**Decision Tree Classifier**

All theoretical parts are same as ‘Decision Tree Regressor’ but only there is a small difference in how entropy is calculated with continuous numerical target and categorical target values

Entropy calculation for categorical target is very simple while entropy calculation for numerical target is a bit complex and performance intensive

In this technique we do not require to learn any mathematical formula calculation as it is simple nested if-else tree

* What is Entropy
* How to calculate Entropy
* Why we need gini while it works as entropy
* Handling numerical data instead of categorical data
* Decision tree hyper parameters
* Dtreeviz library to visualize tree

**What is a Decision Tree?**

A **decision tree** is a supervised learning algorithm that can be used for both classification and regression tasks. It works by recursively partitioning the data into subsets based on feature values, making decisions at each node to maximize a specific criterion (e.g., information gain or Gini index).

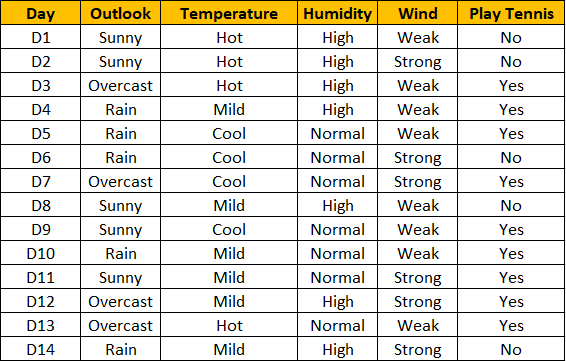
**Key Components:**

* **Root Node:** The top node in the tree that represents the best feature to split the data.
* **Internal Nodes:** Represent the features used for splitting the data based on specific decision rules.
* **Leaf Nodes:** Terminal nodes that represent the predicted outcome (class label or numerical value).
* **Branches:** Connections between nodes representing the possible values of the features.

**How Decision Trees Work?**

The process of creating a decision tree involves:

1. **Selecting the Best Attribute**: Using a metric like Gini impurity, entropy, or information gain, the best attribute to split the data is selected.
2. **Splitting the Dataset**: The dataset is split into subsets based on the selected attribute.
3. **Repeating the Process**: The process is repeated recursively for each subset, creating a new internal node or leaf node until a stopping criterion is met (e.g., all instances in a node belong to the same class or a predefined depth is reached).



Dataset where play tennis is the output feature and rest all are input features

Zoom image will be displayed



Decision tree example where the root node chosen is outlook (random)

A **leaf node** in a decision tree is the terminal node at the bottom of the tree, where no further splits are made. Leaf nodes represent the final output or prediction of the decision tree. Once a data point reaches a leaf node, a decision or prediction is made based on the majority class (for classification) or the average value (for regression) of the data points that reach that leaf.

To check **mathematically**if any split is**pure split**or not we use**entropy or gini impurity. Information Gain**helps us to determine which features need to be selected

**Decision Tree algorithm works in simpler steps**:

* **Starting at the Root**: The algorithm begins at the top, called the “root node,” representing the entire dataset.
* **Asking the Best Questions:** It looks for the most important feature or question that splits the data into the most distinct groups. This is like asking a question at a fork in the tree.
* **Branching Out**: Based on the answer to that question, it divides the data into smaller subsets, creating new branches. Each branch represents a possible route through the tree.
* **Repeating the Process**: The algorithm continues asking questions and splitting the data at each branch until it reaches the final “leaf nodes,” representing the predicted outcomes or classifications.

Types of Decision Tree

* **ID3**: This algorithm measures how mixed up the data is at a node using something called [**entropy**](https://www.analyticsvidhya.com/blog/2020/11/entropy-a-key-concept-for-all-data-science-beginners/). It then chooses the feature that helps to clarify the data the most.**C4.5**: This is an improved version of ID3 that can handle missing data and continuous attributes.
* **CART**: This algorithm uses a different measure called Gini impurity to decide how to split the data. It can be used for both classification (sorting data into categories) and regression (predicting continuous values) tasks.

**Feature Independence**

These trees often assume that the features used for splitting nodes are independent. In practice, feature independence may not hold, but it can still perform well if features are correlated.

**Homogeneity**

It aim to create homogeneous subgroups in each node, meaning that the samples within a node are as similar as possible regarding the target variable. This assumption helps in achieving clear decision boundaries.

**Top-Down Greedy Approach**

They are constructed using a top-down, greedy approach, where each split is chosen to maximize information gain or minimize impurity at the current node. This may not always result in the globally optimal tree.

**Advantages of Decision Trees**

* **Easy to Understand:** They are simple to visualize and interpret, making them easy to understand even for non-experts.
* **Handles Both Numerical and Categorical Data:** They can work with both types of data without needing much preprocessing.
* **No Need for Data Scaling:** These trees do not require normalization or scaling of data.
* **Automated Feature Selection:** They automatically identify the most important features for decision-making.
* **Handles Non-Linear Relationships:** They can capture non-linear patterns in the data effectively.

**Disadvantages of Decision Trees**

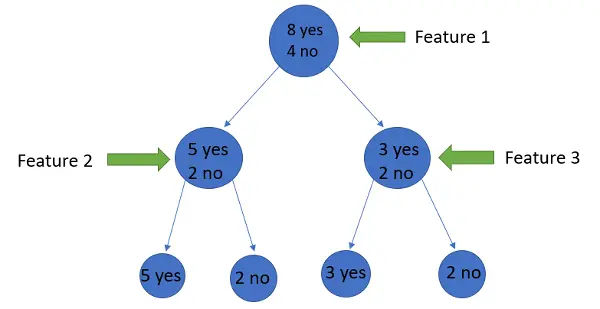
* **Overfitting Risk:** It can easily overfit the training data, especially if they are too deep.
* **Unstable with Small Changes:** Small changes in data can lead to completely different trees.
* **Biased with Imbalanced Data:** They tend to be biased if one class dominates the dataset.
* **Limited to Axis-Parallel Splits:** They struggle with diagonal or complex decision boundaries.
* **Can Become Complex:** Large trees can become hard to interpret and may lose their simplicity.

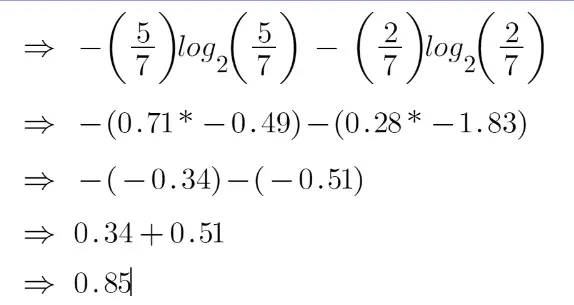
**How do Decision Trees use Entropy?**

**Decision Tree used Entropy in the given Points:**

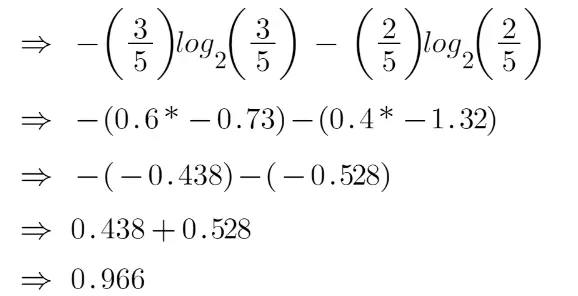
* Now we know what entropy is and what is its formula, Next, we need to know that how exactly does it work in this algorithm.
* Entropy basically measures the impurity of a node. Impurity is the degree of randomness; it tells how random our data is. A**pure sub-split** means that either you should be getting “yes”, or you should be getting “no”.
* Suppose a *feature* has 8 “yes” and 4 “no” initially, after the first split the left node*gets 5 ‘yes’ and 2 ‘no’* whereas right node*gets 3 ‘yes’ and 2 ‘no’.*
* We see here the split is not pure, why? Because we can still see some negative classes in both the nodes. In order to make a this tree, we need to calculate the impurity of each split, and when the purity is 100%, we make it as a leaf node.

To check the impurity of feature 2 and feature 3 we will take the help for Entropy formula.





For feature 3,



* We can clearly see from the tree itself that left node has low entropy or more purity than right node since left node has a greater number of “yes” and it is easy to decide here.
* Always remember that the higher the Entropy, the lower will be the purity and the higher will be the impurity.
* As mentioned earlier the goal of machine learning is to decrease the uncertainty or impurity in the dataset, here by using the entropy we are getting the impurity of a particular node, we don’t know if the parent entropy or the entropy of a particular node has decreased or not.
* For this, we bring a new metric called “Information gain” which tells us how much the parent entropy has decreased after splitting it with some feature.

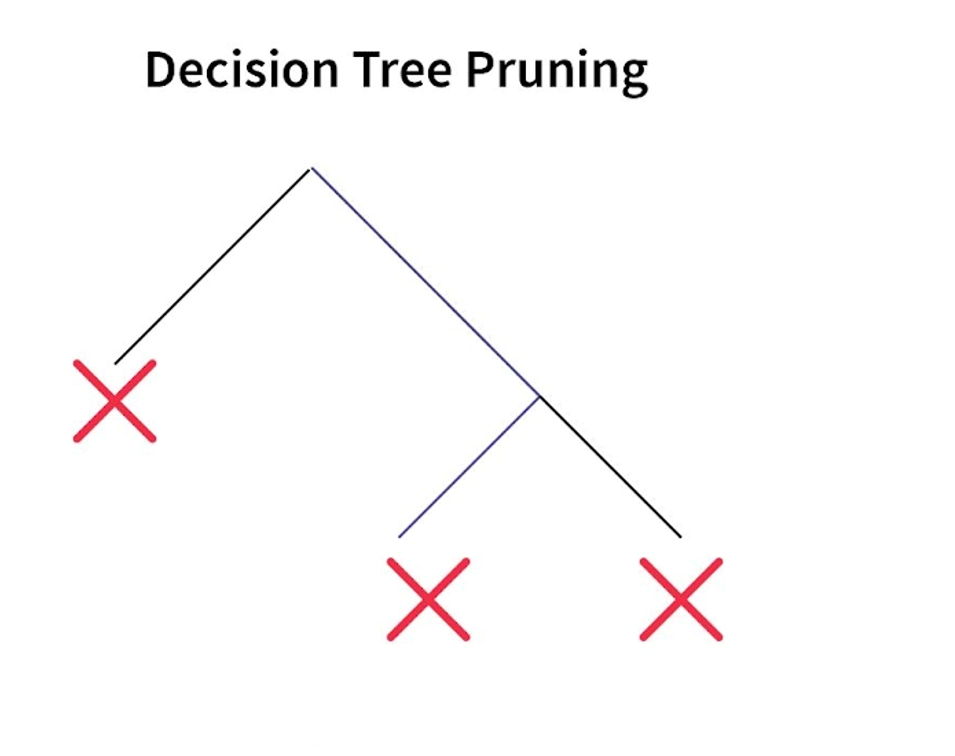
**Why Use a Decision Tree in Machine Learning?**

Decision trees offer several unique advantages that make them a popular choice in machine learning:

* **Interpretability**: Decision trees are highly interpretable. The tree structure provides a clear visual representation of the decision-making process, making it easy for both technical and non-technical stakeholders to understand.
* **No Data Preprocessing Required**: Decision trees can handle data without the need for normalization or scaling, unlike other algorithms like **SVM** or **neural networks**.
* **Handling of Both Categorical and Numerical Data**: Decision trees can work with both types of data, making them versatile for different problem types.
* **Non-linear Relationships**: Decision trees can model non-linear relationships between features and the target variable, enabling more complex decision-making.

**Pruning Decision Trees**

One major challenge with decision trees is their tendency to **overfit**, especially when the tree becomes too deep. To address this, **pruning** is used to simplify the tree and improve its generalization to unseen data.



**Pre-pruning**

Pre-pruning involves stopping the tree-building process before it becomes too complex. This is done by setting conditions such as limiting the tree’s maximum depth or requiring a minimum number of instances for a split. Pre-pruning helps avoid overfitting early on.

**Post-pruning**

Post-pruning occurs after the tree has been fully built. In this method, branches that do not improve the model’s performance on a validation set are removed. This results in a smaller, more efficient tree that performs better on unseen data.

Pruning is an essential technique for achieving the right balance between model complexity and accuracy, ensuring that the decision tree performs well on both training and test data.

**Working on Decision Tree Algorithm**

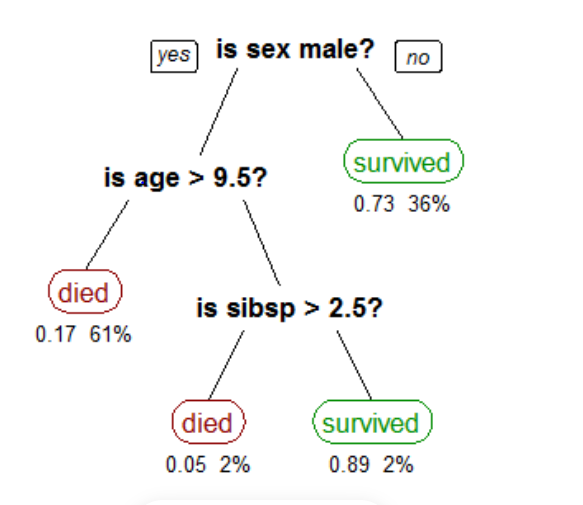
In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree, and the decision tree algorithm compares the values of the root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node. The algorithm again compares the attribute value with the other sub-nodes for the next node and moves further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

**Step-1: Select** the root node based on the information gained value from the complete dataset.

**Step-2:** Divide the root node into sub-nodes based on information gain and entropy values.

**Step-3:** Continue this process till we cannot further classify the node into sub-nodes called leaf nodes.

An elementary example uses titanic data set for predicting whether a passenger will survive or not survive. I have used only three Columns/attributes/features for this example—namely, Sex, age, and sibs(number of spouses or children).



In the above figure, is sex male a root node? It will divide into two sub-nodes based on condition(yes or no). Is age>9.5?, is branch node and survived is Leaf node. Is sibsp>2.5?, is also a branch node, and died is a Leaf node. Both died and survived are Leaf nodes; there is no chance to split. In real datasets, there are more columns/attributes/features.

Implementing the Decision Tree Algorithm

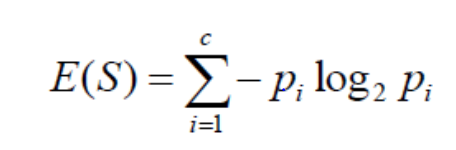
While implementing the decision tree algorithm, everyone will doubt how to select the Root node and sub-nodes. We have a technique called ASM(Attributes Selection Measures). In this technique, there are two methods:

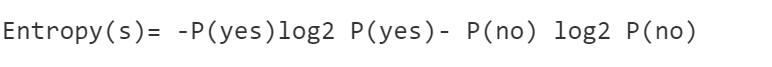
1. Information Gain:

2. Gini Index

Information Gain

**entropy:** It is the sum of the probability of each label times, the log probability of that same label. It is an average rate at which a stochastic data source produces information, Or it measures the uncertainty associated with a random variable.





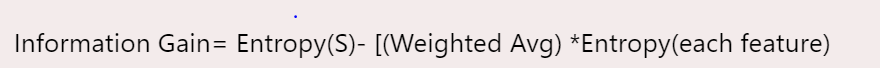
Where S=Total number of samples

            P(yes)=Probability of Yes

            P(No)=Probability of  No

**information Gain:**An amount of information gained about a random variable or signal from observing another random variable.

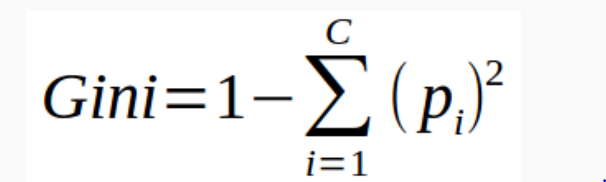
It favours smaller partitions with distinct values.



**Gini Index**

It is calculated by subtracting the sum of the squared probabilities of each class from one.

It favours larger partitions.



Decision Tree Algorithm: Pruning

In the tree algorithm, the Pruning concept will play a significant role. We may get the model overfitting issue when a model builds on a large dataset. For reducing this issue, Pruning will help.

Pruning is the process to stop the splitting of nodes into sub-nodes.

**In this, there are two types:**

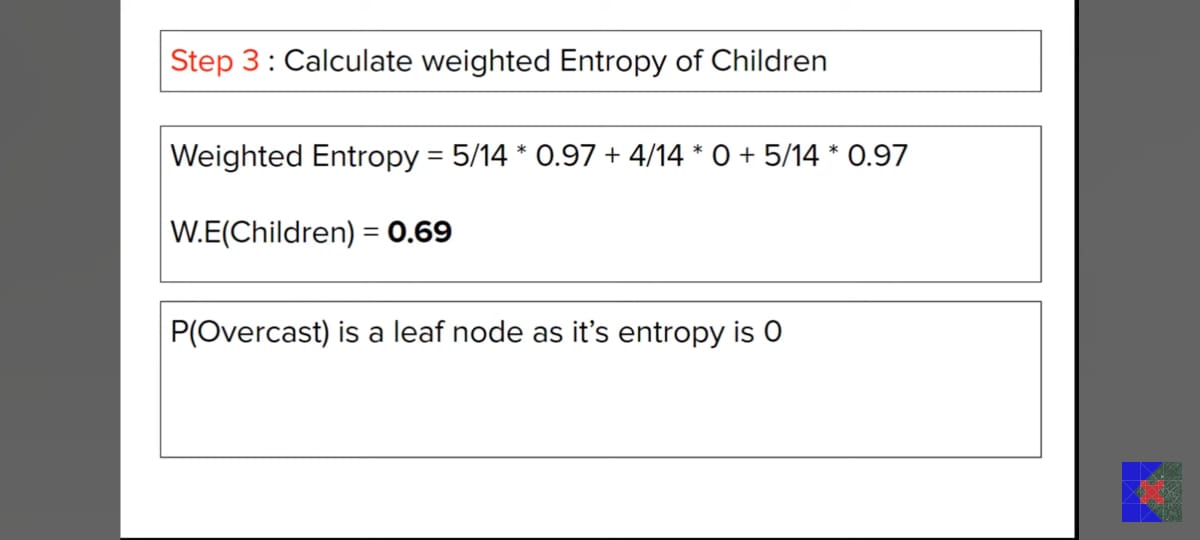
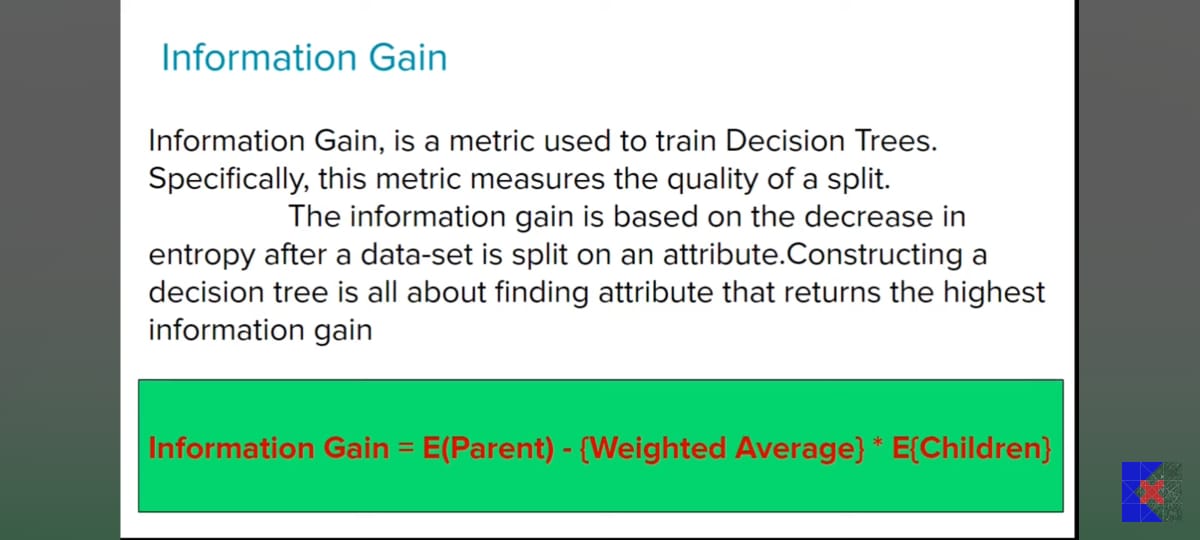
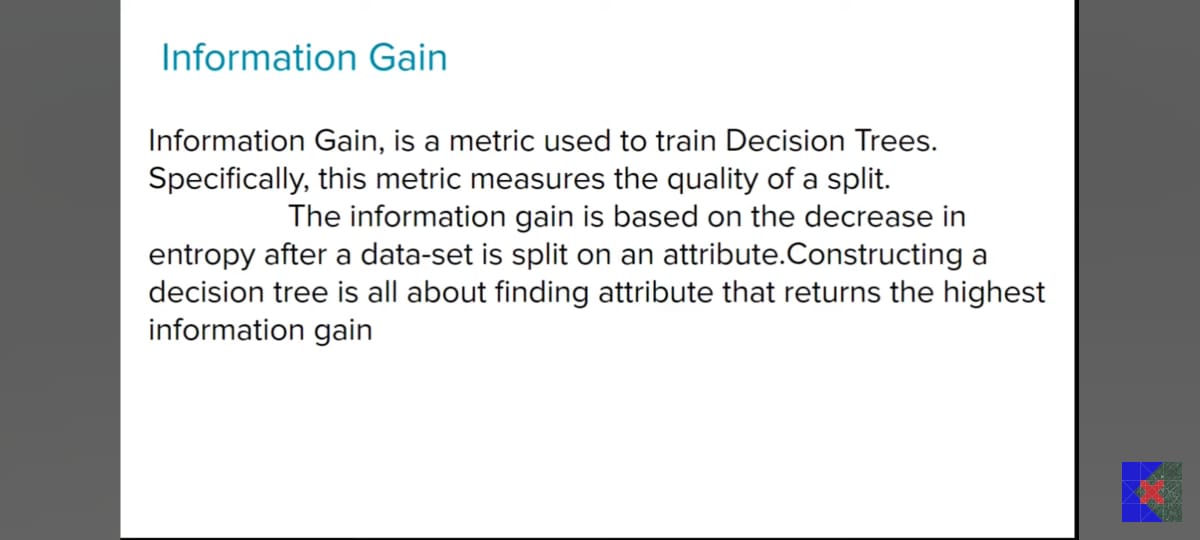
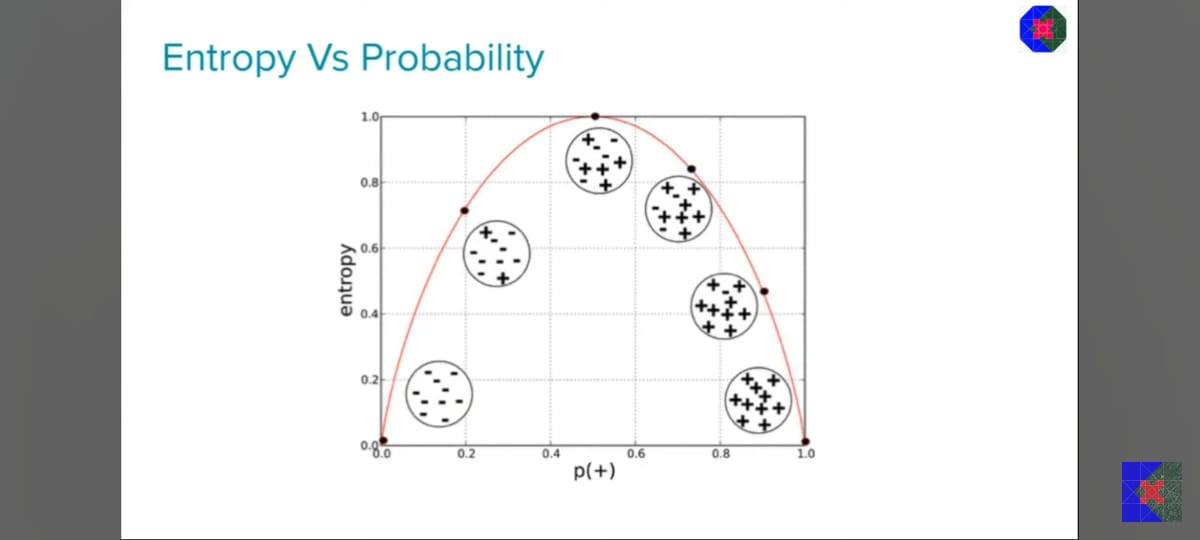
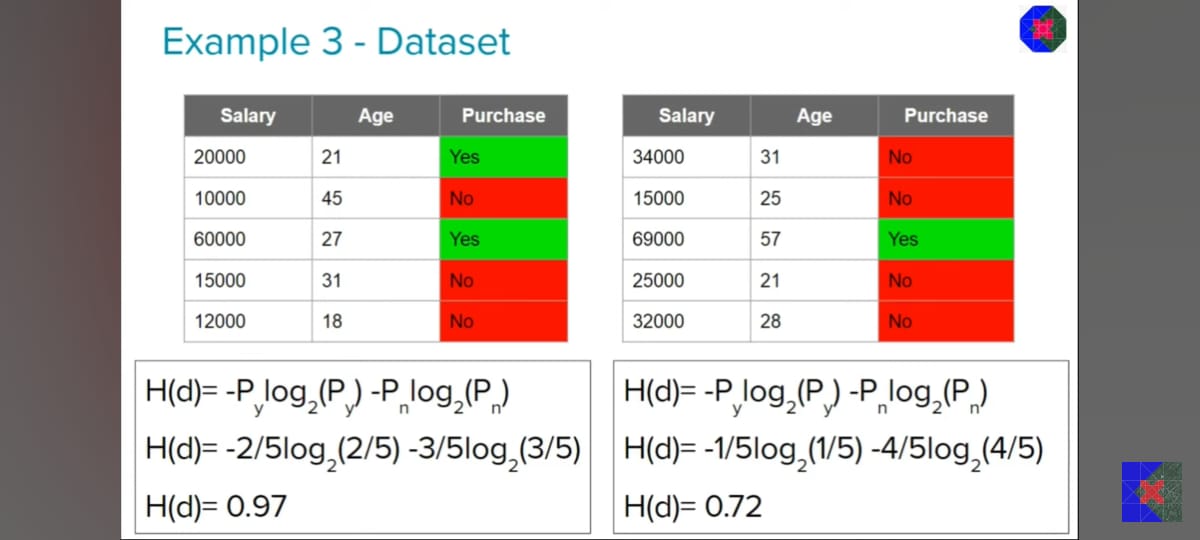
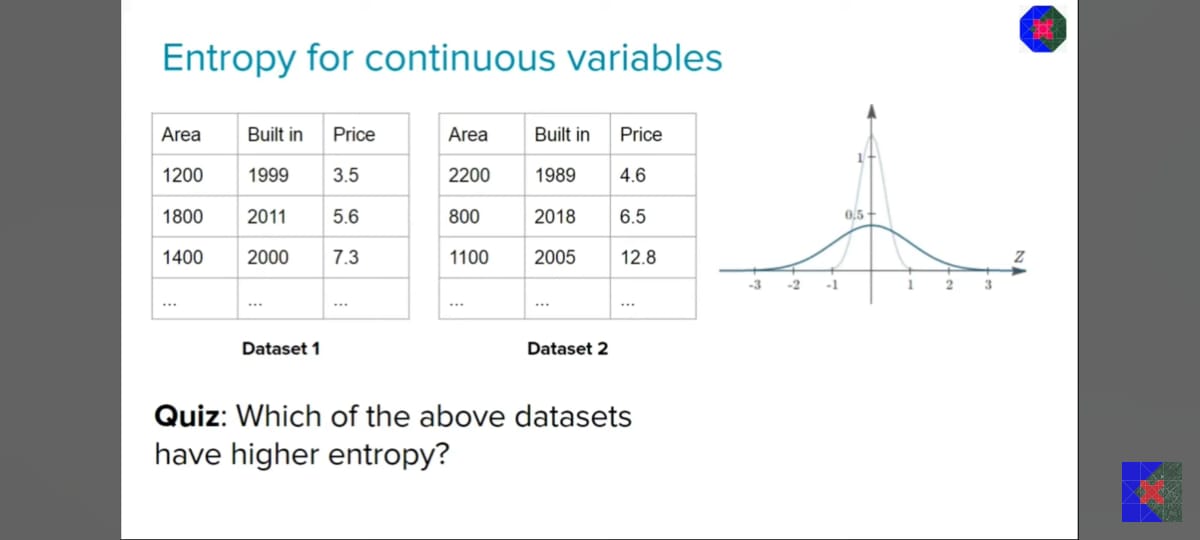
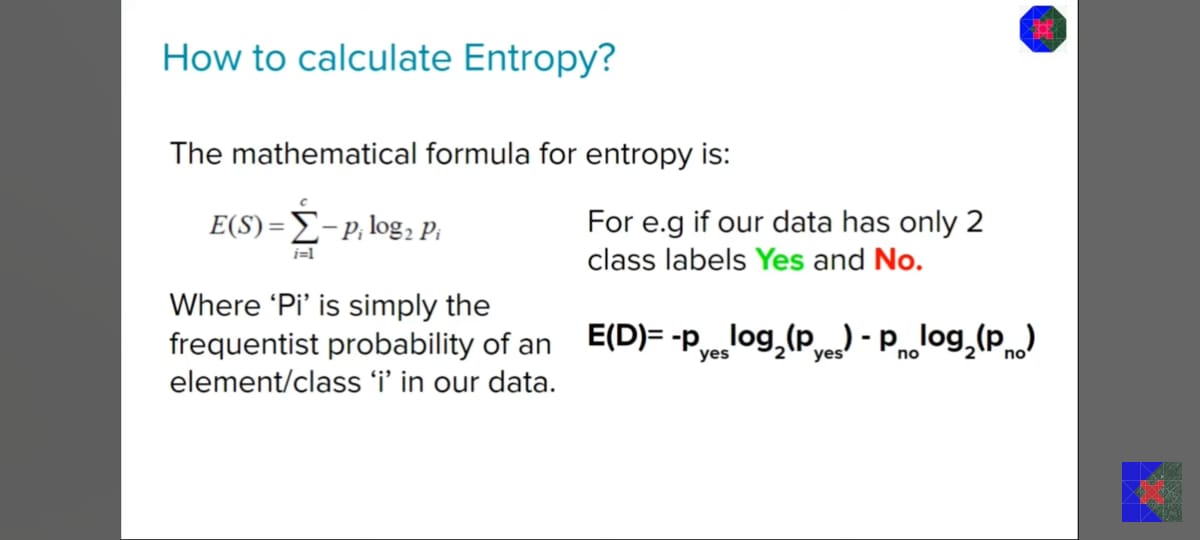
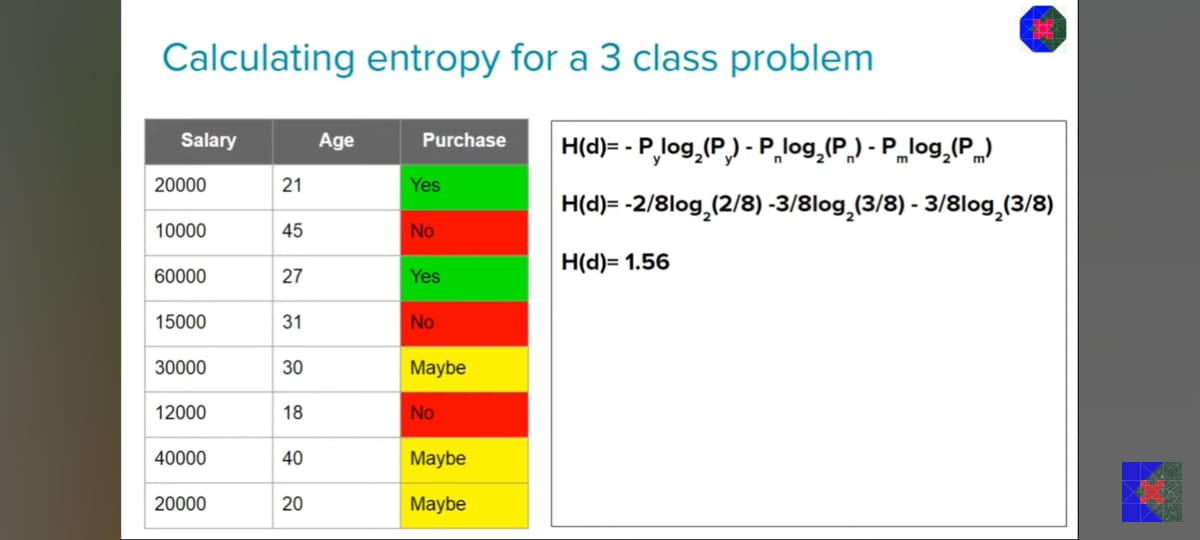
1. Cost Complexity Pruning

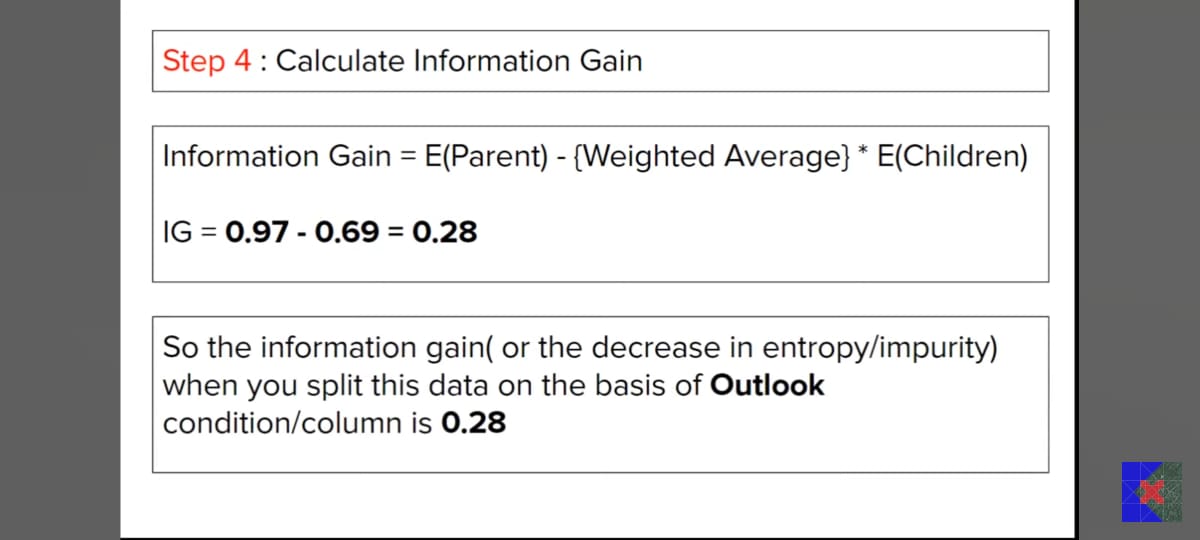
2. Reduced Error Pruning

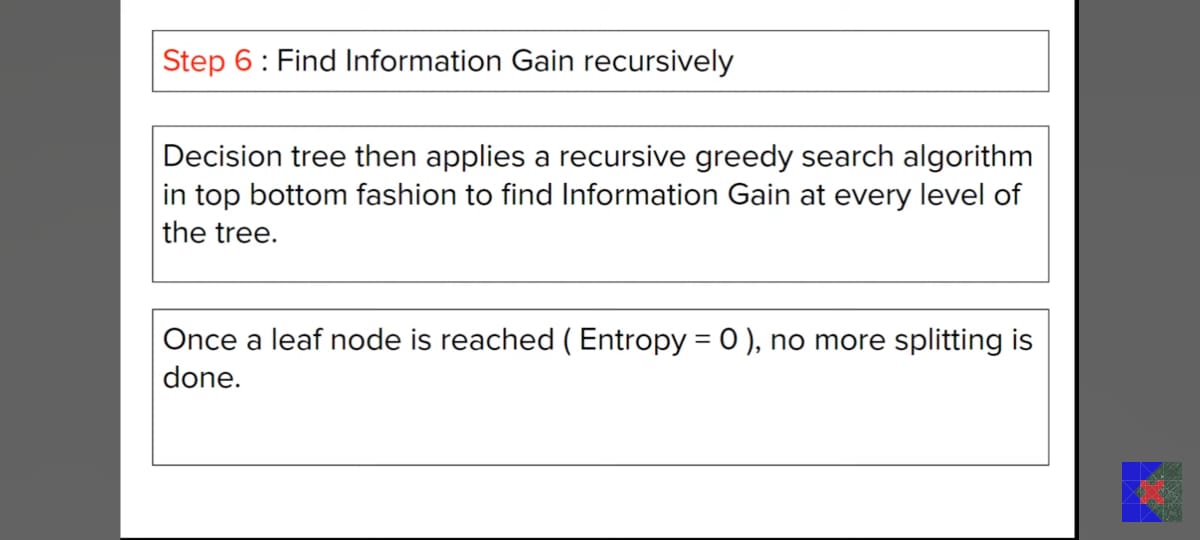
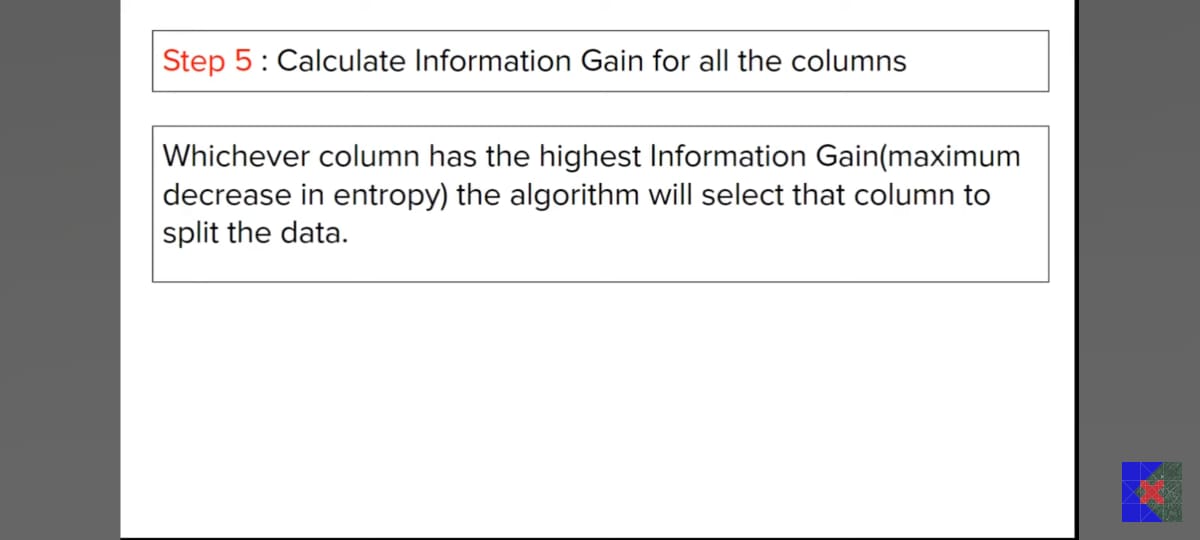
Implementation of Decision Tree Algorithm

For building a model, we need to preprocess the data, Transform the data, split the data into train and test, and then make the model.

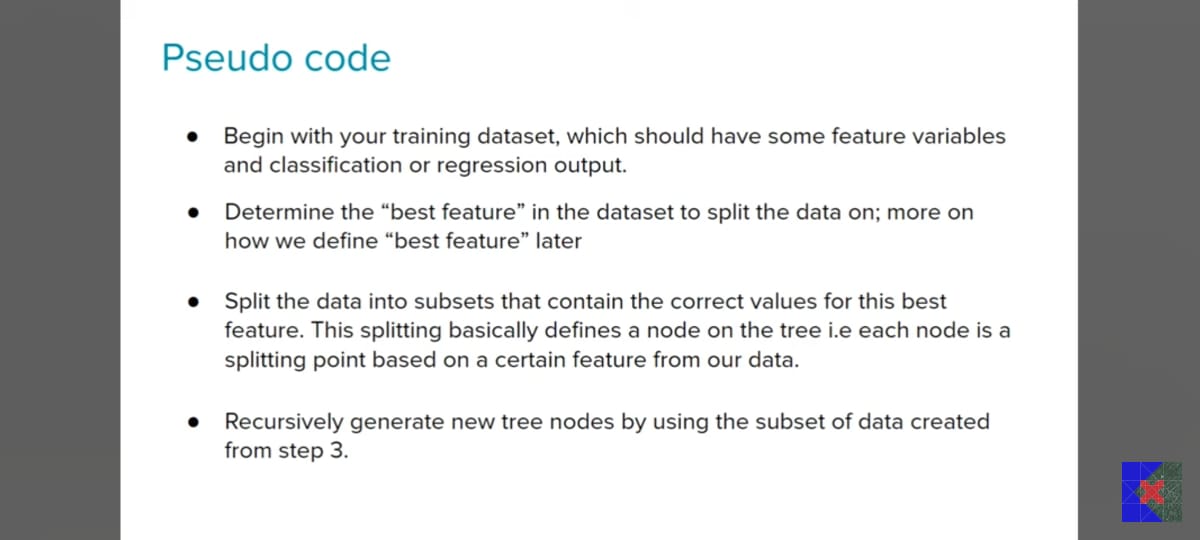
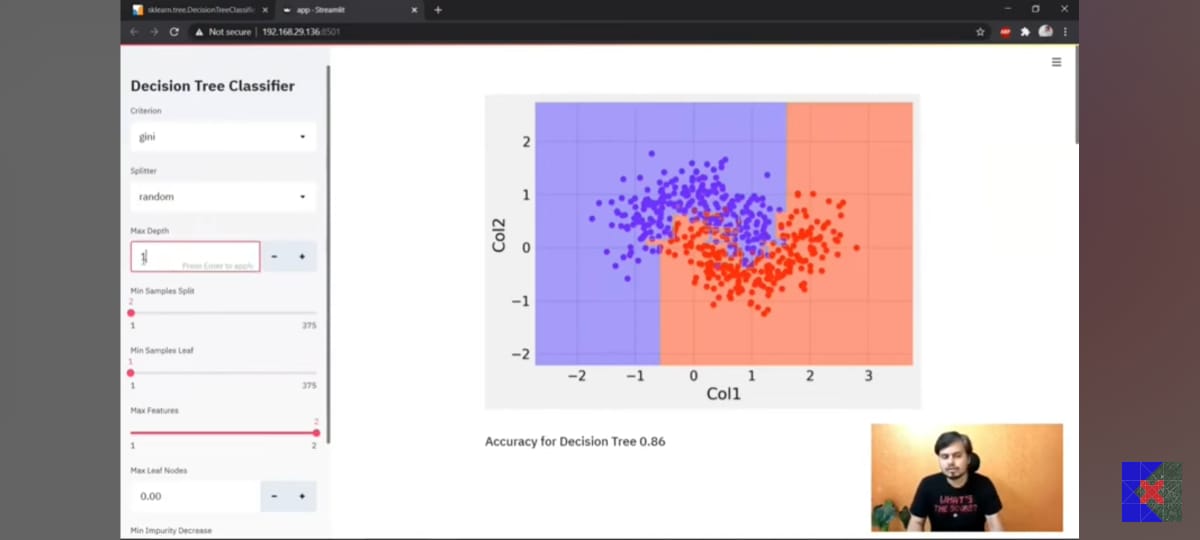
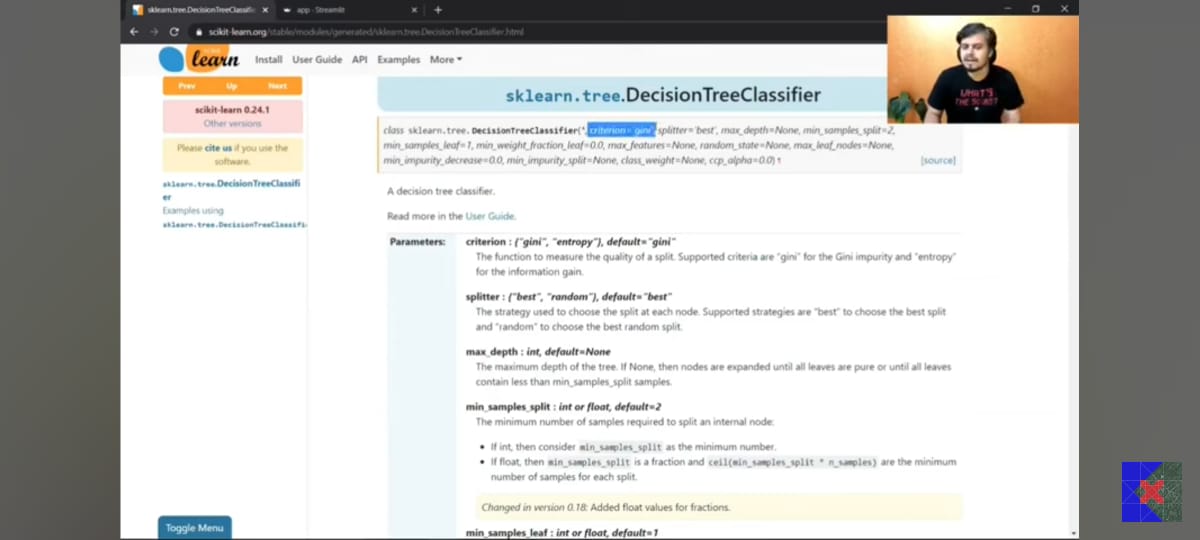
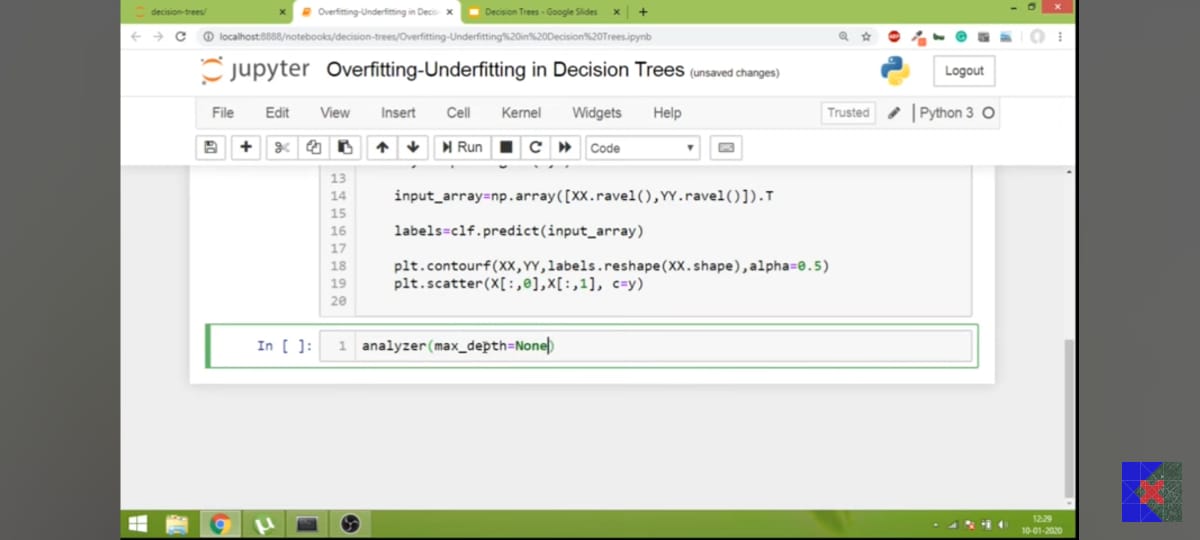
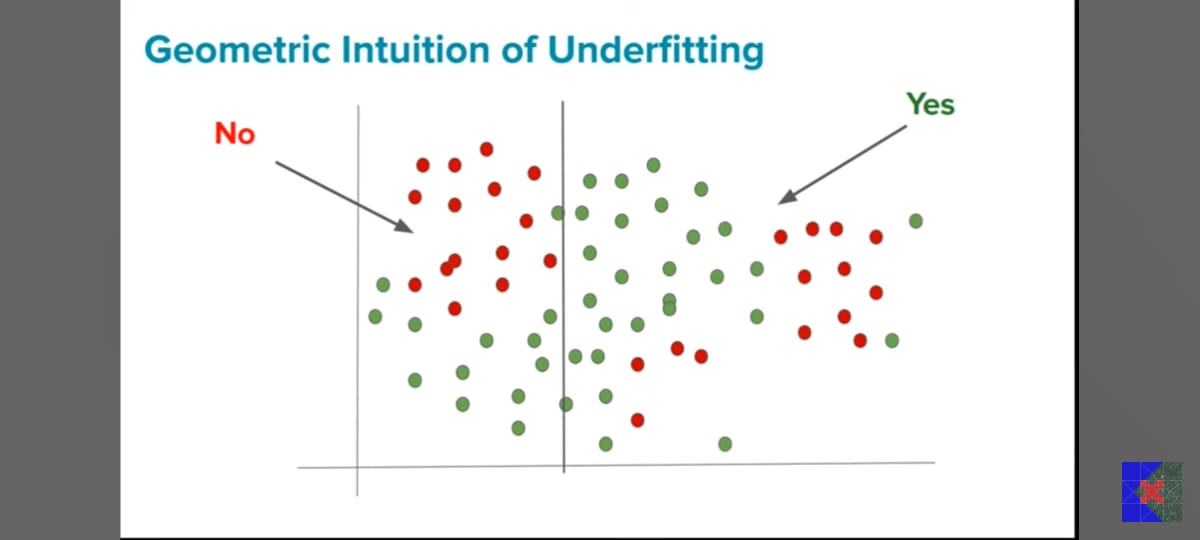
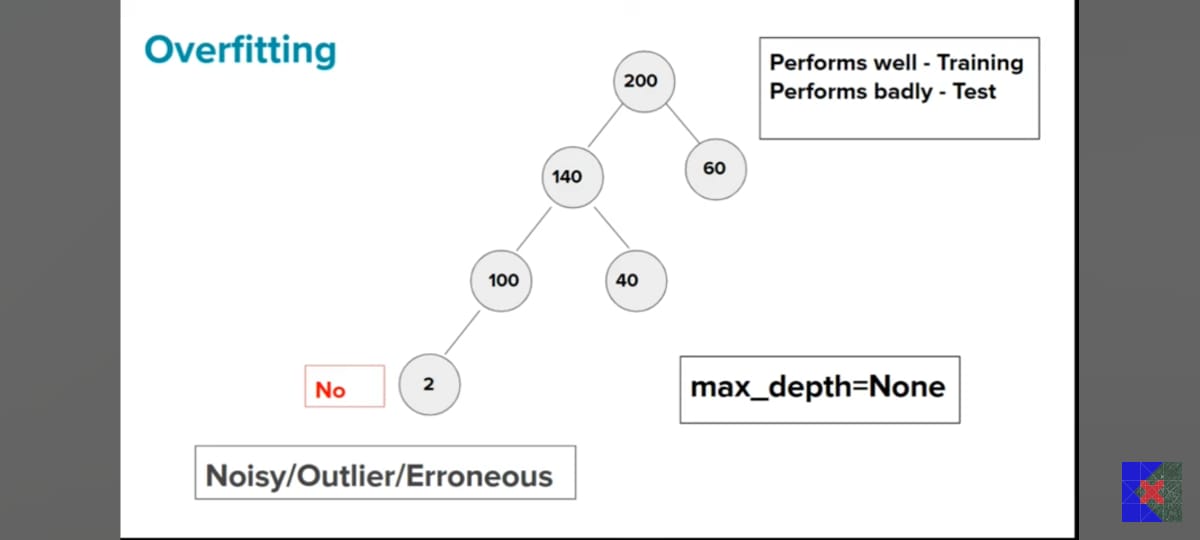
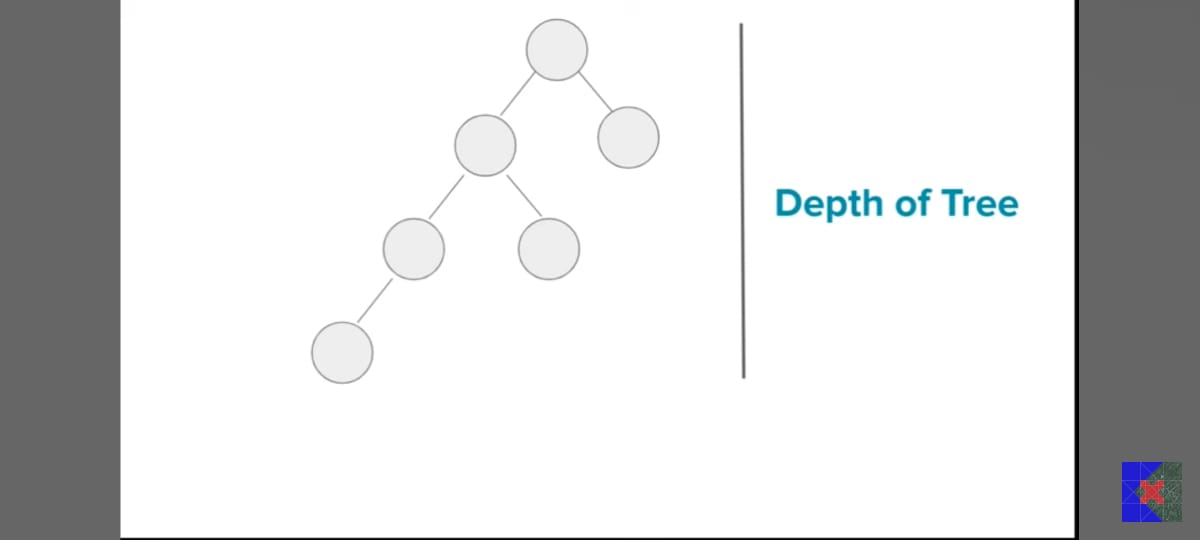
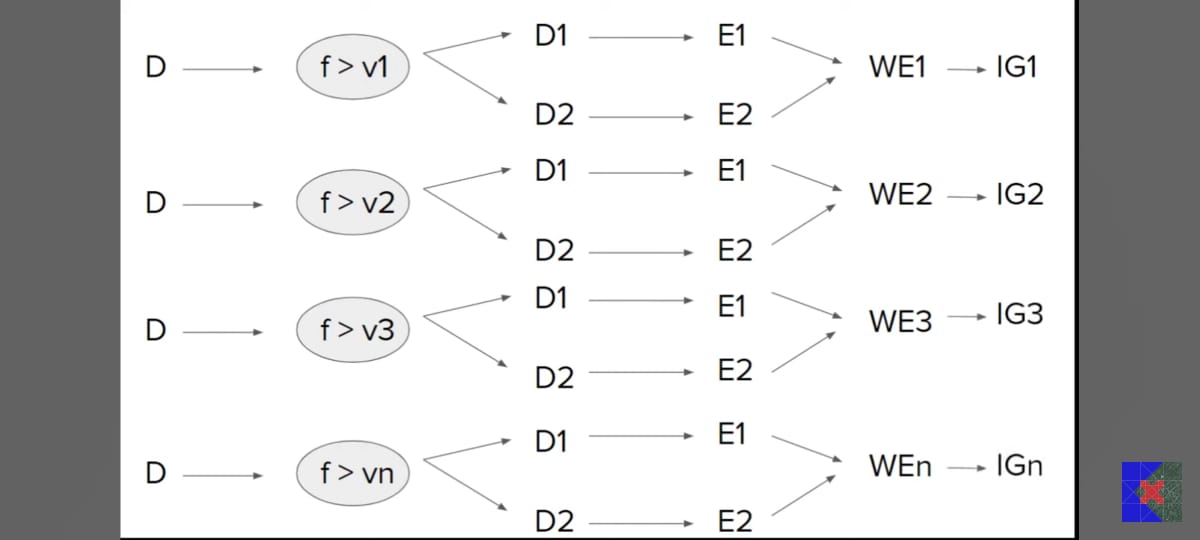
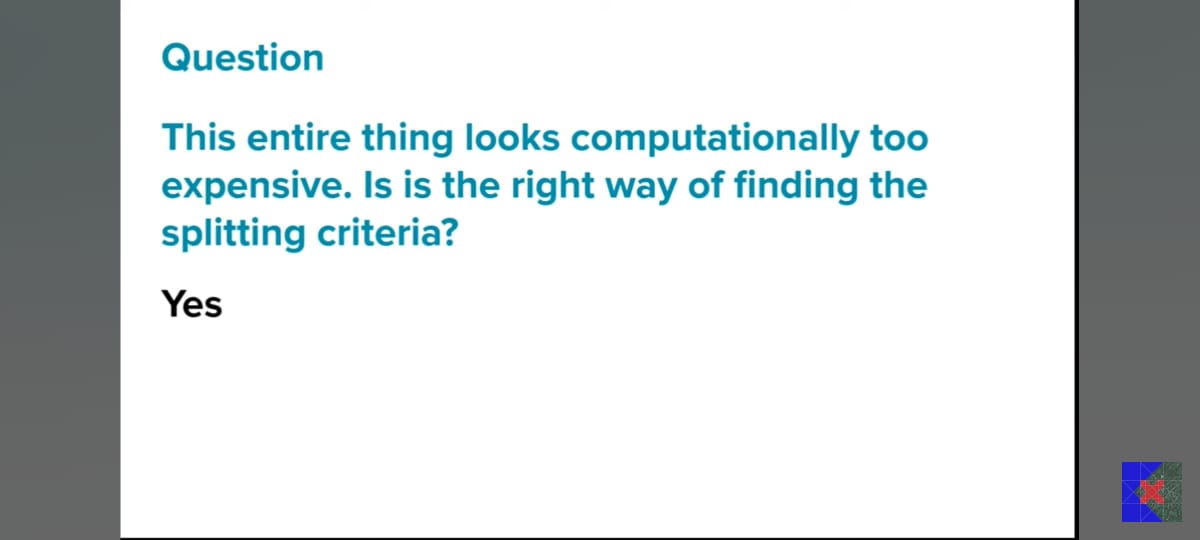
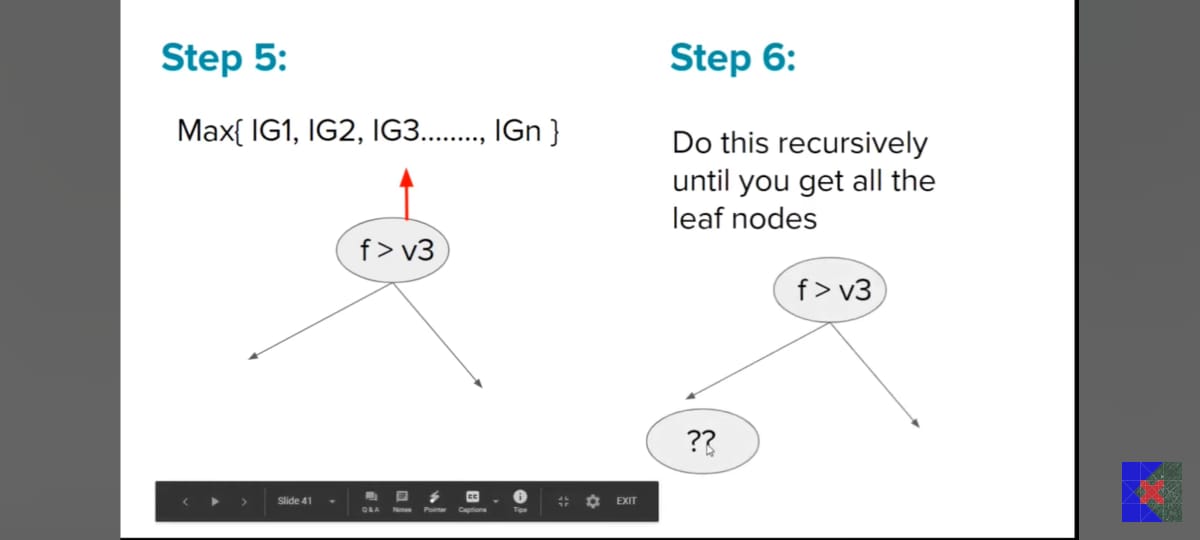
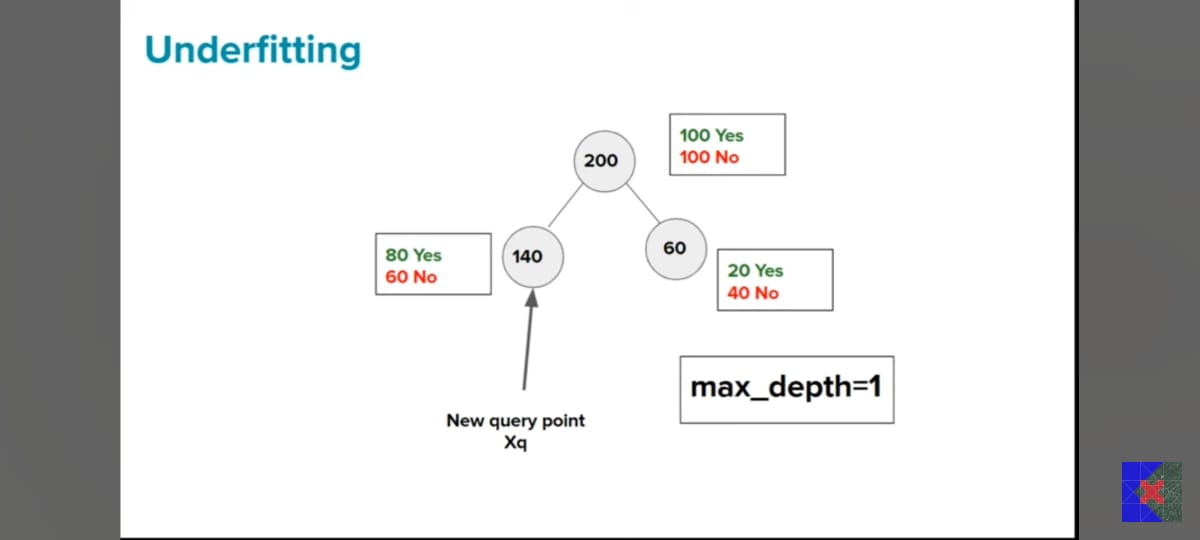
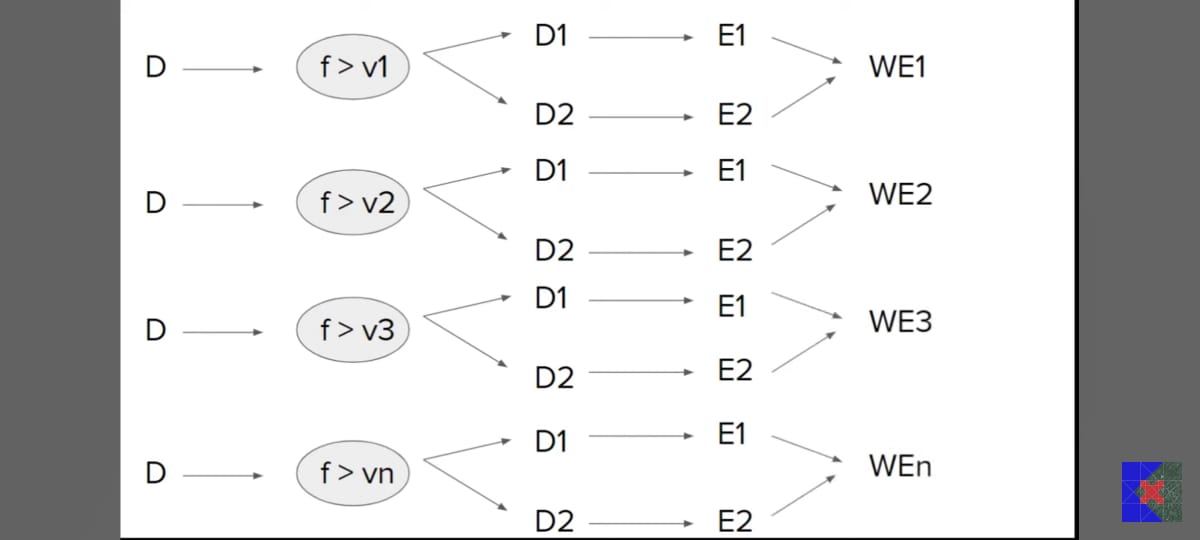
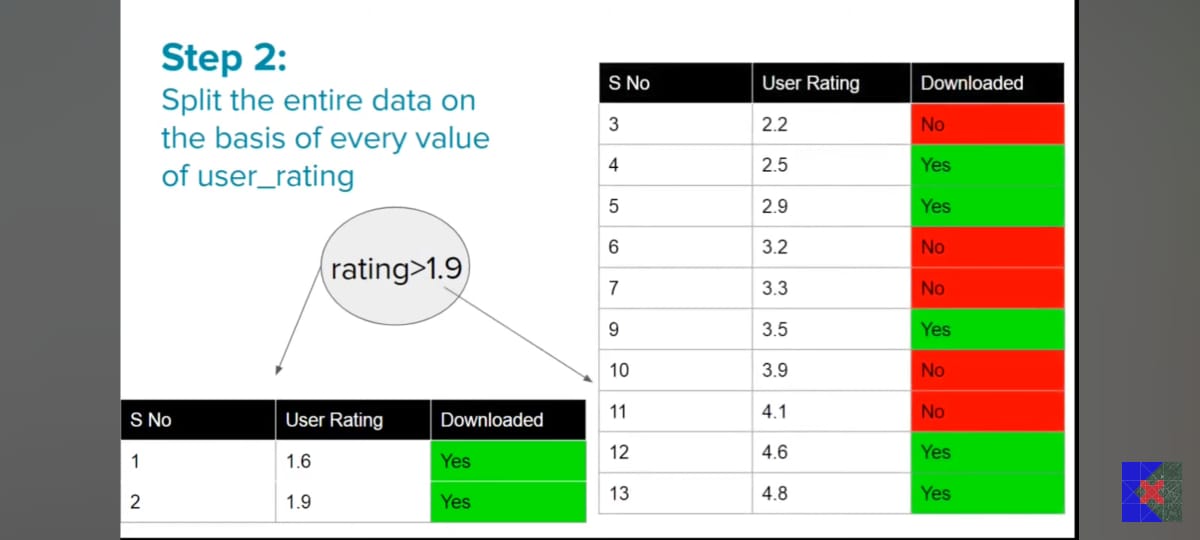
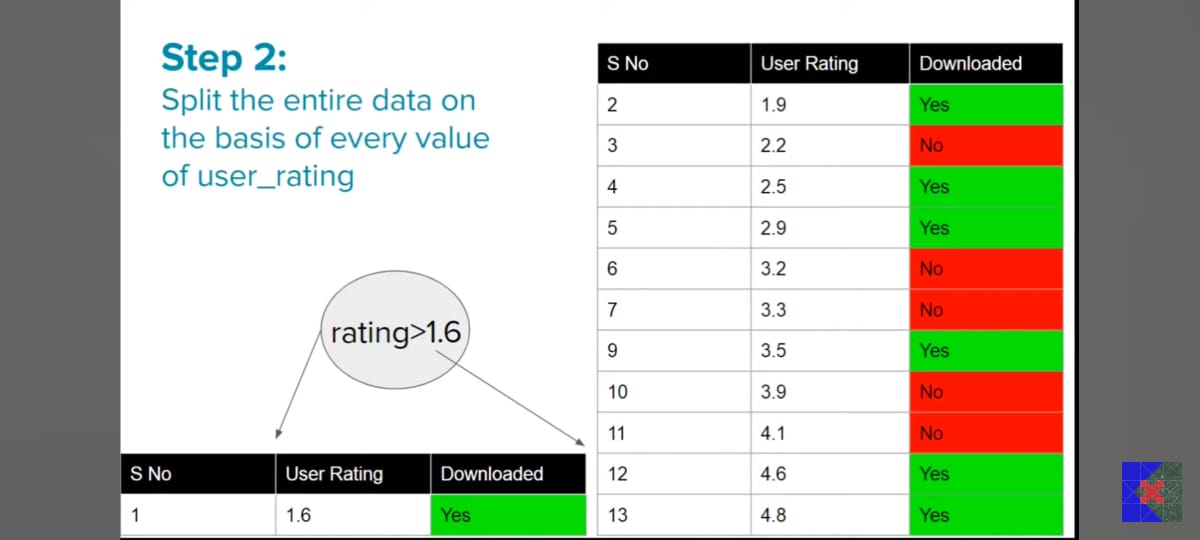
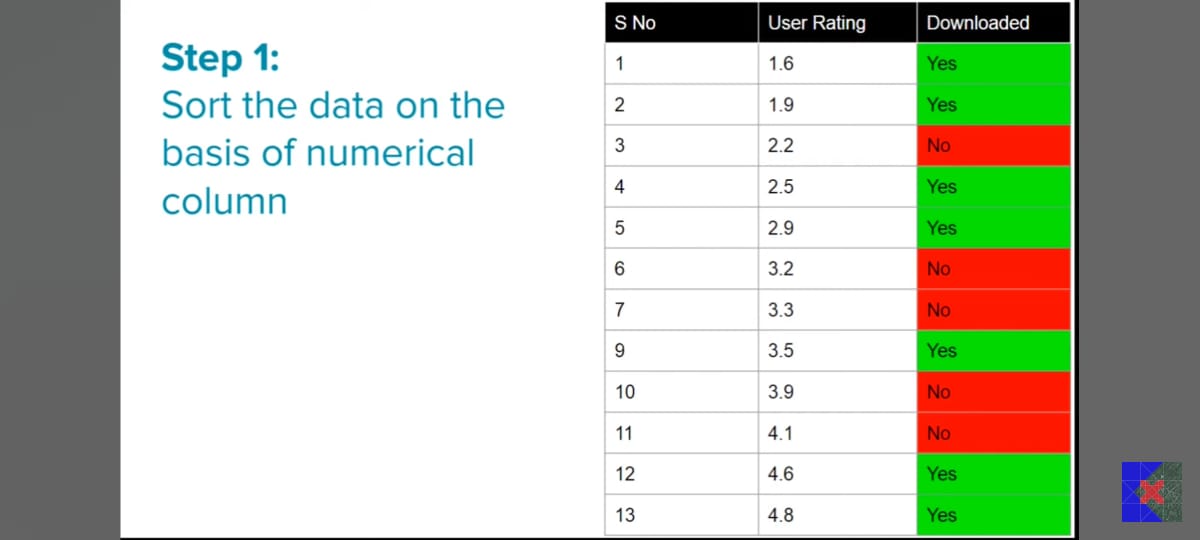
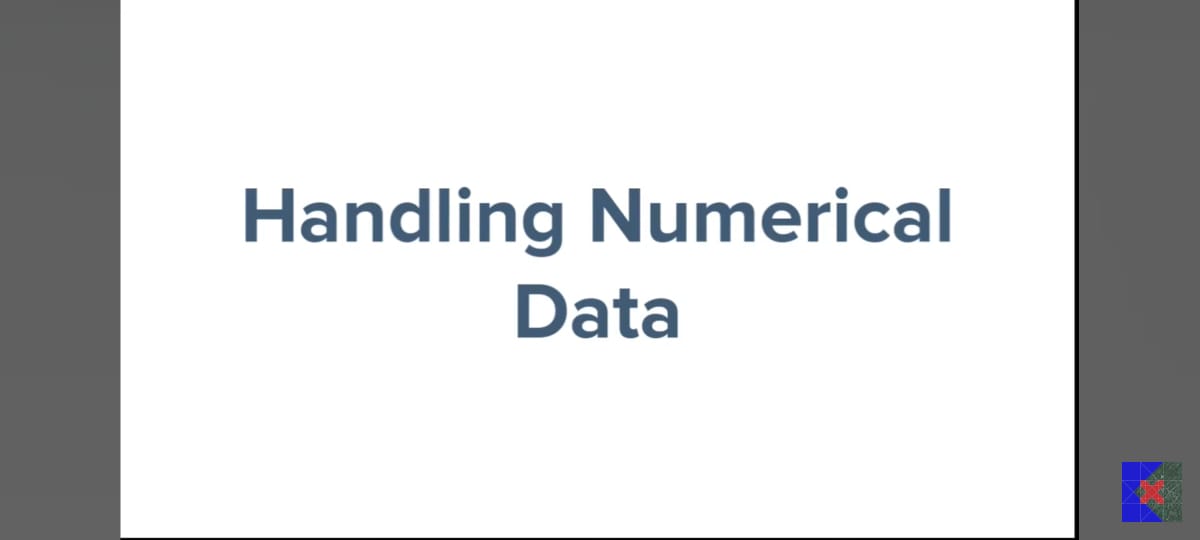
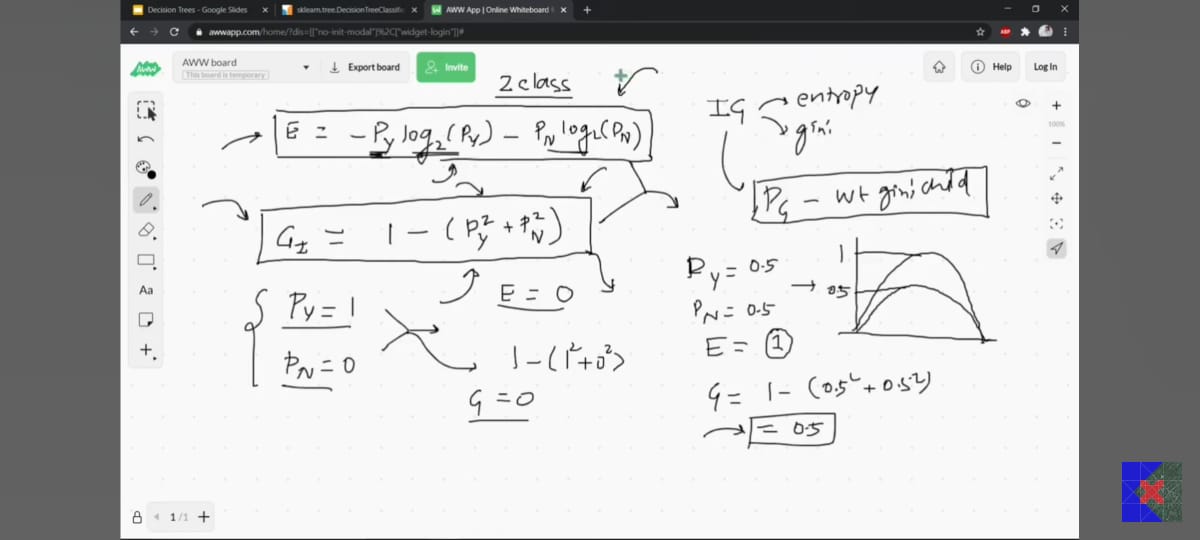
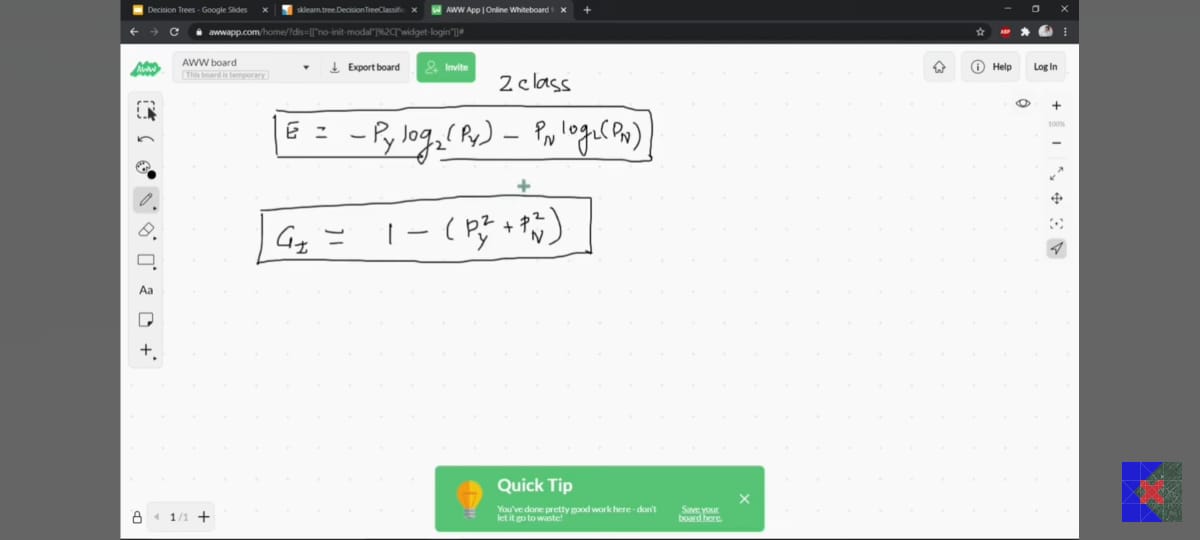
Entropy Calculation:







**Decision Trees Gini Impurity:**



**Full Derivation of decision tree:**

**Step-by-Step Decision Tree Using Information Gain (ID3 Algorithm)**

**🎯 Goal:**

Classify whether to **Play Tennis** based on features:

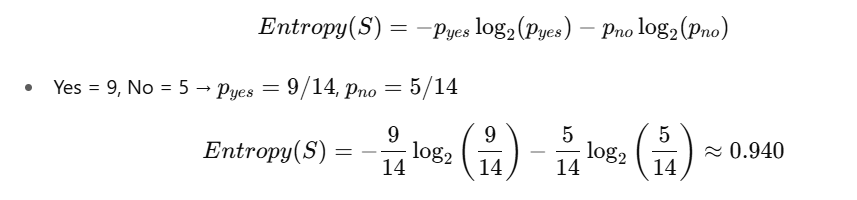
* **Outlook** (Sunny, Overcast, Rain)
* **Temperature** (Hot, Mild, Cool)
* **Humidity** (High, Normal)
* **Windy** (True, False)

**🧮 Sample Dataset (14 rows)**

| **ID** | **Outlook** | **Temp** | **Humidity** | **Windy** | **PlayTennis** |
| --- | --- | --- | --- | --- | --- |
| 1 | Sunny | Hot | High | False | No |
| 2 | Sunny | Hot | High | True | No |
| 3 | Overcast | Hot | High | False | Yes |
| 4 | Rain | Mild | High | False | Yes |
| 5 | Rain | Cool | Normal | False | Yes |
| 6 | Rain | Cool | Normal | True | No |
| 7 | Overcast | Cool | Normal | True | Yes |
| 8 | Sunny | Mild | High | False | No |
| 9 | Sunny | Cool | Normal | False | Yes |
| 10 | Rain | Mild | Normal | False | Yes |
| 11 | Sunny | Mild | Normal | True | Yes |
| 12 | Overcast | Mild | High | True | Yes |
| 13 | Overcast | Hot | Normal | False | Yes |
| 14 | Rain | Mild | High | True | No |

**🪜 Step 1: Calculate Entropy of the Dataset (Level 0)**

We compute the **Entropy of the root node** (target: PlayTennis)



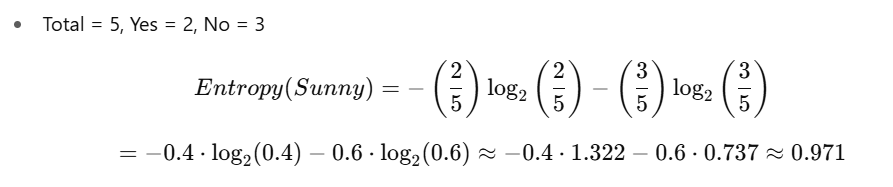
**🪜 Step 2: Choose the Best Feature to Split (Level 1)**

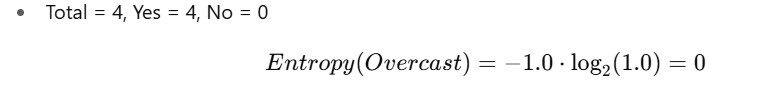
We calculate **Information Gain** for each feature. Let's compute for **Outlook**:

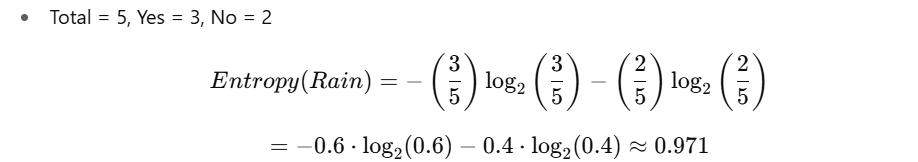
**Feature: Outlook (Sunny, Overcast, Rain)**

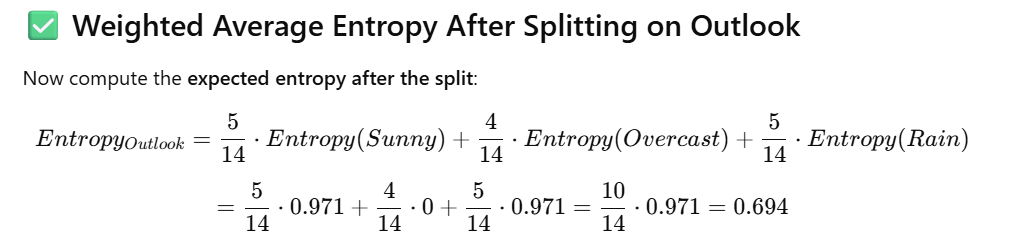
| **Outlook** | **Count** | **Yes** | **No** | **Entropy** |
| --- | --- | --- | --- | --- |
| Sunny | 5 | 2 | 3 | 0.971 |
| Overcast | 4 | 4 | 0 | 0.0 |
| Rain | 5 | 3 | 2 | 0.971 |

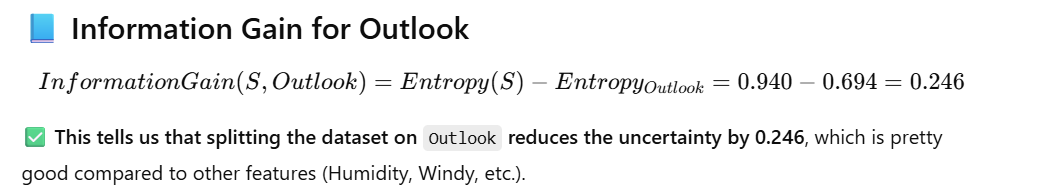
Weighted avg entropy:











Similarly, calculate for Temp, Humidity, Windy. (I'll skip full math here for brevity.)

Assume **Outlook** gives highest gain ⇒ select as **root node**.

**🌳 Tree after Level 1 (Height 1)**

csharp

[Outlook]

/ | \

Sunny Overcast Rain

**🪜 Step 3: Split Remaining Nodes (Height 2)**

**Branch: Outlook = Sunny → Subset = 5 rows**

| **ID** | **Temp** | **Humidity** | **Windy** | **PlayTennis** |
| --- | --- | --- | --- | --- |
| 1 | Hot | High | False | No |
| 2 | Hot | High | True | No |
| 8 | Mild | High | False | No |
| 9 | Cool | Normal | False | Yes |
| 11 | Mild | Normal | True | Yes |

Entropy(Sunny subset) = 0.971

Try features: **Humidity**, **Windy**, etc.

**Split by Humidity:**

| **Humidity** | **Count** | **Yes** | **No** | **Entropy** |
| --- | --- | --- | --- | --- |
| High | 3 | 0 | 3 | 0.0 |
| Normal | 2 | 2 | 0 | 0.0 |

Weighted entropy = 0



✅ Best split: Humidity

**🌳 Tree after Level 2 (Height 2)**

yaml

CopyEdit

[Outlook]

/ | \

Sunny Overcast Rain

|

[Humidity]

/ \

High Normal

No Yes

**🪜 Step 4: Split Rain Branch (Height 2)**

Subset:

| **ID** | **Temp** | **Humidity** | **Windy** | **PlayTennis** |
| --- | --- | --- | --- | --- |
| 4 | Mild | High | False | Yes |
| 5 | Cool | Normal | False | Yes |
| 6 | Cool | Normal | True | No |
| 10 | Mild | Normal | False | Yes |
| 14 | Mild | High | True | No |

Try splitting by Windy:

| **Windy** | **Count** | **Yes** | **No** | **Entropy** |
| --- | --- | --- | --- | --- |
| False | 3 | 3 | 0 | 0.0 |
| True | 2 | 0 | 2 | 0.0 |



✅ Best split: Windy

**🌳 Tree after Level 3 (Height 3)**

yaml

CopyEdit

[Outlook]

/ | \

Sunny Overcast Rain

| |

[Humidity] [Windy]

/ \ / \

High Normal False True

No Yes Yes No

**✅ Final Observations**

* **Information Gain** helps choose the most “useful” feature at each level.
* The tree keeps splitting until:
  + All samples are pure (same class).
  + A stopping condition is met (max depth, min samples).
* In our case, the tree is 3 levels high (root + 2 splits).

**Dtreeviz library:**

****



