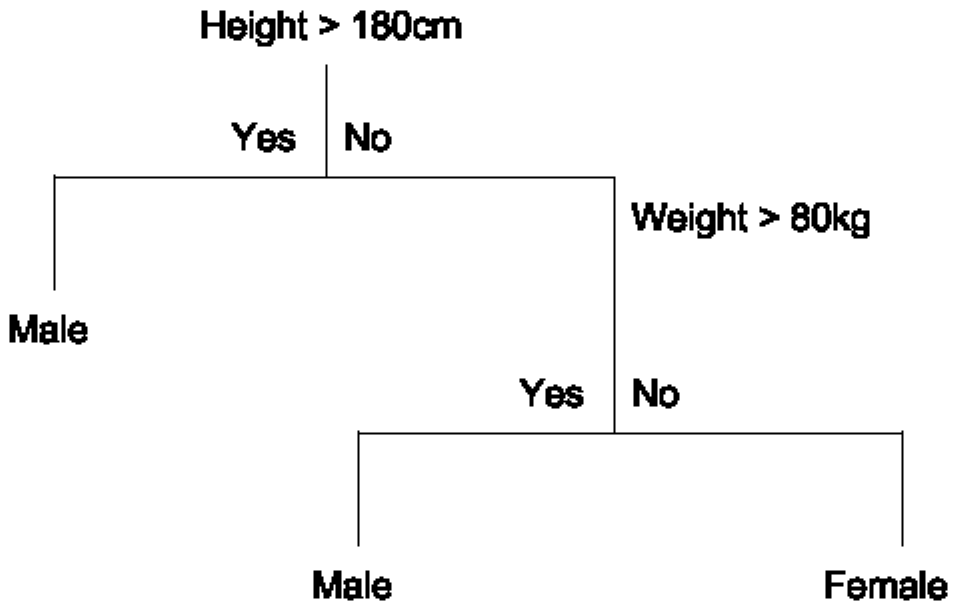
Machine Learning

Decision Tree Algorithm

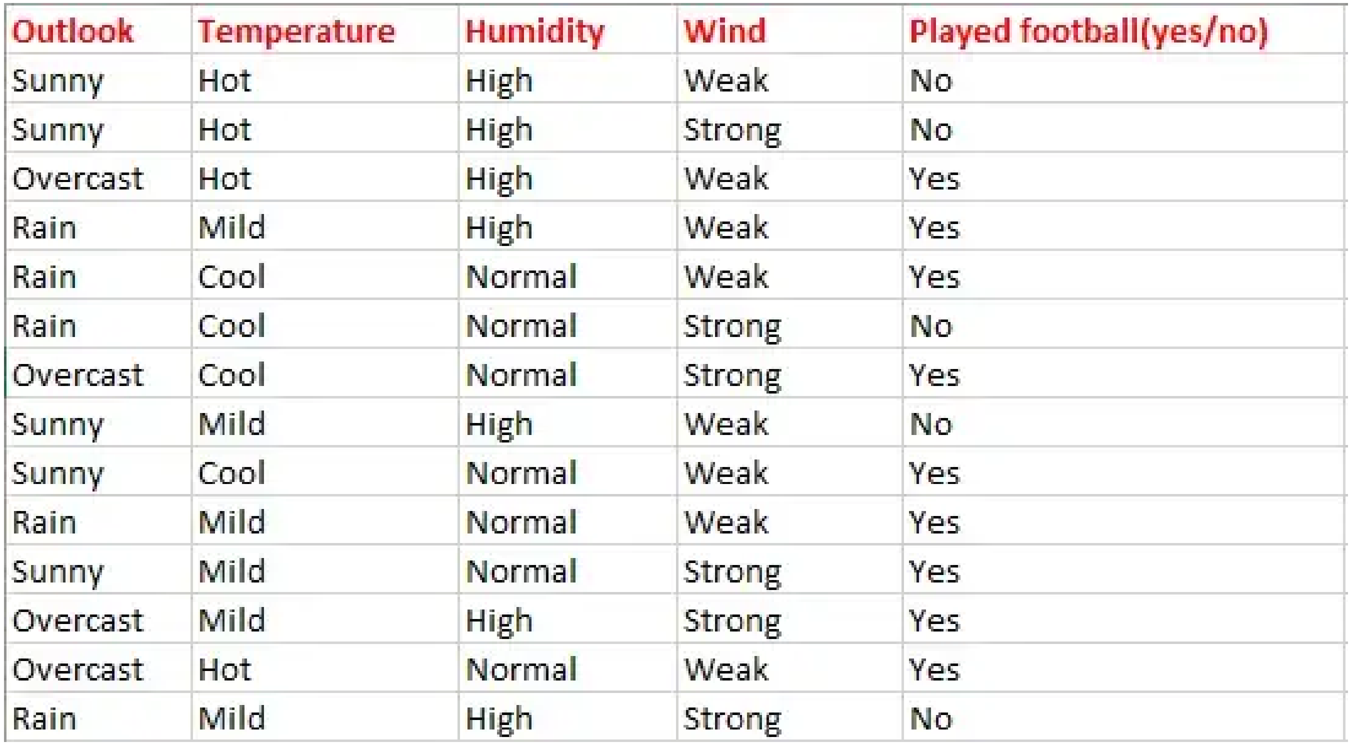
What is a decision tree?

- Let us say, we want to identify male or female from height and weight..



- The decision tree is that it is highly interpretable.

An Example

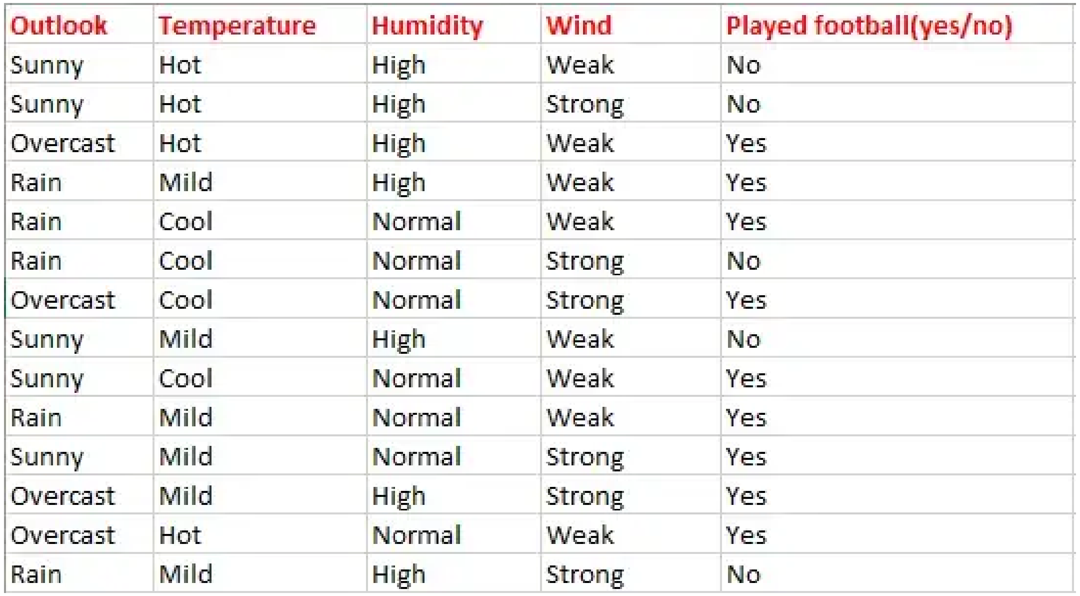


Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild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

Build DECISION TREE CLASSIFIER

Step 1

- ***Find the entropy of the class variable.***



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

Entropy

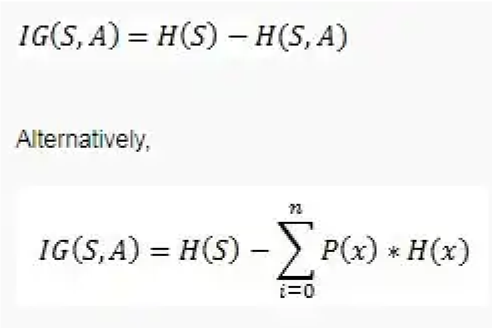
Entropy is a measure of the **randomness** in the information being processed.

The higher the entropy, the harder it is to draw any conclusions from that information.

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i)$$

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

Step 2:
Find the information gain of each attribute

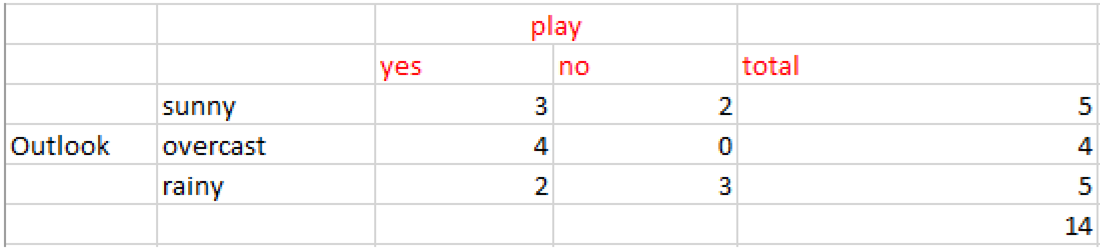


$IG(S,A) = H(S) - H(S,A)$

Alternatively,

$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Take “Outlook” attribute



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

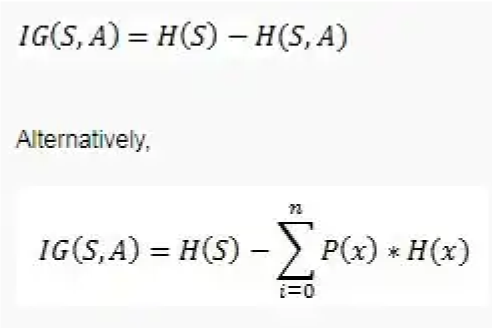
Now we have to calculate average weighted entropy.
ie, we have found the total of weights of each feature multiplied by probabilities.

E(S, 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, outlook) = 0.94 - 0.693 = 0.247

Step 2:
Find the information gain of each attribute



$IG(S,A) = H(S) - H(S,A)$

Alternatively,

$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Take “Temperature” attribute

		Play		
		yes	No	Total
	Hot	2	2	4
Temp	Mild	4	2	6
	cool	3	1	4
				14

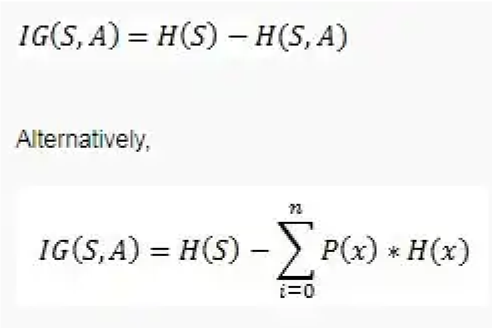
Now we have to calculate average weighted entropy.
ie, we have found the total of weights of each feature multiplied by probabilities.

E(S, Temp) = (4/14)*E(2,2) + (6/14)*E(4,2) + (4/14)*E(3,1)
= (4/14)(-(2/4)log(2/4)-(2/4)log(2/4))+ (6/14)(-(4/6)log(4/6)-(2/6)log(2/6))) +
(4/14)(-(3/4)log(3/4)-(1/4)log(1/4)) = 0.911

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, temperature) = 0.94 - 0.911 = 0.029

Step 2:
Find the information gain of each attribute



$$IG(S,A) = H(S) - H(S,A)$$

Alternatively,

$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Take “Humidity” attribute

		Play		
		yes	No	Total
	high	3	4	7
humidity	normal	6	1	7
				14

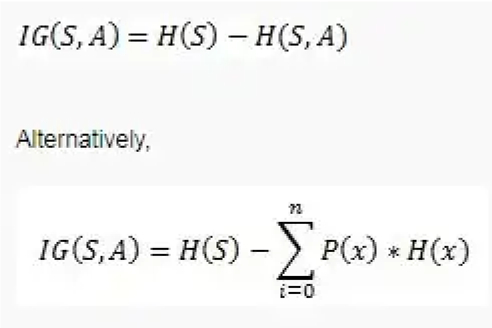
Now we have to calculate average weighted entropy.
ie, we have found the total of weights of each feature multiplied by probabilities.

E(S, Humidity) = (7/14)*E(3,4) + (7/14)*E(6,1)
= (7/14)(-(3/7)log(3/7)-(4/7)log(4/7))+ (7/14)(-(6/7)log(6/7)-(1/7)log(1/7))) = 0.788

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, humidity) = 0.94 - 0.788 = 0.152

Step 2:
Find the information gain of each attribute



$IG(S, A) = H(S) - H(S, A)$

Alternatively,

$$IG(S, A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Take “Wind” attribute

		Play		
		yes	No	Total
	Strong	3	3	6
wind	week	6	2	8
				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, wind= (6/14)*E(3,3) + (8/14)*E(6,2)

= (6/14)(-(3/6)log(3/6)-(3/6)log(3/6))+ (8/14)(-(6/8)log(6/8)-(2/8)log(2/8))) = 0.8932

***The next step is to find the information gain*.**

It is the difference between parent entropy and average weighted entropy we found above.

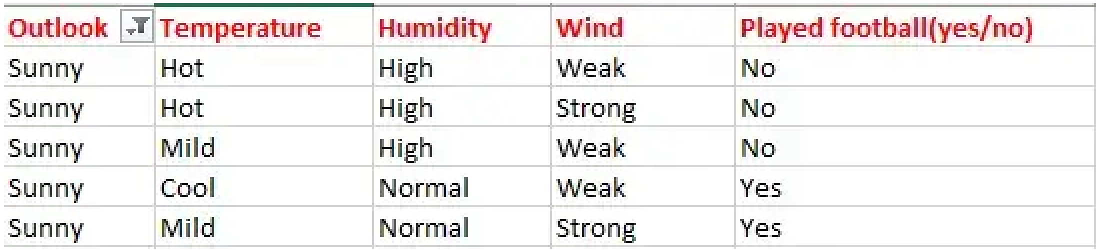
IG(S, wind) = 0.94 - 0.8932 = 0.048

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

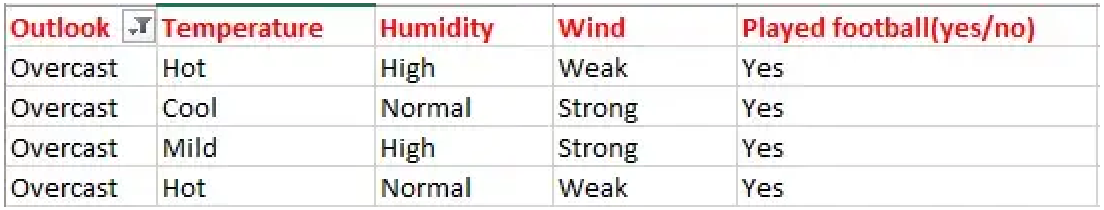
IG(S, outlook) = 0.94 - 0.693 = 0.247
IG(S, Temperature) = 0.940 - 0.911 = 0.029
IG(S, Humidity) = 0.940 - 0.788 = 0.152
IG(S, Windy) = 0.940 - 0.8932 = 0.048

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.

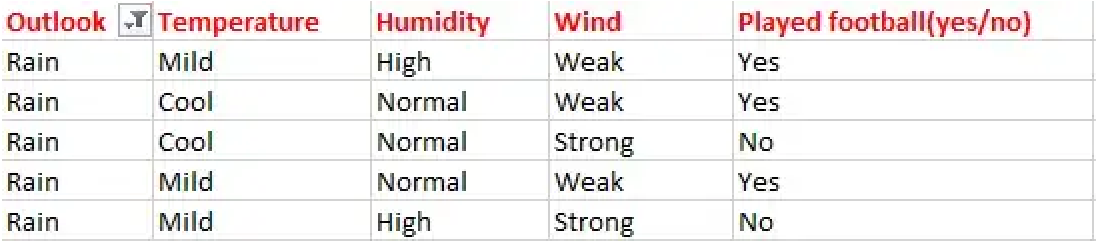
Now our data look as follows



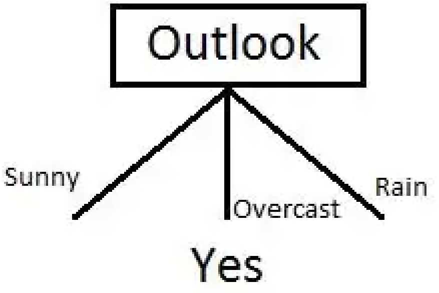
Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes



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

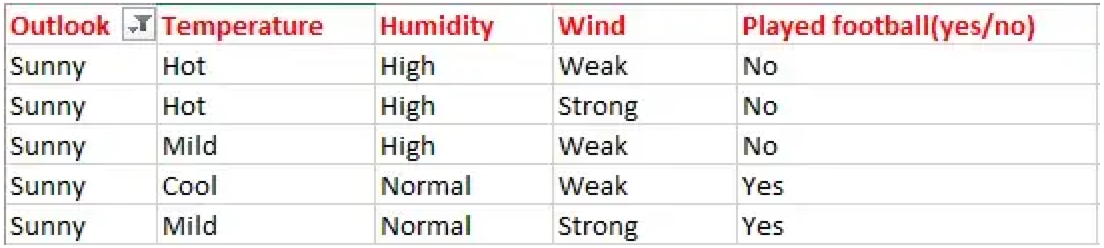


Since overcast contains only examples of class ‘Yes’ we can set it as yes. That means If outlook is overcast football will be played. Now our decision tree looks as follows.

Step 3

- 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.

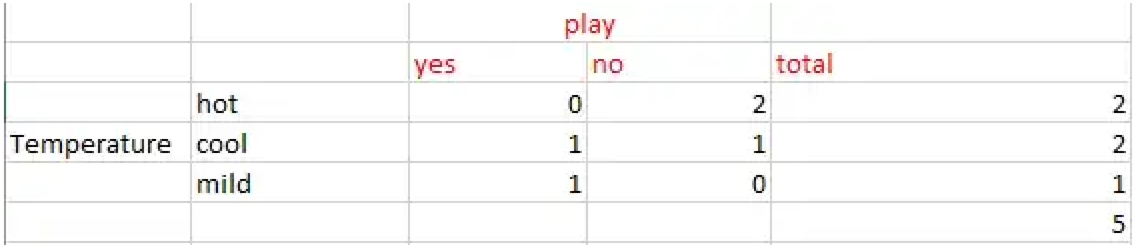


Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes

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)

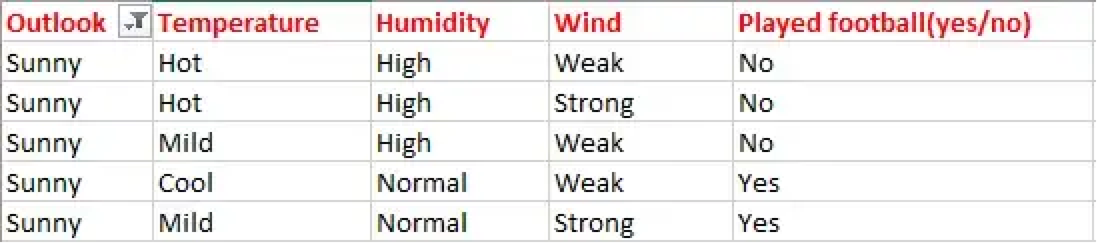


		play		
		yes	no	total
	hot	0	2	2
Temperature	cool	1	1	2
	mild	1	0	1
				5

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



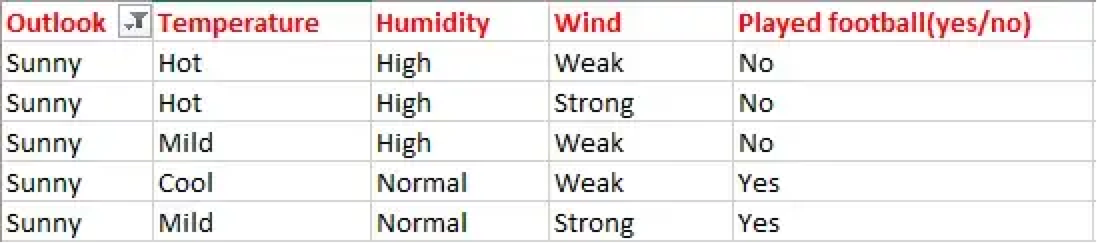
Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes

Now Humidity Attribute

		Play		
		yes	No	Total
	high	0	3	3
humidity	normal	2	0	2
				5

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

E(sunny, Humidity)
= (3/5)*E(0,3) + (2/5)*E(2,0) =0
Now calculate information gain.
IG(sunny, humidity)
= 0.971–0.0=0.971



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

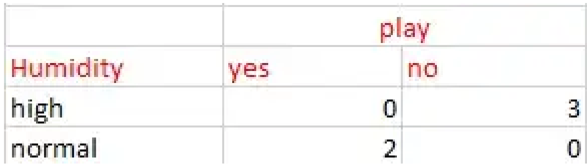
Now wind Attribute

		Play		
		yes	No	Total
	Weak	1	2	3
wind	Strong	1	1	2
				5

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

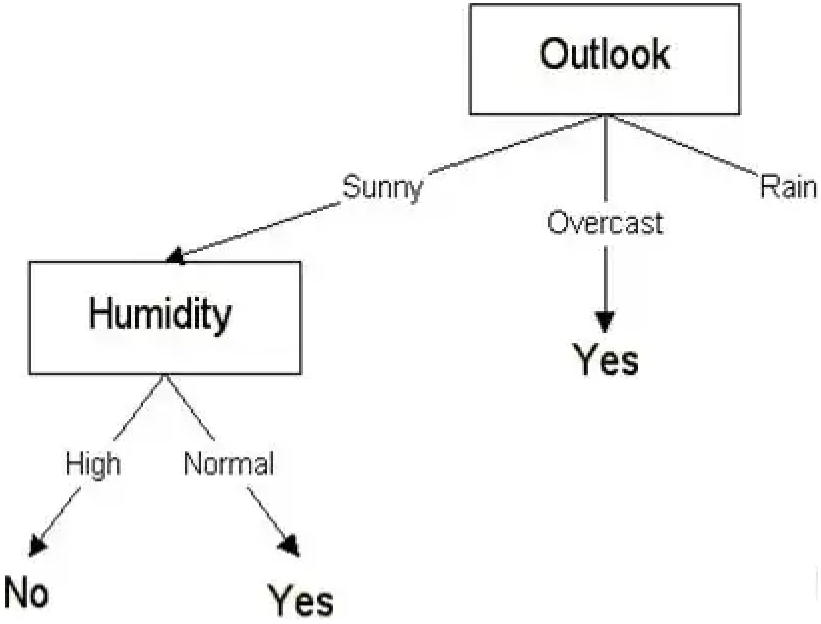
E(sunny, wind)
= (3/5)*E(1,2) + (2/5)*E(1,1) =0.95098
Now calculate information gain.
IG(sunny, wind)
= 0.971–0.95098=0.020

We get
IG(sunny, Temperature) = 0.571
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.

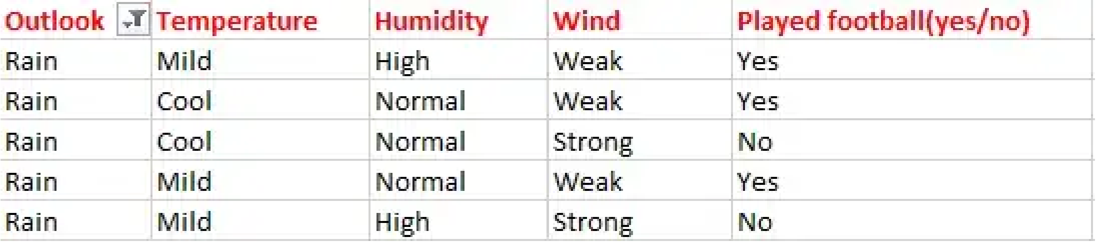


	play	
Humidity	yes	no
high	0	3
normal	2	0

Finally, our decision tree will look as below:



Now we will find one under rain. We have to determine which of the following Temp, Humidity or Wind has higher information gain.



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

Calculate parent entropy E(rain)

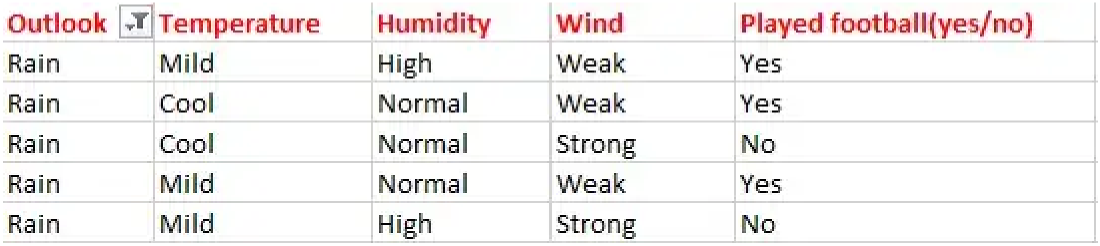
E(rain) = (-(3/5)log(3/5)-(2/5)log(2/5)) = 0.971.

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

		Play		
		yes	No	Total
Temp	Mild	2	1	3
	Cool	1	1	2
				5

E(rain, Temperature)
= (3/5)*E(2,1) + (2/5)*E(1,1)
= 0.95098
Now calculate information gain.
IG(rain, Temperature)
= 0.971–0.95098 =0.020

Now consider Humidity



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

Calculate parent entropy E(rain)

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

Now Calculate the information gain of Humidity. IG(rain, humidity)

		Play		
		yes	No	Total
humidity	High	1	1	2
	Normal	2	1	3
				5

E(rain, Humidity)
= (3/5)*E(2,1) + (2/5)*E(1,1)
= 0.95098
Now calculate information gain.
IG(rain, Humidity)
= 0.971–0.95098 =0.020

Now consider wind

		Play		
		yes	No	Total
wind	Weak	3	0	3
	strong	0	2	2
				5

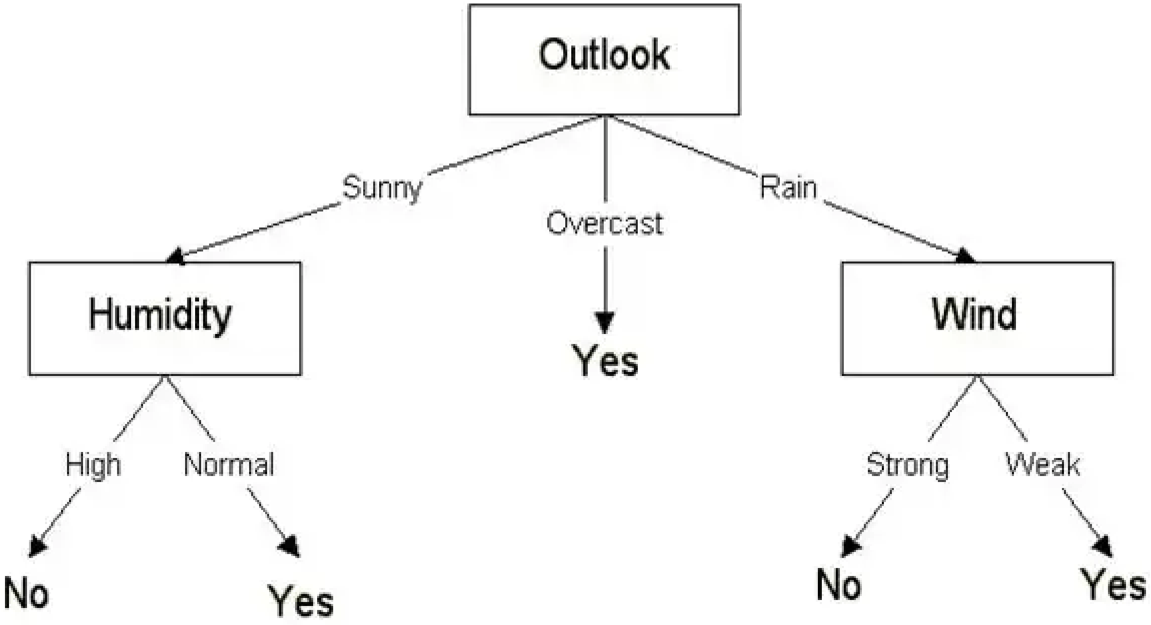
Now Calculate the information gain of wind. IG(rain, wind)

E(rain, wind)
= (3/5)*E(3,0) + (2/5)*E(0,2)
= 0.0
Now calculate information gain.
IG(rain, wind)
= 0.971–0.0 =0.971

IG(rain, Temperature) =0.020
IG(rain, Humidity) =0.020
IG(rain, wind) =0.971

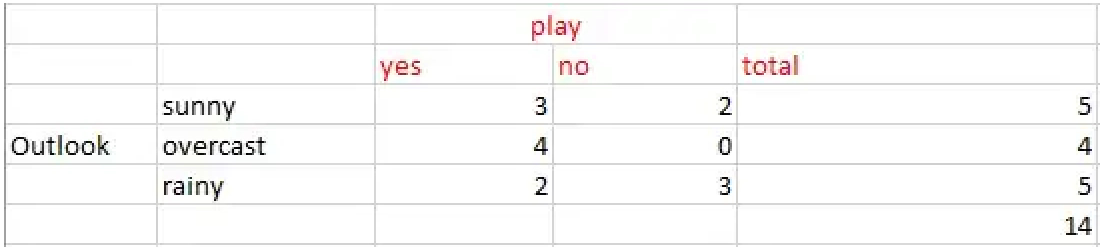
Finally, our decision tree will look as below:

		Play		
		yes	No	Total
wind	Weak	3	0	3
	strong	0	2	2
				5

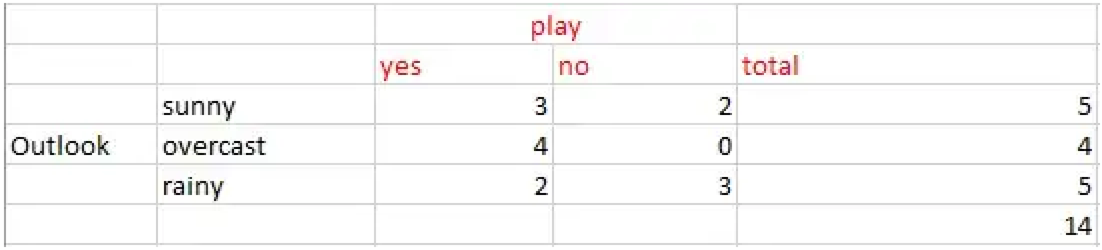


Classification using CART algorithm

- **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** $Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$
- Gini(S) = 1 - [(9/14)^2 + (5/14)^2] = 0.4591
- **As the next step, we will calculate the Gini gain.**



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



		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.

Decision Tree Algorithm

- **Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.**
- It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

Why use Decision Trees?

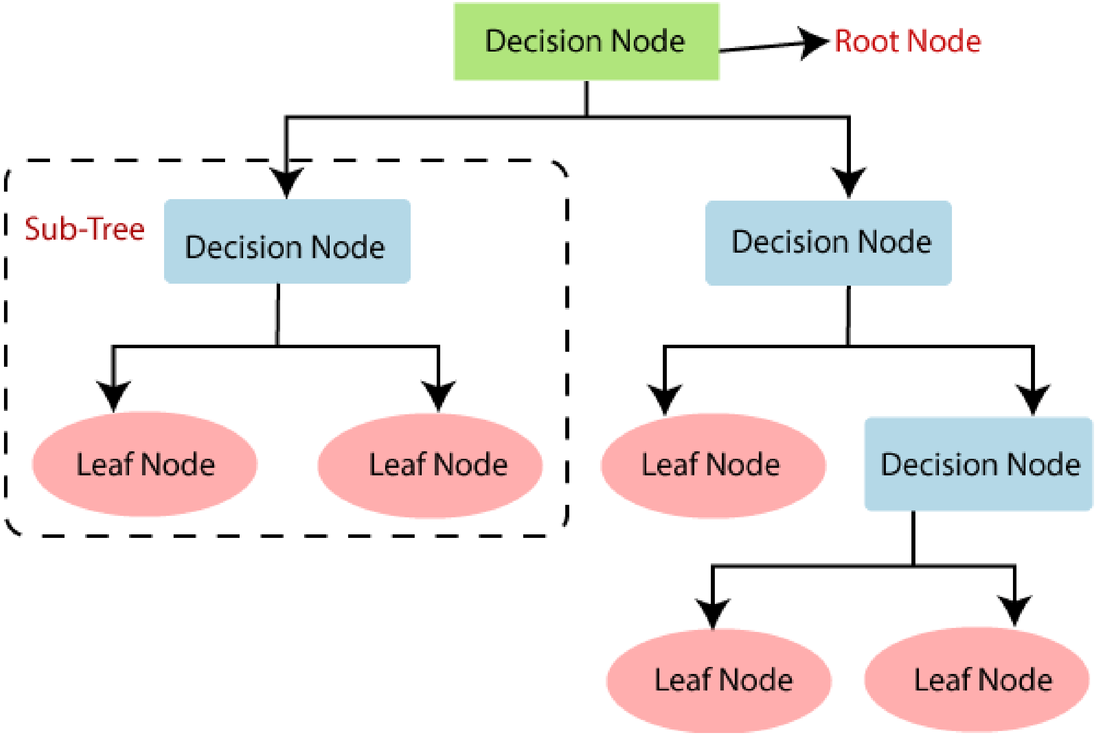
- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- Easy to understand as based on if/else conditions

Assumptions

- At the beginning, we consider the whole training set as the root.
- Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model. On the basis of attribute values, records are distributed recursively.
- We use statistical methods for ordering attributes as root or the internal node.

Decision Tree Terminologies

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are call



How does the Decision Tree algorithm Work?

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Attribute Selection Measures

- By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:
- **Information Gain**
- **Gini Index**

There are many algorithms there to build a decision tree.
They are **CART** (Classification and Regression Trees) — This makes use of Gini impurity as the metric.
ID3 (Iterative Dichotomiser 3) — This uses entropy and information gain as metric.

Information Gain

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

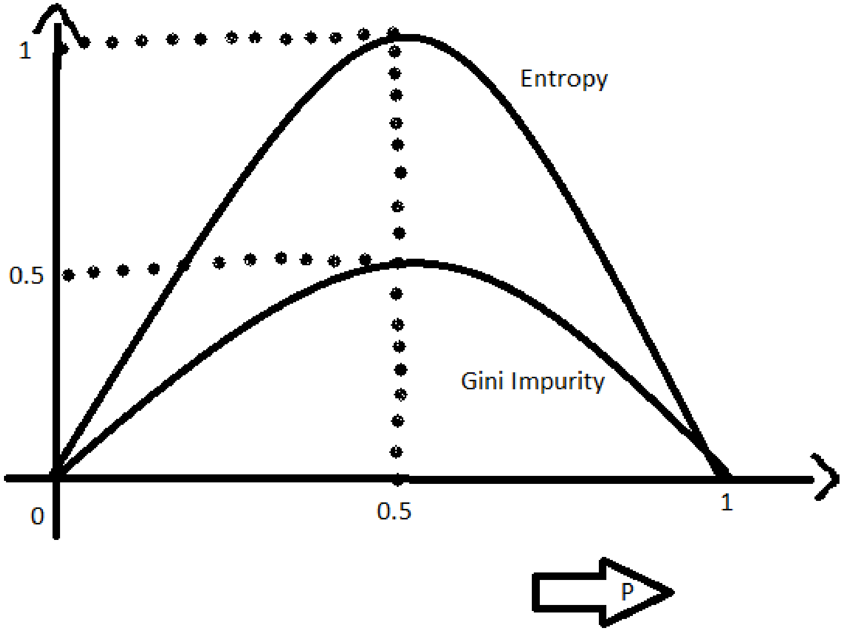
Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)]

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data.
Entropy can be calculated as:
Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Gini Index

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- Gini index can be calculated using the below formula:

$$\text{Gini Index}= 1- \sum_j P_j^2$$



$$E(S)=\sum_{i=1}^{c}-p_i\log_2 p_i$$

$$Gini(E)=1-\sum_{j=1}^{c}p_j^2$$

Pruning: Getting an Optimal Decision tree

- *Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.*

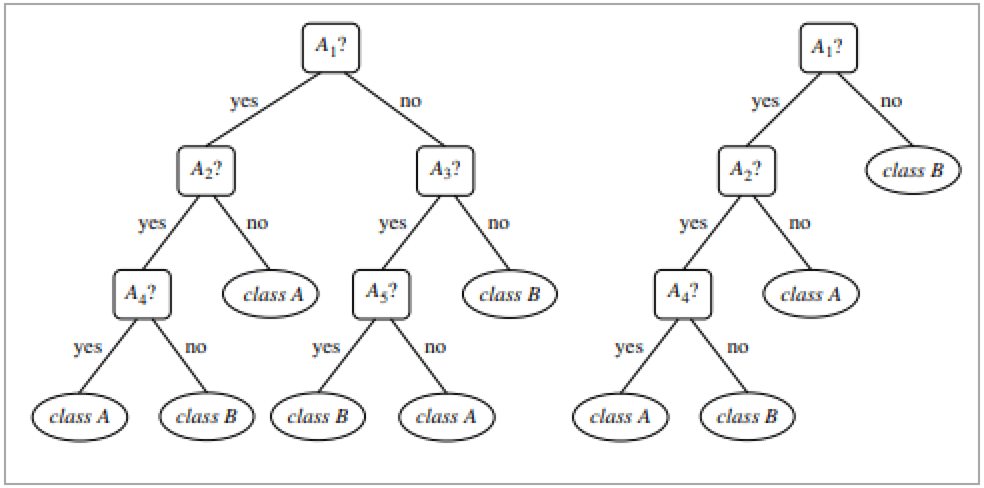


Image shows an unpruned and pruned tree.

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**

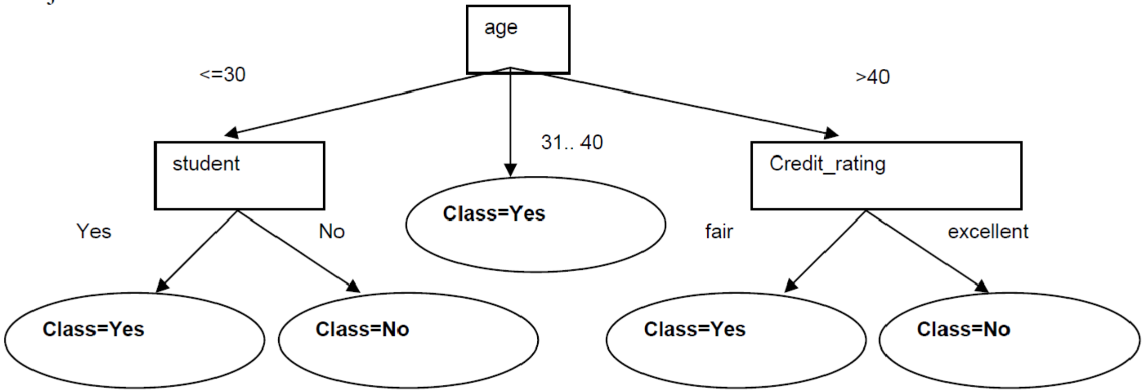
For more class labels, the computational complexity of the decision tree may increase.

Exercise

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Exercise

We then split based on credit_rating. These splits give partitions each with records from the same class. We just need to make these into leaf nodes with their class label attached:



New example: age<=30, income=medium, student=yes, credit-rating=fair
Follow branch(age<=30) then student=yes we predict Class=yes ➔ Buys_computer = yes

Applications

- **In healthcare industries**

  In healthcare industries, decision tree can tell whether a patient is suffering from a disease or not based on conditions such as age, weight, gender and other factors.

  Other applications such as deciding the effect of the medicine based on factors such as composition, period of manufacture, etc.

  Also, in diagnosis of medical reports, a decision tree can be very effective.

- **In banking sectors.**

  A person eligible for a loan or not based on his financial status, family member, salary, etc. can be decided on a decision tree.

  Other applications may include credit card frauds, bank schemes and offers, loan defaults, etc. which can be prevented by using a proper decision tree.

- **In educational Sectors**

  In colleges and universities, the shortlisting of a student can be decided based upon his merit scores, attendance, overall score etc.

  A decision tree can also decide the overall promotional strategy of faculties present in the universities.

  There are many other applications too where a decision tree can be a problem-solving strategy despite its certain drawbacks.

Challenges faced in Decision Tree

- **Overfitting**
  - **Pruning**
  - **Ensemble method or bagging and boosting**
- **Discretization**

  When the data contains too many numerical values, discretization is required as the algorithm fails to make a decision on such small and rapidly changing values. Such a process can be time consuming and produce inaccurate results when it comes in training the data.

Random Forest Algorithm

- ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."***

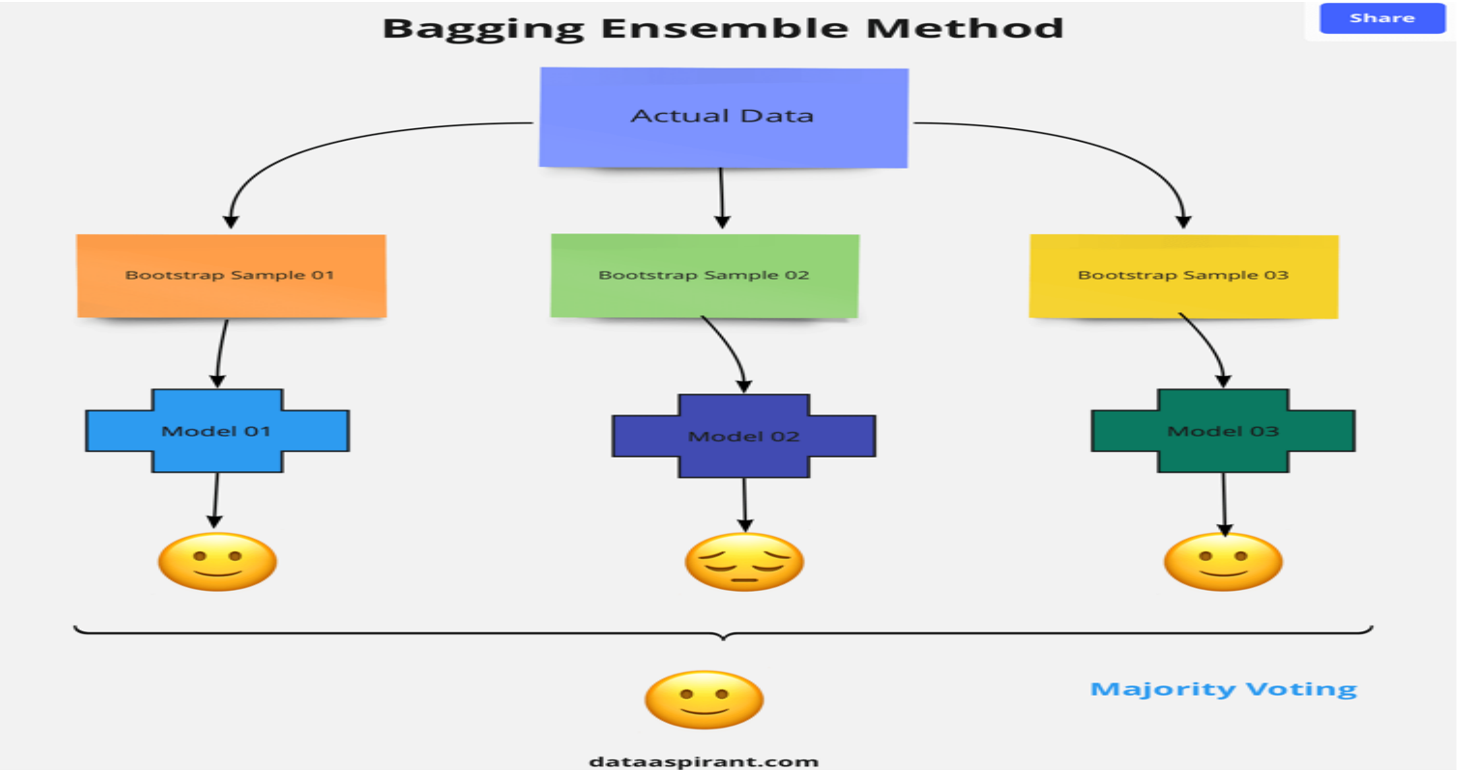
Working of Random Forest Algorithm

- ***Ensemble*** simply means combining multiple models. Thus a collection of models is used to make predictions rather than an individual model.
- 1. **Bagging**– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest.
- 2. **Boosting**– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST

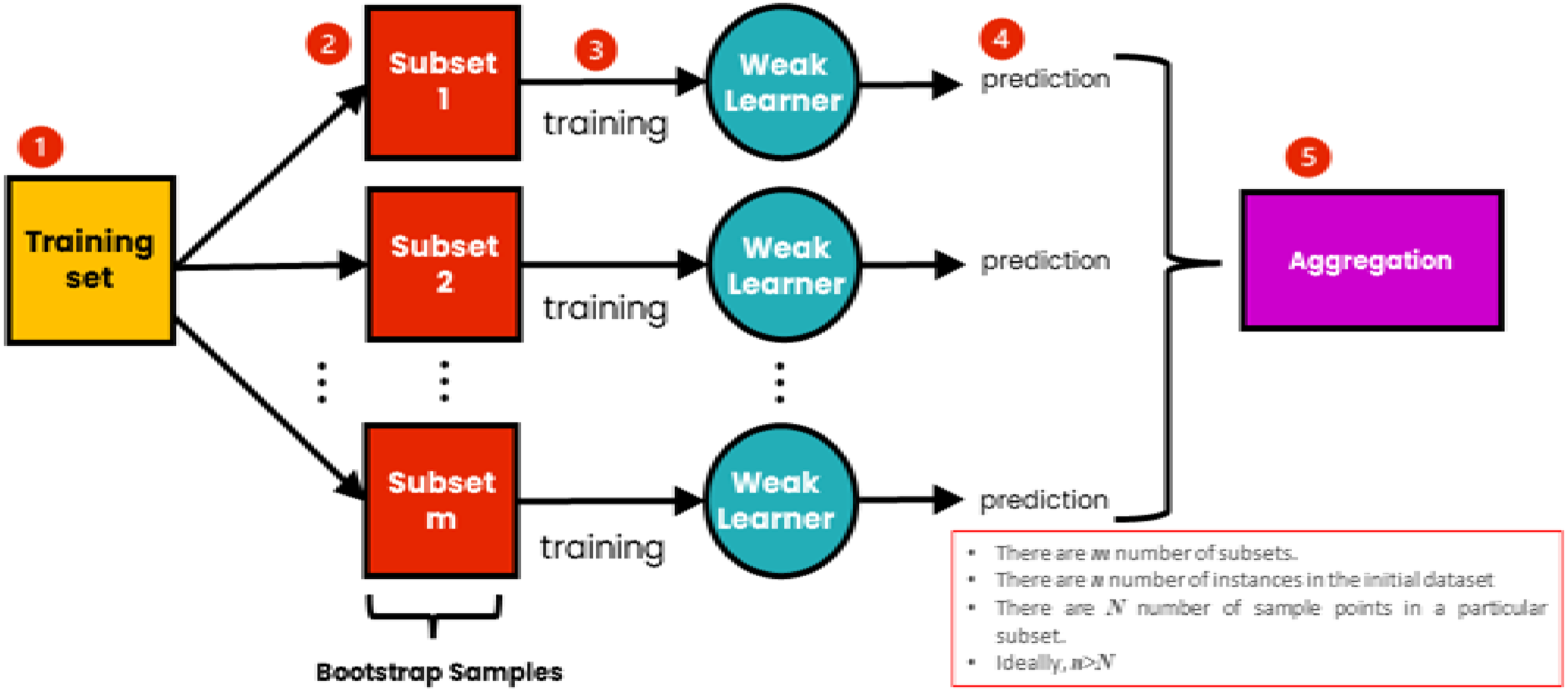
Bagging

Bagging, also known as ***Bootstrap Aggregation*** is the ensemble technique used by random forest.

Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as ***row sampling***. This step of row sampling with replacement is called ***bootstrap***.

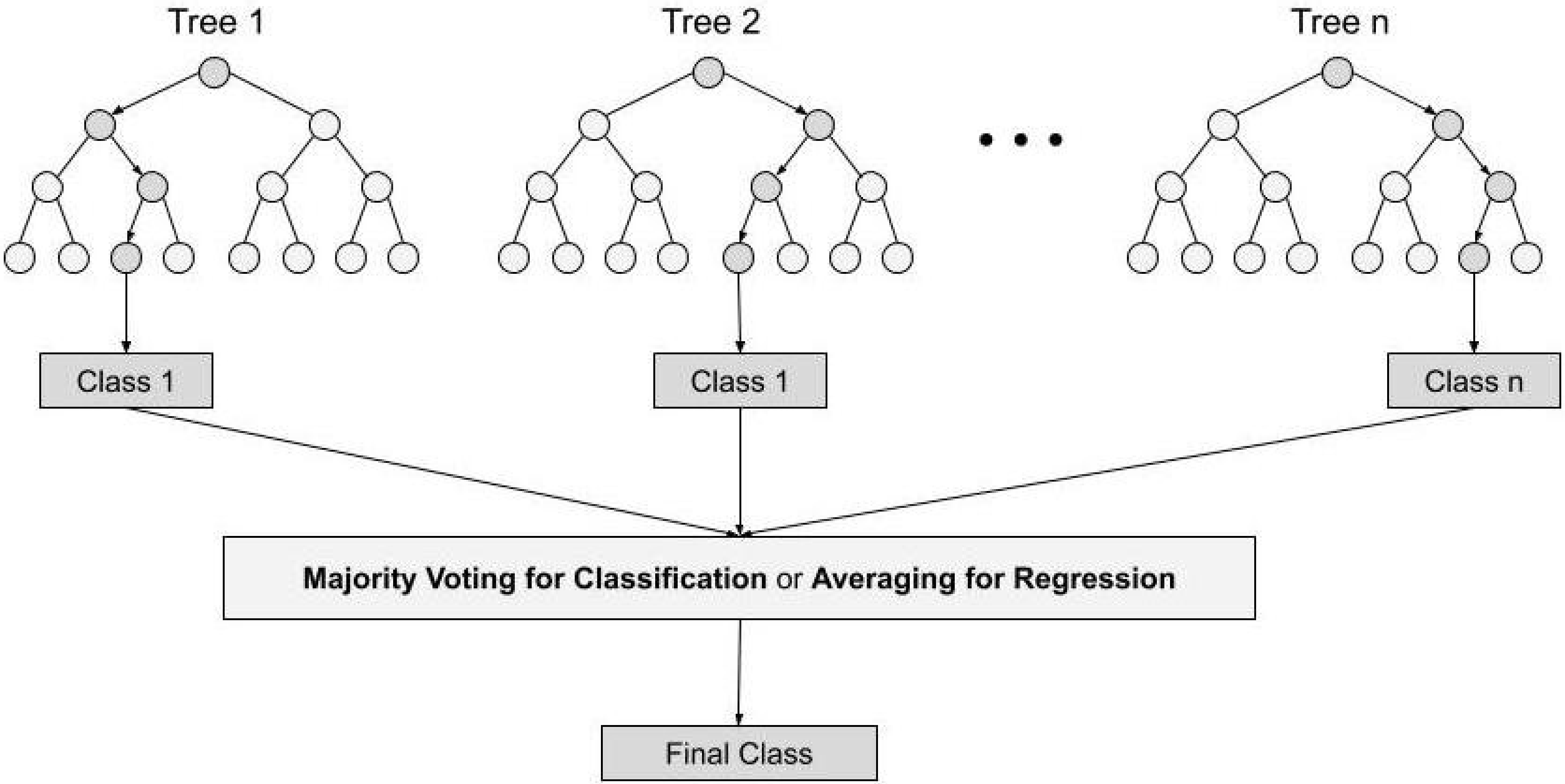


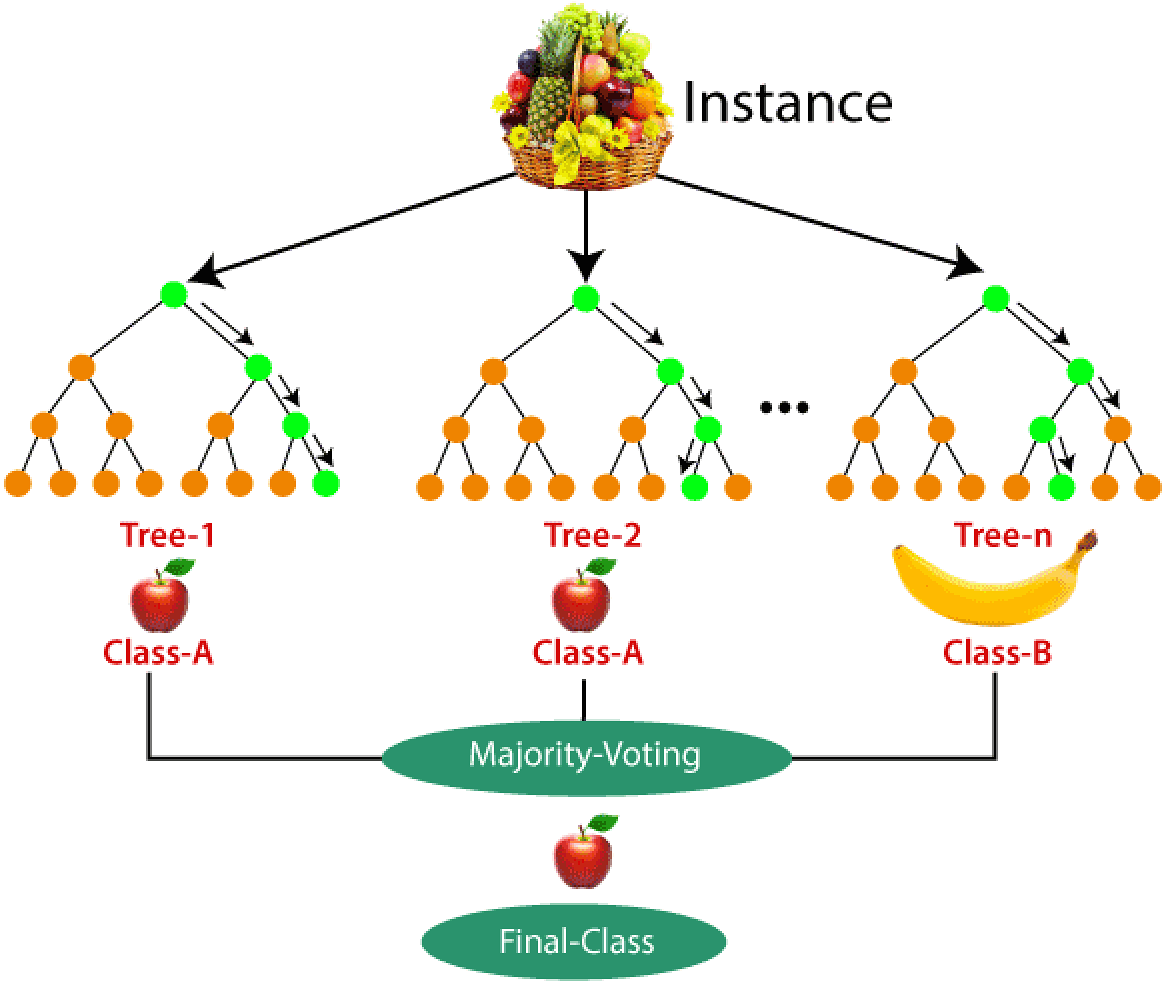
The Process of Bagging (Bootstrap Aggregation)



- There are m number of subsets.
- There are n number of instances in the initial dataset
- There are N number of sample points in a particular subset.
- Ideally, n>N

Random forest algorithm:


Tree 1
Tree 2
Tree n
Class 1
Class 1
Class n
Majority Voting for Classification or Averaging for Regression
Final Class


Instance
Tree-1
Tree-2
Tree-n
Class-A
Class-A
Class-B
Majority-Voting
Final-Class

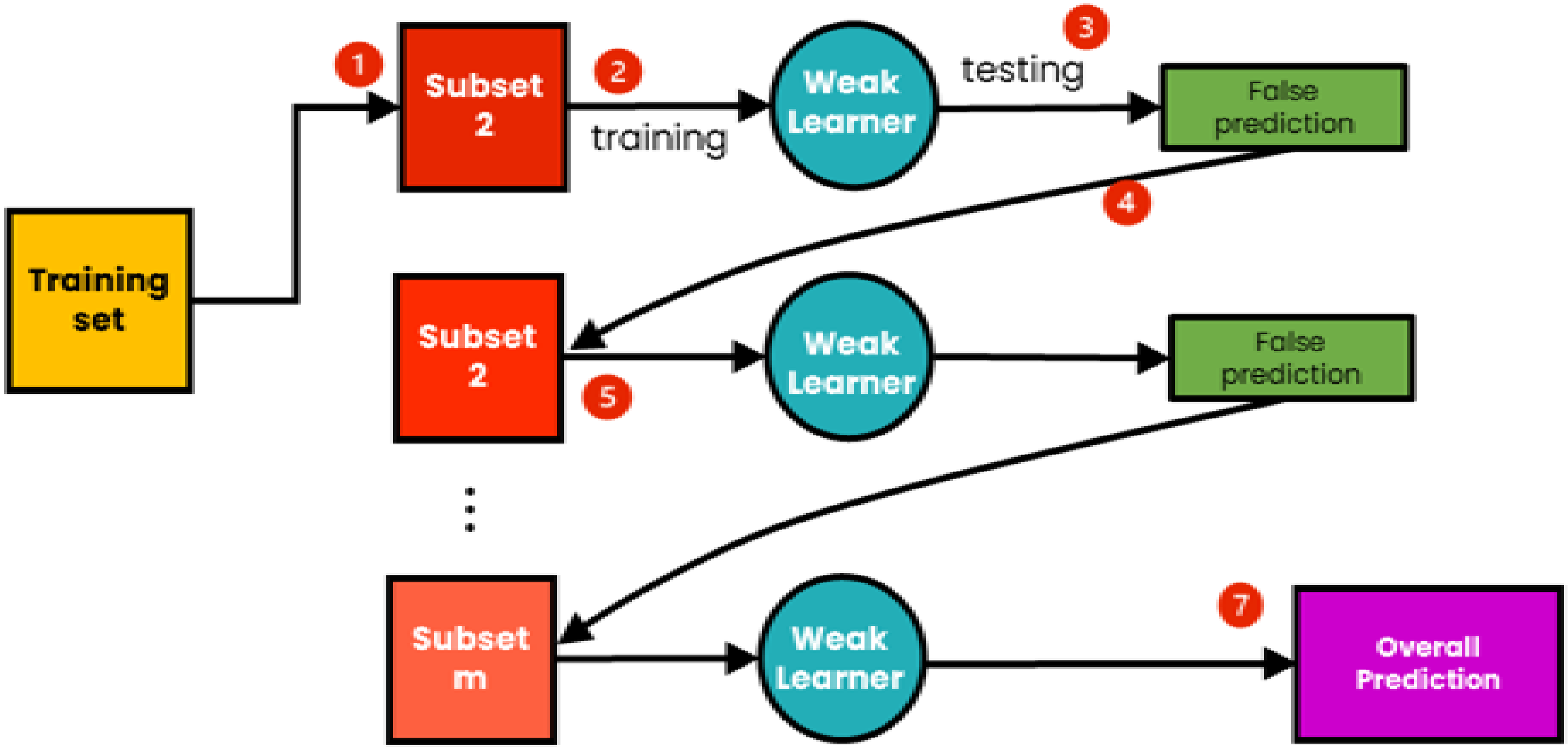
Difference Between Decision Tree & Random Forest

Decision trees	Random Forest
1. Decision trees normally suffer from the problem of overfitting if it’s allowed to grow without any control.	1. Random forests are created from subsets of data and the final output is based on average or majority ranking and hence the problem of overfitting is taken care of.
2. A single decision tree is faster in computation.	2. It is comparatively slower.
3. When a data set with features is taken as input by a decision tree it will formulate some set of rules to do prediction.	3. Random forest randomly selects observations, builds a decision tree and the average result is taken. It doesn’t use any set of formulas.

Boosting

- Boosting is an efficient algorithm that converts a weak learner into a strong learner.
- **Adaptive boosting or AdaBoost:** This method operates iteratively, identifying misclassified data points and adjusting their weights to minimize the training error. The model continues to optimize sequentially until it yields the strongest predictor.
- **Gradient Boosting:** Gradient Boosting is also based on sequential ensemble learning. Here the base learners are generated sequentially so that the present base learner is always more effective than the previous one, i.e., and the overall model improves sequentially with each iteration.
- The difference in this boosting type is that the weights for misclassified outcomes are not incremented. Instead, the Gradient Boosting method tries to optimize the loss function of the previous learner by adding a new model that adds weak learners to reduce the loss function.
- **Extreme gradient boosting or XGBoost:** The main aim of this algorithm is to increase the speed and efficiency of computation. The Gradient Descent Boosting algorithm computes the output slower since they sequentially analyze the data set. Therefore XGBoost is used to boost or extremely boost the model's performance.

The Process of Boosting


Training set
1
Subset 2
2
training
Weak Learner
3
testing
False prediction
4
Subset 2
5
Weak Learner
False prediction
Subset m
Weak Learner
7
Overall Prediction