**KNN: (K-Nearest Neighbor)**

KNN algorithm can be applied to both classification and regression problems. This algorithm takes the entire dataset as the training data, instead of partitioning the dataset into training set and test set. Since this algorithm takes the entire dataset as training set, there is no model and hence no learning required. But as it stores the entire data in memory, we should be able to guarantee that data is updated frequently, removal of outlier data and missing data. When the new instance is given, KNN algorithm searches for entire dataset for finding K most similar instances (neighbor) against the new instance, and predicts the output for the K-instances.

For **regression** KNN algorithm finds the mean of k similar instances to predict the output.

For **classification,** KNN finds the classes having higher frequency from the k-most similar instances.

To find out that k similar instances from the training dataset, Euclidean distance and hamming distance are used. Euclidean distance (x, xi) = sqrt (sum ((xj-xij) ^2)), where x is the new point, xi is the existing point and j is the input attributes across the dataset.

**Curse of dimensionality:**

KNN algorithm works very well for small input variables (p). If we have two inputs say x1, x2 the input space requires 2-dimensional space. As the number of inputs increases, the input space also increases, and hence the distance between the k-similar instances also increases, which makes the algorithm behave in an unpredicted way. This is called as curse of dimensionality as KNN works well for 2 and 3 dimensional spaces.

**Advantages of KNN:**

* Does not require any model for learning
* No training is required
* Robust
* Complex models can be easily learnt

**Disadvantages of KNN:**

* Slow testing phase
* Requires more storage space
* Does not work well for high dimensional space