Module Code & Module Title

CU6051NA - Artificial Intelligence

Assignment

Year and Semester

2018-19 Spring

Student Name: Rajat Shrestha

London Met ID: 17030954

College ID: np01cp4a170021

Table of Contents

Table of Figures

Decision Tree Assignment Solution

Build a decision tree using data from the given table. The data is about whether to play tennis or not given the weather conditions. So, when the tree is built, it should be able to give out a decision on whether to play tennis or not given the weather conditions.

Outlook	Temp.	Humidity	Wind	Decision
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Figure 1: Table of data

To build a decision tree we must go through various process to find the nodes:

Selecting the root node:

Here, we have four different features which is given the final decision to construct a decision tree we must first find the root node. To choose the optimal root node we must find the attribute with the maximum amount of gain.

$$\boldsymbol{Entropy(S) = -p_{+}log_{2}(p_{+}) - p_{-}log_{2}(p_{-})}$$

$$\boldsymbol{Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)}$$

Figure 2: Formulae to calculate Entropy and Gain

Since we need to do this calculation multiple times a function was created to take the count of positive and negative outcomes and give the entropy as well as another function to take the overall positive and negative outcomes with the list of all the positive and negative outcomes of individual category of an attribute:

```
[ ]
def entropy(positive,negative):
    sum = negative+positive
    try:
        positive_ratio = math.log2(positive/sum)
        negative_ratio = math.log2(negative/sum)
    except ValueError:
        positive_ratio = 0
        negative_ratio = 0
    return (-(positive/sum)*positive_ratio-(negative/sum)*negative_ratio)
```

```
entropy(9,5)
```

```
0.9402859586706309
```

Figure 3: Python function to calculate Entropy

```
[ ]
def gain(positive,negative,list):
    parent_entropy = entropy(negative,positive)
    sum = negative+positive
    sum_of_values = 0
    for i in list:
        a = i[0]
        b = i[1]
        sum_of_values = sum_of_values + abs(abs(a+b)/abs(sum))*entropy(a,b)
    return parent_entropy - sum_of_values

```

Figure 4: Python Function to calculate Gain

Now, to calculate the gain of each column we need to calculate the entropy of the all the labels categorized in an attribute, we must tally down all the positive and negative outcomes of all existing categories of the attributes.

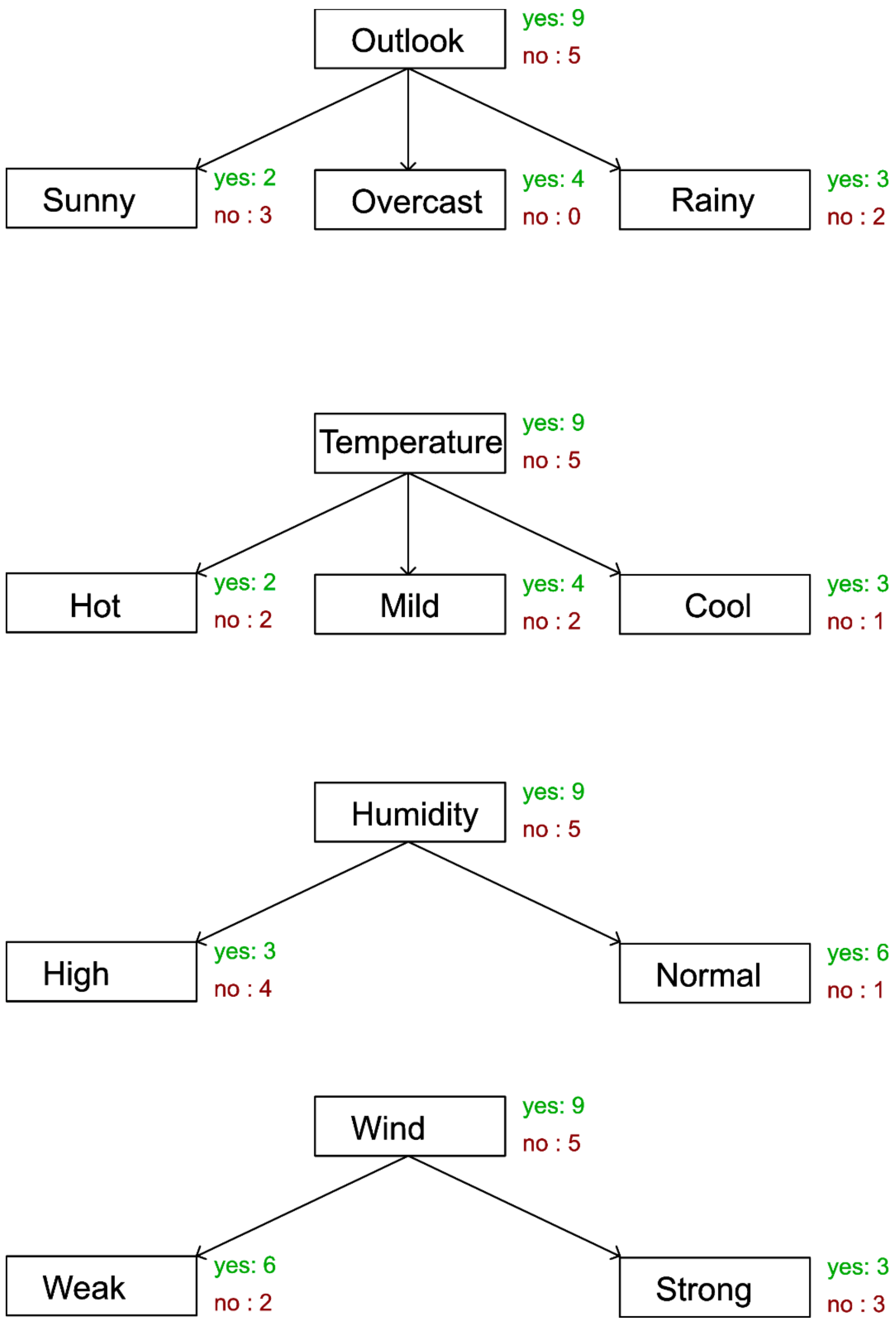

Figure 5: Tally of each category of attributes

```
[ ]
outlook = [[2,3],[4,0],[3,2]]
temprature = [[2,2],[4,2],[3,1]]
humidity = [[5,4],[6,1]]
wind = [[6,2],[3,3]]

print("outlook_gain = "+str(gain(9,5,outlook)))
print("temprature_gain = "+str(gain(9,5,temprature)))
print("humidity_gain = "+str(gain(9,5,humidity)))
print("wind_gain = "+str(gain(9,5,wind)))
```

```
outlook_gain = 0.2467498197744391
temprature_gain = 0.029222565658954647
humidity_gain = 0.007329245197752798
wind_gain = 0.04812703040826927
```

Figure 6: Storing above data in lists and calculating the Gain value of each attributes

As we can see the highest gain value is of outlook so taking outlook as the root node.

Now, as we got the root node as Outlook, observing the three root nodes, overcast is totally saturated while sunny and rainy is not. So, at the next step the same procedure is carried out with stripped data including the selected category.

Selecting nodes for Sunny and Rainy nodes:

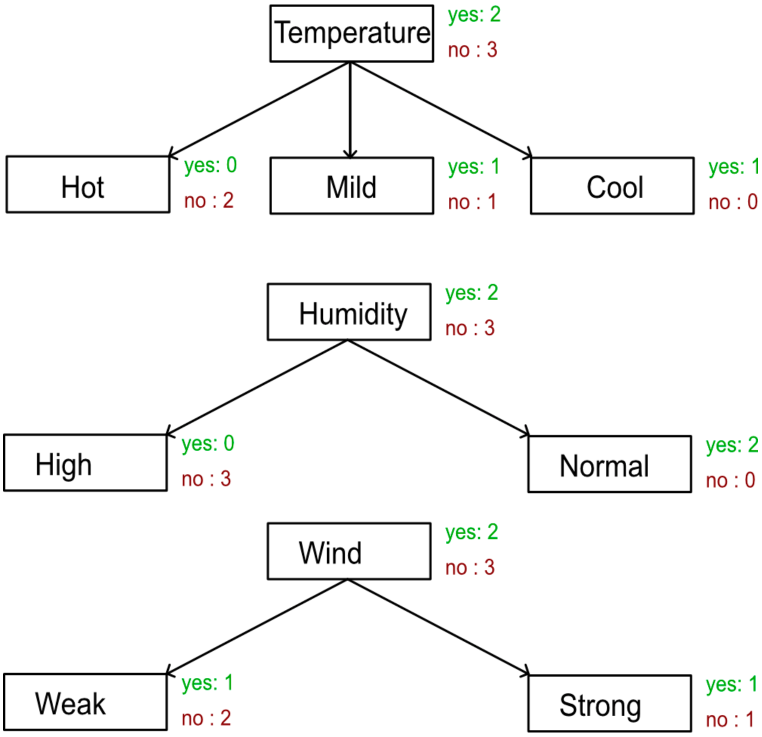

Figure 7: Tally of each category on Sunny node

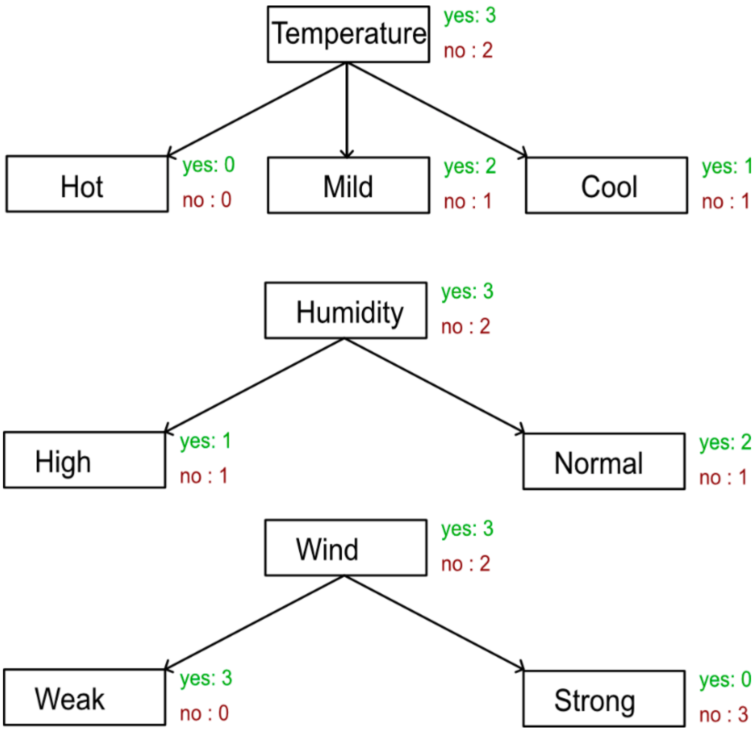

Figure 8:Tally of each category on Rainy node

Calculations using above data:

```
[11]
sunny_humidity = [[0,3],[2,0]]
sunny_temperature = [[0,2],[1,1],[1,0]]
sunny_wind = [[1,2],[1,1]]

print("sunny_humidity_gain = "+str(gain(2,3,sunny_humidity)))
print("sunny_temperature_gain = "+str(gain(2,3,sunny_temperature)))
print("sunny_wind_gain = "+str(gain(2,3,sunny_wind)))
```

```
sunny_humidity_gain = 0.9709505944546686
sunny_temperature_gain = 0.5709505944546686
sunny_wind_gain = 0.01997309402197489
```

Figure 9: Gain calculated in Sunny node

```
[12]
rain_humidity = [[1,1],[2,1]]
rain_temperature = [[2,1],[1,1]]
rain_wind = [[3,0],[0,2]]

print("rain_humidity_gain = "+str(gain(3,2,rain_humidity)))
print("rain_temperature_gain = "+str(gain(3,2,rain_temperature)))
print("rain_wind_gain = "+str(gain(3,2,rain_wind)))
```

```
rain_humidity_gain = 0.01997309402197489
rain_temperature_gain = 0.01997309402197489
rain_wind_gain = 0.9709505944546686
```

Figure 10: Gain Calculated in Rainy node

As we can see in the sunny node, humidity is the node with lowest saturation 0 and the highest gain and in the rainy node, wind gives the lowest degree of saturation and highest gain value so taking them as the child nodes.

Note: these calculations were unnecessary but if the nodes were not saturated it should be done to get the node with highest gain value.

Building the final Decision tree:

Finally, from the above calculations and observations, the resulting decision tree is as follows:

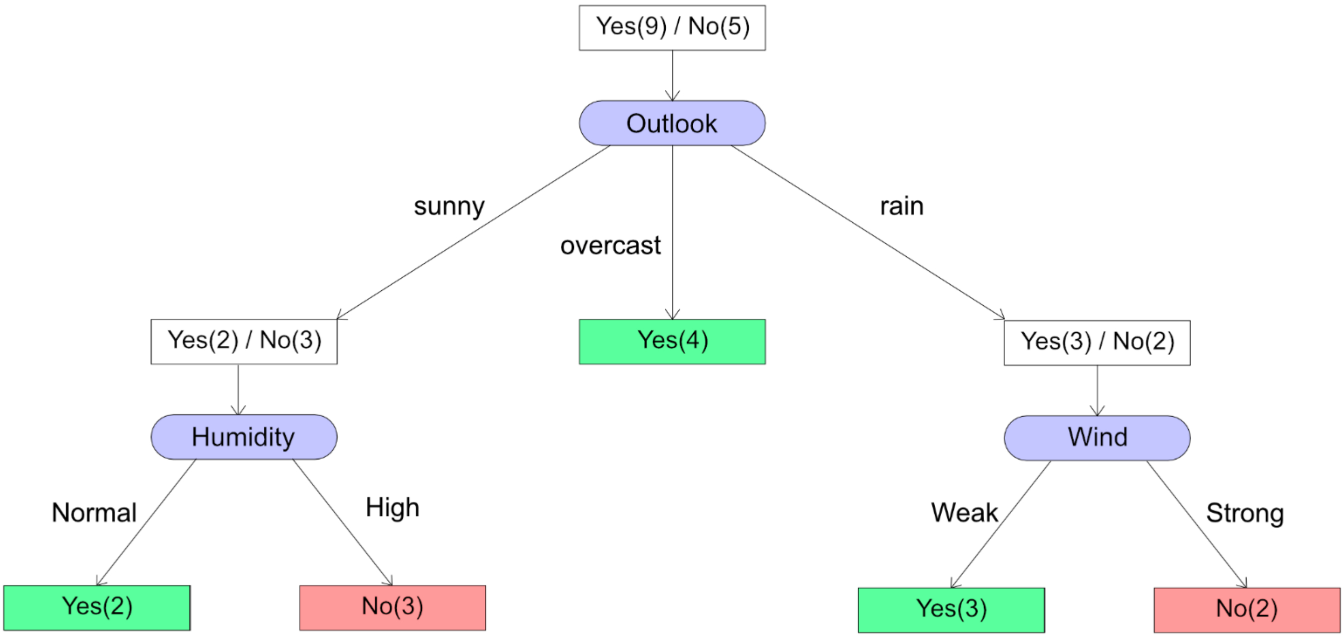

Figure 11: Final decision tree

The link to the notebook used to calculate the entropy and gain(a):

https://colab.research.google.com/drive/1fj8nGbi0hC6d0_8mUww1Ea14LHhi_X4Q