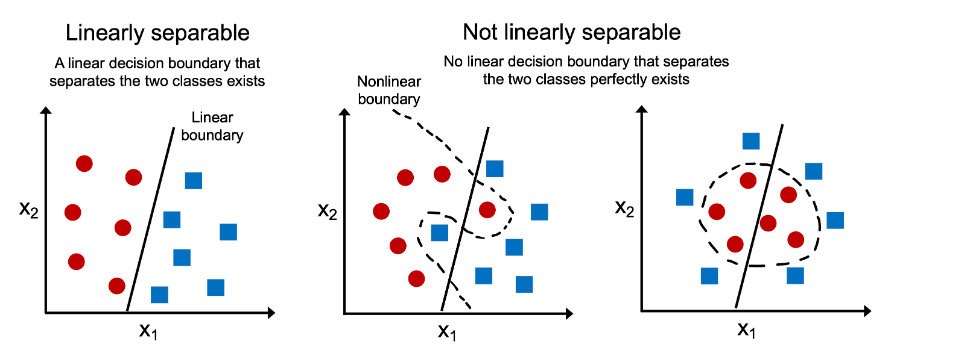
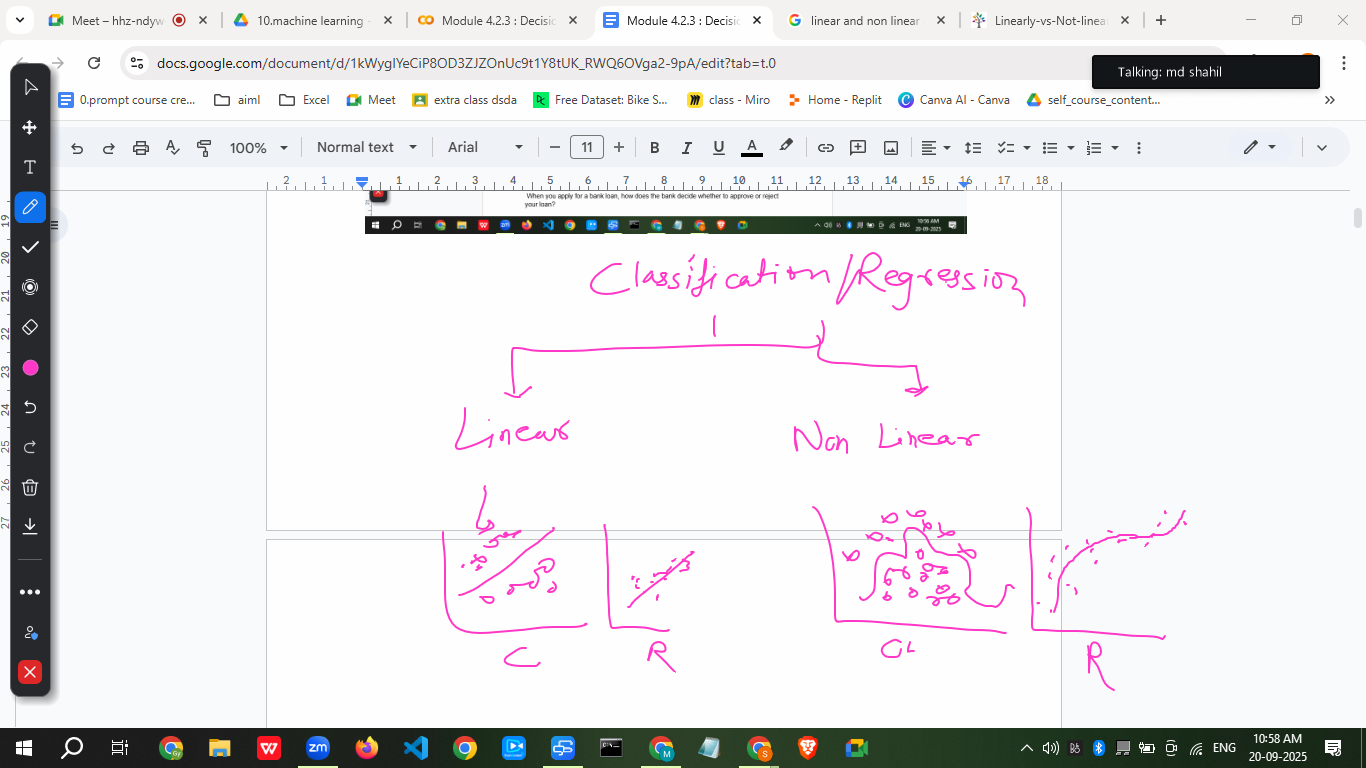
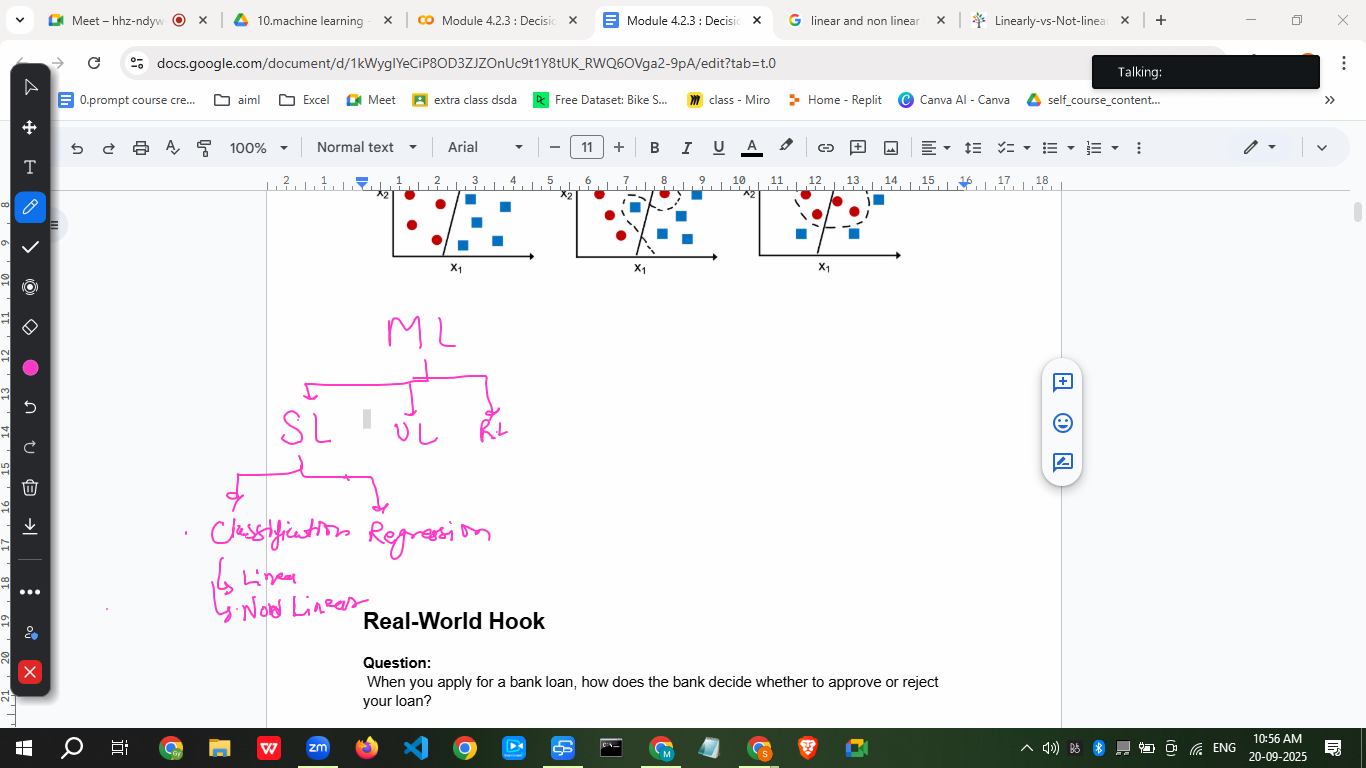
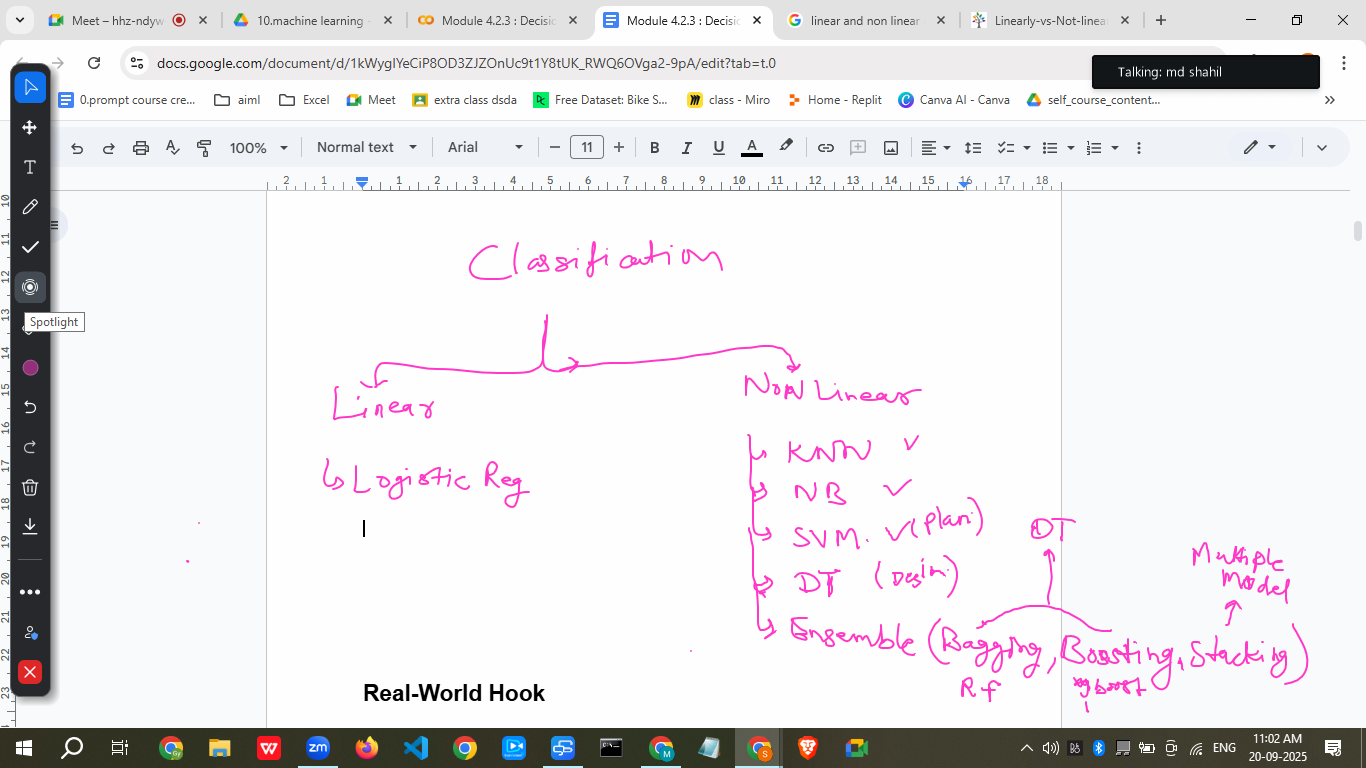
# **Module 4.2.3 : Decision Trees**

Use datasets in notebook→ iris dataset

## **1. Introduction to Decision Trees(Non-Linear)**







## **Real-World Hook**

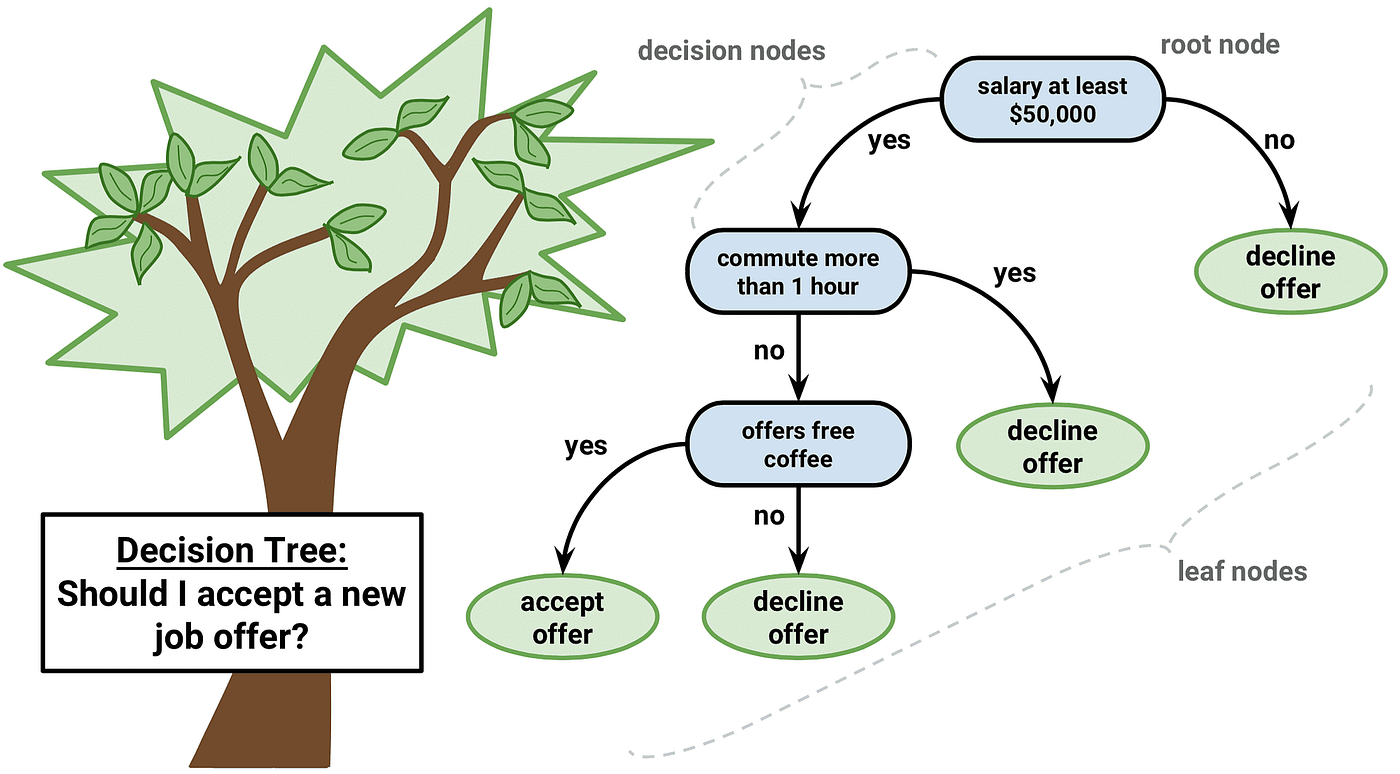
**Question:** When you apply for a bank loan, how does the bank decide whether to approve or reject your loan?

They ask for your **income**, **credit score**, **employment type**, and **past loan history**.  
 Then they make a decision like **“Approve” or “Reject”**.

This is exactly how **Decision Trees** work: step-by-step decisions based on conditions.

Decision → a choice or judgement that you make after thinking about various possibilities

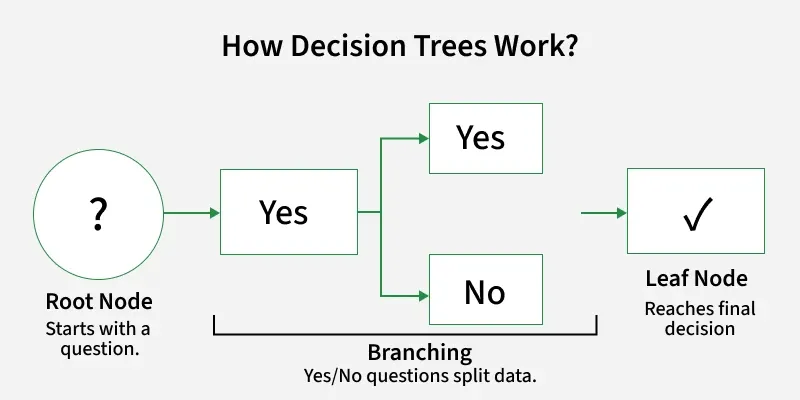
Tree →

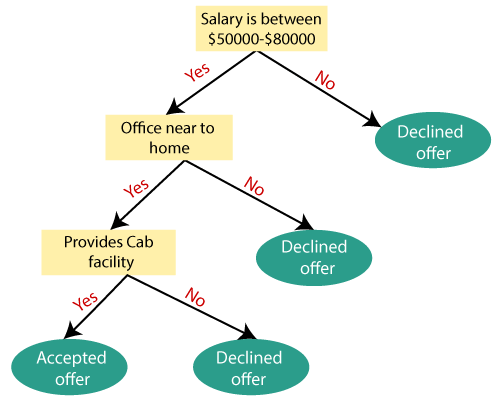


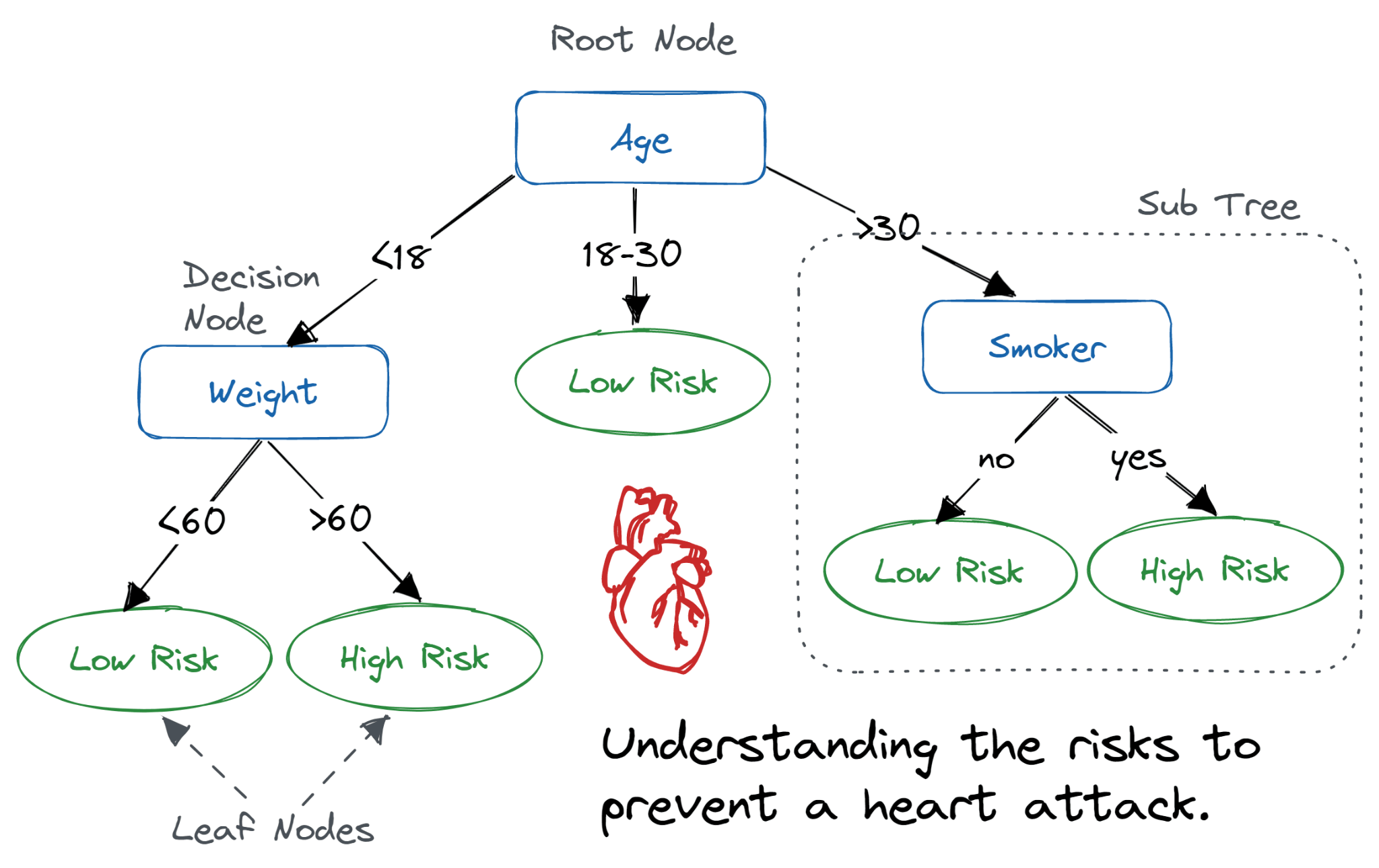
A decision tree is a visual, tree-like model that uses a series of questions to guide a decision-making process from a starting point to a final outcome, much like a flowchart. In machine learning, it's a supervised algorithm that can classify data or predict values (regression) by creating branches from nodes that represent tests on features, leading to leaf nodes that are the final decision or prediction.

## **Introduce Topic as a Solution**

In many real-life scenarios, we need to make **decisions based on conditions**.  
 Decision Tree helps **automate decision-making** using a tree-like structure, making it easy to **interpret and visualize**.







## **Theory Explanation**

### **✅ What is a Decision Tree?**

A Decision Tree is a **supervised machine learning algorithm** that splits data into branches based on conditions to predict an outcome.

### **✅ Why use Decision Trees?**

* Easy to **understand and interpret**.
* Works on **both classification and regression**.
* Handles **numerical and categorical data**.

### **✅ When to Use It?**

* When you want **transparent, interpretable models**.
* When dataset has **mixed data types**.
* When you need **fast predictions**.

### **✅ When NOT to Use It?**

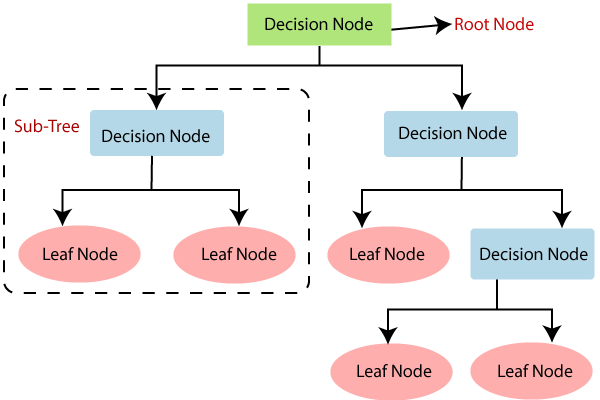
* When you have **high-dimensional data** → leads to overfitting.
* When **accuracy matters more than interpretability** → use ensemble methods (Random Forest, XGBoost).

### **✅ How It Works (Step-by-Step Algorithm)**

1. Start with the **entire dataset** as the root.
2. Select the **best feature to split** (using **Gini**, **Entropy**, or **Variance Reduction**).
3. Split the dataset into **branches**.
4. Repeat until:  
   * Node is pure (all same class).
   * Max depth reached.
   * No more splits possible.
5. Assign **leaf nodes** with the final prediction.

### **✅ Key Characteristics**

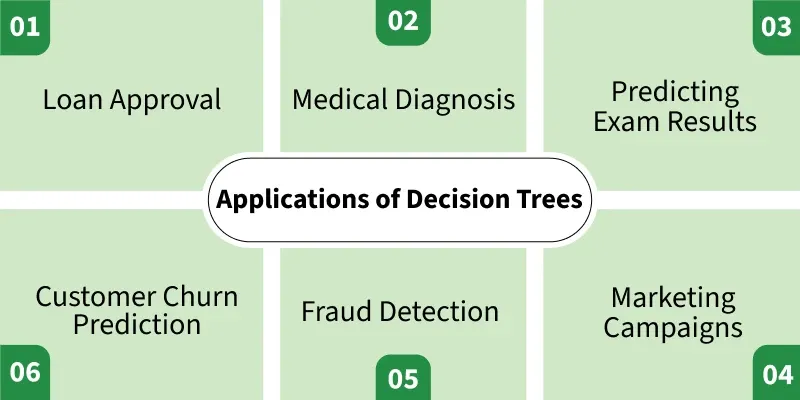
* Structure: **Root → Internal Nodes → Leaf Nodes**.
* Can handle **classification & regression**.
* Splitting criteria: **Gini, Entropy, Chi-square**.



* **Root Node** → The starting point of the tree.
* **Decision Nodes** → Where the data is split.
* **Leaf Nodes** → The final output (class or value).
* **Branches** → Paths from a node to another node.

**Real-World Examples:**

* Whether to give a **loan** to a customer.
* Whether an **email** is spam or not.
* Predicting **disease diagnosis** from patient data.



## **TYPES OF DECISION TREE**

## 

## 

**# START WITH THE CLASSIFICATIONS**

## **How Does the Decision Tree Algorithm Work?**

## Here are the **steps to build a Decision Tree**:

## **Start with the entire dataset** as the root node.

## **Select the best feature to split** using a criterion (e.g., Gini, Entropy, Information Gain, or Variance Reduction).

## **Split the dataset** into subsets based on the selected feature.

## **Create decision nodes and branches** for each split.

## **Repeat steps 2–4** for each subset (recursive splitting).

## **Stop splitting when**:

## All data in a node belongs to the same class.

## No more features are left.

## Maximum depth or minimum samples per node is reached.

## **Assign leaf nodes** with a class label (classification) or mean value (regression).

## **Prune the tree** (optional) to avoid overfitting.

============================================================================================================================================

## **Here are the key steps in how the Decision Tree algorithm works:**

## **Root Node Selection:** The process begins by selecting the "best" attribute from the dataset to serve as the root node. The "best" attribute is determined by metrics like **Information Gain (for classification) or variance reduction** (for regression), which quantify how effectively an attribute can separate the data into distinct groups.

## 

**What is Information Gain (IG)?**

## **1. Information Gain**

Information gain is a concept derived from Information Theory (like Entropy). In the machine learning field, the information gain (IG) is used in decision trees classification to decide on the best feature to consider to split each node of the tree.

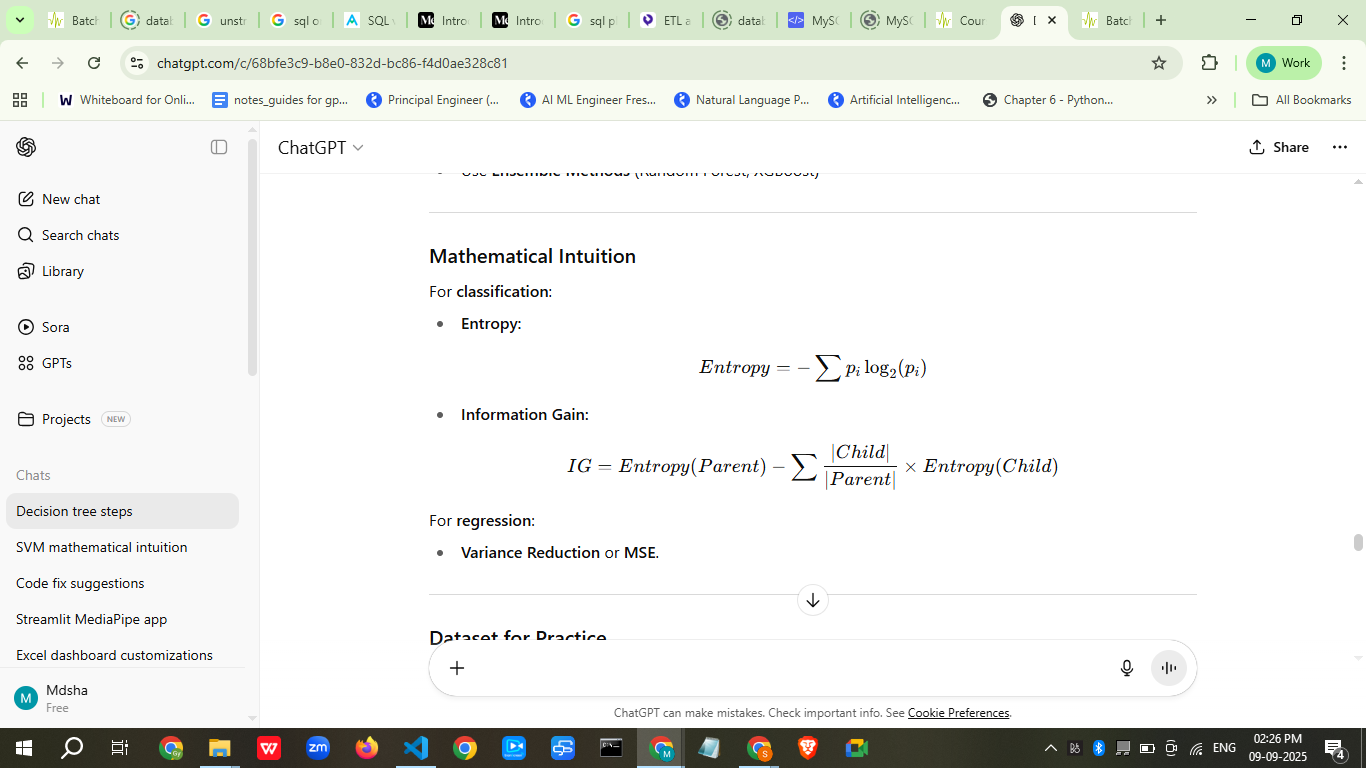
## **Mathematical Formula**

The information Gain is the difference between the Entropy of a dataset in a node of a subtree, and the weighted average entropies generated by using a given feature and threshold as a separator for this node. Here is the formula:

**IG(T,a) = H(T) – H(T|a)**

with:

* **T:** a set of training data
* **H(T)** : Is the **entropy** of this training dataset
* **H(T|a)** : Is the entropy of this set given a condition on the attribute *a*. This attribute is one of the features in the training dataset.



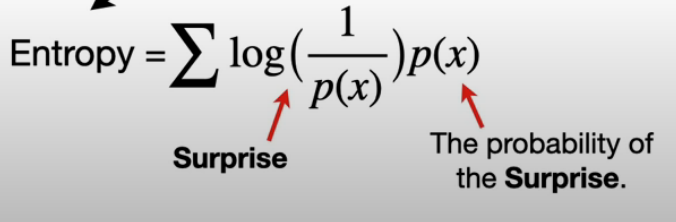
Go with information gain → <https://machinelearning-basics.com/information-gain-ig/>

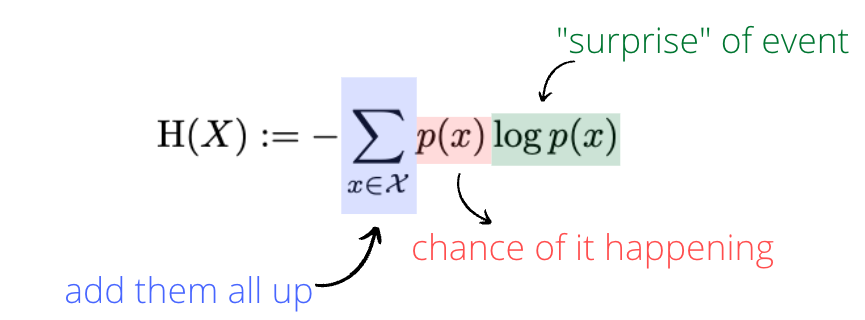
## 

## **2. Entropy**

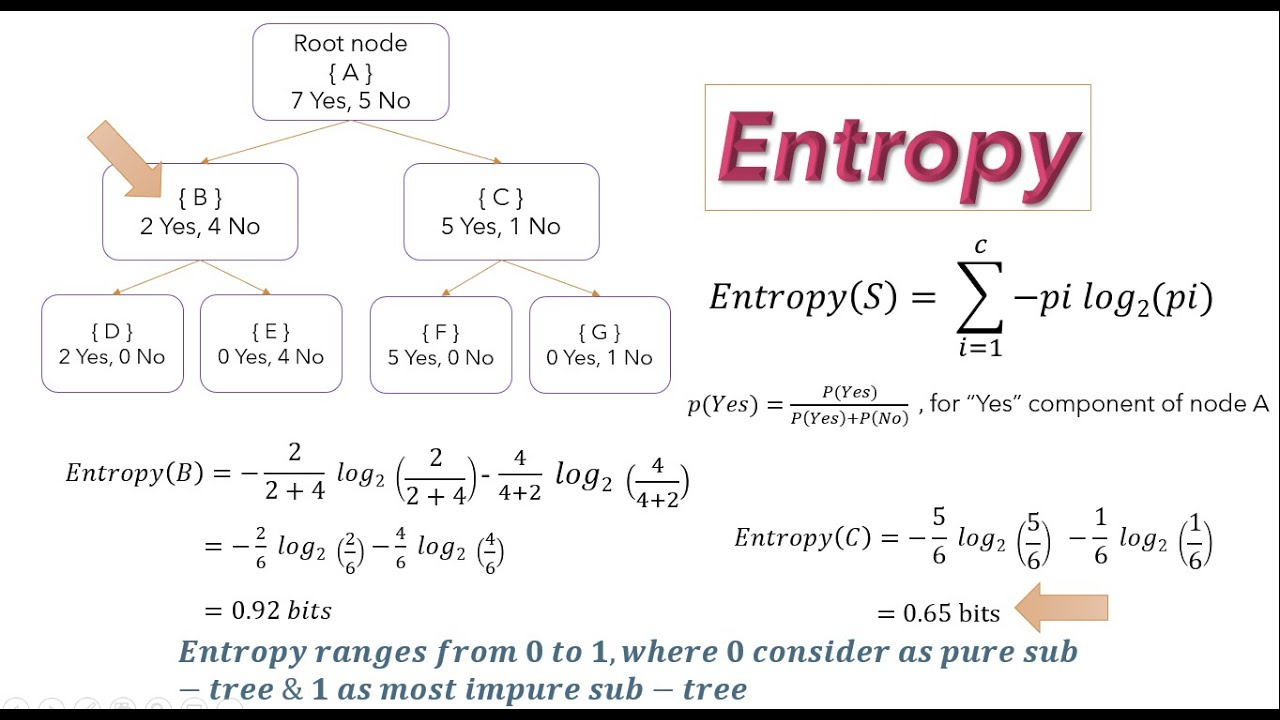
### **Entropy**

A measure of **uncertainty or randomness** in the dataset.  
 Formula:







****

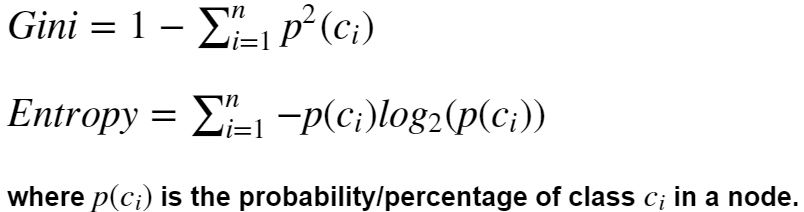
**Example:** If we have 10 emails, 6 spam and 4 non-spam:

## **3. Gini Index**

Another metric to measure impurity.

Gini(S)=1−∑i=1n(pi)2Gini(S) = 1 - \sum\_{i=1}^n (p\_i)^2

* Gini = 0 → Perfectly pure (all same class).
* Gini closer to 0 → better.



**NOTES:**

1. Information gain is calculated based on the entropy of the class distribution before and after the split, and it ranges from 0 (no gain) to 1 (maximum gain).
2. The entropy or the impurity measure can only take value from 0 to 1 as the probability ranges from 0 to 1 and hence, we do not want the above situation.

The Gini coefficient, a measure of income or wealth inequality, ranges from 0 to 1 (or 0% to 100%). A Gini coefficient of 0 indicates perfect equality, where every individual or household earns the same amount, while a coefficient of 1 represents perfect inequality, meaning one person holds all the income or wealth, and everyone else has none.

Understanding the Range

* **0 = Perfect Equality:** Everyone has the same income/wealth.
* **1 = Perfect Inequality:** One person has all the income/wealth, and others have none.
* **Scores Between 0 and 1:** Most countries fall between these extremes, with a lower score indicating greater equality and a higher score indicating greater inequality.

### **✅ Popular Algorithms**

| **Algorithm** | **Split Criterion** |
| --- | --- |
| **ID3** | Information Gain |
| **C4.5** | Gain Ratio |
| **C5.0** | Gain Ratio (Optimized) |
| **CART** | Gini / Variance |
| **CHAID** | Chi-square |

## **. CART and CHAID**

* **CART (Classification and Regression Trees):** Uses **Gini Index** for classification & **MSE** for regression.
* **CHAID (Chi-square Automatic Interaction Detection):** Uses **Chi-square test** for splitting categorical data.

We use CART Algorithms as a default

We use **different decision tree algorithms** (ID3, C4.5, CART, CHAID, etc.) because:

## **1️⃣ Different criteria for choosing splits**

* **ID3** → Uses **Information Gain (Entropy)** → works well for categorical features, but biased toward attributes with many distinct values.
* **C4.5** → Uses **Gain Ratio** (fixes ID3 bias) → handles both categorical & continuous data, supports missing values.
* **CART** → Uses **Gini Index** (for classification) or **MSE** (for regression) → always produces binary splits, efficient for large datasets.
* **CHAID** → Uses **Chi-square tests** → often used in marketing, survey, and social science datasets.

## **2️⃣ Different data types and needs**

* **Categorical data** → ID3, C4.5 handle directly.
* **Continuous data** → C4.5, CART can find optimal split thresholds.
* **Mixed data** (categorical + numerical) → CART works efficiently.

## **3️⃣ Handling missing values**

* Some algorithms (like C4.5) can split even if some rows have missing values, while others (like ID3) cannot without preprocessing.

## **4️⃣ Output style**

* **CART** → Binary splits only.
* **ID3 / C4.5** → Multiway splits possible (more than two branches at a node).

## **5️⃣ Performance and scalability**

* Large datasets → CART and modern gradient-boosted trees (XGBoost, LightGBM) are faster and more memory-efficient than classic ID3/C4.5.

## **6️⃣ Domain-specific preferences**

* Marketing analytics → CHAID (statistical tests).
* Machine learning competitions → CART-based algorithms (Random Forest, Gradient Boosting) dominate.
* Educational examples → ID3 (easy to calculate manually).

✅ **In short:** We use different algorithms in decision trees because **the best splitting criterion, handling of data types, and efficiency vary depending on the dataset and the problem requirements**.

A CART decision tree is a type of decision tree algorithm that handles both classification and regression tasks by creating a binary tree structure from data through recursive splitting. It uses criteria like the [Gini index](https://www.google.com/search?sca_esv=25dcd6d97260b75a&sxsrf=AE3TifOQ9zTBOo9DsbkW1e4wasJuf9NWtw%3A1758350192818&q=Gini+index&sa=X&ved=2ahUKEwiK0Ij13OaPAxWTT2wGHTl4DKMQxccNegQIHRAB&mstk=AUtExfBZrDvGRqb6yFJseu2v1BmRsDuIlF4oAmCZejdEMXfatTXr-3J6NURJ0HZ7b4XG4oPFjlu1bbwxjCkOiIyFXrrk-UsCTY20JwEg4OTIZSoitjxuSAwO1ZEgma9YXZoyGa7Q3wwApYVN-ODE5tOyg8eRtuEk0qfowLmPqmw0BoIhlzXzEwnV379x8HXtqhRvScPNAk9BAc7ImQuFoVn4uWmv2KXqXi4gmP3RvoUQ-c1vUKsV-ySqavthrHhzGbAiC3EMWolaR4jqMAVK8BFno4yhIp6cxjcWo7EsO3bAKv82AQ&csui=3) to split data into purer subsets at each node, with the goal of predicting a categorical outcome (classification) or a continuous value (regression) at the leaf nodes. CART trees are also known for their ability to be pruned to prevent overfitting and form the basis for more advanced algorithms like [random forests](https://www.google.com/search?sca_esv=25dcd6d97260b75a&sxsrf=AE3TifOQ9zTBOo9DsbkW1e4wasJuf9NWtw%3A1758350192818&q=random+forests&sa=X&ved=2ahUKEwiK0Ij13OaPAxWTT2wGHTl4DKMQxccNegQIHhAB&mstk=AUtExfBZrDvGRqb6yFJseu2v1BmRsDuIlF4oAmCZejdEMXfatTXr-3J6NURJ0HZ7b4XG4oPFjlu1bbwxjCkOiIyFXrrk-UsCTY20JwEg4OTIZSoitjxuSAwO1ZEgma9YXZoyGa7Q3wwApYVN-ODE5tOyg8eRtuEk0qfowLmPqmw0BoIhlzXzEwnV379x8HXtqhRvScPNAk9BAc7ImQuFoVn4uWmv2KXqXi4gmP3RvoUQ-c1vUKsV-ySqavthrHhzGbAiC3EMWolaR4jqMAVK8BFno4yhIp6cxjcWo7EsO3bAKv82AQ&csui=3).

How CART Works

* **Data Splitting:** The algorithm starts with a dataset and recursively splits it into smaller, purer subsets.
* **Splitting Criteria:**
  + **Classification:** It uses a measure called the Gini index to find the best split that reduces the "impurity" (or increases the homogeneity) of the resulting child nodes.
  + **Regression:** It uses the mean squared error (MSE) to find splits that minimize the variance of the target variable within each subset.
* **Binary Tree Structure:** Each split in the data creates a node in the tree, with a specific feature and value used to divide the data. Internal nodes represent decisions or rules, and the terminal nodes (or "leaf" nodes) contain the final predicted class label or value.
* **Pruning:** To prevent the model from becoming too complex and overfitting the training data, CART uses techniques like cost-complexity pruning to remove less informative branches and simplify the tree.

Key Characteristics

* **Binary Splits:** All splits in a CART decision tree are binary, meaning each internal node has exactly two child nodes.
* **Versatility:** It can be applied to both classification (predicting categories) and regression (predicting continuous numbers) problems.
* **Basis for Ensemble Methods:** The CART algorithm serves as a fundamental building block for more powerful ensemble methods such as Random Forests and Boosted Decision Trees.

<https://www.geeksforgeeks.org/machine-learning/cart-classification-and-regression-tree-in-machine-learning/>

Here’s a **clear comparison table** of the most common decision tree algorithms:

## **📊 Decision Tree Algorithms Comparison**

| **Feature / Aspect** | **ID3** | **C4.5** | **CART** | **CHAID** |
| --- | --- | --- | --- | --- |
| **Full form** | Iterative Dichotomiser 3 | – (successor to ID3) | Classification And Regression Trees | Chi-square Automatic Interaction Detector |
| **Split criterion** | **Information Gain** (Entropy) | **Gain Ratio** (fixes IG bias) | **Gini Index** (classification) or **MSE** (regression) | **Chi-square** test |
| **Type of split** | Multiway (can be >2 branches) | Multiway | **Binary only** | Multiway |
| **Handles numerical features?** | ❌ (only categorical unless pre-binned) | ✅ (finds thresholds) | ✅ | ✅ |
| **Handles categorical features?** | ✅ | ✅ | ✅ | ✅ |
| **Handles missing values?** | ❌ | ✅ (assigns fractional weights to branches) | ❌ (needs imputation) | ✅ |
| **Overfitting tendency** | High | Lower than ID3 (due to pruning) | Moderate (with pruning) | Moderate |
| **Speed on large datasets** | Slower | Slower than CART | **Faster** | Slower |
| **Interpretability** | Easy | Easy | Easy | Easy |
| **Used for** | Education, small datasets | General-purpose, mixed data | Industry standard (ML tasks) | Marketing, social sciences |
| **Advantages** | Simple, intuitive | Handles both data types, missing values | Fast, works with regression & classification | Finds statistically significant splits |
| **Disadvantages** | Biased toward attributes with many values, can’t handle continuous data | More complex, slower | Requires preprocessing for missing values, binary splits only | Statistical assumption limits flexibility |
| **Example use case** | Teaching entropy calculations | Medical diagnosis | Credit scoring, fraud detection | Market segmentation |

## 

## **⚡ Summary:**

* **ID3** → Good for **learning** and **small categorical datasets**.
* **C4.5** → Better than ID3, handles real-world messy data.
* **CART** → Most widely used in industry, basis for **Random Forest** & **Boosted Trees**.
* **CHAID** → Best for **marketing & survey analysis** with categorical data.

## **5. Performance Metrics**

Once the Decision Tree is built, evaluate using:

* **Confusion Matrix**
* **Accuracy**
* **Precision**
* **Recall**
* **F1-Score**
* **ROC-AUC**

## **6. Pruning Techniques**

Decision Trees can easily **overfit**.  
 **Pruning** helps reduce complexity:

* **Pre-pruning:** Stop splitting early (max depth, min samples).
* **Post-pruning:** Build full tree, then remove unimportant branches.

### **✅ Advantages**

## ✔ Easy to interpret and visualize ✔ Handles both **numerical & categorical data** ✔ Works without scaling or normalization

## 

### **✅ Disadvantages**

## ✖ Prone to **overfitting** ✖ Sensitive to **small changes in data** ✖ Not great for **high-dimensional data**

## **How to Overcome:**

## Use **Pruning**

## Use **Ensemble Methods** (Random Forest, XGBoost)

## 

## **Business Scenario**

Banks use Decision Trees to:

* Approve/Reject Loans
* Detect Fraud
* Credit Risk Analysis

## **Practice Session (5 Questions)**

1. What is a Decision Tree and where is it used?
2. Explain the difference between **ID3** and **CART**.
3. Why does overfitting occur in Decision Trees?
4. Write Python code to build a Decision Tree on the given dataset.
5. Explain Gini Index vs Entropy.

## **Case Study**

**Company:** A bank wants to automate **loan approval**.  
 You need to:

* Build a **Decision Tree Model**.
* Deploy it using **Streamlit** for employees to use.

## **Interview Questions**

### **Theoretical (10)**

1. What is a Decision Tree?
2. Explain Entropy and Information Gain.
3. Difference between ID3, C4.5, and CART.
4. What is Gini Index?
5. Why do we prune trees?
6. Advantages & disadvantages of Decision Trees.
7. When to use Decision Trees?
8. Difference between classification and regression trees.
9. What is overfitting and how to prevent it?
10. What is CHAID?

### **Practical (10)**

1. Implement a Decision Tree in Python.
2. Visualize a Decision Tree.
3. Encode categorical variables for Decision Trees.
4. Change splitting criterion from Gini to Entropy.
5. Limit the max depth of the tree.
6. Perform pruning in scikit-learn.
7. Predict a new sample using a trained tree.
8. Calculate accuracy score of your tree.
9. Plot feature importance.
10. Export the tree as text.

## **Recap**

* Decision Trees = Simple, interpretable ML algorithm.
* Works for **classification & regression**.
* Popular algorithms: ID3, C4.5, CART, CHAID.
* Use when interpretability is key.

## **Resource Links**

* [Scikit-learn Decision Trees](https://scikit-learn.org/stable/modules/tree.html)
* [ID3 Algorithm Explanation](https://en.wikipedia.org/wiki/ID3_algorithm)

## **Practical example**

## Let’s take the same dataset as in the entropy chapter to understand how we compute this metric.

## 

## *They are fake data, built just for illustration*

## Consider a scenario where a bank collects this dataset with the intention of assigning scores to clients.

## A client having a good credit note, will have access to a loan with the bank. And the client, having a bad note, will see his request refused by the bank.

## The goal is to predict the credit note of a client, based on his/her information: savings, assets, and salary.

## We want to build a tree for this dataset. The first question is what would be the best feature (Savings, Assets, Salary) to begin with and put in the root node?

## First we need to compute the general entropy for the root node, which is the general entropy for the whole dataset:

### **General Entropy**

## Here is how we compute the entropy of this dataset:

## We have 4 “Good” notes and 3 “Bad” notes.

## The probability to have a “Good” note is:

## 

## It’s entropy is:

## 

## The probability to have a “Bad” note is:

## 

## It’s entropy is

## 

## The general Entropy of this dataset is:

## 

## Now let’s consider the first feature *Savings* as a candidate to split the root node, and let’s compute its information gain:

### **Feature 1 – Categorical: Savings**

## **Information Gain (IG)**

## The entropy computed based on the Savings split, is equal to 0.464.

## Thus the information gain for this split is:

## H(T) = 0.985

## H(T|a) = 0.464

## IG(T,a) = H(T) – H(T|a) = 0.522

## Pretty much an important reduction of entropy, and greater information value.

## For *Assets*, if you follow the same logic, you will find an information Gain about 0.02: very small reduction of entropy. *Assets* feature is not giving valuable information value to classify credit notes. Therefore, we will not chose it. For the moment Savings feature has the largest value.

## Now, Let’s compute the *Information Gain* for *Salary*, the only continuous feature in our dataset. In the Entropy article you will see how we compute the different thresholds for a continuous variable. In the following, let’s do it for the threshold *$27.5.*

### **Feature 2 – Continuous: Salary**

## Here is the subtree of *Salary* feature with threshold *$27.5K*:

## Information Gain (IG)

## The information Gain is 0.006. There is almost no information value gained. This is the worst feature to begin with.

## Now that you understand how we compute the information gain:

## You can do it for other remaining features/values for the root node.

## You sort the Information Gain, to keep the largest value. You have now the first candidate for the root node.

## If you have done the exercise, you will find that Savings was the best candidate for the root node.

## Now our first node looks like:

## Information Gain (IG)

## The left side is a leaf because we only have 3 observations that are all good.

## But for the right side, you need to figure out which feature to use in this node.

## To do that, you need first to compute the entropy of this new subset:

## Information Gain (IG)

## Then consider the different features on this subset one by one, by following the above method to compute the Information Gain.

## Let’s do it for one feature, *Assets*, and I let you do it for the others.

## Information Gain (IG)

## For Assets, the IG = 0.311.

## Now, you need to do it for the other features and choose the one with the highest IG value.

## **Summary**

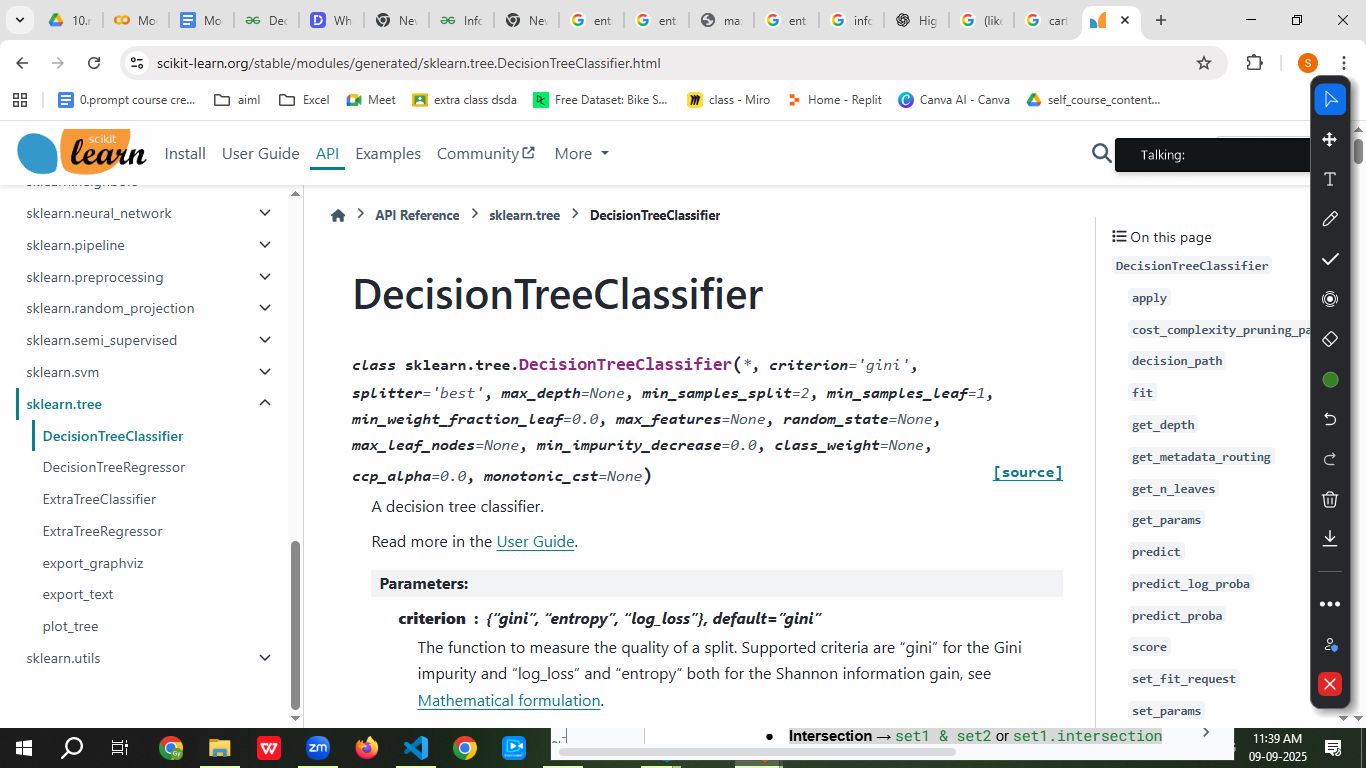
## Important thing to remember: *The larger the value of Information Gain of a subtree, the better is the reduction of entropy*, and the better is the feature (and its value) used to split the node of this tree.

## 

## 

## 

## **7. Python Example: Decision Tree Classification**



import pandas as pd

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier, plot\_tree

import matplotlib.pyplot as plt

# Load dataset

iris = load\_iris()

X = pd.DataFrame(iris.data, columns=iris.feature\_names)

y = iris.target

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create model

model = DecisionTreeClassifier(criterion='entropy', max\_depth=3, random\_state=42)

model.fit(X\_train, y\_train)

# Predictions

y\_pred = model.predict(X\_test)

# Accuracy

from sklearn.metrics import accuracy\_score

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

# Plot Tree

plt.figure(figsize=(12,8))

plot\_tree(model, feature\_names=iris.feature\_names, class\_names=iris.target\_names, filled=True)

plt.show()

## **8. Assignment**

1. Build a Decision Tree to classify **Titanic survivors**.
2. Try both **Entropy** and **Gini Index** as splitting criteria.
3. Compare results and explain differences.
4. Apply **pruning** to prevent overfitting.
5. Evaluate using accuracy, precision, recall, and F1-score.