

- Products
- Solutions
- Buy
- Trials
- Support

Textbook-k-Nearest Neighbors

What can we help you find?

Search

## Looking for info about statistics?

We wrote the book on it.  
And you can read it for free!

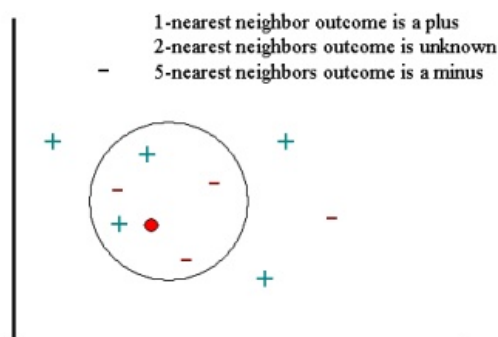
- Elementary Concepts
- Statistics Glossary
- Basic Statistics
- ANOVA / MANOVA
- Association Rules
- Boosting Trees
- Canonical Analysis
- CHAID Analysis
- C & R Trees
- Classification Trees
- Cluster Analysis
- Correspondence Analysis
- Data Mining Techniques
- Discriminant Analysis
- Distribution Fitting
- Experimental Design
- Factor Analysis
- General Discrim. Analysis
- General Linear Models
- Generalized Additive Mod.
- Generalized Linear Mod.
- General Regression Mod.
- Graphical Techniques
- Ind. Components Analysis
- Linear Regression
- Log-Linear Analysis
- MARSplines
- Machine Learning
- Multidimensional Scaling
- Neural Networks
- Nonlinear Estimation
- Nonparametric Statistics
- Partial Least Squares
- Power Analysis
- Process Analysis
- Quality Control Charts
- Reliability / Item Analysis
- SEPATH (Structural eq.)
- Survival Analysis
- Text Mining
- Time Series / Forecasting
- Variance Components
- Statistical Advisor

## k-Nearest Neighbors







- Classification
- Regression
- Technical Details
- Cross-Validation
- Distance Metric
- k-Nearest Neighbor Predictions
- Distance Weighting

### Classification

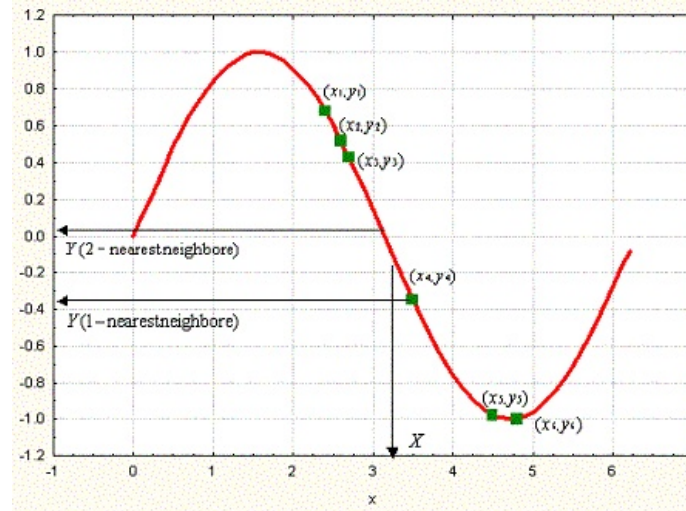
To demonstrate a  $k$ -nearest neighbor analysis, let's consider the task of classifying a new object (query point) among a number of known examples. This is shown in the figure below, which depicts the examples (instances) with the plus and minus signs and the query point with a red circle. Our task is to estimate (classify) the outcome of the query point based on a selected number of its nearest neighbors. In other words, we want to know whether the query point can be classified as a plus or a minus sign.



To proceed, let's consider the outcome of KNN based on 1-nearest neighbor. It is clear that in this case KNN will predict the outcome of the query point with a plus (since the closest point carries a plus sign). Now let's increase the number of nearest neighbors to 2, i.e., 2-nearest neighbors. This time KNN will not be able to classify the outcome of the query point since the second closest point is a minus, and so both the plus and the minus signs achieve the same score (i.e., win the same number of votes). For the next step, let's increase the number of nearest neighbors to 5 (5-

-  [Distribution Tables](#)
-  [References Cited](#)
-  [Send Comments](#)
-  [Business Solutions](#)
-  [Free Resources](#)
-  [About Textbook](#)

nearest neighbors). This will define a nearest neighbor region, which is indicated by the circle shown in the figure above. Since there are 2 and 3 plus and minus signs, respectively, in this circle KNN will assign a minus sign to the outcome of the query point.



## Regression

In this section we will generalize the concept of  $k$ -nearest neighbors to include regression problems. Regression problems are concerned with predicting the outcome of a dependent variable given a set of independent variables. To start with, we consider the schematic shown above, where a set of points (green squares) are drawn from the relationship between the independent variable  $x$  and the dependent variable  $y$  (red curve). Given the set of green objects (known as examples) we use the  $k$ -nearest neighbors method to predict the outcome of  $X$  (also known as query point) given the example set (green squares).

To begin with, let's consider the 1-nearest neighbor method as an example. In this case we search the example set (green squares) and locate the one closest to the query point  $X$ . For this particular case, this happens to be  $x_4$ . The outcome of  $x_4$  (i.e.,  $y_4$ ) is thus then taken to be the answer for the outcome of  $X$  (i.e.,  $Y$ ). Thus for 1-nearest neighbor we can write

$$Y = y_4$$

For the next step, let's consider the 2-nearest neighbor method. In this case, we locate the first two closest points to  $X$ , which happen to be  $y_3$  and  $y_4$ . Taking the average of their outcome, the solution for  $Y$  is then given by:

$$Y = \frac{y_3 + y_4}{2}$$

The above discussion can be extended to an arbitrary number of nearest neighbors  $K$ . To summarize, in a  $k$ -nearest neighbor method, the outcome  $Y$  of the query point  $X$  is taken to be the average of the outcomes of its  $k$ -nearest neighbors.

## Technical Details

*STATISTICA k-Nearest Neighbors (KNN)* is a memory-based model defined by a set of objects known as examples (also known as instances) for which the outcome are known (i.e., the examples are labeled). Each example consists of a data case having a set of independent values labeled by a set of dependent outcomes. The independent and dependent variables can be either continuous or categorical. For continuous dependent variables, the task is regression; otherwise it is a classification. Thus, *STATISTICA KNN* can handle both regression and classification tasks.

Given a new case of dependent values (query point), we would like to estimate the outcome based on the *KNN* examples. *STATISTICA KNN* achieves this by finding *K* examples that are closest in distance to the query point, hence, the name *k-Nearest Neighbors*. For regression problems, *KNN* predictions are based on averaging the outcomes of the *K* nearest neighbors; for classification problems, a majority of voting is used.

The choice of *K* is essential in building the *KNN* model. In fact, *k* can be regarded as one of the most important factors of the model that can strongly influence the quality of predictions. One appropriate way to look at the number of nearest neighbors *k* is to think of it as a smoothing parameter. For any given problem, a small value of *k* will lead to a large variance in predictions. Alternatively, setting *k* to a large value may lead to a large model bias. Thus, *k* should be set to a value large enough to minimize the probability of misclassification and small enough (with respect to the number of cases in the example sample) so that the *K* nearest points are close enough to the query point. Thus, like any smoothing parameter, there is an optimal value for *k* that achieves the right trade off between the bias and the variance of the model. *STATISTICA KNN* can provide an estimate of *K* using an algorithm known as cross-validation (Bishop, 1995).

## Cross-Validation

Cross-validation is a well established technique that can be used to obtain estimates of model parameters that are unknown. Here we discuss the applicability of this technique to estimating *k*.

The general idea of this method is to divide the data sample into a number of *v* folds (randomly drawn, disjointed sub-samples or segments). For a fixed value of *k*, we apply the *KNN* model to make predictions on the *v*th segment (i.e., use the *v-1* segments as the examples) and evaluate the error. The most common choice for this error for regression is sum-of-squared and for classification it is most conveniently defined as the accuracy (the percentage of correctly classified cases). This process is then successively applied to all possible choices of *v*. At the end of the *v* folds (cycles), the computed errors are averaged to yield a measure of the stability of the model (how well the model predicts query points). The above steps are then repeated for various *k* and the value achieving the lowest error (or the highest classification accuracy) is then selected as the optimal value for *k* (optimal in a cross-validation sense). Note that cross-validation is computationally expensive and you should be prepared to let the algorithm run for some time especially when the size of the examples sample is large. Alternatively, you can specify *k*. This may be a reasonable course of action should you have an idea of which value *k* may take (i.e., from previous *KNN* analyses you may have conducted on similar data) .

## Distance Metric

As mentioned before, given a query point, *KNN* makes predictions based on the outcome of the *K* neighbors closest to that point. Therefore, to make predictions with *KNN*, we need to define a metric for measuring the distance between the query point and cases from the examples sample. One of the most popular choices to measure this distance is known as Euclidean. Other measures include Euclidean squared, City-block, and Chebyshev:

$$D(x, p) = \begin{cases} \sqrt{(x - p)^2} & \text{Euclidean} \\ (x - p)^2 & \text{Euclidean squared} \\ \text{Abs}(x - p) & \text{Cityblock} \\ \text{Max}(|x - p|) & \text{Chebyshev} \end{cases}$$

where *x* and *p* are the query point and a case from the examples sample, respectively.

<

## k-Nearest Neighbor Predictions

After selecting the value of  $k$ , you can make predictions based on the  $KNN$  examples. For regression,  $KNN$  predictions is the average of the  $k$ -nearest neighbors outcome.

$$y = \frac{1}{K} \sum_{i=1}^K y_i$$

where  $y_i$  is the  $i$ th case of the examples sample and  $y$  is the prediction (outcome) of the query point. In contrast to regression, in classification problems,  $KNN$  predictions are based on a voting scheme in which the winner is used to label the query.

**Note.** For binary classification tasks, odd values of  $y = 1, 3, 5$  are used to avoid ties, i.e., two classes labels achieving the same score.

So far we have discussed  $KNN$  analysis without paying any attention to the relative distance of the  $K$  nearest examples to the query point. In other words, we let the  $K$  neighbors have equal influence on predictions irrespective of their relative distance from the query point. An alternative approach (Shepard 1968) is to use arbitrarily large values of  $K$  (if not the entire prototype sample) with more importance given to cases closest to the query point. This is achieved using so-called distance weighting.

## Distance Weighting

Since  $KNN$  predictions are based on the intuitive assumption that objects close in distance are potentially similar, it makes good sense to discriminate between the  $K$  nearest neighbors when making predictions, i.e., let the closest points among the  $K$  nearest neighbors have more say in affecting the outcome of the query point. This can be achieved by introducing a set of weights  $W$ , one for each nearest neighbor, defined by the relative closeness of each neighbor with respect to the query point. Thus:

$$W(x, p_i) = \frac{\exp(-D(x, p_i))}{\sum_{i=1}^K \exp(-D(x, p_i))}$$

where  $D(x, p_i)$  is the distance between the query point  $x$  and the  $i$ th case  $p_i$  of the example sample. It is clear that the weights defined in this manner above will satisfy:

$$\sum_{i=1}^K W(x_o, x_i) = 1$$

Thus, for regression problems, we have:

$$y = \sum_{i=1}^K W(x_o, x_i) y_i$$

For classification problems, the maximum of the above equation is taken for each class variables.

It is clear from the above discussion that when  $k > 1$ , one can naturally define the standard deviation for predictions in regression tasks using:

$$error\ bar = \mp \sqrt{\frac{1}{K-1} \sum_{i=1}^K (y - y_i)^2}$$