KNN:

 the ***k*-nearest neighbors algorithm** (***k*-NN**) is an algorithm used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis).

K-nearest neighbor classifier algorithms is to predict the target label by the majority vote of its closest class neighbors. The closest class will be identified using the distance measures like Euclidean distance.

Or

k-nearest-neighbor classification, the unknown tuple is assigned the most common class among its k nearest neighbors.

In *k-NN regression*, the output is the property value for the object. This value is the average of the values of its *k* nearest neighbors.

* KNN stores the entire training dataset which it uses as its representation.
* KNN does not learn any model.
* KNN makes predictions just-in-time by calculating the similarity between an input sample and each training instance.

**Advantages of K-nearest neighbors algorithm**

* Knn is simple to implement.
* Knn executes quickly for small training data sets.
* performance asymptotically approaches the performance of the Bayes Classifier.
* Don’t need any prior knowledge about the structure of data in the training set.
* No retraining is required if the new training pattern is added to the existing training set.
* Insensitive to outliers — accuracy can be affected from noise or irrelevant features
* No assumptions about data — useful, for example, for nonlinear data
* High accuracy (relatively) — it is pretty high but not competitive in comparison to better supervised learning models
* Versatile — useful for classification or regression

**Limitation to K-nearest neighbors algorithm**

* When the training set is large, it may take a lot of space.
* For every test data, the distance should be computed between test data and all the training data. Thus a lot of time may be needed for the testing.
* Computationally expensive — because the algorithm stores all of the training data
* High memory requirement
* Stores all (or almost all) of the training data
* Prediction stage might be slow (with big N)
* Sensitive to irrelevant features and the scale of the data

Note:

* When we say a technique is **non-parametric** , it means that it does not make any assumptions on the underlying data distribution. In other words, the model structure is determined from the data. If you think about it, it’s pretty useful, because in the “real world”, most of the data does not obey the typical theoretical assumptions made (as in linear regression models, for example). Therefore, KNN could and probably should be one of the first choices for a classification study when there is little or no prior knowledge about the distribution data.
* KNN is also a **lazy** algorithm (as opposed to an eager algorithm). Does that mean that KNN does nothing, like these polar bears imply??? Not quite. What this means is that it does not use the training data points to do any generalization. In other words, there is no explicit training phaseor it is very minimal. This also means that the training phase is pretty fast . Lack of generalization means that KNN keeps all the training data. To be more exact, all (or most) the training data is needed during the testing phase. KNN Algorithm is based on **feature similarity**: How closely out-of-sample features resemble our training set determines how we classify a given data point: