DS103-Supervised Classification Models - K-Nearest Neighbors (KNN)

أكاديمية طويق
Tuwaiq Academy

By: eng. Esraa Madhi

What is KNN Algorithm?

https://youtu.be/0p0o5cmgLdE

https://youtu.be/0p0o5cmgLdE

The k-nearest neighbor (k-NN) algorithm is a type of **non-parametric** method used for classification and regression.

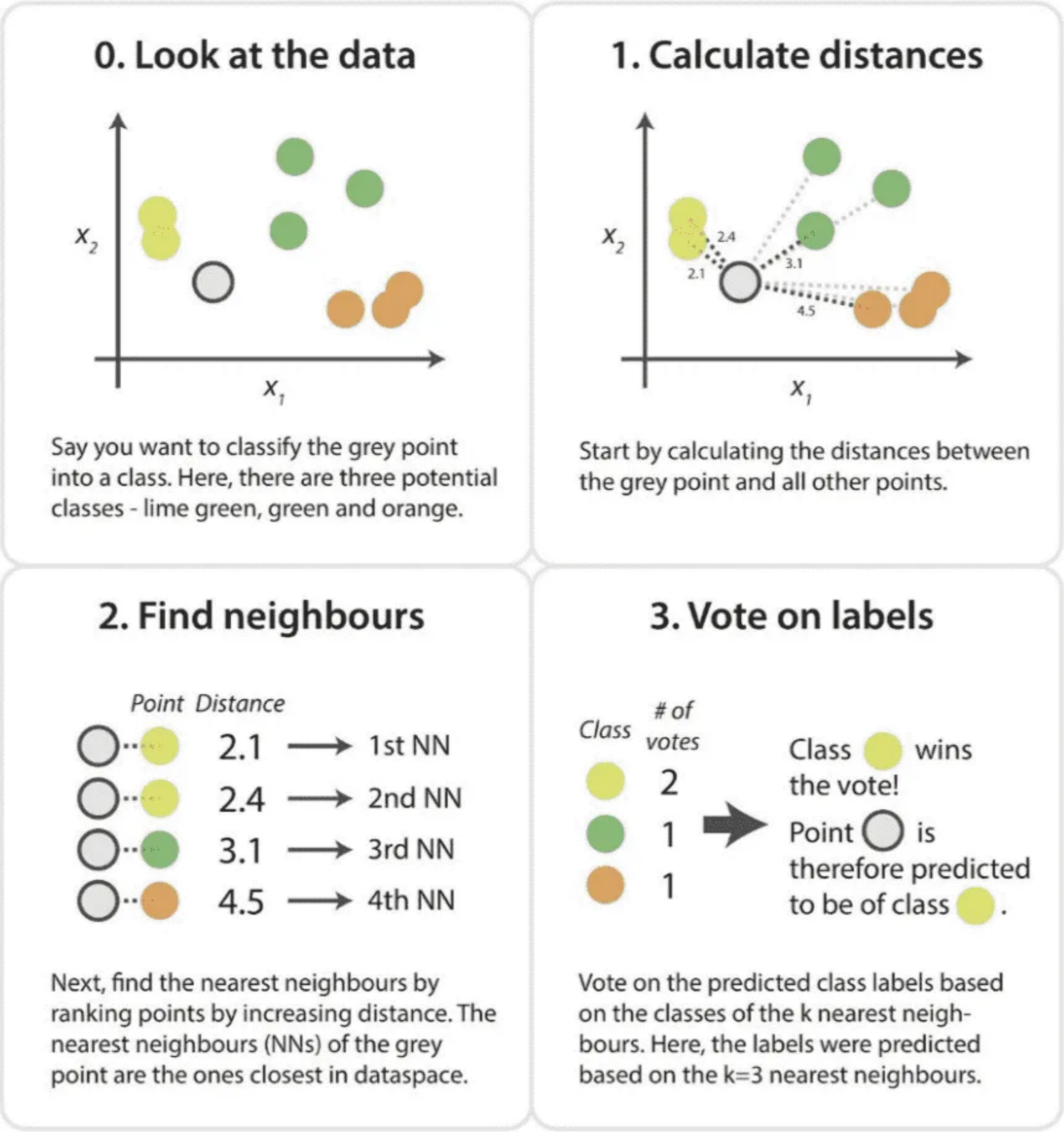
How KNN Works?

The k-NN algorithm is considered a **“lazy learning”** algorithm because it doesn’t build a model until a new instance is given for classification or regression. This also means that the algorithm **doesn’t require any training data in advance.**

In Short: The basic idea behind the algorithm is to **find the k number of closest instances in the training dataset for a new instance and have them “vote”** on the class or value of the new instance. The class or value with the most votes is then assigned to the new instance.

In Steps:

1. Initialize the value of k
2. For getting the predicted class, iterate from 1 to total number of training data points
    - Calculate the distance between test data and each row of training dataset.
    - Sort the calculated distances in ascending order based on distance values
    - Get top k rows from the sorted array
    - Get the most frequent class of these rows
    - Return the predicted class



0. Look at the data

Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

1. Calculate distances

Start by calculating the distances between the grey point and all other points.

2. Find neighbours

Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

3. Vote on labels

Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

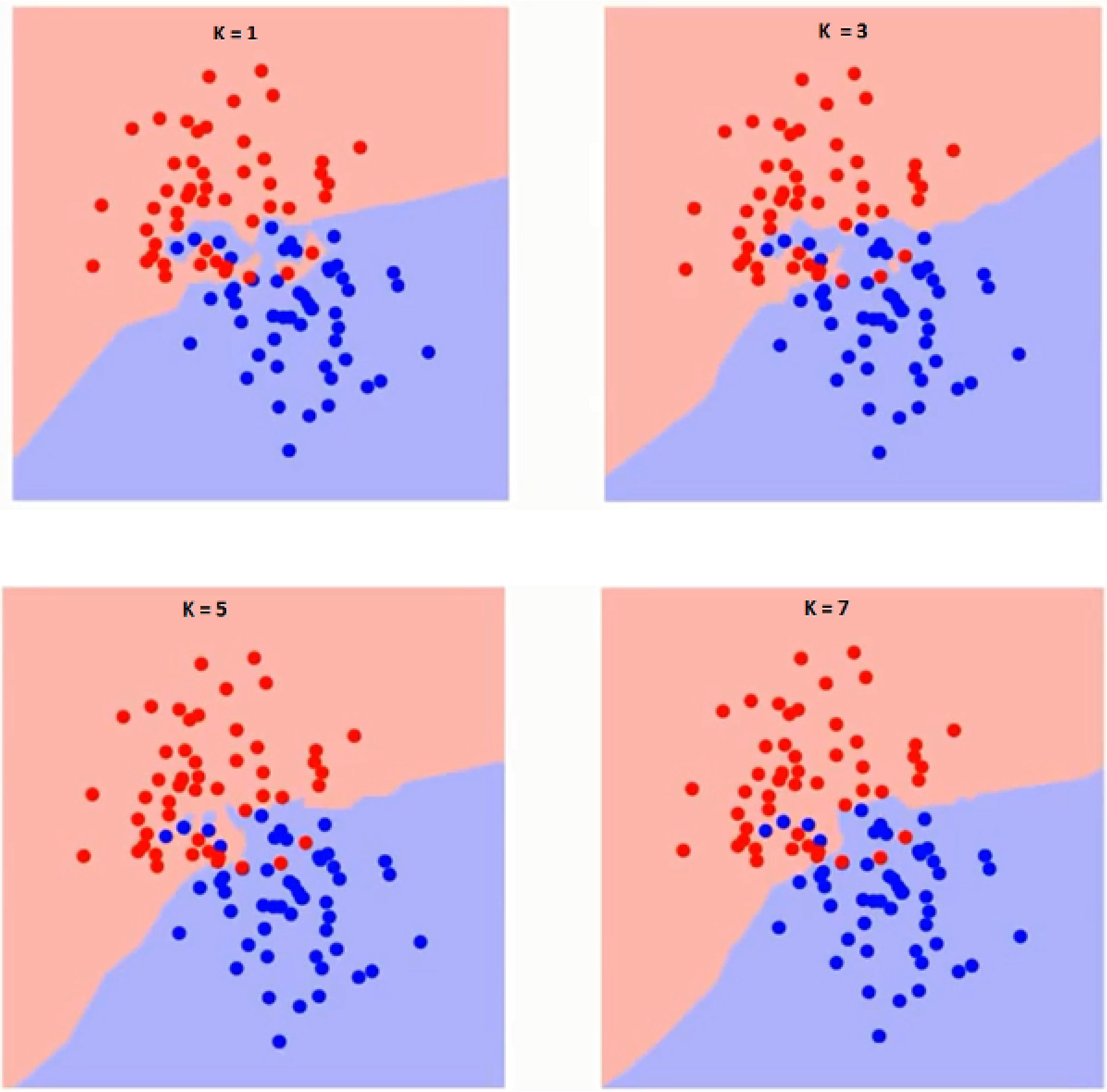
Demo:

- https://adotg.github.io/knn-what-how-why/
- https://www.philippe-fournier-viger.com/tools/KNN.php
- https://codepen.io/gangtao/pen/PPoqMW

How to fine best K?

- In k-NN classification, the value of k is **typically small,** such as k=1 or k=3. A k-value of 1 means that the new instance is assigned to the class of its nearest neighbor. A larger k-value, such as k=3, means that the new instance is assigned to the class that is most common among its 3 nearest neighbors.

- In k-NN regression, the value of k is typically set to an odd number to avoid tiebreakers.
- If you watch bellow carefully, you can see that the boundary becomes smoother with increasing value of K. With K increasing to infinity it finally becomes all blue or all red depending on the total majority.



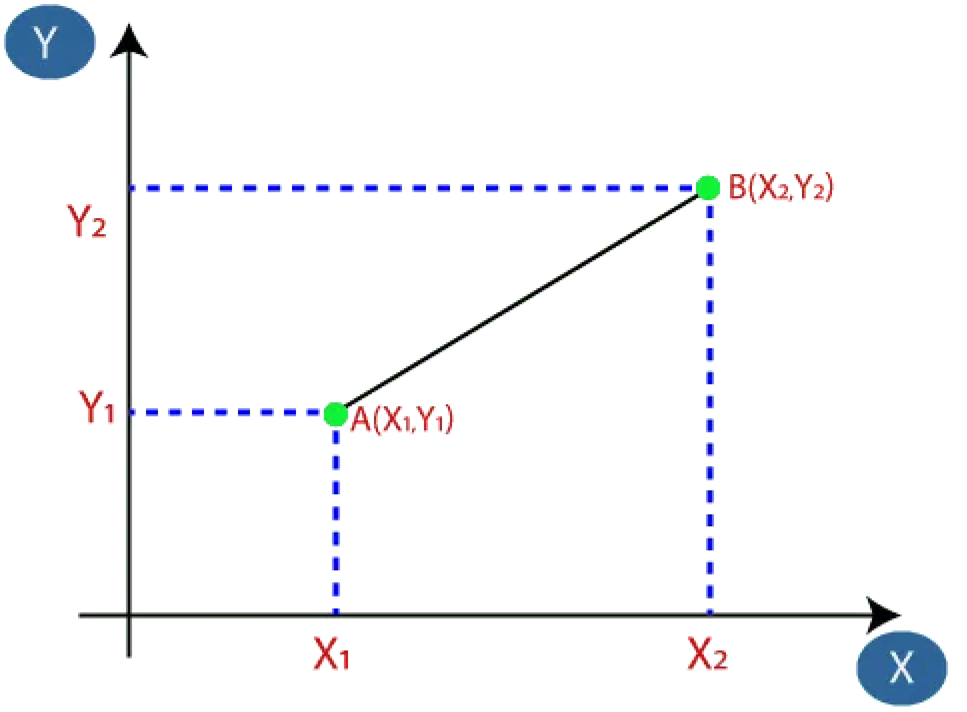
- One can use cross-validation to select the optimal value of k for the k-NN algorithm, which helps improve its performance and prevent overfitting or underfitting.

How to calculate distance?

k-NN algorithm uses the **distance metric** to measure the similarity between a new instance and the instances in the training dataset

Distance metric type:

1. **Euclidean Distance:** Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (y).



Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

2. **Manhattan Distance:** This is the distance between real vectors using the sum of their absolute difference.

Distance functions

Euclidean

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^{k}\left|x_i - y_i\right|$$

3. **Hamming Distance:** It is used for categorical variables. If the value (x) and the value (y) are the same, the distance D will be equal to 0 . Otherwise D=1.

$$D_H = \sum_{i=1}^{k}\left|x_i - y_i\right|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

Preprocessing steps for features:

1. Create dummy variables out of a categorical variable and include them instead of original categorical variable
2. Standardize variables before calculating distance.

Example Calculation:

Assume:

- We have a data set with 4 features:
  - First point:
    - P = (1, 2, "red", "small")
  - Second point:
    - Q = (5, 6, "green", "big"))
- Distance for continuous features:
  - $d_c = \sqrt{(1-5)^2 + (2-6)^2} = \sqrt{16+16} = \sqrt{32}$
- Distance for categorical features:

    - $d_k = 2$ (since both features differ)
- Combined Distance:
    - $d = \sqrt{32 + 4} = \sqrt{36}$

Pros

1. Easy to understand
2. No assumptions about data
3. Can be applied to both classification and regression
4. Works easily on multi-class problems

Cons

1. Memory Intensive / Computationally expensive
2. Sensitive to scale of data
3. Low k-value is sensitive to outliers and a higher K-value is more resilient to outliers as it considers more voters to decide prediction.
4. Struggle when high number of independent variables
5. Choosing the right value of **K** can be tricky.

Resources:

- https://www.freecodecamp.org/news/k-nearest-neighbors-algorithm-classifiers-and-model-example/