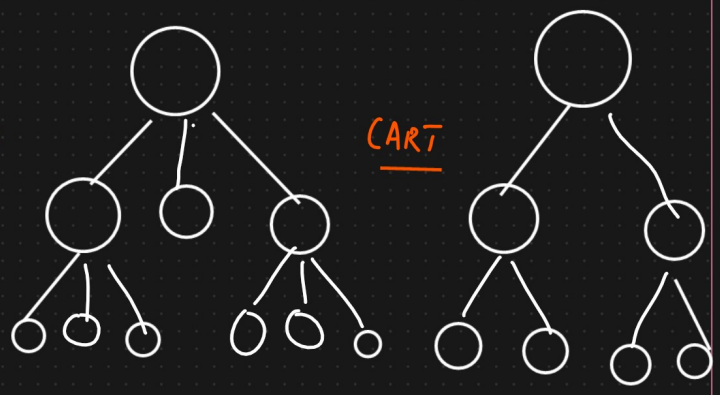
Decision Tree

A decision tree algorithm is a popular machine learning technique used for classification and regression tasks. It works by splitting the data into subsets based on the value of input features, creating a tree-like model of decisions. Each internal node represents a feature (or attribute), each branch represents a decision rule, and each leaf node represents an outcome (class or value).

1. Decision Tree Classifier [classification]
2. Decision Tree Regressor [Regression]

ID 3 CART



Decision Tree Classifier =>

Two types

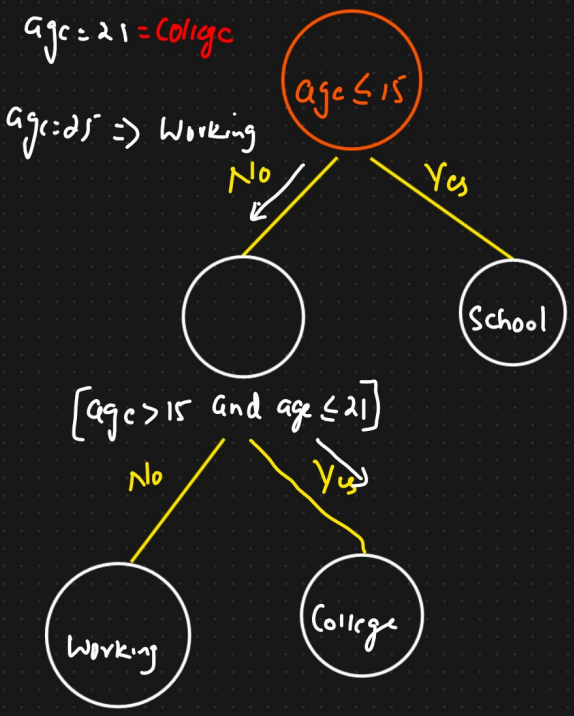
1. ID3 [Iterative Dichotomiser 3]
2. CART [Classification And Regression Tree]

age = 14

```
if (age ≤ 15):
    Print ("School").
clif (age > 15 and age ≤ 21):
    Print ("College")
else:
    Print ("Working")
```

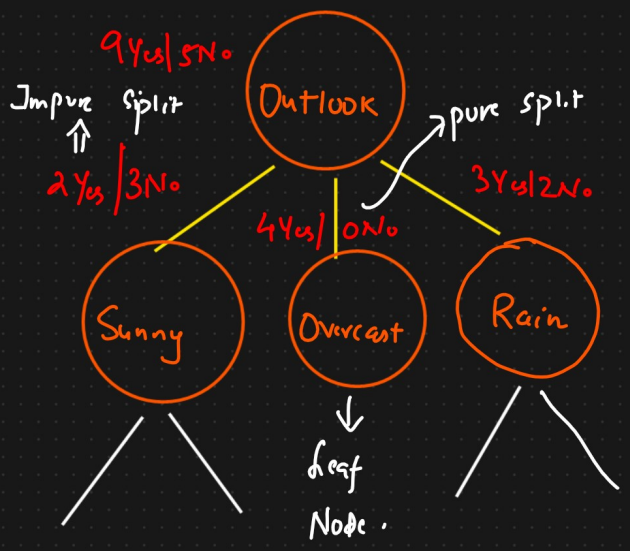
age = 21 = College

age = 25 => Working



Dataset → Predict Play Tennis OR Not

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



① Purity check : Pure Split or Impure Split

- Entropy
- Gini Impurity

} Measure of purity. ✓

② What feature you need to select to start the split ? → Information Gain

Binary classification

① Entropy

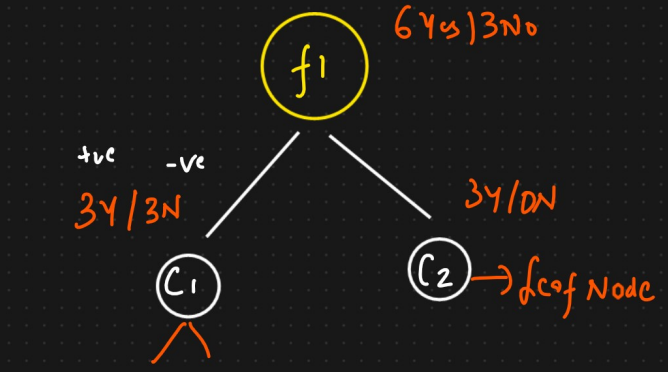
$$H(S) = -P_+ \log_2 P_+ - P_- \log_2 P_-$$

P_+ = probability of positive category

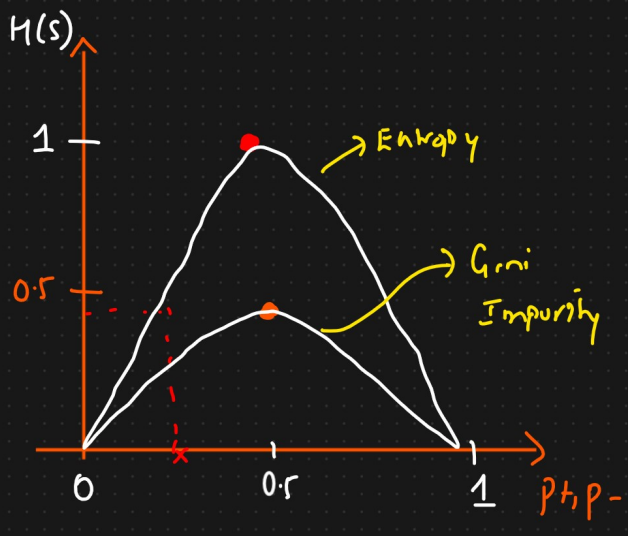
P_- = probability of negative category

② Gini Impurity

$$G.I = 1 - \sum_{i=1}^{n} (p)^2$$



$$H(C_1) = -P_+ \log_2 P_+ - P_- \log_2 P_-$$
$$= -\frac{3}{6} \log_2 \left(\frac{3}{6}\right) - \left(\frac{3}{6}\right) \log_2 \left(\frac{3}{6}\right)$$



$= -\frac{1}{2}\log_2(1/2) - (1/2)\log_2(1/2)$.

$= 1$ ⇒ Impure Split

$H(C_2) = -\frac{3}{3}\log_2(3/3) - 0/3\log_2(0/3)$

$= 0$ ⇒ Pure Split

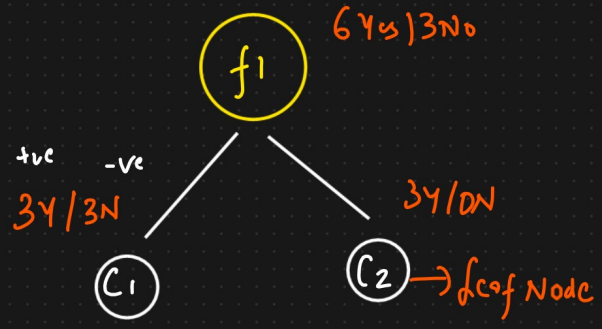
Multiclass

C_1	C_2	C_3
Yes	No	May Be

$H(S) = -P_{C_1}\log_2 P_{C_1} - P_{C_2}\log_2 P_{C_2} - P_{C_3}\log_2 C_3$

② Gini Impurity

$G.I = 1 - \sum_{i=1}^{n} (p)^2$



$= 1 - [(p_+)^2 + (p_-)^2]$

$= 1 - \left[\left(\frac{3}{6}\right)^2 + (3/6)^2\right]$

$= 1 - \left[\frac{1}{4} + \frac{1}{4}\right]$

$= 1/2 = 0.5$ ⇒ Impure Split

$1 - [(3/3)^2 + (0/3)^2]$

$= 1 - 1$

$= 0/1$ ⇒ Pure Split

② What feature you need to select to start the split? → Information Gain

f1 f2 f3 O/P

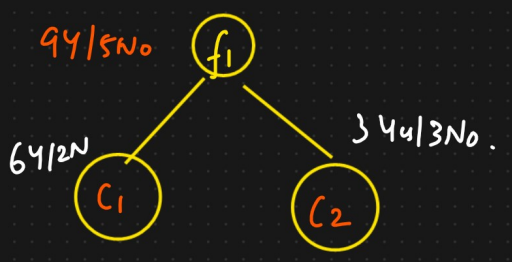
f1 f2

Information Gain

Entropy of the root node

$$Gain(S, f_1) = |H(S)| - \sum_{v \in val} \frac{|S_v|}{|S|} H(S_v)$$

f1	f2	f3	O/P
			9Y/5No
			6Y/2N
			3Y/3No.



$$H(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

$$= -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 5/14$$

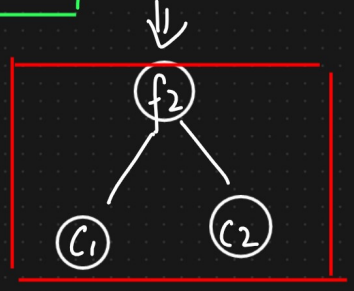
$$\approx 0.94 //$$

$$H(C_1) = -\frac{6}{8} \log_2 (6/8) - \frac{2}{8} \log_2 \frac{2}{8} \approx 0.81$$

$$H(C_2) = 1 //.$$

$$Gain(S, f_1) = 0.94 - \left[\frac{8}{14} * 0.81 + \frac{6}{14} * 1\right]$$

$$Gain(S, f_1) = 0.049$$



$$Gain(S, f_2) = 0.051 \quad > \quad Gain(S, f_1) = 0.049$$

Information gain is more when we split using f2.

Entropy Vs Gini Impurity

Whenever dataset is small → Entropy

dataset is huge → Gini Impurity

IMP

A decision tree is a widely used machine learning algorithm for classification and regression tasks. It models decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Here's a comprehensive overview:

Key Components
Root Node: The top node of the tree that represents the entire dataset, which is then split into subsets.
Decision Nodes: Nodes where the data is split based on a certain attribute.
Leaf Nodes (Terminal Nodes): Nodes that represent a class label or a continuous value (depending on whether it's a classification or regression tree).
Branches: Represent the outcome of a decision node split.

How Decision Trees Work
Splitting: The process of dividing a node into two or more sub-nodes.
Decision: At each node, the tree uses a feature of the data to make a decision about which branch to follow.
Stopping Criteria: The process continues until a stopping criterion is met. Criteria can include maximum tree depth, minimum number of samples required to split a node, or minimum number of samples in a leaf node.
Building a Decision Tree
Select the Best Attribute: Use a measure like Gini impurity or information gain to select the attribute that best separates the data.
Splitting: Split the dataset into subsets based on the selected attribute.
Repeat: Repeat the process for each subset using the remaining attributes.

Advantages
Simple to Understand and Interpret: Trees can be visualized.
Requires Little Data Preparation: No need for normalization or scaling.
Handles Both Numerical and Categorical Data: Capable of working with mixed data types.
Non-Parametric and Non-Linear: No assumptions about the space distribution and the classifier structure.
Disadvantages
Overfitting: Trees can become complex and overfit the data.
Unstable: Small changes in the data can lead to a completely different tree.
Bias: Decision trees can be biased with imbalanced datasets.
Mitigating Overfitting
Pruning: Remove branches that have little importance.
Setting Constraints: Limit the maximum depth, minimum samples per leaf, or minimum samples to split a node.
Advanced Topics
Random Forest: An ensemble of decision trees to improve accuracy and control overfitting.
Gradient Boosting Trees: Build trees sequentially to correct errors of the previous trees.

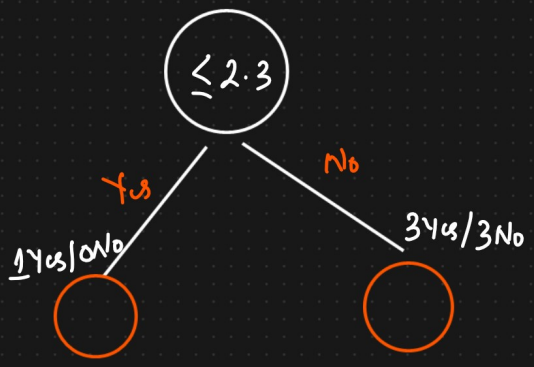
Decision Tree For Numerical Split

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

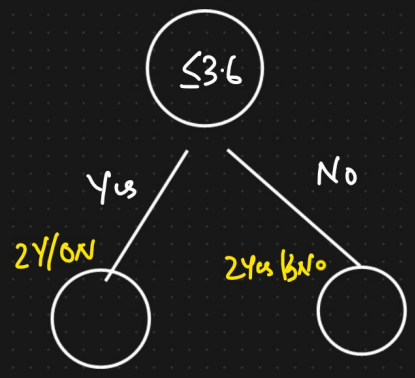
① Sorting the feature

f2	f1	O/P
	→ 2.3	Yes
	→ 3.6	Yes
	→ 4	No
	5.2	No
	6.7	Yes
	7.8	No
	9.0	Yes

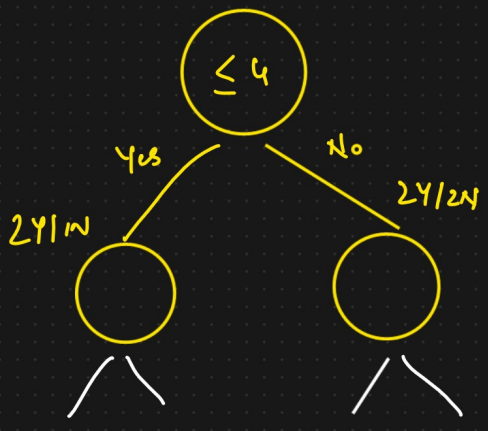
① Threshold = 2.3



② Threshold = 3.6



③ Threshold = 4



Millions of Records

Time Complexity ↑↑

Creating a decision tree for numerical data follows the same fundamental principles as for categorical data, but the splitting criteria are adapted to handle continuous values. Below is a step-by-step guide and example implementation using Python's scikit-learn library.

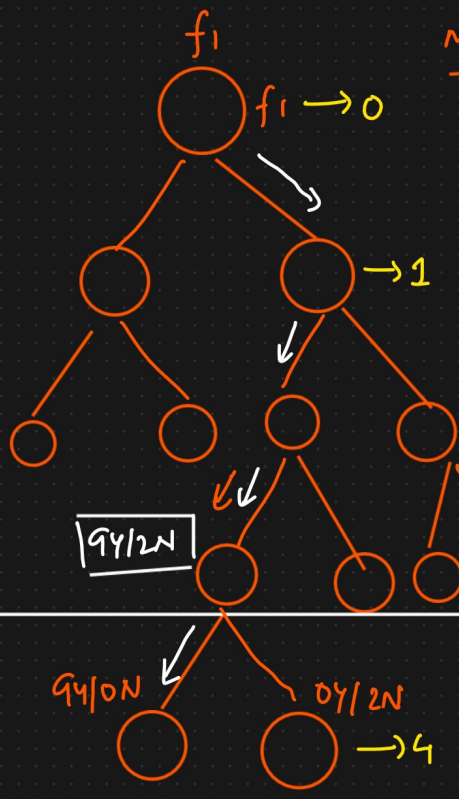
Key Concepts

Splitting Criteria: For numerical data, the algorithm considers all possible splits for each feature and selects the one that minimizes the impurity (e.g., Gini impurity for classification or variance reduction for regression).

Decision Tree Post Pruning And Pre Pruning [Reduce Overfitting]

Traning Data

① Post Pruning
② Pre Pruning
To Reduce Overfitting.



MaxDepth

Overfitting

TRAINING ACC ↑↑

TEST DATA ACC ↓↓

Small Dataset Yes

① Post Pruning

① Decision Tree Construct

② Prune the Tree at some level of node

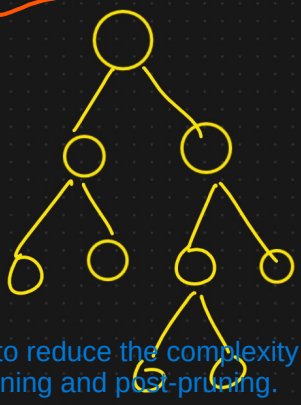
② Pre pruning = Hyperparameter Tuning

Large Dataset

Post-Pruning (Cost Complexity Pruning)
Post-pruning, also known as cost complexity pruning or error-based pruning, involves growing the tree to its maximum depth and then removing nodes that provide little power in predicting target variables. This is done after the tree has been fully built.

max depth

Criterion



Pre-Pruning and Post-Pruning in Decision Trees
Pruning is a technique used in decision tree algorithms to reduce the complexity of the model and prevent overfitting. There are two main types of pruning: pre-pruning and post-pruning.

Pre-Pruning (Early Stopping)
Pre-pruning, also known as early stopping, involves halting the growth of the tree before it fully develops. This is done by setting constraints during the construction of the tree. Common pre-pruning techniques include:

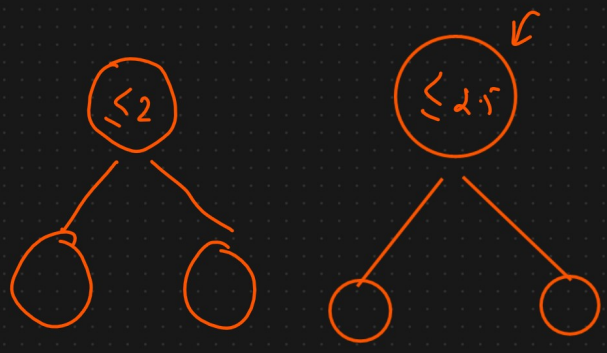
Maximum Depth (max_depth): Limits the depth of the tree. Trees that are too deep might capture noise in the data.
Minimum Samples Split (min_samples_split): The minimum number of samples required to split an internal node.
Minimum Samples Leaf (min_samples_leaf): The minimum number of samples required to be at a leaf node.
Maximum Leaf Nodes (max_leaf_nodes): Limits the number of leaf nodes in the tree.
Maximum Features (max_features): The number of features to consider when looking for the best split.

```
regressor = DecisionTreeRegressor(max_depth=5, min_samples_split=10, min_samples_leaf=5)
```

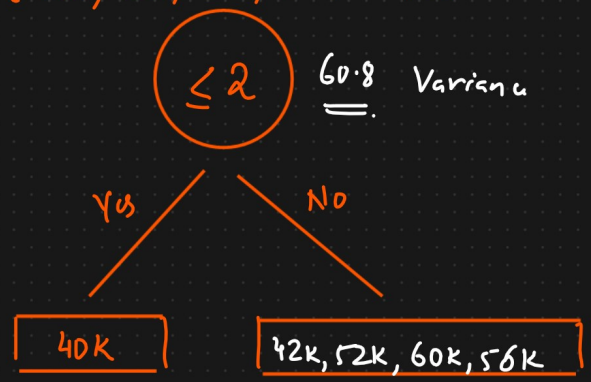
Decision Tree Regressor

Dataset

Exp	Career Gap	Salary
→ 2	Yes	40K
→ 2.5	Yes	42K
3	No	52K
4	No	60K
4.5	Yes	56K
		50K = $\bar{y}$



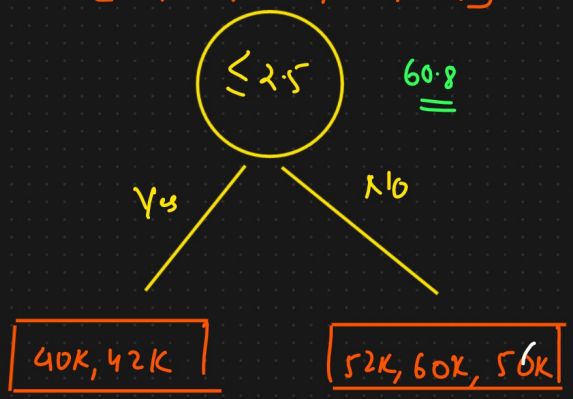
[40K, 42K, 52K, 60K, 50K]



100 (Variance)

Variance (51)

[40K, 42K, 52K, 60K, 50K]



Variance 82

Variance Reduction

$$\text{Variance} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$ [Mean Squared Error].

$$\text{Variance (Root)} = \frac{1}{5}\left[(40-50)^2 + (42-50)^2 + (52-50)^2 + (60-50)^2 + (56-50)^2\right]$$

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

$$\text{Var(left)} = \frac{1}{n}\sum_{i=1}^{n}(y-\bar{y})^2$$

$$= \frac{1}{2}\left[(40-50)^2 + (42-50)^2\right]$$

$$= \frac{1}{2}\left[100 + 64\right]$$

$$= \frac{164}{2} = 82//.$$

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

$= 60.8$

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

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

$= \frac{1}{1}\left[(40-50)^2\right]$

$\text{Variance (left)} = 100$

$\text{Variance Reduction} = 60.9 - \left[\frac{2}{5} * 82 + \frac{3}{5} * 46.66\right]$

$= 0.004$

$\text{Variance (Right)} = \frac{1}{4}\left[(42-50)^2 + (52-50)^2 + (60-50)^2 + (56-50)^2\right]$

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

$= 51$

$Wi(L) = \frac{1}{5} \quad Wi(R) = 4/5$

$\text{Variance Reduction} = \text{Var(Root)} - \sum Wi\,\text{Var(child)}$

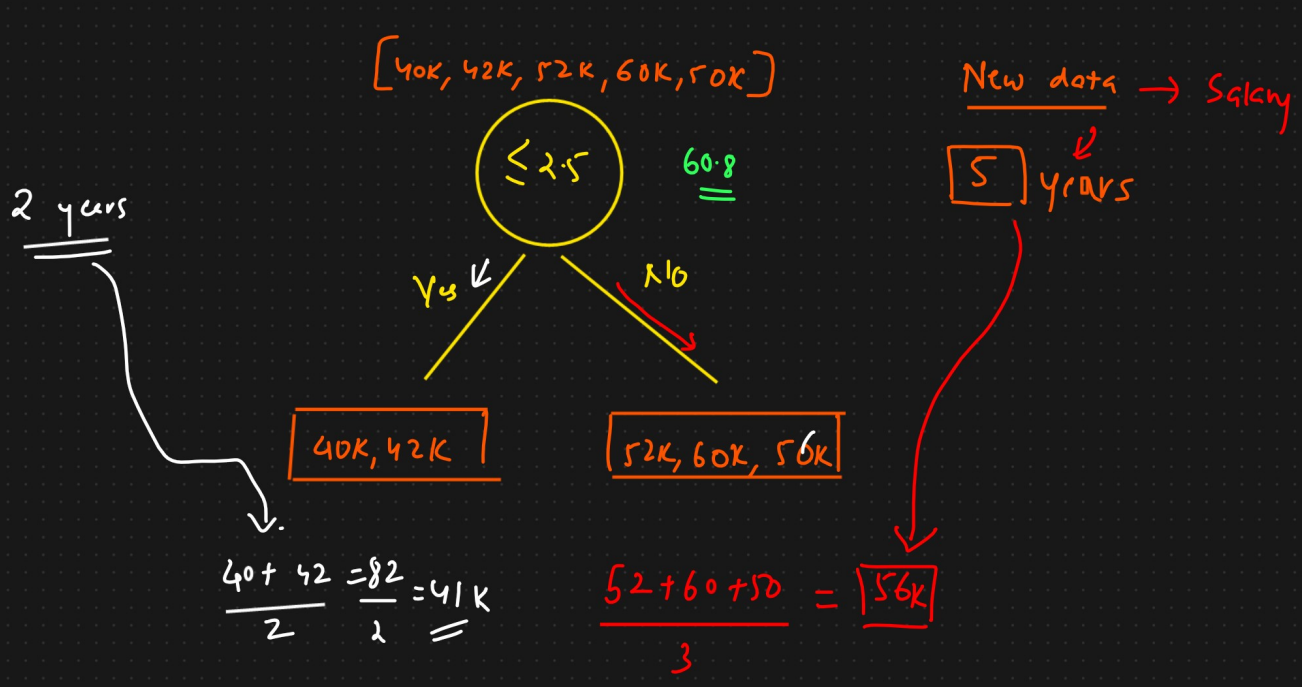
$= 60.8 - \left[\frac{1}{5} * 100 + \frac{4}{5} * 51\right]$

$= 60.8 - 20 - 40.8$

Variance Reduction $= 0$

0.004

Variance Reduction (Left Split) $<$ VR (Right Split)



A decision tree regressor is a machine learning algorithm used for predicting continuous values. It splits the data into subsets based on feature values, creating a tree-like model of decisions that leads to the prediction of numerical values.

Principle of Decision Tree Regressor
A decision tree regressor works on the principle of recursively partitioning the feature space into smaller regions and fitting a simple model (e.g., a constant value) in each region. Here's a step-by-step explanation of how it works:

1. Splitting the Data
The main goal of the decision tree algorithm is to partition the data in such a way that the variance within each partition (region) is minimized. This is achieved through a series of splits based on the input features.

Root Node: The root node represents the entire dataset.
Decision Nodes: Each decision node represents a test on an individual feature, splitting the data into two or more subsets.
Leaf Nodes: These are terminal nodes that contain the predicted output value for each region.
2. Choosing the Best Split
At each node, the algorithm chooses the best feature and the best threshold to split the data. The goal is to minimize the sum of squared errors (SSE) within each of the resulting subsets. The criterion for finding the best split involves:

Variance Reduction: Measures the reduction in variance after a split. The split that results in the greatest reduction in variance is chosen.