Lesson 18

Classification Techniques – K-Nearest Neighbors

Kush Kulshrestha

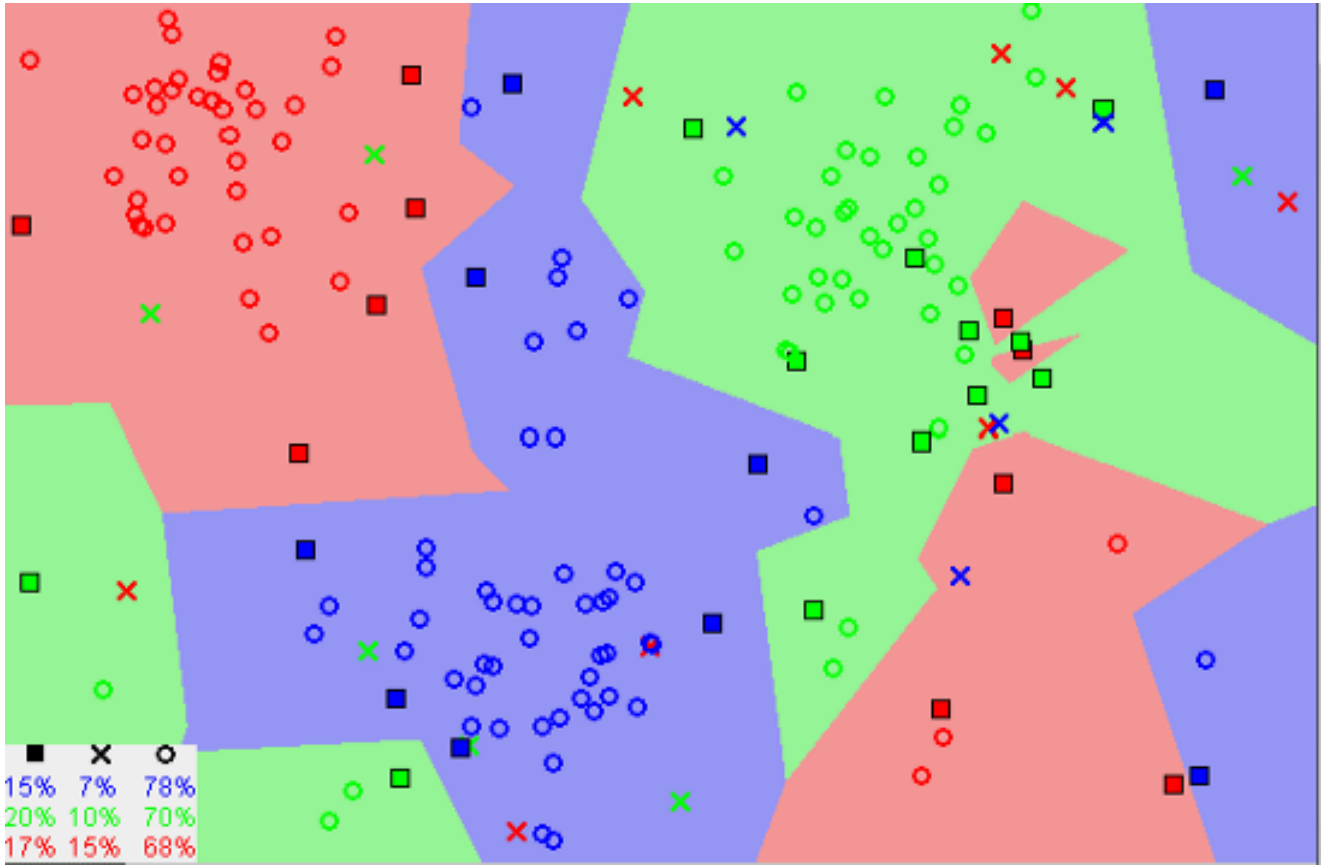
kNN Algorithm Summed up

kNN Algorithm

The KNN algorithm assumes that similar things exist in close proximity.

In other words, similar things are near to each other.

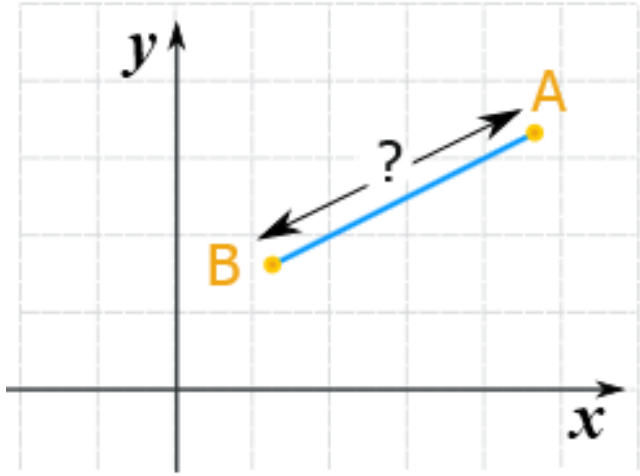
Notice in the image above that most of the time, similar data points are close to each other. The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful.



Which data points are near to me?

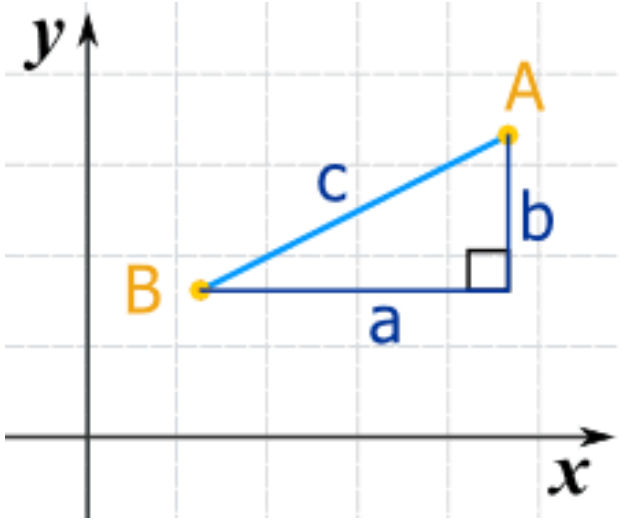
KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) by calculating the distance between points on a graph.

How could you find the distance between two points?



One way is to use the Pythagoras theorem and draw some perpendicular lines.

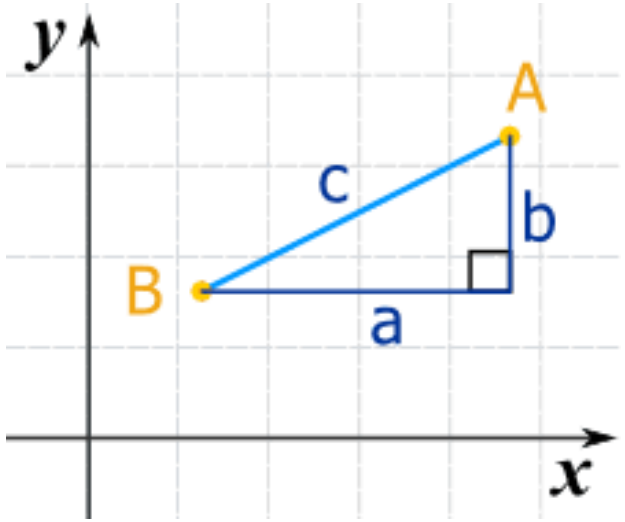
$$a^2 + b^2 = c^2$$



Which data points are near to me?

By solving below equation, we can get the value of c, which could be used as effective distance between A and B.

$$a^2 + b^2 = c^2$$



This distance is called Euclidean distance. There are other distances that could be used to calculate the number representing the distance between two points.

Check here:

https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d

For now, we will be going forward with Euclidean distance.

The Algorithm

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data:
    1. Calculate the distance between the training data and the current example from the data.
    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

Choosing the right 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.

Things to care about:

a. As we decrease the value of K to 1, our predictions become less stable

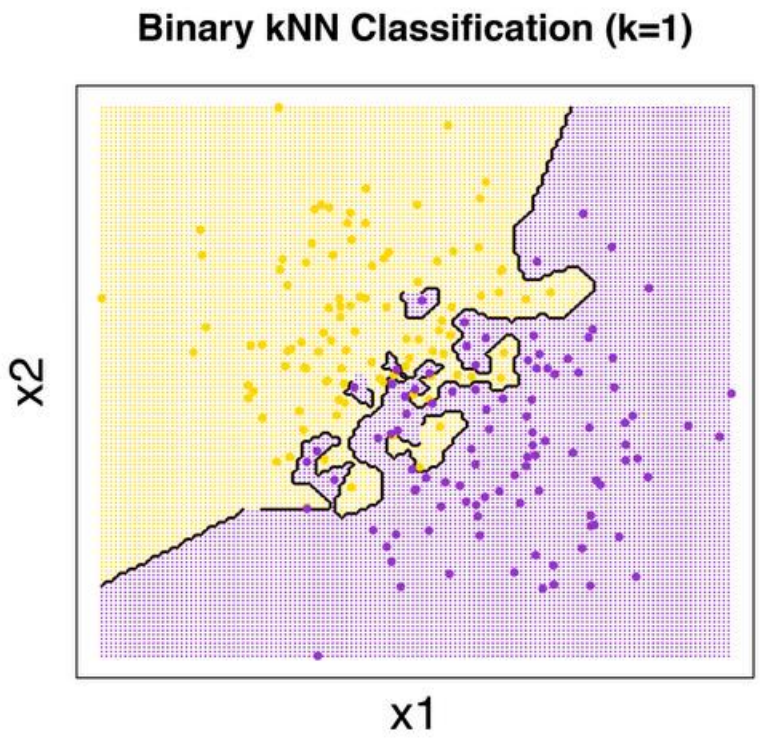
b. Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.

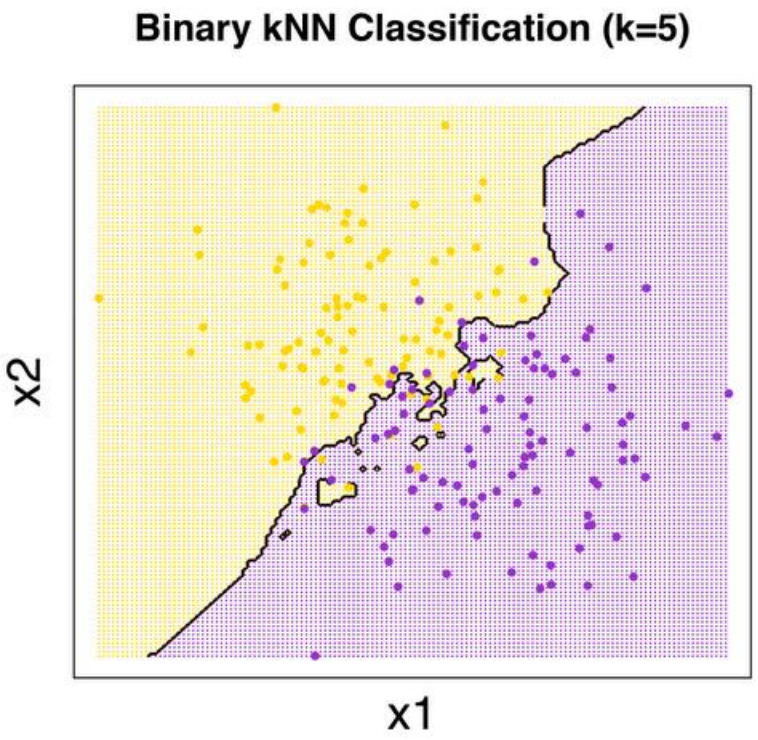
c. In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

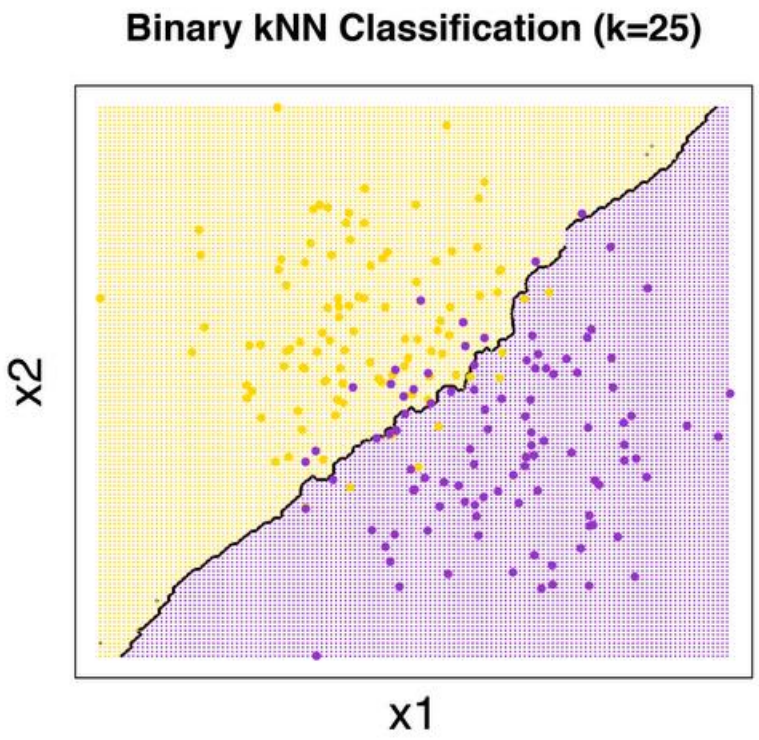
Choosing the right k

Under fitting – Over fitting.

Sounds familiar?





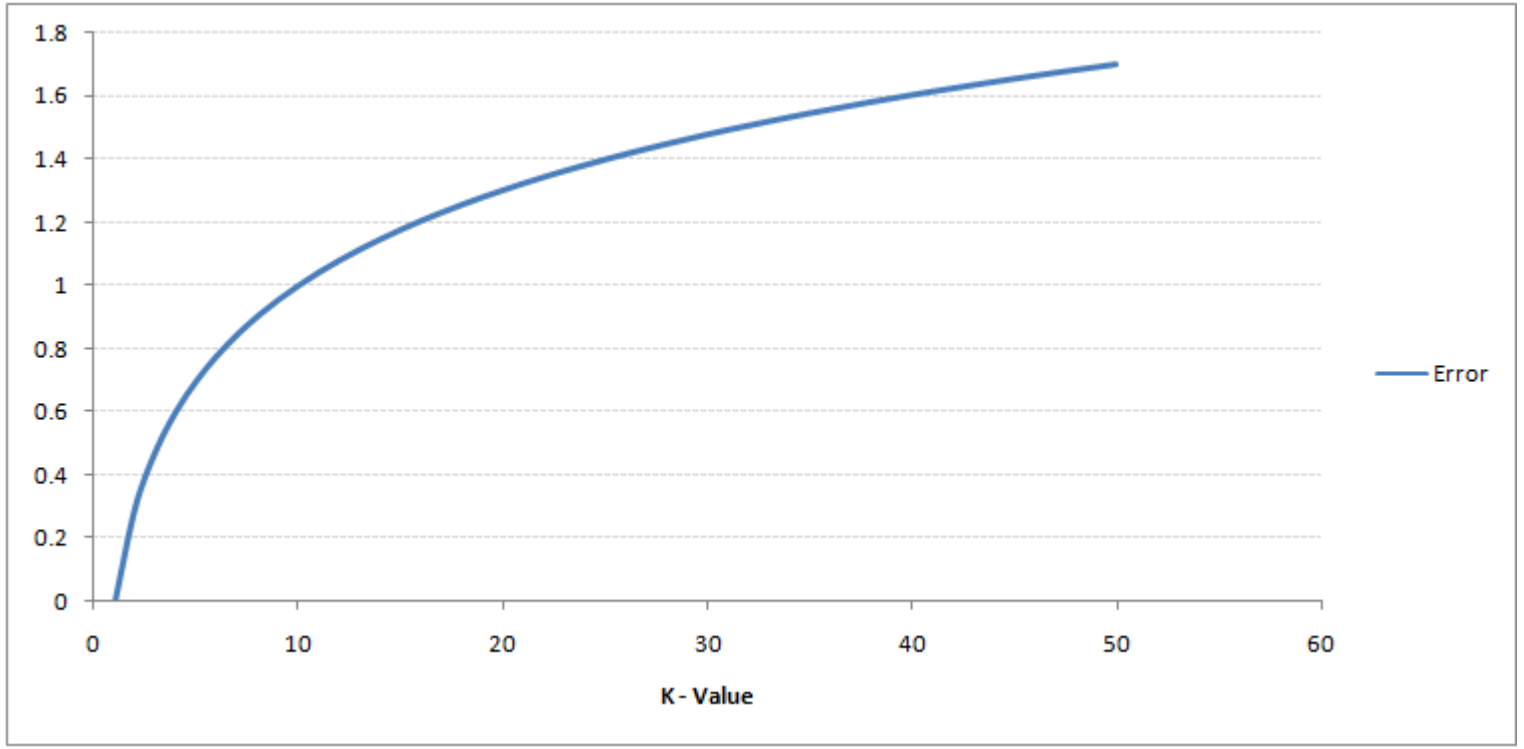


Choosing the right k

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.

Trend in Training error:

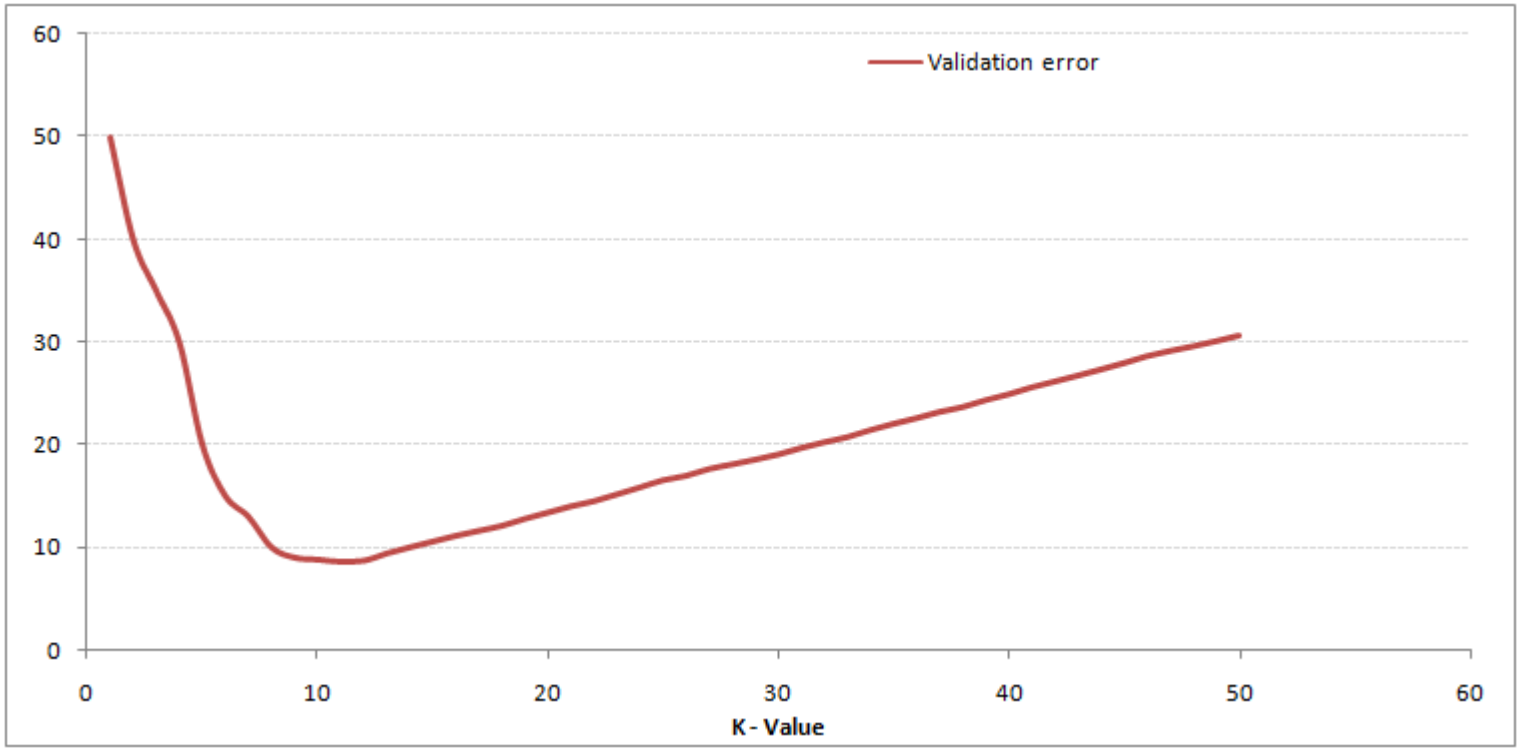
With k=1, training error will always be 0 because, because we are just looking at ourselves to classify us. As k increases, the training error will also increase, until k is so big that all of the examples are classified as one class.



Choosing the right k

Trend in Testing error:

At K=1, we were overfitting the boundaries. Hence, error rate initially decreases and reaches a minima. After the minima point, it then increase with increasing K. To get the optimal value of K, you can segregate the training and validation from the initial dataset. Now plot the validation error curve to get the optimal value of K. This value of K should be used for all predictions.



Pros and Cons

Advantages:

The algorithm is simple and easy to implement.

There’s no need to build a model, tune several parameters, or make additional assumptions.

Disadvantages:

The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase. This is called the curse of Dimensionality.

Reference: https://en.wikipedia.org/wiki/Curse_of_dimensionality