KNN (k-Nearest Neighbors)

-MUKESH KUMAR

What is KNN

- The k-Nearest Neighbors (k-NN) algorithm is a simple, yet powerful, machine learning algorithm used primarily for classification tasks.
- It can also be used for regression, but its most common application is in classification.

Overview of k-NN Algorithm

- **Type:** Supervised learning algorithm.
- **Use cases:** Classification (most common) and regression.
- **Concept:** The algorithm classifies a data point based on how its neighbors are classified.

HOW K-NN WORKS

Training Phase

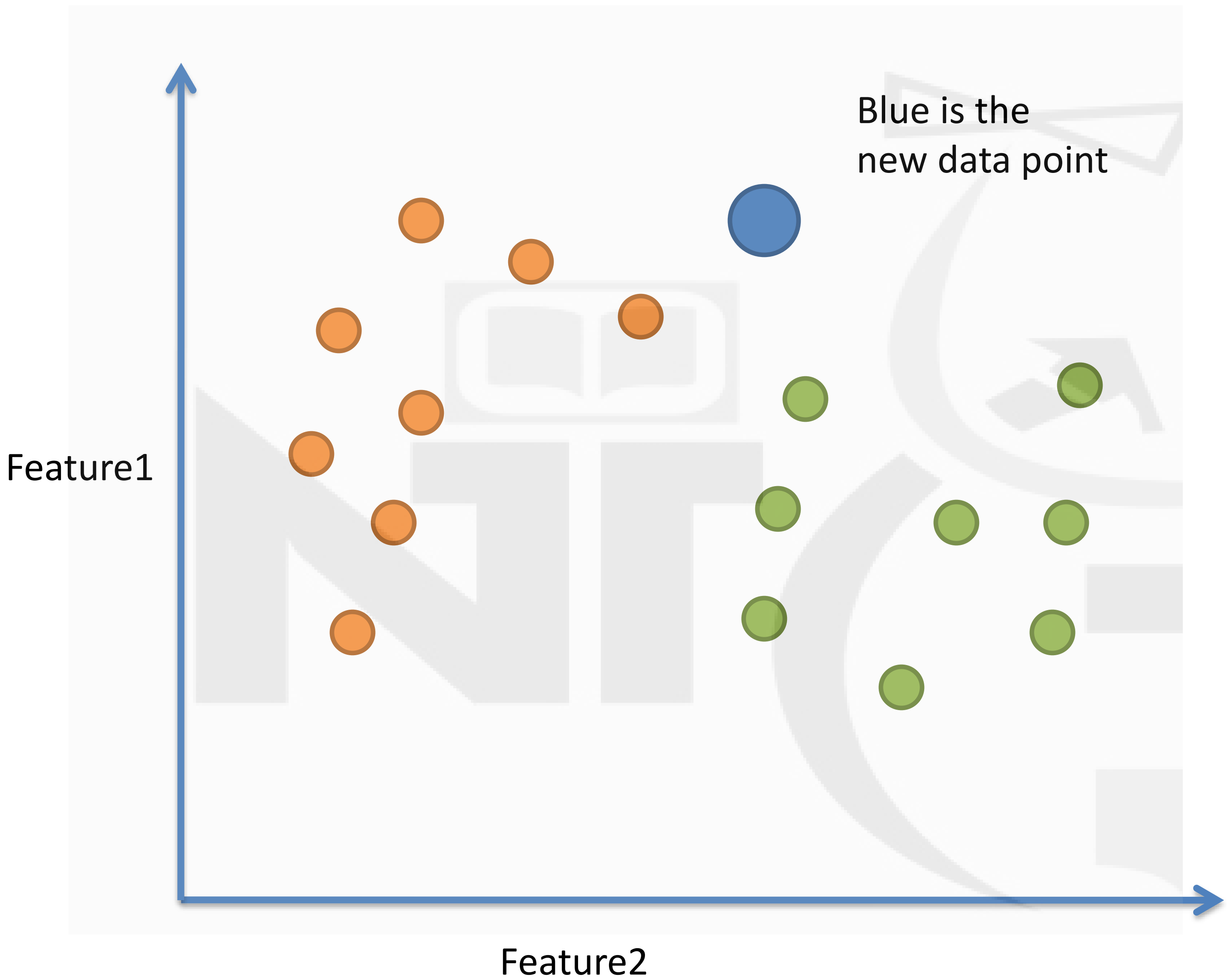
- The algorithm simply stores the training examples, which consist of feature vectors and their corresponding labels.

- **Lazy learner:** KNN does not build a model during the training phase. Instead, it stores the entire training dataset and performs computations during the classification phase, which can be computationally intensive

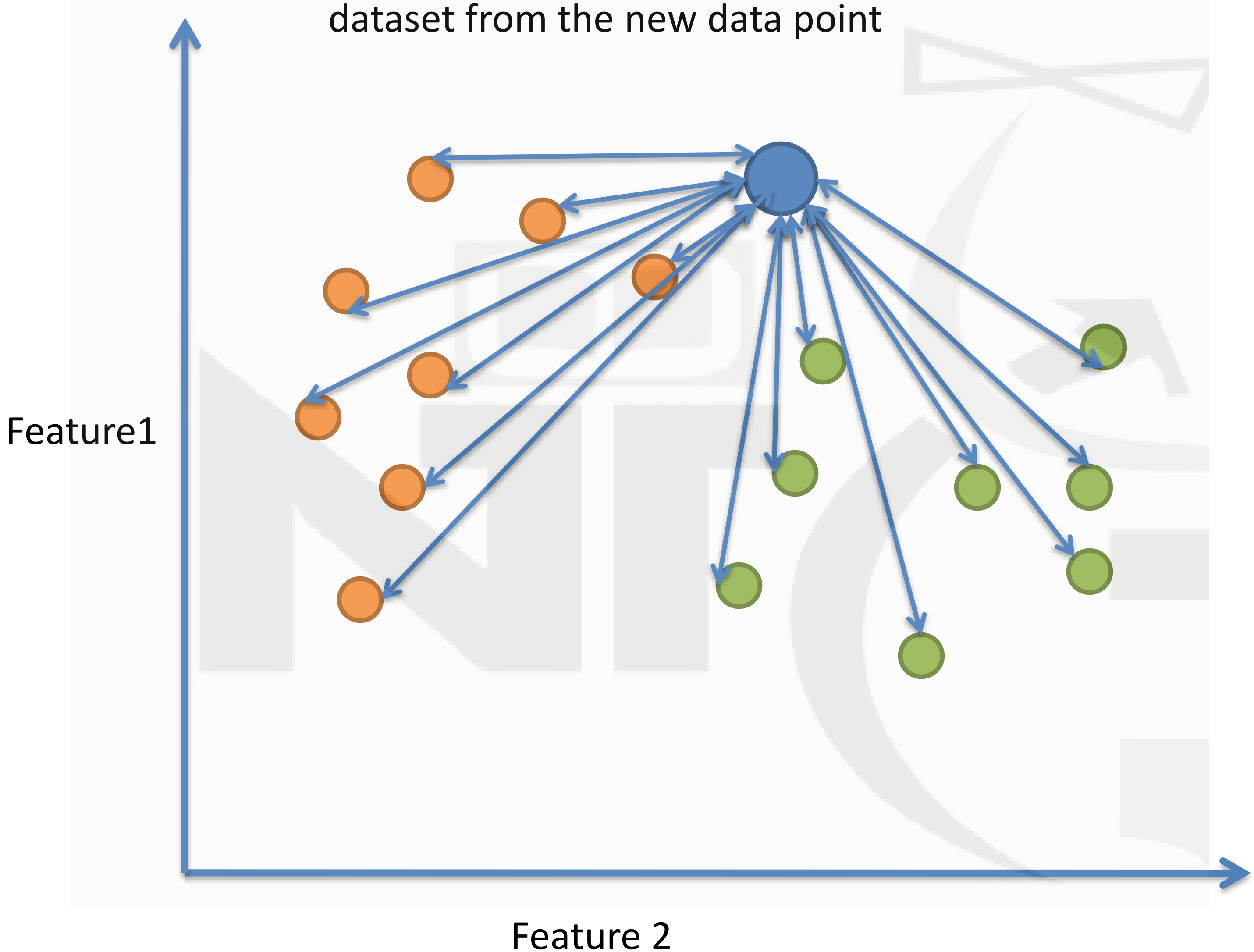
Prediction Phase

- For a new data point (query point), KNN identifies the *k* nearest neighbors from the training data based on a distance metric (commonly Euclidean distance).
- The algorithm then assigns a class label to the new point based on the majority class among its *k* nearest neighbors.
- For regression tasks, it typically predicts the average of the values from the nearest neighbors

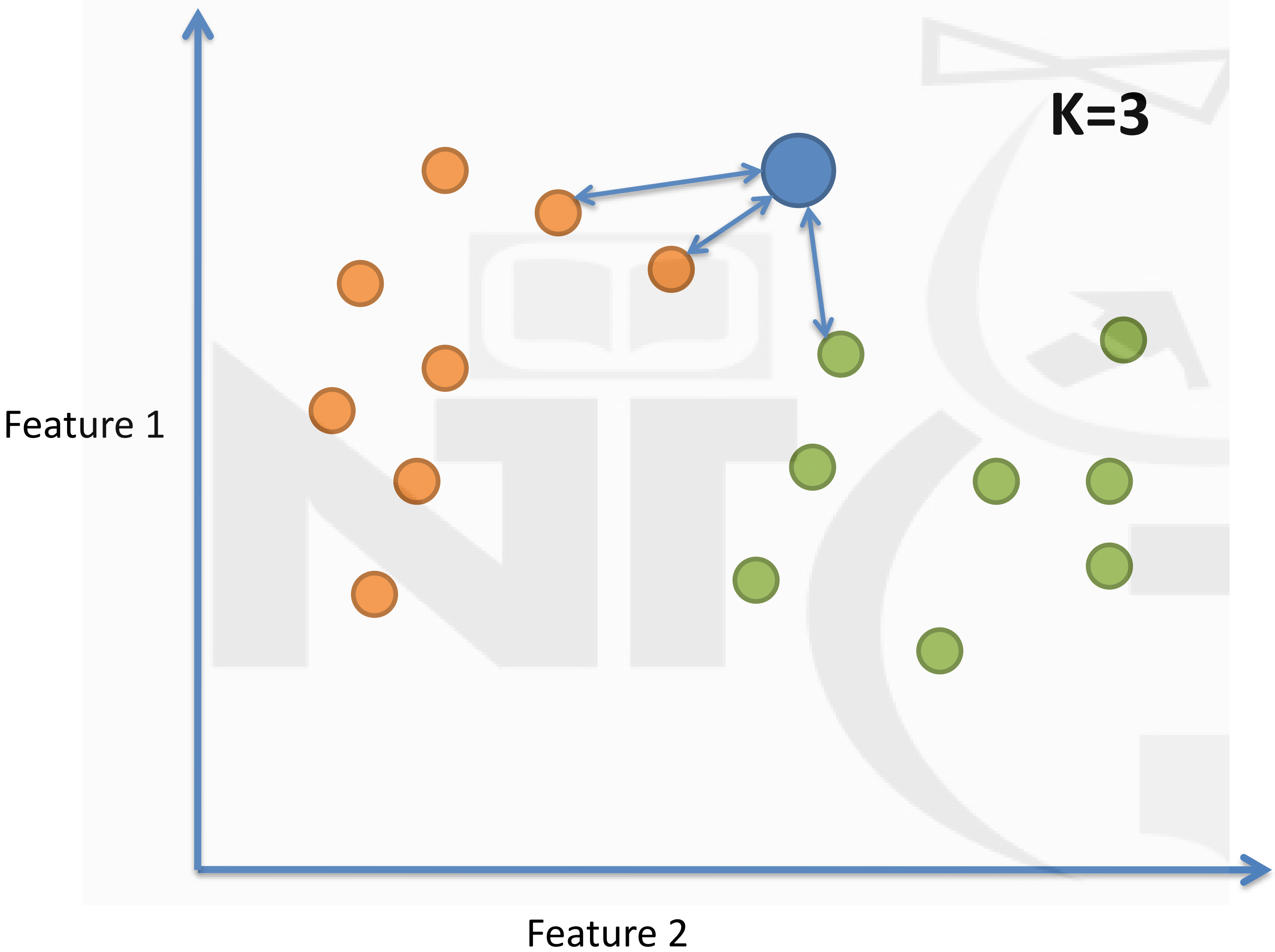
UNDERSTANDING HOW KNN WORKS


Blue is the
new data point
Feature1
Feature2

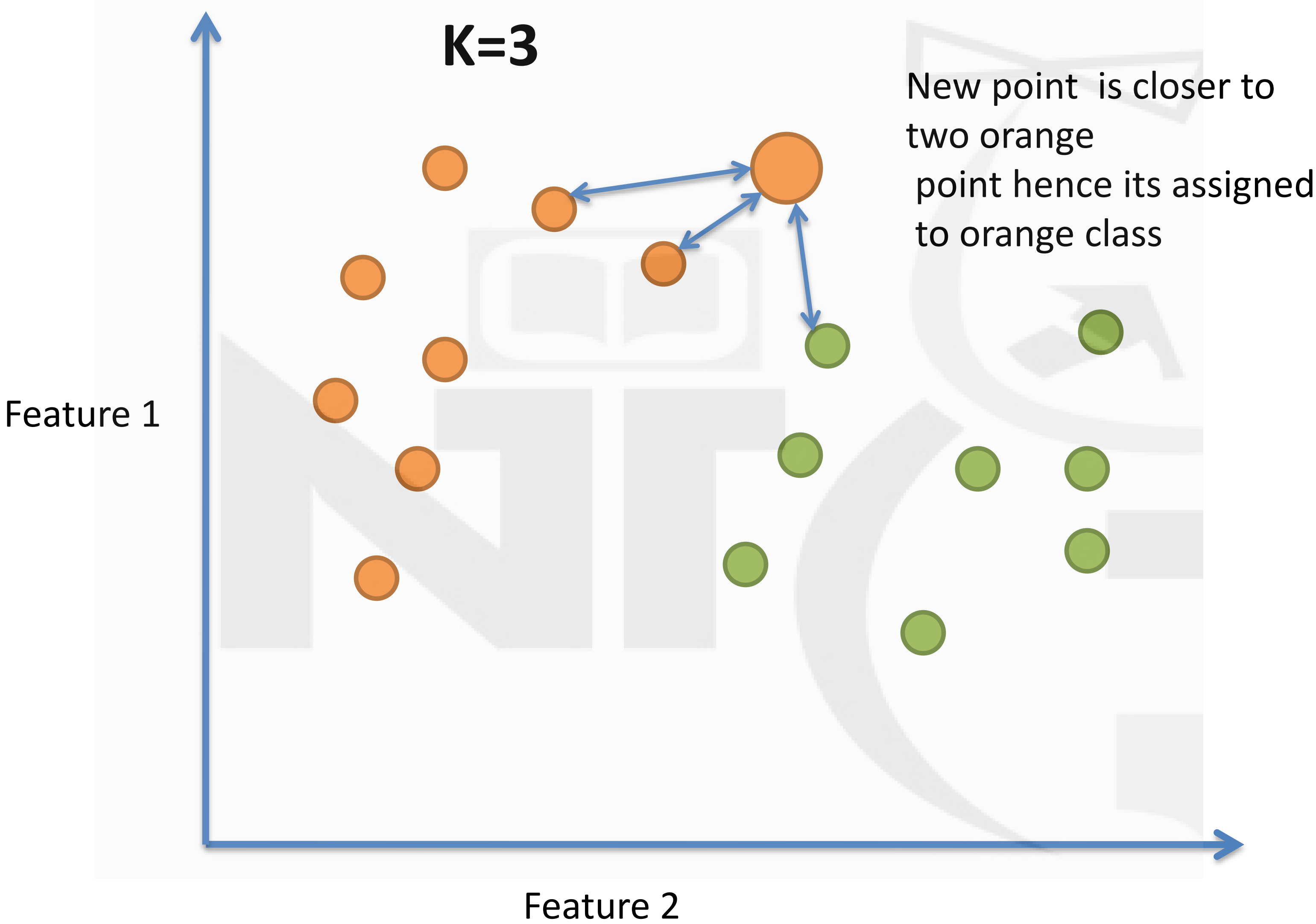
KNN finds the distance of each point in dataset from the new data point



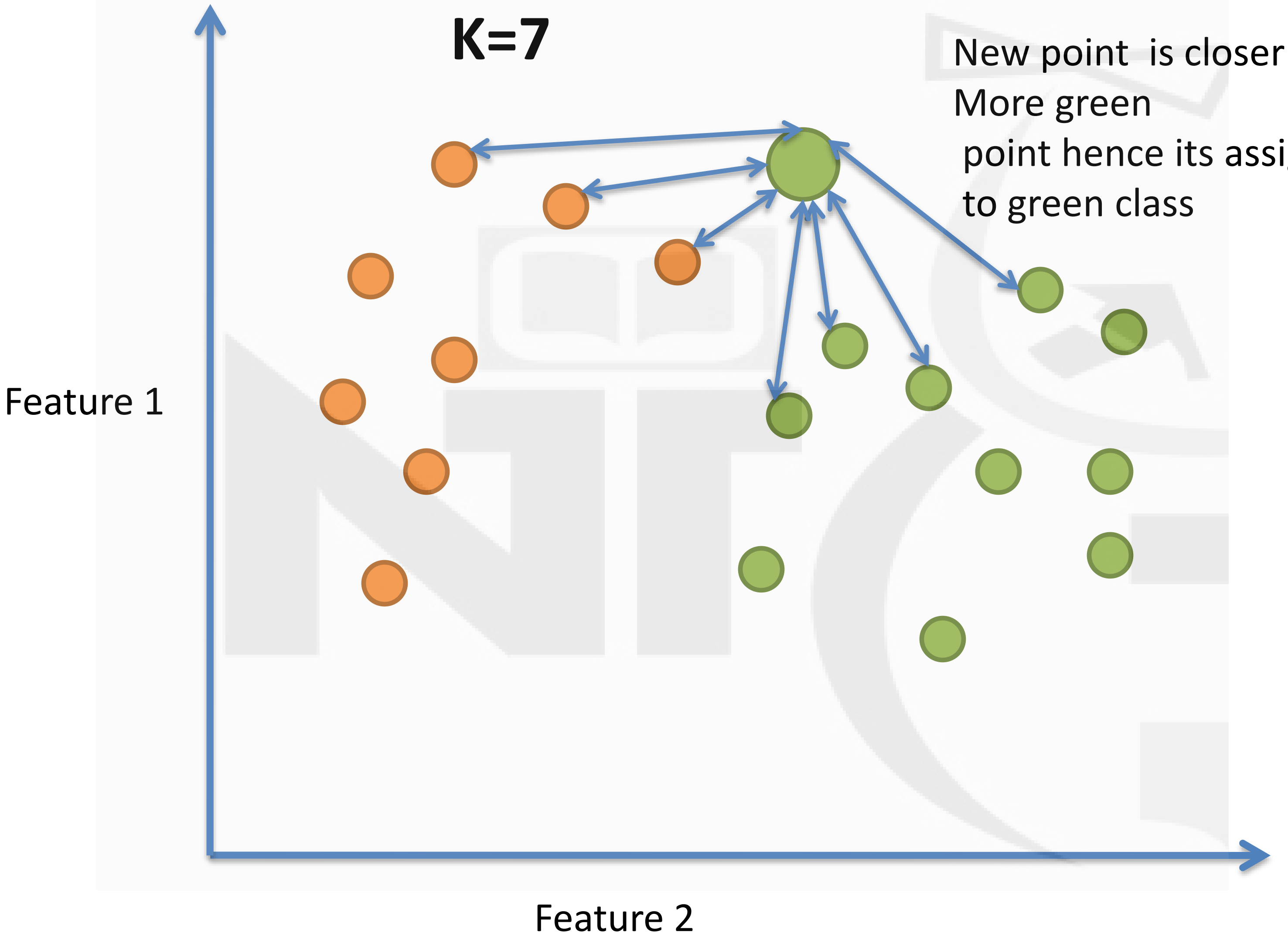
Based on the value of K it considers the nearest k points



KNN takes the majority votes and classifies the new point



KNN takes the majority votes and classifies the new point



KNN Algorithm Steps

- **Step 1:** Determine the number of neighbors, **k**. (e.g., 3, 5, 7).
- **Step 2:** Calculate the distance between the new data point and all other data points in the training set. Common distance metrics include:
  - Euclidean Distance: $d(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$
  - Manhattan Distance: $d(p, q) = \sum_{i=1}^{n} |p_i - q_i|$
  - Minkowski Distance: A generalization of Euclidean and Manhattan.

KNN Algorithm Steps

- **Step 3:** Identify the **k** closest neighbors to the new data point.
- **Step 4 (For Classification):** Assign the class label to the new data point based on the majority vote among the **k** nearest neighbors.
- **Step 4 (For Regression):** Assign the value to the new data point based on the average of the values of the **k** nearest neighbors.

Choosing the Value of K

- The choice of *k* is crucial:A small value of *k* (e.g., 1 or 2) can lead to overfitting, as the model may be too sensitive to noise in the training data.

- A larger value of *k* smooths out the decision boundary but may overlook local patterns.

Rule of Thumb: Start with $k = \sqrt{n}$, where n is the number of data points in the training set, and adjust based on performance.

Advantages of k-NN

- **Simple and intuitive:** Easy to understand and implement.
- **No training phase:** Can be very fast in scenarios with small datasets.
- **Versatile:** Can handle multi-class classification problems.

Disadvantages of k-NN

- **Computationally expensive:** Especially with large datasets, since the algorithm needs to calculate the distance to every training data point for each prediction.
- **Storage requirements:** It requires storing the entire dataset.
- **Sensitive to irrelevant or redundant features:** If features are not properly scaled or relevant, it may affect the distance calculations.
- Will not be effective when the class distributions overlap
- Fixing the optimal value of K is a challenge

Distance Metrices

1. Euclidean Distance
2. Manhattan Distance
3. Minkowski Distance
4. Cosine Similarity

More metrices:

- https://scikit-learn.org/0.24/modules/generated/sklearn.neighbors.DistanceMetric.html

Common Distance Metrics in KNN

- **Euclidean Distance**: The most commonly used metric, which is the straight-line distance between two points in Euclidean space.

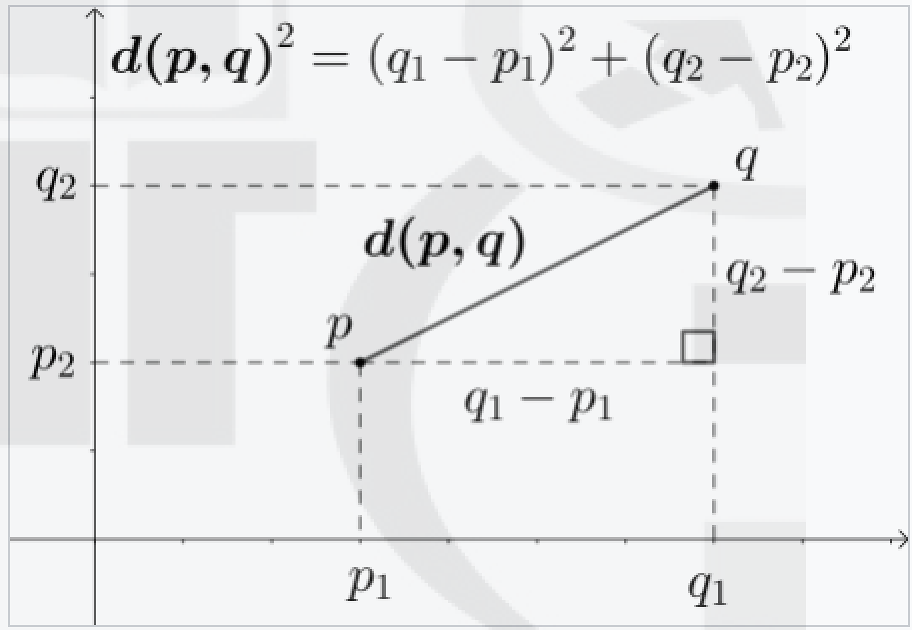
$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- .

Euclidean

https://en.wikipedia.org/wiki/Euclidean_distance

$$d(p,q) = \sqrt{(p-q)^2}.$$



Using the Pythagorean theorem to compute two-dimensional Euclidean distance

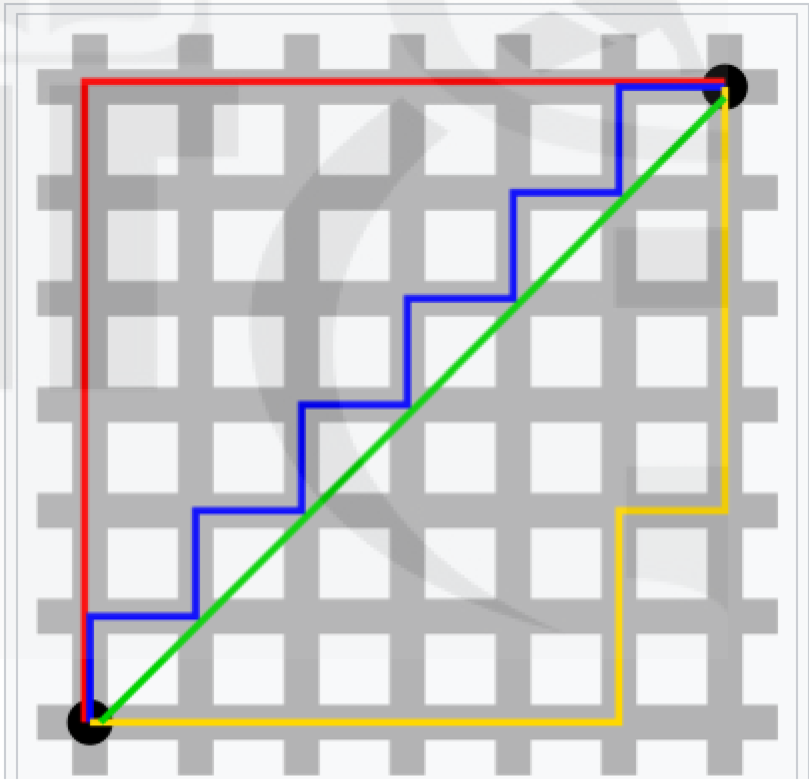
Common Distance Metrics in KNN

- **Manhattan Distance**: Also known as the L1 distance, it is the sum of the absolute differences of the coordinates.

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Manhattan Distance

- https://en.wikipedia.org/wiki/Taxicab_geometry



- **Minkowski Distance**: A generalized form of both Euclidean and Manhattan distances. It is defined as:

- https://en.wikipedia.org/wiki/Minkowski_distance

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

- When $p = 2$, it is equivalent to Euclidean distance.
- When $p = 1$, it is equivalent to Manhattan distance.

- **Cosine Similarity**: Although not a distance metric, it's often used to measure the cosine of the angle between two vectors, which can be used in KNN.

$$\text{cosine similarity} = \frac{x \cdot y}{\|x\|\|y\|}$$

metric	Function
'cityblock'	metrics.pairwise.manhattan_distances
'cosine'	metrics.pairwise.cosine_distances
'euclidean'	metrics.pairwise.euclidean_distances
'haversine'	metrics.pairwise.haversine_distances
'l1'	metrics.pairwise.manhattan_distances
'l2'	metrics.pairwise.euclidean_distances
'manhattan'	metrics.pairwise.manhattan_distances
'nan_euclidean'	metrics.pairwise.nan_euclidean_distances

- https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html