Mithilesh Singh

KNN regression/Classifier

-non-parametric method

- approximates the association between independent variables and the continuous outcome by averaging the observations in the same *neighborhood*.

-The size of the neighborhood K value needs to be set by the analyst or can be chosen using cross-validation to select the size that minimizes the mean-squared error.

Parametric V/s Non-parametric method

Parametric Method-assume the data is of sufficient "Quality", exp :- Linear Regression, Logistic Regression.

- **The result can be misleading if assumptions are wrong.**
- **Quality is defined in terms of certain properties of the data, like Normally distributed ,Symmetrical linear distribution, homogeneity of variance etc.**

Non Parametric tests can be used when the data is not of sufficient quality to satisfied the assumptions of parametric test. Non parameter tests STILL have assumptions but are less stringent.

Non parameter tests can be applied to Normal distributed data but parametric tests have greater power IF assumptions met.

- Parametric tests are preferred when the assumptions are met because they are more sensitive.

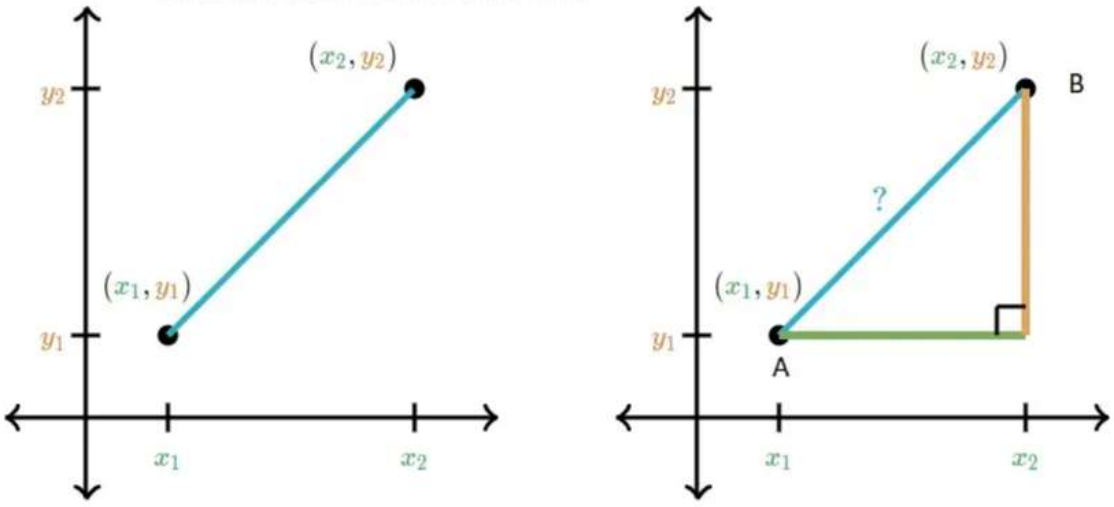
KNN model classifies the points based on proximity or distance.

Important Distance Metrics in Machine Learning

- **Euclidean Distance**
- **Manhattan Distance**
- **Minkowski distance**

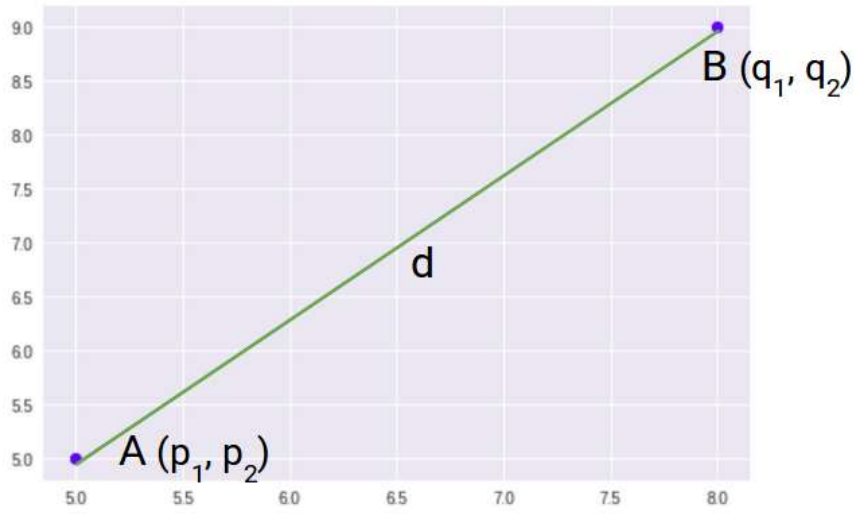
1. Euclidean Distance

Euclidean Distance represents the shortest distance between two points.



Pythagorean theorem

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



The 2 dimensions' formula for Euclidean Distance:

$$d = ((p_1 - q_1)^2 + (p_2 - q_2)^2)^{1/2}$$

For n-dimensional space as:

$$D_e = \left(\sum_{i=1}^{n} (p_i - q_i)^2 \right)^{1/2}$$

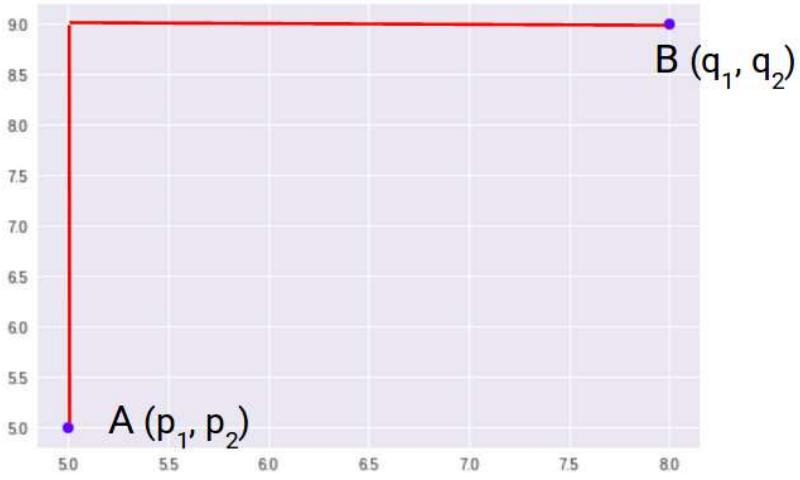
Where,

- n = number of dimensions
- pi, qi = data points

2. Manhattan Distance

Manhattan Distance is the sum of absolute differences between points across all the dimensions. Also called L1 Norm ,Taxi Cab Norm.

We can represent Manhattan Distance as:



Manhattan Distance, sum of absolute distances x and y directions.

In a 2-dimensional space is given as:

$$d = |p_1 - q_1| + |p_2 - q_2|$$

And the generalized formula for an n-dimensional space is given as:

$$D_m = \sum_{i=1}^{n} |p_i - q_i|$$

Where,

- n = number of dimensions
- pi, qi = data points

(3)Minkowski distance:

This distance measure is the generalized form of Euclidean and Manhattan distance metrics.

Euclidean distance is represented by this formula when p is equal to 2, and Manhattan distance is denoted with p equal to 1.

Minkowski distance=

$$\left(\sum_{i=1}^{n} |p_i - q_i|^p\right)^{1/p}$$

→ KNN model used for classification (and regression).

→ → KNN uses distance metrics in order to find similarities or dissimilarities.

- working off the assumption that similar points can be found near one another.
- The distinction between these terminologies is that "majority voting"

Example of KNN- Euclidean distance in real life

Liability→ Person	a	b	c	d	Status (loan approved)
1	160	10	20	2000	No
2	180	20	30	5000	Yes
3	200	30	35	8000	Yes
4	150	20	25	4000	No
5	350	60	100	6500	Yes

--					
100	200	50	80	7500	Yes
A	175	35	30	4500	?

Euclidean distance: -

A-1→square-root(15 sq +25 sq+10 sq +2500 sq)

A-2→ square-root (5 sq+15 sq+ 0 sq+500 sq)

A-3→ square-root (25 sq+5 sq+5 sq+3500 sq)

A-4→ square-root (25 sq+15 sq +5 sq+ 500 sq)

A-5→ square-root (175 sq+25 sq+70 sq+2000 sq)

A-100→……

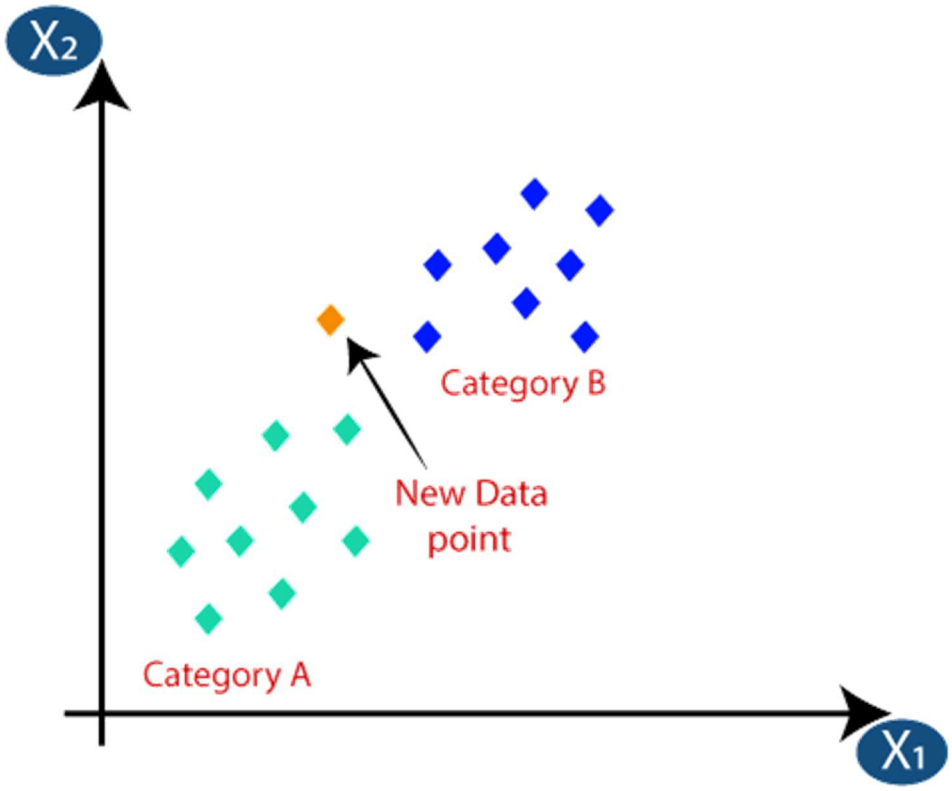
Nearest neighbor →lowest distance…

KNN method working concept:-

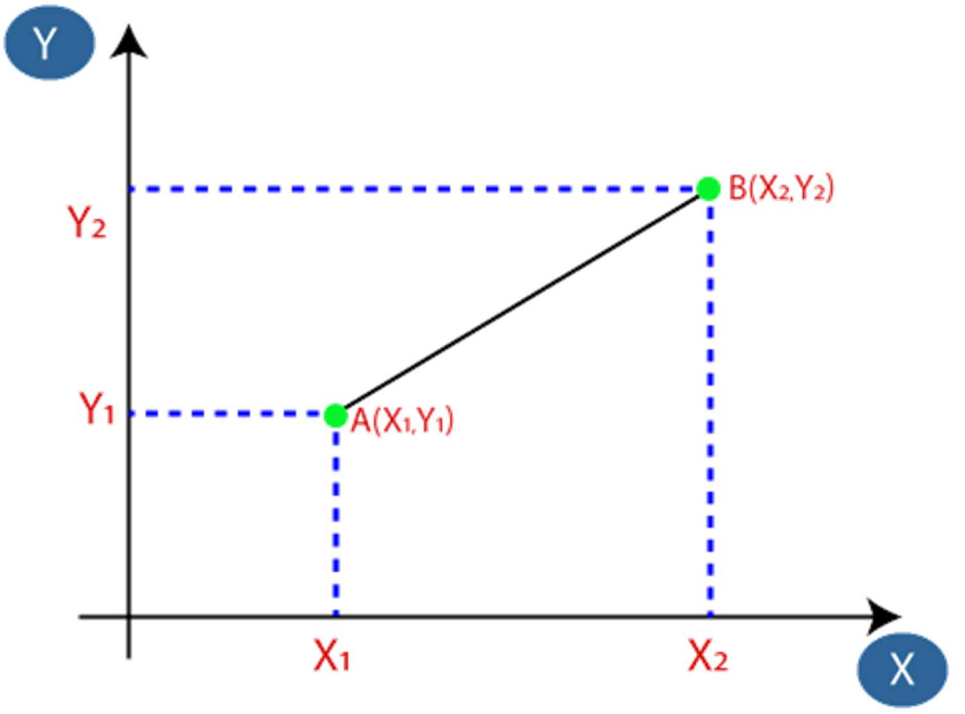
The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

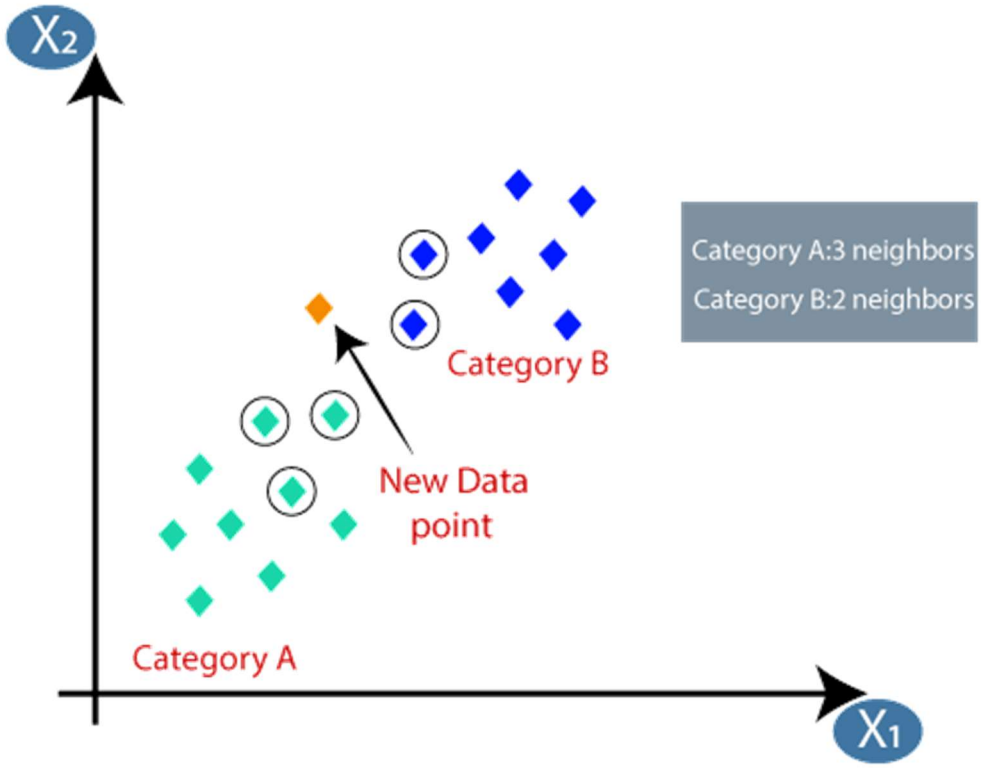


- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

- KNN can be used in regression, the target continuous value is computed as the mean of the target value of k nearest neighbors.

For classification Problems:-

from sklearn.neighbors import KNeighborsClassifier

FOR Regression Problems: -

from sklearn.neighbors import KNeighborsRegressor

How to select the optimal value of k?

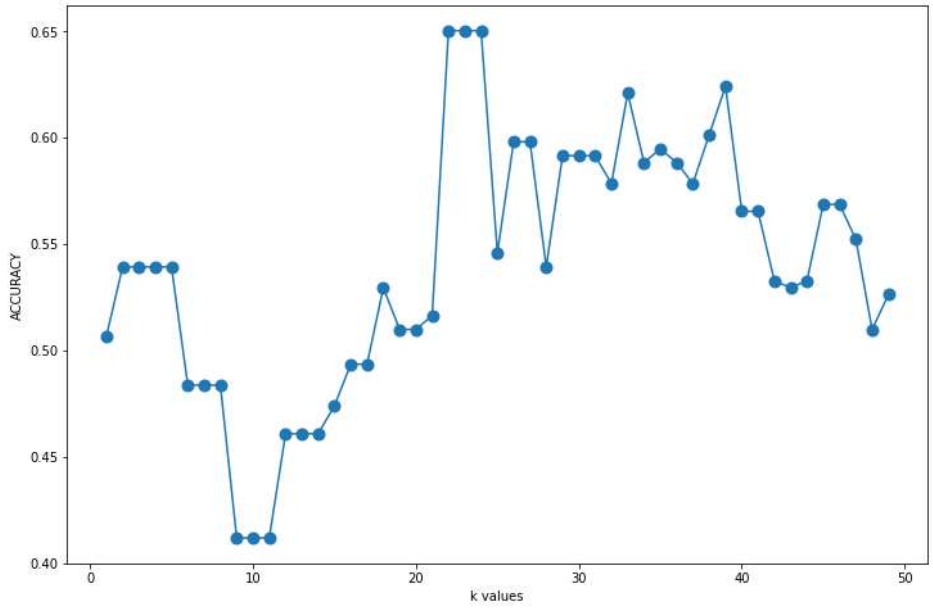
- Prefer odd k values, as there are chances of tie in even values.
- K should not be too small.
- Thumb rule: k should be generally sqrt(n), where n denotes the total number of data points
- Further, one can try different values of k, and then observe the evaluation metrics to decide upon the best value of k

Python code:-

```
from sklearn.neighbors import KNeighborsClassifier
model_name = ‘K-Nearest Neighbor Classifier’
knnClassifier = KNeighborsClassifier(n_neighbors = 5, metric =
‘minkowski’, p=2)
knn_model = Pipeline(steps=[(‘preprocessor’,
preprocessorForFeatures), (‘classifier’ , knnClassifier)])
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
```

 # find the more effective value of n_neighbors parameter k value:

```
accuracy_K = []
for k in range(1, 50):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train1, Y_train)
    Y_pred = knn.predict(X_test)
    accuracy =accuracy_score(Y_test, Y_pred)
    accuracy_K.append(accuracy)
plt.figure(figsize=(12,8))
plt.xlabel("k values")
plt.ylabel("ACCURACY")
plt.plot(range(1,50),accuracy_K, marker='o', markersize=9)
```



KNN is a Lazy Learner model

- Almost all the models get trained on the training dataset, but KNN does not get

trained on the training dataset.

- When we use knn.fit (X train, Y train), this model 'memorizes' the dataset. It does not understand it or tries to learn the underlying trend.
- Now when we ask the model to predict some value, then it takes a lot of time because now it actually will have to recall all the points and work around them so that it can help predict the correct value.
- Hence where most of the models, take time during training, this model does not take any time during training.
- Most of the models take no time in prediction, but the KNN model takes a lot of time during the prediction stage.

Important points

- **Since it is a distance-based model, feature scaling is a must for it.**
- **Besides logistic regression, the rest all the classification models can work on multi class classification.**

Advantages of KNN Algorithm:

- It is simple to implement.
- When the label data is too expensive or impossible to obtain.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- Entire dataset is processed for every prediction, not good for large dataset.

The computation cost is high because of calculating the distance between the data points for all the training samples. computational Time complexity for each prediction is equal to MNlog(k)

M is the dimension of data, N is the size of instance of training data and K is nearest point selected.