Decision tree Example

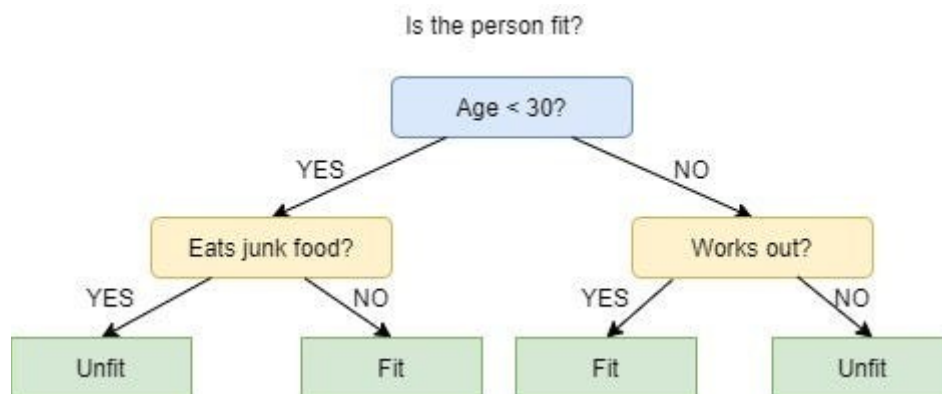Is the person fit?

The picture above depicts a decision tree that is used to classify whether a person is **Fit** or **Unfit.** The decision nodes here are questions like '' *Is the person less than 30 years of age?', 'Does the person eat junk?'*, etc. and the leaves are one of the two possible outcomes viz. **Fit** and **Unfit**. Looking at the Decision Tree we can say make the following decisions:
if a person is less than 30 years of age and doesn't eat junk food then he is Fit, if a person is less than 30 years of age and eats junk food then he is Unfit and so on.

The initial node is called the **root node** *(colored in blue),* the final nodes are called the **leaf nodes** *(colored in green)* and the rest of the nodes are called **intermediate** or **internal** nodes. The root and intermediate nodes represent the decisions while the leaf nodes represent the outcomes.

ID3 in brief

ID3 stands for Iterative Dichotomiser 3 and is named such because the algorithm iteratively (repeatedly) dichotomizes(divides) features into two or more groups at each step.

Invented by Ross Quinlan, ID3 uses a **top-down greedy** approach to build a decision tree. In simple words, the **top-down** approach means that we start building the tree from the top and the **greedy** approach means that at each iteration we select the best feature at the present moment to create a node.

Most generally ID3 is only used for classification problems with nominal features only.

Dataset description

In this article, we'll be using a sample dataset of COVID-19 infection. A preview of the entire dataset is shown below.

```
+----+-------+-------+------------------+----------+
| ID | Fever | Cough | Breathing issues | Infected |
+----+-------+-------+------------------+----------+
| 1  | NO    | NO    | NO               | NO       |
+----+-------+-------+------------------+----------+
| 2  | YES   | YES   | YES              | YES      |
+----+-------+-------+------------------+----------+
| 3  | YES   | YES   | NO               | NO       |
+----+-------+-------+------------------+----------+
| 4  | YES   | NO    | YES              | YES      |
```

```
+----+-------+-------+----------------+---------+
| 5  | YES   | YES   | YES            | YES     |
+----+-------+-------+----------------+---------+
| 6  | NO    | YES   | NO             | NO      |
+----+-------+-------+----------------+---------+
| 7  | YES   | NO    | YES            | YES     |
+----+-------+-------+----------------+---------+
| 8  | YES   | NO    | YES            | YES     |
+----+-------+-------+----------------+---------+
| 9  | NO    | YES   | YES            | YES     |
+----+-------+-------+----------------+---------+
| 10 | YES   | YES   | NO             | YES     |
+----+-------+-------+----------------+---------+
| 11 | NO    | YES   | NO             | NO      |
+----+-------+-------+----------------+---------+
| 12 | NO    | YES   | YES            | YES     |
+----+-------+-------+----------------+---------+
| 13 | NO    | YES   | YES            | NO      |
+----+-------+-------+----------------+---------+
| 14 | YES   | YES   | NO             | NO      |
+----+-------+-------+----------------+---------+
```

The columns are self-explanatory. Y and N stand for Yes and No respectively. The values or **classes** in Infected column Y and N represent Infected and Not Infected respectively.

The columns used to make decision nodes viz. 'Breathing Issues', 'Cough' and 'Fever' are called feature columns or just features and the column used for leaf nodes i.e. 'Infected' is called the target column.

Metrics in ID3

As mentioned previously, the ID3 algorithm selects the best feature at each step while building a Decision tree.
Before you ask, the answer to the question: 'How does ID3 select the best feature?' is that ID3 uses **Information Gain** or just **Gain** to find the best feature.

Highest Information Gain best

In simple words, **Entropy** is the measure of disorder and the Entropy of a dataset is the measure of disorder in the target feature of the dataset.
In the case of binary classification (where the target column has only two types of classes) entropy is **0** if all values in the target column are homogenous(similar) and will be **1** if the target column has equal number values for both the classes.

We denote our dataset as **S,** entropy is calculated as:

$$\text{Entropy(S)} = - \sum p_i * \log_2(p_i) \; ; \; i = 1 \text{ to } n$$

where,
n is the total number of classes in the target column (in our case n = 2 i.e YES and NO)
p_i is the **probability of class 'i'** or the ratio of "*number of rows with class i in the target column*" to the "*total number of rows*" in the dataset.

Information Gain for a feature column **A** is calculated as:

$$\text{IG(S, A)} = \text{Entropy(S)} - \sum((|S_v| \, / \, |S|) * \text{Entropy}(S_v))$$

where S_v is the set of rows in **S** for which the feature column **A** has value **v**, $|S_v|$ is the number of rows in S_v and likewise $|S|$ is the number of rows in **S.**

ID3 Steps

1. Calculate the Information Gain of each feature.
2. Considering that all rows don't belong to the same class, split the dataset **S** into subsets using the feature for which the Information Gain is maximum.
3. Make a decision tree node using the feature with the maximum Information gain.
4. If all rows belong to the same class, make the current node as a leaf node with the class as its label.
5. Repeat for the remaining features until we run out of all features, or the decision tree has all leaf nodes.

Implementation on our Dataset

As stated in the previous section the first step is to find the best feature i.e. the one that has the maximum Information Gain**(IG)**. We'll calculate the IG for each of the features now, but for that, we first need to calculate the entropy of **S**

From the total of 14 rows in our dataset **S**, there are **8** rows with the target value **YES** and **6** rows with the target value **NO**. The entropy of **S** is calculated as:

```
Entropy(S) = − (8/14) * log₂(8/14) − (6/14) * log₂(6/14) = 0.99
```

Note: If all the values in our target column are same the entropy will be zero (meaning that it has no or zero randomness).

We now calculate the Information Gain for each feature:

IG calculation for Fever:
In this(Fever) feature there are **8** rows having value **YES** and **6** rows having value **NO.**
As shown below, in the **8** rows with **YES for** Fever, there are **6** rows having target value **YES** and **2** rows having target value **NO.**

```
+-------+-------+-----------------+----------+
| Fever | Cough | Breathing issues | Infected |
+-------+-------+-----------------+----------+
| YES   | YES   | YES             | YES      |
+-------+-------+-----------------+----------+
| YES   | YES   | NO              | NO       |
+-------+-------+-----------------+----------+
| YES   | NO    | YES             | YES      |
+-------+-------+-----------------+----------+
| YES   | YES   | YES             | YES      |
+-------+-------+-----------------+----------+
| YES   | NO    | YES             | YES      |
+-------+-------+-----------------+----------+
| YES   | NO    | YES             | YES      |
+-------+-------+-----------------+----------+
| YES   | YES   | NO              | YES      |
+-------+-------+-----------------+----------+
| YES   | YES   | NO              | NO       |
+-------+-------+-----------------+----------+
```

As shown below, in the **6** rows with **NO**, there are **2** rows having target value **YES** and **4** rows having target value **NO.**

```
+-------+-------+------------------+----------+
| Fever | Cough | Breathing issues | Infected |
+-------+-------+------------------+----------+
| NO    | NO    | NO               | NO       |
+-------+-------+------------------+----------+
| NO    | YES   | NO               | NO       |
+-------+-------+------------------+----------+
| NO    | YES   | YES              | YES      |
+-------+-------+------------------+----------+
| NO    | YES   | NO               | NO       |
+-------+-------+------------------+----------+
| NO    | YES   | YES              | YES      |
+-------+-------+------------------+----------+
| NO    | YES   | YES              | NO       |
+-------+-------+------------------+----------+
```

The block, below, demonstrates the calculation of Information Gain for **Fever.**

```
# total rows
|S| = 14

For v = YES, |Sᵥ| = 8
Entropy(Sᵥ) = - (6/8) * log₂(6/8) - (2/8) * log₂(2/8) = 0.81

For v = NO, |Sᵥ| = 6
Entropy(Sᵥ) = - (2/6) * log₂(2/6) - (4/6) * log₂(4/6) = 0.91

# Expanding the summation in the IG formula:
IG(S, Fever) = Entropy(S) - (|Sʏᴇѕ| / |S|) * Entropy(Sʏᴇѕ) -
(|Sɴᴏ| / |S|) * Entropy(Sɴᴏ)

∴ IG(S, Fever) = 0.99 - (8/14) * 0.81 - (6/14) * 0.91 = 0.13
```
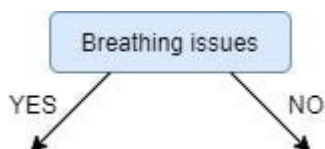
Next, we calculate the **IG** for the features **"Cough"** and **"Breathing issues".**
You can use this free online tool to calculate the Information Gain.

```
IG(S, Cough) = 0.04
IG(S, BreathingIssues) = 0.40
```

Since the feature **Breathing issues** have the highest Information Gain it is used to create the root node.
Hence, after this initial step our tree looks like this:

Next, from the remaining two unused features, namely, **Fever** and **Cough**, we decide which one is the best for the left branch of **Breathing Issues**.
Since the left branch of **Breathing Issues** denotes **YES,** we will work with the subset of the original data i.e the set of rows having **YES** as the value in the Breathing Issues column. These **8 rows** are shown below:
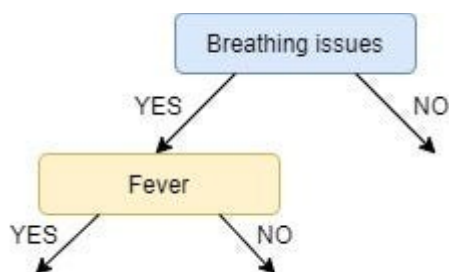
```
+-------+-------+------------------+----------+
```

```
| Fever | Cough | Breathing issues | Infected |
+-------+-------+------------------+----------+
| YES   | YES   | YES              | YES      |
+-------+-------+------------------+----------+
| YES   | NO    | YES              | YES      |
+-------+-------+------------------+----------+
| YES   | YES   | YES              | YES      |
+-------+-------+------------------+----------+
| YES   | NO    | YES              | YES      |
+-------+-------+------------------+----------+
| YES   | NO    | YES              | YES      |
+-------+-------+------------------+----------+
| NO    | YES   | YES              | YES      |
+-------+-------+------------------+----------+
| NO    | YES   | YES              | YES      |
+-------+-------+------------------+----------+
| NO    | YES   | YES              | NO       |
+-------+-------+------------------+----------+
```

Next, we calculate the IG for the features Fever and Cough using the subset S_{BY} (Set **Breathing Issues Y**es) which is shown above :

> Note: For **IG** calculation the Entropy will be calculated from the subset S_{BY} and not the original dataset **S.**

```
IG(SBY, Fever) = 0.20
IG(SBY, Cough) = 0.09
```

IG of Fever is greater than that of Cough, so we select **Fever** as the left branch of Breathing Issues: Our tree now looks like this:

Next, we find the feature with the maximum IG for the right branch of **Breathing Issues**. But, since there is only one unused feature left we have no other choice but to make it the right branch of the root node.
So our tree now looks like this:
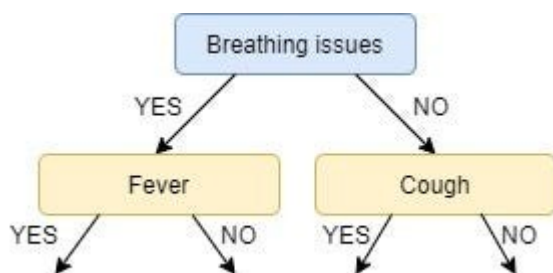
There are no more unused features, so we stop here and jump to the final step of creating the leaf nodes.

For the left leaf node of Fever, we see the subset of rows from the original data set that has **Breathing Issues** and **Fever** both values as **YES**.

```
+-------+-------+------------------+----------+
| Fever | Cough | Breathing issues | Infected |
+-------+-------+------------------+----------+
| YES   | YES   | YES              | YES      |
+-------+-------+------------------+----------+
| YES   | NO    | YES              | YES      |
+-------+-------+------------------+----------+
| YES   | YES   | YES              | YES      |
+-------+-------+------------------+----------+
| YES   | NO    | YES              | YES      |
+-------+-------+------------------+----------+
| YES   | NO    | YES              | YES      |
+-------+-------+------------------+----------+
```

Since all the values in the target column are **YES,** we label the left leaf node as **YES**, but to make it more logical we label it **Infected.**

Similarly, for the right node of Fever we see the subset of rows from the original data set that have **Breathing Issues** value as **YES** and **Fever** as **NO**.
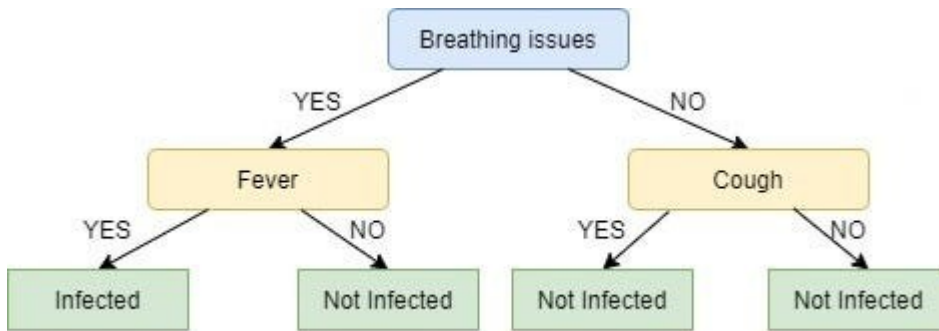
```
+-------+-------+------------------+----------+
| Fever | Cough | Breathing issues | Infected |
+-------+-------+------------------+----------+
| NO    | YES   | YES              | YES      |
+-------+-------+------------------+----------+
| NO    | YES   | YES              | NO       |
+-------+-------+------------------+----------+
| NO    | YES   | YES              | NO       |
+-------+-------+------------------+----------+
```

Here not all but **most** of the **values** are **NO,** hence **NO** or **Not Infected** becomes our **right leaf node.**
Our tree, now, looks like this:
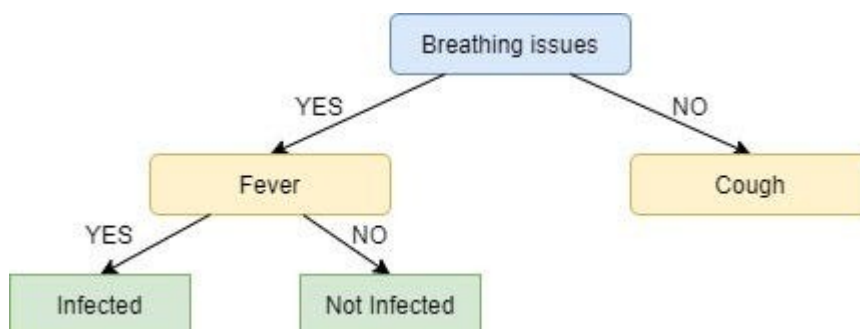
We repeat the same process for the node **Cough**, however here both left and right leaves turn out to be the same i.e. **NO** or **Not Infected** as shown below:

Looks Strange, doesn't it?
I know! The right node of Breathing issues is as good as just a leaf node with class 'Not infected'. This is one of the Drawbacks of ID3, it doesn't do pruning.

Pruning is a mechanism that reduces the size and complexity of a Decision tree by removing unnecessary nodes. More about pruning can be found here.

Another drawback of ID3 is overfitting or high variance i.e. it learns the dataset it used so well that it fails to generalize on new data.

Classification using the ID3 algorithm

Consider whether a dataset based on which we will determine whether to play football or not.

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
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

Here There are for independent variables to determine the dependent variable. The independent variables are Outlook, Temperature, Humidity, and Wind. The dependent variable is whether to play football or not.

As the first step, we have to find the parent node for our decision tree. For that follow the steps:

Find the entropy of the class variable.

$E(S) = -[(9/14)\log(9/14) + (5/14)\log(5/14)] = 0.94$

note: Here typically we will take log to base 2.Here total there are 14 yes/no. Out of which 9 yes and 5 no.Based on it we calculated probability above.

From the above data for outlook we can arrive at the following table easily

		play		
		yes	no	total
	sunny	3	2	5
Outlook	overcast	4	0	4
	rainy	2	3	5
				14

Now we have to calculate average weighted entropy. ie, we have found the total of weights of each feature multiplied by probabilities.

$E(S, \text{outlook}) = (5/14)*E(3,2) + (4/14)*E(4,0) + (5/14)*E(2,3) = (5/14)(-(3/5)\log(3/5)-(2/5)\log(2/5))+ (4/14)(0) + (5/14)((2/5)\log(2/5)-(3/5)\log(3/5)) = 0.693$

The next step is to find the information gain. It is the difference between parent entropy and average weighted entropy we found above.

$IG(S, \text{outlook}) = 0.94 - 0.693 = 0.247$

Similarly find Information gain for Temperature, Humidity, and Windy.

IG(S, Temperature) = 0.940 - 0.911 = 0.029

IG(S, Humidity) = 0.940 - 0.788 = 0.152

IG(S, Windy) = 0.940 - 0.8932 = 0.048

Now select the feature having the largest entropy gain. Here it is Outlook. So it forms the first node(root node) of our decision tree.

Now our data look as follows

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Overcast	Hot	High	Weak	Yes
Overcast	Cool	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Rain	Mild	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Since overcast contains only examples of class 'Yes' we can set it as yes. That means If outlook is overcast football will be played. Now our decision tree looks as follows.
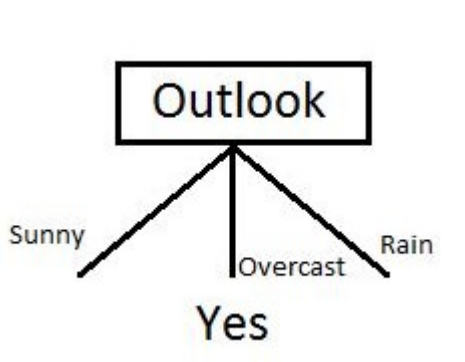
The next step is to find the next node in our decision tree. Now we will find one under sunny. We have to determine which of the following Temperature, Humidity or Wind has higher information gain.

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes

Calculate parent entropy E(sunny)

E(sunny) = (-(3/5)log(3/5)-(2/5)log(2/5)) = 0.971.

Now Calculate the information gain of Temperature. IG(sunny, Temperature)

		play		
		yes	no	total
	hot	0	2	2
Temperature	cool	1	1	2
	mild	1	0	1
				5

E(sunny, Temperature) = (2/5)*E(0,2) + (2/5)*E(1,1) + (1/5)*E(1,0)=2/5=0.4

Now calculate information gain.

IG(sunny, Temperature) = 0.971–0.4 =0.571

Similarly we get

IG(sunny, Humidity) = 0.971

IG(sunny, Windy) = 0.020

Here IG(sunny, Humidity) is the largest value. So Humidity is the node that comes under sunny.

Humidity	play	
	yes	no
high	0	3
normal	2	0

For humidity from the above table, we can say that play will occur if humidity is normal and will not occur if it is high. Similarly, find the nodes under rainy.

Note: A branch with entropy more than 0 needs further splitting.

Finally, our decision tree will look as below:
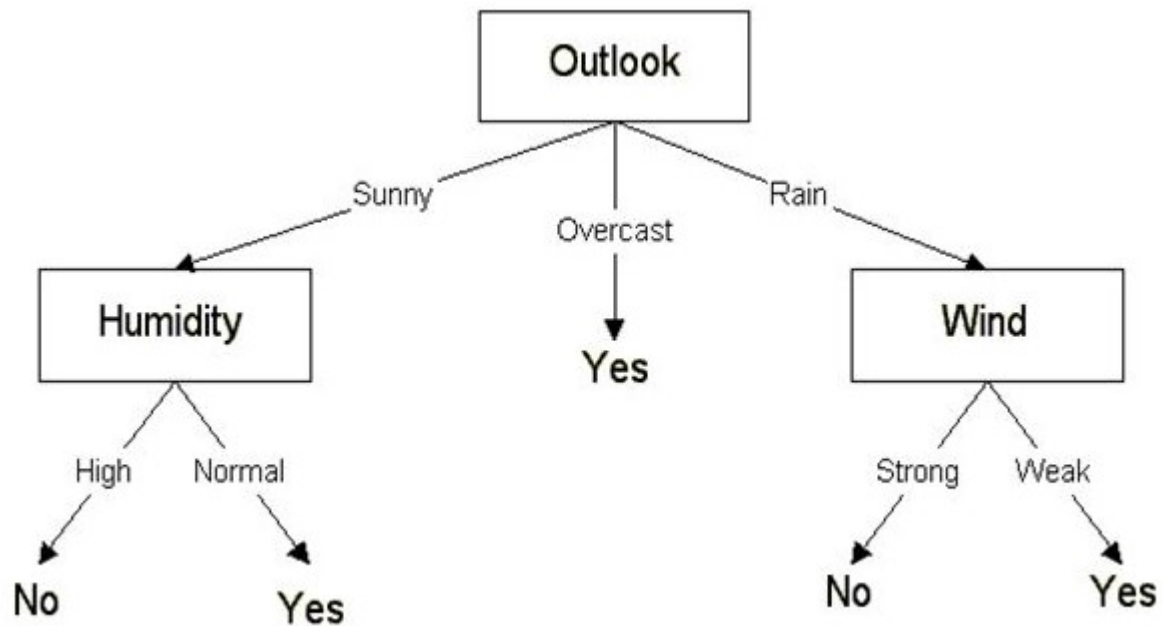
Classification using CART algorithm

Classification using CART is similar to it. But instead of entropy, we use Gini impurity.

So as the first step we will find the root node of our decision tree. For that Calculate the Gini index of the class variable

Gini(S) = 1 - [(9/14)² + (5/14)²] = 0.4591

As the next step, we will calculate the Gini gain. For that first, we will find the average weighted Gini impurity of Outlook, Temperature, Humidity, and Windy.

First, consider case of Outlook

		play		
		yes	no	total
	sunny	3	2	5
Outlook	overcast	4	0	4
	rainy	2	3	5
				14

Gini(S, outlook) = (5/14)gini(3,2) + (4/14)*gini(4,0)+ (5/14)*gini(2,3) = (5/14)(1 - (3/5)² - (2/5)²) + (4/14)*0 + (5/14)(1 - (2/5)² - (3/5)²)= 0.171+0+0.171 = 0.342

Gini gain (S, outlook) = 0.459 - 0.342 = 0.117

Gini gain(S, Temperature) = 0.459 - 0.4405 = 0.0185

Gini gain(S, Humidity) = 0.459 - 0.3674 = 0.0916

Gini gain(S, windy) = 0.459 - 0.4286 = 0.0304

Choose one that has a higher Gini gain. Gini gain is higher for outlook. So we can choose it as our root node.

Now you have got an idea of how to proceed further. Repeat the same steps we used in the ID3 algorithm.