Overview:
The k-Nearest Neighbors Algorithm is one of the most fundamental and powerful Algorithm to understand, implement and use in classification problems when there is no or little knowledge about the distribution of data.

The algorithm determines the value of an unseen data point based on the value of its neighbors. It calculates euclidean distance, a beefed up Pythagorean Theorem, to determine the distance between a new point and the closest data. Depending on how many points are in the vicinity, it classifies the new data point accordingly. KNN can handle binary classification and beyond to "n" number of classes.

Difference Between KNN and K-means
KNN - Classification - Supervised - Labeled data
K-means - Clustering - Unsupervised - Unlabeled data

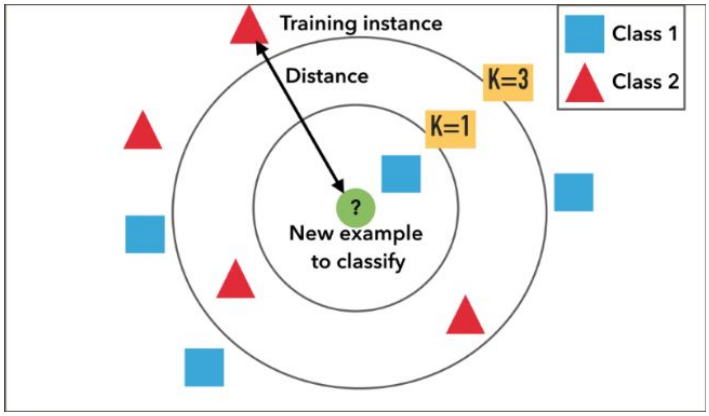
Example:
Suppose you are a class teacher who got a new student admitted in his/her class. You want to find a proper friend's circle for the new student. Since you want to place the student with similar students, you look for certain characteristics like aptitude, interests, affinity towards sports. Based on these three characteristics you place the student to its 'nearest neighbors'. The 'nearness' is measured bases on the three characteristics.

KNN:

KNN algorithm is one of the simplest classification algorithm and it is one of the most used learning algorithms. So what is the KNN algorithm? I'm glad you asked! KNN is a non-parametric, lazy learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.

When we say a technique is **non-parametric** , it means that it does not make any assumptions on the underlying data distribution. 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 Algorithm is based on **feature similarity:** How closely out-of-sample features resemble our training set determines how we classify a given data point:

Fig: Example of k-NN classification. The test sample (inside circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (outside circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If, for example k = 5 it is assigned to the first class (3 squares vs. 2 triangles outside the outer circle).



Examples:
1. Should the bank give a loan to an individual? Would an individual default on his or her loan? Is that person closer in characteristics to people who defaulted or did not default on their loans?

Pros:
- No assumptions about data — useful, for example, for nonlinear data
- Simple algorithm — to explain and understand/interpret
- High accuracy (relatively) — it is pretty high but not competitive in comparison to better supervised learning models
- Versatile — useful for classification or regression

Cons:
- 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

Quick summary of KNN
The algorithm can be summarized as:
1. A positive integer k is specified, along with a new sample
2. We select the k entries in our database which are closest to the new sample
3. We find the most common classification of these entries
4. This is the classification we give to the new sample
5. KNN stores the entire training dataset which it uses as its representation.
6. KNN does not learn any model.
7. KNN makes predictions just-in-time by calculating the similarity between an input sample and each training instance.