**[KNN\_Plant\_Classification](https://github.com/Estrada-John/KNN_Plant_Classification)**

**Introduction**

The idea is simple: the algorithm classifies each new data in the corresponding group, according to whether it has k neighbors closest to one group or another. That is, it calculates the distance of the new element from each of the existing ones and sorts these distances from least to greatest to select the group to which it belongs. This group will therefore be the most frequent with the shortest distances.

**Theory**

In contrast to other supervised learning algorithms, K-NN does not generate a model that is the result of learning with training data, but learning occurs while the test data is tested. These types of algorithms are called lazy learning methods.

It is a method that simply searches the closest observations to the one that is trying to predict and classifies the point of interest based on most of the data that surrounds it. As we said before, it is an algorithm:

**Supervised**: this briefly means that we have tagged our training data set, with the class or expected result given "one row" of data.

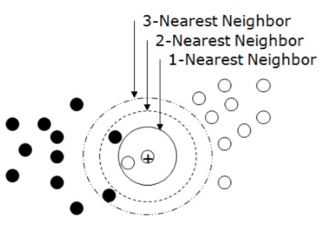
**Instance Based**: This means that our algorithm does not explicitly learn a model (such as in Linear Regression). Instead, it memorizes the training instances that are used as the "knowledge base" for the prediction phase.

**How does it work**?

1. Calculate the distance between the item to classify and the other items in the training dataset.
2. Select the closest "k" elements (with less distance, depending on the function used)
3. Perform a "majority vote" among the k points: those of a class / label that domain will decide their fine classification

**The variable k**, so that with different values of k we can also obtain very different results. This value is usually set after a multi-instance testing process.

**The similarity metric** used, since it will strongly influence the close relationships that will be established in the algorithm construction process. The distance metric can contain weights that will help us calibrate the classification algorithm, turning it, in fact, into a custom metric.



For k = 1 the algorithm will classify the ball with a + sign as white

For k = 2 the algorithm has no criteria to classify the ball with sign +

For k> = 3 the algorithm will classify the signed ball as black

**Data Set Description**

Attribute Characteristics: Real

Associated Tasks: Classification

Number of Instances: 150

Number of Attributes: 4

Split data percentage: 70% Train, 30% Test

The size of train data: 105

The size of test data:70

**Result**

I created 5 functions that will work with any data set. I also included description for each one of the functions and their respective input and output. I am confident with my result. Overall, my code includes features to explore the data through the learning process. The entire data is manipulated using numpy libraries and for splitting the data I used sklearn library.   
The result of my KNN algorithm has an accuracy of 97.77% accuracy

**Discussion**

The implementation of the KNN algorithm was straight forward and simple to follow. I will post this code in my github repository: <https://github.com/Estrada-John/KNN_Plant_Classification>