Neural Network Classifier

Summary

The **Probabilistic Neural Network Classifier (PNN)** implements a nonparametric method for classifying observations into one of *g* groups based on *p* observed quantitative variables. Rather than making any assumption about the nature of the distribution of the variables within each group, it constructs a nonparametric estimate of each group's density function at a desired location based on neighboring observations from that group. The estimate is constructed using a Parzen window that weights observations from each group according to their distance from the specified location.

Observations are assigned to groups based on the product of three factors:

1. the estimated density function in the neighborhood of the point.
2. the prior probabilities of belonging to each group.
3. the costs of misclassifying cases that belong to a given group.

The sphere of influence of the Parzen weighting function may be specified by the user or optimized via jackknifing.

Sample StatFolio: *neuralclassifier.sgp*

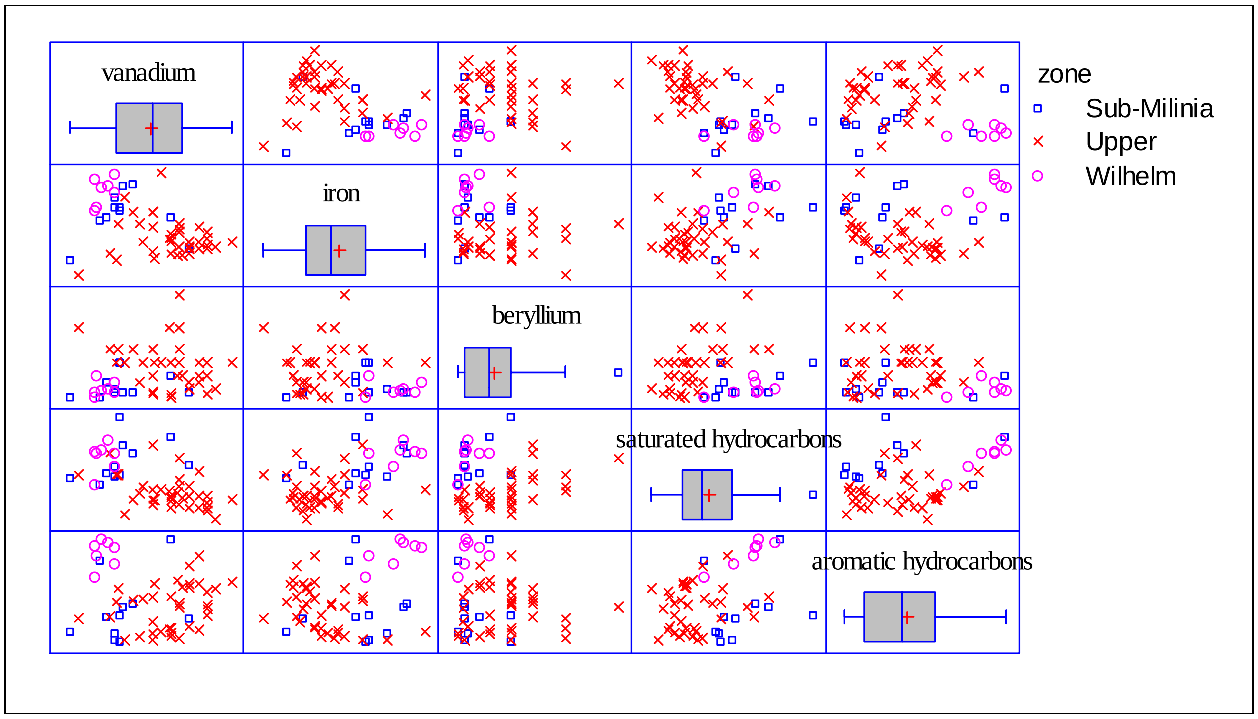
Sample Data:

The file *sandstone.sgd* contains a data set from Gerrild and Lantz (1969) that is described by Johnson and Wichern (2002). The data consist of a total of *n* = 56 samples of sandstone from three zones: *Wilhelm*, *Sub-Milinia*, and *Upper*. Each sample has been chemically analyzed and the values of five variables have been measured. The table below shows a partial list of the data in that file:

Sample	*Vanadium*	*Iron*	*Beryllium*	*Saturated hydrocarbons*	*Aromatic hydrocarbons*	*Zone*
1	3.9	51	0.20	7.06	12.19	Wilhelm
2	2.7	49	0.07	7.14	12.23	Wilhelm
3	2.8	36	0.30	7.00	11.30	Wilhelm
4	3.1	45	0.08	7.20	13.01	Wilhelm
5	3.5	46	0.10	7.81	12.63	Wilhelm
6	3.9	43	0.07	6.25	10.42	Wilhelm
7	2.7	35	0.00	5.11	9.00	Wilhelm
8	5.0	47	0.07	7.06	6.10	Sub-Milinia
9	3.4	32	0.20	5.82	4.69	Sub-Milinia
10	1.2	12	0.00	5.54	3.15	Sub-Milinia
…	…	…	…	…	…	

It is of interest to be able to classify samples into zones based on the 5 measurements.

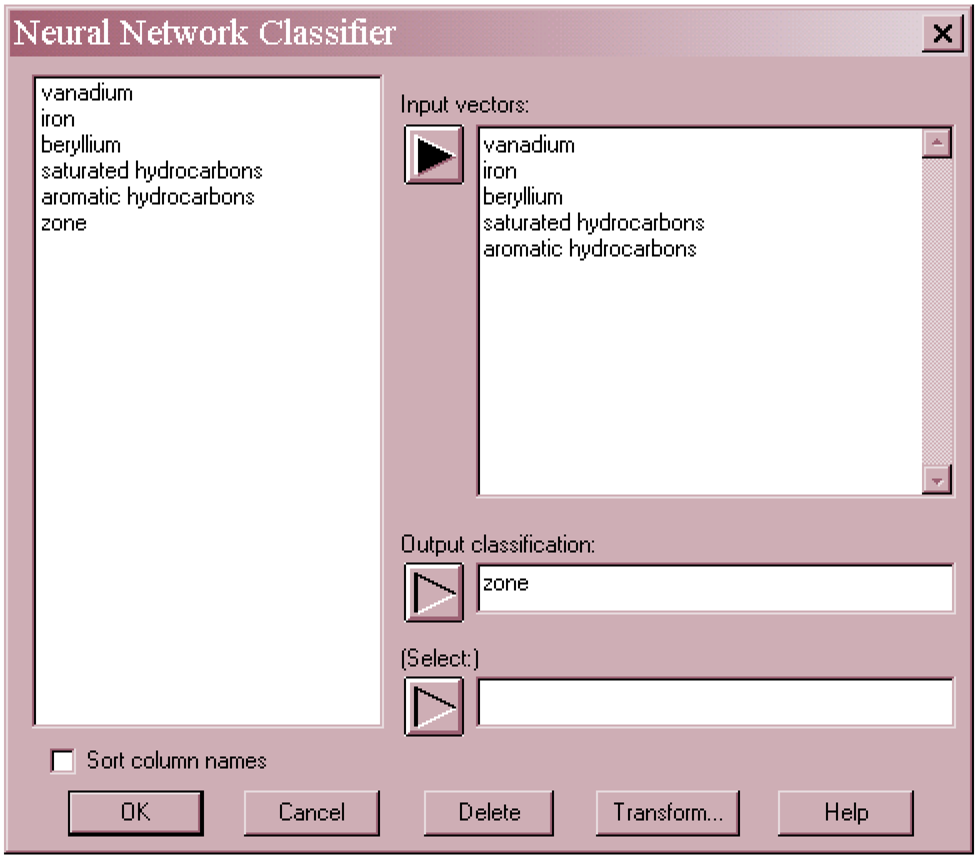
A matrix plot of the observed data is shown below:



There is quite a bit of clustering, but also quite a bit of overlap.

Data Input

The data input dialog box requests the names of *p* input variables to be used to classify cases and a single output variable that defines the known groups from the training sample:



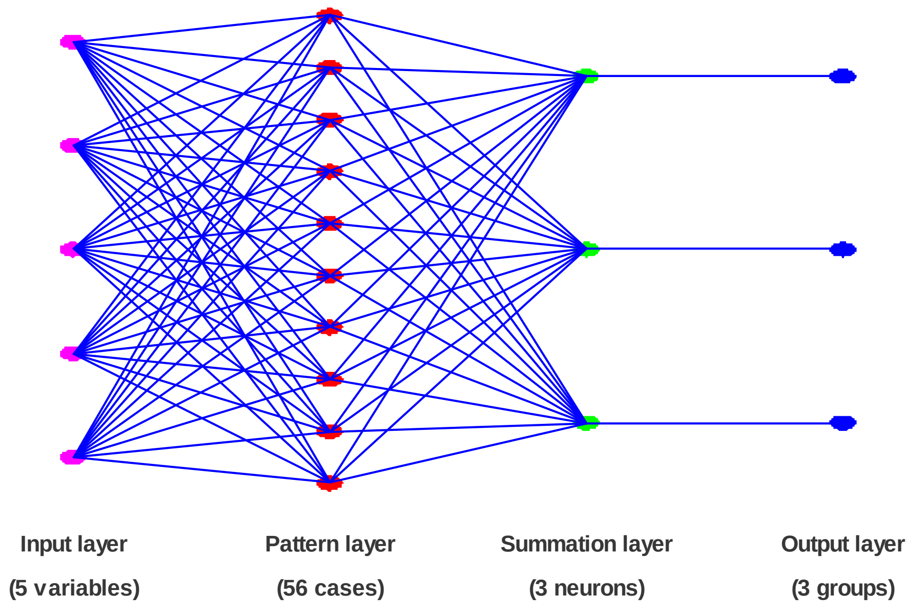
- **Input variables:** the names of *p* input variables, which should be quantitative factors characterizing the samples.

- **Classification Factor:** numeric or nonnumeric output variable containing an identifier of which group each observation belongs to. The number of unique values in this column will be represented by *g*.

- **Select:** subset selection.

Network Diagram

The approach to classifying cases can be formulated as a neural network. The *Neural Diagram* illustrates the basic setup of the network:



The network consists of four layers:

1. An *input layer* with *p* neurons, one for each of the input variables.
2. A *pattern layer* with *n* neurons, one for each case that will be used to train the network.
3. A *summation layer* with *g* neurons, one for each output class.
4. An *output layer*, also having one binary neuron for each output class that turns on or off depending on whether or not a case is assigned to the corresponding group.

Conceptually, the *input layer* provides the information from the *p* predictor variables by feeding their values (standardized by subtracting the mean and dividing by the standard deviation) to the neurons in the *pattern layer*. The pattern layer passes the values through an *activation function*, which uses the input values to estimate the probability density function for each group at a given location. The density estimates are then passed to the summation layer, which combines the information from the *n* training cases with prior probabilities and misclassification costs to derive a score for each group. The scores are then used to turn on the binary neuron in the output layer corresponding to the group with the largest score and turn off all other output neurons.

Input Layer

The neurons in the input layer represent the values of *p* input variables. These values, denoted by X_1 through X_p, are standardized by subtracting the sample mean of the *n* training cases and dividing by the sample standard deviation. The standardized values are then passed to the pattern layer.

Pattern Layer

The pattern layer takes each input variable X_i and calculates its contribution to the estimate of the probability density function for the group to which it belongs by passing it through an activation function. In this network, the activation function quantifies the contribution of the *i-th* value in the training case to the estimate of the density function for group *j* and is given by

$$g_{ij} = W\left(\frac{X - X_i}{\sigma}\right) \text{ if observation } i \text{ belongs to group } j \tag{1}$$

and

$$g_{ij} = 0 \text{ otherwise.} \tag{2}$$

σ is a scale parameter that controls how quickly the influence of a point decays as a function of its distance from *X*. Because of its shape, the function W is often taken to be the Gaussian function defined by

$$\exp\left(-\frac{\|X - X_i\|^2}{\sigma^2}\right) \tag{3}$$

where $\|X - X_i\|^2$ is the squared Euclidian distance between *X* and X_i.

Summation Layer

The neurons in the summation layer combine the information from all of the members of the training set. Letting n_j represent the number of observations in the training set that belong to group *j*, the estimated density function for group *j* at location X is proportional to

$$g_j(X) = \frac{1}{n_j}\sum_{i=1}^{n} g_{ij} \tag{4}$$

To determine which group the observation at location X should be assigned to, two other quantities are needed:

1. The prior probability h_j that an observation belongs to group *j*, without consideration of the input variables. This would normally represent the relative proportion of all samples in the population that belong to group *j*. Using *Analysis Options*, the prior probabilities may be assumed to be equal, be represented by the fraction of the training set that comes from each group, or be entered by the user.

2. The cost c_j of incorrectly classifying an observation that belongs to group *j*. In some cases, as when screening for the presence of a disease, incorrectly classifying an individual that belongs to one group (has the disease) may be more serious that misclassifying an individual who belongs to the other group (does not have the disease).

The output layer assigns a score to each group by multiplying the estimated density function by the prior probability and the cost:

$$Score_j = h_j c_j g_j(X) \tag{5}$$

These scores are then passed to the output layer.

<u>Output Layer</u>
The neurons in the output layer are binary and can only be turned on or off. Based on the scores, output neuron *j* is turned on if

$$Score_j > Score_k \tag{6}$$

for all *k* not equal to *j*. Otherwise, it is turned off. In the case of a tie, the neuron that is turned on is determined randomly.

Training the Network

The one parameter in the formulation of the network that can be varied is the scaling parameter σ, which affects how fast the influence of an observation on the density at point X decays as its distance from *X* increases. The procedure provides 3 options for determining σ:

1. σ may be specified by the user. The default value is σ = 1, which is not unreasonable since the input variables have been standardized.
2. σ may be ignored and an observation at point *X* always matched to the group corresponding to its nearest neighbor.
3. Different values of σ may be tried and the value picked which maximizes the percentage of the *n* observations in the training set that are correctly classified.

When the third case is selected, the network is trained using a procedure called *jackknifing*. Jackknifing removes one point at a time from the training set and determines how often it is correctly classified when it is *not* used to estimate the group scores. The value of σ correctly classifies the highest percentage of the removed points becomes the selected value.

Analysis Summary

The *Analysis Summary* summarizes the performance of the algorithm on the training set:

Neural Networks - zone
Classification factor: zone
Input factors:
  vanadium (percent ash)
  iron (percent ash)
  beryllium (percent ash)
  saturated hydrocarbons (percent area)
  aromatic hydrocarbons (percent area)

Number of cases in training set: 56
Number of cases in validation set: 0

Spacing parameter used: nearest neighbor

Training Set

		Percent Correctly
zone	*Members*	*Classified*
Sub-Milinia	11	63.6364
Upper	38	100.0
Wilhelm	7	85.7143
Total	56	91.0714

Validation Set

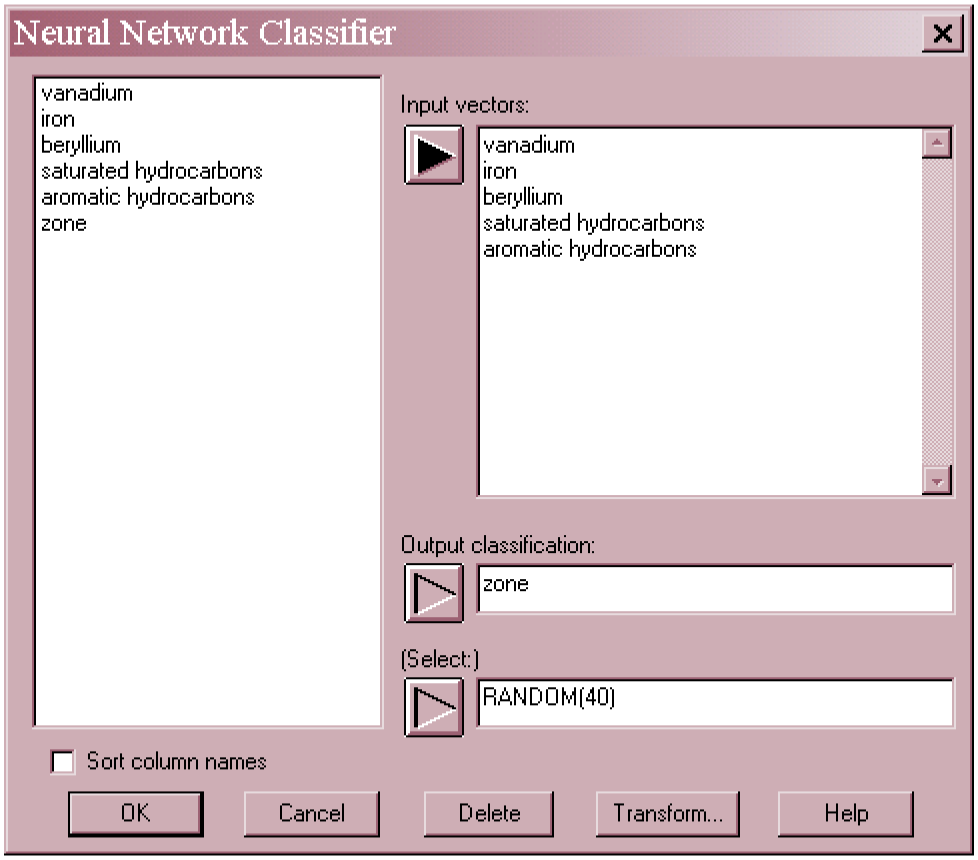
	Number	*Percent Correctly*
zone	*Classified As*	*Classified*
Sub-Milinia	0	
Upper	0	
Wilhelm	0	
Total	0	

Included in the table are:

- **Input variables**: identification of the input variables.
- **Number of cases in training set**: the number of observations *n* in the training set.
- **Number of cases in validation set**: the number of cases withheld from the training set. Cases can be withheld using the *Select* field on the data input dialog box.
- **Spacing parameter used**: method for determining probability density function. If σ is specified by the user or estimated by jackknifing, its value will be displayed. If each point is matched to its *nearest neighbor*, that will be indicated.
- **Training Set**: the number and percentage of observations in the training set that were correctly classified.
- **Classification Set**: the number and percentage of observations withheld from the training set that were correctly classified.

Foe example, the above table shows that the nearest neighbor rule correctly classifies slightly more than 91% of the observations. This would serve as a benchmark against which to compare other methods. It is also interesting to note that the nearest neighbor rule works well on the largest group, but not nearly so well on the smaller groups. This is not unexpected, since the larger group provides a better chance of having a nearby neighbor. The *Sub-Milinia* group is particularly hard to classify, since they tend to be more dispersed than the other two groups.

Example -Withholding a random set of observations
As mentioned above, the *Select* field on the data input dialog box can be used to withhold observations from the training set. For example, one could randomly withhold 16 of the 56 observations using the RANDOM function, as shown below:


Neural Network Classifier

vanadium
iron
beryllium
saturated hydrocarbons
aromatic hydrocarbons
zone

Input vectors:
vanadium
iron
beryllium
saturated hydrocarbons
aromatic hydrocarbons

Output classification: zone

(Select:) RANDOM(40)

The randomly selected 40 observations will be used as the *training set*, while the remaining 16 will form the *validation set.*

Neural Networks - zone (random(40))

Classification factor: zone
Input factors:
  vanadium (percent ash)
  iron (percent ash)
  beryllium (percent ash)
  saturated hydrocarbons (percent area)
  aromatic hydrocarbons (percent area)
Selection variable: random(40)

Number of cases in training set: 39
Number of cases in validation set: 17

Spacing parameter used: nearest neighbor

Training Set

		Percent Correctly
zone	*Members*	*Classified*
Sub-Milinia	9	77.7778
Upper	25	96.0
Wilhelm	5	80.0
Total	39	89.7436

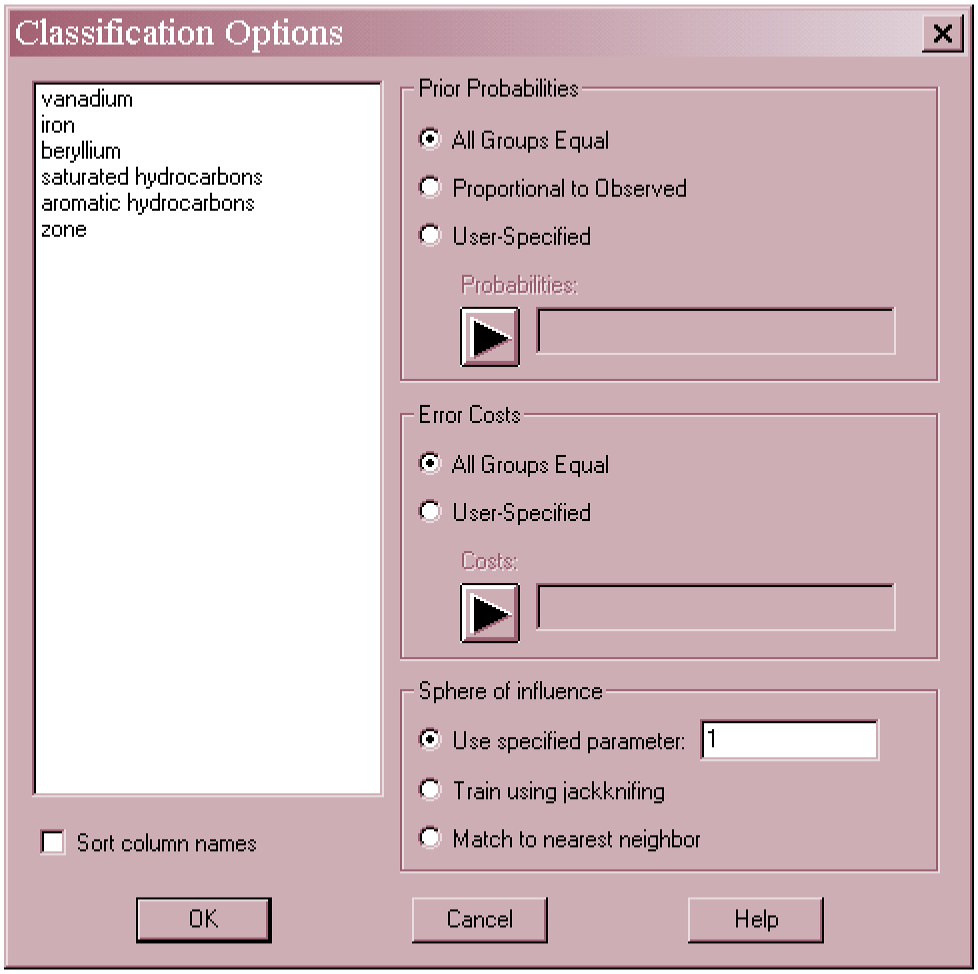
Validation Set

		Percent Correctly
zone	*Members*	*Classified*
Sub-Milinia	2	0.0
Upper	13	100.0
Wilhelm	2	100.0
Total	17	88.2353

Admittedly, the validation set is extremely small, but it confirms that the *Sub-Milinia* samples are hard to classify accurately.

Analysis Options

The *Analysis Options* dialog box allows the user to control the classification algorithm:



Classification Options

vanadium
iron
beryllium
saturated hydrocarbons
aromatic hydrocarbons
zone

☐ Sort column names

Prior Probabilities: [✓ All Groups Equal] Proportional to Observed User-Specified
Probabilities:

Error Costs: [✓ All Groups Equal] User-Specified
Costs:

Sphere of influence: [✓ Use specified parameter: 1] Train using jackknifing Match to nearest neighbor

- **Prior Probabilities:** method for determining the probability of group membership before the data is examined. Select *All Groups Equal* to assume equal priors for all groups, *Proportional to Observed* to set the priors equal to the fraction of the training set represented by each group, or *User-Specified* to enter a column with *g* values that sum to 1.

- **Error Costs:** relative costs for misclassifying a member of each group. *All Groups Equal* assigns equal costs to all groups. If *User-Specified*, enter a column containing *g* positive values.

- **Sphere of Influence**: method for estimating the density function. *Use specified parameter* specifies the desired value of σ. *Train using jackknifing* withholds values from the training set one at a time and determines σ based upon the percentage of time that the withheld point is correctly classified. *Match to nearest neighbor* ignores all prior probabilities and costs and matches each point to the group corresponding to its closest neighbor in the space of the X variables.

Example – Optimizing sigma using jackknifing

The following output shows the result of optimizing σ using the jackknife method:

Neural Networks - zone

Classification factor: zone
Input factors:
    vanadium (percent ash)
    iron (percent ash)
    beryllium (percent ash)
    saturated hydrocarbons (percent area)
    aromatic hydrocarbons (percent area)
Prior probabilities: proportional to occurrence in training set
Error costs: equal for all classes

Number of cases in training set: 56
Number of cases in validation set: 0

Spacing parameter used: 0.0 (optimized by jackknifing during training)

Training Set

		Percent Correctly
zone	*Members*	*Classified*
Sub-Milinia	11	63.6364
Upper	38	100.0
Wilhelm	7	85.7143
Total	56	91.0714

Despite trying a large number of different values of σ, the best value was 0, which corresponds to the nearest neighbor method. By fixing the value of σ, it will be noted that even a small positive value of σ reduces the percentage of cases correctly classified:

Neural Networks - zone

Classification factor: zone
Input factors:
    vanadium (percent ash)
    iron (percent ash)
    beryllium (percent ash)
    saturated hydrocarbons (percent area)
    aromatic hydrocarbons (percent area)
Prior probabilities: proportional to occurrence in training set
Error costs: equal for all classes

Number of cases in training set: 56
Number of cases in validation set: 0

Spacing parameter used: 0.25 (specified by user)

Training Set

		Percent Correctly
zone	*Members*	*Classified*
Sub-Milinia	11	72.7273
Upper	38	89.4737
Wilhelm	7	85.7143
Total	56	85.7143

Classification Table

The *Classification Table* shows the result of using the derived classification rule to assign both observed cases and new cases to groups. For a given set of *X* values, a case is assigned to whichever group gives the largest score, where the score is based upon the estimated probability density function, the prior probability, and the error cost. Since the size of the populations from which each group's samples are taken may not be the same, the probability that an individual belongs to a particular group prior to examining the data may vary from group to group. For example, in screening for a disease, the proportion of individuals given a diagnostic test who actually have the disease may be very small, a fact that needs to be accounted for. Using *Pane Options*, the user specifies how to handle the prior probabilities. They can be assumed to be the same for all groups, to be proportional to the fraction of the data within each group, or input by the user.

The table below shows typical output:

Classification Table

Actual	*Group*	*Predicted*		
zone	*Size*	*Sub-Milinia*	*Upper*	*Wilhelm*
Sub-Milinia	11	7	2	2
		(63.64%)	(18.18%)	(18.18%)
Upper	38	0	38	0
		(0.00%)	(100.00%)	(0.00%)
Wilhelm	7	1	0	6
		(14.29%)	(0.00%)	(85.71%)

Percent of training cases correctly classified: 91.07%

	Prior	*Error*
zone	*Probability*	*Cost*
Sub-Milinia	0.3333	1.0
Upper	0.3333	1.0
Wilhelm	0.3333	1.0

	Actual	*Nearest*	*Nearest*	*2nd Nearest*	*2nd Nearest*
Row	*Group*	*Neighbor*	*Distance*	*Neighbor*	*Distance*
7	Wilhelm	Sub-Milinia*	0.161634	Wilhelm	0.2889
11	Sub-Milinia	Upper*	0.227325	Sub-Milinia	0.334108
12	Sub-Milinia	Upper*	0.265937	Sub-Milinia	0.315371
16	Sub-Milinia	Wilhelm*	0.294938	Upper	0.298606
18	Sub-Milinia	Wilhelm*	0.135704	Sub-Milinia	0.397795
57		Sub-Milinia	0.202902	Upper	0.22956

* = incorrectly classified.

The top section of the table shows how well the classification rule performed in classifying the training data. Each row tabulates the results for cases that actually belong to a particular group. The columns show how often they were classified as belonging to each group. Displayed along the bottom is the percentage of cases correctly classified.

The center part of the table displays the prior probabilities. For the example data, the prior probabilities were assumed to be the same for all groups.

The lower part of the table shows the two groups that received the highest scores for selected cases. The table shows:

1. *Highest and second highest group* (or *Nearest and second nearest neighbors*) – the two groups with the highest scores or smallest distances.

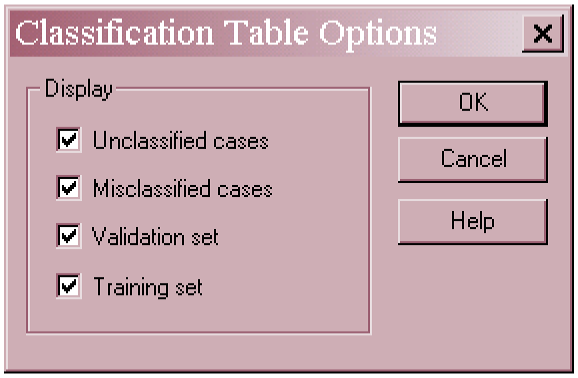
2. *Highest and second highest score* (or *Nearest and second nearest distance*) – the scores or closest distance of the two groups.

Depending upon *Pane Options*, the table may include all rows in the datasheet or only selected rows. Predictions for observations for which the group membership is not known can also be included by adding additional information to the datasheet for the *X* variables but leaving the cell for the group indicator blank. For example, suppose a new sample was taken with the following measurements:

vanadium = 5.7 percent ash
iron = 32 percent ash
beryllium = 0.5 percent ash
saturated hydrocarbons = 4.99 percent area
aromatic hydrocarbons = 3.62 percent area

These values would be placed in row #57 of the datasheet. The table shows that the group with the highest score for those values is *Sub-Milinia*, followed by *Upper*.

Pane Options

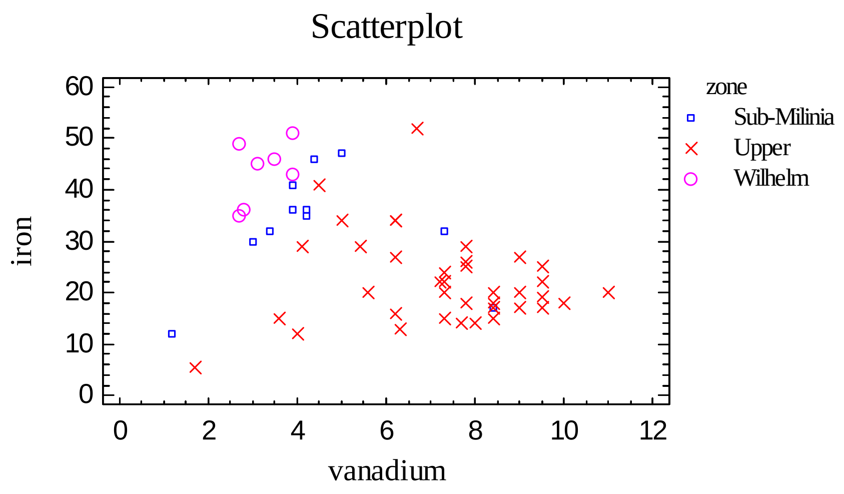


☑ Unclassified cases
☑ Misclassified cases
☑ Validation set
☑ Training set

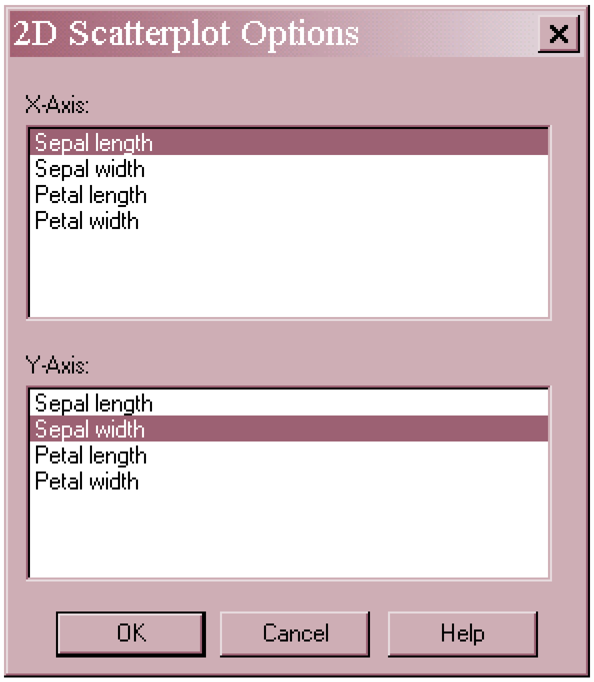
- **Display**: *Unclassified Cases* will include all observations in the datasheet with information for the X variables but no indication of group membership. *Misclassified Cases* will include any cases that were classified incorrectly. *Validation set* will include all cases in the validation set. *Training set* will include all cases in the training set.

2D Scatterplot

The *2D Scatterplot* plots the data for any two of the *X* variables.



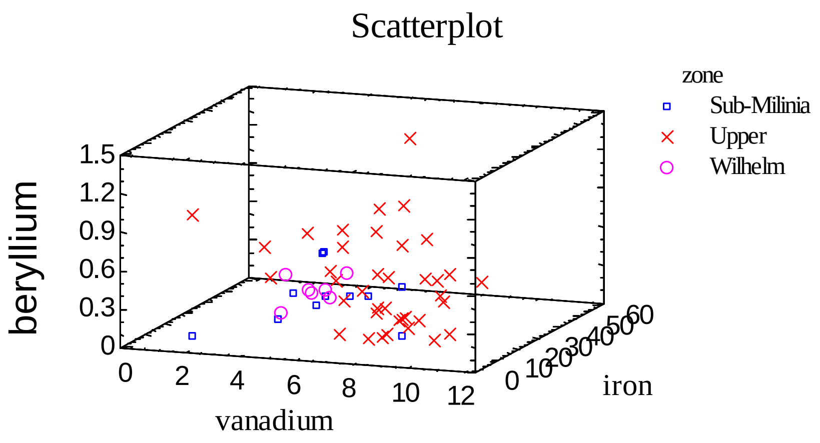
Pane Options



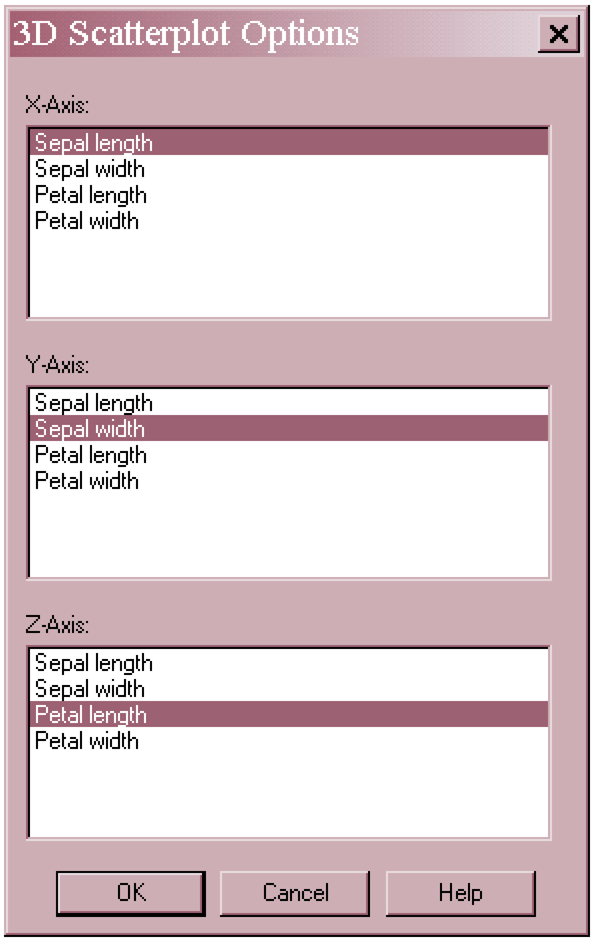
Select variables to define the horizontal and vertical axes.

3D Scatterplot

The 3*D Scatterplot* plots the data for any three of the *X* variables.



Pane Options



3D Scatterplot Options

X-Axis:
- Sepal length
- Sepal width
- Petal length
- Petal width

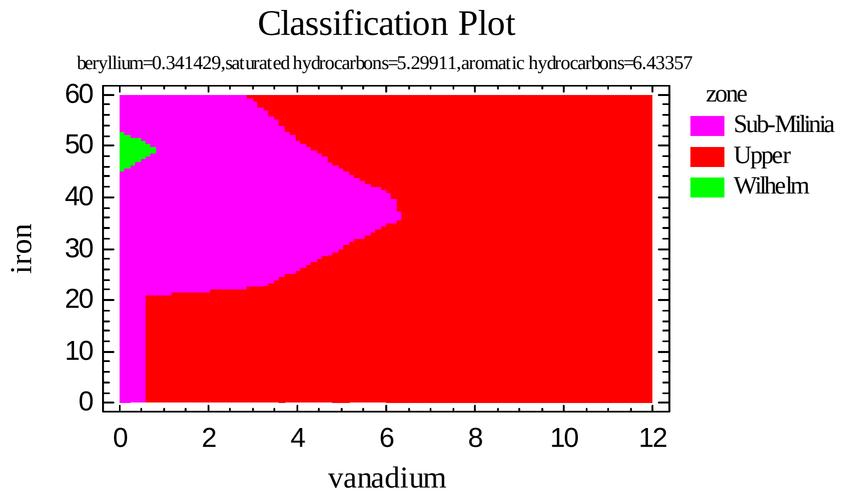
Y-Axis:
- Sepal length
- Sepal width
- Petal length
- Petal width

Z-Axis:
- Sepal length
- Sepal width
- Petal length
- Petal width

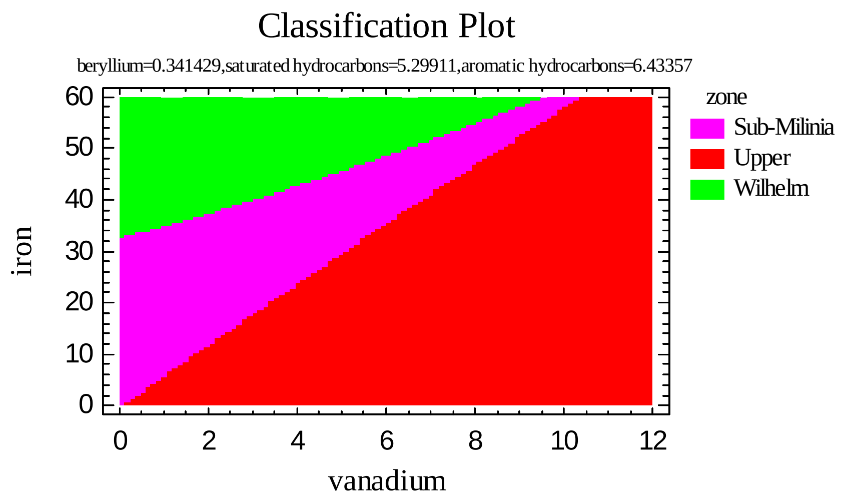
Select variables to define the three axes.

Classification Plot

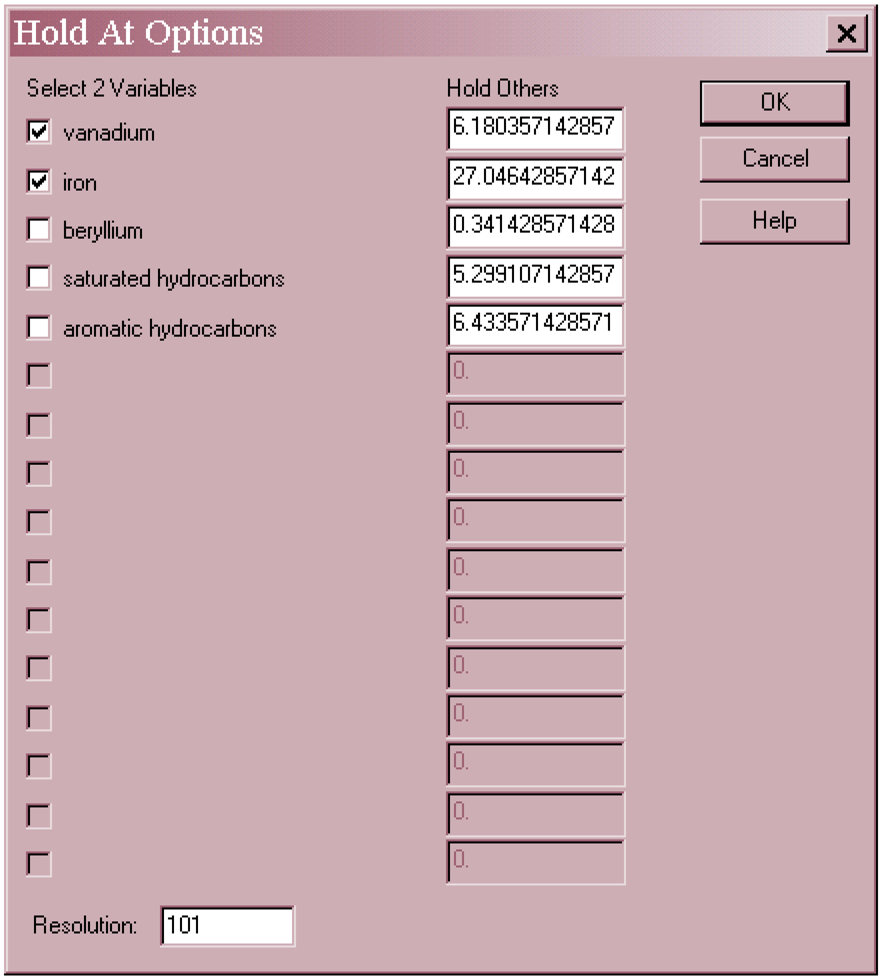
The *Classification Plot* can be used to gain a better understanding of how the region defined by the *X* variables is divided into areas that would result in samples being classified as belonging to different groups.



Each color-coded region corresponds to a different group. Two of the *X* variables are used to define the horizontal and vertical axes, while the other variables are held at fixed values. Notice the ragged nature of the above plot, which corresponds to the use of the nearest neighbor method. The plot below, created using σ = 1, is much smoother.



Pane Options



Hold At Options

Select 2 Variables	Hold Others
[✓] vanadium	6.180357142857
[✓] iron	27.04642857142
[ ] beryllium	0.341428571428
[ ] saturated hydrocarbons	5.299107142857
[ ] aromatic hydrocarbons	6.433571428571
[ ]	0.
[ ]	0.
[ ]	0.
[ ]	0.
[ ]	0.
[ ]	0.
[ ]	0.
[ ]	0.
[ ]	0.
[ ]	0.
[ ]	0.

Resolution: 101

- **Select 2 Variables**: the variables to plot on the horizontal and vertical axes.

- **Hold Others**: values to fix the non-selected variables at.

- **Resolution**: number of locations along the horizontal and vertical axes at which to evaluate the classification algorithm. Larger resolutions will result in a smoother plot but will increase the time required to generate it.