Implementing Decision Tree using Scikit Learn

Reg.No : 21BCE1964

In [1]:
```
#numpy and pandas initialization
import numpy as np
import pandas as pd
```

In [2]:
```
#Loading the PlayTennis data
PlayTennis = pd.read_csv("../input/PlayTennis.csv")
```

Dataset Description

In [3]:
```
PlayTennis
```

Out[3]:

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes
10	sunny	mild	normal	True	yes
11	overcast	mild	high	True	yes
12	overcast	hot	normal	False	yes
13	rainy	mild	high	True	no

It is easy to implement Decision Tree with numerical values. We can convert all the non numerical values into numerical values using LabelEncoder

In [4]:
```
from sklearn.preprocessing import LabelEncoder
Le = LabelEncoder()

PlayTennis['outlook'] = Le.fit_transform(PlayTennis['outlook'])
PlayTennis['temp'] = Le.fit_transform(PlayTennis['temp'])
PlayTennis['humidity'] = Le.fit_transform(PlayTennis['humidity'])
PlayTennis['windy'] = Le.fit_transform(PlayTennis['windy'])
PlayTennis['play'] = Le.fit_transform(PlayTennis['play'])
```

In [5]:
```
PlayTennis
```

Out[5]:

	outlook	temp	humidity	windy	play
0	2	1	0	0	0
1	2	1	0	1	0
2	0	1	0	0	1
3	1	2	0	0	1
4	1	0	1	0	1
5	1	0	1	1	0
6	0	0	1	1	1
7	2	2	0	0	0
8	2	0	1	0	1
9	1	2	1	0	1
10	2	2	1	1	1
11	0	2	0	1	1
12	0	1	1	0	1
13	1	2	0	1	0

Methodology

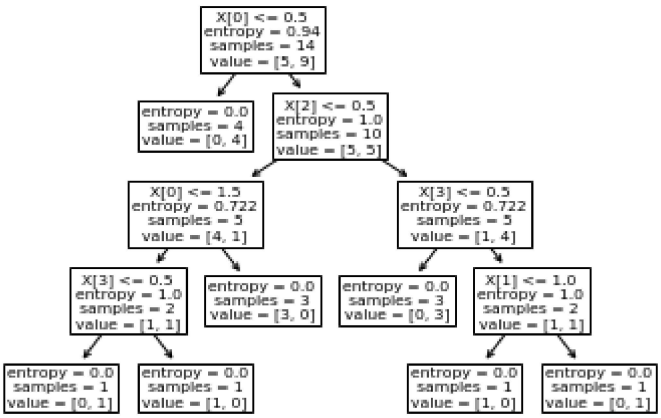
- Lets split the training data and its coresponding prediction values.
- y - holds all the decisions.
- X - holds the training data.

In [6]:
```
y = PlayTennis['play']
X = PlayTennis.drop(['play'],axis=1)
```

In [7]:
```
# Fitting the model
from sklearn import tree
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X, y)
```

In [8]:
```
# We can visualize the tree using tree.plot_tree
tree.plot_tree(clf)
```

Out[8]:
```
[Text(133.92000000000002, 195.696, 'X[0] <= 0.5\nentropy = 0.94\nsamples =
14\nvalue = [5, 9]'),
 Text(100.44000000000001, 152.208, 'entropy = 0.0\nsamples = 4\nvalue = [0,
4]'),
 Text(167.40000000000003, 152.208, 'X[2] <= 0.5\nentropy = 1.0\nsamples = 1
0\nvalue = [5, 5]'),
 Text(100.44000000000001, 108.72, 'X[0] <= 1.5\nentropy = 0.722\nsamples =
5\nvalue = [4, 1]'),
 Text(66.96000000000001, 65.232, 'X[3] <= 0.5\nentropy = 1.0\nsamples = 2\n
value = [1, 1]'),
 Text(33.480000000000004, 21.744, 'entropy = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(100.44000000000001, 21.744, 'entropy = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(133.92000000000002, 65.232, 'entropy = 0.0\nsamples = 3\nvalue = [3,
0]'),
 Text(234.36, 108.72, 'X[3] <= 0.5\nentropy = 0.722\nsamples = 5\nvalue =
[1, 4]'),
 Text(200.88000000000002, 65.232, 'entropy = 0.0\nsamples = 3\nvalue = [0,
3]'),
 Text(267.84000000000003, 65.232, 'X[1] <= 1.0\nentropy = 1.0\nsamples = 2
\nvalue = [1, 1]'),
 Text(234.36, 21.744, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(301.32000000000005, 21.744, 'entropy = 0.0\nsamples = 1\nvalue = [0,
1]')]
```

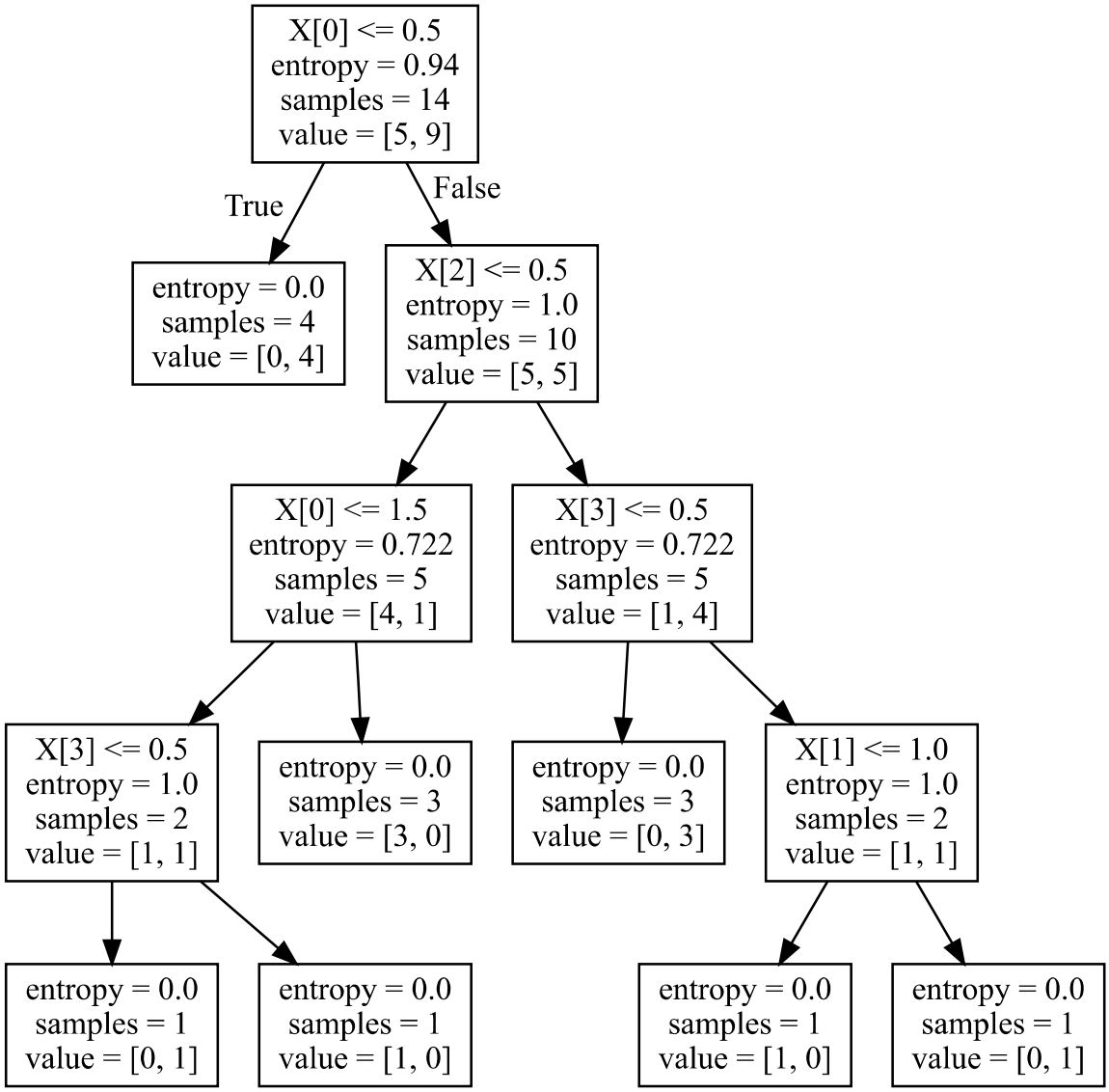


GraphViz (https://www.graphviz.org/) gives a better and clearer Graph.

Results analysis

In [9]:
```
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph
```

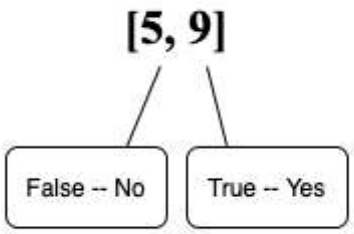
Out[9]:



In the above graph,

- X[0] -> Outlook
- X[1] -> Temperature
- X[2] -> Humidity
- X[3] -> Wind

values



Conclusion

Make the model to predict our train data.

In [10]:
```
# The predictions are stored in X_pred
X_pred = clf.predict(X)
```

In [11]:
```
# verifying if the model has predicted it all right.
X_pred == y
```

Out[11]:
```
0     True
1     True
2     True
3     True
4     True
5     True
6     True
7     True
8     True
9     True
10    True
11    True
12    True
13    True
Name: play, dtype: bool
```

References

https://www.kaggle.com/code/sdk1810/decision-tree-for-playtennis/
(https://www.kaggle.com/code/sdk1810/decision-tree-for-playtennis/)