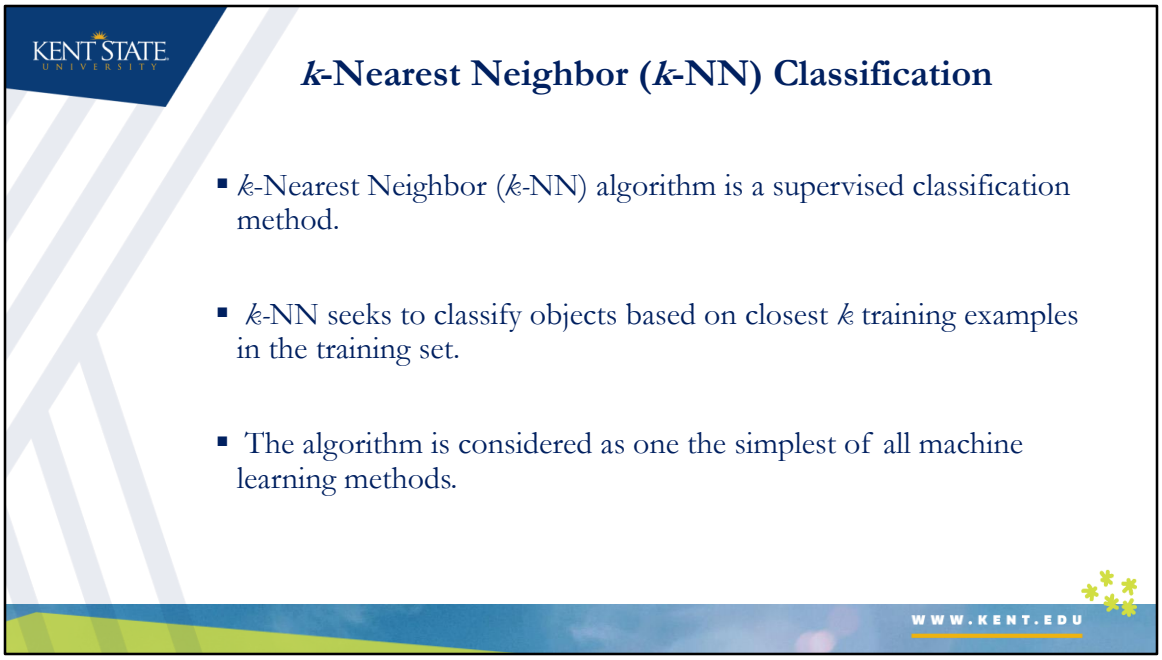
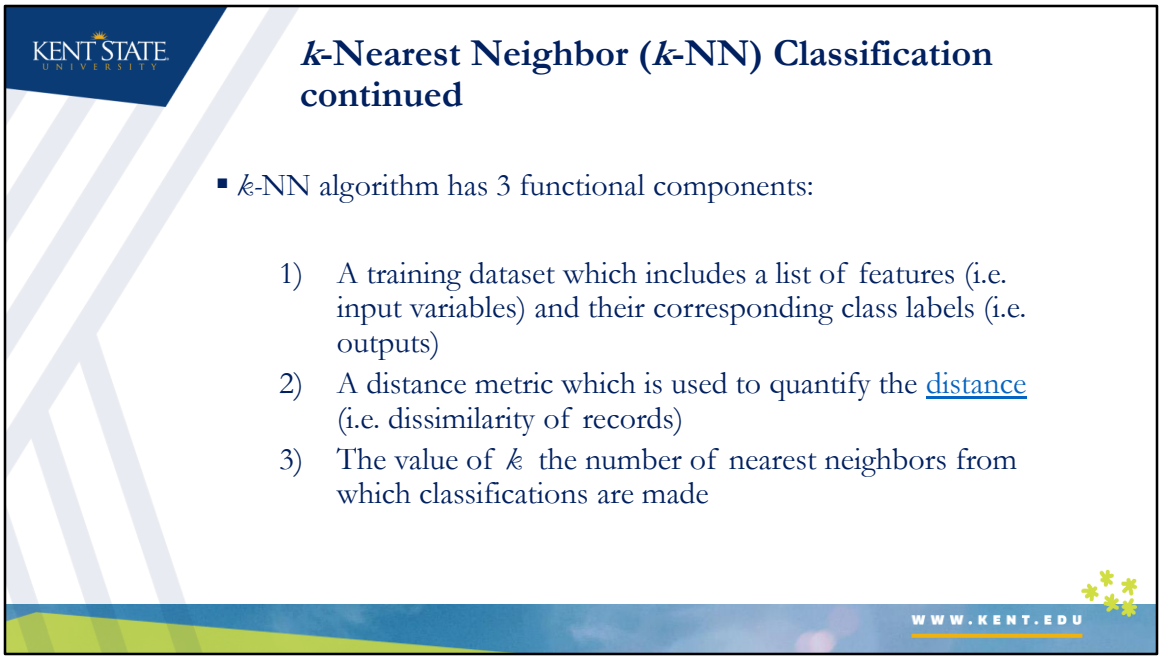
In this module, we discuss one of the simplest classifiers, k-NN.



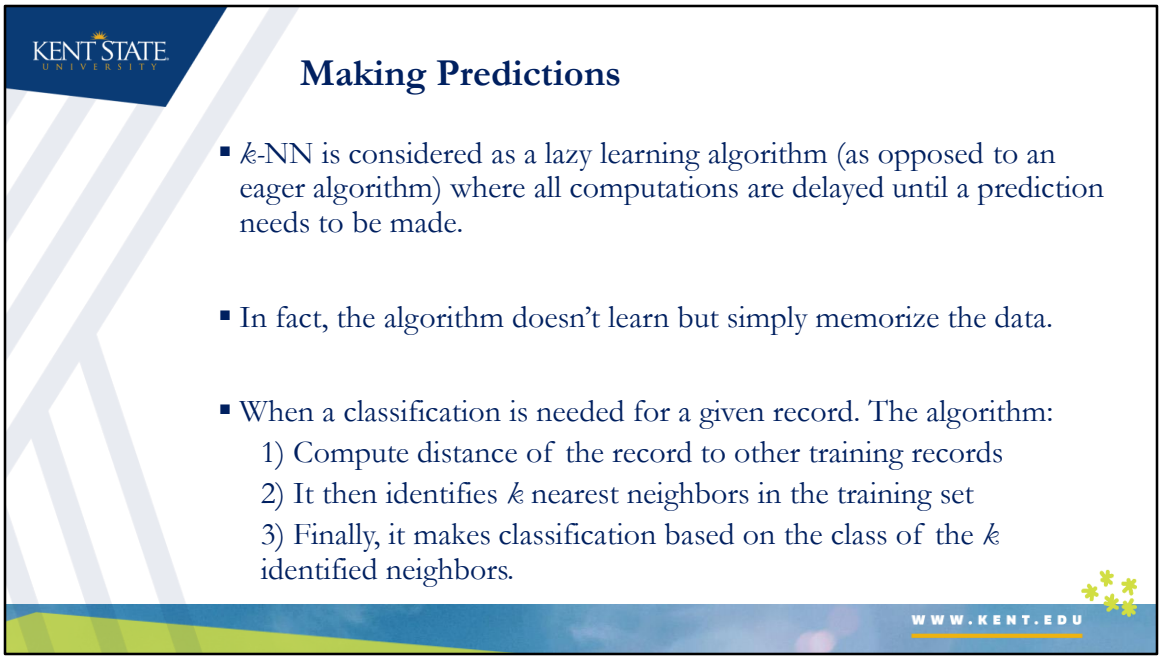
K-NN is a non-parametric classifier, i.e., it does not require the estimation of parameters. To classify a new record, the method relies on finding "similar" records in the training data. These neighbors are then used to derive a classification for the new record by voting. Here, we explain how similarity is calculated, how the numbers (k) is chosen, and how the classification is computed. K-NN can be used for both categorical variables, classification, and for continuous variables, prediction.



The training data contains the list of values for the predictor (input) variables, x1, x2, …, and the class membership Y (output variable). A central question in calculating similarity with existing records is the concept of distance. The most common measure for distance calculation is the Euclidean distance, which is nothing but the square root of the sum of squared differences between the value of the variable in the training set, and the value of the variable in the new data point which needs to be classified. The link in the slide provides more explanation. There are many other distance measures that can be used. One consideration in selecting distance measures is the realization that these measures must be calculated for each value in the training set across all input features. So, the computational cost of this is an important consideration, especially for a dataset with many features and records. The Euclidean distance calculation is computationally cheap, and thus a commonly used measure here.

After computing the distances between the record to be classified and existing records, we need a rule to assign a class to the record to be classified, based on the classes of its neighbors. The simplest case is k = 1, where we look for the record that is closest (the nearest neighbor) and classify the new record as belonging to the same class as its closest neighbor. It is a remarkable fact that this simple, intuitive idea of

using a single nearest neighbor to classify records can be very powerful when we have a large number of records in our training set. It turns out that the misclassification error of the 1-nearest neighbor scheme has a misclassification rate that is no more than twice the error when we know exactly the probability density functions for each class.

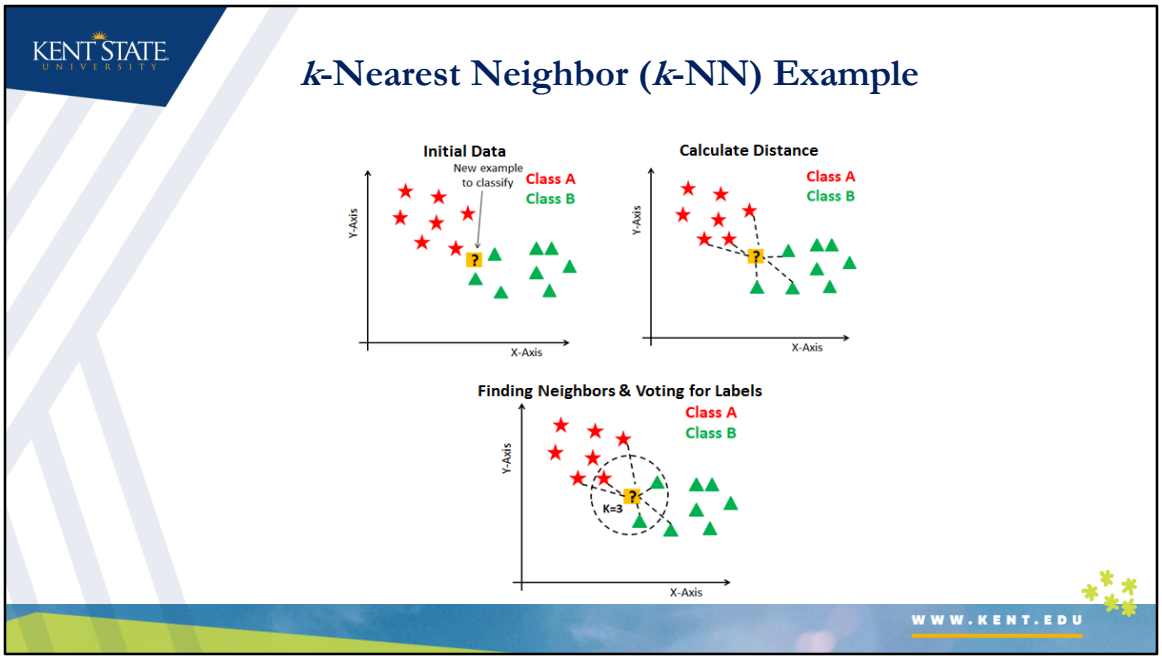


k-NN is a "lazy learner": the time-consuming computation is deferred to the time of prediction. For every record to be predicted, we compute its distances from the entire set of training records only at the time of prediction. This behavior prohibits using this algorithm for real-time prediction of a large number of records simultaneously.

The idea of the 1-nearest neighbor can be extended to k > 1 neighbors as follows:

1. Find the nearest k neighbors to the record to be classified.
2. Use a majority decision rule to classify the record, where the record is classified as a member of the majority class of the k neighbors.

For example, if the majority of neighbors belong to class A, then the observation is grouped into Class A. Let us see an example.



This example graphically illustrates the distance calculation and the assignment of class to the data point. Here, we use k= 3, i.e., we choose the classes of the three nearest neighbors to determine the class of our data point. For this example, the new data point will be classified at Class B on that basis.

The question is, how do we determine k, and what are the relative advantage or disadvantage of using different values of k?

The advantage of choosing k > 1 is that higher values of k provide smoothing that reduces the risk of overfitting due to noise in the training data. Generally speaking, if k is too low, we may be fitting to the noise in the data. However, if k is too high, we will miss out on the method's ability to capture the local structure in the data, one of its main advantages. In the extreme, k = n = the number of records in the training dataset. In that case, we simply assign all data to the majority class, which essentially is the naïve rule.

So how is k chosen? Answer: We choose the k with the best classification performance. Remember our discussion on training, validation and test sets. We choose k by its performance on the validation set.

k-NN: Prediction Using Majority Voting

- In this approach, all votes are equal. For each class, we count how many of the *k* neighbors have that class. We return the class with the majority of the votes.

- In case of a tie vote, different strategies can be followed:
  a) Randomly choose one of the classes (least accurate)
  b) Choose the class of the record that is closest to the observation (as if k was 1)
  c) Decrease k by 1 and decide the class

This rule was illustrated by the graphical example that we saw earlier. There is one further issue that needs to be considered when using the majority voting rule.

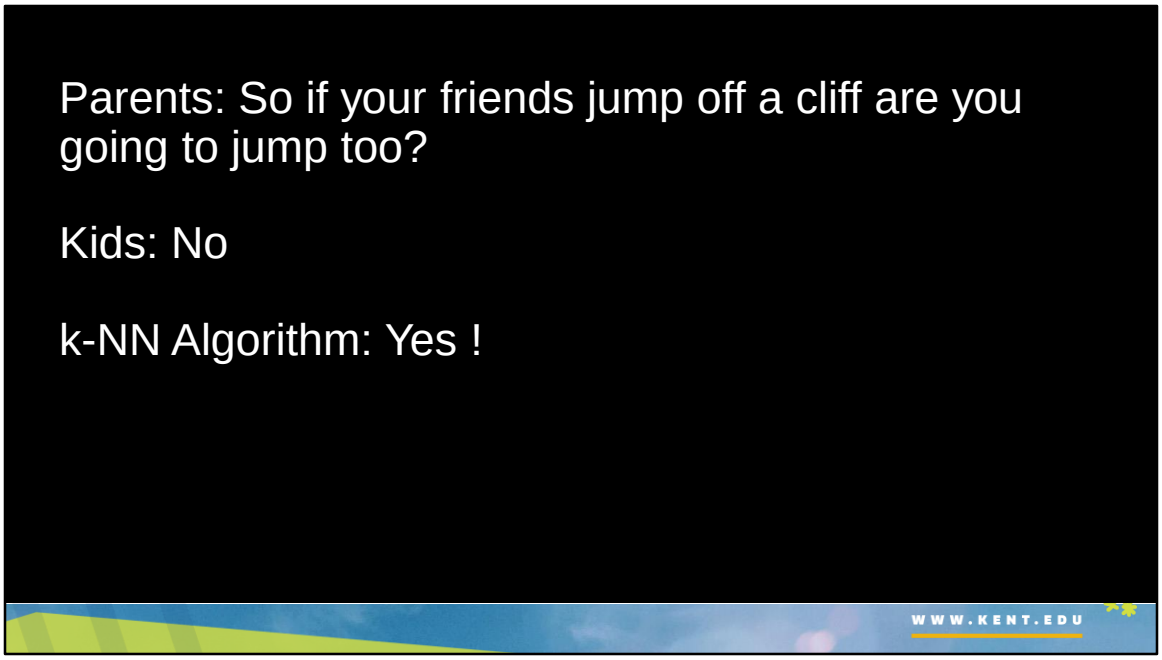
The definition of "majority" is directly linked to the notion of a cutoff value applied to the class membership probabilities. Specfically, using a simple majority rule is equivalent to setting the cutoff value to 0.5. As we saw earlier, the cutoff values affect the confusion matrix , i.e., the error rates. As such, sometimes, it might be advantageous to set cutoff values other than 0.5 to account for differences in the cost of misclassification.

KENT STATE UNIVERSITY

k-NN: Prediction Using Inverse Distance-Weighted Voting

- In this approach, closer neighbors get higher votes.
- While there are better-motivated methods, the simplest version is to take a neighbor's vote to be the inverse of its distance to the given observation.
- Then, we sum the votes and return the class with the highest vote.

This is a modification to the majority rule where we assign different weights depending on the distance of the neighbor from the point. Essentially, a weighted-average approach. For this approach, what would be the cutoff probability?



While our discussion in this module has primarily been on classification, k-NN can be extended to examples where the number of class m > 2, and to examples where the response is a numerical value. In that case, the distance measure is calculated similarly, but we would use a weighted average as the measure of distance between points.

This concludes our discussion on k-NN.