Decision Tree Classifier

In [1]:

```python
#step 1: import all required libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn import tree
```

```
c:\users\vamsi2001\appdata\local\programs\python\python39\lib\site-packages\num
py\_distributor_init.py:30: UserWarning: loaded more than 1 DLL from .libs:
c:\users\vamsi2001\appdata\local\programs\python\python39\lib\site-packages\num
py\.libs\libopenblas.EL2C6PLE4ZYW3ECEVIV3OXXGRN2NRFM2.gfortran-win_amd64.dll
c:\users\vamsi2001\appdata\local\programs\python\python39\lib\site-packages\num
py\.libs\libopenblas.XWYDX2IKJW2NMTWSFYNGFUWKQU3LYTCZ.gfortran-win_amd64.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
```

In [2]:

```python
df=pd.read_csv('playgolf_data.csv')
df1=df.copy()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Outlook      14 non-null     object
 1   Temperature  14 non-null     object
 2   Humidity     14 non-null     object
 3   Wind         14 non-null     object
 4   PlayGolf     14 non-null     object
dtypes: object(5)
memory usage: 688.0+ bytes
```

In [3]:

```python
df.head()
```

Out[3]:

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

In [4]:

```python
from sklearn.preprocessing import LabelEncoder

# Encode features
label_encoders = {}
for col in ['Outlook', 'Temperature', 'Humidity', 'Wind']:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le  # Save encoders

# Encode target
target_encoder = LabelEncoder()
df['PlayGolf'] = target_encoder.fit_transform(df['PlayGolf'])

print(df)

```

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

In [5]:

```python
#divide X and y variables
X = df.drop(['PlayGolf'], axis=1)
y = df['PlayGolf']
```

In [6]:

```python
# split X and y into training and testing sets

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33,
```

In [7]:

```python
X_train.shape
```

Out[7]:

```
(9, 4)
```

In [8]:

```python
# model Building
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion='entropy', random_state=0)
clf = clf.fit(X, y)
```

In [9]:

```
#predictions
y_pred=clf.predict(X_test)
y_pred
```

Out[9]:

```
array([1, 1, 0, 1, 0])
```

In [10]:

```
#performance Evaluation
from sklearn.metrics import accuracy_score,classification_report,confusion_m
accuracy_score(y_test,y_pred)
```

Out[10]:

```
1.0
```

In [11]:

```
print(classification_report(y_test,y_pred))
```

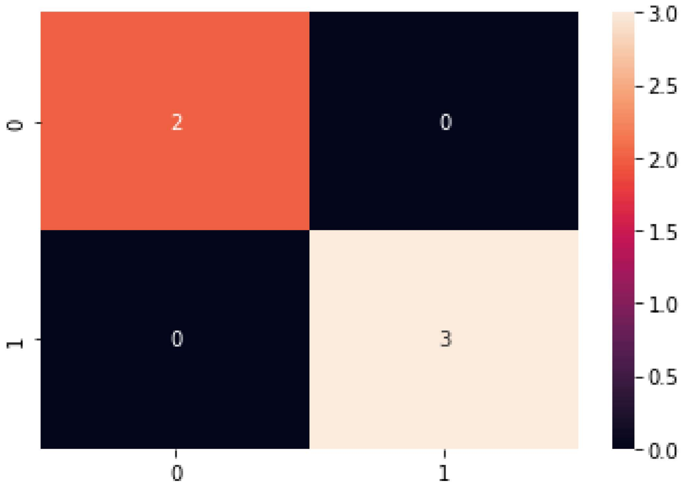
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         3

    accuracy                           1.00         5
   macro avg       1.00      1.00      1.00         5
weighted avg       1.00      1.00      1.00         5
```

In [12]:

```
import seaborn as sns
sns.heatmap(data=confusion_matrix(y_test,y_pred),annot=True)
```

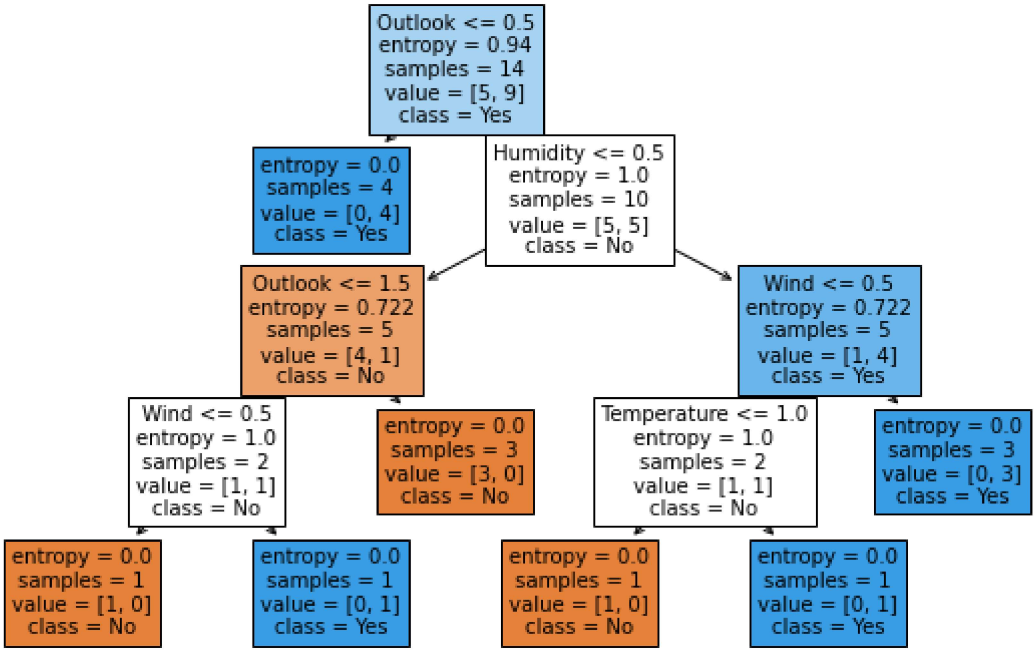
Out[12]:

```
<AxesSubplot:>
```



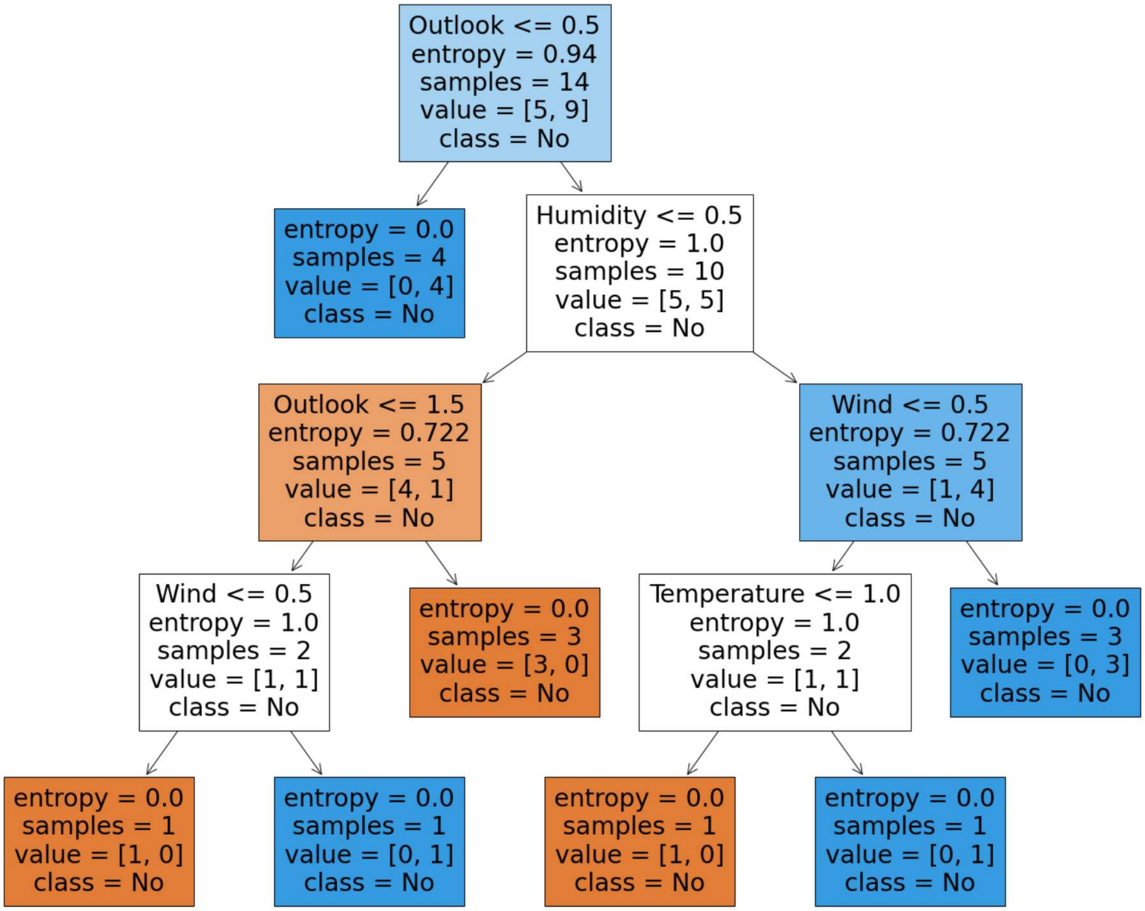
In [13]:

```python
from sklearn import tree
plt.figure(figsize=(10,6))
tree.plot_tree(clf, feature_names=X.columns, class_names=target_encoder.clas
plt.show()
```



In [14]:

```python
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clf,
                   feature_names=df1.columns,
                   class_names=df1['PlayGolf'],
                   filled=True)
```



In [15]:

```python
text_representation = tree.export_text(clf)
print(text_representation)
```

```
|--- feature_0 <= 0.50
|   |--- class: 1
|--- feature_0 >  0.50
|   |--- feature_2 <= 0.50
|   |   |--- feature_0 <= 1.50
|   |   |   |--- feature_3 <= 0.50
|   |   |   |   |--- class: 0
|   |   |   |--- feature_3 >  0.50
|   |   |   |   |--- class: 1
|   |   |--- feature_0 >  1.50
|   |   |   |--- class: 0
|   |--- feature_2 >  0.50
|   |   |--- feature_3 <= 0.50
|   |   |   |--- feature_1 <= 1.00
|   |   |   |   |--- class: 0
|   |   |   |--- feature_1 >  1.00
|   |   |   |   |--- class: 1
|   |   |--- feature_3 >  0.50
|   |   |   |--- class: 1
```

With GINI INDEX

In [16]:

```python
# model Building
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion='gini', random_state=0)
clf = clf.fit(X, y)
```

In [17]:

```python
y_pred=clf.predict(X_test)
y_pred
```

Out[17]:

```
array([1, 1, 0, 1, 0])
```

In [18]:

```python
#performance Evaluation
from sklearn.metrics import accuracy_score,classification_report,confusion_m
accuracy_score(y_test,y_pred)
```

Out[18]:

```
1.0
```

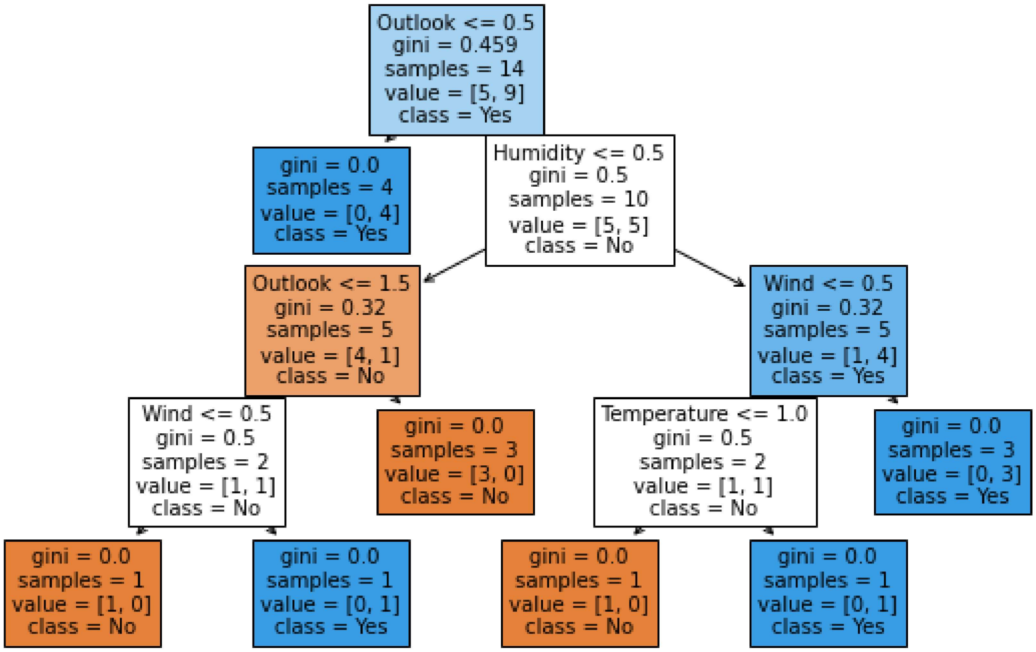
In [19]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         3

    accuracy                           1.00         5
   macro avg       1.00      1.00      1.00         5
weighted avg       1.00      1.00      1.00         5
```

In [20]:

```
from sklearn import tree
plt.figure(figsize=(10,6))
tree.plot_tree(clf, feature_names=X.columns, class_names=target_encoder.clas
plt.show()
```


Outlook <= 0.5
gini = 0.459
samples = 14
value = [5, 9]
class = Yes
gini = 0.0
samples = 4
value = [0, 4]
class = Yes
Humidity <= 0.5
gini = 0.5
samples = 10
value = [5, 5]
class = No
Outlook <= 1.5
gini = 0.32
samples = 5
value = [4, 1]
class = No
Wind <= 0.5
gini = 0.32
samples = 5
value = [1, 4]
class = Yes
Wind <= 0.5
gini = 0.5
samples = 2
value = [1, 1]
class = No
gini = 0.0
samples = 3
value = [3, 0]
class = No
Temperature <= 1.0
gini = 0.5
samples = 2
value = [1, 1]
class = No
gini = 0.0
samples = 3
value = [0, 3]
class = Yes
gini = 0.0
samples = 1
value = [1, 0]
class = No
gini = 0.0
samples = 1
value = [0, 1]
class = Yes
gini = 0.0
samples = 1
value = [1, 0]
class = No
gini = 0.0
samples = 1
value = [0, 1]
class = Yes