## **K-Nearest Neighbors (KNN)**

**1. Introduction:**

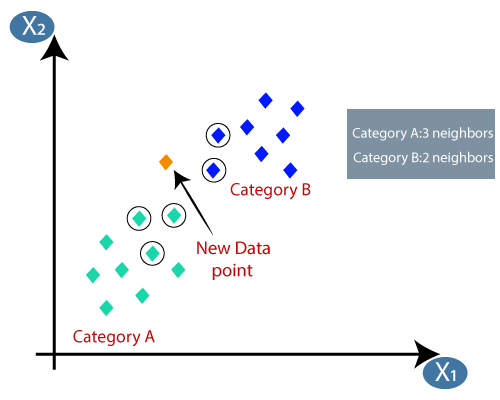
* **What it is:** KNN is a simple, versatile, and intuitive supervised machine learning algorithm used for both **classification** and **regression** tasks.
* **Algorithm Type:**
  + **Instance-Based Learning:** It doesn't learn an explicit function from the training data but relies on storing the entire dataset. Predictions are made based on similarity to stored instances.
  + **Lazy Learning:** It does minimal computation during the "training" phase (essentially just storing the data). Most of the work happens during the prediction phase.
  + **Non-parametric:** It makes no strong assumptions about the underlying data distribution.

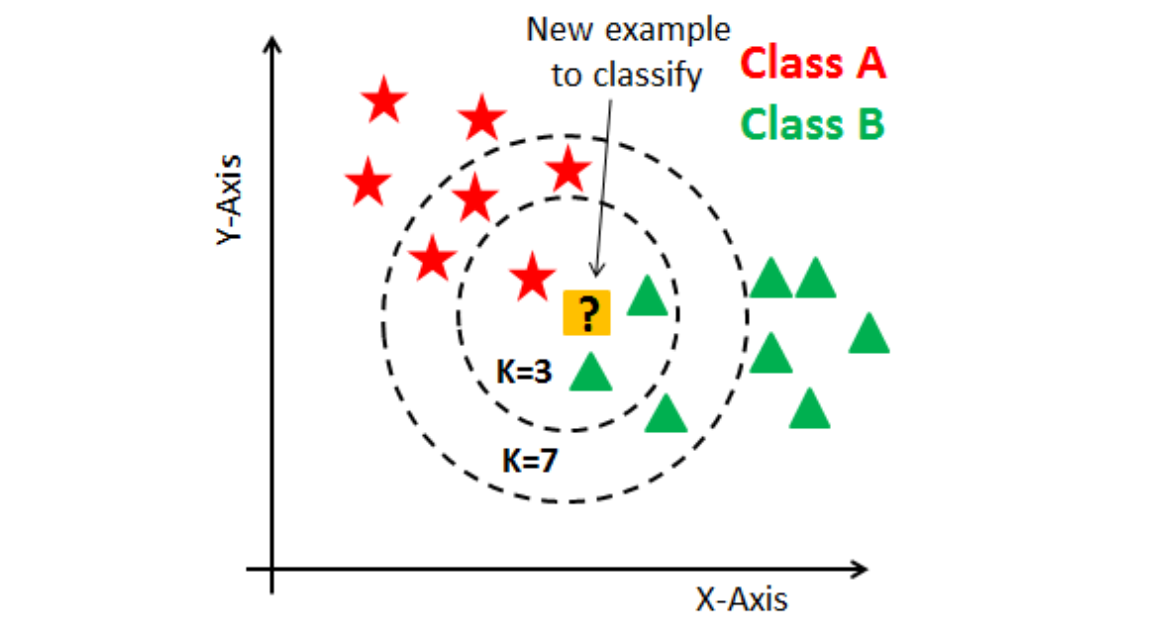
**2. Core Idea:**

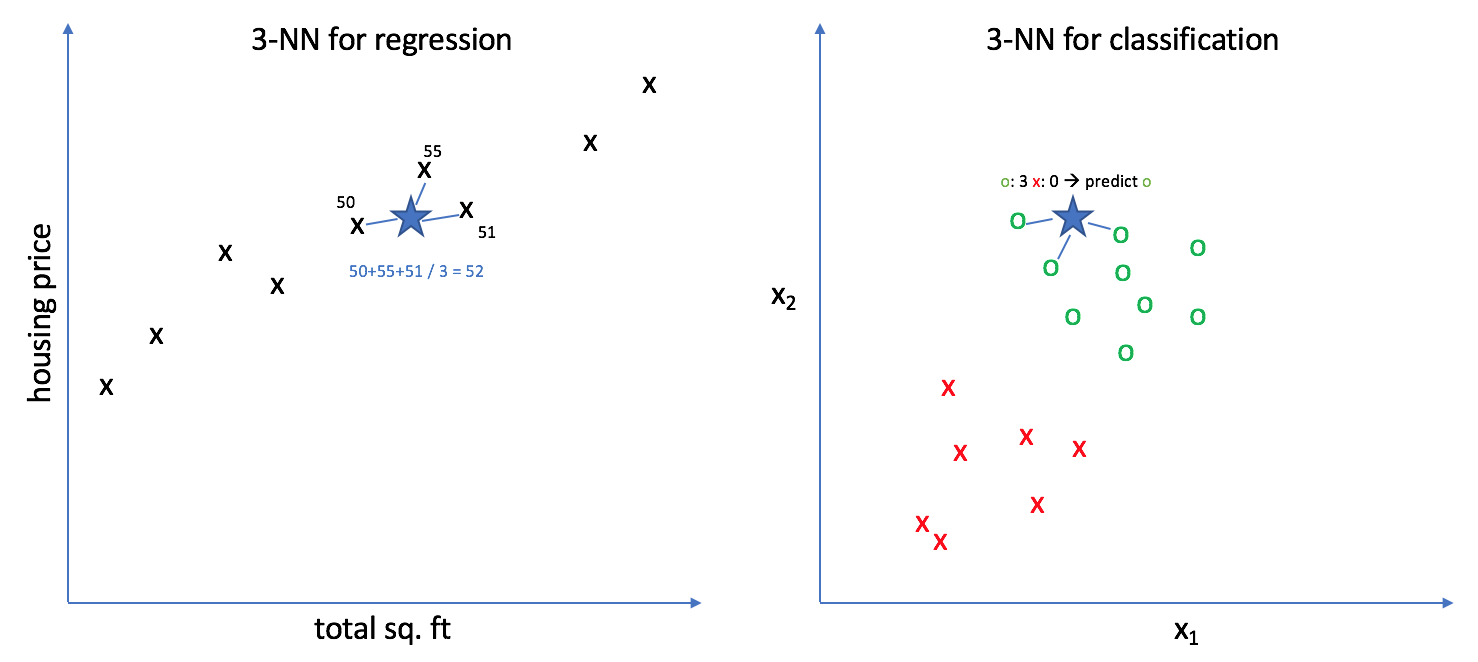
* The fundamental principle is **"similarity"** or **"proximity"**.
* To classify or predict a value for a new, unseen data point, KNN looks at the K closest data points (neighbors) to it in the training dataset.
* The prediction is based on the properties (class label or value) of these K neighbors. "Birds of a feather flock together."

**3. How KNN Works (Steps):**

1. **Choose the value of K:** Decide how many neighbors (e.g., 3, 5, 10) to consider. This is a hyperparameter.
2. **Choose a Distance Metric:** Select a method to measure the "closeness" or distance between data points (e.g., Euclidean, Manhattan).
3. **For a new data point (x\_new):**
   * Calculate the distance between x\_new and **every** data point in the training dataset using the chosen metric.
   * Identify the K training data points that have the smallest distances to x\_new. These are the "K nearest neighbors".
   * **For Classification:** Perform a **majority vote** among the class labels of the K neighbors. The most frequent class among the neighbors is assigned as the predicted class for x\_new. (Using an odd K helps avoid ties).
   * **For Regression:** Calculate the **average** (or sometimes the median) of the target values of the K neighbors. This average/median value is assigned as the predicted value for x\_new.





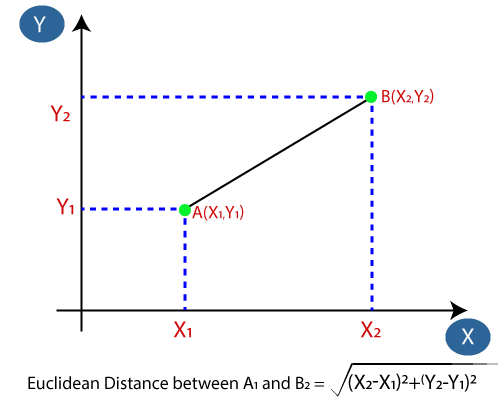
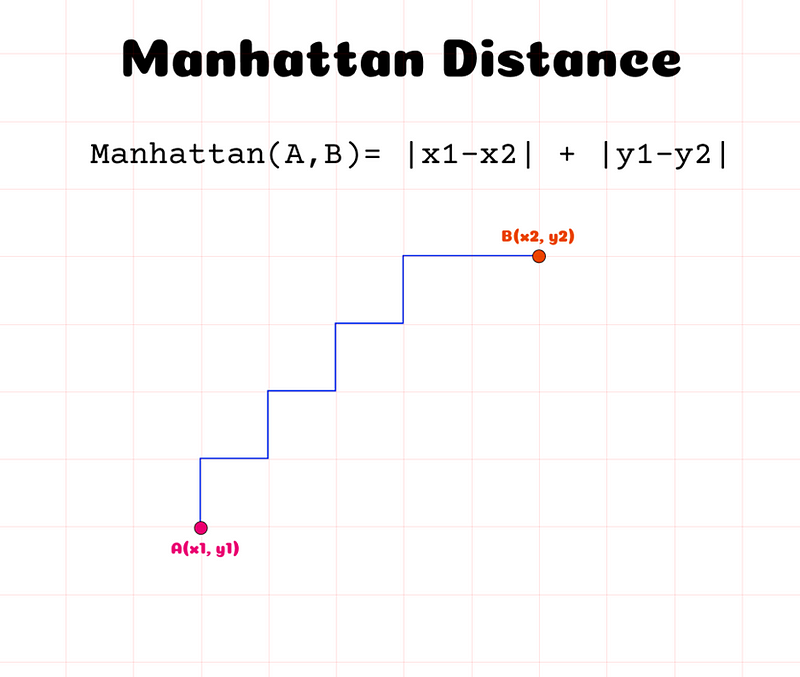
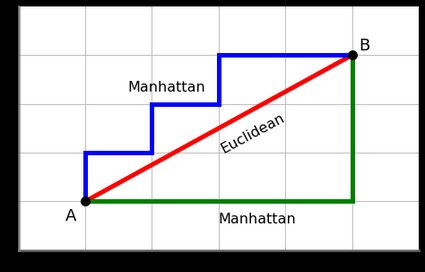


**4. The Parameter K:**

* K determines the number of neighbors influencing the prediction.
* **Choosing K:**
  + **Small K (e.g., K=1):** Model is very sensitive to noise and outliers. Decision boundary can be complex. Low bias, high variance (prone to overfitting).
  + **Large K:** Model is smoother, less sensitive to noise. Decision boundary is simpler. High bias, low variance (prone to underfitting).
  + **Selection Method:** Often chosen using cross-validation on the training set to find a K that balances bias and variance well. An odd number is usually preferred for classification to prevent ties.



**5. Distance Metrics:**

* Crucial for defining "closeness". Choice depends on the data type and problem context.
* Common Metrics:
  + **Euclidean Distance (L2​ norm):** Most common; the straight-line distance between two points in Euclidean space. Formula: d(x,y)=∑i=1n​(xi​−yi​)2​
  + 
  + **Manhattan Distance (L1​ norm):** Sum of the absolute differences of their coordinates ("city block" distance). Often better for high-dimensional data. Formula: d(x,y)=∑i=1n​∣xi​−yi​∣
  + Imagine a city grid where you can only move horizontally or vertically. The Manhattan distance is the shortest path between two points in such a grid.
  + 
  + 
  + **Minkowski Distance:** A generalization of both Euclidean (p=2) and Manhattan (p=1). Formula: d(x,y)=(∑i=1n​∣xi​−yi​∣p)1/p
  + **Hamming Distance:** Used for categorical variables (measures the number of positions at which corresponding symbols are different).

**6. Importance of Feature Scaling:**

* **Crucial for KNN!** Since KNN relies on distances, features with larger ranges/values can disproportionately influence the distance calculation compared to features with smaller ranges.
* **Solution:** Scale features before applying KNN. Common methods include:
  + **Normalization (Min-Max Scaling):** Scales data to a fixed range, usually [0, 1].
  + **Standardization (Z-score Normalization):** Scales data to have zero mean and unit variance.

**7. Advantages:**

* **Simple and Intuitive:** Easy to understand and implement.
* **No Training Phase:** Just stores data (lazy learner). Can be updated easily with new data.
* **Non-parametric:** Makes no assumptions about data distribution.
* **Handles Multi-class Problems:** Works naturally with more than two classes.
* **Flexible Decision Boundaries:** Can learn complex boundaries.

**8. Disadvantages:**

* **Computationally Expensive Prediction:** Must compute distances to *all* training points for each prediction. Slow for large datasets.
* **High Memory Requirement:** Needs to store the entire training dataset.
* **Sensitive to Irrelevant Features:** Irrelevant features can distort distance calculations ("Curse of Dimensionality"). Feature selection/engineering is important.
* **Sensitive to Feature Scaling:** Performance heavily depends on proper scaling.
* **Performance Dependent on K and Distance Metric:** Requires careful tuning of these hyperparameters.

**9. Common Applications:**

* Recommendation Systems (finding similar items/users)
* Image Recognition / Computer Vision (finding similar images)
* Anomaly Detection (identifying points far from others)
* Gene Expression Analysis
* Basic classification and regression tasks, especially as a baseline model.