**4.3 Applications**

KNN application on student result prediction

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. k-Nearest Neighborhood (k-NN) classification is a method adoptable for classifying entities based on closest training examples in a feature space. k-NN is a lazy learning classifiers that adopts instance-base learning hence having prediction done in two stages. Firstly, it undergoes minimal operations of analyzing the attribute values of individual instances in training dataset [23]. k-NN can be been trained for online and genuine time analysis of data to identify interestingness intrinsical to data stream, match particular utilizer group for relegation, or recommend exhaustive options that meet categorical users’ needs. Though the technique requires extravagant resources during computation, but it is transparent, consistent, straightforward, simple and facile to implement with high proclivity to possess desirable qualities than most other data mining techniques, categorically when there is diminutive or no prior erudition about data distribution. Popular techniques of K-NN classifiers are Euclidean distance or cosine homogeneous attribute between training and test datasets. In both techniques, the entity to be presaged is assigned a mundane class among its k-most proximate neighbors, weights are assigned to culled variables, and the strepitous data are pruned.



Fig : [Image showing how similar data points typically exist close to each other](https://commons.wikimedia.org/wiki/File:Map1NNReducedDataSet.png)

Notice in the image above that most of the time, kindred data points are proximate to each other. The KNN algorithm hinges on this postulation being true enough for the algorithm to be utilizable. KNN captures the conception of kindred attribute (sometimes called distance, proximity, or propinquity) 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.

**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

**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.

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It’s easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows.

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

In the case of classification and regression, we saw that choosing the right K for our data is done by trying several Ks and picking the one that works best.

In the proposed system we use result of those 500 students, whose data has been trained by KNN Classifier method. The data is genuinely a combination of their CGPA's and results in their SSC, HSC & 1-12th semesters in their university period & evaluate the performance of each student predicated on the data they provided or predicated on their educational information.

After experimenting the system with KNN classifiers we input the student’s data to verify which level of performance the student performs. Here the performance level is an attributes to verify the student. In this system we collect 500 student’s data and use 80% data to train the system and 20 % data is predicted by the system using KNN classifiers method and this prediction of the system is the result of the proposed model. In the end we predict 96% accurate result and 4% is false predictions.

We use KNN classifiers method in this system because we gain maximum accuracy for the model. Here –

1. We utilize the missing value properly
2. No unnecessary attributes.
3. Use different values to gain maximum accuracy

We have majorly focused on more the accuracy & gaining correct information. The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

**4.4 Conclusion**

KNN relies on majority voting based on class membership of K nearest samples for a given test data sample. The nearness of samples is typically based on Euclidean distance. Most of the times, your dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Eucledian distance between two data points in their computations, this is a problem. If left alone, these algorithms only take in the magnitude of features neglecting the units.

The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes. To suppress this effect, we need to bring all features to the same level of magnitudes. Any algorithm where distance play a vital role for prediction or classification, we should normalize the feature as we do the same process in KNN also.