In [1]:
```
#importing necessary libraries

import pandas as pd
import numpy as np
import seaborn as sns
import warnings
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:
```
# reading tennis play dataset
tennis = pd.read_csv('https://gist.githubusercontent.com/DiogoRibeiro7/c6590d0cf1
```

In [3]:
```
#finding first 5 rows of the dataset
tennis.head()
```

Out[3]:

	Outlook	Temperature	Humidity	Wind	Play Tennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes

In [4]:
```
#finding the shape of the data
tennis.shape
```

Out[4]:
```
(14, 5)
```

In [5]:
```
#statistical analysis of data
tennis.describe().T
```

Out[5]:

	count	unique	top	freq
Outlook	14	3	Sunny	5
Temperature	14	3	Mild	6
Humidity	14	2	High	7
Wind	14	2	Weak	8
Play Tennis	14	2	Yes	9

In [6]:
```
#finding the number of columns
tennis.columns
```

Out[6]:
```
Index(['Outlook', 'Temperature', 'Humidity', 'Wind', 'Play Tennis'], dtype='obj
ect')
```

In [7]:
```
#finding the null values
tennis.isnull().sum()
```

Out[7]:
```
Outlook        0
Temperature    0
Humidity       0
Wind           0
Play Tennis    0
dtype: int64
```

Performing Lable encoding

```
#since our data is in categorial format, need to change that to numerical
format using lable encoding
```

In [8]:
```
#importing lable encoder from sklearn
from sklearn.preprocessing import LabelEncoder
```

In [9]:
```
le = LabelEncoder()
```

In [10]:
```
le
```

Out[10]:
```
LabelEncoder()
```

In [11]:
```
tennis['Outlook'] = le.fit_transform(tennis['Outlook'])
tennis['Humidity'] = le.fit_transform(tennis['Humidity'])
tennis['Wind'] = le.fit_transform(tennis['Wind'])
tennis['Temperature'] = le.fit_transform(tennis['Temperature'])
tennis['Play Tennis'] = le.fit_transform(tennis['Play Tennis'])
```

In [12]:
```
#now checking the dataset
tennis.head()
```

Out[12]:

	Outlook	Temperature	Humidity	Wind	Play Tennis
0	2	1	0	1	0
1	2	1	0	0	0
2	0	1	0	1	1
3	1	2	0	1	1
4	1	0	1	1	1

In [13]:
```
#Independent and dependent variable
x = tennis.iloc[:,:-1]
y = tennis.iloc[:,-1]
```

In [14]:
```
x
```

Out[14]:

	Outlook	Temperature	Humidity	Wind
0	2	1	0	1
1	2	1	0	0
2	0	1	0	1
3	1	2	0	1
4	1	0	1	1
5	1	0	1	0
6	0	0	1	0
7	2	2	0	1
8	2	0	1	1
9	1	2	1	1
10	2	2	1	0
11	0	2	0	0
12	0	1	1	1
13	1	2	0	0

In [15]:
```
y
```

Out[15]:
```
0     0
1     0
2     1
3     1
4     1
5     0
6     1
7     0
8     1
9     1
10    1
11    1
12    1
13    0
Name: Play Tennis, dtype: int32
```

In [16]:
```
#permorming train, test split
from sklearn.model_selection import train_test_split
```

In [17]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

In [18]:
```
x_train.shape, x_test.shape
```

Out[18]:
```
((9, 4), (5, 4))
```

Model building

In [19]:
```
#importing Decision tree classifer from sklearn
from sklearn.tree import DecisionTreeClassifier
```

In [67]:
```
dst = DecisionTreeClassifier(criterion = 'entropy', random_state =100)
```

In [68]:
```
dst
```

Out[68]:
```
DecisionTreeClassifier(criterion='entropy', random_state=100)
```

In [21]:
```
#fitting our data in to model
dst.fit(x_train, y_train)
```

Out[21]:
```
DecisionTreeClassifier(criterion='entropy')
```

In [22]:
```
#predicting the x test data
y_pred = dst.predict(x_test)
```

In [23]:
```
x_test
```

Out[23]:

	Outlook	Temperature	Humidity	Wind
12	0	1	1	1
9	1	2	1	1
11	0	2	0	0
3	1	2	0	1
1	2	1	0	0

In [24]:
```
y_test
```

Out[24]:
```
12    1
9     1
11    1
3     1
1     0
Name: Play Tennis, dtype: int32
```

In [25]:
```
#checking the training accuracy
dst.score(x_train, y_train)
```

Out[25]:
```
1.0
```

In [26]:
```
#importing accuracy score from sklearn
from sklearn.metrics import accuracy_score
```

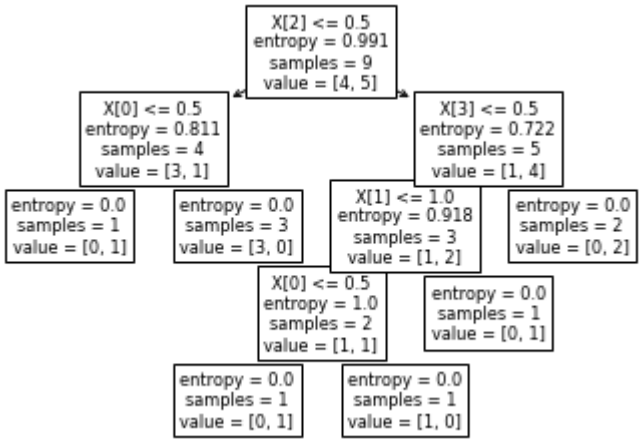
In [27]:
```
accuracy_score(y_test, y_pred)
```

Out[27]:
```
0.8
```

In [28]:
```
from sklearn import tree
```

In [29]:
```
tree.plot_tree(dst)
```

Out[29]:
```
[Text(0.5, 0.9, 'X[2] <= 0.5\nentropy = 0.991\nsamples = 9\nvalue = [4, 5]'),
 Text(0.25, 0.7, 'X[0] <= 0.5\nentropy = 0.811\nsamples = 4\nvalue = [3, 1]'),
 Text(0.125, 0.5, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.375, 0.5, 'entropy = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(0.75, 0.7, 'X[3] <= 0.5\nentropy = 0.722\nsamples = 5\nvalue = [1, 4]'),
 Text(0.625, 0.5, 'X[1] <= 1.0\nentropy = 0.918\nsamples = 3\nvalue = [1, 2]'),
 Text(0.5, 0.3, 'X[0] <= 0.5\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(0.375, 0.1, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.625, 0.1, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.75, 0.3, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.875, 0.5, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]')]
```



In [30]:
```
#importing Gridsearch CV hyperperameter
from sklearn.model_selection import GridSearchCV
```

In [31]:
```
grid_params = {'criterion' : ['gini', 'entropy'],
               'max_depth' : range(2, 32,1),
               'min_samples_leaf' : range(1,10,1),
               'min_samples_split' : range(2,10,1),
               'splitter':['best', 'random']}
```

In [32]:
```
grid_search = GridSearchCV(estimator=dst,
             param_grid=grid_params, cv = 5)
```

In [33]:
```
#fitting data into gridsearch
grid_search.fit(x_train, y_train)
```

Out[33]:
```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(criterion='entropy'),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 32),
                         'min_samples_leaf': range(1, 10),
                         'min_samples_split': range(2, 10),
                         'splitter': ['best', 'random']})
```

In [34]:
```
#finding the best parameters
grid_search.best_params_
```

Out[34]:
```
{'criterion': 'gini',
 'max_depth': 2,
 'min_samples_leaf': 3,
 'min_samples_split': 2,
 'splitter': 'best'}
```

In [35]:
```
model_with_best_params = DecisionTreeClassifier(criterion = 'gini', max_depth =2,
)
```

In [36]:
```
#fitting train data in to best_parameters
model_with_best_params.fit(x_train, y_train)
```

Out[36]:
```
DecisionTreeClassifier(max_depth=2, min_samples_split=4, splitter='random')
```

In [37]:
```
#predicting the x_test
y_pred2 = model_with_best_params.predict(x_test)
```

In [38]:
```
accuracy_score(y_test, y_pred2)
```

Out[38]:
```
0.6
```

In [39]:
```
data_p=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
data_p
```

Out[39]:

	Actual	Predicted
12	1	1
9	1	1
11	1	1
3	1	0
1	0	0

In [40]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [43]:
```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[1 0]
 [1 3]]
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       1.00      0.75      0.86         4

    accuracy                           0.80         5
   macro avg       0.75      0.88      0.76         5
weighted avg       0.90      0.80      0.82         5
```