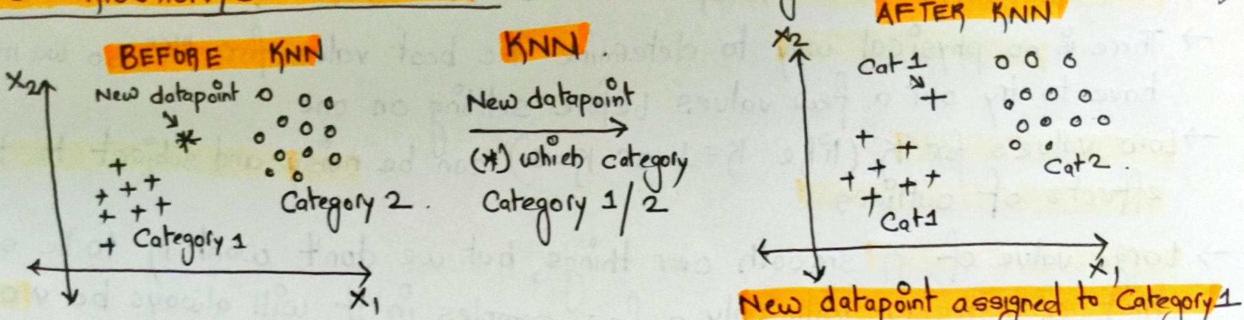
KNN DISTRIBUTION/CLASSIFICATION → K Nearest Neighbour Algorithim (Non parametric)



New datapoint assigned to Category1

STEPS TO FOLLOW IN KNN

Step 1 - Choose the number K of neighbours.

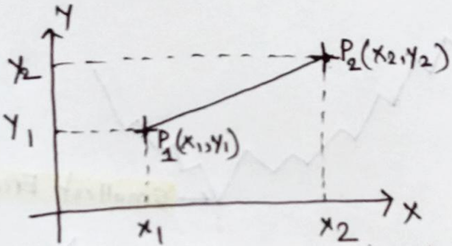
Step 2 - Take the K nearest neighbour of new datapoint, according to the Euclidean distance.

Step 3 - Among the K neighbours, count the number of data point in each category.

Step 4 - Assign the new datapoints to the category where you counted the most neighbour.

Your Model is Ready.

Euclidean Distance →

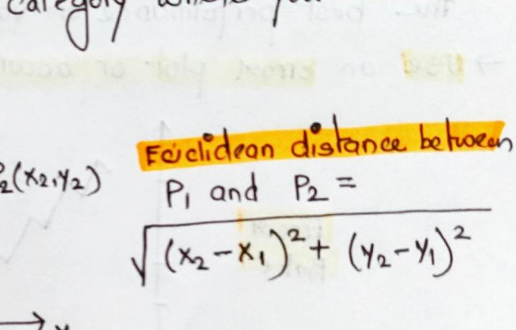


Euclidean distance between P_1 and P_2 =

$$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

Suppose we choose K=5, so according to new data point, new data point consist of 2 categories. Category 1 = 3 + Category 2 = 2 O so, based on 3 +, new data point is assigned to '+' category.

Example → We need to define the cell types → Stem Cells, Blood Vessel Cells, Fat Cells



Step 1 → Start with the dataset with know categories. In this case, we have different cell types from a tumor. Then cluster the data. In this case, we used PCA.

Step 2 → Add a new cell, with unknown category to the PCA plot. We don't know this cell category because it was taken from another tumor. So we need to classify the new unknown cell.

Step 3 → We classify the new cell by looking at the nearest annotated Cells (i.e, nearest neighbours) to define the category.

If K is equal to 1, then we only use the nearest neighbour to define the category.

If K is equal to 11, we would use 11 nearest neighbours and based on majority assign the class.

Based on the most votes, assign the new data point to the class having most votes.

NOTE → If K is odd, then we can avoid ties (equal count to each group) and if we still get a tied vote, we can flip a coin/decide not to assign any category

A FEW THOUGHTS ON PICKING A VALUE OF "K".

→ There is no physical way to determine the best value for "K", so we may have to try out a few values before settling on one.

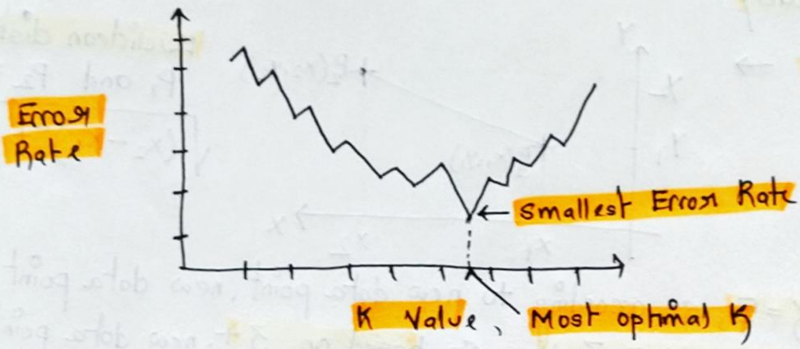
→ Low values for K (like K=1 or K=2) can be noisy and subject to the effects of outliers.

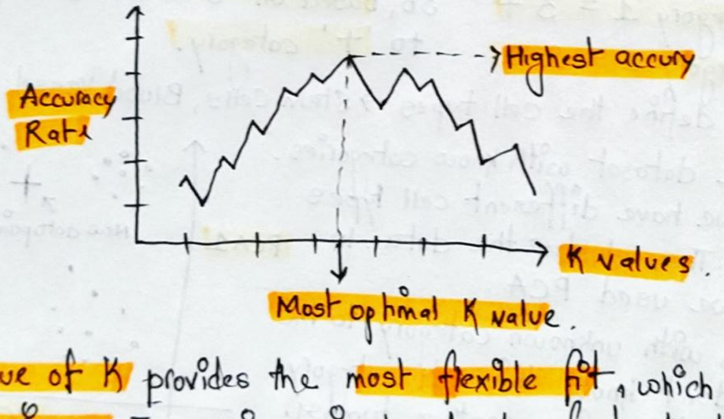
→ Large value of K smooth over things, but we don't want K to be so large that a category with only a few samples in it will always be ~~p~~ out voted by other categories.

→ In general practise, choosing the value of K is K = sqrt (N) where N stands for Number of samples in training dataset.

→ Another way to choose K is through cross-validation. One way to select different possible value of K and check for what value of K gives us the best performance on validation set.

→ Use an error plot or accuracy plot to find the most favourable K value.





→ A small value of K provides the most flexible fit, which will have low bias and high variance. The variance is due to the fact that the prediction in a given region is entirely dependent on just one observation.

→ Large value of K provide a smoother and less variable fit, the prediction in a region is an average of ~~seva~~ several points and so changing one observation has smaller effects. However smoothing may cause bias. High bias, low variance.

KNN (K- Nearest Neighbour) - Non linear classifier

- Identifies data point that are seperated into several classes to predict the classification of a new sample point.
- It is a LAZY algorithim, it does not learn anything, what it does simply is based on current set (training set) it classify the new sample point based on majority.
- KNN classifies new point based on similarity measure.

Algorithim -

i) Initialize K - ~~K~~ For number of K, it will search only K elements. For eg, if K=5, it will try to find nearest 5 elements to the new sample point. if K=1, it will find the closest 1 element element to the sample point.

- Remember K must be odd, because if K = even, then there is a chance of division of equal votes. If K = odd, maximum chance division of votes will be unequal

ii - K must not be multiple of classes (target), suppose target = 9, then K should not be 9 or 18 because of equal division of votes between target.

ii) For each sample in training data, calculate distance between query points and other points. Distance can be manhatten distance, $= \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

iii) Collect the distance and index them in ordered collection.

iv) Sort the ordered collection of distances and index from small to large.

v) Pick first K elements from sorted Collection.

vi) Get the labels of selected K entries.

vii) For classification → ~~Retu~~ Calculate mode of K labels, highest mode class, assign it

regression → Calculate mean of K labels, assign the mean value.