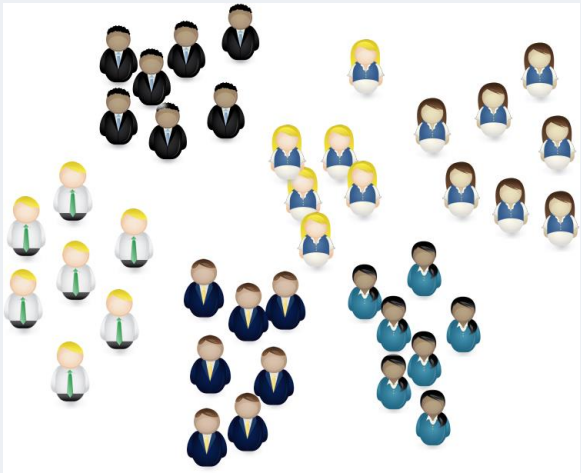
K-Nearest-Neighbor

Mahdi Akbari Zarkesh
9612762638

INSTRUCTIONS

"Show me who your friends are and I'll tell you who you are?"



KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.
The k-Nearest-Neighbor Classifier (KNN) works directly on the learned samples, instead of creating rules compared to other classification methods.

The KNN Algorithm

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data
3.1 Calculate the distance between the query example and the current example from the data.
3.2 Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

Advantages

1. The algorithm is simple and easy to implement.
2. There's no need to build a model, tune several parameters, or make additional assumptions.
3. The algorithm is versatile. It can be used for classification, regression, and search (as we will see in the next section).

Disadvantages

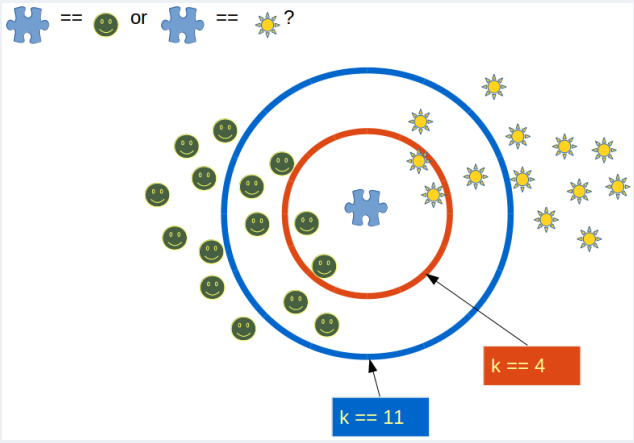
1. The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

Choosing the right value for K

To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.

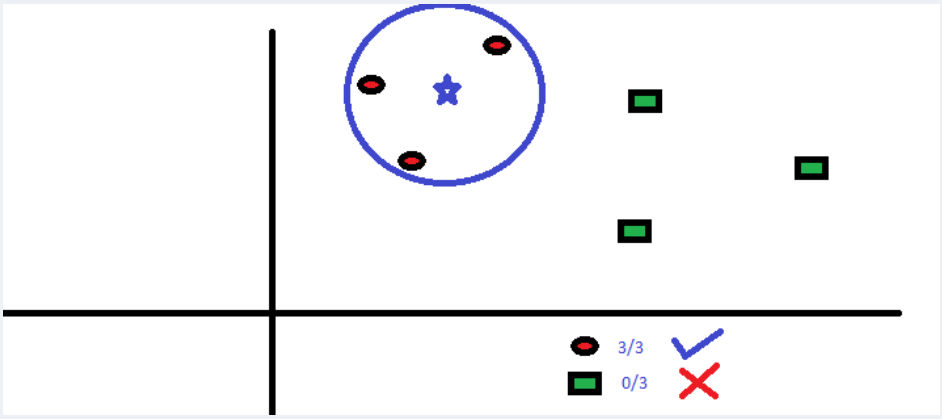
Here are some things to helpful point:

1. As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, imagine K=1 and we have a query point surrounded by several reds and one green (I'm thinking about the top left corner of the colored plot above), but the green is the single nearest neighbor. Reasonably, we would think the query point is most likely red, but because K=1, KNN incorrectly predicts that the query point is green.
2. Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.
3. In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.



Voting to get a Single Result

Voting for the most frequent label to find single Result.



The Weighted Nearest Neighbor Classifier

To pursue this strategy, we can assign weights to the neighbors in the following way: The nearest neighbor of an instance gets a weight 1/1, the second closest gets a weight of 1/2 and then going on up to 1/k for the farthest away neighbor.

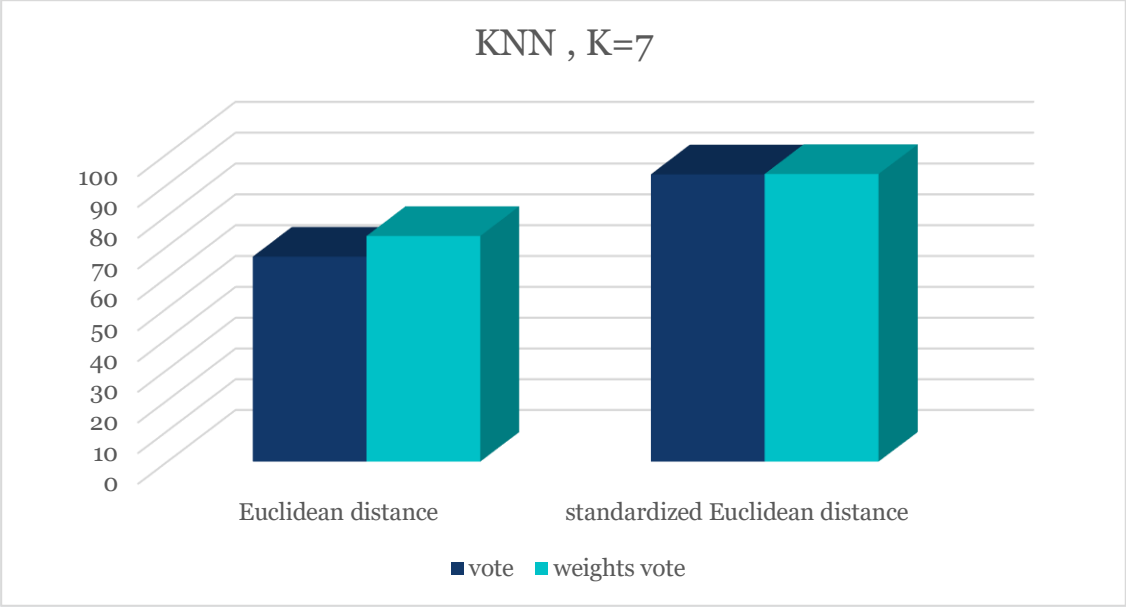
This means that we are using the harmonic series as weights:

$$\sum 1/(i+1)=1+1/2+1/3+...+1/k$$

The DataSet : wine

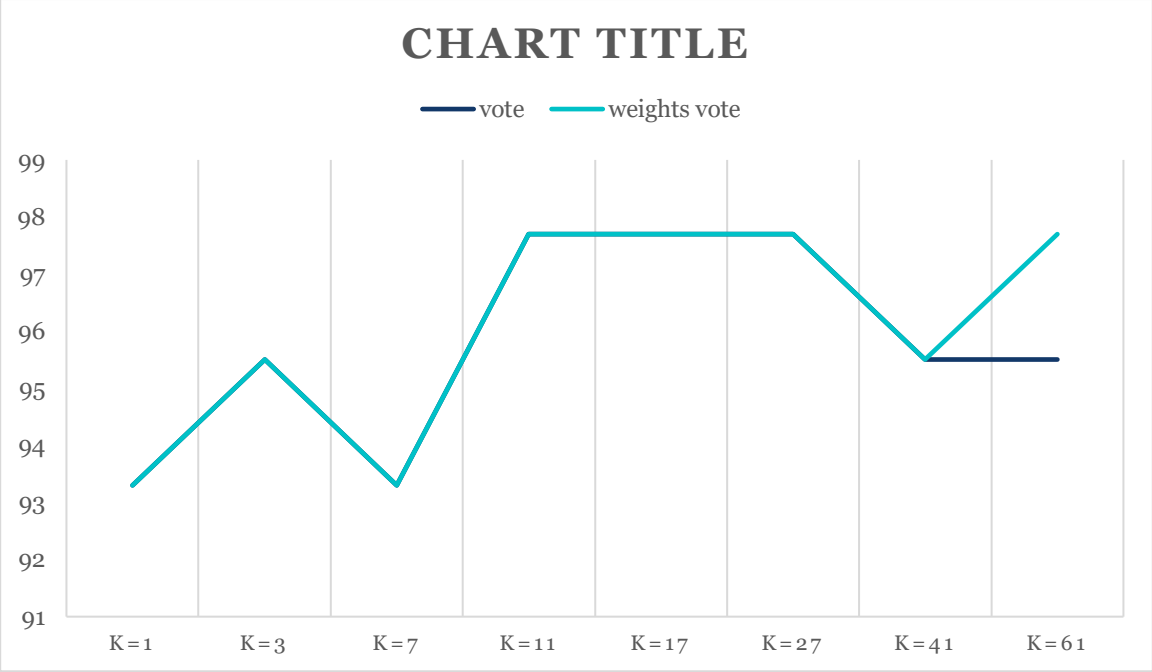
Classes : 3
Samples per class : [59,71,48]
Samples total : 178
Dimensionality : 13
Features : real, positive

Change distance Function Effect



From the output we can see that the SED is better than ED because in SED the data make more normalize.

Change K In K_Neighbors Effect



From the output we can see that the mean error is zero when the value of the K is between 11 and 27. I would advise you to play around with the value of K to see how it impacts the accuracy of the predictions.

Thank you :)