_names=["Iris-setosa","Iris-versicolor", "Iris-virginica"], filled=True)
print(fig)
fig.savefig("drive/MyDrive/ML_As1/decision_tree.png")
```

Output :



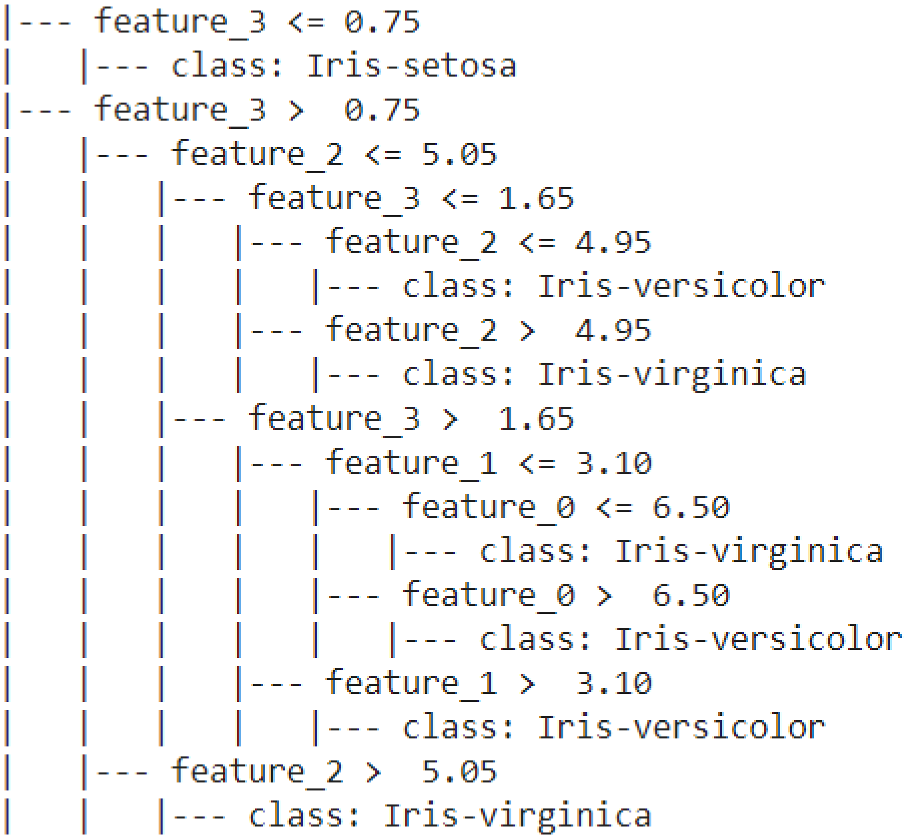
With “entropy” criterion :

> **criterion="entropy", max_depth=10**

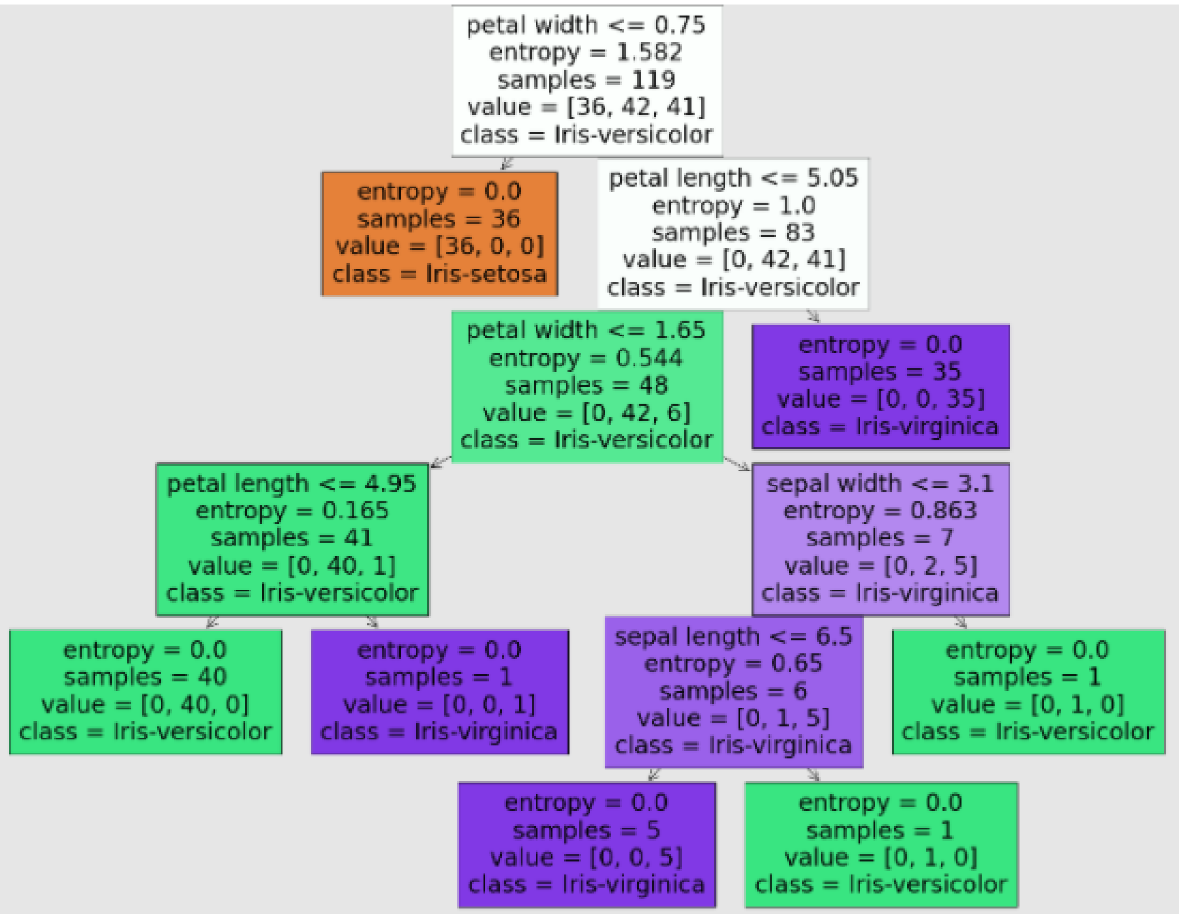


```
classifier = DecisionTreeClassifier(criterion="entropy", max_depth=10);
```

<u>Output</u> :

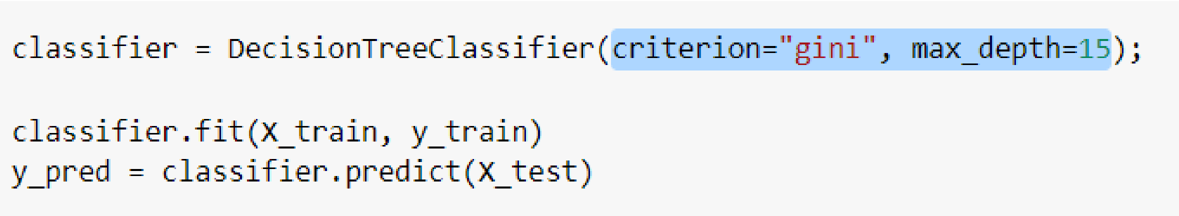


```
|--- feature_3 <= 0.75
|   |--- class: Iris-setosa
|--- feature_3 >  0.75
|   |--- feature_2 <= 5.05
|   |   |--- feature_3 <= 1.65
|   |   |   |--- feature_2 <= 4.95
|   |   |   |   |--- class: Iris-versicolor
|   |   |   |--- feature_2 >  4.95
|   |   |   |   |--- class: Iris-virginica
|   |   |--- feature_3 >  1.65
|   |   |   |--- feature_1 <= 3.10
|   |   |   |   |--- feature_0 <= 6.50
|   |   |   |   |   |--- class: Iris-virginica
|   |   |   |   |--- feature_0 >  6.50
|   |   |   |   |   |--- class: Iris-versicolor
|   |   |   |--- feature_1 >  3.10
|   |   |   |   |--- class: Iris-versicolor
|   |--- feature_2 >  5.05
|   |   |--- class: Iris-virginica
```



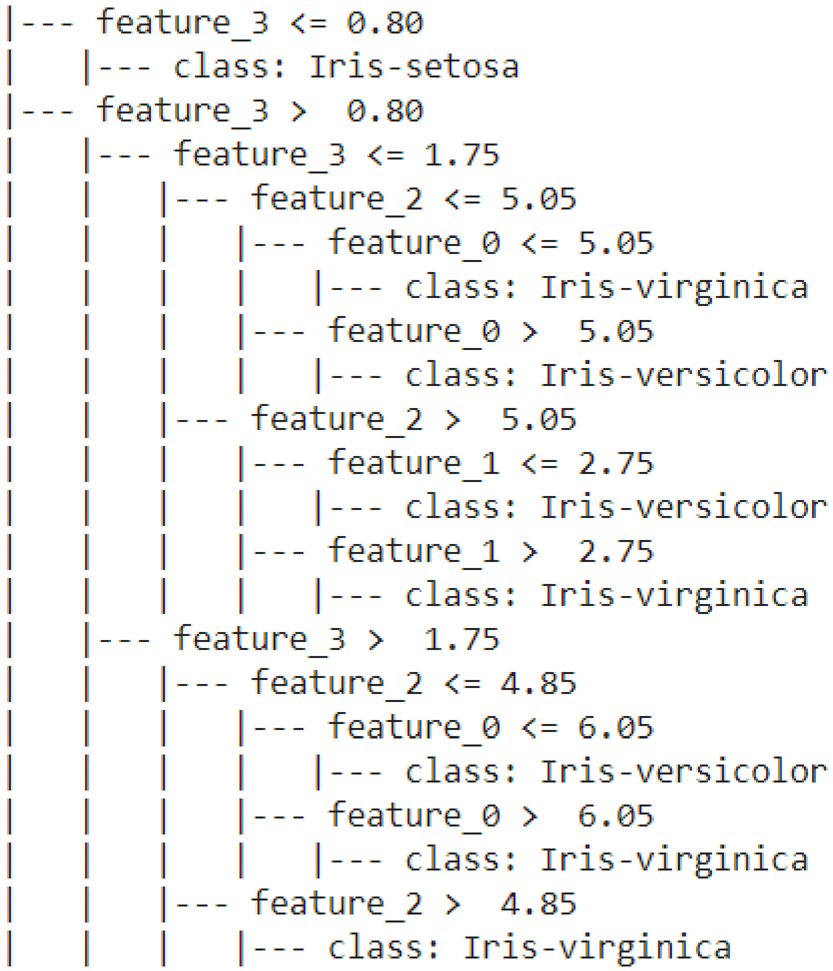
With “gini” criterion :

> **criterion="gini", max_depth=15**

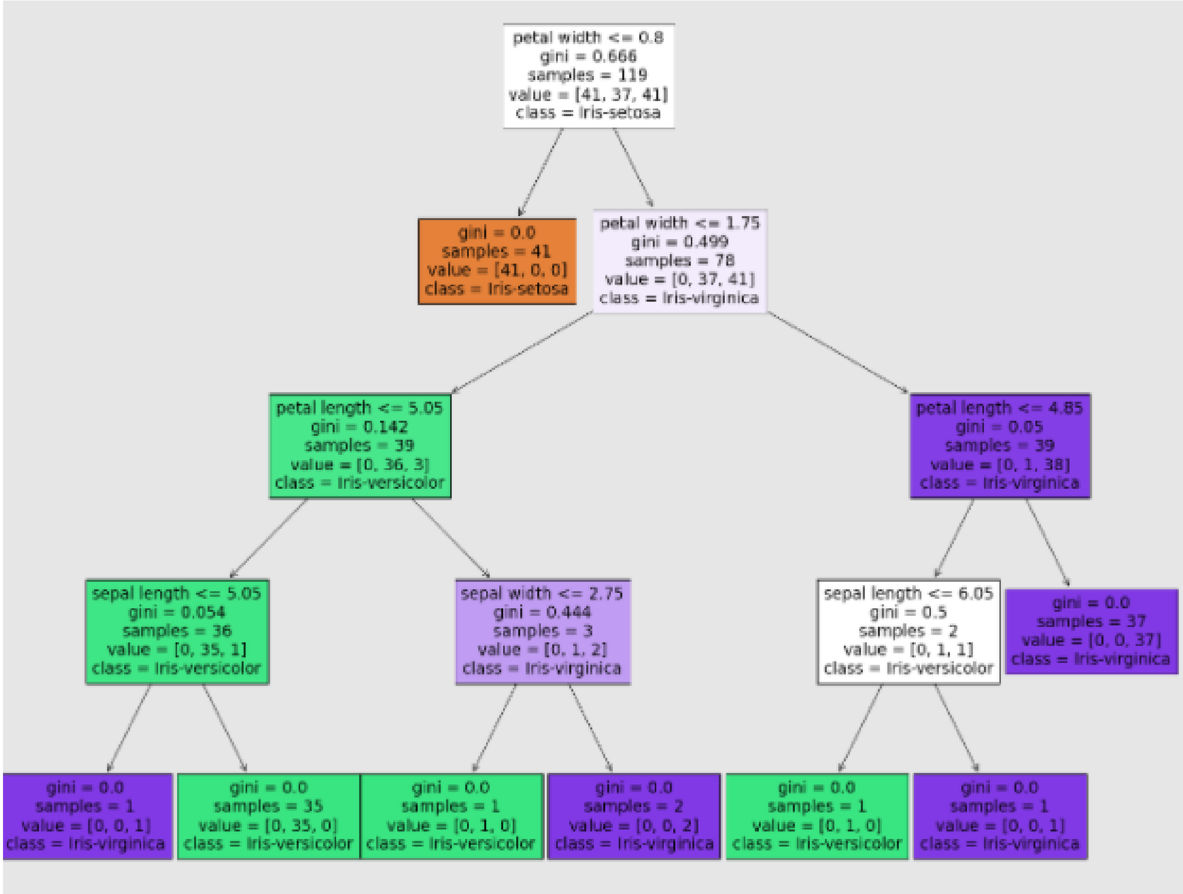


```
classifier = DecisionTreeClassifier(criterion="gini", max_depth=15);

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

Output :



```
|--- feature_3 <= 0.80
|   |--- class: Iris-setosa
|--- feature_3 >  0.80
|   |--- feature_3 <= 1.75
|   |   |--- feature_2 <= 5.05
|   |   |   |--- feature_0 <= 5.05
|   |   |   |   |--- class: Iris-virginica
|   |   |   |--- feature_0 >  5.05
|   |   |   |   |--- class: Iris-versicolor
|   |   |--- feature_2 >  5.05
|   |   |   |--- feature_1 <= 2.75
|   |   |   |   |--- class: Iris-versicolor
|   |   |   |--- feature_1 >  2.75
|   |   |   |   |--- class: Iris-virginica
|   |--- feature_3 >  1.75
|   |   |--- feature_2 <= 4.85
|   |   |   |--- feature_0 <= 6.05
|   |   |   |   |--- class: Iris-versicolor
|   |   |   |--- feature_0 >  6.05
|   |   |   |   |--- class: Iris-virginica
|   |   |--- feature_2 >  4.85
|   |   |   |--- class: Iris-virginica
```



1. Employ Naive Bayes (Gaussian, Multinomial & Bernoulli) classifier and show classification results (Accuracy, Precision, Recall, F-score, confusion matrix) with and without parameter tuning

Without parameter tuning :

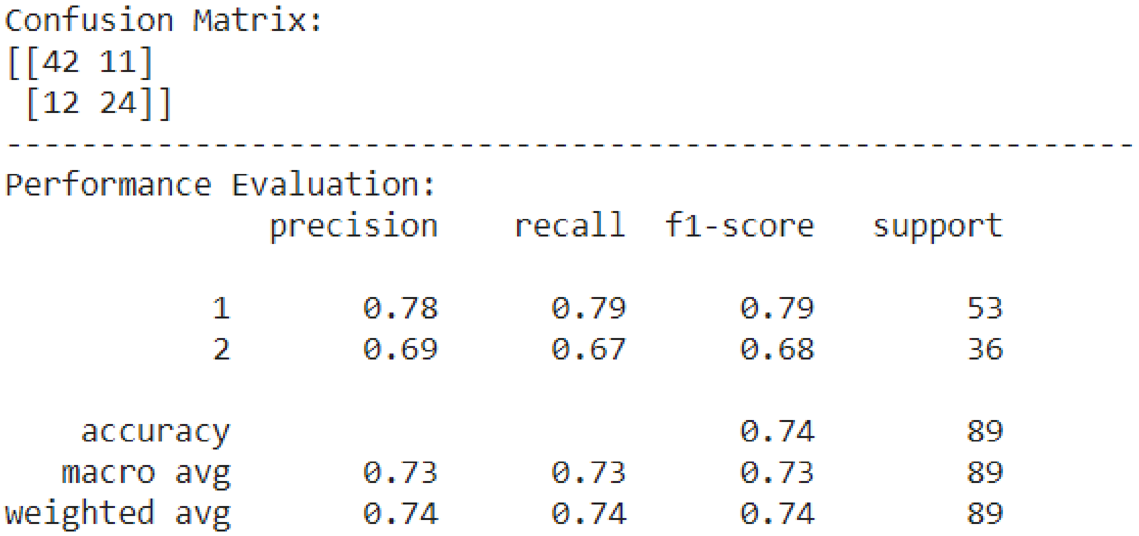
The following code contains the python program to Use Decision Tree classifier for all the three datasets.

To run this program on each of the three classifier, we have to remove the comment on the respective classifier and comment out all the remaining classifier.

```
#Decision Tree for Classification

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Dataset Preparation

# 2. Diabetes dataset:
dataset = pd.read_csv("drive/MyDrive/ML_As1/diabetes.tab.txt",sep='\t')

X = dataset.drop(columns=['SEX'])
y = dataset["SEX"]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

# Classification

# Multinomial
from sklearn.naive_bayes import MultinomialNB
```

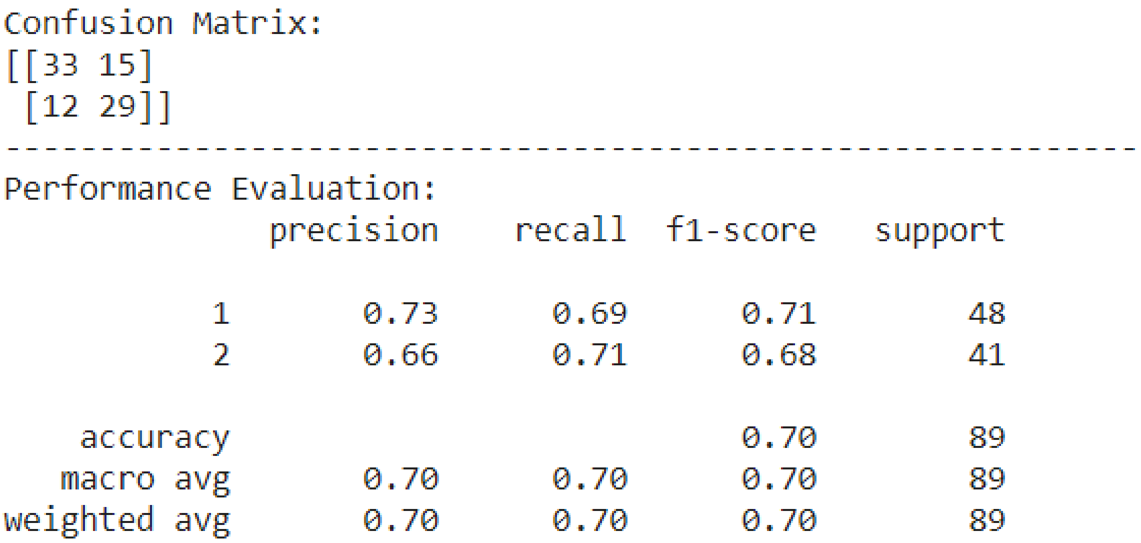
```
classifier = MultinomialNB().fit(X_train, y_train)

# Gaussian
# from sklearn.naive_bayes import GaussianNB
# classifier = GaussianNB().fit(X_train, y_train)

# Bernoulli
# from sklearn.naive_bayes import BernoulliNB
# classifier = BernoulliNB().fit(X_train, y_train)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

# Evaluation of Classifier Performance

from sklearn.metrics import classification_report, confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("------------------------------------------------------------")

print("Performance Evaluation:")
print(classification_report(y_test, y_pred))
```

Gaussian Classifier :



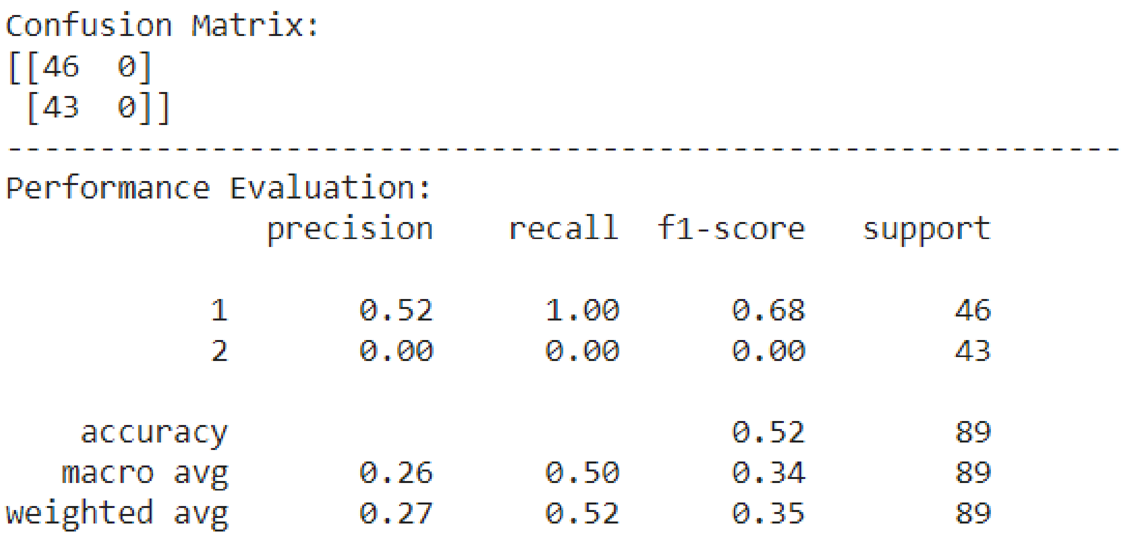
```
Confusion Matrix:
[[42 11]
 [12 24]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           1       0.78      0.79      0.79        53
           2       0.69      0.67      0.68        36

    accuracy                           0.74        89
   macro avg       0.73      0.73      0.73        89
weighted avg       0.74      0.74      0.74        89
```

Multinomial Classifier :



```
Confusion Matrix:
[[33 15]
 [12 29]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           1       0.73      0.69      0.71        48
           2       0.66      0.71      0.68        41

    accuracy                           0.70        89
   macro avg       0.70      0.70      0.70        89
weighted avg       0.70      0.70      0.70        89
```

Bernoulli Classifier :



```
Confusion Matrix:
[[46  0]
 [43  0]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           1       0.52      1.00      0.68        46
           2       0.00      0.00      0.00        43

    accuracy                           0.52        89
   macro avg       0.26      0.50      0.34        89
weighted avg       0.27      0.52      0.35        89
```

With parameter tuning :

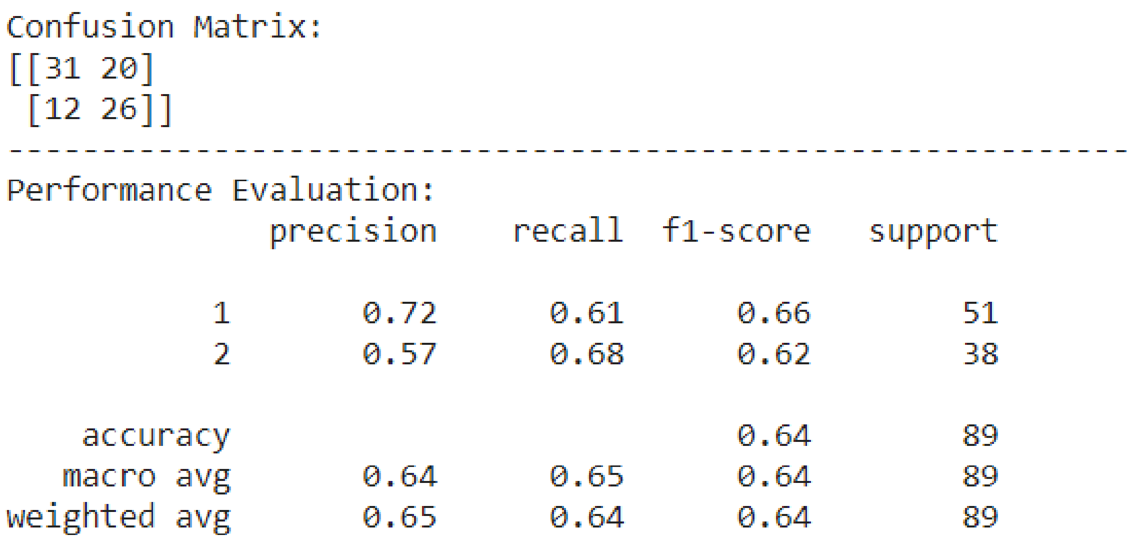
The following code contains the python program to Use Decision Tree classifier for all the three datasets.

To run this program on each of the three classifier, we have to remove the comment on the respective classifier and comment out all the remaining classifier.

```
#Decision Tree for Classification

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Dataset Preparation

dataset = pd.read_csv("drive/MyDrive/ML_As1/diabetes.tab.txt",sep='\t')

X = dataset.drop(columns=['SEX'])
y = dataset["SEX"]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

```
# Classification

# Gaussian
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB(priors=None, var_smoothing=1e-05).fit(X_train, y_train)

# # Multinomial
# from sklearn.naive_bayes import MultinomialNB
# classifier = MultinomialNB(alpha=2.5, fit_prior=True, class_prior=None).fit(X_train,
y_train)

# Bernoulli
# from sklearn.naive_bayes import BernoulliNB
# classifier = BernoulliNB(alpha=1.0, binarize=0.0, fit_prior=True,
class_prior=None).fit(X_train, y_train)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

# Evaluation of Classifier Performance

from sklearn.metrics import classification_report, confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("------------------------------------------------------------")

print("Performance Evaluation:")
print(classification_report(y_test, y_pred))
```

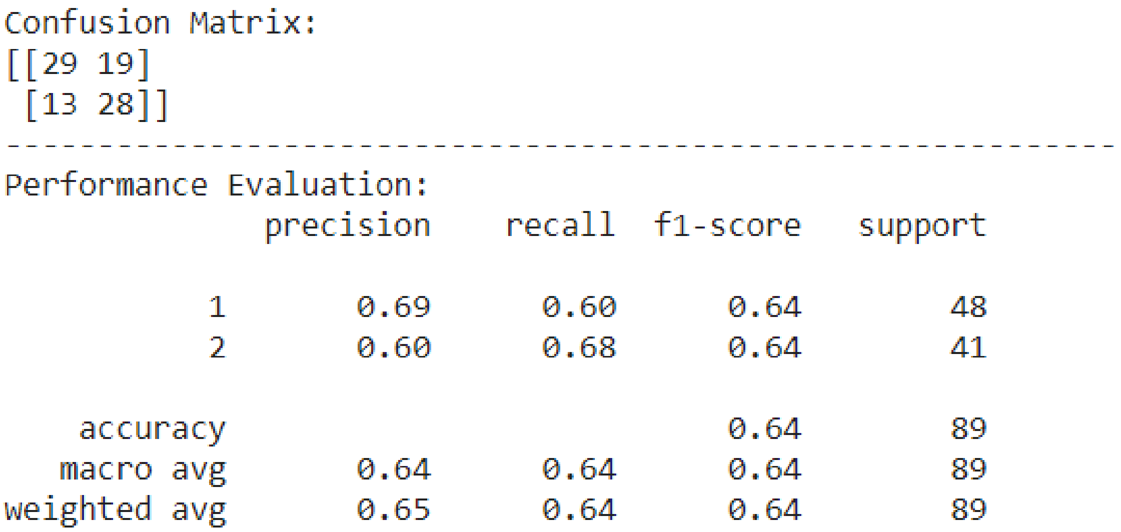
Gaussian Classifier :



```
Confusion Matrix:
[[31 20]
 [12 26]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           1       0.72      0.61      0.66        51
           2       0.57      0.68      0.62        38

    accuracy                           0.64        89
   macro avg       0.64      0.65      0.64        89
weighted avg       0.65      0.64      0.64        89
```

Multinomial Classifier :

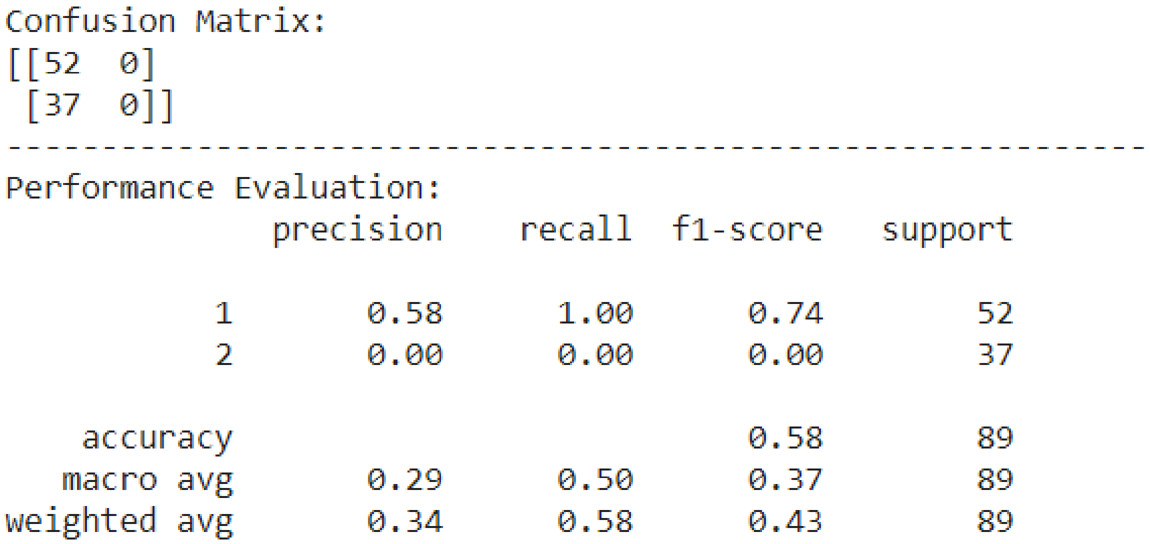
```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB(alpha=2.5, fit_prior=True, class_prior=None).fit(X_train,
y_train)
```



```
Confusion Matrix:
[[29 19]
 [13 28]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           1       0.69      0.60      0.64        48
           2       0.60      0.68      0.64        41

    accuracy                           0.64        89
   macro avg       0.64      0.64      0.64        89
weighted avg       0.65      0.64      0.64        89
```

Bernoulli Classifier :

```
from sklearn.naive_bayes import BernoulliNB
classifier = BernoulliNB(alpha=1.0, binarize=0.0, fit_prior=True,
class_prior=None).fit(X_train, y_train)
```



```
Confusion Matrix:
[[52  0]
 [37  0]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           1       0.58      1.00      0.74        52
           2       0.00      0.00      0.00        37

    accuracy                           0.58        89
   macro avg       0.29      0.50      0.37        89
weighted avg       0.34      0.58      0.43        89
```