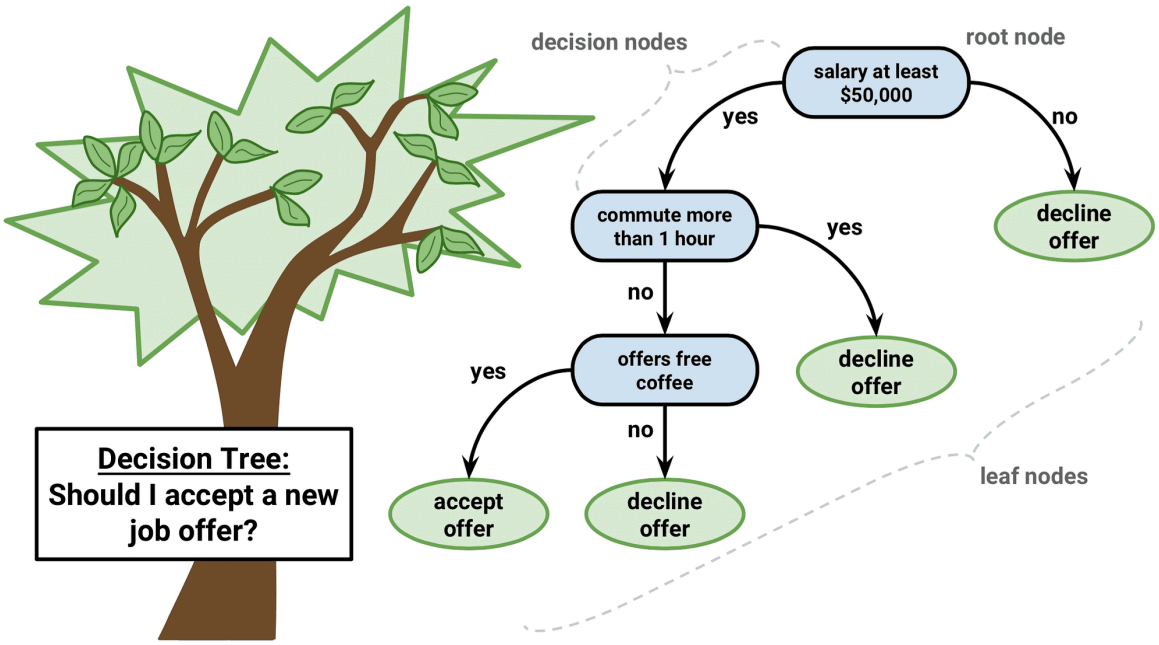
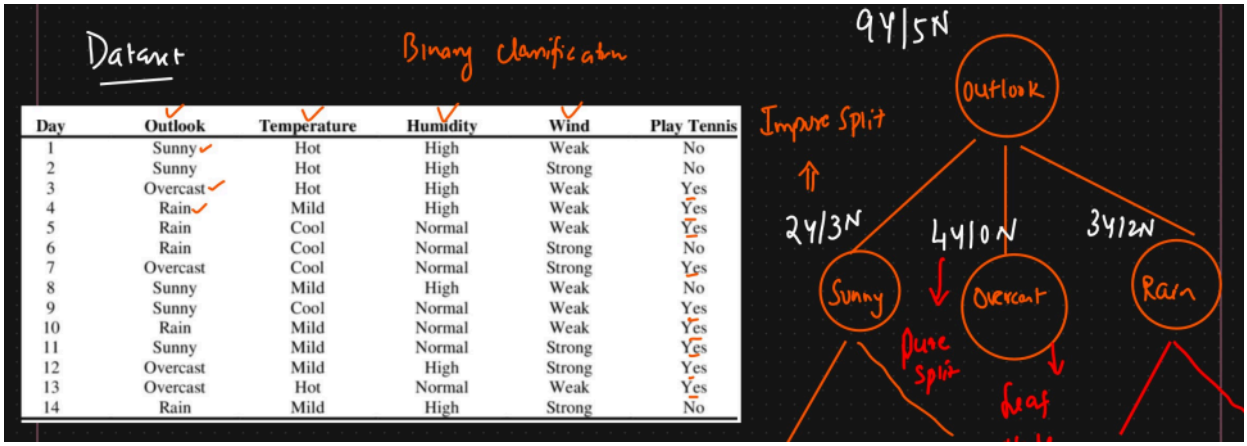
# Decision Trees

- **What it is:** A supervised machine learning algorithm used for both **classification** and **regression** tasks.
- **Structure:** It uses a tree-like model of decisions and their possible consequences. It visually resembles an upside-down tree or a flowchart.
- **How it works:** It breaks down a dataset into smaller and smaller subsets while incrementally developing an associated decision tree. The final result is a tree with decision nodes and leaf nodes.




# Types of Decision Trees

1. **Classification Trees:** Used when the target variable is categorical (discrete classes). Leaf nodes represent the predicted class label.
2. **Regression Trees:** Used when the target variable is continuous (numerical values). Leaf nodes typically represent the mean (or median) of the target variable for the training samples that fall into that leaf.

# Structure of a Decision Tree

1. **Root Node:** The topmost node representing the entire dataset or population. It's the starting point of the decision-making process.
2. **Internal Nodes (Decision Nodes):** Nodes that branch out, representing a test or decision based on a specific feature (attribute). Each branch emerging from an internal node represents an outcome of that test.
3. **Branches:** Links connecting nodes, representing the decision rules or outcomes of tests.
4. **Leaf Nodes (Terminal Nodes):** Nodes at the end of the branches that do not split further. They represent the final outcome or prediction (a class label in classification or a continuous value in regression).

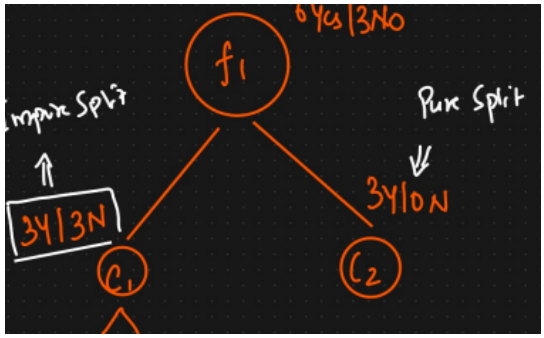
# How Decision Trees Work: Splitting

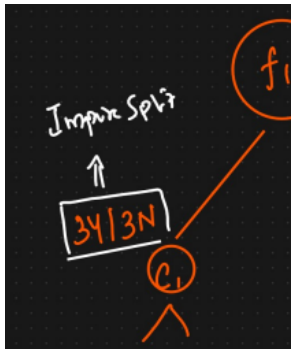
The core idea is **recursive partitioning**: repeatedly splitting the data into more homogeneous (purer) subsets based on feature values.

- **Goal:** To split the nodes in a way that the resulting child nodes are as pure as possible with respect to the target variable.
- 

- **Splitting Criteria:** Algorithms use metrics to evaluate the "goodness" of a potential split based on different features. The feature providing the best split according to the chosen criterion is selected at each node. Common criteria include:

    1. **Purity in Decision Trees:**
        - In the context of a decision tree node (which contains a subset of data), "purity" refers to how **homogeneous** the data in that node is with respect to the target variable (the class label in classification).
        - A node is **perfectly pure** if all the data points within it belong to the **same class**.

- 

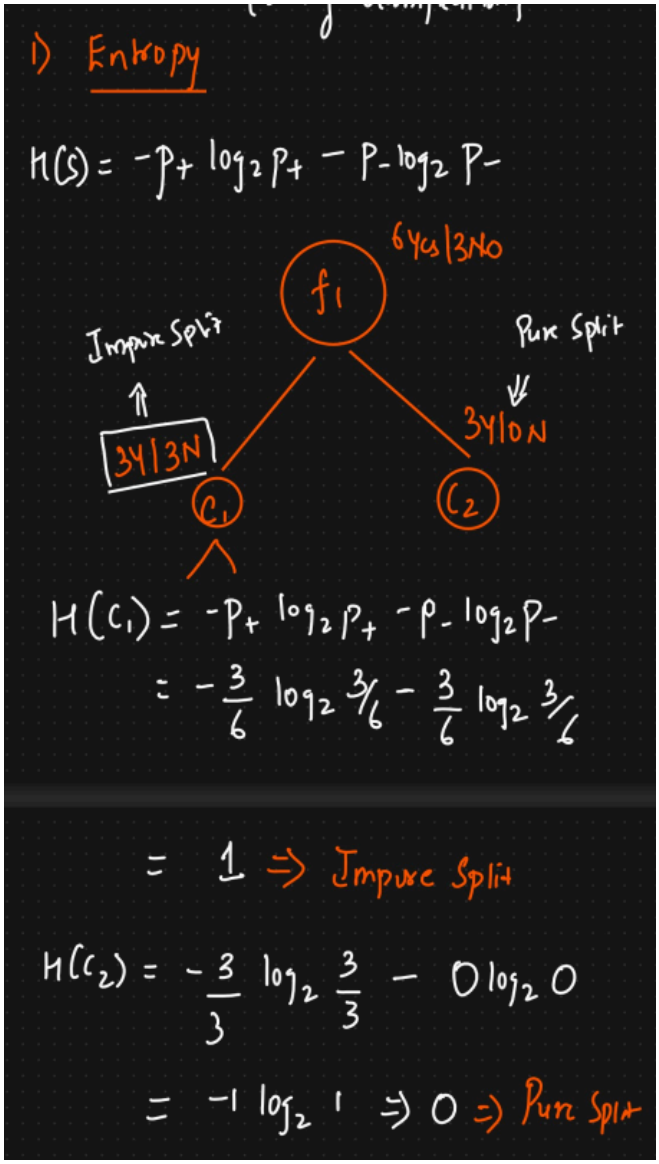
- A node is **maximally impure** if the data points within it are **evenly distributed** among all possible classes.
- 

- The goal of building a decision tree is to progressively split the data in a way that increases the purity of the resulting child nodes.

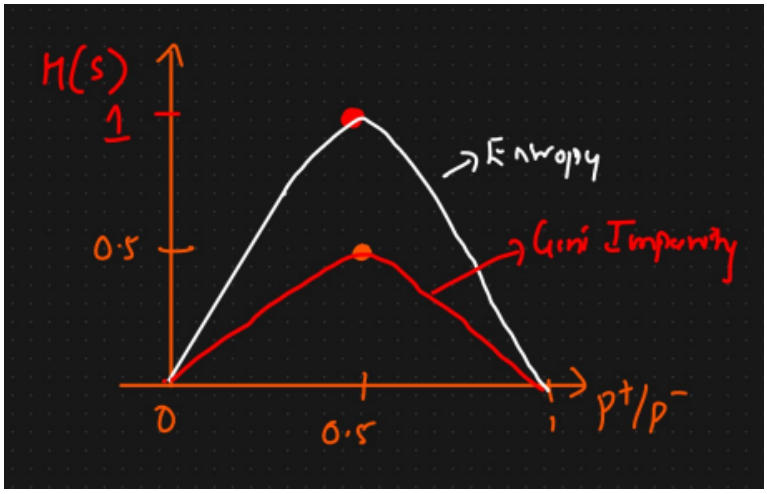
2. **Pure vs. Impure Split:**
   - **Pure Split:** A split that results in child nodes where each node contains data points belonging to only *one* class.
   - **Impure Split:** A split that results in child nodes where data points still belong to *multiple* classes. Most splits during the tree-building process are impure to some degree. The algorithm aims to find splits that *reduce impurity* the most compared to the parent node.
3. **Measuring Impurity:** Decision tree algorithms need a quantitative way to measure the impurity of a node to decide which split is best. The two most common metrics are Entropy and Gini Impurity.
   - **Entropy:**
     - **Concept:** Borrowed from information theory, entropy measures the level of **uncertainty, randomness, or disorder** in a set of data points.
     - **In Decision Trees:** It quantifies the impurity of a node based on the distribution of classes within it.
     - **Calculation:** For a node with C classes, entropy is calculated as:

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

     - where $p_i$ is the proportion (probability) of data points belonging to class $i$ in that node.

- 

- **Range:**
  - Entropy = 0: Perfectly pure node (all data points belong to one class, pi=1 for one class and 0 for others). There's no uncertainty.
  - Entropy = Maximum value (e.g., 1 for binary classification): Maximally impure node (data points are evenly split among classes, e.g., 50% Class A, 50% Class B). Maximum uncertainty.

- 

- **Use in Splitting (Information Gain):** Algorithms like ID3 and C4.5 aim to find the split that results in the largest **Information Gain**, which is the reduction in entropy achieved by the split. (Information Gain = Entropy(parent) - Weighted Average Entropy(children)).

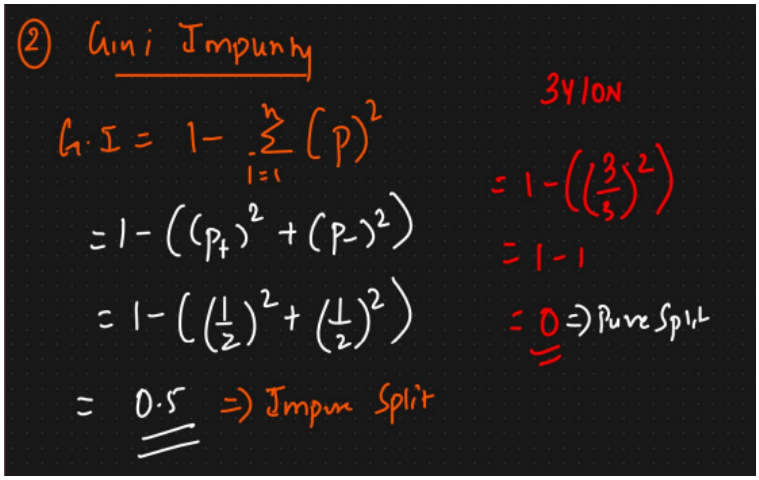
■ **Gini Impurity:**

- **Concept:** Measures the probability of **misclassifying** a randomly chosen element from the node if it were randomly labeled according to the class distribution in that node.
- **In Decision Trees:** It quantifies the impurity of a node.
- **Calculation:** For a node with C classes, Gini Impurity is calculated as:

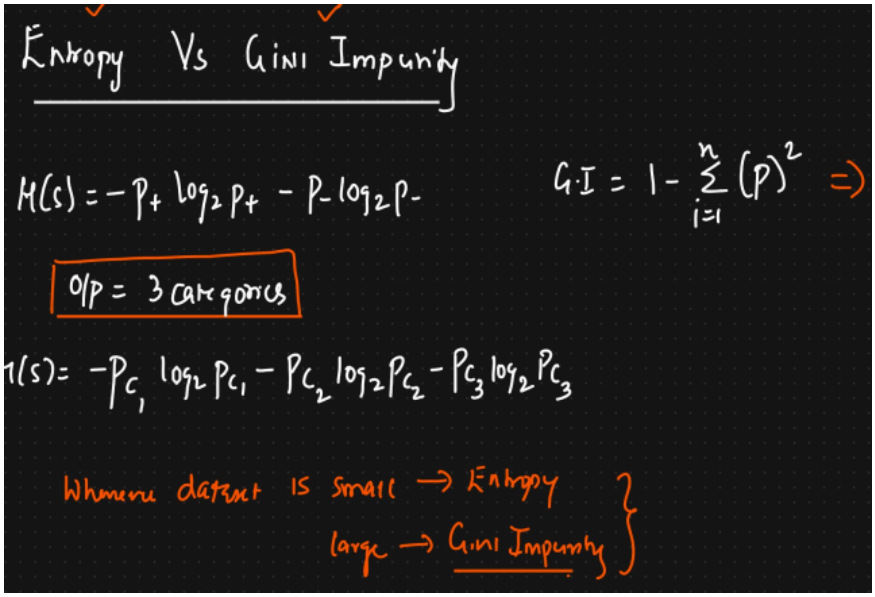
$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

where $p_i$ is the proportion (probability) of data points belonging to class $i$ in that node.

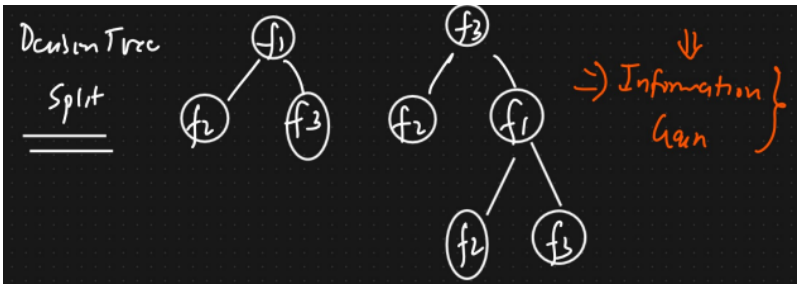
- **Range:**
  - Gini = 0: Perfectly pure node (all data points belong to one class). Zero chance of misclassification within the node.
  - Gini = Maximum value (e.g., 0.5 for binary classification): Maximally impure node (data points are evenly split). Highest chance of misclassification.

- 


- **Use in Splitting (CART Algorithm):** The CART algorithm seeks splits that minimize the weighted average Gini Impurity of the resulting child nodes. It favors splits that create purer nodes.



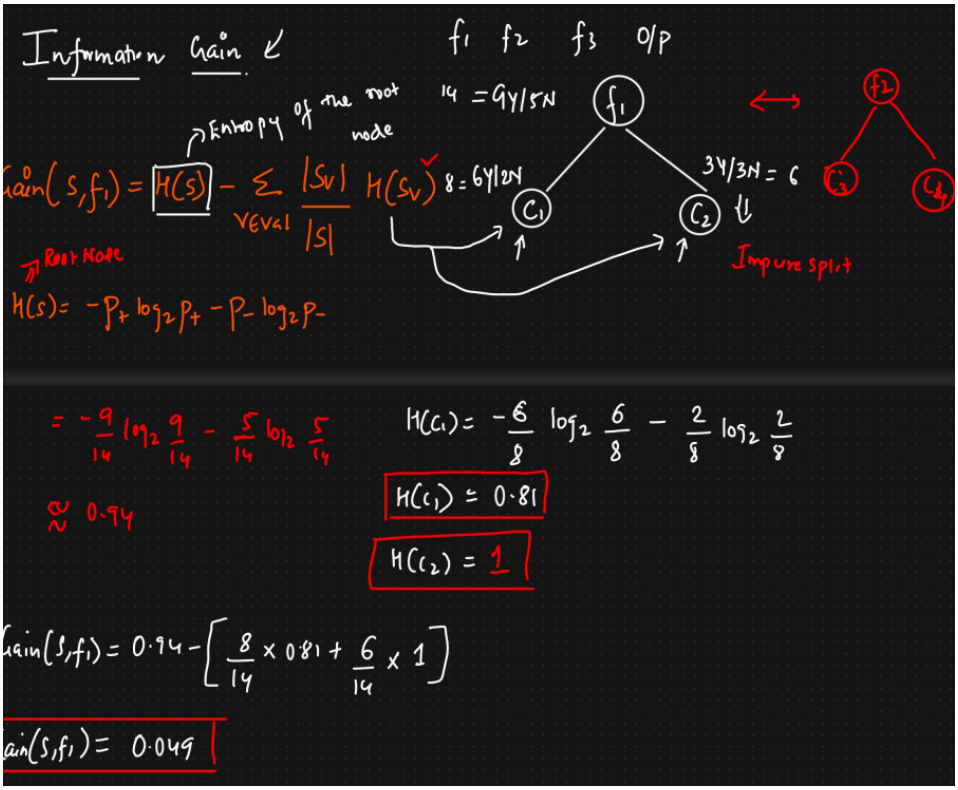

4. **Information Gain (Based on Entropy, used in ID3, C4.5):**

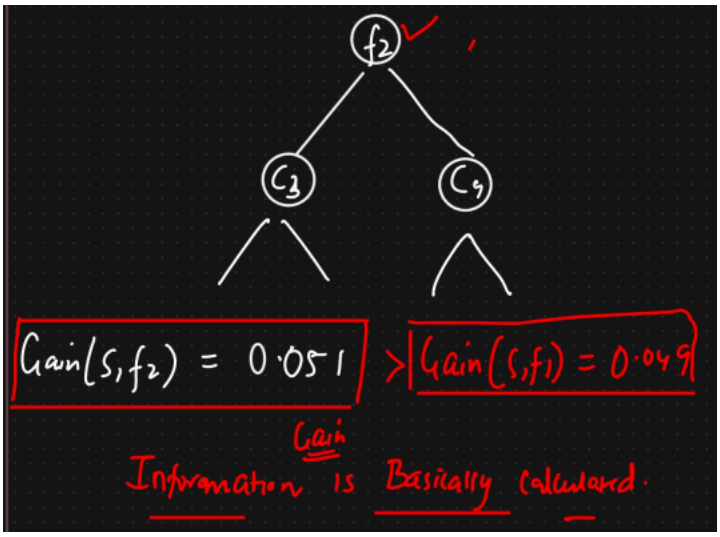
   - Measures the reduction in entropy (uncertainty) achieved by splitting the data on a particular feature.
   - 

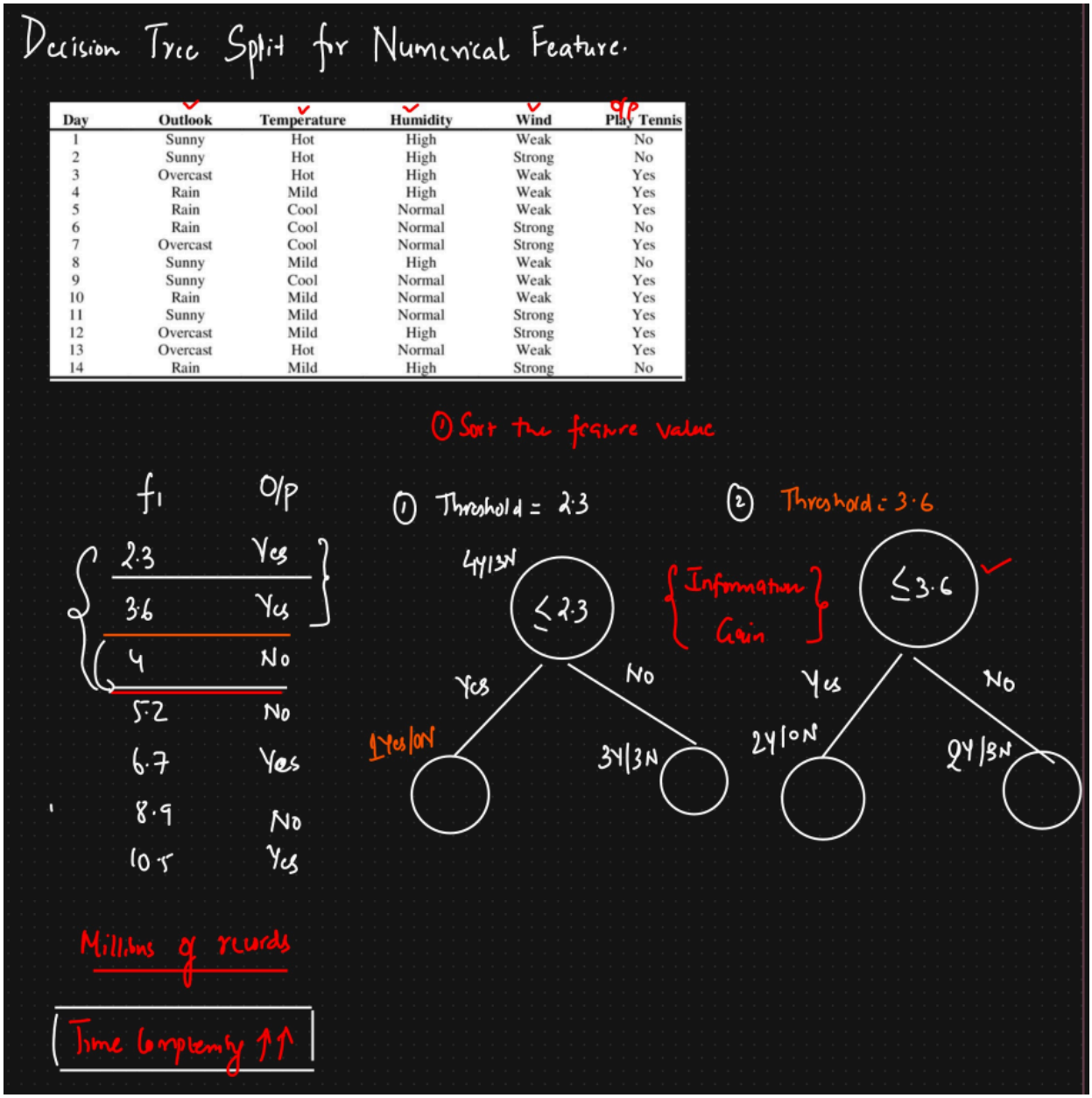
**Entropy:** Measures the impurity or randomness in a set of examples. Formula: $Entropy = -\sum_{i=1}^{C} p_i \log_2(p_i)$. Entropy is 0 for a pure node and maximal when classes are equally distributed.

**Information Gain:** $Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$, where S is the set of examples, A is the feature, and $S_v$ is the subset of S for which feature A has value $v$.

- The algorithm seeks splits that maximize Information Gain. (Note: Information Gain can be biased towards features with many values. C4.5 uses Gain Ratio to compensate for this).



- 
f2
C3
C9
Gain(S, f2) = 0.051
>
Gain(S, f1) = 0.049
Information Gain is Basically calculated.



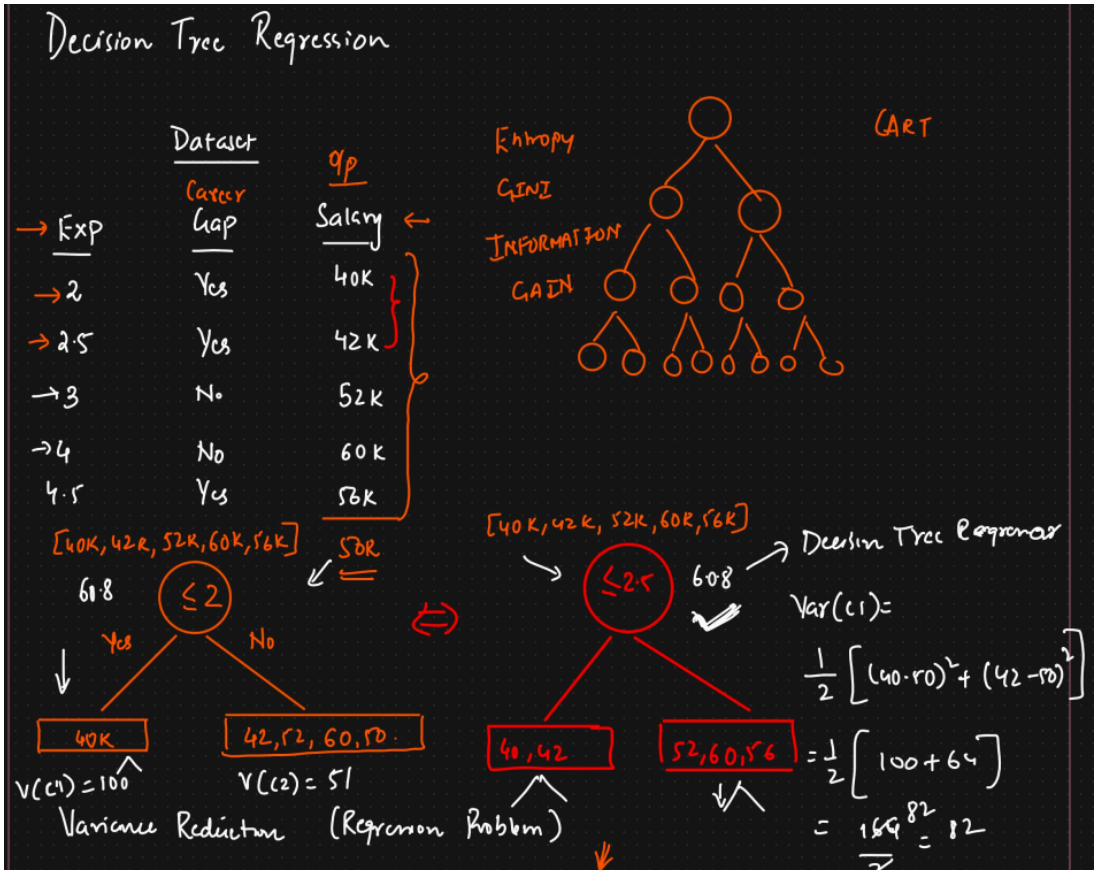

**Reduction in Variance (Used for Regression Trees):**

Decision Trees can also be effectively used for regression tasks, where the goal is to predict a **continuous numerical value** (like the price of a house, temperature, or sales amount) instead of a discrete class label.

Here are the key basics of Decision Tree Regressors:

1. **Purpose:** To predict a continuous target variable.

2. **Structure:** It still uses the same tree-like structure as a classification tree, with a root node, internal nodes representing tests on features, branches representing outcomes of tests, and leaf nodes.
3. **Splitting Criterion (How it Differs from Classification):**
    - Instead of using Gini Impurity or Entropy/Information Gain (which measure class purity), regression trees aim to minimize the **variance** or **Mean Squared Error (MSE)** within the nodes.
    - At each internal node, the algorithm searches for the feature and the threshold value that splits the data into two subsets such that the variance (or MSE) of the target variable within those subsets is minimized the most.
    - **Intuition:** The algorithm tries to find splits that make the target values within each resulting child node as similar to each other (i.e., having low variance) as possible. It wants to group data points with similar target values together.
    - MSE Calculation for a node: $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2$, where $N$ is the number of samples in the node, $y_i$ is the target value of the $i$-th sample, and $\bar{y}$ is the mean of the target values in that node. The algorithm chooses the split that minimizes the weighted average MSE of the child nodes.
4. **Prediction:**
    - To predict the value for a new input instance, the instance traverses the tree from the root node down, following the decision rules at each internal node based on its feature values.
    - When the instance reaches a **leaf node**, the predicted value is typically the **average (mean)** of the target variable values of all the *training* instances that ended up in that same leaf node during the training phase.
5. **Output Characteristics:**
    - Because the prediction for any instance falling into a specific leaf is the *same* constant value (the average of the training samples in that leaf), the overall output of a Decision Tree Regressor is **piecewise constant**. When plotted, the predictions often look like steps or blocks rather than a smooth curve.
6. **Overfitting and Regularization:**
    - Just like classification trees, regression trees are prone to overfitting if allowed to grow very deep. An overfit regression tree will essentially create leaves for very small groups of training samples, fitting the noise in the data.
    - Techniques like **pruning** (post-pruning or pre-pruning by setting hyperparameters like `max_depth`, `min_samples_split`, `min_samples_leaf`) are crucial to control complexity and improve generalization to unseen data.
7. **Advantages/Disadvantages (in Regression Context):**
    - **Advantages:** Simple to understand and interpret, handles non-linear relationships, requires little data preparation.
    - **Disadvantages:** Piecewise constant predictions (cannot produce smooth outputs), sensitivity to small changes in data, potential for overfitting.


Decision Tree Regression
Dataset
Career Gap
Exp
Salary
Yes
Yes
No
No
Yes
40K
42K
52K
60K
56K
2
2.5
3
4
4.5
[40K, 42K, 52K, 60K, 56K]
50K
61.8
≤2
Yes
No
40K
42, 52, 60, 50
V(C1) = 100
V(C2) = 51
Variance Reduction (Regression Problem)
Entropy
GINI
INFORMATION GAIN
CART
[40K, 42K, 52K, 60K, 56K]
≤2.5
60.8
Decision Tree Regressor
Var(C1) =
1/2 [(40-50)^2 + (42-50)^2]
40, 42
52, 60, 56
= 1/2 [100 + 64]
= 164/2 = 82

① Variance $= \frac{1}{n}\sum_{i=1}^{n}(y-\bar{y})^2$ — Mean Square Error → Average

Variance (Root) $= \frac{1}{5}\left[(40-50)^2+(42-50)^2+(82-50)^2+(60-50)^2+(56-50)^2\right]$

$= \frac{1}{5}\left[100+64+4+100+36\right]$

$= 60.8$

Variance (C1) $= \frac{1}{n}\sum_{i=1}^{n}(y-\bar{y})^2$

$= \frac{1}{1}(40-50)^2$

$= 100$

Var(child2) $= \frac{1}{3}\left[4+100+36\right]$

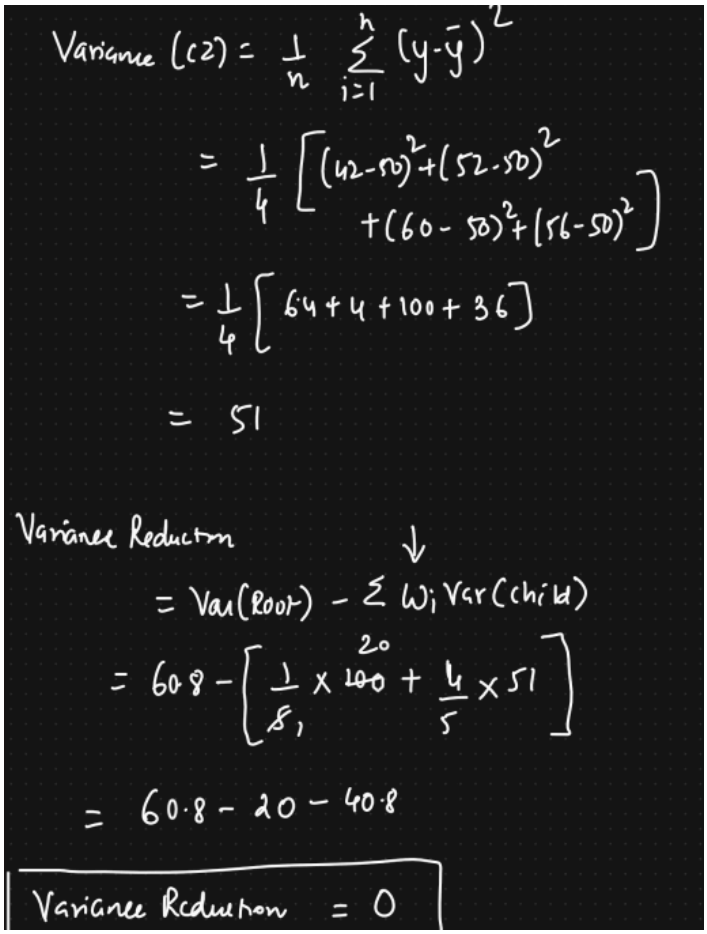
$= \frac{140}{3} = 46.66$

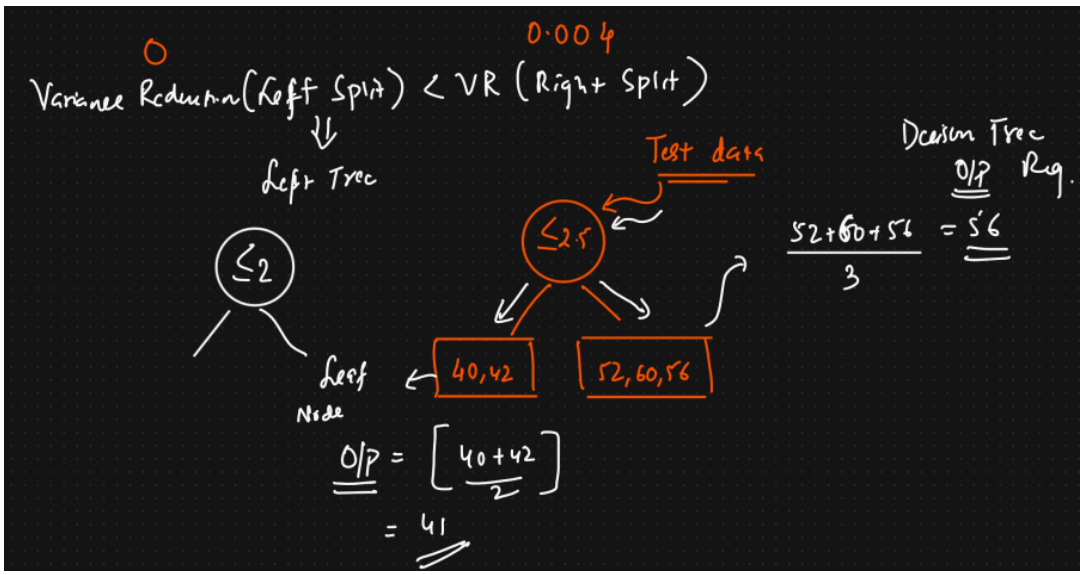
Variance Reduction

$= 60.8 - \left[\frac{2}{5}\times 82 + \frac{3}{5}\times 46.66\right]$

$= 60.8 - \left[32.8 + 27.996\right]$

$= +0.004$




## Common Decision Tree Algorithms

- **ID3 (Iterative Dichotomiser 3):** Uses Information Gain for splitting. Typically handles categorical features.
- **C4.5:** An extension of ID3. Uses Gain Ratio (an improvement over Information Gain) and can handle both continuous and categorical features, as well as missing values.
- **CART (Classification And Regression Tree):** Can handle both classification (using Gini Impurity) and regression (using variance reduction). Creates **binary splits** (each node splits into exactly two branches). (Used in sklearn).

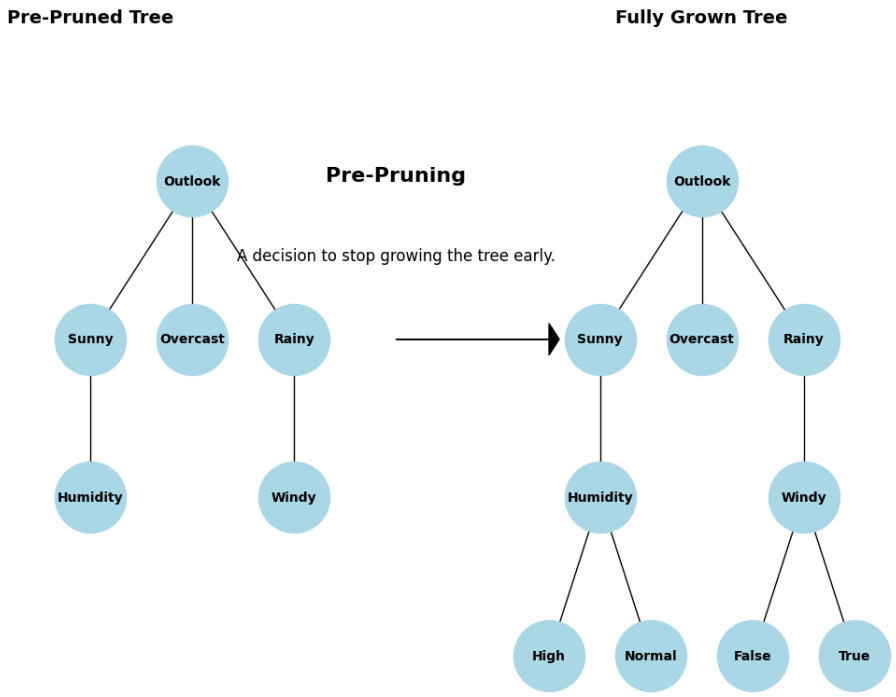
# Overfitting and Pruning

- **Overfitting:** Decision trees can easily become overly complex, learning the noise and specific patterns of the training data too well. This leads to poor performance on unseen data. A tree grown to its maximum depth often overfits.

- **Pruning:** A technique used to reduce the size and complexity of the decision tree, improving its generalization ability and preventing overfitting.

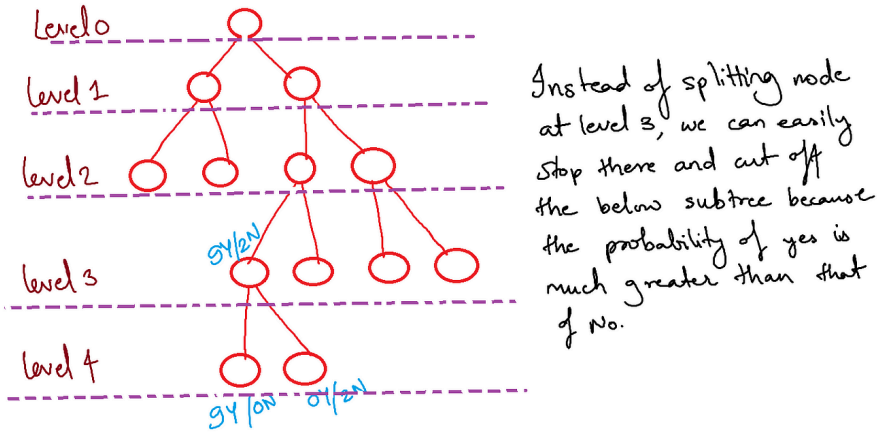
    1. **Pre-pruning (Early Stopping):**

        - Stops the tree-building process early, before it perfectly classifies the training data.
        - Criteria include: setting a maximum tree depth, minimum number of samples required in a node to split, minimum number of samples required in a leaf node, or minimum improvement in the impurity measure required for a split.
        - Faster, but risks stopping too early and creating an underfit model (the "horizon effect").




2. **Post-pruning:**

   - Allows the tree to grow to its full depth (potentially overfitting).
   - Then, it removes or collapses branches/nodes that provide little predictive power or might be based on noise.
   - Common techniques include Reduced Error Pruning (using a validation set) and Cost Complexity Pruning (balancing tree complexity against its error rate).
   - Often results in better-performing trees but is more computationally intensive.




# Advantages of Decision Trees

- **Interpretability:** Easy to understand, visualize, and explain (white-box model).
- **Minimal Data Preparation:** Often requires less data preprocessing compared to other algorithms (e.g., no need for feature scaling or normalization). Can handle both numerical and categorical data.
- **Handles Non-linearity:** Can capture complex non-linear relationships between features and the target.
- **Feature Selection:** Implicitly performs feature selection, as important features are used higher up in the tree.
- **Versatility:** Applicable to both classification and regression problems.
- **Robustness:** Relatively robust to outliers (though extreme outliers can still influence splits).

# Disadvantages of Decision Trees

- **Overfitting:** Highly prone to overfitting if not pruned or regularized.
- **Instability:** Small variations in the data can lead to significantly different trees being generated.
- **Greedy Approach:** The splitting process is greedy (makes the best local choice at each step) and may not result in a globally optimal tree.
- **Bias:** Can be biased towards features with more levels or dominant classes in imbalanced datasets.
- **Complexity:** Can become computationally expensive to train on very large datasets.
- **Piecewise Predictions:** Regression predictions are piecewise constant (step-like), not smooth.