A machine learning algorithm called a decision tree, to predict if a player will play golf that day based on the weather (Outlook, Temperature, Humidity, Windy).

In [ ]:

```

```

Decision Trees are a type of Supervised Learning Algorithms(meaning that they were given labeled data to train on). The training data is continuously split into two more sub-nodes according to a certain parameter. The tree can be explained by two things, leaves and decision nodes. The decision nodes are where the data is split. The leaves are the decisions or the final outcomes. You can think of a decision tree in programming terms as a tree that has a bunch of “if statements” for each node until you get to a leaf node (the final outcome).

In [26]:

```

## import dependencies
from sklearn import tree #For our Decision Tree
import pandas as pd # For our DataFrame
import pydotplus # To create our Decision Tree Graph
#the following is important to display the graph.
#search for the following in your computer
import os
os.environ["PATH"] += os.pathsep + 'C:/Users/HP/Anaconda3/Library/bin/graphviz'



from IPython.display import Image  # To Display a image of our graph
```

Data Description:very column/feature/attribute (Outlook, Temperature, Humidity, Windy, Play).

Outlook = The outlook of the weather

Temperature = The temperature of the weather

Humidity = The humidity of the weather

Windy = A variable if it is windy that day or not

Play = The target variable, tells if the golfer played golf that day or not

values in each of the columns

Outlook values: sunny, overcast, rainy

Temperature values: hot, mild, cold

Humidity values: high, normal

Windy values: true, false

Play values: yes, no

In [ ]:

```

```

In [15]:

```
#Create the dataset
#create empty data frame
golf_df = pd.DataFrame()

#add outlook
golf_df['Outlook'] = ['sunny', 'sunny', 'overcast', 'rainy', 'rainy', 'rainy',
                     'overcast', 'sunny', 'sunny', 'rainy', 'sunny', 'overcast',
                     'overcast', 'rainy']

#add temperature
golf_df['Temperature'] = ['hot', 'hot', 'hot', 'mild', 'cool', 'cool', 'cool',
                         'mild', 'cool', 'mild', 'mild', 'mild', 'hot', 'mild']

#add humidity
golf_df['Humidity'] = ['high', 'high', 'high', 'high', 'normal', 'normal', 'normal',
                      'high', 'normal', 'normal', 'normal', 'high', 'normal', 'high']

#add windy
golf_df['Windy'] = ['false', 'true', 'false', 'false', 'false', 'true', 'true',
                   'false', 'false', 'false', 'true', 'true', 'false', 'true']

#finally add play
golf_df['Play'] = ['no', 'no', 'yes', 'yes', 'yes', 'no', 'yes', 'no', 'yes', 'yes', '
                  'yes', 'yes', 'no']


#Print/show the new data
print(golf_df)
```

```
     Outlook Temperature 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
```

In [16]:

```
#Do not use
#Create the dataset
#create empty data frame
#golf_df = pd.read_csv("Weather1.csv")
```

In [17]:

```
#golf_df
```

Convert categorical variable into dummy/indicator variables or (binary vairbles) essentialy 1's and 0's . WE chose the variable name one_hot_data bescause in ML one-hot is a group of bits among which the legal combinations of values are only those with a single high (1) bit and all the others low (0)

In [ ]:

```

```

Outlook are of 3 types. On Row 0 we have Outlook=sunny so the value of Outlook_sunny is 1

In [18]:

```

one_hot_data = pd.get_dummies(golf_df[ ['Outlook', 'Temperature', 'Humidity', 'Windy']
#print the new dummy data
one_hot_data
```

Out[18]:

	Outlook_overcast	Outlook_rainy	Outlook_sunny	Temperature_cool	Temperature_hot	Temp
0	0	0	1	0	1	
1	0	0	1	0	1	
2	1	0	0	0	1	
3	0	1	0	0	0	
4	0	1	0	1	0	
5	0	1	0	1	0	
6	1	0	0	1	0	
7	0	0	1	0	0	
8	0	0	1	1	0	
9	0	1	0	0	0	
10	0	0	1	0	0	
11	1	0	0	0	0	
12	1	0	0	0	1	
13	0	1	0	0	0	

In [19]:

```
golf_df['Play']
```

Out[19]:

```
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: Play, dtype: object
```

clf.fit. Fit means train

one_hot_data has all the independent data and golf_df Play has the dependent variable

In [20]:

```

# The decision tree classifier.
clf = tree.DecisionTreeClassifier()
# Training the Decision Tree
clf_train = clf.fit(one_hot_data, golf_df['Play'])
```

In [ ]:

```

```

In [21]:

```
print(golf_df['Play'])
golf_df
```

```
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: Play, dtype: object
```

Out[21]:

	Outlook	Temperature	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

In [22]:

```
print(golf_df['Play'])

clf_train
```

```
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: Play, dtype: object
```

Out[22]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

X[0] is outlook overcast Label X[0] has total samples nsamples = 14 , nvalue = [5, 9] 5=yes and 9=no

In [34]:

```

# Export/Print a decision tree in DOT format.

print(tree.export_graphviz(clf_train, None))



```

```
digraph Tree {
node [shape=box] ;
0 [label="X[0] <= 0.5\ngini = 0.459\nsamples = 14\nvalue = [5, 9]"] ;
1 [label="X[6] <= 0.5\ngini = 0.5\nsamples = 10\nvalue = [5, 5]"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="X[9] <= 0.5\ngini = 0.32\nsamples = 5\nvalue = [1, 4]"] ;
1 -> 2 ;
3 [label="gini = 0.0\nsamples = 3\nvalue = [0, 3]"] ;
2 -> 3 ;
4 [label="X[5] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]"] ;
2 -> 4 ;
5 [label="gini = 0.0\nsamples = 1\nvalue = [1, 0]"] ;
4 -> 5 ;
6 [label="gini = 0.0\nsamples = 1\nvalue = [0, 1]"] ;
4 -> 6 ;
7 [label="X[1] <= 0.5\ngini = 0.32\nsamples = 5\nvalue = [4, 1]"] ;
1 -> 7 ;
8 [label="gini = 0.0\nsamples = 3\nvalue = [3, 0]"] ;
7 -> 8 ;
9 [label="X[9] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]"] ;
7 -> 9 ;
10 [label="gini = 0.0\nsamples = 1\nvalue = [0, 1]"] ;
9 -> 10 ;
11 [label="gini = 0.0\nsamples = 1\nvalue = [1, 0]"] ;
9 -> 11 ;
12 [label="gini = 0.0\nsamples = 4\nvalue = [0, 4]"] ;
0 -> 12 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
}
```

In [35]:

```
#Create Dot Data
dot_data = tree.export_graphviz(clf_train, out_file=None, feature_names=list(one_hot_da
                                class_names=['Not_Play', 'Play'], rounded=True, filled

```

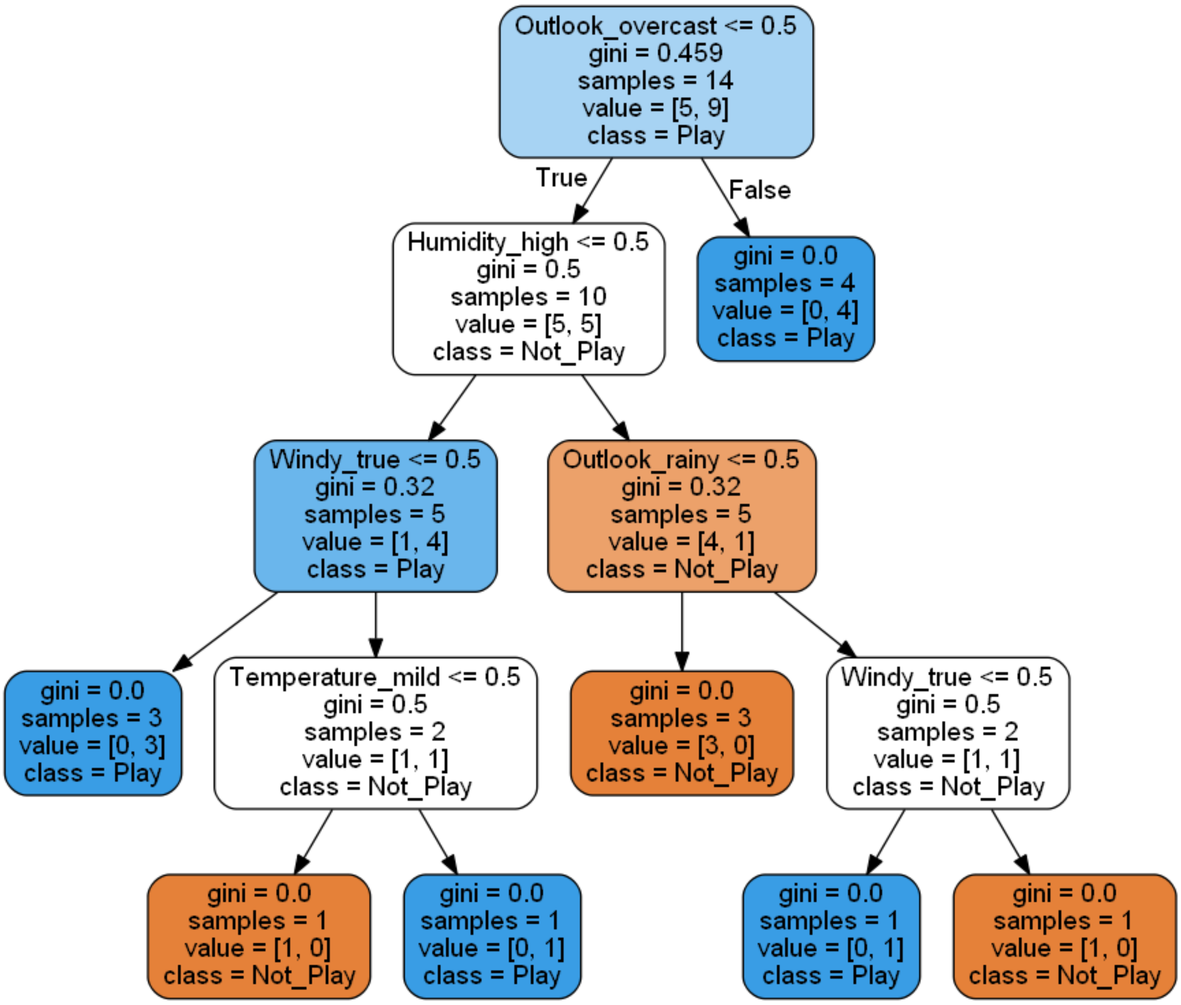
If you read the data you will see there are overcast Label X[0] Overcast Label has total samples nsamples = 14 , nvalue = [5, 9] 5=yes and 9=no

In [36]:

```
#Gini decides which attribute/feature should be
#placed at the root node, which features will act as internal nodes or leaf nodes
#Create Graph from DOT data
graph = pydotplus.graph_from_dot_data(dot_data)

# Show graph
Image(graph.create_png())
```

Out[36]:



Last but not least, make the prediction, by inputting the Outlook as ‘sunny’, Temperature as ‘hot’, Humidity as ‘normal’ and Windy as ‘false’. My model predicted that input to be ‘yes’, meaning the golfer will play golf that day.

Test model prediction input:

Outook = sunny -0 0 1

Temperature = hot - 0 1 0

Humidity = normal

Windy = false

In [40]:

```

prediction = clf_train.predict([[0,0,1,0,1,0,0,1,1,0]])
prediction
```

Out[40]:

```
array(['yes'], dtype=object)
```

In [ ]:

```

```