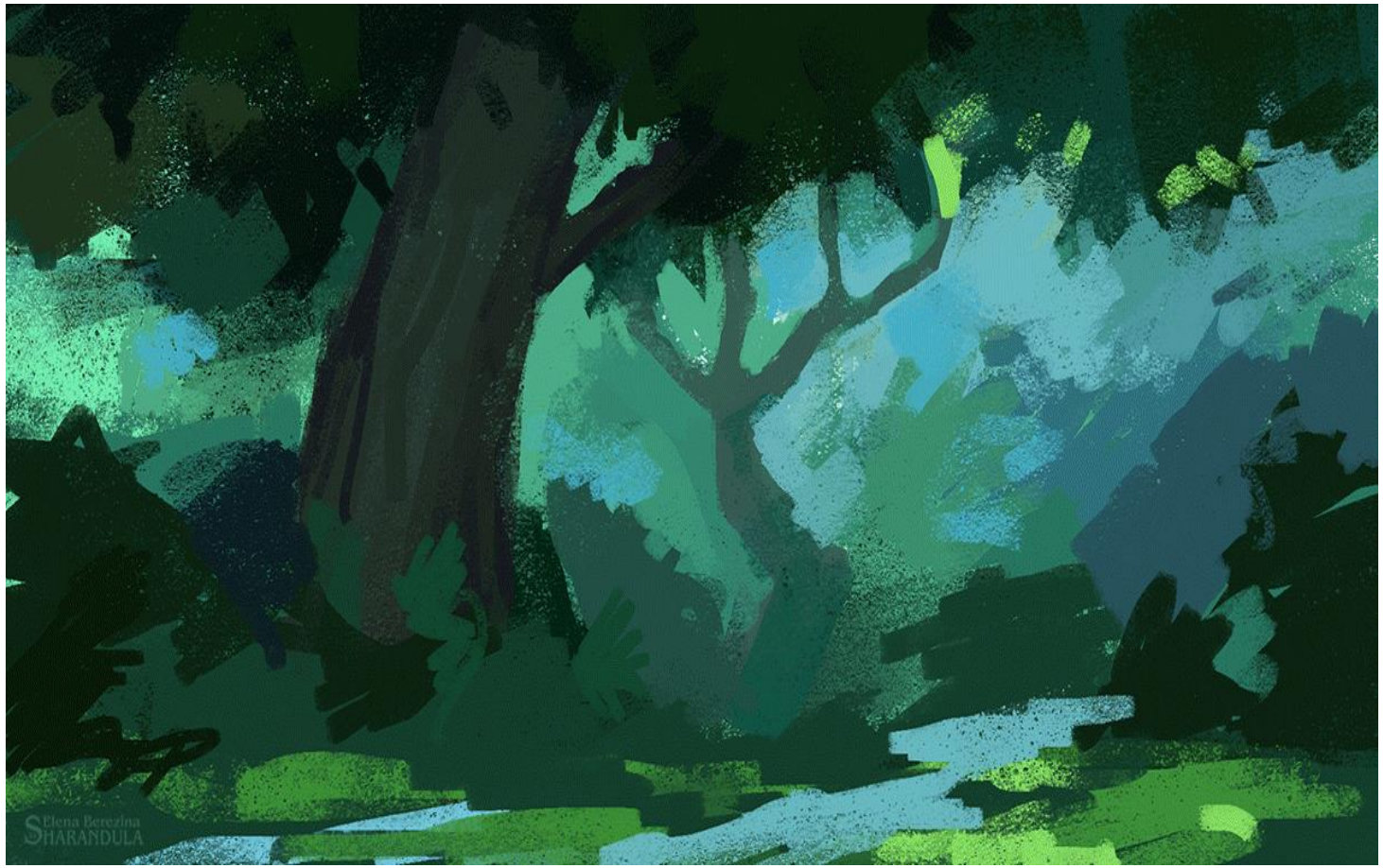
Decision Trees and Random Forest

TOP MENTOR

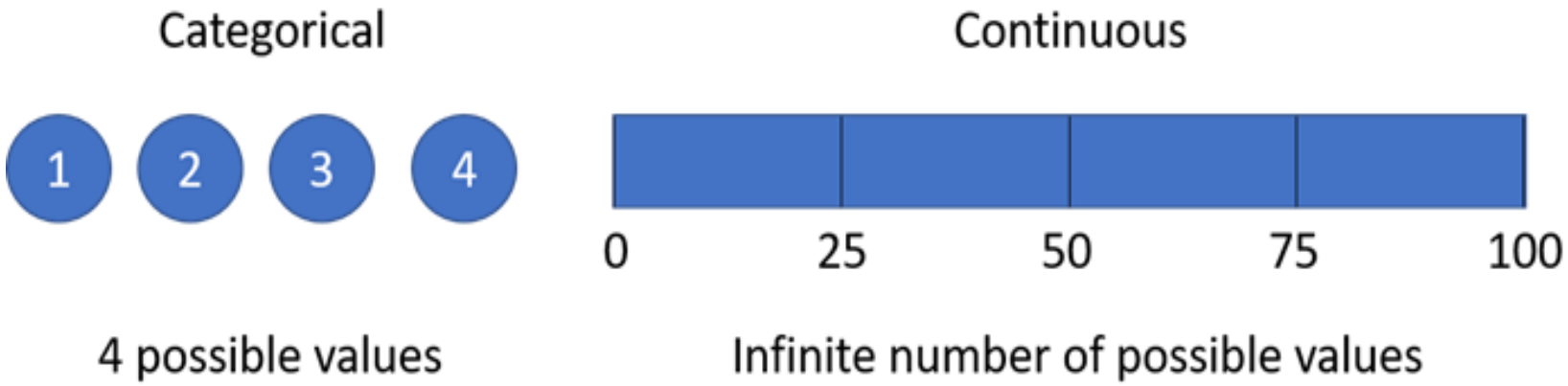


Tree Based Algorithm

Trees provide us with many benefits necessary for survival, including clean air, filtered water, shade, and food. They also give us hope and insight, and courage to persevere — even in the harshest conditions. Trees teach us to stay rooted while soaring to great heights.

Tree based learning algorithms are considered to be one of the best and mostly used supervised learning methods. Tree based methods empower predictive models with high accuracy, stability and ease of interpretation.

Decision Tree Algorithm



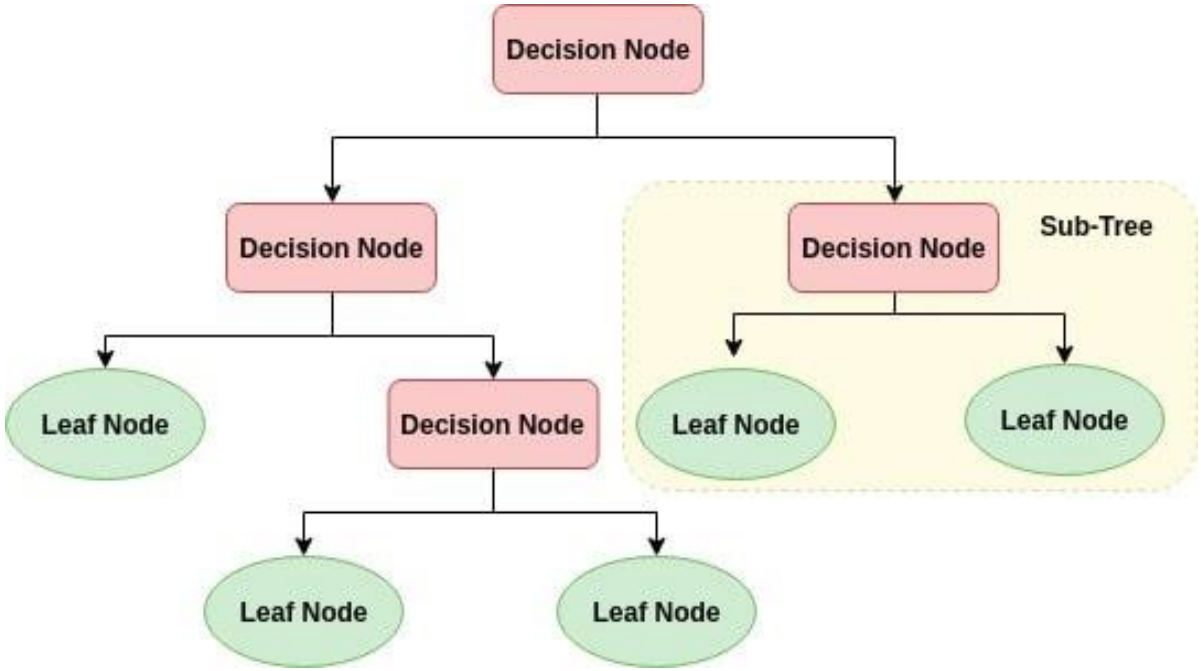
Decision Trees can be utilized for both classification and regression kind of problem.

What is Decision Tree Algorithm

A decision tree is a flowchart-like tree structure where an internal node represents feature (or attribute), the branch represents a decision rule, and each leaf node represents the outcome

Top-Down approach

The topmost node in a decision tree is known as the root node. It learns to partition based on the attribute value. It partitions the tree in a recursive manner called recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human-level thinking. That is why decision trees are easy to understand and interpret.



Terminologies

1.Root Node (Top Decision Node): It represents the entire population or sample and this further gets divided into two or more homogeneous sets.

2.Splitting: It is a process of dividing a node into two or more sub-nodes.

3.Decision Node: When a sub-node splits into further sub-nodes, then it is called a decision node.

4.Leaf/ Terminal Node: Nodes with no children (no further split) is called Leaf or Terminal node.

5.Pruning: When we reduce the size of decision trees by removing nodes (opposite of Splitting), the process is called pruning.

6.Branch / Sub-Tree: A sub section of the decision tree is called branch or sub-tree.

7.Parent and Child Node: A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.

How does the Decision Tree algorithm work?

The basic idea behind any decision tree algorithm is as follows:

1.Select the best attribute using Attribute Selection Measures(ASM) to split the records.

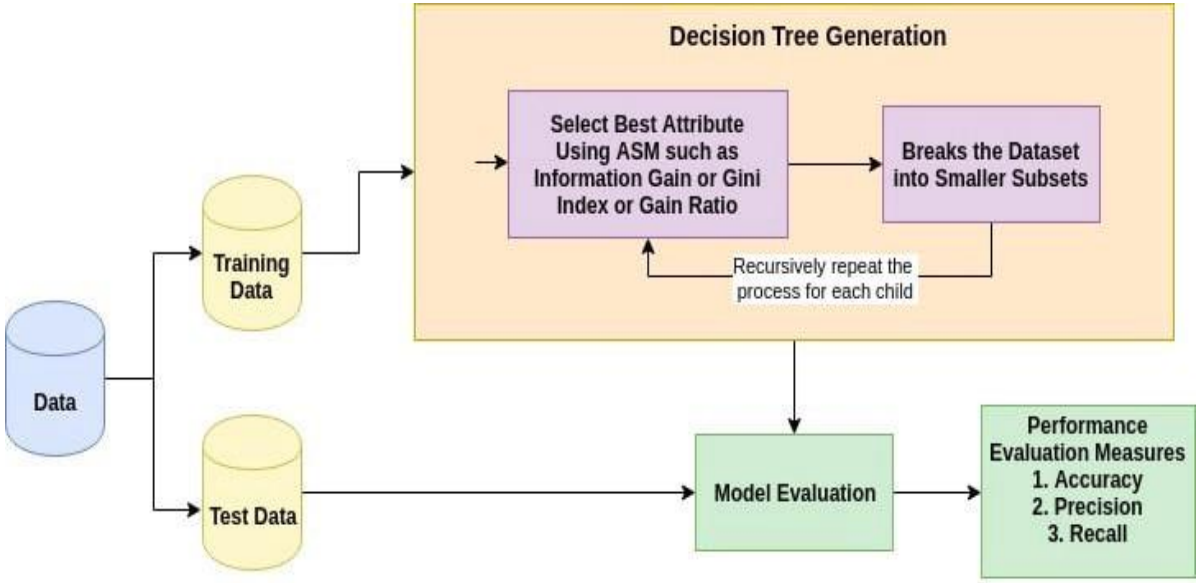
2.Make that attribute a decision node and breaks the dataset into smaller subsets.

3.Starts tree building by repeating this process recursively for each child until one of the condition will match:

o All the tuples belong to the same attribute value.

o There are no more remaining attributes.

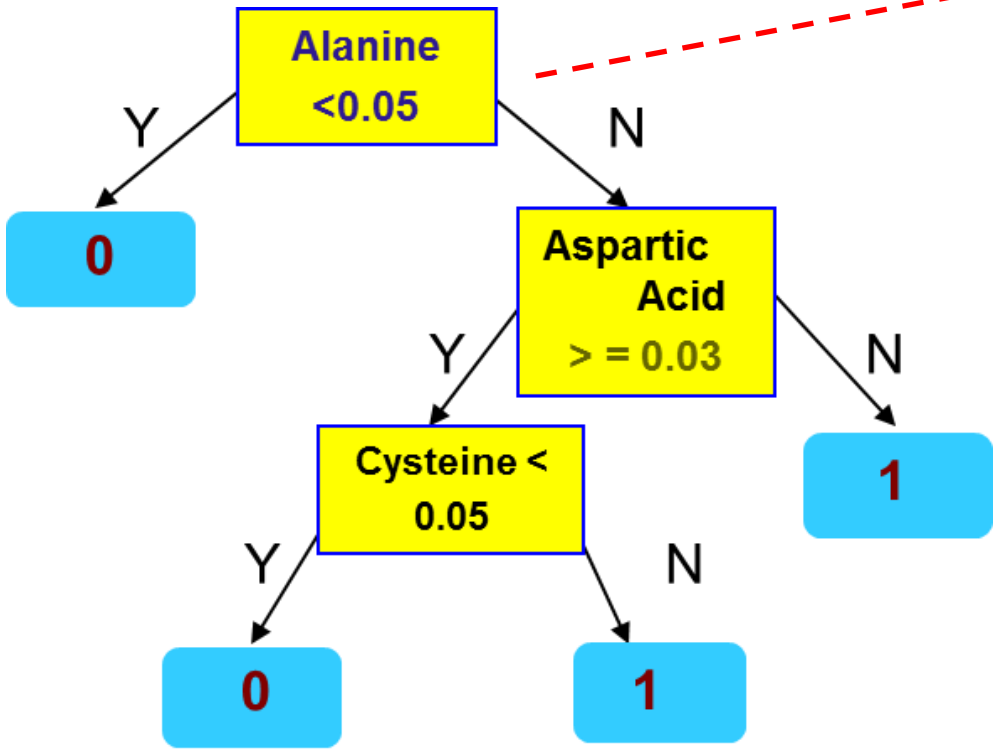
o There are no more instances.



Example

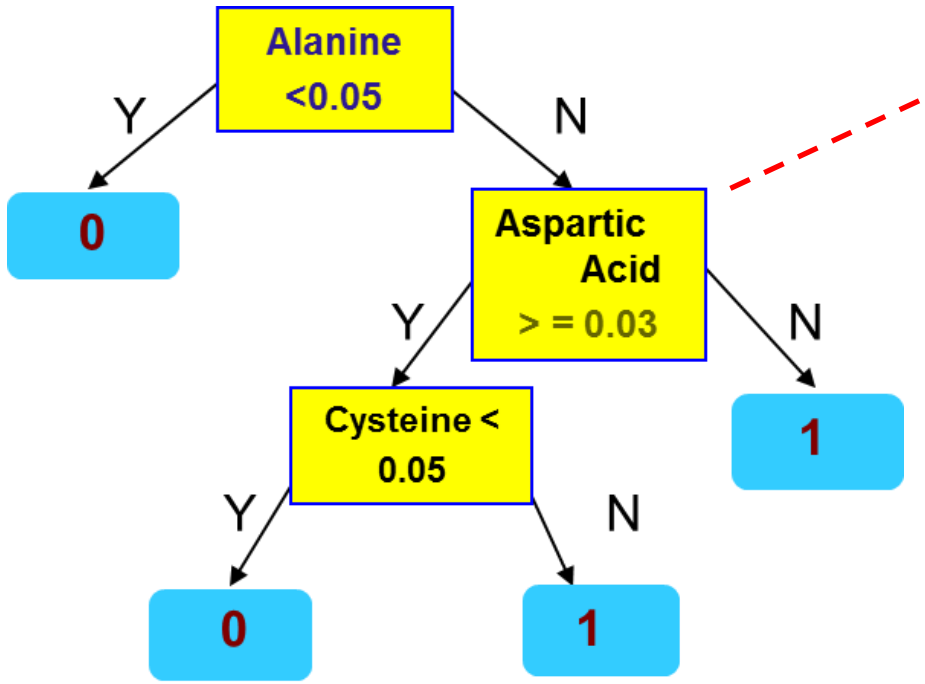
Test Data

Id	*Alanine*	*Cysteine*	*Aspartic Acid*	*Defensin*
1	0.19	0.06	0.04	?



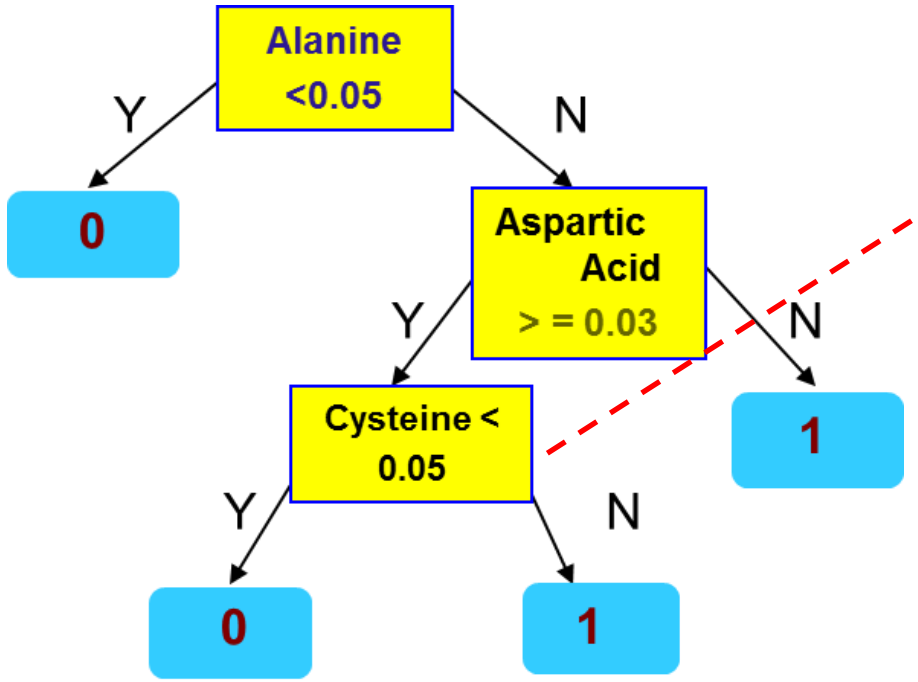
Test Data

Id	Alanine	Cysteine	Aspartic Acid	Defensin
1	0.19	0.06	0.04	?



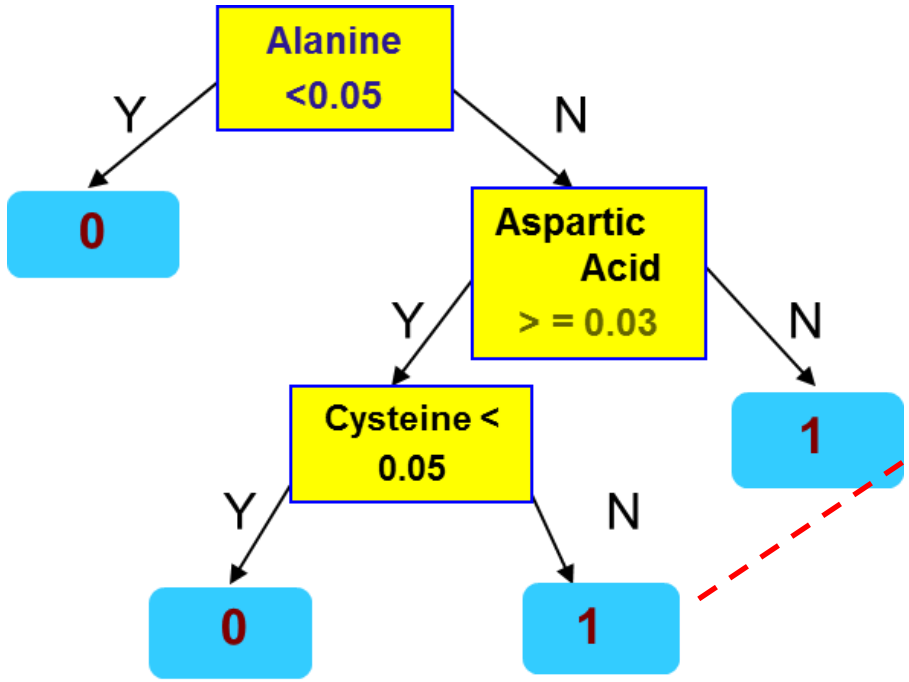
Test Data

Id	*Alanine*	*Cysteine*	*Aspartic Acid*	*Defensin*
1	0.19	0.06	0.04	?



Test Data

Id	*Alanine*	*Cysteine*	*Aspartic Acid*	*Defensin*
1	0.19	0.06	0.04	1



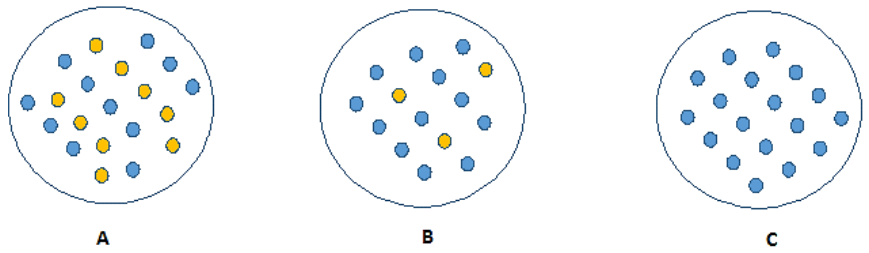
Information Gain

Attribute Selection Measures

Attribute selection measure is a heuristic for selecting the splitting criterion that partition data into the best possible manner. It is also known as splitting rules because it helps us to determine breakpoints for tuples on a given node. ASM provides a rank to each feature(or attribute) by explaining the given dataset. Best score attribute will be selected as a splitting attribute

1.Information Gain:

Information gain is a statistical property that measures how well a given attribute separates the training examples according to their target classification. Look at the image below and think which node can be described easily. I am sure, your answer is C because it requires less information as all values are similar. On the other hand, B requires more information to describe it and A requires the maximum information. In other words, we can say that C is a Pure node, B is less Impure and A is more impure.



Now, we can build a conclusion that less impure node requires less information to describe it. And, the more impure node requires more information.

Entropy

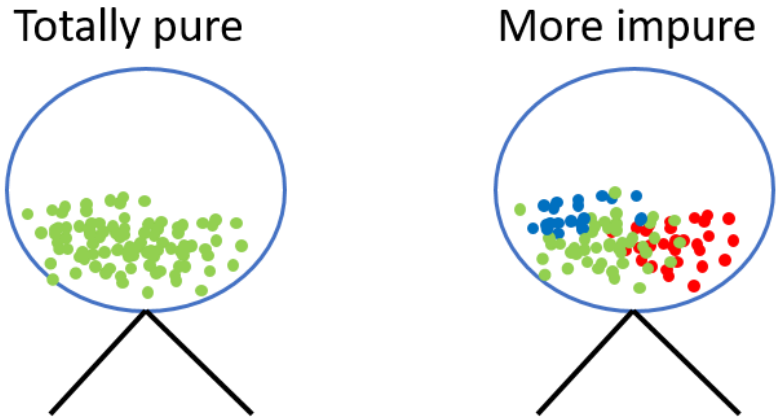
In the figure below, we can see that an attribute with low information gain (bottom) splits the data relatively evenly and as a result doesn't bring us any closer to a decision. Whereas, an attribute with high information gain (top) splits the data into groups with an uneven number of positives and negatives and as a result, helps in separating the two from each other.

To be able to calculate the information gain, we have to first introduce the term ***entropy of a dataset***.

Entropy

Shannon invented the concept of entropy, which measures the impurity of the input set. In physics and mathematics, entropy referred to as the randomness or the impurity in the system. In information theory, it refers to the impurity in a group of examples. Information gain is a decrease in entropy.

Imagine you have a lottery wheel which includes 100 green balls. The set of balls within the lottery wheel can be said to be totally pure because only green balls are included. To express this in the terminology of entropy, this set of balls has an entropy of 0 (we can also say zero impurity). Consider now, 30 of these balls are replaced by red and 20 by blue balls.



If you now draw another ball from the lottery wheel, the probability of receiving a green ball has dropped ***from 1.0 to 0.5***. Since the impurity increased, the purity decreased, hence also the entropy increased. Hence we can say, the more “impure” a dataset, the higher the entropy and the less “impure” a dataset, the lower the entropy

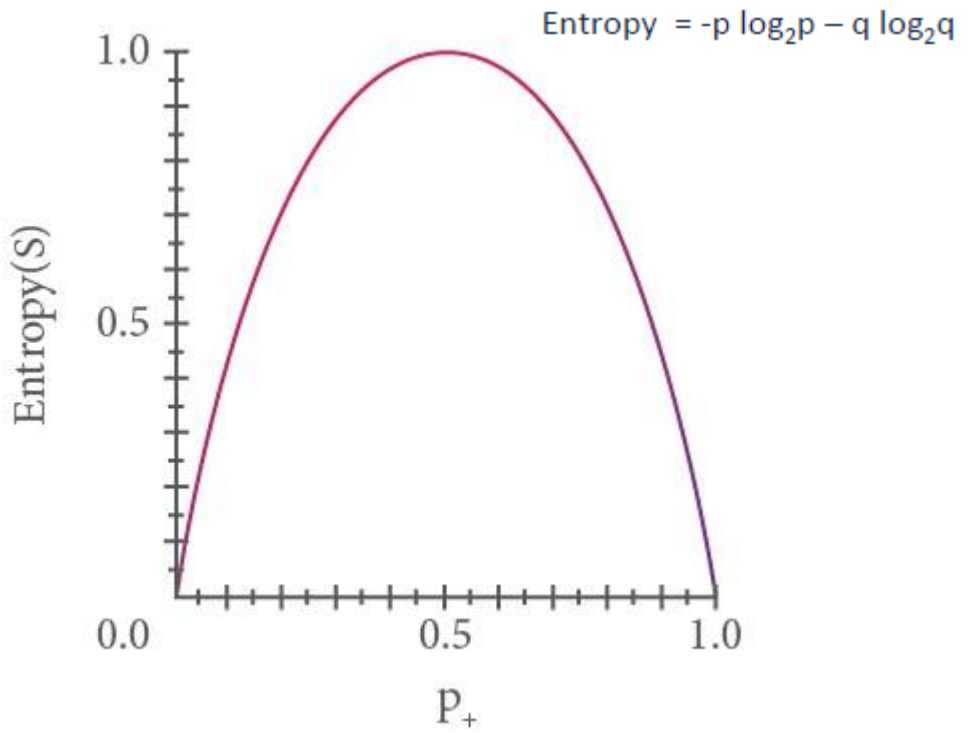
Entropy

Note that entropy is 0 if all the members of S belong to the same class. For example, if all members are positive, Entropy(S) = 0. Entropy is 1 when the sample contains an equal number of positive and negative examples. If the sample contains an unequal number of positive and negative examples, entropy is between 0 and 1. The following figure shows the form of the entropy function relative to a Boolean classification as entropy varies between 0 and 1.

Here p and q is the probability of success and failure respectively in that node. Entropy is also used with the categorical target variable. It chooses the split which has the lowest entropy compared to the parent node and other splits. **The lesser the entropy, the better it is.**

Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values

Information Gain = Entropy(parent node) — [Avg Entropy(children)]



Example - 2

Classify Bank Loan Applications by assigning Applications to one of the three risk classes

Owns Home	Married	Gender	Employed	Credit Rating	Risk Class
Yes	Yes	Male	Yes	A	B
No	No	Female	Yes	A	A
Yes	Yes	Female	Yes	B	C
Yes	No	Male	No	B	B
No	Yes	Female	Yes	B	C
No	No	Female	Yes	B	A
No	No	Male	No	B	B
Yes	No	Female	Yes	A	A
No	Yes	Female	Yes	A	C
Yes	Yes	Female	Yes	A	C

Entropy= -(n/s) log(n/s) –(p/s) log(p/s)

Assume there are two classes P and N and let the set of training data S (with a total number of object s) contain p element of class P and n elements of Class N

There are 10 (s = 10) samples and three classes. The frequencies of these classes are :

A = 3
B = 3
C = 4

Information in the data due to uncertainity of outcome regarding the risk class each person belongs to is given by:

E = -(3/10) log(3/10) –(3/10) log(3/10) – (4/10) log(4/10) = 1.57

Risk Class
B
A
C
B
C
A
B
A
C
C

1. Attribute : "Owns Home"

Value = Yes . There are 5 applicants who own their home. They are in class:

A = 1, B = 2, C = 2 total = 5 (p = 5/10)

Value = No . There are 5 applicants who own their home. They are in class:

A = 2, B = 1, C = 2 total = 5 (p = 5/10)

Given the above values, it does not appear as if this attribute will reduce the uncertainty by much. Computing the information for each of these two sub trees.

I (Yes) = I(Y) = -(1/5) log(1/5) – (2/5) log(2/5) - (2/5) log(2/5) = 1.52

I (No) = I(N) = -(2/5) log(2/5) – (1/5) log(1/5) - (2/5) log(2/5) = 1.52

Total Entropy of the two sub trees = 0.5*I(Y) + 0.5 *I(N) = 1.52

Owns Home
Yes
No
Yes
Yes
No
No
No
Yes
No
Yes

2. Attribute : “Married”

Value = Yes . There are 5 applicants who are married and five are not. They are in class:

A = 0, B = 1, C = 4 total = 5 (p = 5/10)

Value = No . There are 5 applicants who are not married. They are in class:

A = 3, B = 2, C = 0 total = 5 (p = 5/10)

Given the above values, it appears that this attribute will reduce the uncertainty by more than the last attribute. Computing the information for each of these two sub trees.

I (Yes) = I(Y) = -(1/5) log(1/5) – (4/5) log(4/5) = 0.722

I (No) = I(N) = -(3/5) log(3/5) – (2/5) log(2/5) = 0.971

Total Entropy of the two sub trees = 0.5*I(Y) + 0.5 *I(N) = 0.846

Married
Yes
No
Yes
No
Yes
No
No
No
Yes
Yes

3. Attribute : “Gender”

Value = Male . There are 3 applicants who are Male. They are in class:

A = 0, B = 3, C = 0 total = 3 (p = 3/10)

Value = Female . There are 7 applicants who are Female. They are in class:

A = 3, B = 0, C = 4 total = 5 (p = 7/10)

Given the above values, it appears that this attribute will reduce the uncertainty even more by using attribute since for value = MALE we have only one class. Computing the information for each of these two sub trees.

I (Male) = I(Y) = 0

I (No) = I(N) = -(3/7) log(3/7) – (4/7) log(4/7) = 0.985

Total Entropy of the two sub trees = 0.3*I(Y) + 0.7 *I(N) = 0.69

Male
Female
Female
Male
Female
Female
Male
Female
Female
Female

The values for the information gain can be computed:

Potential Split Attribute	Information before Split	Information After Split	Information Gain
Owns Home	1.57	1.52	0.05
Married	1.57	0.85	0.72
Gender	1.57	0.69	0.88
Employed	1.57	1.12	0.45
Credit rating	1.57	1.52	0.05

Hence the largest information gain is provided by the attribute “Gender” and that is the attribute that is used for the split.

Now we can reduce the data by removing the attribute Gender and removing the class B since all class B have gender = Male.

Owns Home	Married	Employed	Credit Rating	Risk Class
No	No	Yes	A	A
Yes	Yes	Yes	B	C
No	Yes	Yes	B	C
No	No	Yes	B	A
Yes	No	Yes	A	A
No	Yes	Yes	A	C
Yes	Yes	Yes	A	C

The information in this data of two classes due to uncertainty of outcome regarding the class each person belongs to is given by

$$I = -(3/7)\log(3/7) - (4/7)\log(4/7) = 1.33$$

1. Attribute : "Owns Home"

Value = Yes . There are 3 applicants who own their home. They are in class:

A = 1, C = 2 total = 3 (p = 3/7)

Value = No . There are 5 applicants who own their home. They are in class:

A = 2, C = 2 total = 5 (p = 4/7)

Given the above values, it does not appear as if this attribute will reduce the uncertainty by much. Computing the information for each of these two sub trees.

I (Yes) = I(Y) = -(1/3) log(1/3) – (2/3) log(2/3) = 0.92

I (No) = I(N) = -(2/4) log(2/4) – (2/4) log(2/4) = 1.00

Total Information of the two sub trees = 0.5*I(Y) + 0.5 *I(N) = 0.96

Owns Home
No
Yes
No
No
Yes
No
Yes

2. Attribute : “Married”

Value = Yes . There are 4 applicants who are married and three are not. They are in class:

A = 0, C = 4 total = 4

Value = No . There are 5 applicants who own their home. They are in class:

A = 3, C = 0 total = 3

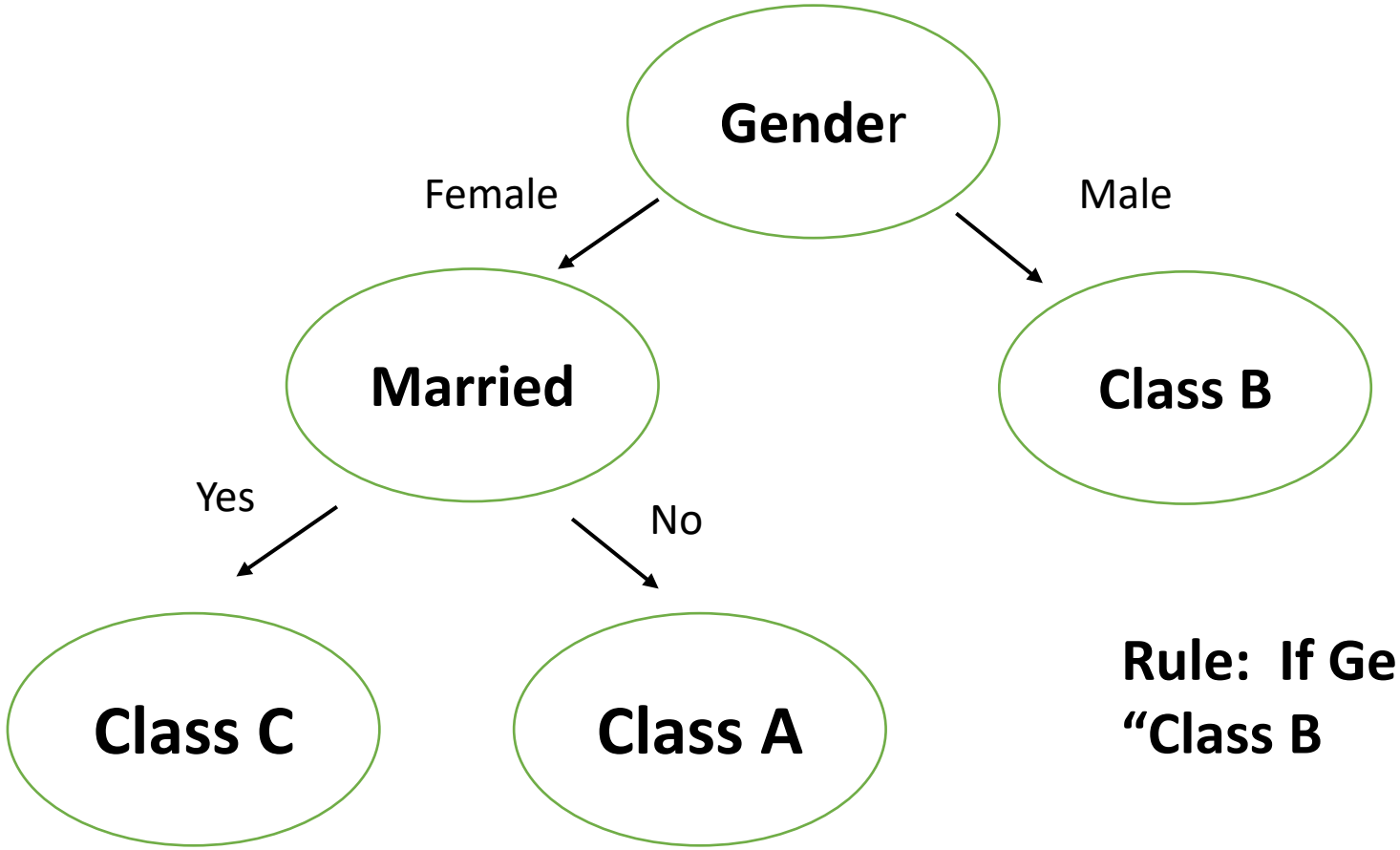
Looking at the values it appears that this attribute will reduce the uncertainty by a lot more than the last attribute since for each value the person belong to only one class and therefore information is zero.

I (Yes) = I(Y) = -(4/4) log(4/4) = 0.0

I (No) = I(N) = -(3/3) log(3/3) = 0.0

Total Information of the two sub trees = 0.5*I(Y) + 0.5 *I(N) = 0.0

There is no need to consider other attributes now , since no other attribute can be better. The split attribute therefore is **“Married”**. It should be noted that a number of attributes that were available in the data were not required in classifying it.



Rule: If Gender = Male then “Class B

Gini Index

Gini says, if we select two items from a population at random then they must be of same class and probability for this is 1 if population is pure.
It works with categorical target variable “Success” or “Failure”.
It performs only Binary splits
Higher the value of Gini higher the homogeneity.
CART (Classification and Regression Tree) uses the Gini method to create binary splits

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

$$= 1 - \sum_{t=0}^{t=1} P_t^2$$

Out of 14 instances ,
yes=9,no=5
1 - (9/14)^2- (5/14)^2

1-0.413-0.127= 0.46
Gini = 0.46

Steps to Calculate Gini for a split

Calculate Gini for sub-nodes, using formula sum of the square of probability for success and failure (p^2+q^2).

Calculate Gini for split using weighted Gini score of each node of that split

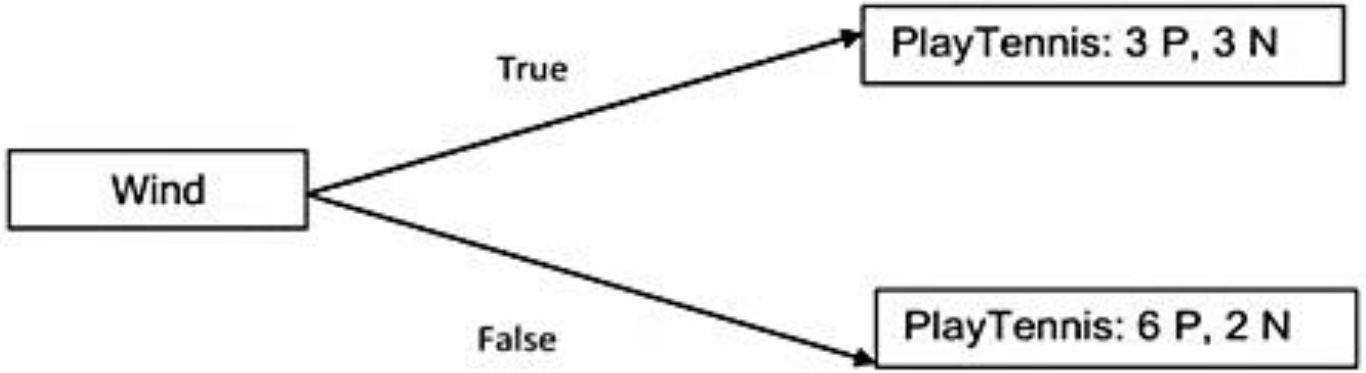
Example - 3

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Gini index calculation:

There are 5 Ns and 9 Ps, so the

- Calculate the information gain after the Wind test is applied:

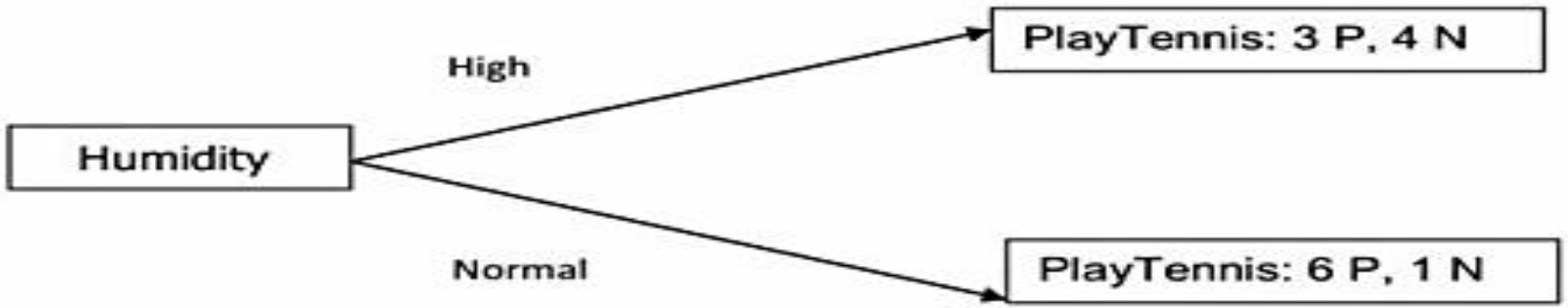


Gini (PlayTennis|Wind=True) = $1- (3/6)^2 – (3/6)^2 = 0.5$
Gini (PlayTennis|Wind=False) = $1- (6/8)^2 – (2/8)^2 = 0.375$

Therefore, the Gini index after the Wind test is applied is

$6/14 \times 0.5 + 8/14 \times 0.375 = 0.4286$ → Reduction in Impurity

- Calculate the information gain after the Humidity test is applied:

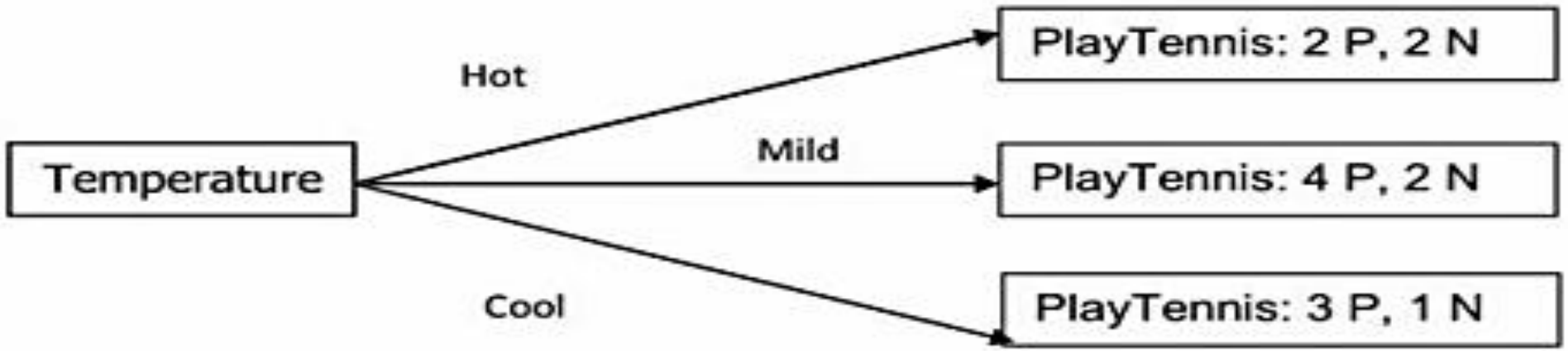


Gini (PlayTennis|Humidity=High) = $1- (3/7)^2 - (4/7)^2 = 0.4898$
Gini (PlayTennis|Humidity=Normal) = $1- (6/7)^2 - (1/7)^2 = 0.2449$

Therefore, the Gini index after the Wind test is applied is

$7/14 \times 0.4898 + 7/14 \times 0.2449 = 0.3674$

- Calculate the information gain after the Temperature test is applied:

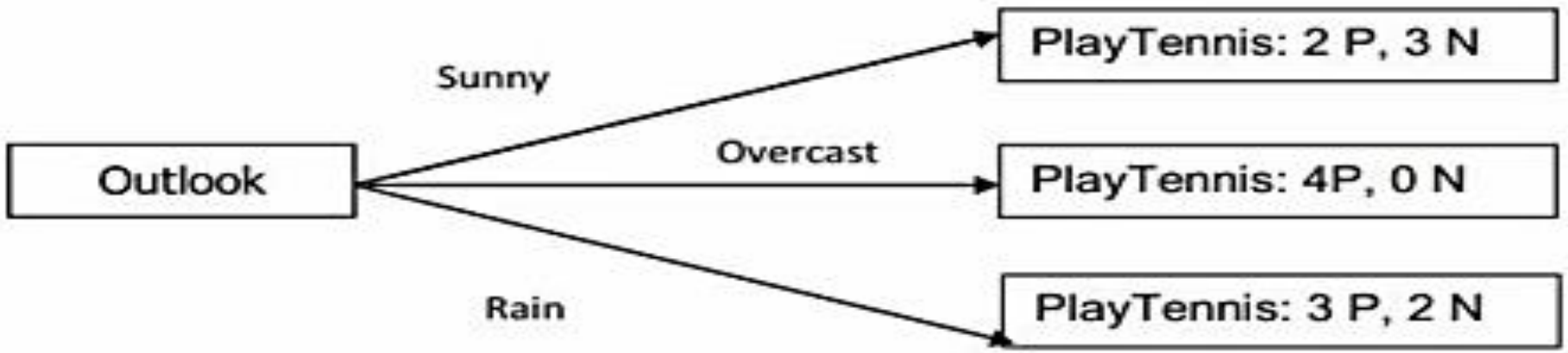


Gini (PlayTennis| Temperature =Hot) = $1- (2/4)^2 - (2/4)^2 = 0.5$
Gini (PlayTennis| Temperature =Mild) = $1- (4/6)^2 - (2/6)^2 = 0.4444$
Gini (PlayTennis| Temperature =Cool) = $1- (3/4)^2 - (1/4)^2 = 0.375$

Therefore, the Gini index after the Temperature test is applied is

$4/14 \times 0.5 + 6/14 \times 0.4444 + 4/14 \times 0.375 = 0.4405$

- Calculate the information gain after the Outlook test is applied:



Gini (PlayTennis| Outlook =Sunny) = $1- (2/5)^2 - (3/5)^2 = 0.48$
Gini (PlayTennis| Outlook =Overcast) = $1- (4/4)^2 - (0/4)^2 = 0$
Gini (PlayTennis| Outlook =Rain) = $1- (3/5)^2 - (2/5)^2 = 0.48$

Therefore, the Gini index after the Temperature test is applied is

$5/14 \times 0.48 + 4/14 \times 0 + 5/14 \times 0.48 = 0.3429$

After calculating all attributes:

- gain(outlook) = 0.3429
- gain(temperature) = 0.4405 → **Will be chosen as a Splitting Attribute.**
- gain(humidity) = 0.3674
- gain(windy) = 0.4286

Same steps will continue as last you defined all the classes.

RANDOM FOREST

Random forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and diversity (it can be used for both classification and regression tasks). In this post we'll learn how the random forest algorithm works, how it differs from other algorithms and how to use it.

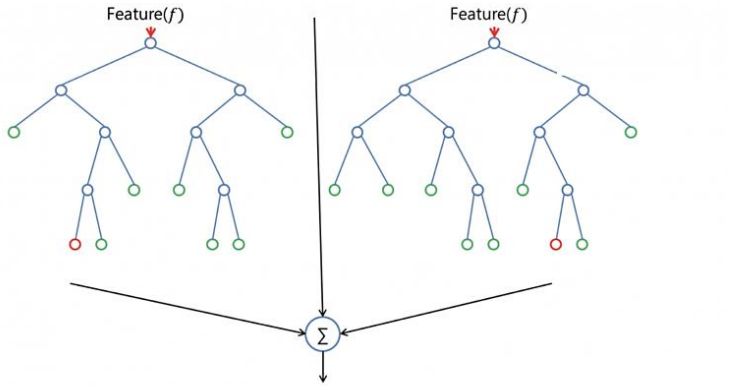
WHAT IS RANDOM FOREST?

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Bootstrap Aggregation

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Put simply: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.



Bootstrap Aggregation (or Bagging for short), is a simple and very powerful ensemble method.

An ensemble method is a technique that combines the predictions from multiple machine learning algorithms together to make more accurate predictions than any individual model