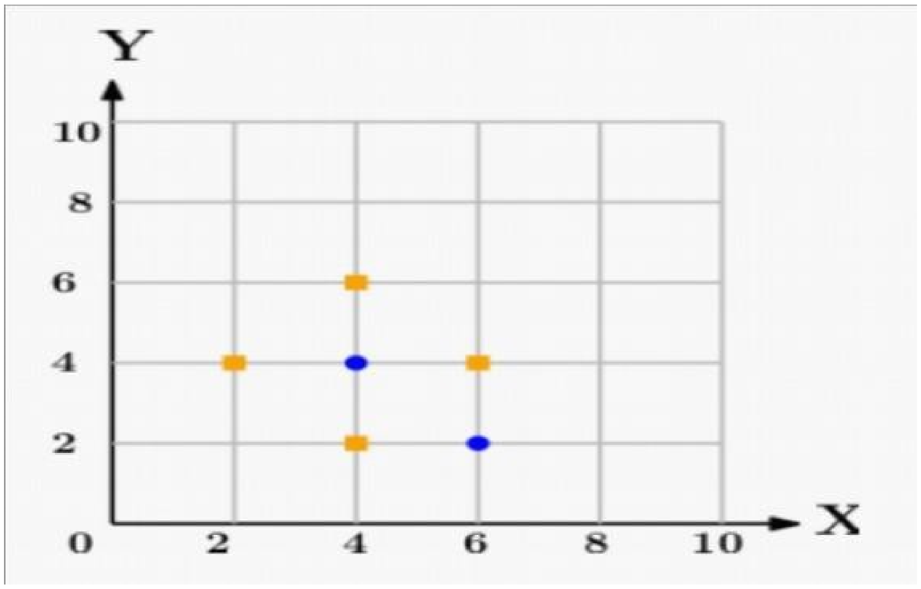
Name : Prem Vinod Bansod
Roll No : 41310
Assignment No : 03 (ML)

Title: k-NN Classification

Problem Statement: In the following diagram let blue circles indicate positive examples and orange squares indicate negative examples. We want to use k-NN algorithm for classifying the points. If k=3, find the class of the point (6,6). Extend the same example for Distance-Weighted k-NN and Locally weighted Averaging.



Objective:

- The Basic Concepts of k-NN algorithm.
- Implementation logic of k-NN algorithm for classifying the points on given graph.

Theory:

Introduction:

KNN is a non-parametric, lazy learning algorithm. Its purpose is to use a database in which the data pointsare separated into several classes to predict the classification of a new sample point. When we say a technique is non-parametric, it means that it does not make any assumptions on the underlying data distribution. In other words, the model structure is determined from the data. Therefore, KNN could and probably should be one of the first choices for a classification study when there is little or no prior knowledge about the distribution data. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph. There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight-line distance (also called the Euclidean distance) is a popular and familiar choice.

KNN Algorithm

We can implement a KNN model by following the below steps:

- Load the data
- Initialize the value of k
- For getting the predicted class, iterate from 1 to total number of training data points
    - Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it‘s the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.

☟ - Sort the calculated distances in ascending order based on distance values

☟ - Get top k rows from the sorted array

☟ - Get the most frequent class of these rows

☟ - Return the predicted class

Choosing the right value for K

To select the K that‘s right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm‘s ability to accurately make predictions when it‘s given data it hasn‘t seen before.

Here are some things to keep in mind:

- As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, image K=1 and we have a query point surrounded by several reds and one green (I‘m thinking about the top left corner of the colored plot above), but the green is the single nearest neighbor. Reasonably, we would think the query point is most likely red, but because K=1, KNN incorrectly predicts that the query point is green.
- Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing

number of errors. It is at this point we know we have pushed the value of K too far.

- In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

Advantages

- No assumptions about data — useful, for example, for nonlinear data
- Simple and easy to implement algorithm — to explain and understand/interpret
- High accuracy (relatively) — it is pretty high but not competitive in comparison to better supervised learning models
- Versatile — useful for classification or regression

Disadvantages

- Computationally expensive — because the algorithm stores all of the training data
- High memory requirement
- Stores all (or almost all) of the training data
- Prediction stage might be slow (with big N)
- Sensitive to irrelevant features and the scale of the data
- The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

Conclusion: We have studied the k-NN Classification algorithm and also implemented successfully.