Decision tree first makes a statement then make a decision based on whether or not the statement is true or false. Like the tree has roots, branches, leaves. Here Decision trees uses these to divide the problem and solve it.

Root node: only have arrows pointing away from it, where the initial decision or feature is considered. it represents the entire dataset or population being analyzed.

Branch node(Internal Node): have arrows pointing towards it and away from it, Branch nodes represent intermediate decisions or features, receiving input from their parent node and passing their output to their child nodes.

Leaves (Terminal Node): only have arrows pointing towards it , Leaf nodes represent the final outcomes or predictions of the decision tree. they represent a specific classification or regression value.

Yes, I can define the components of a decision tree in terms of the flow of decision-making:

* The **root node** is the starting point. It has arrows only pointing *away* from it, representing the initial decision or feature considered for the entire dataset.
* **Branch nodes**, or internal nodes, have arrows pointing both *towards* and *away* from them. They represent intermediate decisions, receiving input and passing their output along the tree.
* **Leaf nodes**, or terminal nodes, have arrows only pointing *towards* them. They represent the final outcomes or predictions of the tree, such as a classification or regression result."

The aim of the decision trees is to obtain the pure leaves i.e. categories present in leave should belong to one class. It try to decrease the impurity as the tree grows from root to leaves.

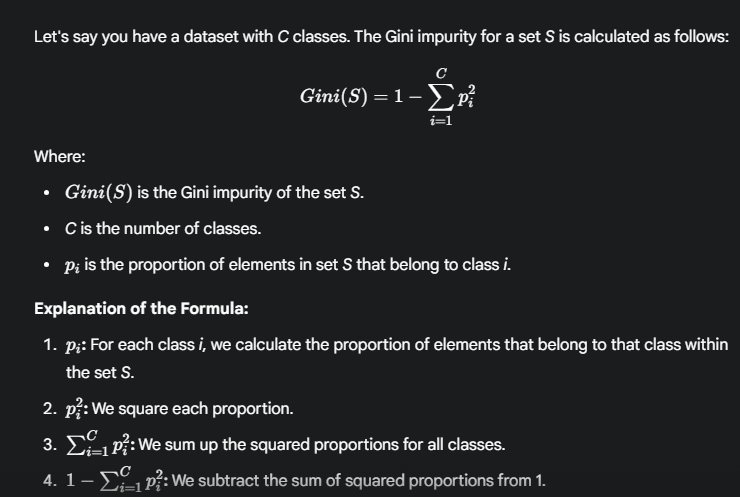
We measure the impurity of the leaves by using two concepts

1. Gini Impurity 2) Entropy or Information Gain

**Gini Impurity Definition:**

Gini impurity is a measure of the impurity or disorder of a set of elements. In the context of decision trees, it's used to evaluate how well a split in the data separates different classes. A lower Gini impurity indicates a higher level of purity, meaning the elements within the subset are more likely to belong to the same class.

Essentially, it quantifies the probability of misclassifying a randomly chosen element in a dataset if that element were randomly labeled according to the distribution of labels in the subset.



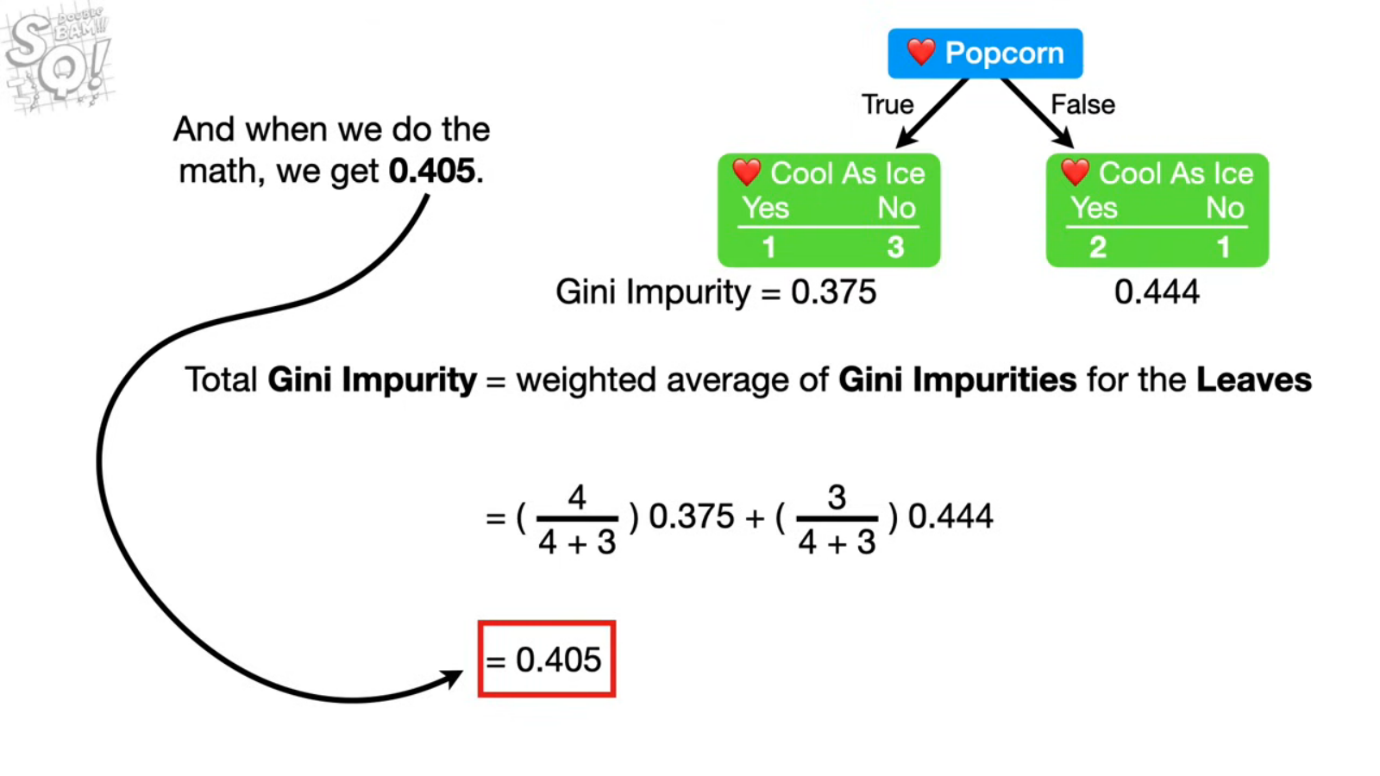
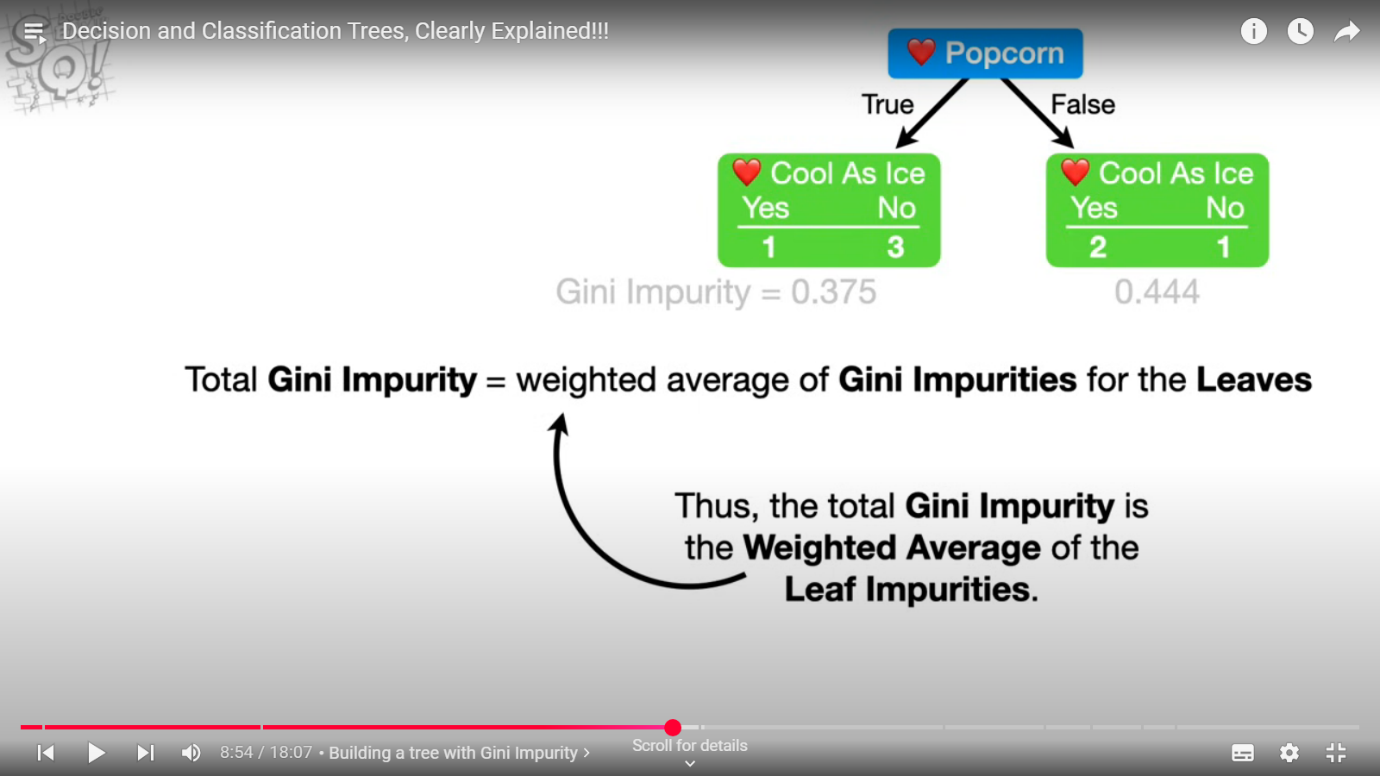
**Example:**

Suppose you have a set *S* with two classes (A and B).

* If all elements belong to class A, then pA​=1 and pB​=0.
  + Gini(S)=1−(12+02)=1−1=0. (Perfect purity)
* If the set is equally split between class A and B, then pA​=0.5 and pB​=0.5.
  + Gini(S)=1−(0.52+0.52)=1−(0.25+0.25)=1−0.5=0.5. (Maximum impurity for a 2 class problem)

Therefore, the Gini impurity ranges from 0 (perfect purity) to a maximum of 1−(1/C) (maximum impurity), where C is the number of classes.

For binary classification this max impurity is 0.5.



After calculating the Total Gini Impurity of the features to select a root node. where the Gini impurity is lowest that one is selected as Root node. then further branch nodes are derived from the rest features until we get pure leaves.

**Regression Tree:**

For regression tree we choose Node having Least MSE to split

**Pruning Trees:**

To avoid overfitting we prune the tree. one such technique is cost complexity pruning or weakest link pruning. i.e. to fit the tree on the testing data well, we prune the tree i.e. we cut the branches. so the question is how may nodes to cut to good fit on testing data as well.

**Regression Tree Pruning:**

we build the full sized tree first and prune some branches and build the trees.

We calculate tree scores for all trees

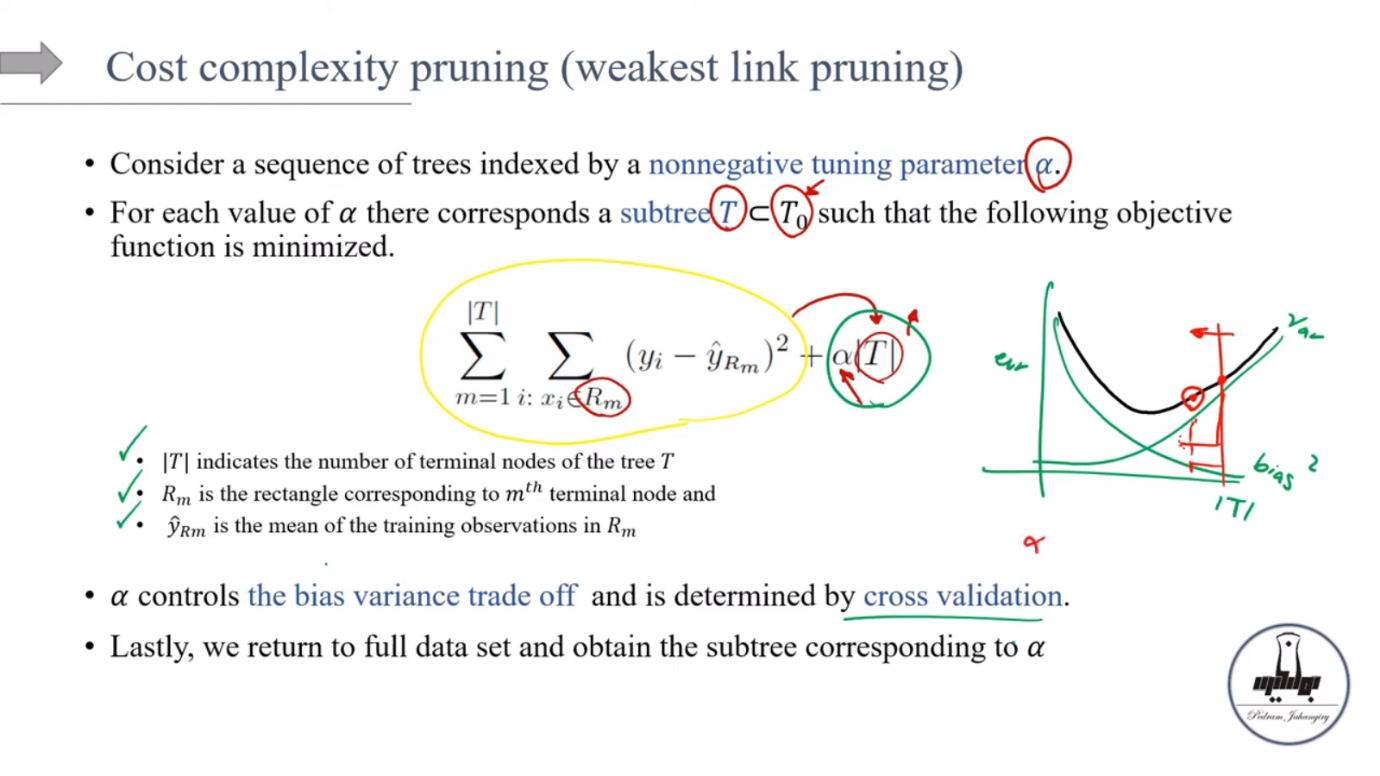
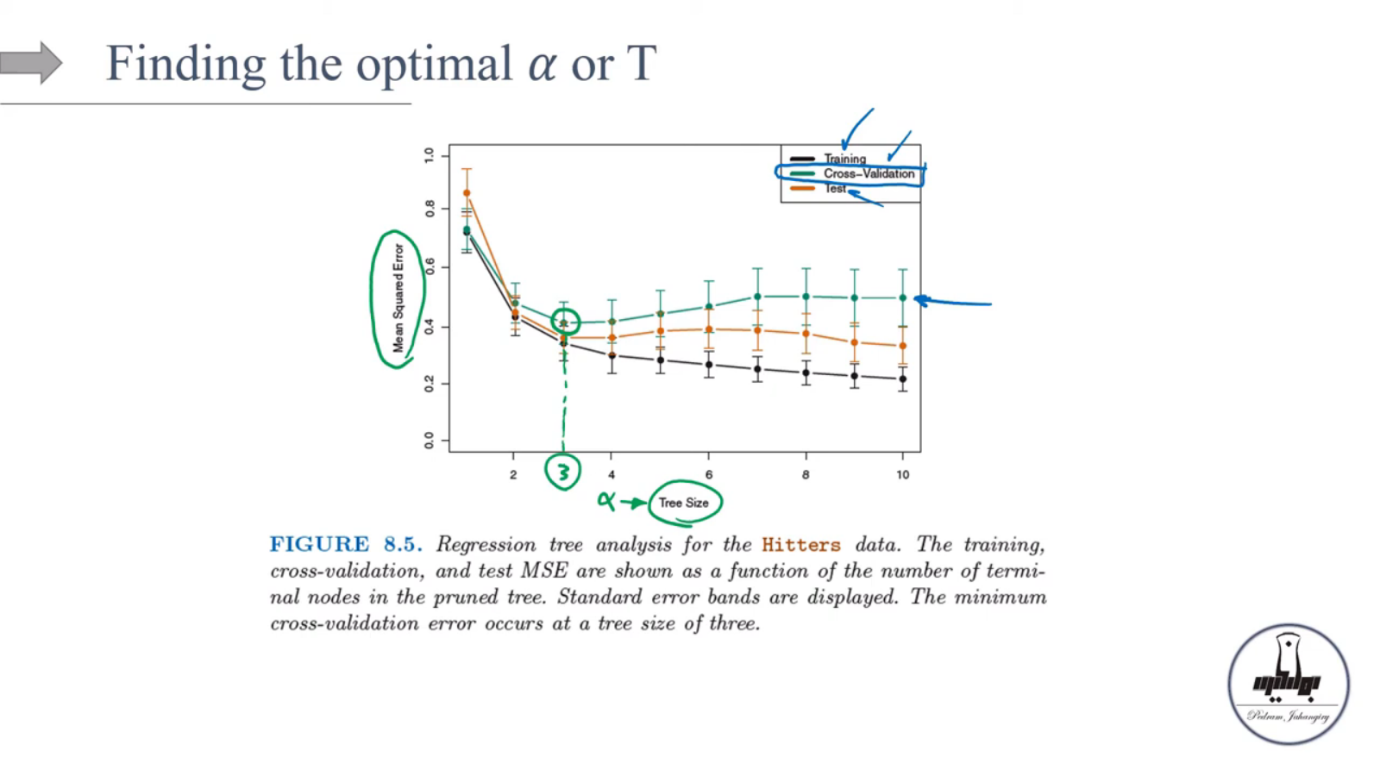
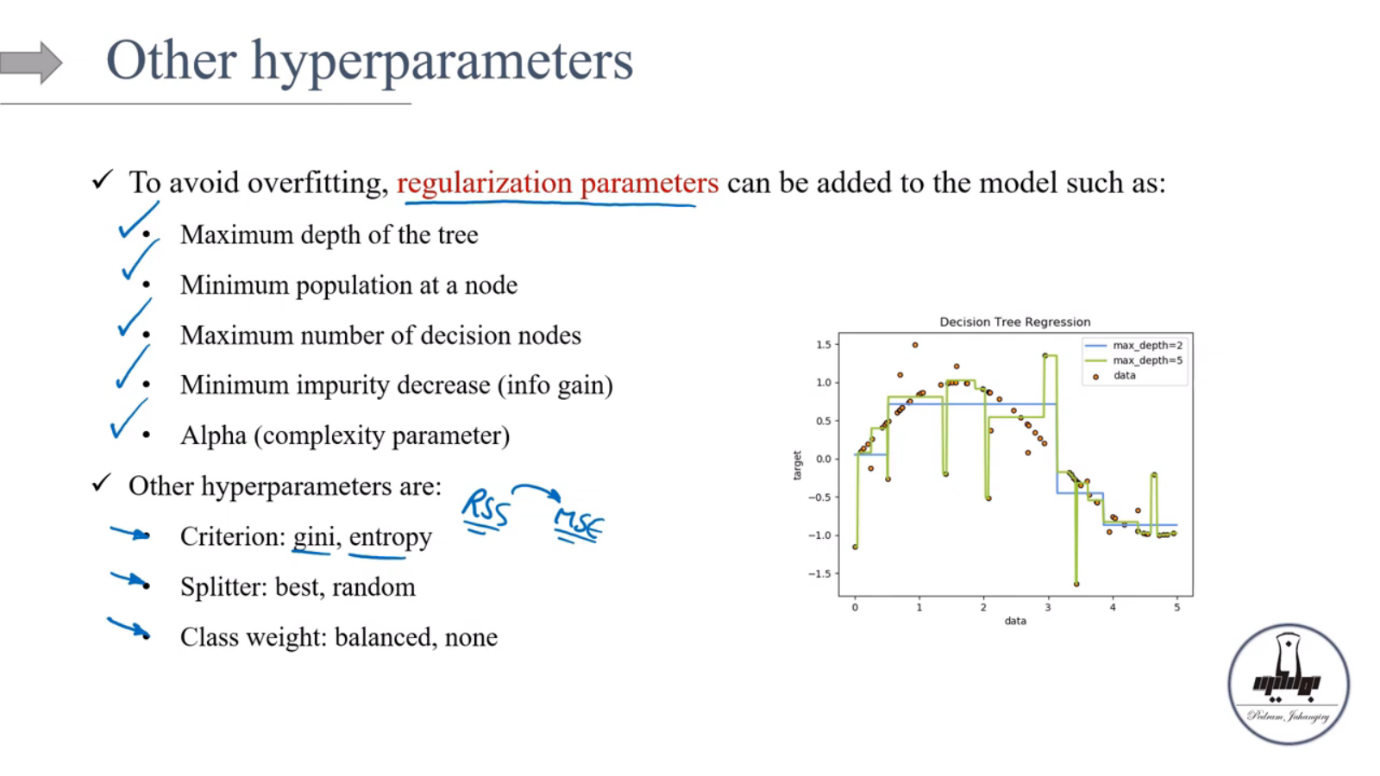
Tree score = SSR + Alpha\*T

SSR = sum of squared Residuals

T = Tree complexity penalty that is a function of number of leaves

Alpha = tuning parameter(non negative) found using cross validation

Alpha\*T= Tree complexity penalty compensates for the difference in the number of leaves

   C5.0 is another algorithm used for building decision trees, and it's an evolution of the earlier ID3 and C4.5 algorithms. Here's a breakdown of C5.0 and how it differs from CART:

**C5.0 Algorithm:**

* **Developed by:** J. Ross Quinlan
* **Purpose:** Primarily for classification tasks.
* **Key Characteristics:**
  + **Information Gain Ratio:** C5.0 uses the information gain ratio as its splitting criterion. This criterion addresses a bias in information gain towards features with many values.
  + **Rule Sets:** C5.0 can generate decision trees or rule sets. Rule sets are often easier to interpret than complex trees.
  + **Pruning:** C5.0 includes sophisticated pruning mechanisms to prevent overfitting.
  + **Handling Missing Values:** C5.0 can handle missing values in the data.
  + **Boosting:** C5.0 also includes boosting capabilities, meaning it can build multiple trees and combine their predictions.
  + **Computational Efficiency:** C5.0 is generally more computationally efficient than its predecessors.

**Differences Between C5.0 and CART:**

* **Splitting Criteria:**
  + C5.0 uses the information gain ratio.
  + CART uses the Gini impurity (for classification) or mean squared error (for regression).
* **Tree Structure:**
  + CART produces binary trees.
  + C5.0 can produce trees with more than two branches per node.
* **Output:**
  + C5.0 has the option of outputting rule sets, as well as decision trees.
  + CART primarily outputs decision trees.
* **Handling of Missing Values:**
  + C5.0 has build in methods for handing missing values.
  + CART also can handle missing values, but the implementations vary.
* **Primary Application:**
  + While CART can handle both classification and regression, C5.0 is mainly used for classification.

**In summary:**

* CART and C5.0 are both decision tree algorithms, but they differ in their splitting criteria, tree structure, and other features.
* C5.0 is typically considered more advanced than CART in certain ways, especially regarding handling missing values, and rule set generation, and the information gain ratio usage.
* Scikit-learn uses CART, while C5.0 is often found in proprietary software or specific data mining tools.