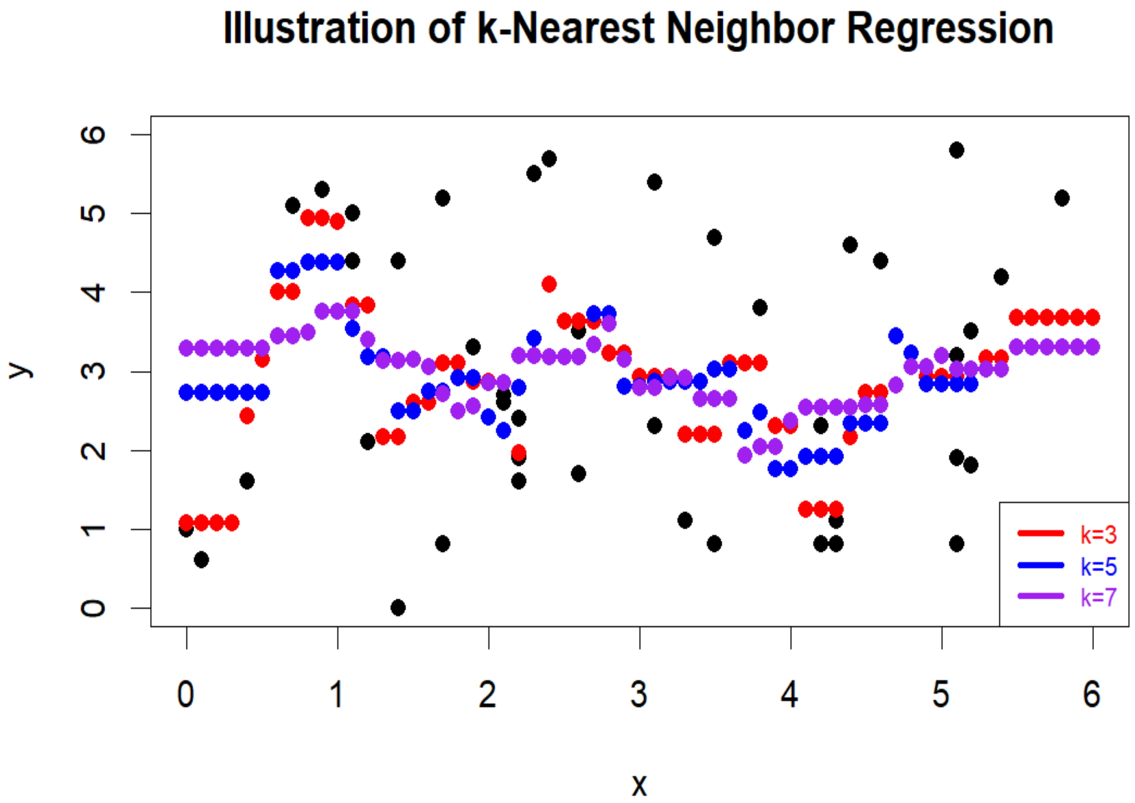
K-NEAREST NEIGHBOR REGRESSION AND CLASSIFICATION

For regression, the **k Nearest-neighbor (kNN) algorithm** works as follows: for any point (on a grid) in the space of predictor variables, we find k nearest neighbors using the regular Euclidean distance. The **Euclidean distance** between two d-dimensional vectors $\mathtt{v} = (v_1, \ldots, v_d)$ and $\mathtt{w} = (w_1, \ldots, w_d)$ is defined as $distance(\mathtt{v}, \mathtt{w}) = \sqrt{(v_1 - w_1)^2 + (v_2 - w_2)^2 + \cdots + (v_d - w_d)^2}$. The predicted value of y for this fixed point on a grid is determined as the average of the target values of the k nearest neighbors.

We illustrate how kNN regression works on a sample set of points with one predictor variable x and a target variable y. The black points in figure below represent the observed data on a scatterplot.

We consider each point on a grid between 0 and 6 with a step size of 0.1 and compute the mean values of y for $k = 3, 5,$ and 7 nearest neighbors. The predicted values are shown as colored dots in the figure below.

Note that as k increases, the predicted values approach to a horizontal line. Indeed if k is very large, all points are considered as nearest neighbors for every grid point, resulting in a single predicted value equal to the mean of all observed y-values.



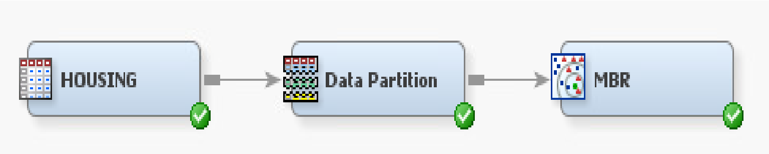
Historical Note: The kNN algorithm was first described in "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties", by Evelyn Fix and Joseph Hodges, report, UC Berkeley, 1951.

Example. We apply the kNN algorithm to build a regression for the data set in the file "housing_data.csv".

In SAS: We save the data file in the sasuser library using the following code.

```
proc import out=sasuser.housing datafile="./housing_data.csv" dbms=csv replace;
run;
```

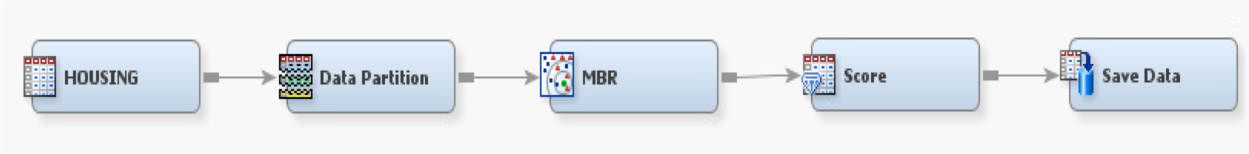
Then we use SAS Enterprise miner to fit a k nearest-neighbor regression (termed **Memory-Based Reasoning (MBR)**, using the path diagram:



For the "Data Partition" node, we specify to split the data into 80% training, and 20% testing sets. We run the paths and note the value for the Root MSE for the testing set (summarized in the table below).

Number of neighbors	Root MSE
7	77519.70
9	**77038.58**
10	76995.94

We can see that root MSE starts leveling out at $k = 9$, so we pick that number of neighbors and run the full path that includes scoring of the testing set. The diagram is given in the following figure:

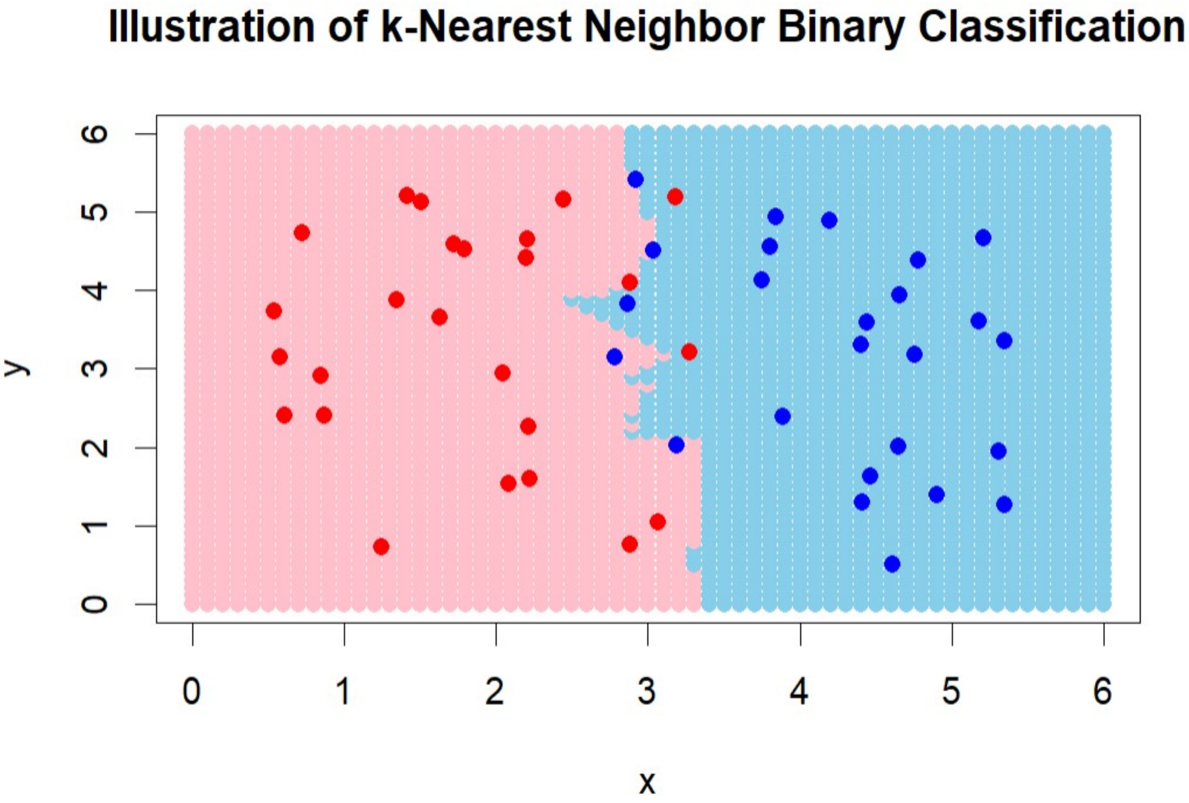


□

k-Nearest Neighbor Binary Classification

For binary classification, each point on a grid is assigned the class that is most frequent among its k nearest neighbors. Euclidean distance is used to measure the distances between grid points and data points.

We illustrate the algorithm using points of two colors: red and blue. The predictor variables are x and y. For each point on a two-dimensional grid with a step size of 0.1, the k nearest neighbors are identified, and the most frequent color among them is assigned to that grid point.



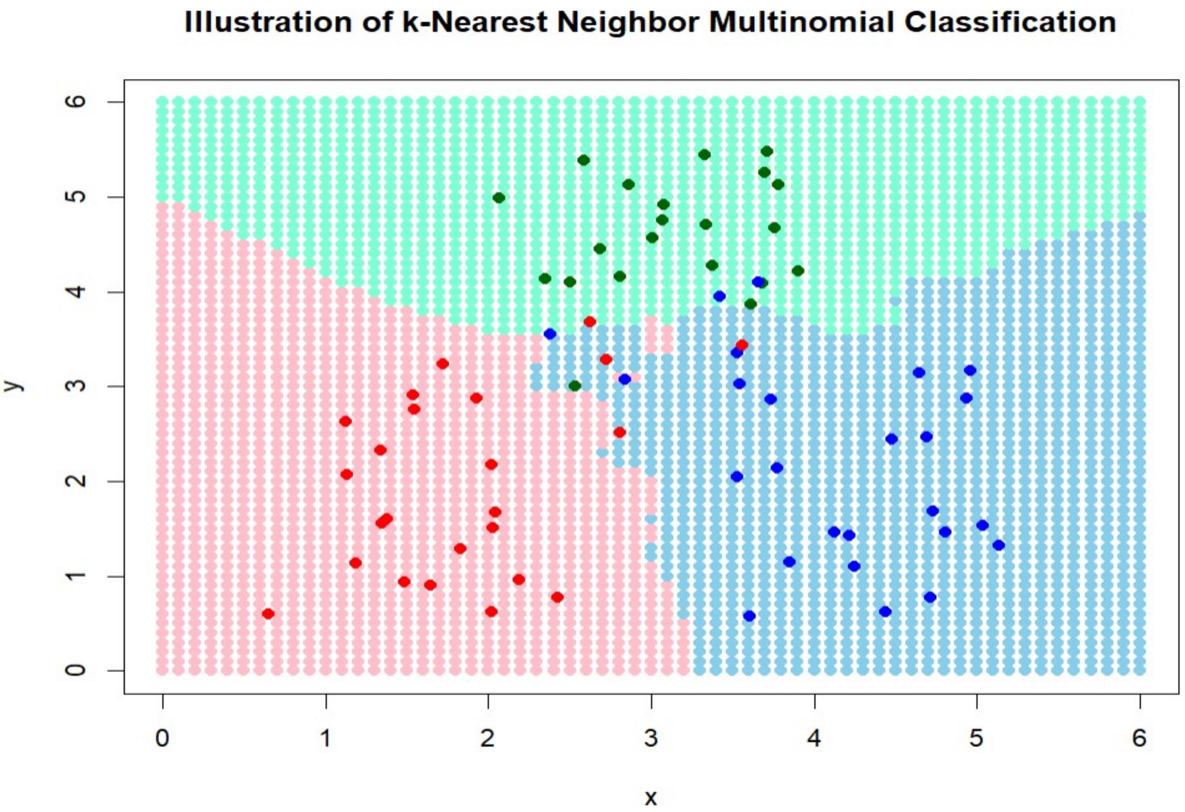
Example. For the data set "pneumonia_data.csv", we build the kNN binary classifier. In SAS Enterprise Miner, we partition the data into 80% training, and 20% testing sets, and run the MBR node, varying the number of neighbors and record the misclassification rate for the testing set. The results are summarized here:

Number of neighbors	Misclassification Rate
5	0.312
7	0.303
9	**0.269**
11	0.286

We choose to utilize $k = 9$ neighbors because it results in the smallest misclassification rate, and run the full path. □

k-Nearest Neighbor Multinomial Classification

The kNN algorithm works with multinomial classification in a manner similar to binary classification. The predicted class for a grid point is determined by a majority vote among the classes of its k nearest neighbors. We illustrate this below using three colors: red, blue, and green.



Example. Consider the data in the file "movie_data.csv". We fit a multinomial classifier using the kNN algorithm. In SAS Enterprise Miner, we run the path ending in the MBR node for $k = 7, 9,$

and 11, and compare the misclassification rates for the testing set.

Number of neighbors	**Misclassification Rate**
7	0.726
9	0.713
11	0.720

We use $k = 9$ as it gives the smallest misclassification rate and run the full path depicted in the figure below. □