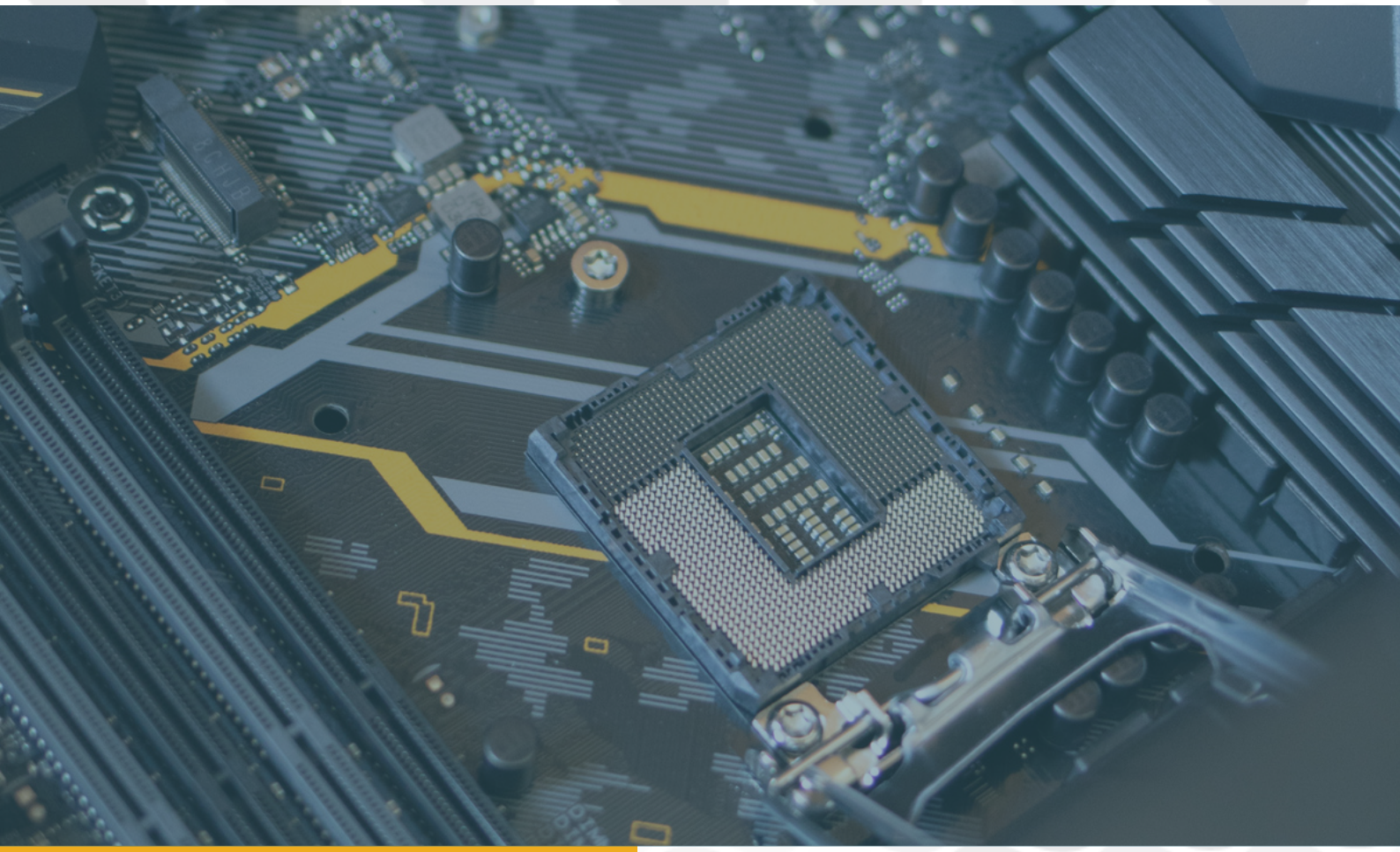
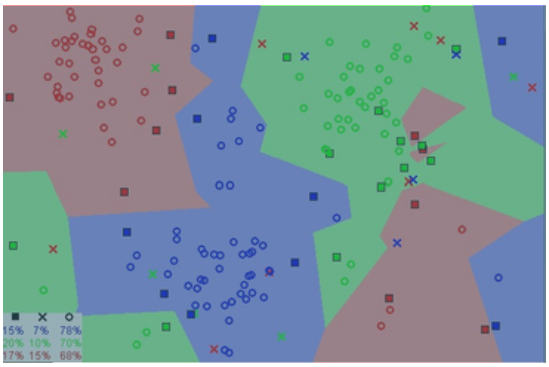


07

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN)

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. The algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph.

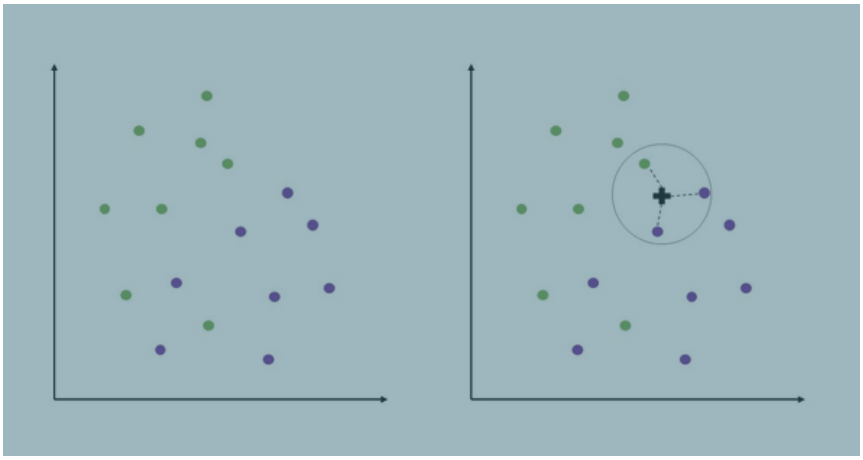


KNN can be used for either regression or classification tasks. KNN is non-parametric/instance based learning, which means that the algorithm does not make assumptions about the underlying distributions of the data. In contrast to a technique like linear regression, which is parametric/model based learning, and requires us to find a function that describes the relationship between dependent and independent variables.

Instance-based methods are sometimes called lazy learning methods because they postpone learning until the new instance/data point is encountered for prediction. There is no math or processes before testing the model, all it does is find the closest K points.

How does KNN work?

See the following example. The right panel shows how a new point (the black cross) is classified, using KNN when k=3. We find the three closest points, and count how many 'votes' each color has within those points. In this case, two of the three points are purple. So, the black cross will be labeled as purple.



KNN Algorithm

- Load the given data file into your program
- Initialize the number of neighbors to be considered i.e. 'K' (must be odd to avoid ties in voting).
- Now for each tuple (entries or data point) in the data file we perform:
  - Calculate the distance between the data point (tuple) to be classified and each data points in the given data file.
  - Then add the distances corresponding to data points (data entries) in the given data file (probably by adding a column for distance).
  - Sort the data in the data file from smallest to largest (in ascending order) by the distances.

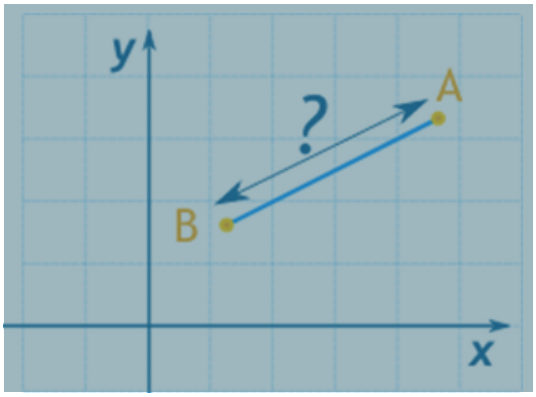
KNN Algorithm

- Pick the first K entries from the sorted collection of data.
- Observe the labels of the selected K entries.
- For classification, return the mode of the K labels and for regression, return the mean of K labels.

Distance Metrics

For the algorithm to work best on a particular dataset we need to choose the most appropriate distance metric accordingly. There are a lot of different distance metrics available, but we are only going to talk about a few widely used ones.

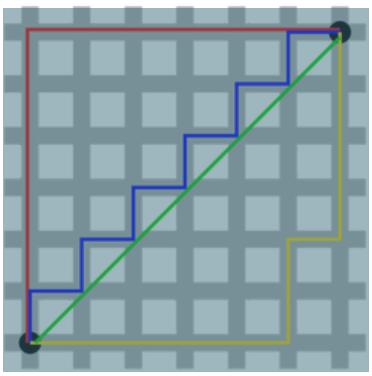
Euclidean distance function is the most popular one among all of them as it is set default in the Sklearn KNN classifier library in python.



Manhattan Distance

Also known as taxicab distance or city block distance, that is because the way this distance is calculated. The distance between two points is the sum of the absolute differences of their Cartesian coordinates.

$$d = \sum_{i=1}^{n} |x_i - y_i|$$

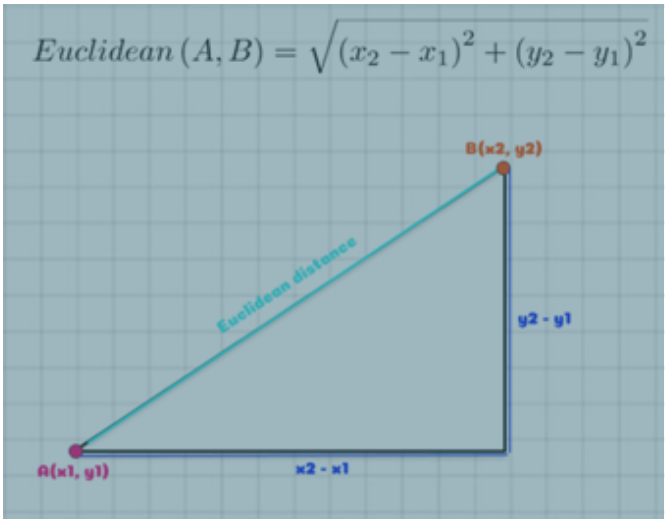


From the above picture, all the red, blue and yellow paths are Manhattan distance

Euclidean Distance

This distance is the most widely used one as it is the default metric that Sklearn library of Python uses for K-Nearest Neighbour. It is a measure of the true straight line distance between two points in Euclidean space.

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$



Above is the example in 2-dimensional space

Minkowski Distance

It is a metric intended for real-valued vector spaces. We can calculate Minkowski distance only in a normed vector space, which means in a space where distances can be represented as a vector that has a length and the lengths cannot be negative. Note that it is the formula for Manhattan when p=1 and Euclidean when p=2

$$\left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{1/p}$$

Chebysev Distance

Also known as the chessboard distance because “in the game of chess the minimum number of moves needed by a king to go from one square on a chessboard to another equals the Chebyshev distance between the centers of the squares.”

$$d(x, y) = \max_i |x_i - y_i|$$

Choosing the K-Value

- There is no structured method to find the best value for “K”. We need to find out various values by trial and error and assuming that training data is unknown.
- Choosing smaller values for K can be noisy and will have a higher influence on the result.
- Larger values of K will have smoother decision boundaries which mean lower variance but increased bias. Also, computationally expensive.
- Another way to choose K is through cross-validation. We’ll discuss later
- In general, practice, choosing the value of k is k = sqrt(N) where N stands for the number of samples in your training dataset.
- Try and keep the value of k odd in order to avoid confusion between two classes of data

Advantages of KNN

- No Training Period: KNN is called Lazy Learner (Instance based learning). It does not learn anything in the training period. It does not derive any discriminative function from the training data. In other words, there is no training period for it. It stores the training dataset and learns from it only at the time of predictions.
- Since the KNN algorithm requires no training before making predictions, new data can be added seamlessly which will not impact the accuracy of the algorithm.
- KNN is very easy to implement. There are only two parameters required to implement KNN i.e. the value of K and the distance function (e.g. Euclidean or Manhattan etc.)

Disadvantages of KNN

- Does not work well with large dataset: In large datasets, the cost of calculating the distance between the new point and each existing points is huge which degrades the performance of the algorithm.
- Does not work well with high dimensions: The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate the distance in each dimension.
- Need feature scaling: We need to do feature scaling (standardization and normalization) before applying KNN algorithm to any dataset. If we don't do so, KNN may generate wrong predictions.
- Sensitive to noisy data, missing values and outliers: KNN is sensitive to noise in the dataset. We need to manually impute missing values and remove outliers.

Usage of KNN

You can use the KNN algorithm for applications that require high accuracy but that do not require a human-readable model. The quality of the predictions depends on the distance measure. Therefore, the KNN algorithm is suitable for applications for which sufficient domain knowledge is available. This knowledge supports the selection of an appropriate measure. The KNN algorithm is a type of lazy learning, where the computation for the generation of the predictions is deferred until classification. Although this method increases the costs of computation compared to other algorithms, KNN is still the better choice for applications where predictions are not requested frequently but where accuracy is important.

Picture Source

- pexels.com