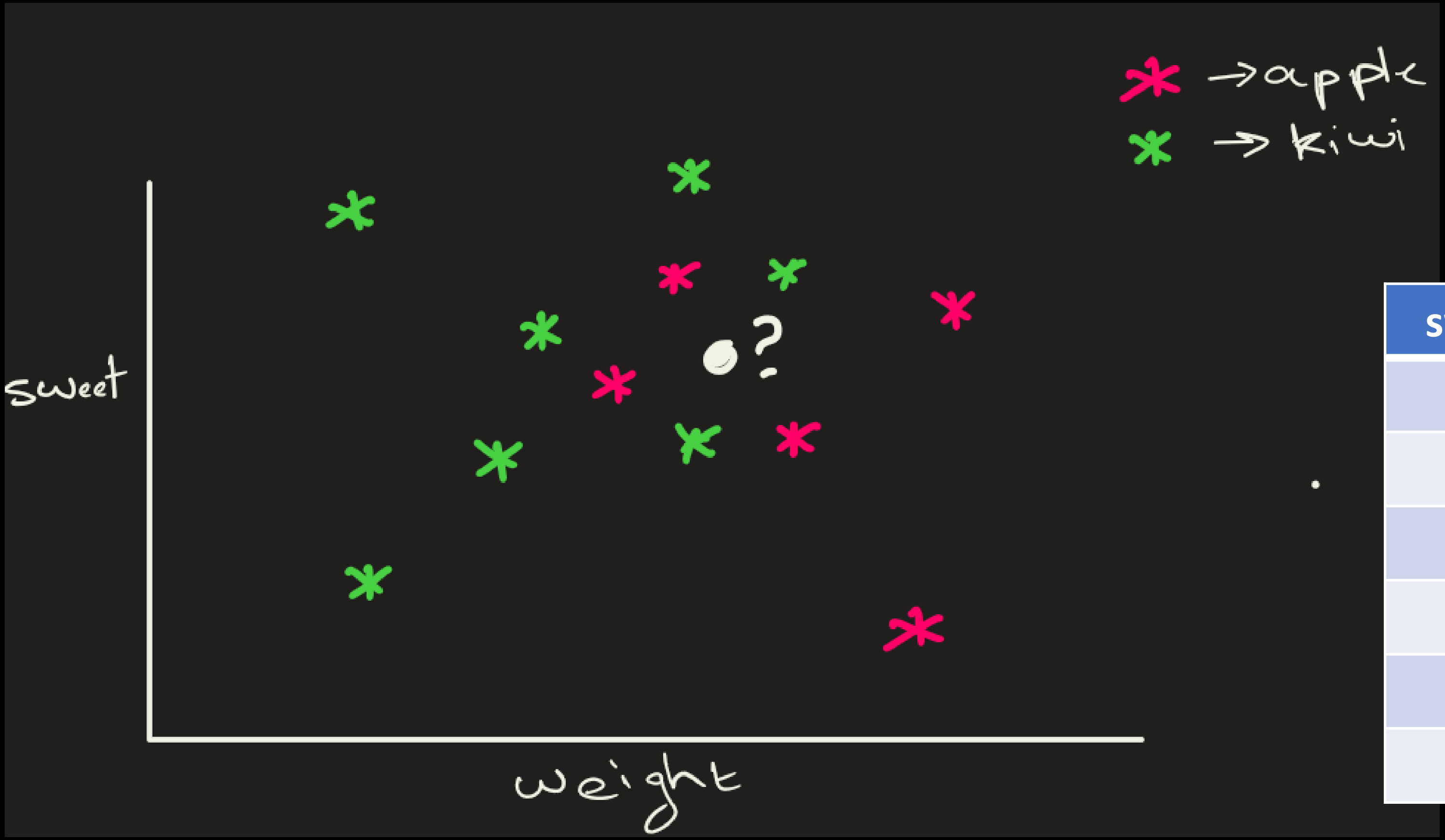
K-Nearest Neighbor (KNN)

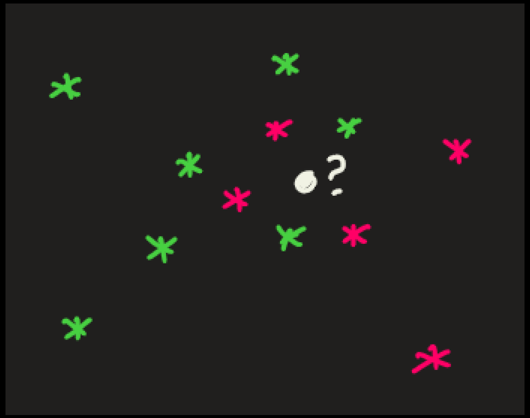
Problem:
You are given labeled data for two types of fruits—kiwi and apple—along with their features: sweetness and weight. Now, you are given a new fruit with known sweetness and weight values, and your task is to determine which fruit category it belongs to.



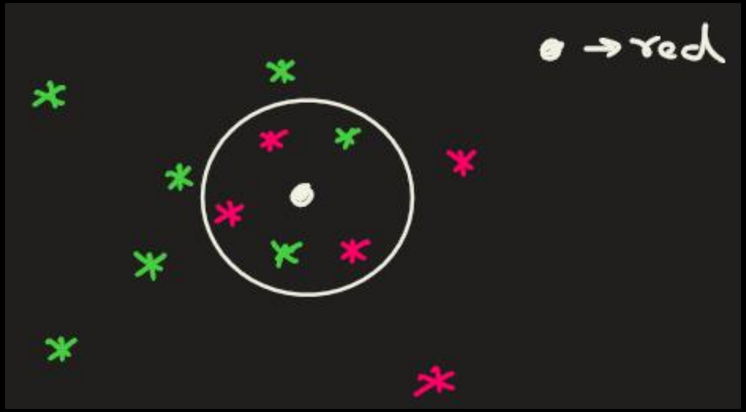
sweetness	weight	label
2	20	apple
4	32	kiwi
3	33	kiwi
1	25	apple
…	…	…
5	**43**	**?**

K-Nearest Neighbor (KNN)

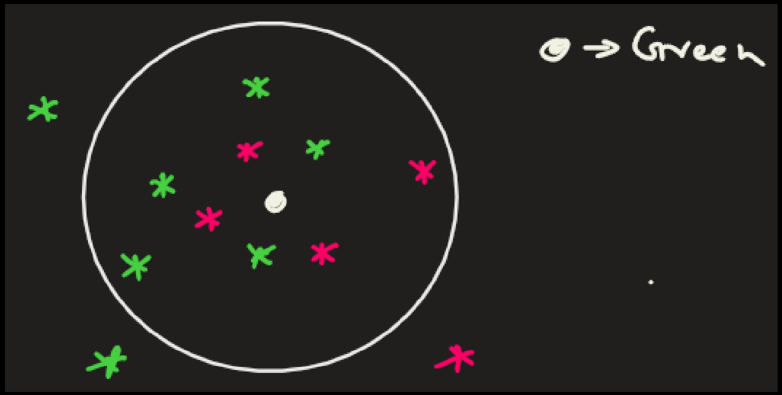
Idea: For a given test point x, **find k nearest neighbor**. Assign x to the class that is most common among the k neighbors.



When K=5, there are 3 red and 2 green.
So, the point is assigned red



When K=9, there are 4 red and 5 green.
The point is assigned green

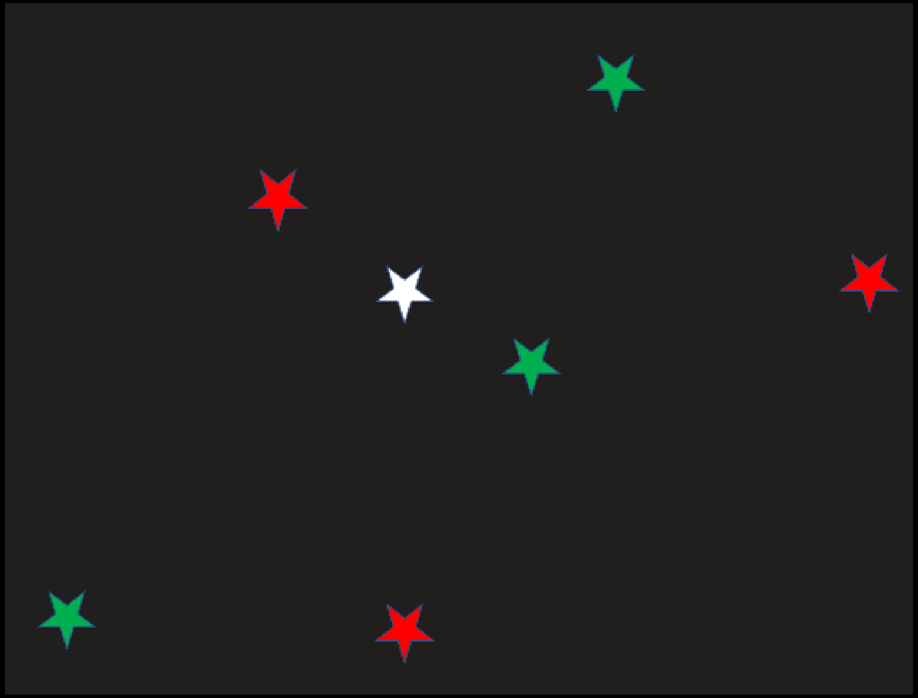


K-Nearest Neighbor (KNN): Algorithm

INPUT: data, K
OUTPUT: predicted class

1. Load the data

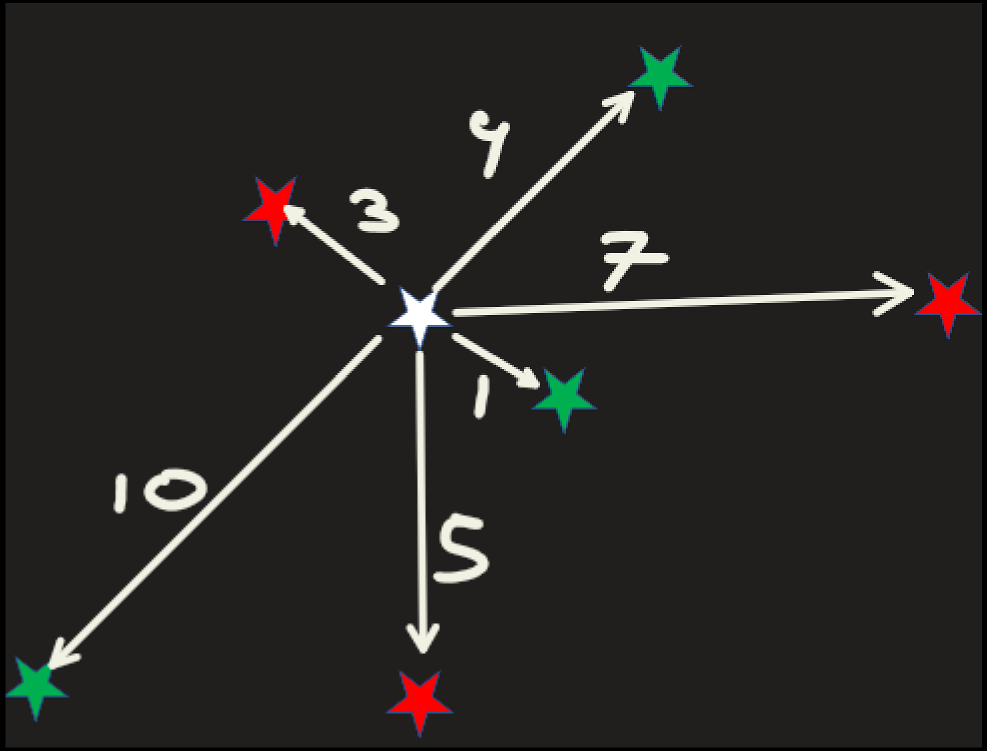
Input: 6 training data points and K=3.
The labels on data are green and red.
The test data is show in white



K-Nearest Neighbor (KNN): Algorithm

2. For getting the predicted class, iterate from 1 to total number of training data points. Calculate the distance between test data and each data-point of the dataset.

The distances are shown next to arrow.

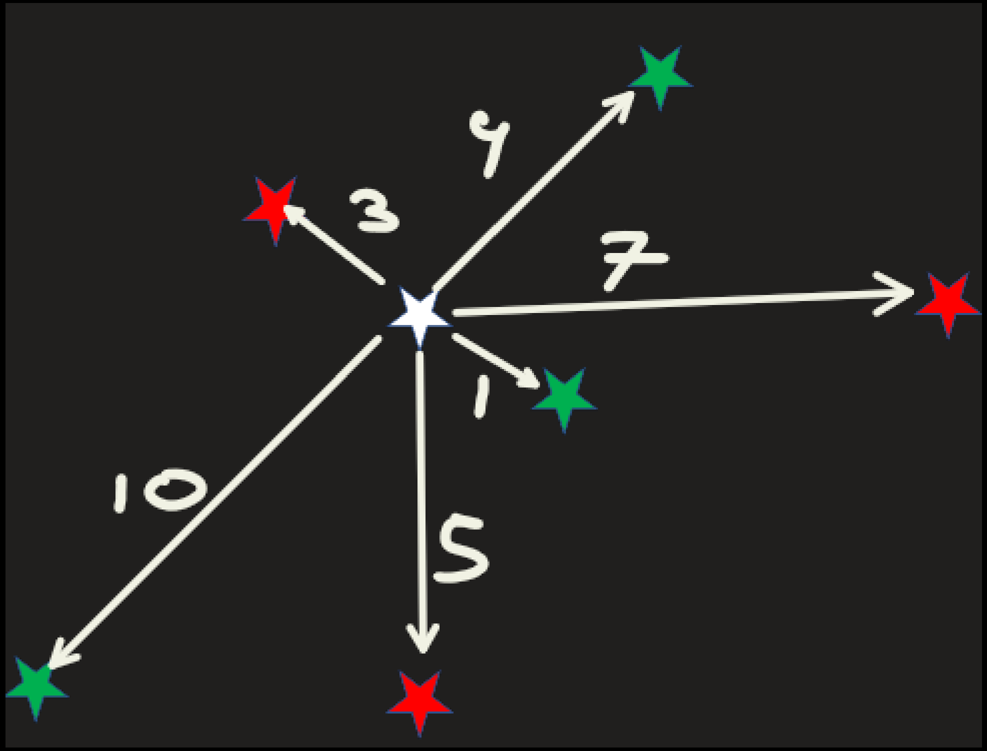


K-Nearest Neighbor (KNN): Algorithm

3. Sort the calculated distances in ascending order based on distance values

4. Get top k rows from the sorted array

5. Get the most frequent class of these rows. This is your predicted class



The points are arranged in ascending order based on distances (with their colors) :

(1 → green)
(3 → red)
(4 → green)
(5 → red)
(7 → red)
(10 → green).

Top K=3 have **green** as majority. So, test data is **green**.

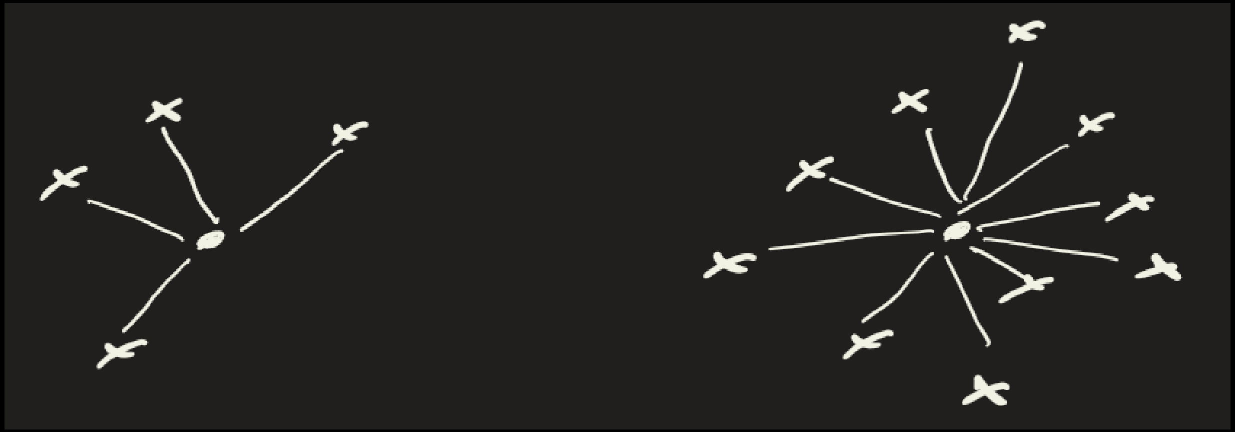
Dataset Characteristics Suitable for KNN

Small to Medium-Sized Datasets

KNN is a **lazy learner** (it **doesn't train a model** but stores the dataset and makes predictions at query time).

Works best when the dataset is **not too large**, typically **less than 100,000 samples**.
Why? Large datasets make KNN slow because it needs to compute distances for every new prediction.

Example: In 1st figure, the 4 distances are calculated. In 2nd figure, 10 distance are calculated

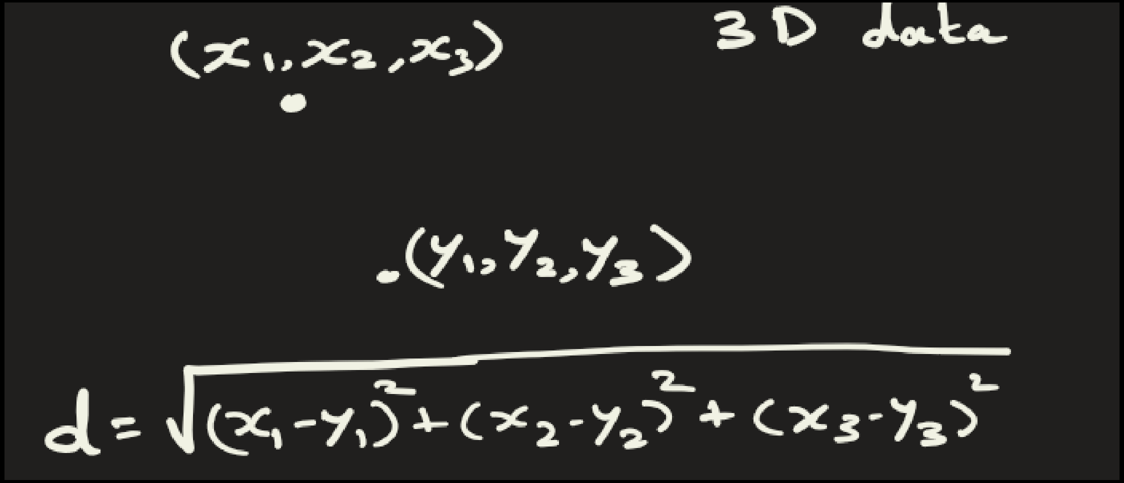


Dataset Characteristics Suitable for KNN

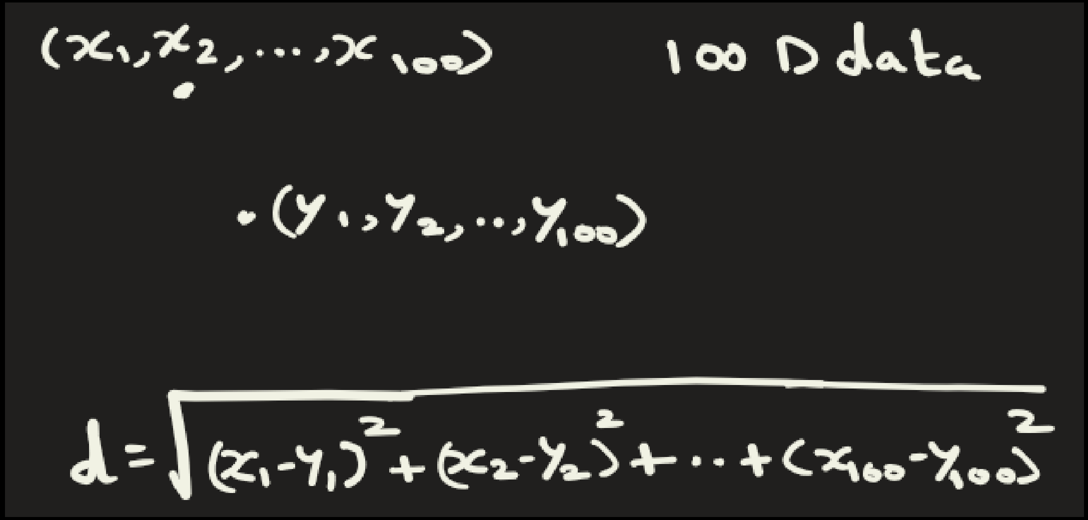
Low-Dimensional Data (i.e. Fewer Features)

KNN performs best when the dataset has a **manageable number of features** (e.g., ≤ 50 dimensions).
Why? High-dimensional data suffers from the **curse of dimensionality**, making Euclidean distance less effective.

Example:



$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}$$



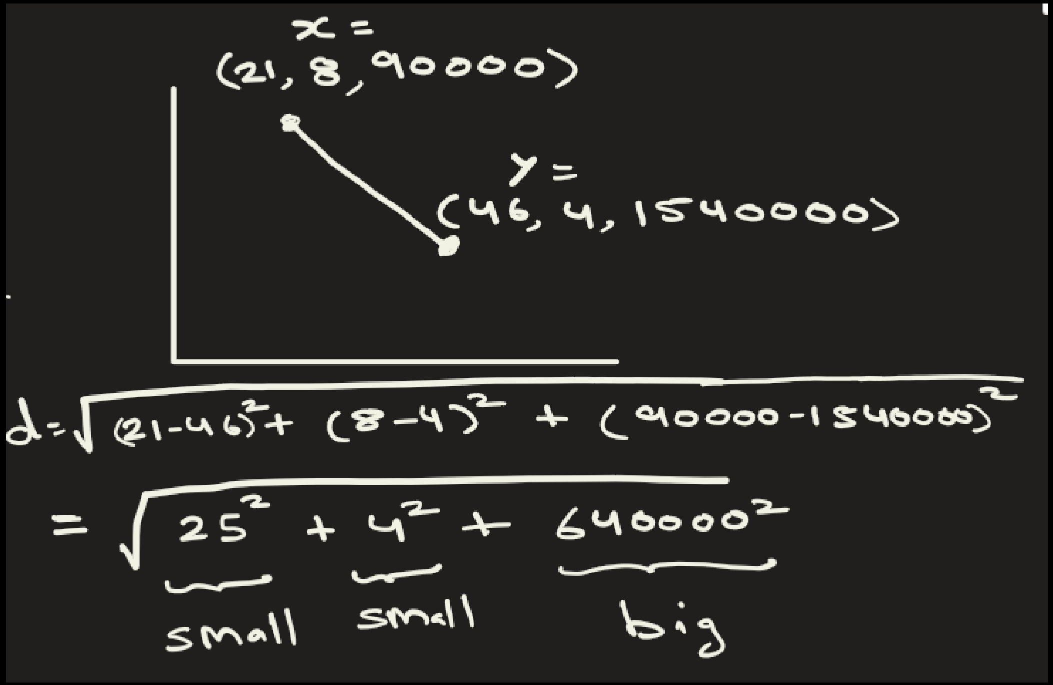
$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_{100} - y_{100})^2}$$

Dataset Characteristics Suitable for KNN

Well-Scaled Data

KNN is sensitive to scale because distance changes if one feature dominates.
The features should be normalized or standardized (e.g., Min–Max, StandardScaler)

Example: Suppose we have data with 3 features: age, work-hours and income. Here age and work-hours are low range numbers. Income is a high range number.
If we have x = (21, 8, 90000) and y = (46, 4, 1540000) and use Euclidean distance, then distance is going to be controlled by income and not much influenced by age and work-hours.



$$d = \sqrt{(21-46)^2 + (8-4)^2 + (90000-1540000)^2}$$

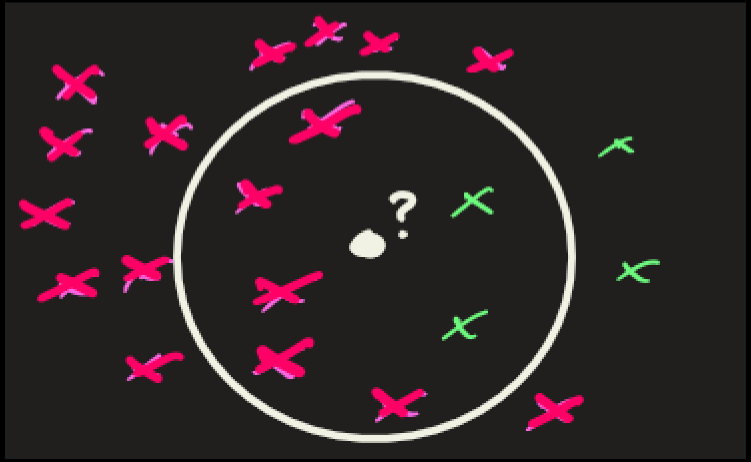
$$= \sqrt{\underbrace{25^2}_{\text{small}} + \underbrace{4^2}_{\text{small}} + \underbrace{640000^2}_{\text{big}}}$$

Dataset Characteristics Suitable for KNN

Balanced Classes in Classification Problems

Performs best when the dataset has **balanced class distributions** in classification tasks.
Why? If one class is overrepresented, majority voting can lead to biased predictions.

Example:
Here the test data would be classified as red because red dominates. This point could have been green but majority dominated.

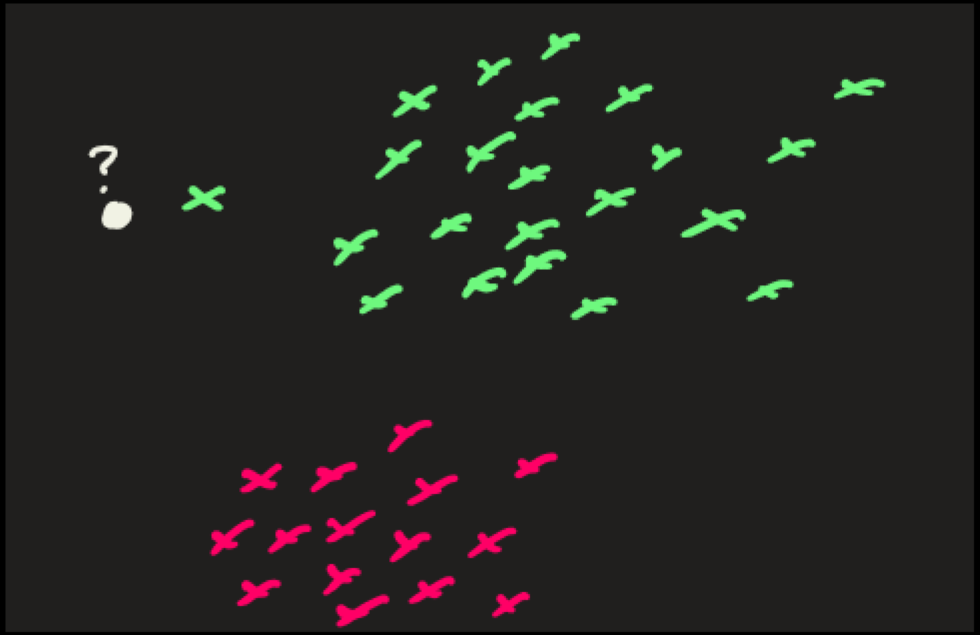


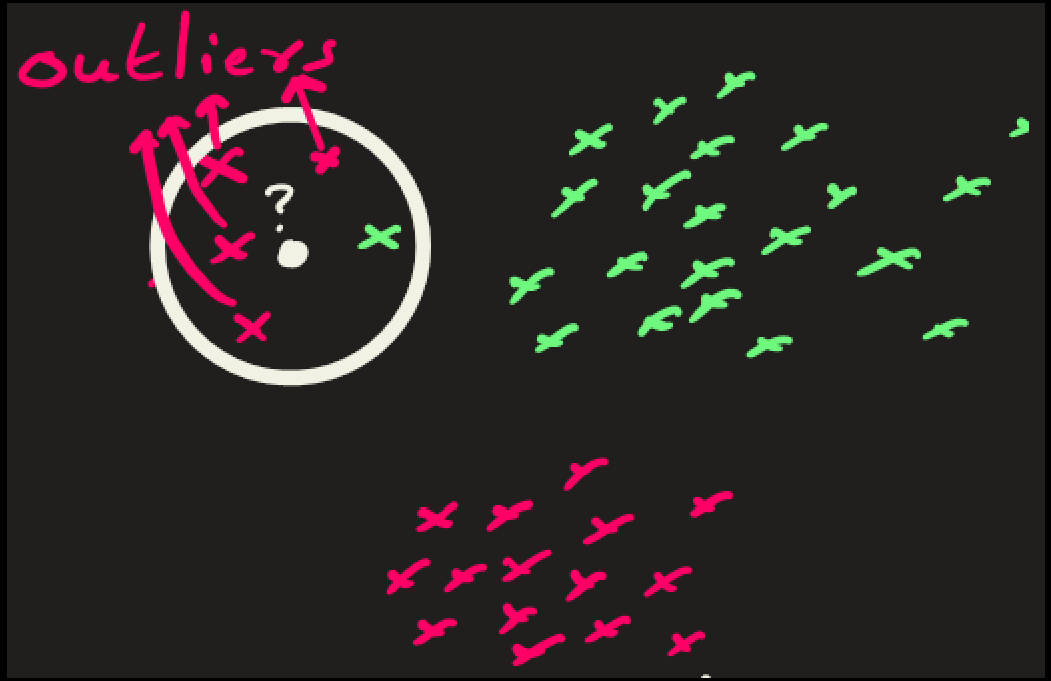
Dataset Characteristics Suitable for KNN

Outliers in Classification Problems

Performs best when dataset does not have many outliers. If there are many outliers, then the presence of test point near these outliers would give wrong classification.

Example: In 1st figure, we see that the point should be green. In 2nd figure, because of outliers, it would be classified as red.

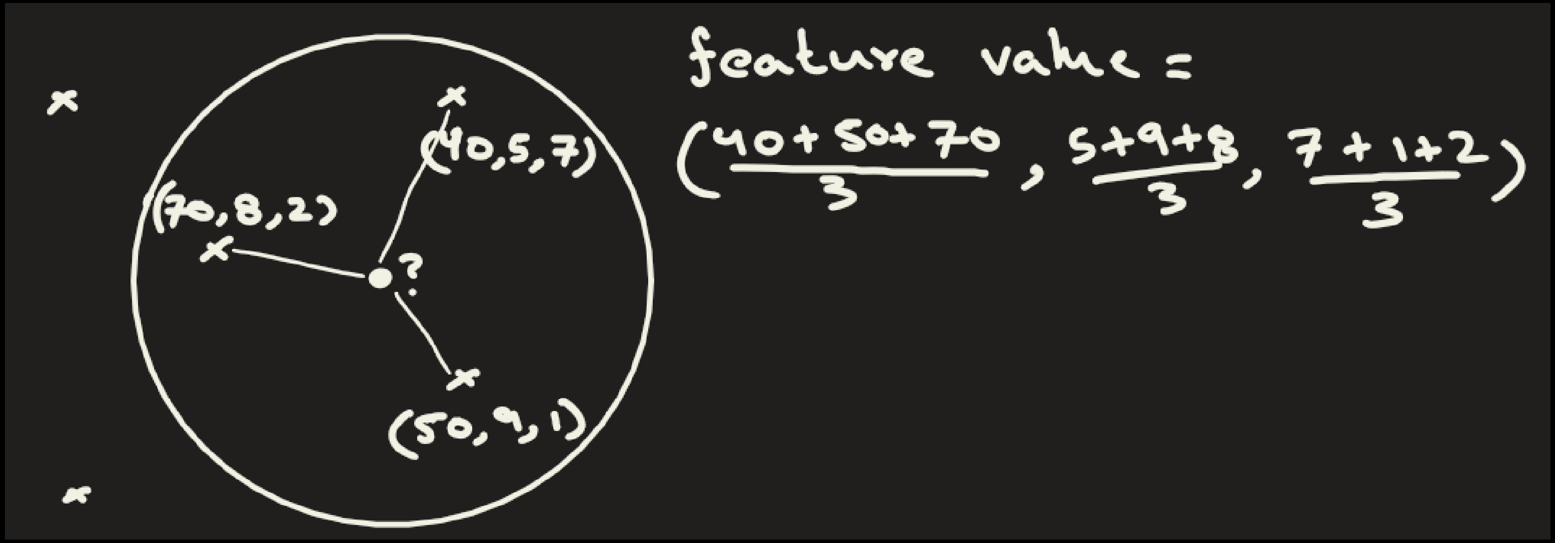




Dataset Characteristics Suitable for KNN

Continuous or Categorical Data

Works well with **numerical data** (e.g., age, salary, height).
Can handle **categorical data** but requires proper encoding (e.g., one-hot encoding).



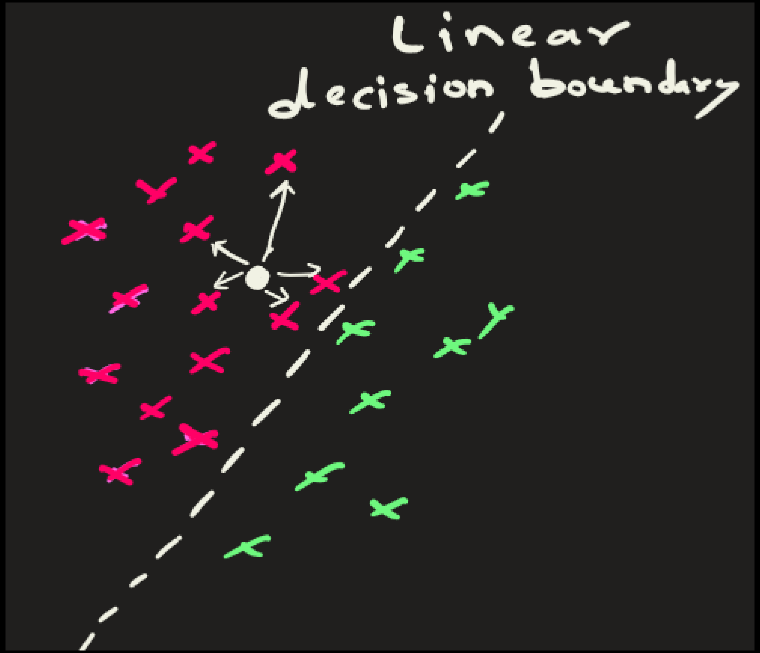
$$\left(\frac{40+50+70}{3}, \frac{5+9+8}{3}, \frac{7+1+2}{3}\right)$$

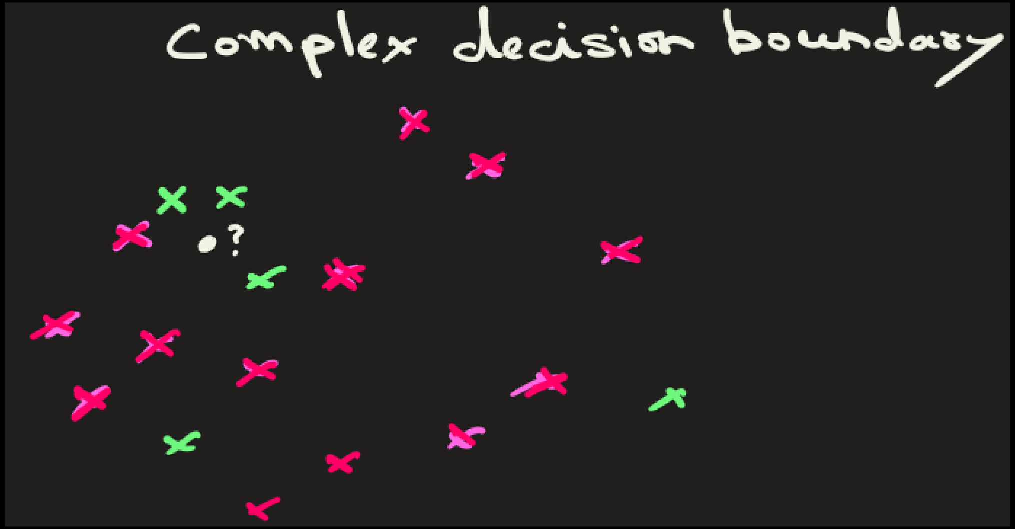
Dataset Characteristics Suitable for KNN

Non-Linearly Separable Data

Works well when **decision boundaries are complex and non-linear**.
Why? KNN captures patterns in data without assuming a fixed decision boundary.

Example: In 1st figure the point can be classified as red using linear decision boundary. In 2nd figure, the point is classified as green using KNN





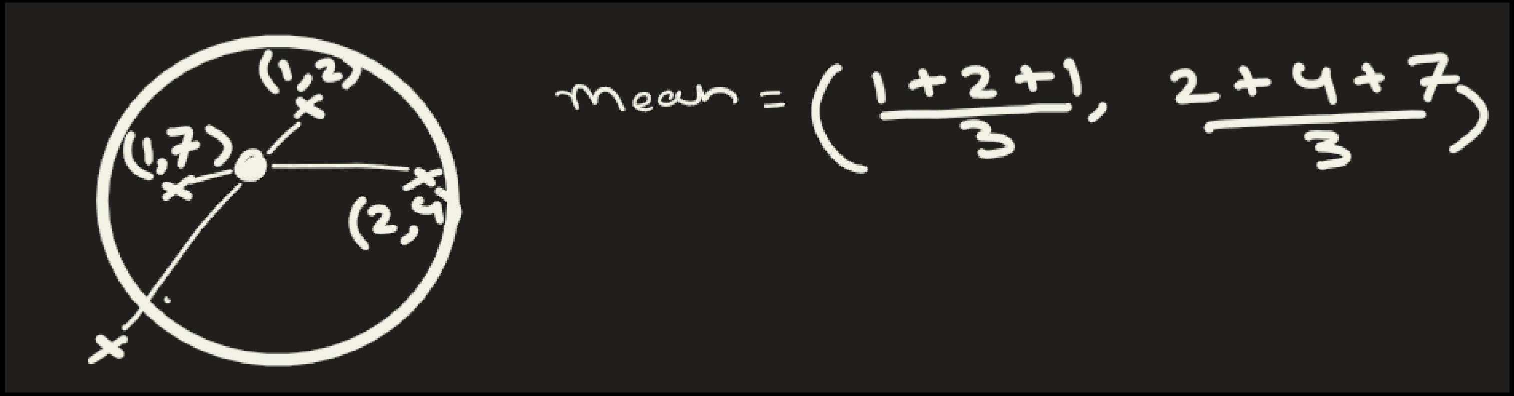
K-Nearest Neighbor (KNN): **Application**

1. Filling in missing values

Use neighbor’s values to fill the missing one

- **Numerical features** → use the **mean** or **median** of neighbors.

Example: If K=3 neighbors have feature values as (1,2), (2,4), (1,7) then using mean, the test data feature value is



$$\text{mean} = \left(\frac{1+2+1}{3}, \frac{2+4+7}{3} \right)$$

- **Categorical features** → use the **mode** (most common category)

K-Nearest Neighbor (KNN): **Application**

2. Recommendation Systems

KNN is widely used for **item-based or user-based collaborative filtering**.

Example:

- Suggesting movies on Netflix based on “nearest” similar users.
- Suggesting products on Amazon based on similar shopping patterns.

3. Medical Diagnosis & Healthcare Analytics

KNN helps classify patients based on symptom similarity.

Example:

- Predicting whether a tumor is **benign or malignant** (Breast Cancer dataset).
- Diagnosing diseases by comparing patient records.

EXTRA