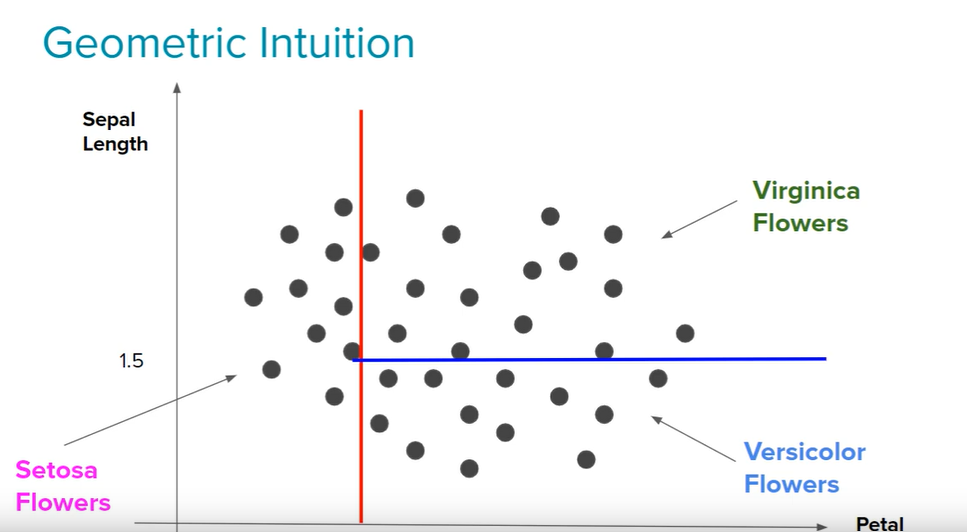
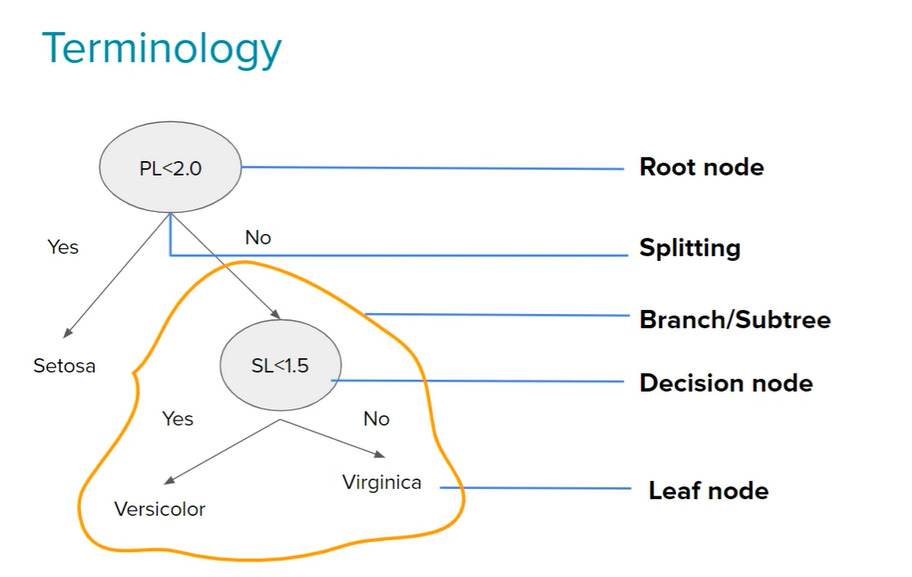
# Decision Tree

Decision tree is a supervised machine learning algorithm used for both classification and regression tasks. It is a tree like model where each node represents a decision or a test on a feature, each branch represents an outcome of that test, and each leaf node represents the final output or a decision. Decision trees are easy to understand, interpret and visualize, making them a popular choice for various applications.

* Decision trees are nothing but a giant structure of nested if-else condition
* Mathematically speaking, Decision trees use hyperplanes which run parallel to any one of the axes to cut your coordinate system into hyper cuboids
* 

Key components of Decision tree:

1. Root node: The topmost node in the tree, represent the initial decision or test based on a feature
2. Decision nodes (Internal nodes): Nodes that follow the root node and represent decision or test based on specific features.
3. Branches: Branches emanating (spread out) from decision nodes, representing the possible outcomes or decision based on feature test.
4. Leaf nodes: Terminal nodes at the end of each branch, representing the final output or decision.



## Decision tree Process

1. Tree Construction: The algorithm recursively splits the dataset based on features to create nodes until a stopping criterion is met
2. Node splitting: At each internal node, the algorithm chooses the feature and split point that maximises information gain (for classification) or minimizes impurity (for regression).
3. Stopping Criteria: Tree construction continues until a predefined stopping criterion is met, such as reaching a maximum depth, having a minimum number of samples in a node or achieving perfect purity.

Types of Decision trees:

1. Classification Trees: Used for predicting the class labels of samples.
2. Regression Trees: Used for predicting numerical values.

## Advantages of Decision Trees:

* Easy to understand.
* Require little data preprocessing.
* Nonlinear relationships between features can be captured.
* Handle both numerical and categorical data.

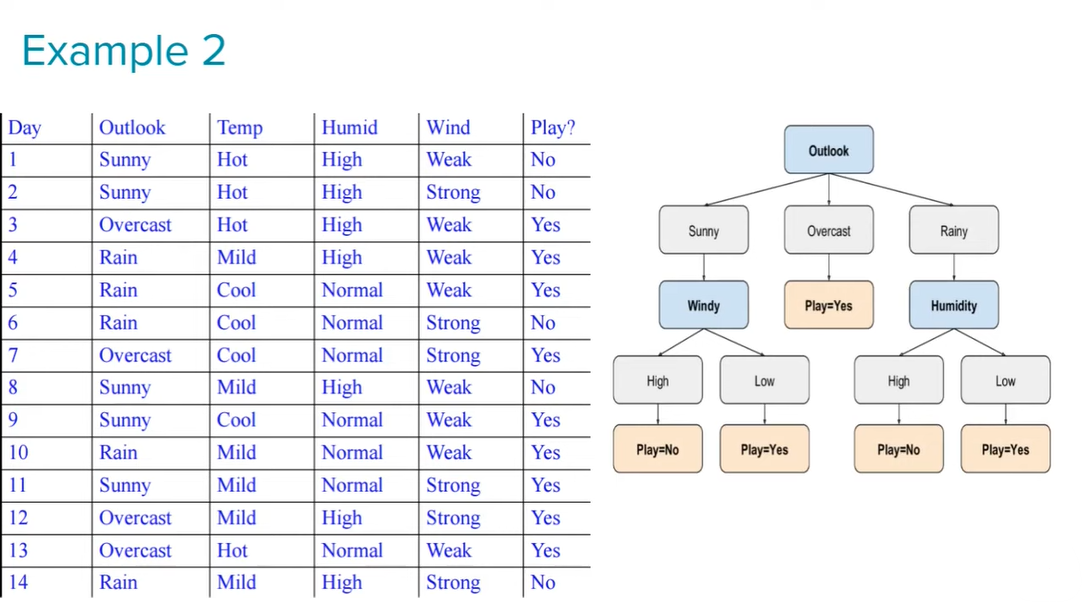
## Disadvantages and considerations:

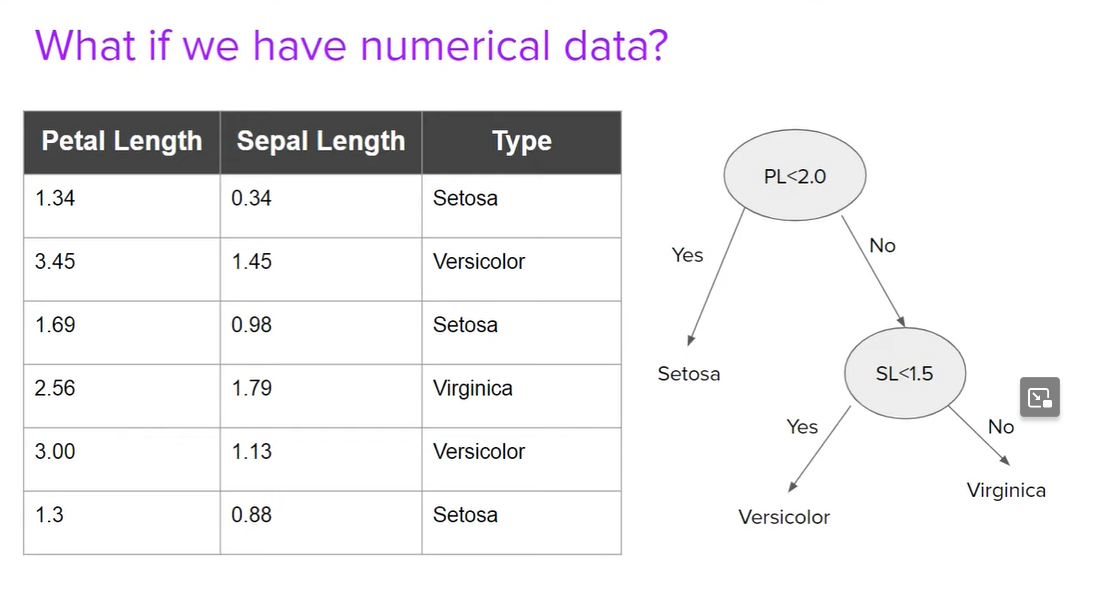
* Prone to overfitting, especially if the tree is deep.
* Sensitive to noisy data.
* May create biased trees if some classes dominate.

## Pseudo code:

* Begin with your training dataset, which should have some feature variables and classification or regression output
* Determine the ‘best feature’ in the dataset to split the data on.
* Split the data into subsets that contain the correct values for this best feature. This splitting basically defines a node on the tree i.e. each node is a splitting point based on a certain feature from our data.
* Recursively generate new tree nodes by using the subset of data created from step 3.

## Geometric intuition:





## What is Entropy

In the most layman term, Entropy is nothing but the measure of amount of disorder/uncertainty. Or you can also call it the measure of purity/impurity.

Entropy Formula:

For a binary classification problem, where there are two classes (e.g. 0 and 1) the entropy formula is given by:

Where:

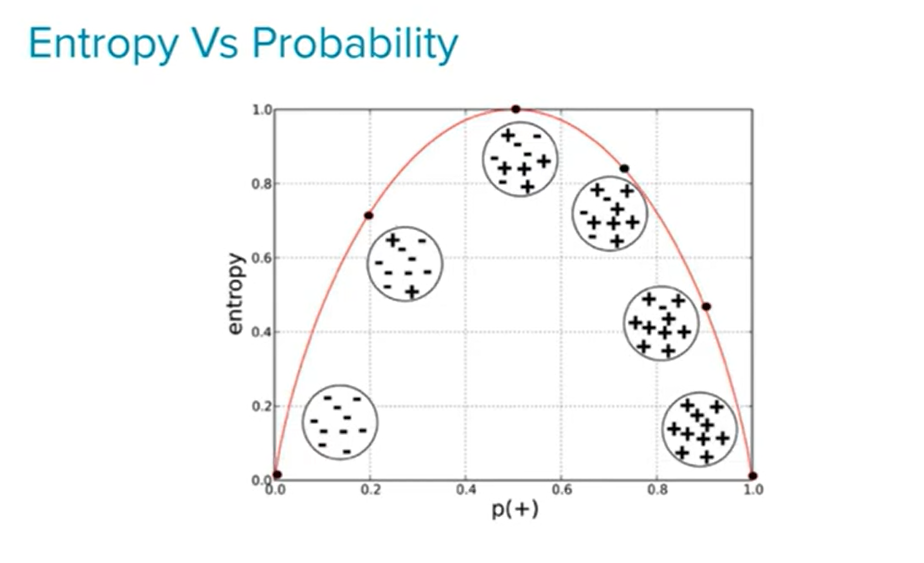
* S is the set of data or a node in the decision tree.
* p1​ and p2​ are the probabilities of belonging to class 1 and class 2, respectively.
* ​ is the base-2 logarithm.

### Interpreting Entropy:

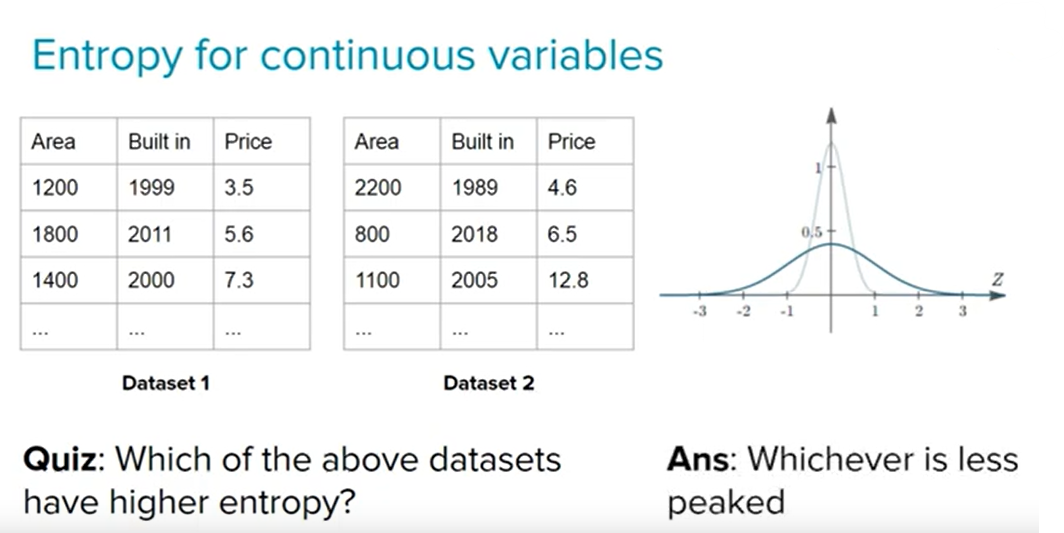
* **Low Entropy (Pure node)**: If a node has low entropy, it means that the data in that node is very pure. For example, if all instances belong to class 0, the entropy is 0. Similarly, if all instances belong to class 1, the entropy is also 0.
* **High Entropy (Impure node):** If anode has high entropy, it means that the data is impure, with instances from multiple classes. A node is split to reduce entropy and increase purity.

Observations:

* More the uncertainty more is entropy.
* For a 2-class problem the minimum entropy is 0 and the maximum is 1.
* For more than 2 classes the minimum entropy is 0 but the maximum can be greater than 1.
* Both ​ or ​ can be used to calculate the entropy.



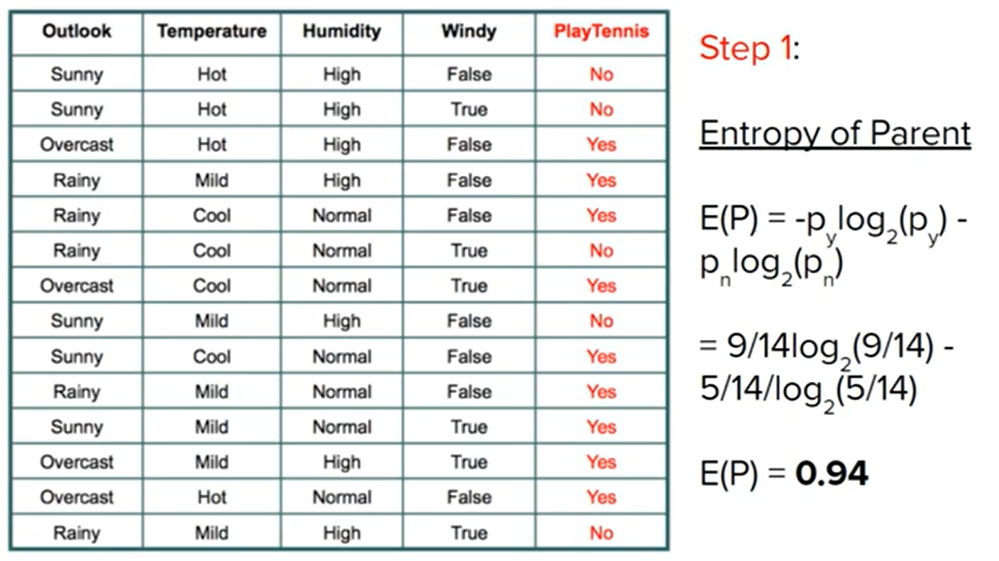
Entropy for continuous problem:

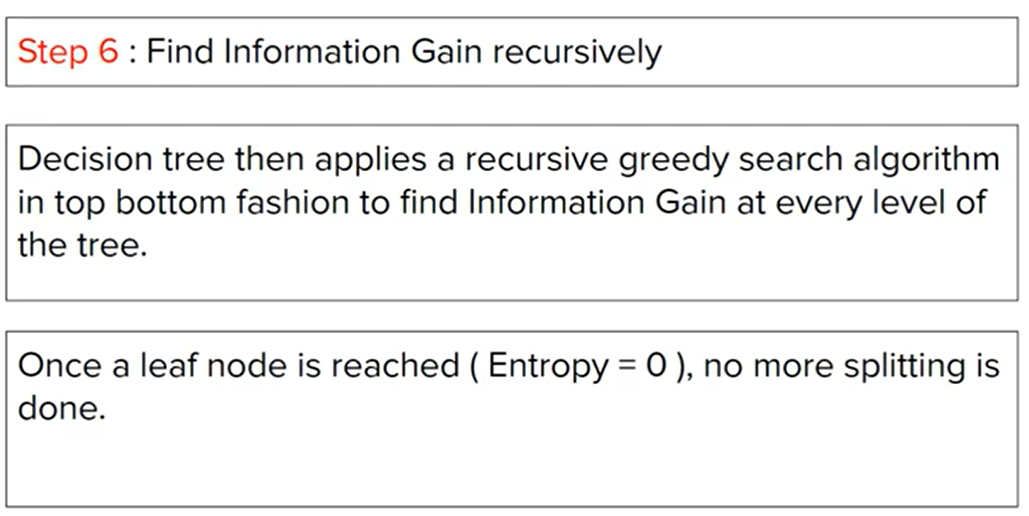
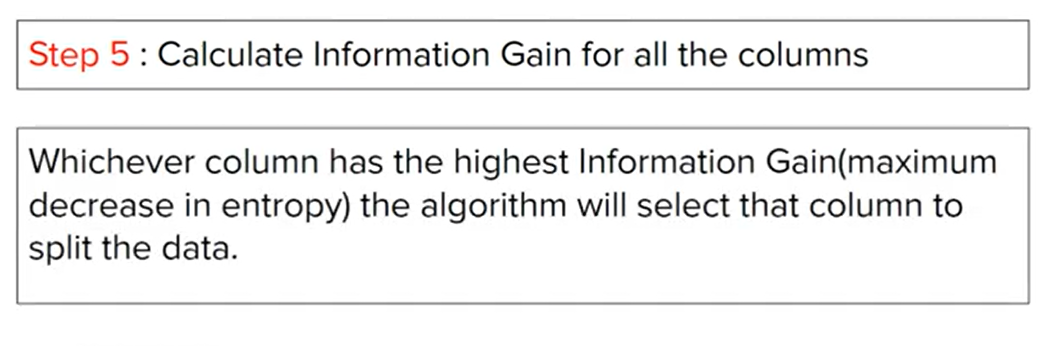
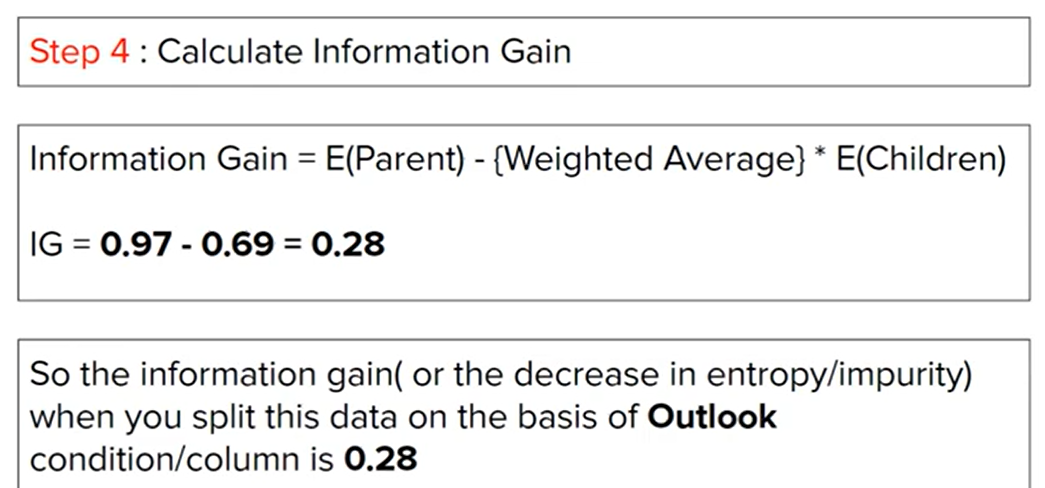
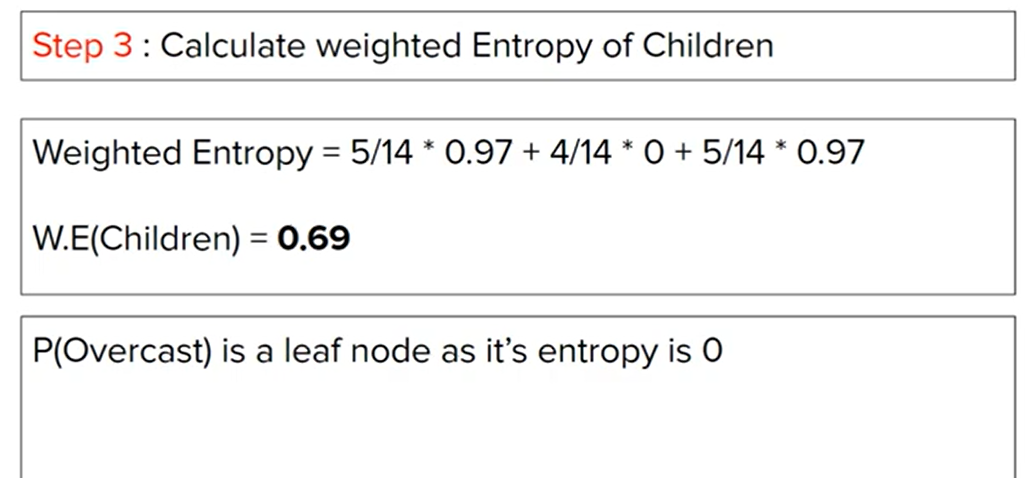
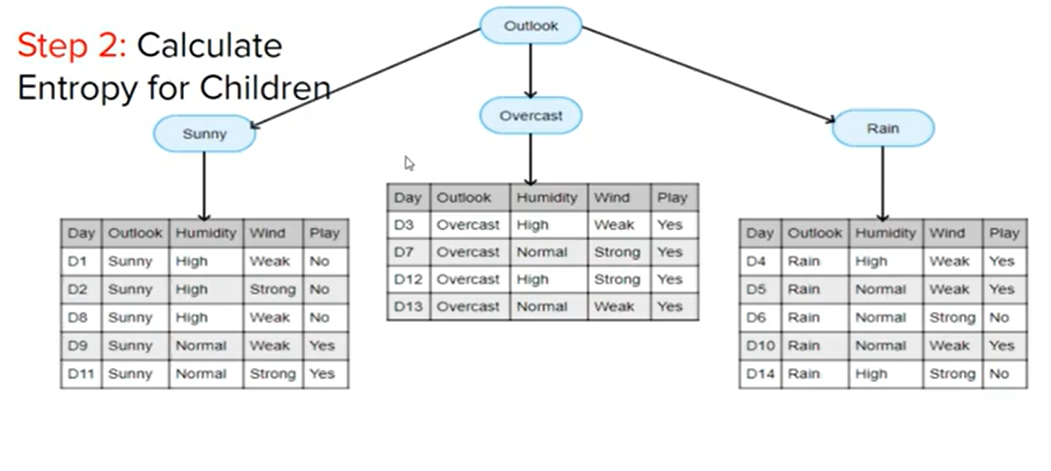


## Information Gain:

Information gain is a metric used to train decision trees. Specifically, this metric measures the quality of a split.

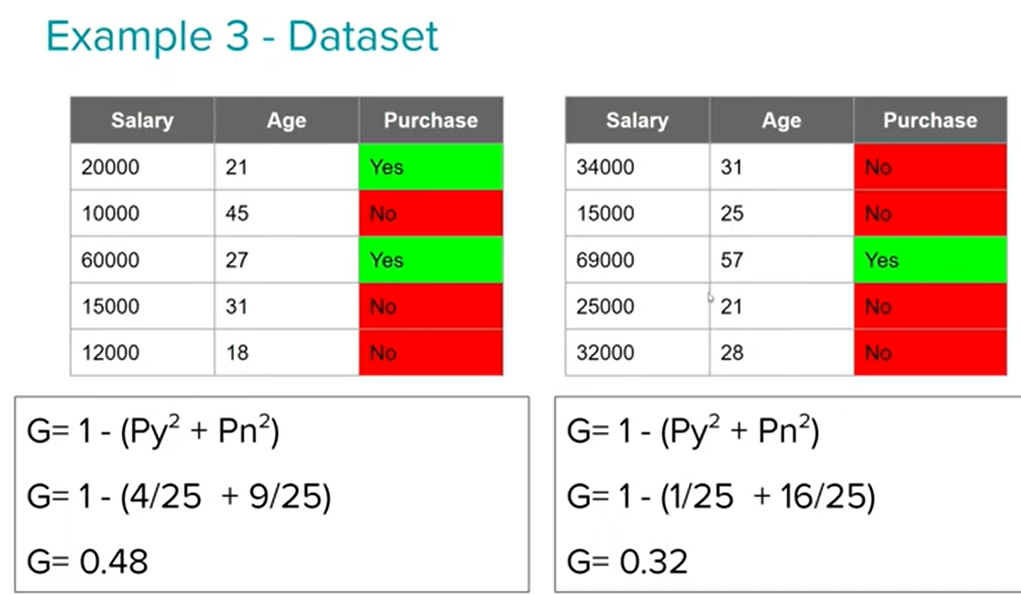
The information gain is based on decrease in entropy after a data-set split on an attribute. Constructing a decision tree is all about finding attribute that return the highest information gain.





## What is Gini Impurity

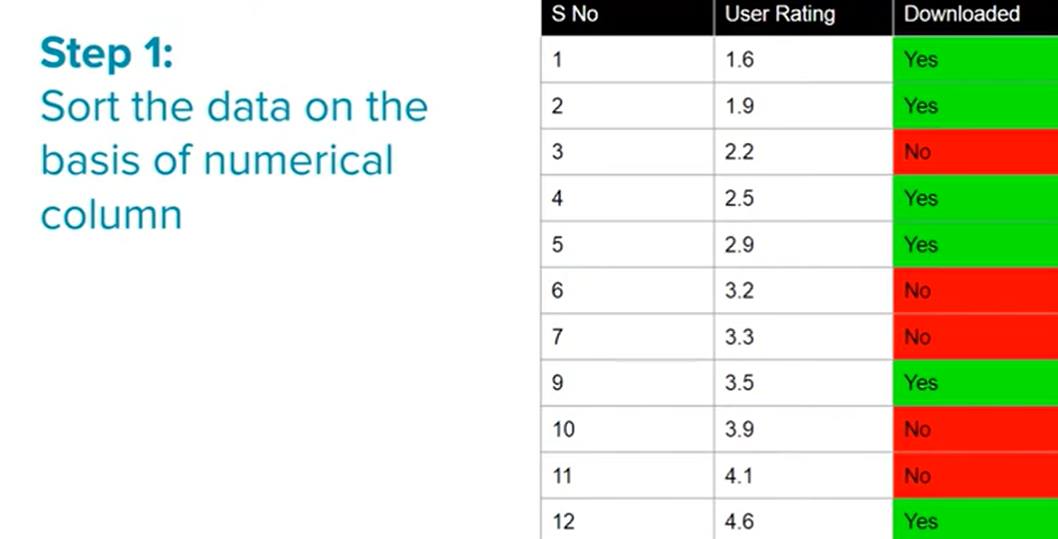
Gini impurity is a measure of impurity or uncertainty used in decision tree algorithms for binary classification. It quantifies the degree of disorder or impurity in a set of data points. The Gini impurity for a given set is calculated as follows:



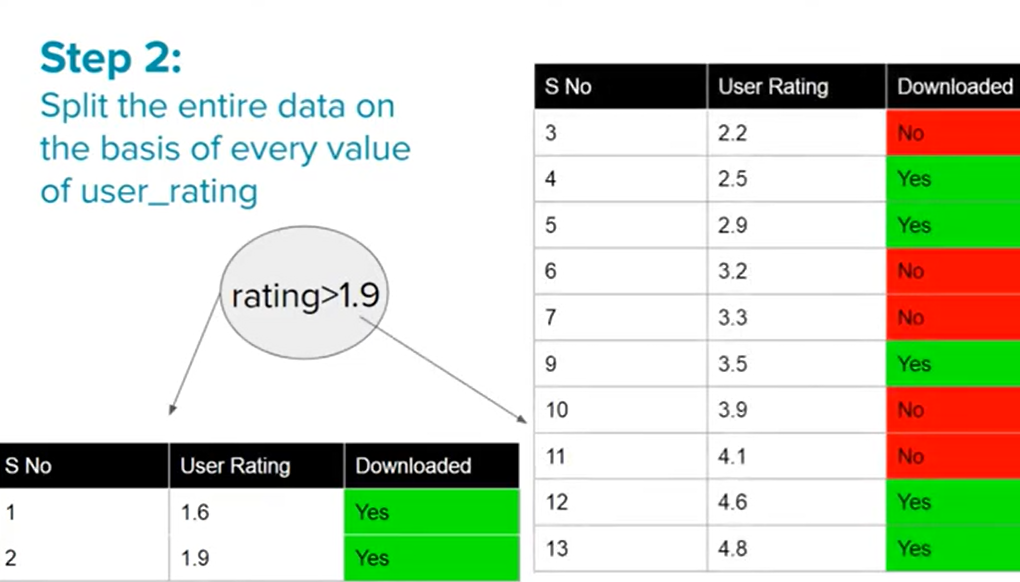
Gini impurity is computationally efficient, and it tends to be faster on large datasets compared to other measures like entropy. While entropy is another measure of impurity used in decision trees, it may lead to more balanced trees, but Gini impurity is often preferred when the goal is to identify and reduce overfitting. Gini tends to favor larger partitions with dominant classes, making it more prone to creating deeper trees that might capture noise in the data. The choice between Gini impurity and entropy may depend on the specific characteristics of the dataset and the goals of the modeling task.

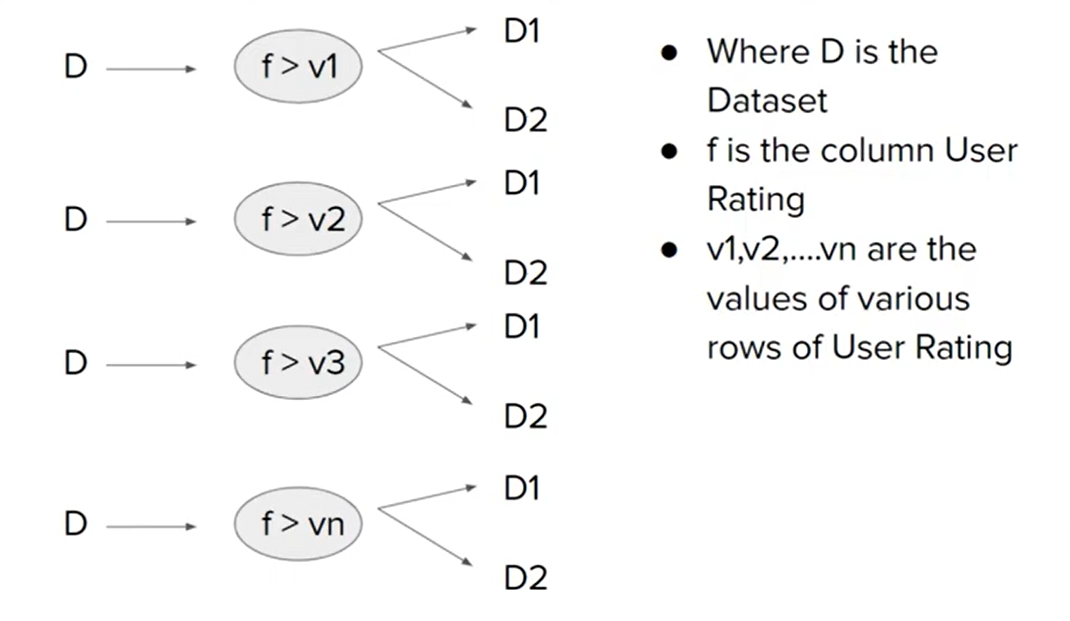
## Handling Numerical values in Decision tree:

What if we have feature column in dataset is continuous variable. First sort the data basis on the continuous variable

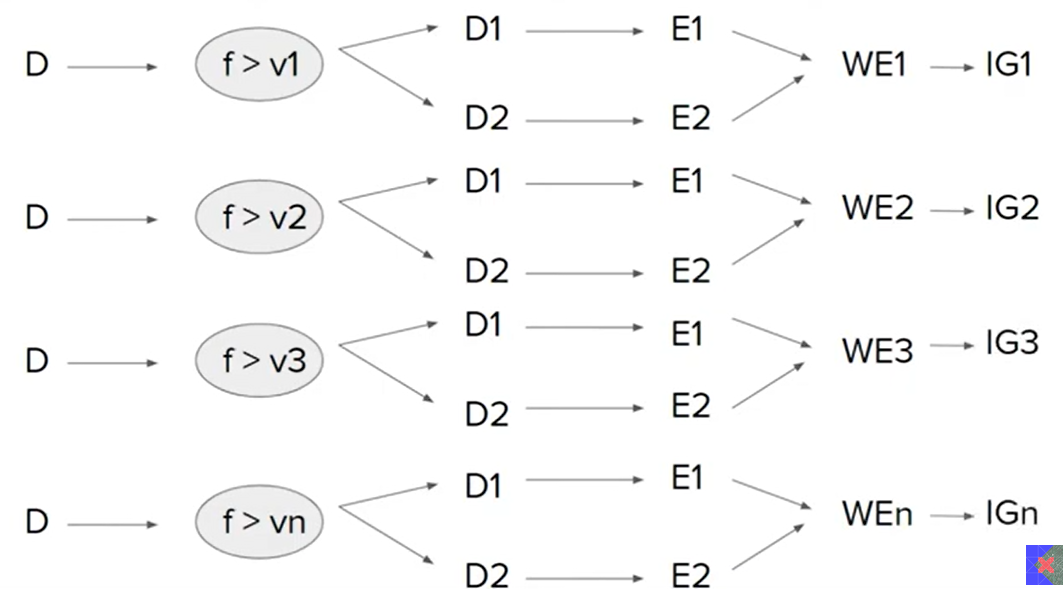


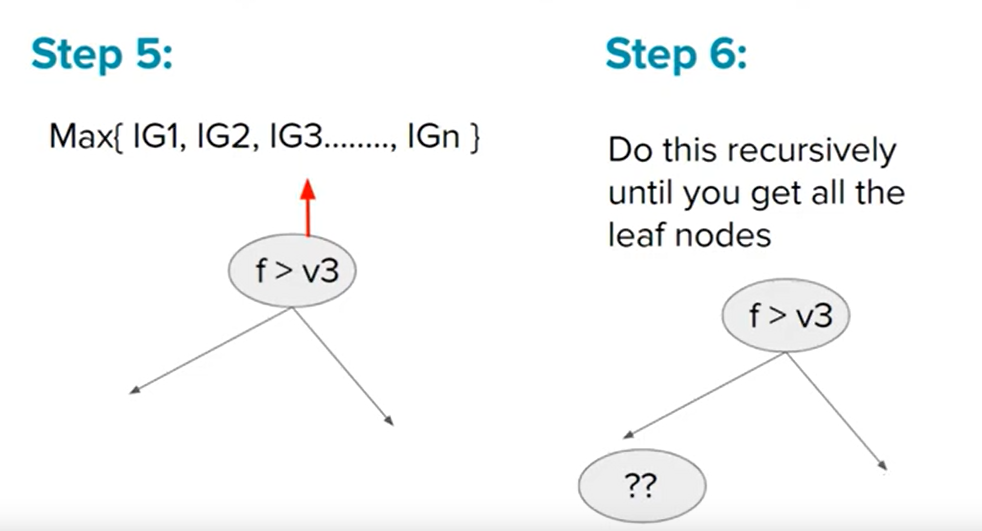
Split the data based on every value





Find the information gain on every split





## Overfitting and Underfitting:

* **Maximum Depth (Max Depth):** The maximum depth of a decision tree is the maximum distance between the root node and a leaf node. It determines the complexity of the tree. A deeper tree is more complex and can potentially capture more intricate patterns in the training data. However, this increased complexity can lead to overfitting.
* **Overfitting:** Overfitting occurs when a model learns not only the underlying patterns in the training data but also captures noise and randomness. As a result, the model performs well on the training data but fails to generalize to new, unseen data. Overfitting is a common concern when the model is too complex, and it essentially memorizes the training data instead of learning the underlying patterns.
* **Underfitting:** On the other hand, underfitting happens when the model is too simple to capture the underlying patterns in the data. It performs poorly on both the training data and new data because it fails to learn the complexities present in the dataset.
* **Role of Max Depth:** The max depth parameter is crucial in controlling the complexity of a decision tree. Setting a higher value for max depth allows the tree to grow deeper, potentially capturing more details in the training data. However, if the depth is too high, the model might fit the noise in the data, leading to overfitting.
* **Balancing Act:** Choosing an appropriate value for max depth is a balancing act. It involves finding the right level of complexity that allows the model to learn essential patterns without fitting the noise. This is often done through techniques like cross-validation, where the model is trained on a subset of the data and validated on another subset, helping to assess its generalization performance.
* **Regularization Techniques:** In addition to adjusting max depth, other regularization techniques, such as pruning, can be applied to control overfitting. Pruning involves removing branches of the tree that do not contribute significantly to the model's performance on validation data.

## Hyperparameter:

### Splitter

1. **Splitter "best":**
   * This is the default option. It means that the algorithm will evaluate all possible splits and choose the one that maximizes a certain criterion, typically either the Gini impurity or information gain (entropy), depending on the criteria specified in the criterion parameter.
2. **Splitter "random":**
   * When you set splitter to "random," the decision tree will consider a random subset of features when making each split decision. This introduces an element of randomness into the tree-building process and can help reduce overfitting, especially in situations with a large number of features.

### Max Depth

* The maximum depth of a decision tree is the maximum distance between the root node and a leaf node. It determines the complexity of the tree. A deeper tree is more complex and can potentially capture more intricate patterns in the training data. However, this increased complexity can lead to overfitting. Choosing an appropriate value for max depth is a balancing act. It involves finding the right level of complexity that allows the model to learn essential patterns without fitting the noise. This is often done through techniques like cross-validation, where the model is trained on a subset of the data and validated on another subset, helping to assess its generalization performance.

### Min Sample split

* It defines the minimum number of samples required to make a split at a node during the construction of the tree. If the number of samples at a node is less than the specified minimum, the node will not be split, and it becomes a leaf node.
* A larger value for the minimum samples split results in a simpler tree with fewer splits and deeper nodes.
* Smaller values allow the tree to be more complex with more splits, potentially capturing finer details in the training data.
* By increasing the minimum samples split, you can prevent the tree from becoming too deep and overfitting the training data.

### Min Sample leaf

* The "min\_samples\_leaf" parameter is another important hyperparameter in decision tree algorithms. Similar to "min\_samples\_split," it helps control the complexity of the decision tree and can have an impact on overfitting. The "min\_samples\_leaf" parameter specifies the minimum number of samples required to be in a leaf node.

### Max Features

The "max\_features" parameter is another hyperparameter in decision tree algorithms that controls the maximum number of features considered for a split at each node. It helps in limiting the number of features that the algorithm evaluates during the tree-building process. This parameter can be crucial in improving the generalization performance of the model, especially when dealing with datasets with a large number of features.

* The "max\_features" parameter can take different values:
* If set to an integer, it represents the exact number of features to consider at each split.
* If set to a float between 0 and 1, it represents the fraction of features to consider at each split.
* If set to "auto" or "sqrt," it considers the square root of the total number of features.
* If set to "log2," it considers the base-2 logarithm of the total number of features.
* If set to "None," it considers all features at each split (equivalent to setting it to the total number of features).

## Regression trees

Decision trees are versatile models that can be used for both classification and regression tasks. When used for regression, they are often referred to as Decision Tree Regressors. The fundamental idea of a decision tree remains the same: it recursively partitions the data into subsets based on feature values, but instead of predicting class labels at the leaves, decision tree regressors predict a continuous target variable.

Here's how a decision tree regression works:

1. **Node Splitting:**
   * Similar to classification, the algorithm starts at the root node and selects a feature to split the data based on a chosen criterion.
   * The goal is to minimize the variance or mean squared error (MSE) of the target variable within each node.
2. **Splitting Criteria:**
   * The choice of splitting criteria for decision tree regression is often based on reducing the variance of the target variable.
   * Popular splitting criteria include minimizing the variance of the target values in each leaf node.
3. **Recursive Splitting:**
   * The splitting process is repeated recursively for each subset until a stopping criterion is met. This can include a maximum depth for the tree, a minimum number of samples in a leaf node, or other criteria.
4. **Prediction:**
   * When a new data point is introduced to the tree, it traverses the tree from the root to a leaf node based on the values of the features.
   * The predicted value for the target variable is often the mean (or median) of the training target values in the leaf node.