19SE02IT058 SEIT4013

In [1]:

```
import numpy as np # linear algebra
import os
```

In [2]:

```
import pandas as pd                               # File Handling
import numpy as np                                # Mathematical Computation
```

In [3]:

```
from sklearn.model_selection import train_test_split            # Splitting Dataset int
```

In [4]:

```
from sklearn.tree import DecisionTreeClassifier                 # For implementing Deci
```

In [5]:

```
from sklearn.metrics import accuracy_score                      # For calculating accur
from sklearn.metrics import classification_report               # For evaluating the mo
```

In [6]:

```
from sklearn import tree                                        # Visualizing Decision
```

In [7]:

```
Dataset = pd.read_csv("Iris.csv")
```

In [8]:

```
Dataset = Dataset.dropna()                        # Dropping empty rows
```

In [9]:

```
Dataset.head()
```

Out[9]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [10]:

```
Dataset.shape
```

Out[10]:

```
(150, 6)
```

In [11]:

```
Dataset["Species"].unique()                         # Unique values of Species
```

Out[11]:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [12]:

```
Dataset = Dataset.replace(to_replace ="Iris-setosa",          value ="0")
Dataset = Dataset.replace(to_replace ="Iris-versicolor",      value ="1")
Dataset = Dataset.replace(to_replace ="Iris-virginica",       value ="2")
```

In [13]:

```
X = np.array(Dataset[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']])
Y = np.array(Dataset["Species"])
```

In [14]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state = 100
```

In [15]:

```
clf_gini = DecisionTreeClassifier(criterion = "gini",                # Criterion
                                  max_depth = 5,                     # Max Height of Tree
                                  min_samples_leaf = 3,              # Maximum Leaf samples
                                  random_state = 100)
```

In [16]:

```
clf_gini.fit(X_train, Y_train)                                       # Training the Model
```

Out[16]:

```
DecisionTreeClassifier(max_depth=5, min_samples_leaf=3, random_state=100)
```

In [17]:

```
clf_entropy = DecisionTreeClassifier(criterion = "entropy",          # Criterion
                                     max_depth = 5,                  # Max Height of Tree
                                     min_samples_leaf = 3,           # Max Leaf samples
                                     random_state = 100)
```

In [18]:

```
clf_entropy.fit(X_train, Y_train)                                  # Training the model
```

Out[18]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=5, min_samples_leaf=3,
                       random_state=100)
```

In [19]:

```
y_pred_gini = clf_gini.predict(X_test)                              # Performing Predicti
```

In [20]:

```
print ("Accuracy : ", accuracy_score(Y_test,y_pred_gini)*100)       # Evaulating predicti
print ("Report : ",  classification_report(Y_test, y_pred_gini))
```

```
Accuracy :  96.66666666666667
Report :                precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.83      0.91         6
           2       0.93      1.00      0.96        13

    accuracy                           0.97        30
   macro avg       0.98      0.94      0.96        30
weighted avg       0.97      0.97      0.97        30
```

In [21]:

```
y_pred_entropy = clf_entropy.predict(X_test)                              # Performing Pr
```

In [22]:

```
print ("Accuracy : ", accuracy_score(Y_test,y_pred_entropy)*100)          # Evaulating pr
print ("Report : ",  classification_report(Y_test, y_pred_entropy))
```

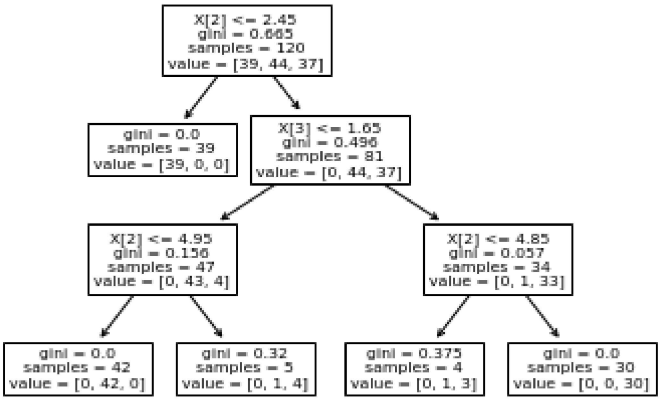
```
Accuracy :  96.66666666666667
Report :                precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.83      0.91         6
           2       0.93      1.00      0.96        13

    accuracy                           0.97        30
   macro avg       0.98      0.94      0.96        30
weighted avg       0.97      0.97      0.97        30
```

In [23]:

```
tree.plot_tree(clf_gini)
```

Out[23]:

```
[Text(0.375, 0.875, 'X[2] <= 2.45\ngini = 0.665\nsamples = 120\nvalue = [39,
44, 37]'),
 Text(0.25, 0.625, 'gini = 0.0\nsamples = 39\nvalue = [39, 0, 0]'),
 Text(0.5, 0.625, 'X[3] <= 1.65\ngini = 0.496\nsamples = 81\nvalue = [0, 44,
37]'),
 Text(0.25, 0.375, 'X[2] <= 4.95\ngini = 0.156\nsamples = 47\nvalue = [0, 4
3, 4]'),
 Text(0.125, 0.125, 'gini = 0.0\nsamples = 42\nvalue = [0, 42, 0]'),
 Text(0.375, 0.125, 'gini = 0.32\nsamples = 5\nvalue = [0, 1, 4]'),
 Text(0.75, 0.375, 'X[2] <= 4.85\ngini = 0.057\nsamples = 34\nvalue = [0, 1,
33]'),
 Text(0.625, 0.125, 'gini = 0.375\nsamples = 4\nvalue = [0, 1, 3]'),
 Text(0.875, 0.125, 'gini = 0.0\nsamples = 30\nvalue = [0, 0, 30]')]
```

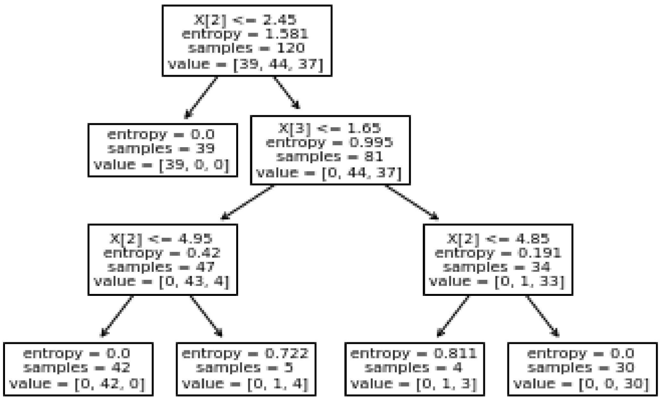


In [24]:

```
tree.plot_tree(clf_entropy)
```

Out[24]:

```
[Text(0.375, 0.875, 'X[2] <= 2.45\nentropy = 1.581\nsamples = 120\nvalue =
[39, 44, 37]'),
 Text(0.25, 0.625, 'entropy = 0.0\nsamples = 39\nvalue = [39, 0, 0]'),
 Text(0.5, 0.625, 'X[3] <= 1.65\nentropy = 0.995\nsamples = 81\nvalue = [0,
44, 37]'),
 Text(0.25, 0.375, 'X[2] <= 4.95\nentropy = 0.42\nsamples = 47\nvalue = [0,
43, 4]'),
 Text(0.125, 0.125, 'entropy = 0.0\nsamples = 42\nvalue = [0, 42, 0]'),
 Text(0.375, 0.125, 'entropy = 0.722\nsamples = 5\nvalue = [0, 1, 4]'),
 Text(0.75, 0.375, 'X[2] <= 4.85\nentropy = 0.191\nsamples = 34\nvalue = [0,
1, 33]'),
 Text(0.625, 0.125, 'entropy = 0.811\nsamples = 4\nvalue = [0, 1, 3]'),
 Text(0.875, 0.125, 'entropy = 0.0\nsamples = 30\nvalue = [0, 0, 30]')]
```


X[2] <= 2.45
entropy = 1.581
samples = 120
value = [39, 44, 37]
entropy = 0.0
samples = 39
value = [39, 0, 0]
X[3] <= 1.65
entropy = 0.995
samples = 81
value = [0, 44, 37]
X[2] <= 4.95
entropy = 0.42
samples = 47
value = [0, 43, 4]
X[2] <= 4.85
entropy = 0.191
samples = 34
value = [0, 1, 33]
entropy = 0.0
samples = 42
value = [0, 42, 0]
entropy = 0.722
samples = 5
value = [0, 1, 4]
entropy = 0.811
samples = 4
value = [0, 1, 3]
entropy = 0.0
samples = 30
value = [0, 0, 30]