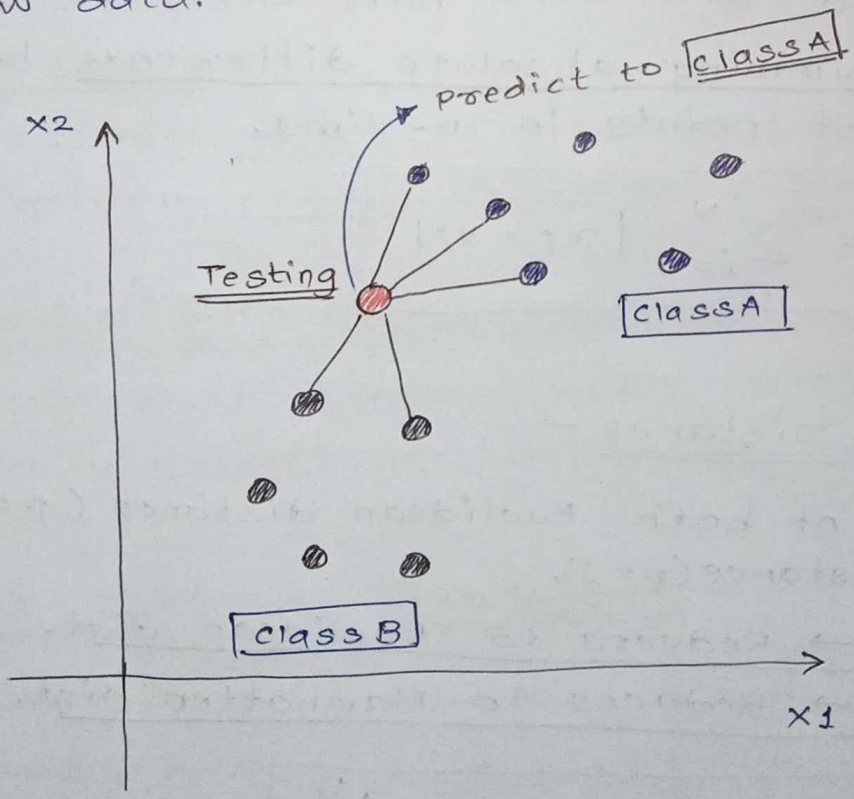
* K-Nearest Neighboors (KNN) *

KNN Introduction :-

- Supervised Learning Algo.
- Intuitive classification & Regression Algo.
- Predict new data, by finding similarity with existing data points in training dataset.
- Mostly used for classification.
- Non-parametric algo: Not make any assump^{n} on underlying data.
- Lazy learner algo - Non learn training set immediatly, instead store dataset & at classifica^{n} time, perform action.
- KNN at training phase just store dataset & on new data, classify data into category similar to new data.



• Dist. Metrics used in KNN :-

↳ Dist. Metrics used to find dist. of new point to the original points.

↳ Then find class based on voting.

① Euclidean Distance -

↳ Calculate straight-line dist. bet^n 2 points in Euclidean space.

↳ Suitable for continuous numerical data.

$$d(x,y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$

② Manhatten Distance -

↳ Calculate dist. by summing absolute differences bet^n corresponding feature values.

↳ Suitable data with attr have different units.

↳ Calculate summing absolute difference bet^n coordinates of points in n-dims.

$$d(x,y) = \sum_{i=1}^{N} |x_i - y_i|$$

③ Minkowski Distance -

↳ Generaliza^n of both Euclidean distance ($p=2$) & Manhatten Distance ($p=1$).

↳ When $p=2 \rightarrow$ Reduces to Euclidean dist.

$p=1 \rightarrow$ Reduces to Manhatten dist.

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

- How to choose value of K :

↳ Value of 'k' in KNN impact algo's performance.

① Cross - Validation —

↳ Split data into training & validation dataset.

↳ Train KNN Model for different values of 'k'.

↳ Evaluate performance : Regr^n (mean squared error) / classifica^n (Accuracy).

↳ choose 'k' with best validation performance.

② Odd 'k' Value —

↳ choose odd 'k' values to avoid ties, when voting.

↳ If 'k' even : result in ambiguous predictions.

③ Rule of Thumb —

↳ 'k' - Around square root of num of data points.

↳ As data ↑es ⇒ smaller 'k' better.

④ Grid Search —

↳ Perform grid search over range of 'k' values.

↳ Perform cross - validation & find best 'k'.

⑤ Consider Data characteristics —

↳ Smaller 'k' capture local patterns.

↳ Noisy dataset ⇒ Larger 'k' smooth.

⑥ Bias - Variance Trade-off —

↳ Smaller 'k' : Low bias & high variance.

↳ Larger 'k' : High bias & low variance.

↳ Should balance based on datates.

- **How KNN Works:**

① **Select Number K of neighbors –**

↳ 'k' should be odd to avoid ties.

↳ 'k' should not so smaller or very large.

↳ Should balance value of 'k'.

② **Calculate Euclidean dist. of K neighbors –**

Example: From below dataset, Predict o/p for:

$X = (Math = 6, CS = 8)$ for $k = 3$.

Math	CS	Result
4	3	Fail
6	7	Pass
7	8	Pass
5	5	Fail
8	8	Pass

- **Find Euclidean dist. of all Points:**

$$d = \sqrt{|x_{01} - x_{A1}|^2 + |x_{02} - x_{A2}|^2}$$

① $\sqrt{(6-4)^2 + (8-3)^2} = \sqrt{29} = 5.38$

② $\sqrt{(6-6)^2 + (8-7)^2} = \sqrt{1} = 1$

③ $\sqrt{(6-7)^2 + (8-8)^2} = \sqrt{1} = 1$

④ $\sqrt{(6-5)^2 + (8-5)^2} = \sqrt{10} = 3.16$

⑤ $\sqrt{(6-8)^2 + (8-8)^2} = \sqrt{4} = 2$

③ Find K-nearest neighbors by distance -

②, ③ & ⑤ are 3-nearest neighbors.

④ Perform Voting in KNN -

② → Pass
③ → Pass
⑤ → Pass

} 'Pass' has majority.

⑤ Assign Point to majority class -

∴ Point (6, 8) belongs to 'Pass'.

- Applications of KNN :-

① Data Preprocessing - can use KNN to find the missing values cla imputation method.

② Pattern Recognition - Train KNN model with MNIST dataset & then perform evaluation process results too high accuracy.

③ Recommendation Engines - Assign each user to particular group & then provide recommendations based on group's preferences.

- **Advantages of KNN –**

① **Easy to implement** – complexity not so high.

② **Adapt Easily** – Data stored in memory, so for new data point, algo adjusts for new point very easily.

③ **Few Hyperparams** – Only K & distance metrics.

④ **Suitable** – For both Classification & Regression.

- **Limitations of KNN –**

① **Does not scale** – Lazy learner: computing power & more storage. Time-consuming & resource-consume.

② **Curse of Dimensionality** – Hard to classify data points when too high dimensionality.

③ **Prone to overfitting** – Lead to overfitting, bcz of curse of dimensionality.
So, use feature selection & dimensionality reduction.