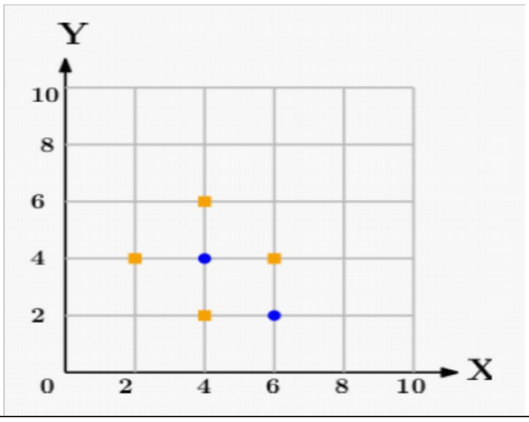
| | |
|---|---|
| **ASSIGNMENT NO** | A3 |
| **TITLE** | **Assignment on k-NN Classification** |
| **PROBLEM STATEMENT/ DEFINITION** | In the following diagram let blue circles indicate positive examples and orange squares indicate negative examples. We want to use k-NN algorithm for classifying the 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** | To understand how kNN algorithm works on the given dataset |
| **OUTCOME** | To implement kNN classification algorithm. |
| **S/W PACKAGES AND HARDWARE APPARATUS USED** | Core 2 DUO/i3/i5/i7 64-bit processor<br>OS-LINUX 64 bit OS<br>Editor-gedit/Eclipse<br>S/W- Jupyter Notebook/Weka/Python |
| **REFERENCES** | 1. Giuseppe Bonaccorso, “ Machine Learning Algorithms”, Packt Publishing Limited, ISBN-10: 1785889621, ISBN-13: 978-1785889622<br><br>2. Josh Patterson, Adam Gibson, “Deep Learning : A Practitioners Approach”, O’REILLY, SPD, ISBN: 978-93-5213-604-9, 2017 Edition 1st.<br><br>3. Nikhil Buduma, “Fundamentals of Deep Learning”, O’REILLY publication, Second Edition, 2017,ISBN: 1491925612 |
| **STEPS** | 1. Load the data |

|  |  |
|---|---|
|  | 2. Initialize K to your chosen number of neighbors<br><br>3. For each example in the data<br><br>3.1 Calculate the distance between the query example and the current example from the data.<br><br>3.2 Add the distance and the index of the example to an ordered collection<br><br>4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances<br><br>5. Pick the first K entries from the sorted collection<br><br>6. Get the labels of the selected K entries<br><br>7. If regression, return the mean of the K labels<br><br>8. If classification, return the mode of the K labels |
| **INSTRUCTIONS FOR WRITING JOURNAL** | 1. Date<br>2. Assignment No.<br>3. Problem Definition<br>4. Learning Objective<br>5. Learning Outcome<br>6. Concepts Related Theory<br>7. Algorithm<br>8. Test Cases<br>9. Conclusion/Analysis |

- **Prerequisites:** Basic knowledge about Algorithms and any programming knowledge Java/python

- **Concepts related Theory**

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well −

- **Lazy learning algorithm** − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
- **Non-parametric learning algorithm** − KNN is also a non-parametric learning algorithm because it doesn’t assume anything about the underlying data.

**Working of KNN Algorithm**

K-nearest neighbors (KNN) algorithm uses ‘feature similarity’ to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps −

**The KNN Algorithm**

1. Load the data

2. Initialize K to your chosen number of neighbors

3. For each example in the data

3.1 Calculate the distance between the query example and the current example from the data.

3.2 Add the distance and the index of the example to an ordered collection

4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances

5. Pick the first K entries from the sorted collection

6. Get the labels of the selected K entries

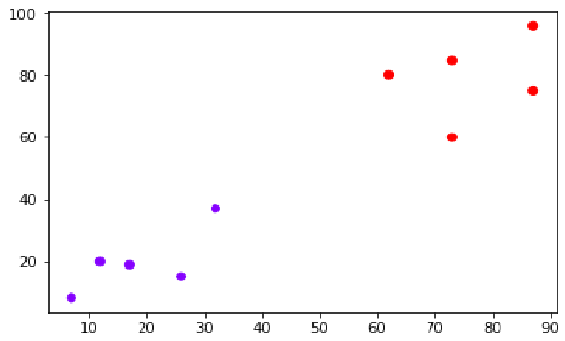
7. If regression, return the mean of the K labels

8. If classification, return the mode of the K labels

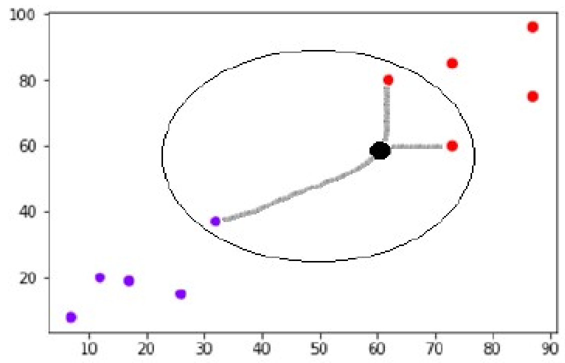
## Example

The following is an example to understand the concept of K and working of KNN algorithm −

Suppose we have a dataset which can be plotted as follows −



Now, we need to classify new data point with black dot (at point 60,60) into blue or red class. We are assuming K = 3 i.e. it would find three nearest data points. It is shown in the next diagram −



We can see in the above diagram the three nearest neighbors of the data point with black dot. Among those three, two of them lies in Red class hence the black dot will also be assigned in red 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:

1. As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, imagine 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.
2. 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.
3. 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

1. The algorithm is simple and easy to implement.
2. There's no need to build a model, tune several parameters, or make additional assumptions.
3. The algorithm is versatile. It can be used for classification, regression, and search (as we will see in the next section).

**Disadvantages**

1. The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

## KNN in practice

KNN's main disadvantage of becoming significantly slower as the volume of data increases makes it an impractical choice in environments where predictions need to be made rapidly. Moreover, there are faster algorithms that can produce more accurate classification and regression results.

However, provided you have sufficient computing resources to speedily handle the data you are using to make predictions, KNN can still be useful in solving problems that have solutions that depend on identifying similar objects. An example of this is using the KNN algorithm in recommender systems, an application of KNN-search.

Test Cases:
Split dataset into training and testing dataset.