Date : Supervised Learning
- Nearest neighbour
- Naive Bayes
- Decision Trees
- Support Vector Machines (SVM)
- Neural Networks.
- Random Forest
- Similarity Learning
- Linear Regression
- Linear Discriminant analysis

K - Nearest Neighbours

It is a non-parametric & Lazy algo

Non parametric means it doesnot depend on the data rather depends upon the proximity to other data points ~~regardless~~.
=) It doesnt make assumptions on the underlying data.

~~Parametric~~ ~~n~~
Lazy learning algo implies there is little to no training phase.
=) We can immediately ~~train~~ classify new data points as they present themselves

Pros:

→ No assumption about ~~data~~ data

→ Simple & easy to understand

→ can be used for classification & regression

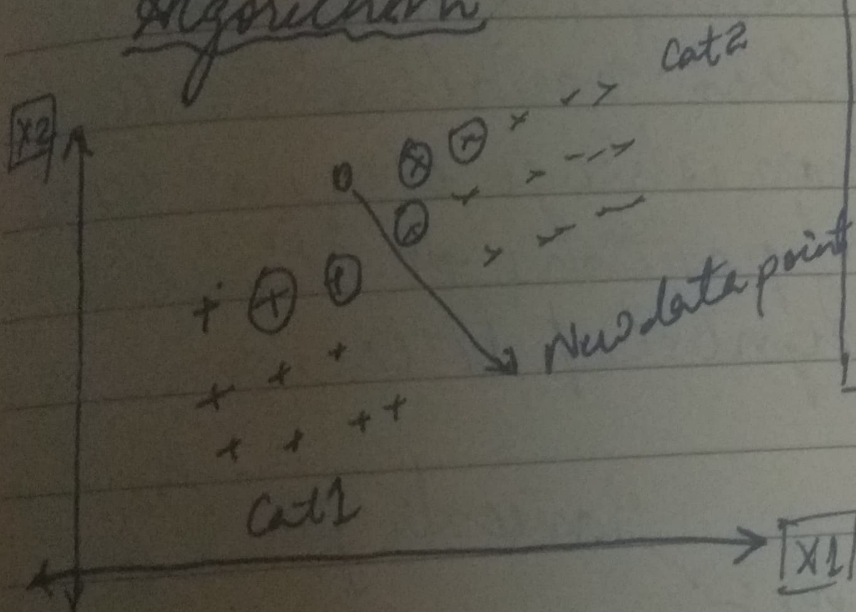
Cons:

→ High memory req: all the training data must be present in memory in order to calc the closest k-neighbours

→ sensitive to irrelevant features

→ sensitive to the scale of the data since we're computing the distance to the closest k-point.

Algorithm



(x) → 3 neighbours

(+) → 2 neighbours

(i) Pick a value for K. (say K = 5)

(ii) Take 'K' nearest neighbours from the data point according to its Euclidean distance

(iii) Among these neighbours count the number of data points in each category & assign the new data point to that category.

Euclidean – input of same dtype
Manhattan – " " diffⁿ dtype

Date: CODE:- TUNING OF KNN:-

→ Trying larger 'k' values to see if we could improve the performance of the algorithm.

→ Try different distance measures like ~~the~~ Hamming distance

Hamming distance: calculates the distance between two binary vectors.
In case of categorical variables, it is the difference b/w two strings of equal length i.e. the number of positions at which the corresponding symbols / letters are different.

Manhattan Distance:- Taxicab geometry

$$d_1(p,q) = \|p-q\| = \sum_{i=1}^{n} |p_i - q_i|$$

the sum of the absolute differences of their cartesian coordinates.

Minkowski Distance → Generalization of both the Euclidean & Manhattan distance

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

Curse of Dimensionality

Date :

KNN works well with a small number of input variables (p), but struggles when the number of inputs is very large.

Each input variable can be considered as each dimension for instance ~~if~~ x_1, x_2 are two ~~if~~
x1 & x2 are two variables..
⇒ the input space would be 2-dimensional

As the number of dimensions increases
⇒ the volume of input space increases
⇒ points may be similar may have large distances
⇒ Curse of Dimensionality.