Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

Import Play Tennis Data

In [149]:

```
# Author : Dr.Thyagaraju G S , Context Innovations Lab , DEpt of CSE , SDMIT - Ujire
# Date : July 11 2018
import pandas as pd
from pandas import DataFrame
df_tennis = DataFrame.from_csv('C:\\Users\\Dr.Thyagaraju\\Desktop\\Data\\PlayTennis.csv')
print("\n Given Play Tennis Data Set:\n\n", df_tennis)
```

```
 Given Play Tennis Data Set:

    PlayTennis   Outlook Temperature Humidity    Wind
0           No     Sunny         Hot     High    Weak
1           No     Sunny         Hot     High  Strong
2          Yes  Overcast         Hot     High    Weak
3          Yes      Rain        Mild     High    Weak
4          Yes      Rain        Cool   Normal    Weak
5           No      Rain        Cool   Normal  Strong
6          Yes  Overcast        Cool   Normal  Strong
7           No     Sunny        Mild     High    Weak
8          Yes     Sunny        Cool   Normal    Weak
9          Yes      Rain        Mild   Normal    Weak
10         Yes     Sunny        Mild   Normal  Strong
11         Yes  Overcast        Mild     High  Strong
12         Yes  Overcast         Hot   Normal    Weak
13          No      Rain        Mild     High  Strong
```

In [206]:

```
#df_tennis.columns[0]
df_tennis.keys()[0]
```

Out[206]:

```
'PlayTennis'
```

Entropy of the Training Data Set

In [215]:

```
#Function to calculate the entropy of probaility of observations
# -p*log2*p

def entropy(probs):
    import math
    return sum( [-prob*math.log(prob, 2) for prob in probs] )

#Function to calulate the entropy of the given Data Sets/List with respect to target attributes
def entropy_of_list(a_list):
    #print("A-list",a_list)
    from collections import Counter
    cnt = Counter(x for x in a_list)   # Counter calculates the propotion of class
   # print("\nClasses:",cnt)
    #print("No and Yes Classes:",a_list.name,cnt)
    num_instances = len(a_list)*1.0   # = 14
    print("\n Number of Instances of the Current Sub Class is {0}:".format(num_instances ))
    probs = [x / num_instances for x in cnt.values()]  # x means no of YES/NO
    print("\n Classes:",min(cnt),max(cnt))
    print(" \n Probabilities of Class {0} is {1}:".format(min(cnt),min(probs)))
    print(" \n Probabilities of Class {0} is {1}:".format(max(cnt),max(probs)))
    return entropy(probs) # Call Entropy :

# The initial entropy of the YES/NO attribute for our dataset.
print("\n INPUT DATA SET FOR ENTROPY CALCULATION:\n", df_tennis['PlayTennis'])
```

```
print("\n INPUT DATA SET FOR ENTROPY CALCULATION:\n", df_tennis['PlayTennis'])

total_entropy = entropy_of_list(df_tennis['PlayTennis'])

print("\n Total Entropy of PlayTennis Data Set:",total_entropy)
```

```
 INPUT DATA SET FOR ENTROPY CALCULATION:
 0      No
1      No
2     Yes
3     Yes
4     Yes
5      No
6     Yes
7      No
8     Yes
9     Yes
10    Yes
11    Yes
12    Yes
13     No
Name: PlayTennis, dtype: object

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:

 Total Entropy of PlayTennis Data Set: 0.9402859586706309
```

Information Gain of Attributes

In [216]:

```
def information_gain(df, split_attribute_name, target_attribute_name, trace=0):
    print("Information Gain Calculation of ",split_attribute_name)
    '''
    Takes a DataFrame of attributes, and quantifies the entropy of a target
    attribute after performing a split along the values of another attribute.
    '''
    # Split Data by Possible Vals of Attribute:
    df_split = df.groupby(split_attribute_name)
   # for name,group in df_split:
    #    print("Name:\n",name)
     #   print("Group:\n",group)

    # Calculate Entropy for Target Attribute, as well as
    # Proportion of Obs in Each Data-Split
    nobs = len(df.index) * 1.0
   # print("NOBS",nobs)
    df_agg_ent = df_split.agg({target_attribute_name : [entropy_of_list, lambda x: len(x)/nobs] })[
target_attribute_name]
    #print([target_attribute_name])
    #print(" Entropy List ",entropy_of_list)
    #print("DFAGGENT",df_agg_ent)
    df_agg_ent.columns = ['Entropy', 'PropObservations']
    #if trace: # helps understand what fxn is doing:
     #   print(df_agg_ent)

    # Calculate Information Gain:
    new_entropy = sum( df_agg_ent['Entropy'] * df_agg_ent['PropObservations'] )
    old_entropy = entropy_of_list(df[target_attribute_name])
    return old_entropy - new_entropy


print('Info-gain for Outlook is :'+str( information_gain(df_tennis, 'Outlook', 'PlayTennis')),"\n"
)
print('\n Info-gain for Humidity is: ' + str( information_gain(df_tennis, 'Humidity', 'PlayTennis'
)),"\n")
print('\n Info-gain for Wind is:' + str( information_gain(df_tennis, 'Wind', 'PlayTennis')),"\n")
print('\n Info-gain for Temperature is:' + str( information_gain(df_tennis,
```

```
'Temperature','PlayTennis')),"\n")
```

```
Information Gain Calculation of  Outlook

 Number of Instances of the Current Sub Class is 4.0:

 Classes: Yes Yes

 Probabilities of Class Yes is 1.0:

 Probabilities of Class Yes is 1.0:

 Number of Instances of the Current Sub Class is 5.0:

 Classes: No Yes

 Probabilities of Class No is 0.4:

 Probabilities of Class Yes is 0.6:

 Number of Instances of the Current Sub Class is 5.0:

 Classes: No Yes

 Probabilities of Class No is 0.4:

 Probabilities of Class Yes is 0.6:

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:
Info-gain for Outlook is :0.246749819774

Information Gain Calculation of  Humidity

 Number of Instances of the Current Sub Class is 7.0:

 Classes: No Yes

 Probabilities of Class No is 0.42857142857142855:

 Probabilities of Class Yes is 0.5714285714285714:

 Number of Instances of the Current Sub Class is 7.0:

 Classes: No Yes

 Probabilities of Class No is 0.14285714285714285:

 Probabilities of Class Yes is 0.8571428571428571:

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:

 Info-gain for Humidity is: 0.151835501362

Information Gain Calculation of  Wind

 Number of Instances of the Current Sub Class is 6.0:

 Classes: No Yes

 Probabilities of Class No is 0.5:

 Probabilities of Class Yes is 0.5:

 Number of Instances of the Current Sub Class is 8.0:
```

```
 Classes: No Yes

 Probabilities of Class No is 0.25:

 Probabilities of Class Yes is 0.75:

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:

 Info-gain for Wind is:0.0481270304083

Information Gain Calculation of  Temperature

 Number of Instances of the Current Sub Class is 4.0:

 Classes: No Yes

 Probabilities of Class No is 0.25:

 Probabilities of Class Yes is 0.75:

 Number of Instances of the Current Sub Class is 4.0:

 Classes: No Yes

 Probabilities of Class No is 0.5:

 Probabilities of Class Yes is 0.5:

 Number of Instances of the Current Sub Class is 6.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:

 Info-gain for Temperature is:0.029222565659
```

ID3 Algorithm

In [217]:

```
def id3(df, target_attribute_name, attribute_names, default_class=None):

    ## Tally target attribute:
    from collections import Counter
    cnt = Counter(x for x in df[target_attribute_name])# class of YES /NO

    ## First check: Is this split of the dataset homogeneous?
    if len(cnt) == 1:
        return next(iter(cnt))  # next input data set, or raises StopIteration when EOF is hit.

    ## Second check: Is this split of the dataset empty?
    # if yes, return a default value
    elif df.empty or (not attribute_names):
        return default_class  # Return None for Empty Data Set

    ## Otherwise: This dataset is ready to be devied up!
    else:
```

```
    else:
        # Get Default Value for next recursive call of this function:
        default_class = max(cnt.keys()) #No of YES and NO Class
        # Compute the Information Gain of the attributes:
        gainz = [information_gain(df, attr, target_attribute_name) for attr in attribute_names] #
        index_of_max = gainz.index(max(gainz)) # Index of Best Attribute
        # Choose Best Attribute to split on:
        best_attr = attribute_names[index_of_max]

        # Create an empty tree, to be populated in a moment
        tree = {best_attr:{}} # Iniiate the tree with best attribute as a node
        remaining_attribute_names = [i for i in attribute_names if i != best_attr]

        # Split dataset
        # On each split, recursively call this algorithm.
        # populate the empty tree with subtrees, which
        # are the result of the recursive call
        for attr_val, data_subset in df.groupby(best_attr):
            subtree = id3(data_subset,
                        target_attribute_name,
                        remaining_attribute_names,
                        default_class)
            tree[best_attr][attr_val] = subtree
        return tree
```

Predicting Attributes

In [218]:

```
# Get Predictor Names (all but 'class')
attribute_names = list(df_tennis.columns)
print("List of Attributes:", attribute_names)
attribute_names.remove('PlayTennis') #Remove the class attribute
print("Predicting Attributes:", attribute_names)
```

```
List of Attributes: ['PlayTennis', 'Outlook', 'Temperature', 'Humidity', 'Wind']
Predicting Attributes: ['Outlook', 'Temperature', 'Humidity', 'Wind']
```

Tree Construction

In [219]:

```
# Run Algorithm:
from pprint import pprint
tree = id3(df_tennis,'PlayTennis',attribute_names)
print("\n\nThe Resultant Decision Tree is :\n")
#print(tree)
pprint(tree)
attribute = next(iter(tree))
print("Best Attribute :\n",attribute)
print("Tree Keys:\n",tree[attribute].keys())
```

```
Information Gain Calculation of  Outlook

 Number of Instances of the Current Sub Class is 4.0:

 Classes: Yes Yes

 Probabilities of Class Yes is 1.0:

 Probabilities of Class Yes is 1.0:

 Number of Instances of the Current Sub Class is 5.0:

 Classes: No Yes

 Probabilities of Class No is 0.4:

 Probabilities of Class Yes is 0.6:

 Number of Instances of the Current Sub Class is 5.0:

 Classes: No Yes
```

```
 Probabilities of Class No is 0.4:

 Probabilities of Class Yes is 0.6:

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:
Information Gain Calculation of  Temperature

 Number of Instances of the Current Sub Class is 4.0:

 Classes: No Yes

 Probabilities of Class No is 0.25:

 Probabilities of Class Yes is 0.75:

 Number of Instances of the Current Sub Class is 4.0:

 Classes: No Yes

 Probabilities of Class No is 0.5:

 Probabilities of Class Yes is 0.5:

 Number of Instances of the Current Sub Class is 6.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:
Information Gain Calculation of  Humidity

 Number of Instances of the Current Sub Class is 7.0:

 Classes: No Yes

 Probabilities of Class No is 0.42857142857142855:

 Probabilities of Class Yes is 0.5714285714285714:

 Number of Instances of the Current Sub Class is 7.0:

 Classes: No Yes

 Probabilities of Class No is 0.14285714285714285:

 Probabilities of Class Yes is 0.8571428571428571:

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:
Information Gain Calculation of  Wind

 Number of Instances of the Current Sub Class is 6.0:

 Classes: No Yes

 Probabilities of Class No is 0.5:
```

```
 Probabilities of Class Yes is 0.5:

 Number of Instances of the Current Sub Class is 8.0:

 Classes: No Yes

 Probabilities of Class No is 0.25:

 Probabilities of Class Yes is 0.75:

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:
Information Gain Calculation of  Temperature

 Number of Instances of the Current Sub Class is 2.0:

 Classes: No Yes

 Probabilities of Class No is 0.5:

 Probabilities of Class Yes is 0.5:

 Number of Instances of the Current Sub Class is 3.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:

 Number of Instances of the Current Sub Class is 5.0:

 Classes: No Yes

 Probabilities of Class No is 0.4:

 Probabilities of Class Yes is 0.6:
Information Gain Calculation of  Humidity

 Number of Instances of the Current Sub Class is 2.0:

 Classes: No Yes

 Probabilities of Class No is 0.5:

 Probabilities of Class Yes is 0.5:

 Number of Instances of the Current Sub Class is 3.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:

 Number of Instances of the Current Sub Class is 5.0:

 Classes: No Yes

 Probabilities of Class No is 0.4:

 Probabilities of Class Yes is 0.6:
Information Gain Calculation of  Wind

 Number of Instances of the Current Sub Class is 2.0:

 Classes: No No

 Probabilities of Class No is 1.0:

 Probabilities of Class No is 1.0:
```

```
 Number of Instances of the Current Sub Class is 3.0:

 Classes: Yes Yes

 Probabilities of Class Yes is 1.0:

 Probabilities of Class Yes is 1.0:

 Number of Instances of the Current Sub Class is 5.0:

 Classes: No Yes

 Probabilities of Class No is 0.4:

 Probabilities of Class Yes is 0.6:
Information Gain Calculation of  Temperature

 Number of Instances of the Current Sub Class is 1.0:

 Classes: Yes Yes

 Probabilities of Class Yes is 1.0:

 Probabilities of Class Yes is 1.0:

 Number of Instances of the Current Sub Class is 2.0:

 Classes: No No

 Probabilities of Class No is 1.0:

 Probabilities of Class No is 1.0:

 Number of Instances of the Current Sub Class is 2.0:

 Classes: No Yes

 Probabilities of Class No is 0.5:

 Probabilities of Class Yes is 0.5:

 Number of Instances of the Current Sub Class is 5.0:

 Classes: No Yes

 Probabilities of Class No is 0.4:

 Probabilities of Class Yes is 0.6:
Information Gain Calculation of  Humidity

 Number of Instances of the Current Sub Class is 3.0:

 Classes: No No

 Probabilities of Class No is 1.0:

 Probabilities of Class No is 1.0:

 Number of Instances of the Current Sub Class is 2.0:

 Classes: Yes Yes

 Probabilities of Class Yes is 1.0:

 Probabilities of Class Yes is 1.0:

 Number of Instances of the Current Sub Class is 5.0:

 Classes: No Yes

 Probabilities of Class No is 0.4:

 Probabilities of Class Yes is 0.6:
Information Gain Calculation of  Wind

 Number of Instances of the Current Sub Class is 2.0:
```

```
 Classes: No Yes

 Probabilities of Class No is 0.5:

 Probabilities of Class Yes is 0.5:

 Number of Instances of the Current Sub Class is 3.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:

 Number of Instances of the Current Sub Class is 5.0:

 Classes: No Yes

 Probabilities of Class No is 0.4:

 Probabilities of Class Yes is 0.6:


The Resultant Decision Tree is :

{'Outlook': {'Overcast': 'Yes',
             'Rain': {'Wind': {'Strong': 'No', 'Weak': 'Yes'}},
             'Sunny': {'Humidity': {'High': 'No', 'Normal': 'Yes'}}}}
Best Attribute :
 Outlook
Tree Keys:
 dict_keys(['Overcast', 'Rain', 'Sunny'])
```

Classification Accuracy

In [220]:

```python
def classify(instance, tree, default=None): # Instance of Play Tennis with Predicted

    #print("Instance:",instance)
    attribute = next(iter(tree)) # Outlook/Humidity/Wind
    print("Key:",tree.keys())  # [Outlook,Humidity,Wind ]
    print("Attribute:",attribute) # [Key /Attribute Both are same ]

    # print("Insance of Attribute :",instance[attribute],attribute)
    if instance[attribute] in tree[attribute].keys(): # Value of the attributs in  set of Tree 
keys
        result = tree[attribute][instance[attribute]]
        print("Instance Attribute:",instance[attribute],"TreeKeys :",tree[attribute].keys())
        if isinstance(result, dict): # this is a tree, delve deeper
            return classify(instance, result)
        else:
            return result # this is a label
    else:
        return default
```

In [138]:

```python
df_tennis['predicted'] = df_tennis.apply(classify, axis=1, args=(tree,'No') ) 
    # classify func allows for a default arg: when tree doesn't have answer for a particular
    # combitation of attribute-values, we can use 'no' as the default guess

print(df_tennis['predicted'])

print('\n Accuracy is:\n' + str( sum(df_tennis['PlayTennis']==df_tennis['predicted'] ) / (1.0*len(d
f_tennis.index)) ))


df_tennis[['PlayTennis', 'predicted']]
```

```
Key: dict_keys(['Outlook'])
Attribute: Outlook
```

```
Instance Attribute: Sunny TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Humidity'])
Attribute: Humidity
Instance Attribute: High TreeKeys : dict_keys(['High', 'Normal'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Sunny TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Humidity'])
Attribute: Humidity
Instance Attribute: High TreeKeys : dict_keys(['High', 'Normal'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Overcast TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Rain TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Wind'])
Attribute: Wind
Instance Attribute: Weak TreeKeys : dict_keys(['Strong', 'Weak'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Rain TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Wind'])
Attribute: Wind
Instance Attribute: Weak TreeKeys : dict_keys(['Strong', 'Weak'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Rain TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Wind'])
Attribute: Wind
Instance Attribute: Strong TreeKeys : dict_keys(['Strong', 'Weak'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Overcast TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Sunny TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Humidity'])
Attribute: Humidity
Instance Attribute: High TreeKeys : dict_keys(['High', 'Normal'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Sunny TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Humidity'])
Attribute: Humidity
Instance Attribute: Normal TreeKeys : dict_keys(['High', 'Normal'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Rain TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Wind'])
Attribute: Wind
Instance Attribute: Weak TreeKeys : dict_keys(['Strong', 'Weak'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Sunny TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Humidity'])
Attribute: Humidity
Instance Attribute: Normal TreeKeys : dict_keys(['High', 'Normal'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Overcast TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Overcast TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Rain TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Wind'])
Attribute: Wind
Instance Attribute: Strong TreeKeys : dict_keys(['Strong', 'Weak'])
0      No
1      No
2     Yes
3     Yes
4     Yes
5      No
6     Yes
```

```
7      No
8     Yes
9     Yes
10    Yes
11    Yes
12    Yes
13     No
Name: predicted, dtype: object

 Accuracy is:
1.0
```

Out[138]:

	PlayTennis	predicted
0	No	No
1	No	No
2	Yes	Yes
3	Yes	Yes
4	Yes	Yes
5	No	No
6	Yes	Yes
7	No	No
8	Yes	Yes
9	Yes	Yes
10	Yes	Yes
11	Yes	Yes
12	Yes	Yes
13	No	No

Classification Accuracy: Training/Testing Set

In [221]:

```
training_data = df_tennis.iloc[1:-4] # all but last four instances
test_data  = df_tennis.iloc[-4:] # just the last four
train_tree = id3(training_data, 'PlayTennis', attribute_names)

test_data['predicted2'] = test_data.apply(                                # <---- test_data source
                                          classify,
                                          axis=1,
                                          args=(train_tree,'Yes') ) # <---- train_data tree


print ('\n\n Accuracy is : ' + str( sum(test_data['PlayTennis']==test_data['predicted2'] ) / (1.0*l
en(test_data.index)) ))
```

```
Information Gain Calculation of  Outlook

 Number of Instances of the Current Sub Class is 2.0:

 Classes: Yes Yes

 Probabilities of Class Yes is 1.0:

 Probabilities of Class Yes is 1.0:

 Number of Instances of the Current Sub Class is 4.0:

 Classes: No Yes

 Probabilities of Class No is 0.25:
```

```
 Probabilities of Class Yes is 0.75:

 Number of Instances of the Current Sub Class is 3.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:

 Number of Instances of the Current Sub Class is 9.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:
Information Gain Calculation of  Temperature

 Number of Instances of the Current Sub Class is 4.0:

 Classes: No Yes

 Probabilities of Class No is 0.25:

 Probabilities of Class Yes is 0.75:

 Number of Instances of the Current Sub Class is 2.0:

 Classes: No Yes

 Probabilities of Class No is 0.5:

 Probabilities of Class Yes is 0.5:

 Number of Instances of the Current Sub Class is 3.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:

 Number of Instances of the Current Sub Class is 9.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:
Information Gain Calculation of  Humidity

 Number of Instances of the Current Sub Class is 4.0:

 Classes: No Yes

 Probabilities of Class No is 0.5:

 Probabilities of Class Yes is 0.5:

 Number of Instances of the Current Sub Class is 5.0:

 Classes: No Yes

 Probabilities of Class No is 0.2:

 Probabilities of Class Yes is 0.8:

 Number of Instances of the Current Sub Class is 9.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:
Information Gain Calculation of  Wind
```

```
Information Gain Calculation of  Wind

 Number of Instances of the Current Sub Class is 3.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:

 Number of Instances of the Current Sub Class is 6.0:

 Classes: No Yes

 Probabilities of Class No is 0.16666666666666666:

 Probabilities of Class Yes is 0.8333333333333334:

 Number of Instances of the Current Sub Class is 9.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:
Information Gain Calculation of  Temperature

 Number of Instances of the Current Sub Class is 2.0:

 Classes: No Yes

 Probabilities of Class No is 0.5:

 Probabilities of Class Yes is 0.5:

 Number of Instances of the Current Sub Class is 2.0:

 Classes: Yes Yes

 Probabilities of Class Yes is 1.0:

 Probabilities of Class Yes is 1.0:

 Number of Instances of the Current Sub Class is 4.0:

 Classes: No Yes

 Probabilities of Class No is 0.25:

 Probabilities of Class Yes is 0.75:
Information Gain Calculation of  Humidity

 Number of Instances of the Current Sub Class is 1.0:

 Classes: Yes Yes

 Probabilities of Class Yes is 1.0:

 Probabilities of Class Yes is 1.0:

 Number of Instances of the Current Sub Class is 3.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:

 Number of Instances of the Current Sub Class is 4.0:

 Classes: No Yes

 Probabilities of Class No is 0.25:

 Probabilities of Class Yes is 0.75:
Information Gain Calculation of  Wind

 Number of Instances of the Current Sub Class is 1.0:
```

```
 Number of Instances of the Current Sub Class is 1.0:

 Classes: No No

 Probabilities of Class No is 1.0:

 Probabilities of Class No is 1.0:

 Number of Instances of the Current Sub Class is 3.0:

 Classes: Yes Yes

 Probabilities of Class Yes is 1.0:

 Probabilities of Class Yes is 1.0:

 Number of Instances of the Current Sub Class is 4.0:

 Classes: No Yes

 Probabilities of Class No is 0.25:

 Probabilities of Class Yes is 0.75:
Information Gain Calculation of  Temperature

 Number of Instances of the Current Sub Class is 1.0:

 Classes: Yes Yes

 Probabilities of Class Yes is 1.0:

 Probabilities of Class Yes is 1.0:

 Number of Instances of the Current Sub Class is 1.0:

 Classes: No No

 Probabilities of Class No is 1.0:

 Probabilities of Class No is 1.0:

 Number of Instances of the Current Sub Class is 1.0:

 Classes: No No

 Probabilities of Class No is 1.0:

 Probabilities of Class No is 1.0:

 Number of Instances of the Current Sub Class is 3.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:
Information Gain Calculation of  Humidity

 Number of Instances of the Current Sub Class is 2.0:

 Classes: No No

 Probabilities of Class No is 1.0:

 Probabilities of Class No is 1.0:

 Number of Instances of the Current Sub Class is 1.0:

 Classes: Yes Yes

 Probabilities of Class Yes is 1.0:

 Probabilities of Class Yes is 1.0:

 Number of Instances of the Current Sub Class is 3.0:

 Classes: No Yes
```

```
 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:
Information Gain Calculation of  Wind

 Number of Instances of the Current Sub Class is 1.0:

 Classes: No No

 Probabilities of Class No is 1.0:

 Probabilities of Class No is 1.0:

 Number of Instances of the Current Sub Class is 2.0:

 Classes: No Yes

 Probabilities of Class No is 0.5:

 Probabilities of Class Yes is 0.5:

 Number of Instances of the Current Sub Class is 3.0:

 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Sunny TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Temperature'])
Attribute: Temperature
Instance Attribute: Mild TreeKeys : dict_keys(['Cool', 'Hot', 'Mild'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Overcast TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Overcast TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Outlook'])
Attribute: Outlook
Instance Attribute: Rain TreeKeys : dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Wind'])
Attribute: Wind
Instance Attribute: Strong TreeKeys : dict_keys(['Strong', 'Weak'])


 Accuracy is : 0.75
```

```
C:\Users\Dr.Thyagaraju\Anaconda3\lib\site-packages\ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
```

End