University Of Canberra

WEEK6

Pattern Recognition and Machine Learning

Created by **Elane Peng**

Objectives

- Understand the working principles of kNN and decision trees and their usage.
- Apply kNN and decision trees for classification (computing distances, majority voting, and impurity measures, e.g., entropy/Gini).
- Analyse the impact of key parameters (e.g., choice of k, split criteria, tree depth) on model performance and decision boundaries.

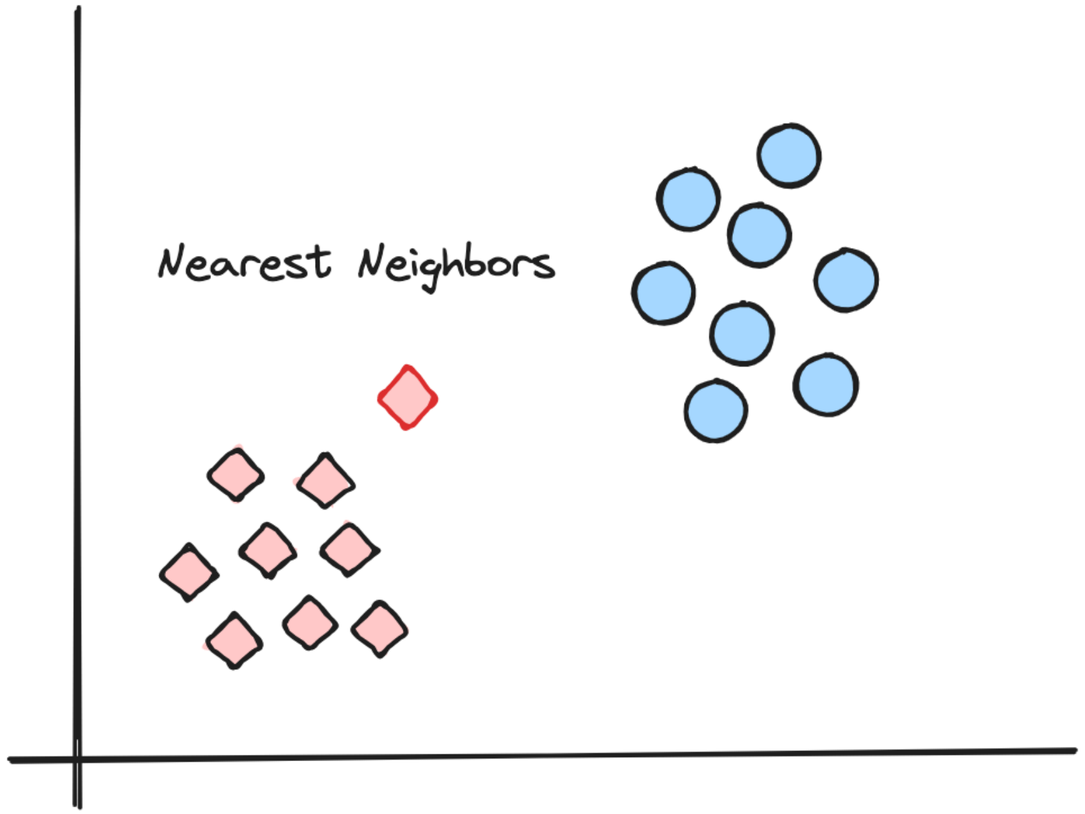
k-Nearest Neighbors (kNN)

Q1: What is the main idea behind the k-Nearest Neighbors (kNN) algorithm?

k-Nearest Neighbors (kNN)

Q1: What is the main idea behind the k-Nearest Neighbors (kNN) algorithm?

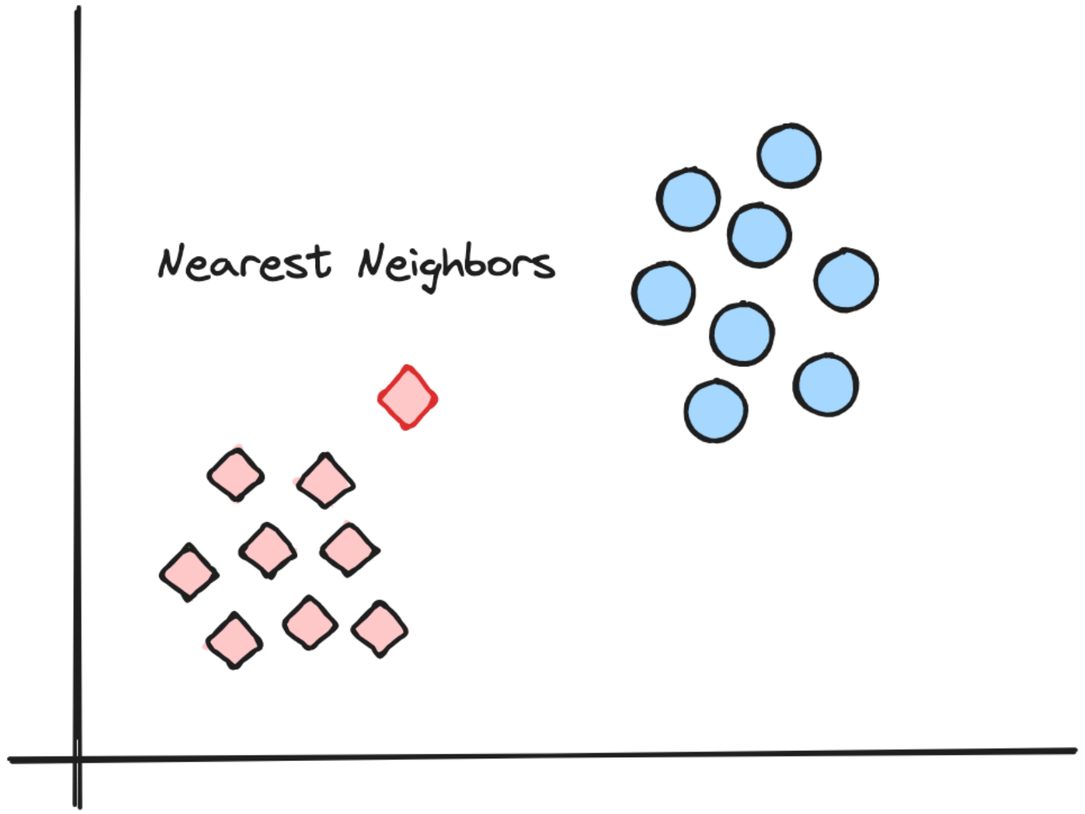
kNN is a lazy learning (or instance-based learning) algorithm that classifies a new data point based on the majority label of its k closest neighbors in the training set. It relies on a distance metric (commonly Euclidean distance) to determine “closeness”.



k-Nearest Neighbors (kNN)

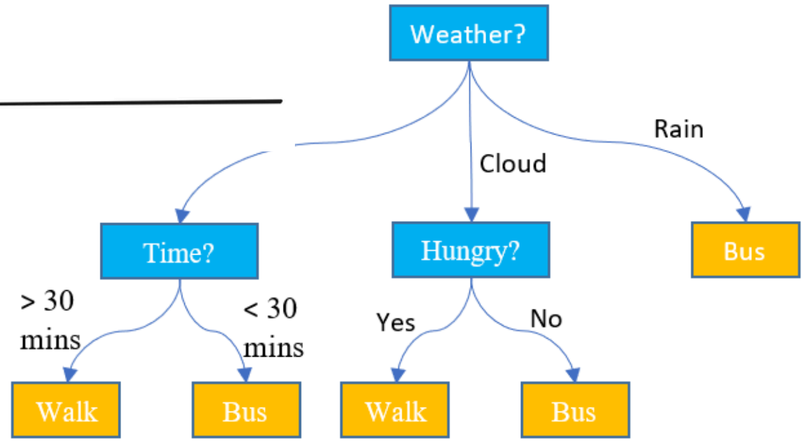
Q1: What is the main idea behind the k-Nearest Neighbors (kNN) algorithm?

Here “lazy learning” means the algorithm does not build an explicit model during training. Instead, it stores the entire training dataset and only does the real computation when a query (new test point) arrives. So, training is cheap (just memorization), but prediction is expensive (must compute distances to many points). Lazy learning is also referred to as instance-based learning.



k-Nearest Neighbors (kNN)

Q1: What is the main idea behind the k-Nearest Neighbors (kNN) algorithm?



The opposite is “eager learning”, or model-based learning. These methods build an explicit model during training (e.g., a decision boundary, probability distribution, or tree). The model summarizes the training data, so predictions are fast. Examples include Decision Trees, Logistic Regression, SVMs, Neural Networks.

k-Nearest Neighbors (kNN)

Q2: What is the effect of choosing a very small vs. very large value of k in kNN?

k is a hyperparameter that represents the number of nearest neighbors to consider when classifying (or predicting) a new data point.

- Small k (e.g., k=1): Model is very sensitive to noise → high variance.
- Large k: Model becomes smoother, but may ignore local structure → high bias.

k-Nearest Neighbors (kNN)

Q2: What is the effect of choosing a very small vs. very large value of k in kNN?

- A good k is often found using cross-validation: you try different values and pick the one that gives the best performance on validation data.
- A common heuristic: start with $\sqrt{n}$ (square root of the number of training samples) as a candidate for k.

k-Nearest Neighbors (kNN)

Q3: Suppose we use kNN with k=3. A test point has nearest neighbors with labels {A, B, B}. What class will kNN assign?

The majority vote is B, so the test point is classified as B.

k-Nearest Neighbors (kNN)

Q4: Why might Euclidean distance not always be the best choice for kNN?

Because features may have different scales or distributions. For example, “age in years” and “income in dollars” have very different ranges. Without normalization, income would dominate.

k-Nearest Neighbors (kNN)

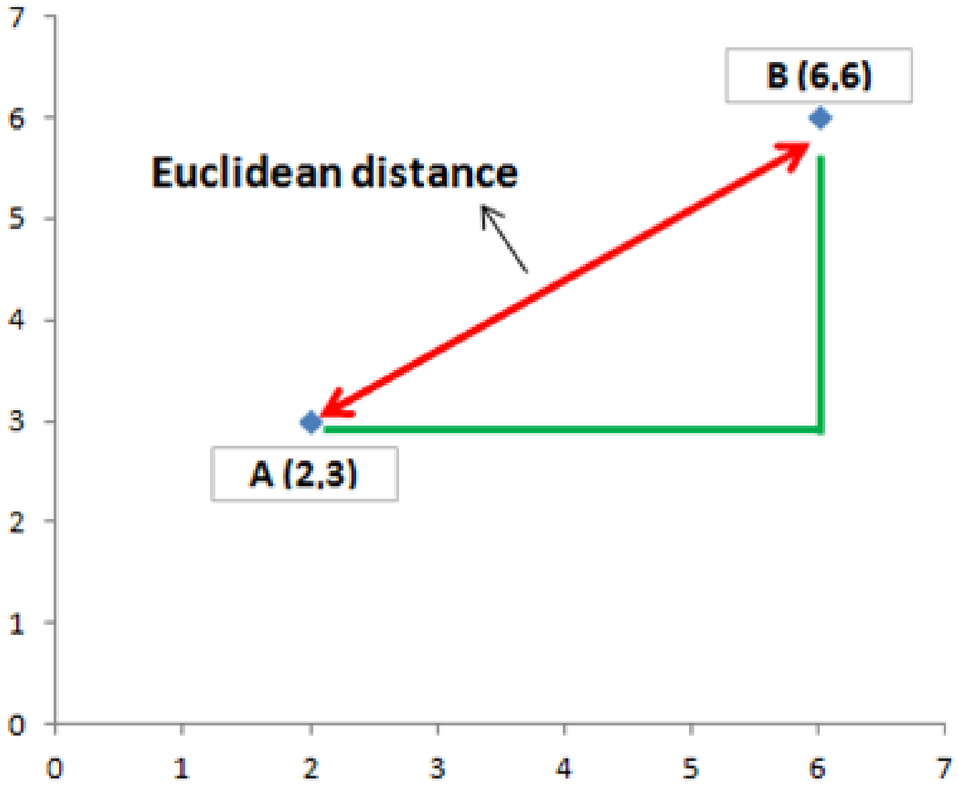
Euclidean Distance

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}.$$

Source: Wikipedia

Euclidean Distance gives the straight line distance between two coordinates in a vector space. This works great for numerical features in the input data.

k-Nearest Neighbors (kNN)



$$Euclidean\ distance\ (a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

k-Nearest Neighbors (kNN)

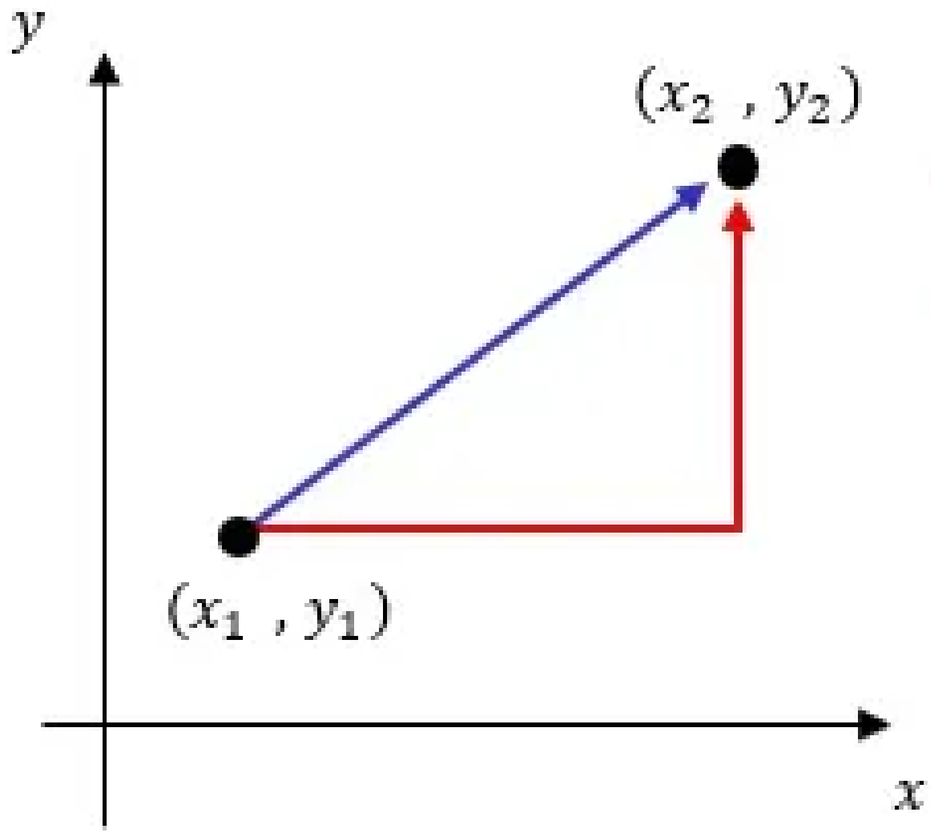
Manhattan Distance

$$\sum_{i=1}^{n} |p_i - q_i|$$

Source: Wikipedia

Manhattan Distance calculates the sum of absolute differences between the vector data points. This is good with categorical features. It is not as sensitive to outliers as Euclidean Distance.

k-Nearest Neighbors (kNN)



Manhattan Distance L^1

Euclidean Distance L^2

$$L^1 = |x_2 - x_1| + |y_2 - y_1|$$

$$L^2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

k-Nearest Neighbors (kNN)

Cosine Similarity

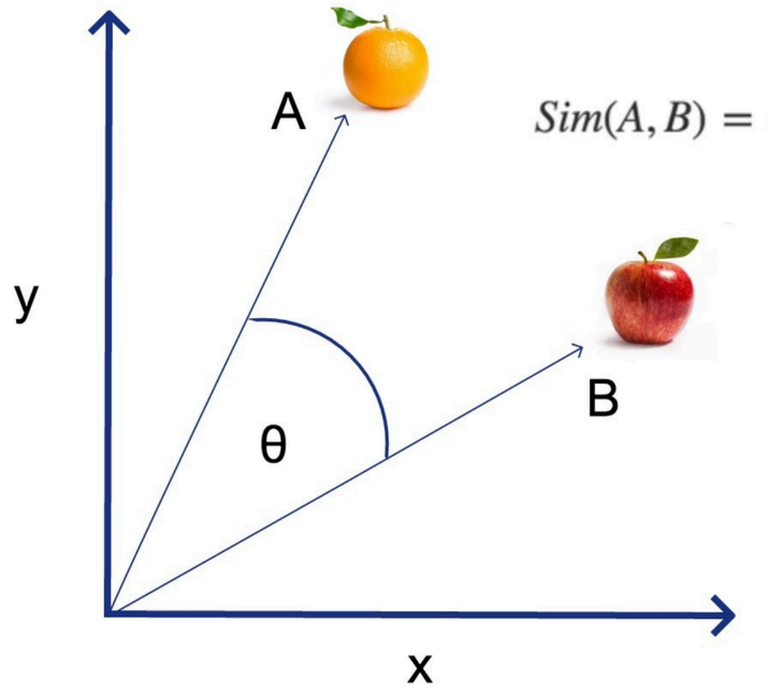
$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}},$$

Source: Wikipedia

This is commonly used for high dimensional vector data or text data and it calculates the distance using the cosine of the angle between two vectors.

k-Nearest Neighbors (kNN)

Cosine Similarity



$$Sim(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

k-Nearest Neighbors (kNN)

Q5: How does the decision boundary of kNN change as k increases?

- For k=1, boundaries are very jagged, closely following training data.
- As k increases, boundaries become smoother, reducing variance but potentially increasing bias.

decision tree

Q6: What criterion is typically used to split nodes in a decision tree?

Measures of impurity such as:
- Gini index (CART)
- Entropy / Information Gain (ID3, C4.5) These determine how well a split separates the data.

decision tree

Q6: What criterion is typically used to split nodes in a decision tree?

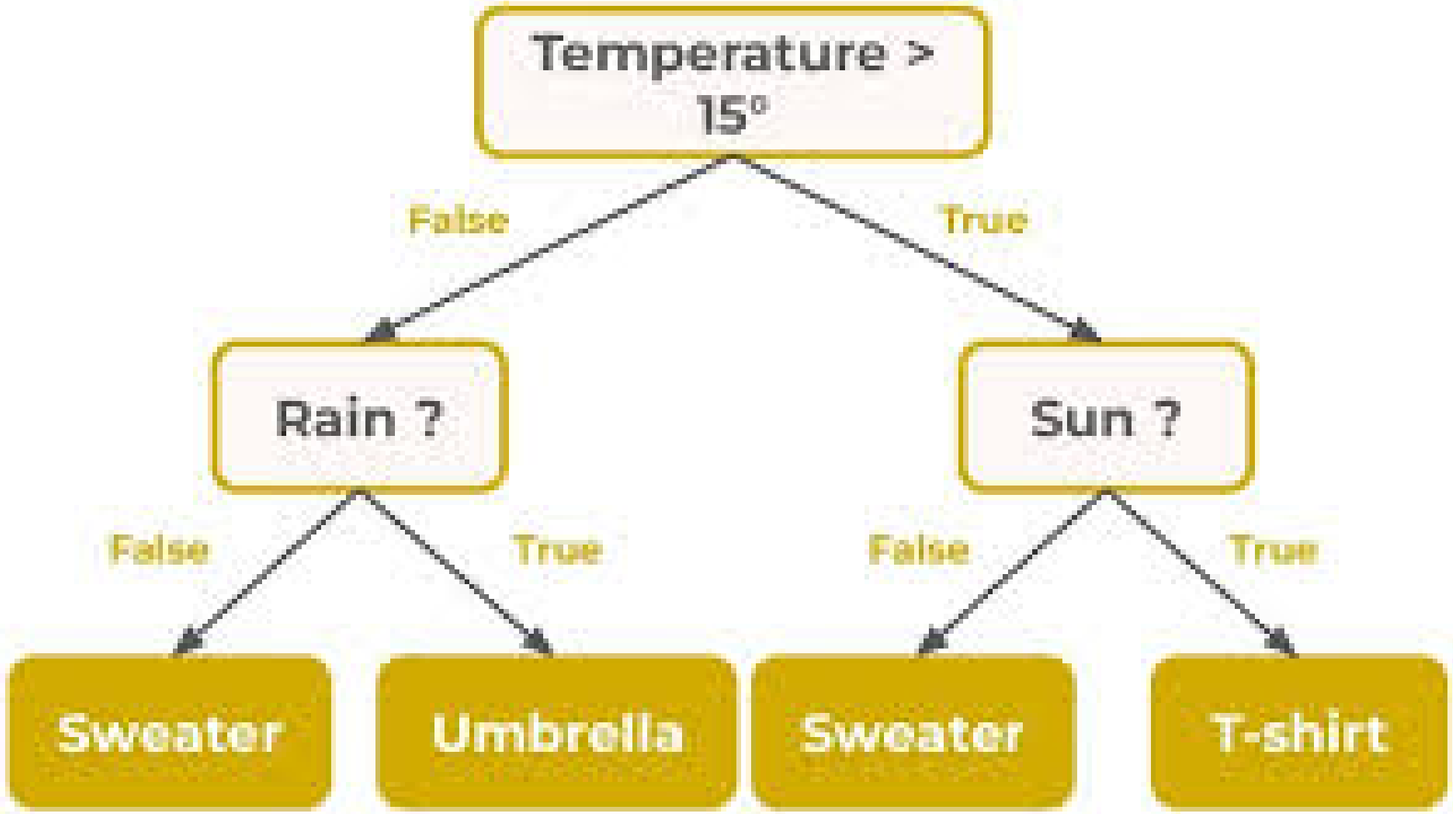
A decision tree is a model similar to a binary tree, used for classification or regression. Its working principle is like the “Twenty Questions” game:

- Start from the root node and ask a question based on a feature (e.g., “Is temperature > 30°C?”).
- Depending on the answer (Yes/No), move down a different branch.
- Continue asking new questions until reaching a leaf node, where you get the classification result or prediction value.

Therefore, the essence of a decision tree is: by continuously splitting the data, each leaf node contains samples that are as similar (pure) as possible.

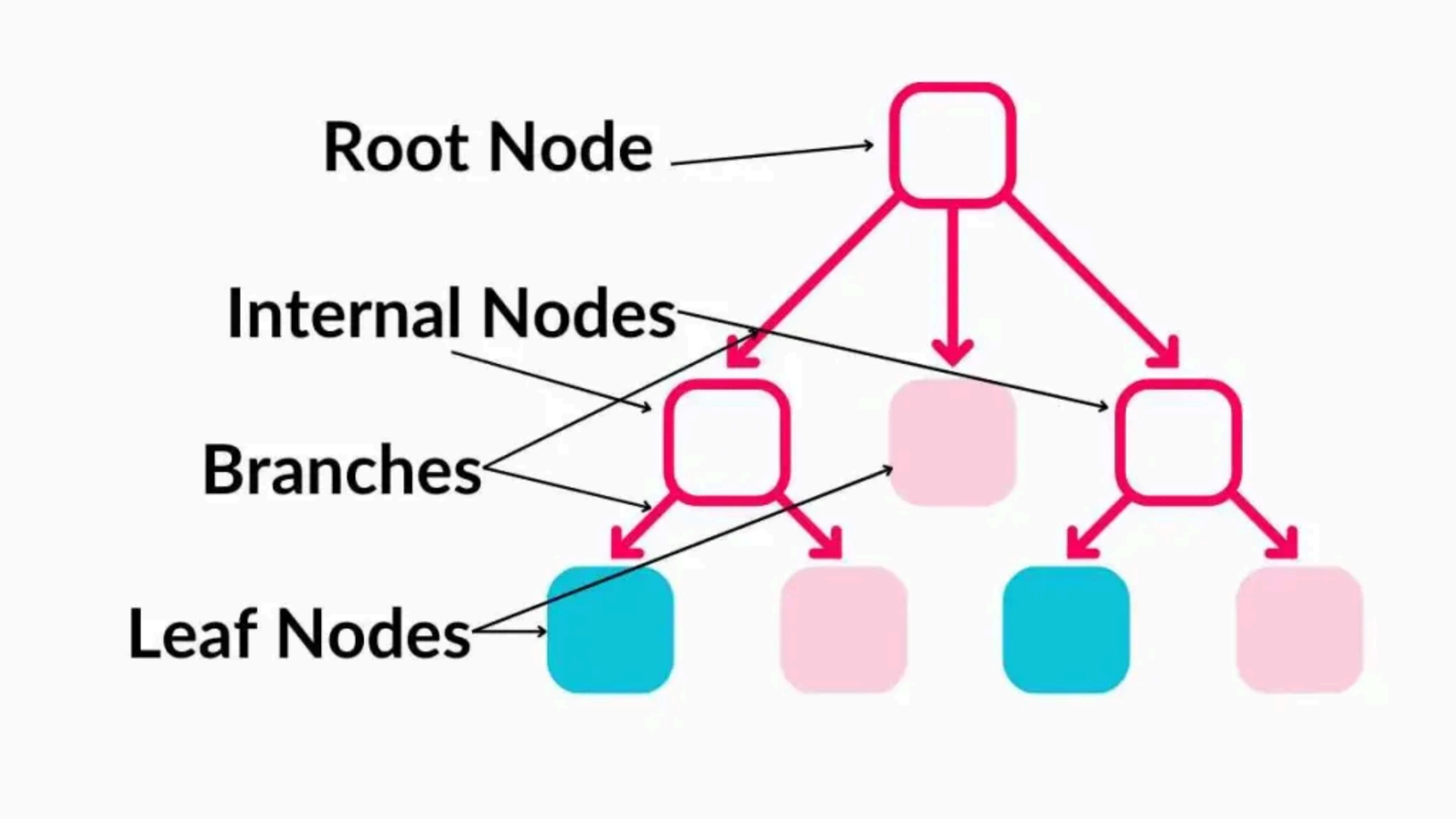
decision tree

Q6: What criterion is typically used to split nodes in a decision tree?



decision tree

Q6: What criterion is typically used to split nodes in a decision tree?



decision tree

Q6: What criterion is typically used to split nodes in a decision tree?

In a decision tree, each node represents a subset of the dataset. A good split should make the data as “pure” as possible, meaning that most samples in the same node belong to the same class. To measure this “purity,” the most common criteria are:

Gini Index

- Measures how mixed the classes are within a node.
- The lower the value, the purer the node.

Entropy

- Comes from information theory and represents the uncertainty of the dataset.
- The reduction in entropy after a split is called Information Gain, which evaluates the effectiveness of a split.

decision tree

Q6: What criterion is typically used to split nodes in a decision tree?

Example:
Suppose a node contains 10 samples:

- If all 10 are “cats,” then purity is the highest (Gini = 0, Entropy = 0).
- If 5 are “cats” and 5 are “dogs,” then the node is highly impure (Gini = 0.5, Entropy = 1).

When building a decision tree, the algorithm calculates possible splits (e.g., by “color,” “height,” etc.) and selects the one that most reduces impurity (i.e., increases purity).

decision tree

Q7: A dataset has 10 samples: 6 positive, 4 negative. What is the entropy of the root node?

$$Entropy = \sum_{i=1}^{c} -P_i * log_2(P_i)$$

$$Entropy = -\left(\frac{6}{10}\log_2\frac{6}{10} + \frac{4}{10}\log_2\frac{4}{10}\right)$$

$$= -(0.6 \cdot -0.737 + 0.4 \cdot -1.322) \approx 0.971$$

decision tree

Q7: A dataset has 10 samples: 6 positive, 4 negative. What is the Gini Index of the root node?

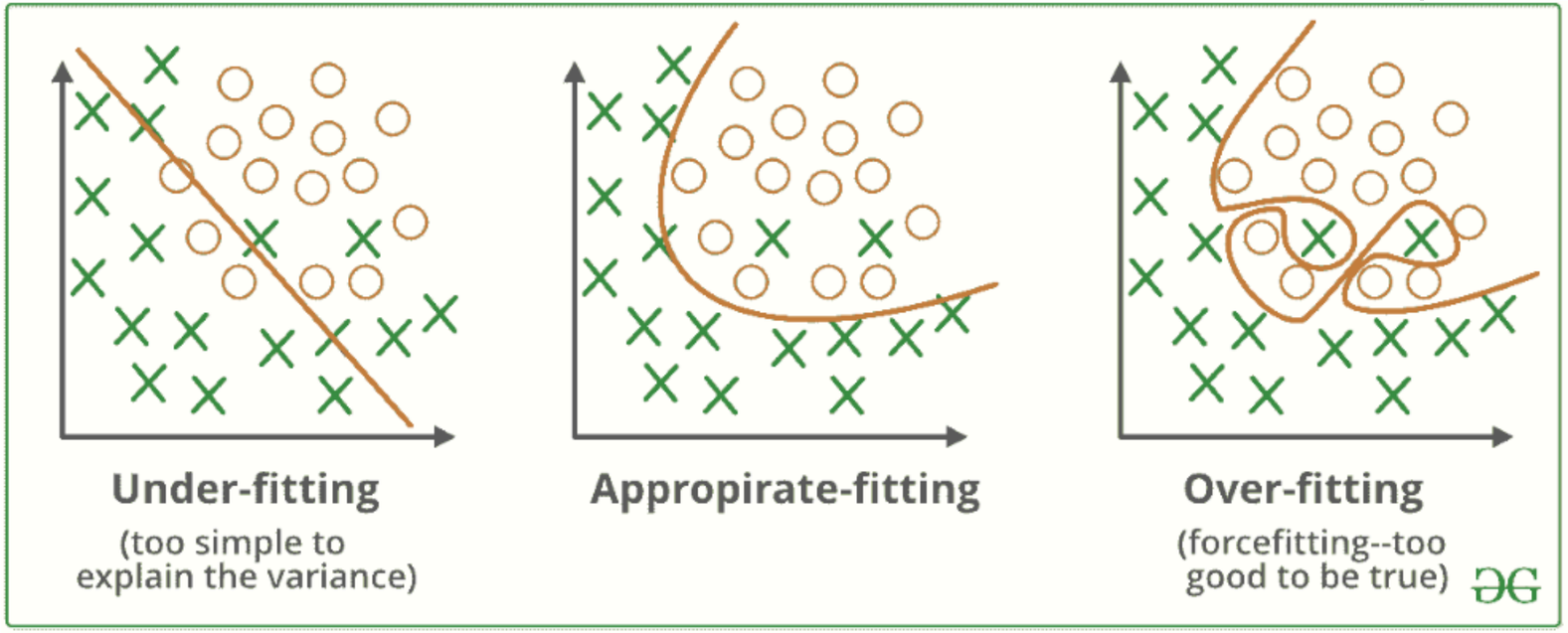
$$Gini = 1 - \sum_j p_j^2$$

$$Gini(root) = 1 - (0.6^2 + 0.4^2) = 0.48$$

decision tree

Q8: Why are decision trees prone to overfitting, and how can we prevent it?

- **Trees can keep splitting until each leaf has one sample → perfect training accuracy but poor generalization.**



decision tree

Q8: Why are decision trees prone to overfitting, and how can we prevent it?

Pre-pruning (restricting complexity while the tree is being built):

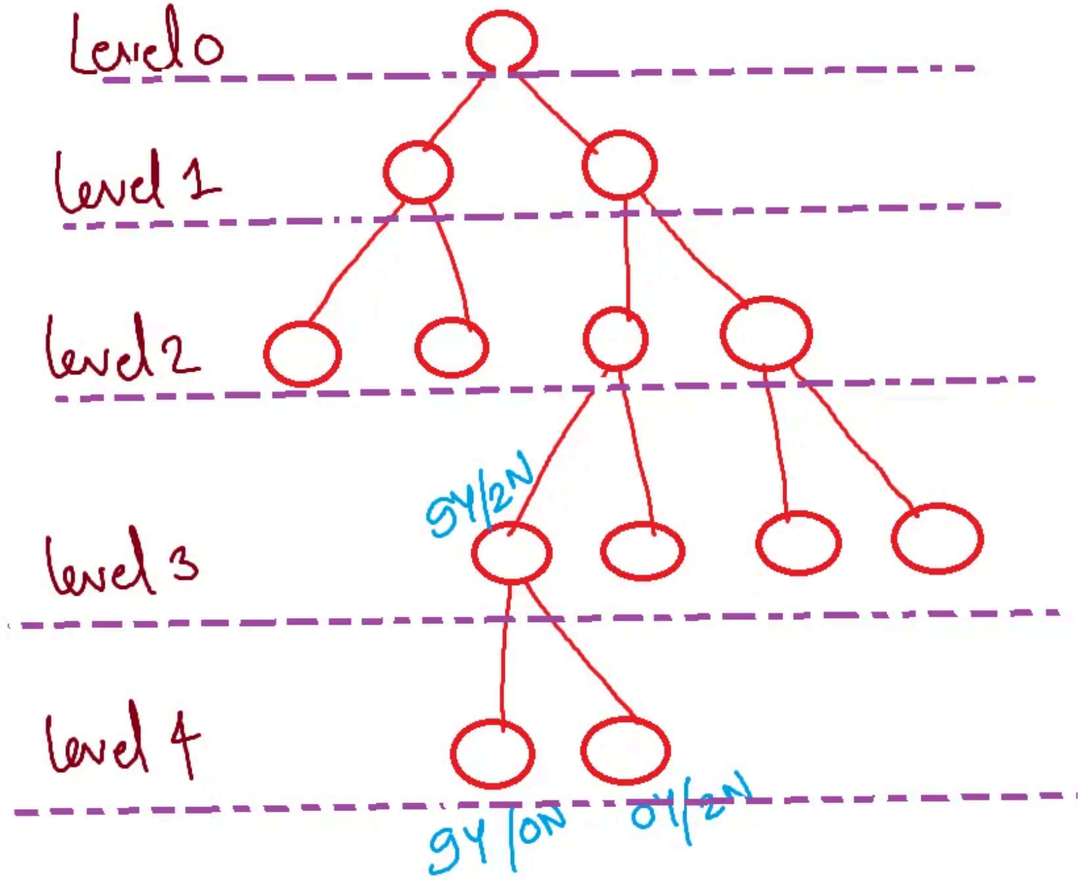
- Limit maximum depth (max_depth).
- Require a minimum number of samples per leaf or per split (min_samples_leaf, min_samples_split).
- Set a minimum threshold for information gain when splitting.

Post-pruning (grow a large tree first, then prune back):

- Build a full tree, then evaluate on a validation set to remove unnecessary branches.

decision tree

Visualizing post-pruning with an example



At Level 3, the algorithm notices that the node already has a very strong class probability (say, 95% "yes" and 5% "no").

Continuing to split into Level 4 might create extra complexity without much gain in predictive power, since the classification is already very confident.

So, instead of building more branches (which risks overfitting to noise in the data), we stop growing the tree early and mark that node as a leaf with the label “yes.”

decision tree

Q9: Compare kNN and decision trees in terms of training time, prediction time, and interpretability.

- Training time: kNN → very fast (just store data); Decision tree → slower (splitting + building).
- Prediction time: kNN → expensive (distance to all points); Decision tree → fast (just follow path).
- Interpretability: Decision tree → high (can visualize rules); kNN → low (hard to explain classification).

decision tree

Q9: Compare kNN and decision trees in terms of training time, prediction time, and interpretability.

- Training time: kNN → very fast (just store data); Decision tree → slower (splitting + building).
- Prediction time: kNN → expensive (distance to all points); Decision tree → fast (just follow path).
- Interpretability: Decision tree → high (can visualize rules); kNN → low (hard to explain classification).

decision tree

Q10: Suppose you are building a medical diagnosis system.

Would you prefer kNN or decision trees? Why?

: Likely decision trees, because:
- They are interpretable (important for doctors/patients).
- Provide clear decision rules.
- Faster at inference.

But kNN could be useful if you have a small dataset and interpretability is less critical.

Q11: Exercise. We want to classify whether a fruit is an Apple (A) or Orange (O) based on Weight (grams) and Sweetness (1–10 scale). Below is the dataset.

Sample	Weight (g)	Sweetness	Label
1	150	8	A
2	170	7	A
3	140	6	A
4	130	7	A
5	180	3	O
6	200	4	O
7	210	5	O
8	190	2	O

Q11: Exercise. How do you classify a new fruit with Weight = 160g, Sweetness = 6.

Steps: 1. Compute Euclidean distance to all training samples. Example distance to sample 1:

$$d = \sqrt{(160 - 150)^2 + (6 - 8)^2} = \sqrt{100 + 4} = \sqrt{104} \approx 10.2$$

2. Find the k nearest neighbors.
For k=3, find the 3 nearest neighbors of this new fruit (will be mostly Apples (A)).
3. Do majority voting → predicted label = Apple (A).

Exercise 2 - Decision Tree: Build the root node of a decision tree using Gini index.

Steps:

1. Compute Gini at root (before splitting):

Apples = 4, Oranges = 4 →

$$Gini = 1 - \left(\frac{4}{8}\right)^2 - \left(\frac{4}{8}\right)^2 = 0.5$$

2. Try splitting on Weight (e.g., threshold 160g).
    - Left group (<160): Samples {1,3,4} → all Apples → Gini = 0.0
    - Right group (≥160): {2,5,6,7,8} → 1 Apple, 4 Oranges →

      $$Gini = 1 - (1/5)^2 - (4/5)^2 = 0.32$$

    - Weighted Gini = (3/8 * 0.0) + (5/8 * 0.32) = 0.20
3. Information gain = 0.5 – 0.20 = 0.30 (good split!).
4. So the first split is on Weight < 160g → Apple, else → Orange (mostly).