DECISION TREES - ID3

Indeed Inspiring Infotech
www.indeedinspiring.com

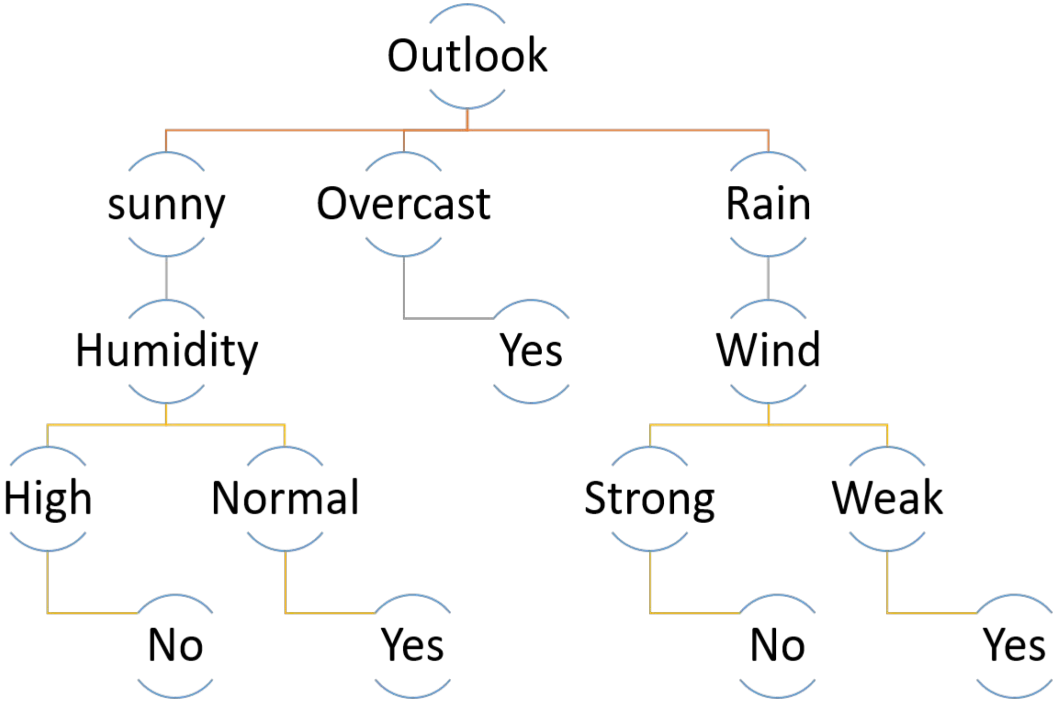
Decision Trees

Decision tree builds classification or regression models in the form of a tree structure.

It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.

The final result is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called **root node**.

Decision trees can handle both categorical and numerical data.



ID3

- In decision tree learning, ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan used to generate a decision tree from a dataset.
- It employs a top-down, greedy search through the space of possible branches with no backtracking.
- ID3 uses *Entropy* and *Information Gain* to construct a decision tree.
- ID3 is typically used in the machine learning and natural language processing domains.

Algorithm

- The ID3 algorithm begins with the original set ***S*** as the root node. On each iteration of the algorithm, it iterates through every unused attribute of the set ***S*** and calculates the entropy **H(*S*)** or the information gain ***IG*(*S*)** of that attribute.
- It then selects the attribute which has the smallest entropy (or largest information gain) value. The set ***S*** is then split or partitioned by the selected attribute to produce subsets of the data.
- For example, a node can be split into child nodes based upon the subsets of the population whose ages are less than 50, between 50 and 100, and greater than 100.
- The algorithm continues to recurse on each subset, considering only attributes never selected before.

Contd…

Recursion on a subset may stop in one of these cases:

- Every element in the subset belongs to the same class; in which case the node is turned into a leaf node and labelled with the class of the examples.
- There are no more attributes to be selected, but the examples still do not belong to the same class. In this case, the node is made a leaf node and labelled with the most common class of the examples in the subset.
- There are no examples in the subset, which happens when no example in the parent set was found to match a specific value of the selected attribute. An example could be the absence of a person among the population with age over 100 years. Then a leaf node is created and labelled with the most common class of the examples in the parent node's set.

Throughout the algorithm, the decision tree is constructed with each non-terminal node (internal node) representing the selected attribute on which the data was split, and terminal nodes (leaf nodes) representing the class label of the final subset of this branch.

Entropy

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

For example we have total 4 red and Blue balls, now we will find out entropy

- **Entropy index for 4 red and 0 blue**

Entropy = — [(Probability of red)*log2(Probability of red)] — [(Probability of blue)*log2(Probability of blue)]

Entropy = — [(4/4)*log(4/4)] — [(0/4*log2(0/4)]

Entropy = 0

- **Entropy index for 2 red and 2 blue**

Entropy = — [(Probability of red)*log2(Probability of red)] — [(Probability of blue)*log2(Probability of blue)]

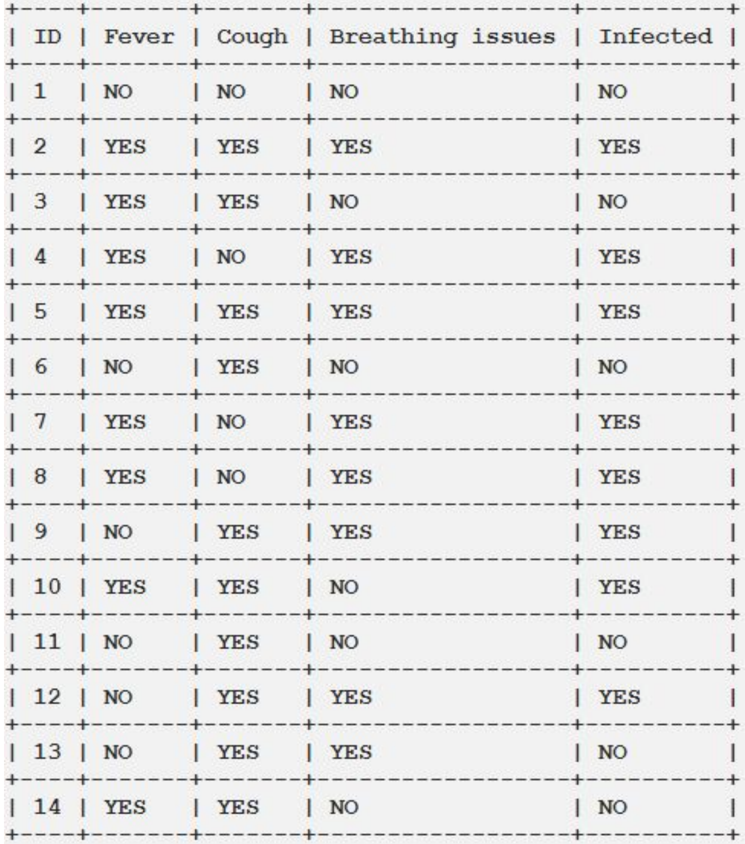
Entropy = — [(2/4)*log(2/4)] — [(2/4*log2(2/4)]

Entropy = 1

Example

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

- we’ll be using a sample dataset of COVID-19 infection. A preview of the entire dataset is shown below.
- 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.



```
+----+-------+-------+------------------+----------+
| 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       |
+----+-------+-------+------------------+----------+
```

Implementation on our dataset

- As stated previously 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**

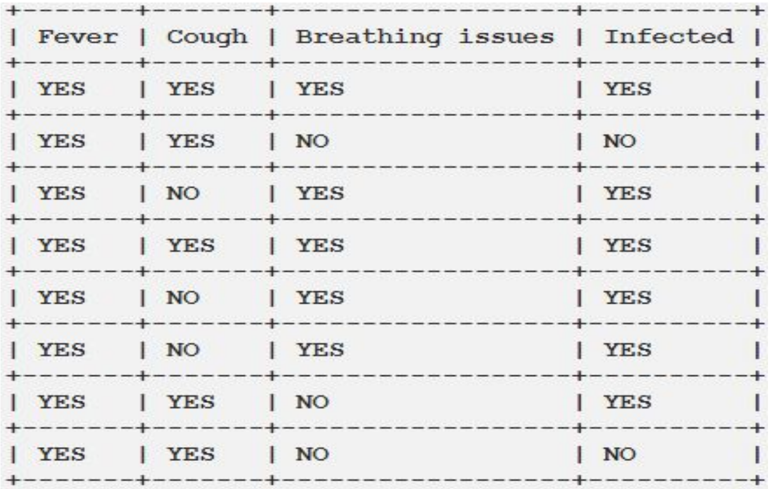
- From the total of 14 rows in our dataset **S**, there are **8** rows with the target value **YES** and **6** rows with the target value **NO**. The entropy of **S** is calculated as:

$$\textbf{Entropy(S)} = — (8/14) * \log_2(8/14) — (6/14) * \log_2(6/14) = 0.99$$

- No we will calculate 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.**
- There are **8** rows with **YES** for Fever, there are **6** rows having target value **YES** and **2** rows having target value **NO.**
- There are **6** rows with **NO** for Fever, there are **2** rows having target value **YES** and **4** 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       |
+-------+-------+--------------------+----------+
```

|S| = 14 # total rows

For v = YES, $|S_v|$ = 8

Entropy(S_v) = - (6/8) * $\log_2$(6/8) - (2/8) * $\log_2$(2/8) = 0.81

For v = NO, $|S_v|$ = 6

Entropy(S_v) = - (2/6) * $\log_2$(2/6) - (4/6) * $\log_2$(4/6) = 0.91

# Expanding the summation in the IG formula:

IG(S, Fever) = Entropy(S) - (|Sʏᴇ⍰| / |S|) * Entropy(Sʏᴇ⍰) - (|Sɴᴏ| / |S|) * Entropy(Sɴᴏ)

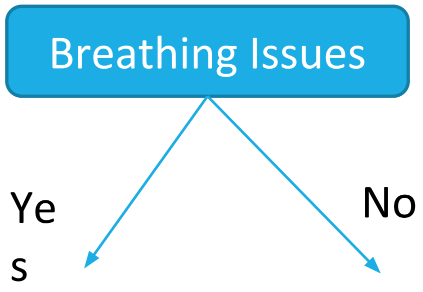
∴ IG(S, Fever) = 0.99 - (8/14) * 0.81 - (6/14) * 0.91 = 0.13

Contd...

- Next, we calculate the **IG** for the features **“Cough”** and **“Breathing issues”.**

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

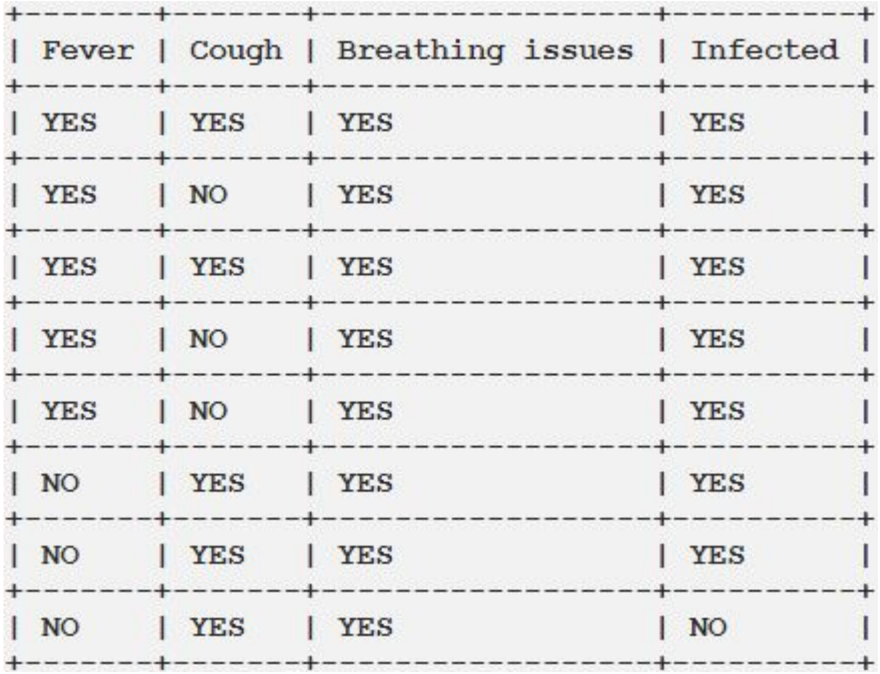
- Since the feature **Breathing issues** have the highest Information Gain it is used to create the root node.
  Hence, after this initial step our tree looks like this:



- 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.

- Next, we calculate the IG for the features Fever and Cough using the subset **S**BY (**S**et **Breathing** Issues **Y**es) which is shown above

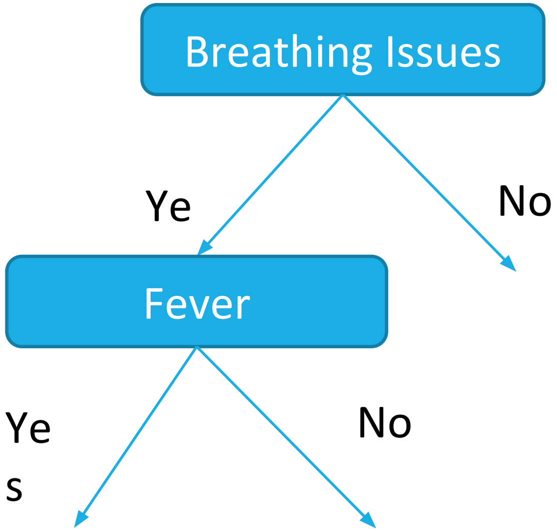
IG(SBY, Fever) = 0.20
IG(SBY, Cough) = 0.09



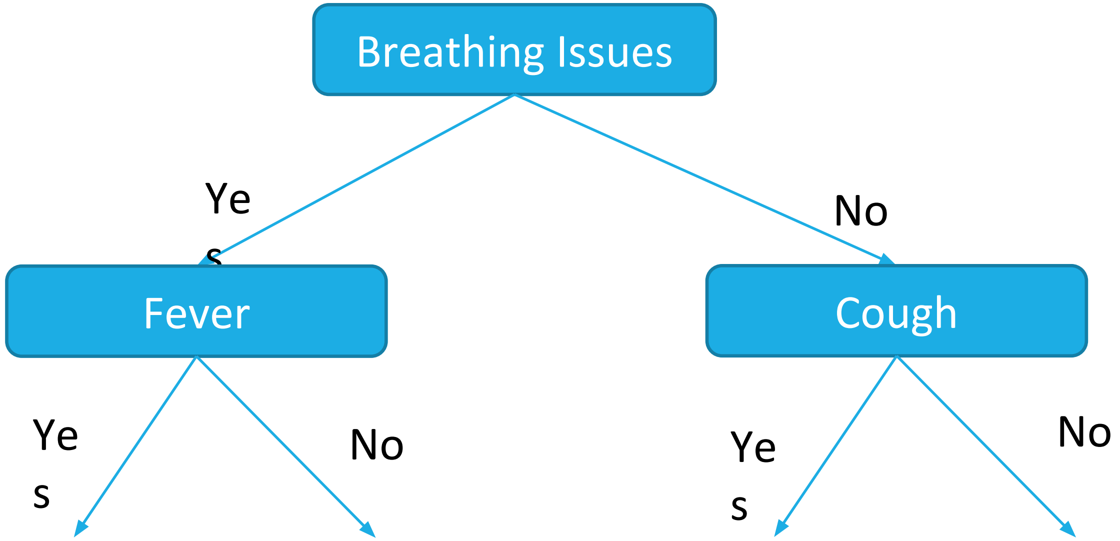
```
+--------+--------+------------------+-----------+
| Fever  | Cough  | Breathing issues | Infected  |
+--------+--------+------------------+-----------+
| YES    | YES    | YES              | YES       |
+--------+--------+------------------+-----------+
| YES    | NO     | YES              | YES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

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



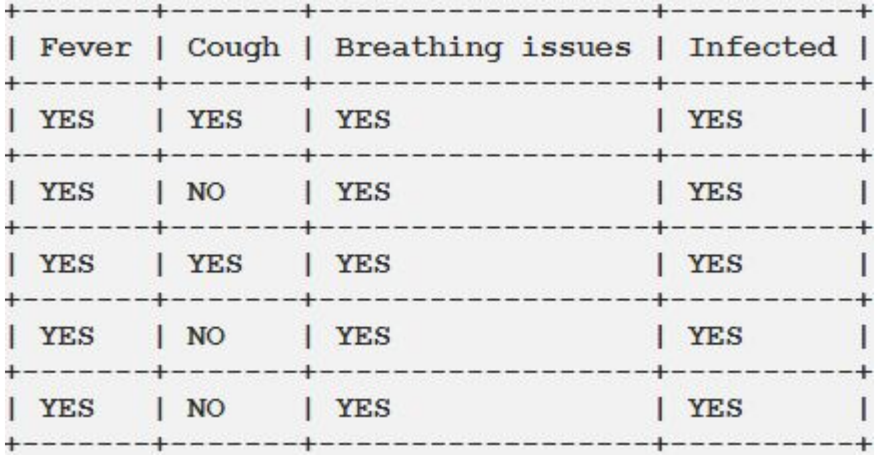
Next, we find the feature with the maximum IG for the right branch of **Breathing Issues**. But, since there is only one unused feature left we have no other choice but to make it the right branch of the root node.
So our tree now looks like this



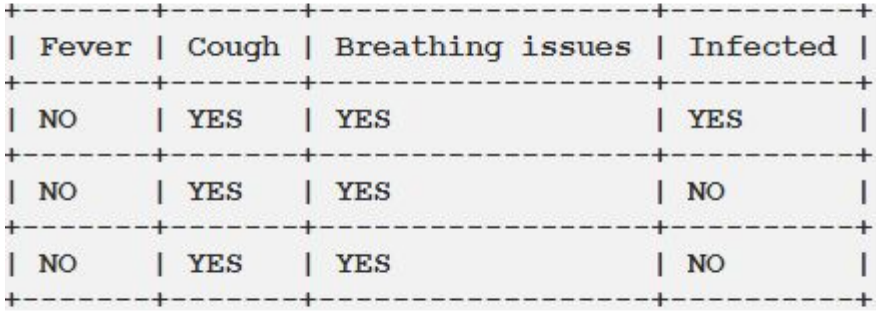
- There are no more unused features, so we stop here and jump to the final step of creating the leaf nodes.
  For the left leaf node of Fever, we see the subset of rows from the original data set that has **Breathing Issues** and **Fever** both values as **YES**.

- 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**.



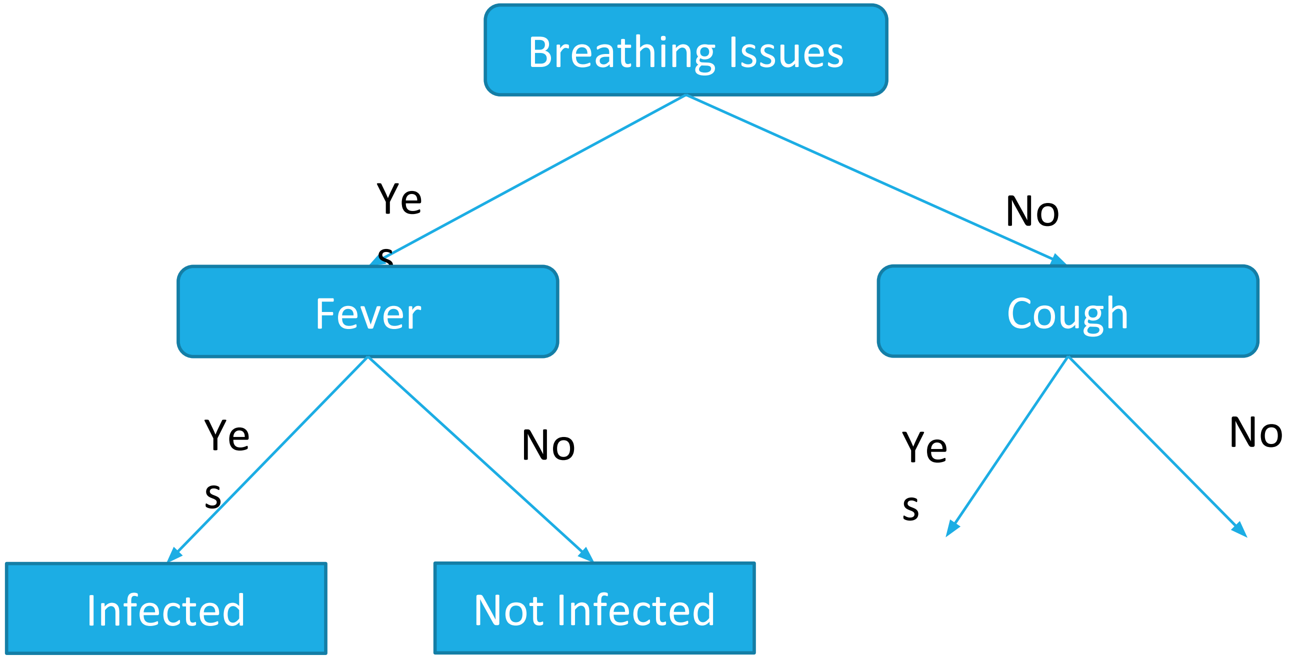
Fever	Cough	Breathing issues	Infected
YES	YES	YES	YES
YES	NO	YES	YES
YES	YES	YES	YES
YES	NO	YES	YES
YES	NO	YES	YES



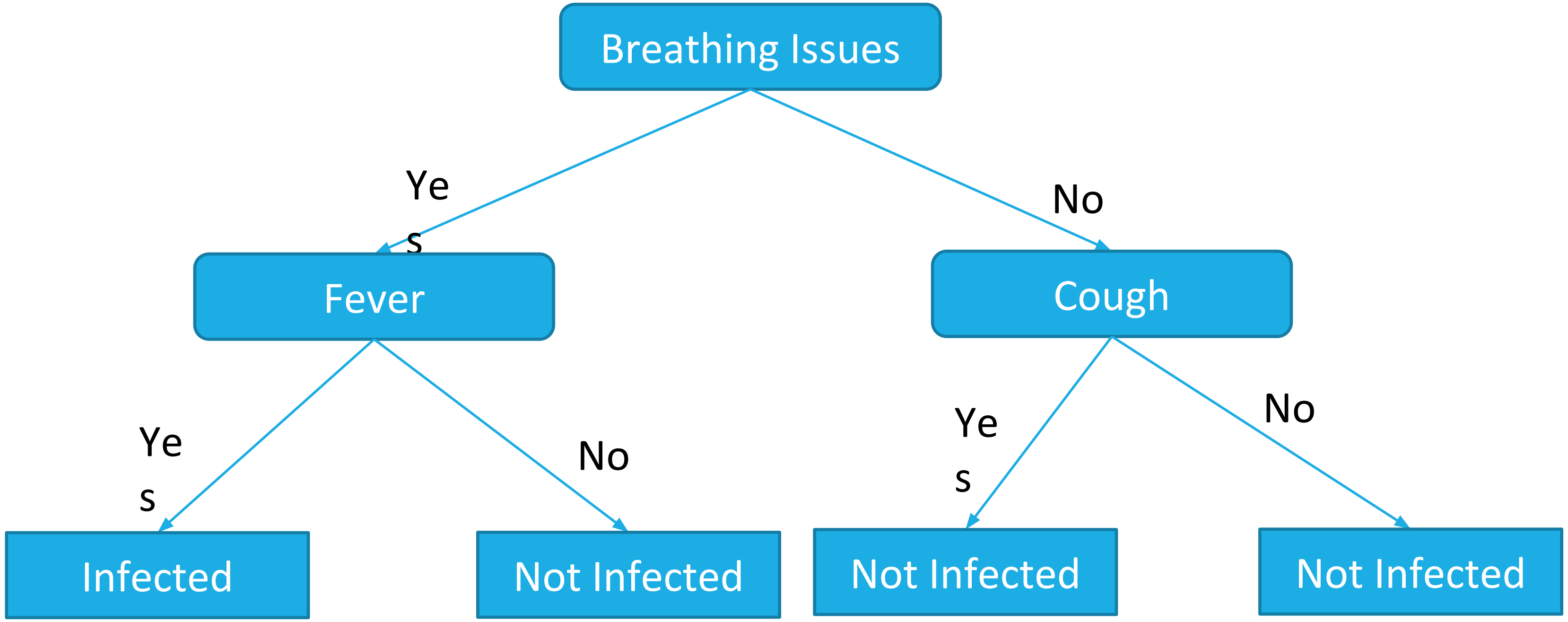
Fever	Cough	Breathing issues	Infected
NO	YES	YES	YES
NO	YES	YES	NO
NO	YES	YES	NO

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:



Pseudocode

```
ID3 (Examples, Target_Attribute, Attributes)
    Create a root node for the tree
    If all examples are positive, Return the single-node tree Root, with label = +.
    If all examples are negative, Return the single-node tree Root, with label = -.
    If number of predicting attributes is empty,
        then Return the single node tree Root, with label = most common value of the target attribute in the examples.
   Otherwise Begin
       A ← The Attribute that best classifies examples.
       Decision Tree attribute for Root = A.
For each possible value,Vi of A,
          Add a new tree branch below Root, corresponding to the test A =Vi.
          Let Examples Vi be the subset of examples that have the value Vi for A
If Examples Vi is empty
             Then below this new branch add a leaf node with label = most common target value in the examples
          Else below this new branch add the subtree ID3 (Examples V , Target_Attribute, Attributes – {A})
    End
    Return Root
```

Properties

- It uses a greedy strategy by selecting the locally best attribute to split the dataset on each iteration.
- The algorithm's optimality can be improved by using backtracking during the search for the optimal decision tree at the cost of possibly taking longer.
- ID3 can overfit the training data. To avoid overfitting, smaller decision trees should be preferred over larger ones.
- ID3 is harder to use on continuous data than on factored data (factored data has a discrete number of possible values, thus reducing the possible branch points)

Applications:

- ID3-generated decision tree used to determine whether a particular nucleotide pair within a pre-mRNA sequence corresponds to an mRNA splice site.
- Web Attack Detection Using ID3. ID3 was able to classify even unseen Web application queries as an attack.
- ID3 in Identifying Cancer. Here ID3 is used to split training examples in to target classes, the one which gives highest classification is selected and used.
- Application of ID3 in Educational Field : ID3 can be used in the field of education for Placement analysis of fourth year students by classifying their overall performances and also to identify the first year student’s dropout classification.