Supervised Machine Learning Wizard

statgraphics centurion

Revised: 1/31/2024

Summary

The Supervised Machine Learning Wizard assists users in applying various machine learning procedures contained in the Python Scikit-Learn library. It creates models of 2 forms:

1. *Classification models* that divide cases into groups based on their observed features.

2. *Regression models* that predict the value of an output variable.

It implements the procedures using a 5-step process:

Step 1: selects the output variable and features that will be considered as predictors.

Step 2: divides the cases into training, test and prediction sets.

Step 3: sets the values of any wizard options.

Step 4: applies one or more of 10 methods for constructing predictive models.

Step 5: uses the models to make predictions for cases in which the output value is unknown.

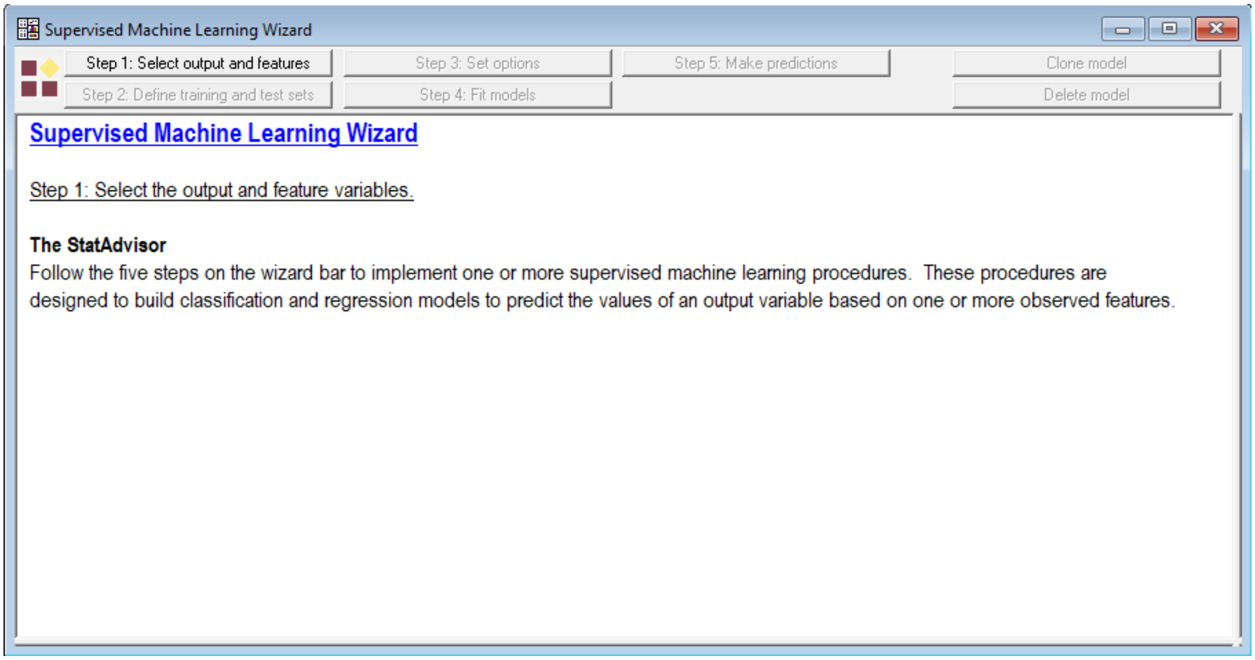
The calculations performed when fitting supervised machine learning models are performed by the Scikit-learn library in Python. To run the procedure, Python must be installed on your computer together with Scikit-learn. For information on downloading and installing Python, refer to the document titled "Python – Installation and Configuration".

Sample StatFolios: *wizardclassifier.sgp* and *wizardregressor.sgp*

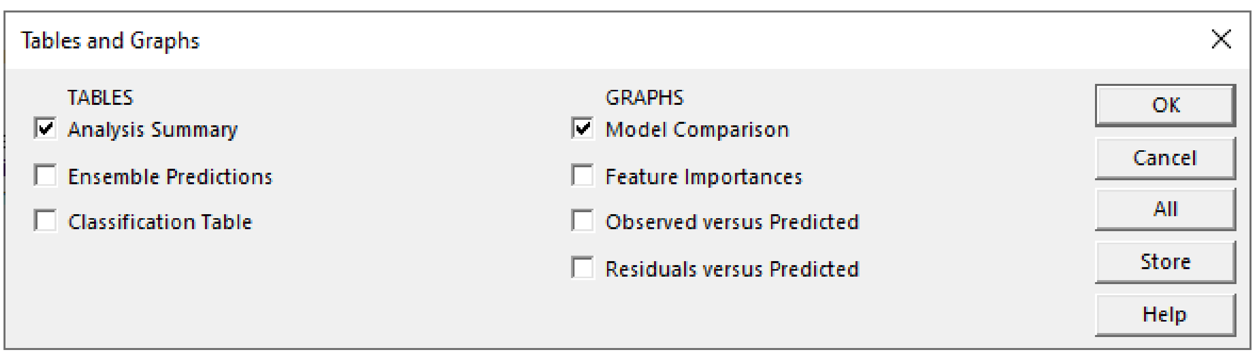
Sample Data Files: *breast cancer.sgd* and *boston house prices.sgd*

Supervised Machine Learning Wizard

When the supervised machine learning wizard is selected from the main Statgraphics menu, the following window is created:



The *Analysis Summary* pane tracks the progress of defining data, building models, and making predictions. Once models are built, the *Tables and Graphs* option may be used to create a plot comparing the performance of different models.



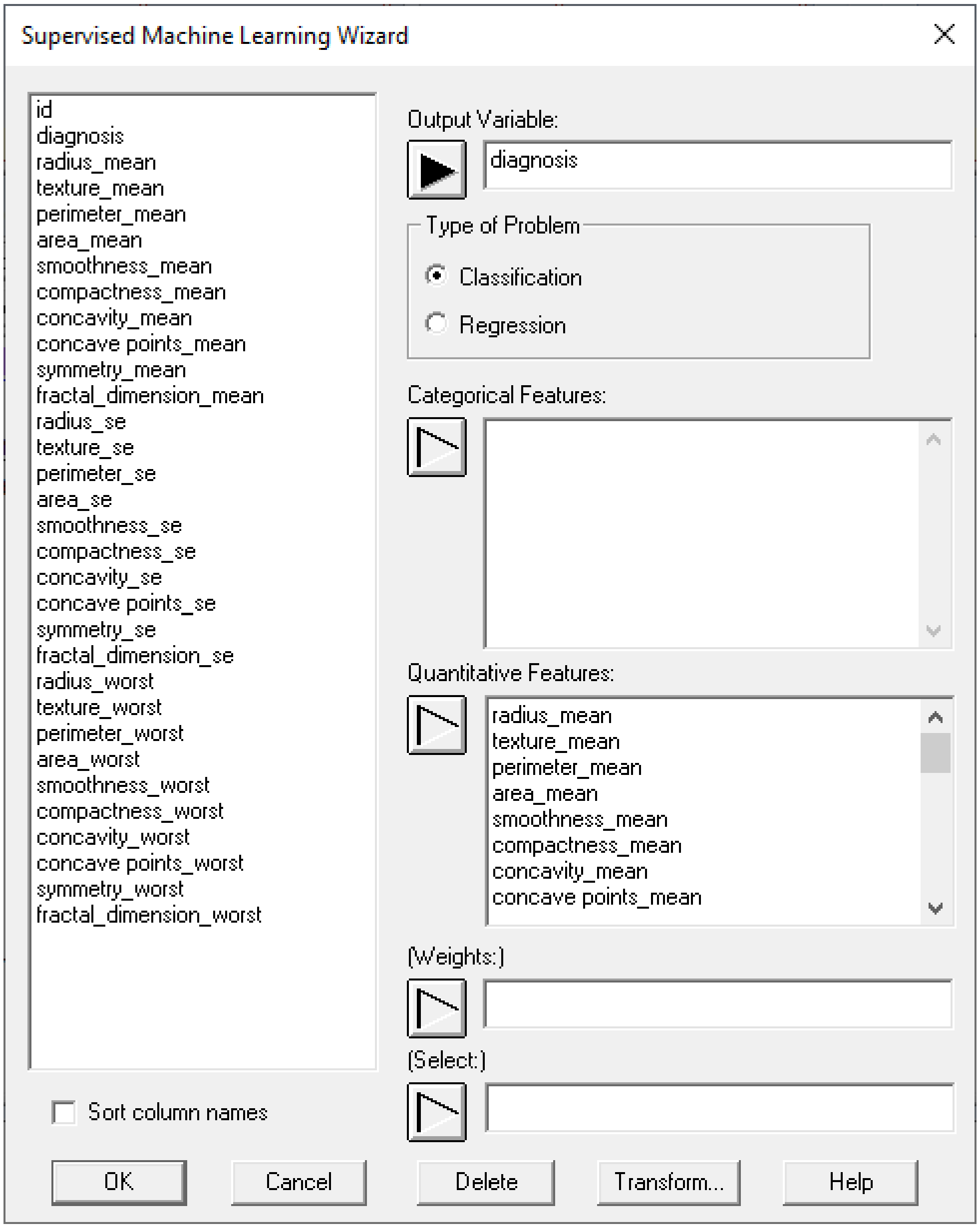
Step 1: Select Output and Features

The first step in using the wizard is loading the data to be modeled and selecting the output and feature variables. As an example, consider the dataset contained in the file named *breast cancer.sgd*. This data was obtained at the University of Wisconsin and provides information about the cell nuclei of 569 patients who developed masses in their breasts. The table below shows a partial list of the data in that file:

sample	*id*	*diagnosis*	*radius_mean*	*texture_mean*	*perimeter_mean*
1	842302	M	17.99	10.38	122.8
2	842517	M	20.57	17.77	132.9
3	84300903	M	19.69	21.25	130
4	84348301	M	11.42	20.38	77.58
5	84358402	M	20.29	14.34	135.1
6	843786	M	12.45	15.7	82.57
7	844359	M	18.25	19.98	119.6
8	84458202	M	13.71	20.83	90.2
9	844981	M	13	21.82	87.5
10	842302	M	17.99	10.38	122.8
…	…	…	…	…	…

The column named *diagnosis* contains either an "M" if the mass was found to be malignant or a "B" if it was found to be benign. The 30 columns beginning with *radius_mean* contain measurements made on each mass. It is desired to predict whether a mass is malignant or benign using those measurements. Additional information about the data may be found in the Machine Learning Repository at:
https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Pressing the button labeled *Step 1* displays the following data input dialog box:



The information to be entered is:

Output Variable: name of the column containing the variable to be predicted.

Type of Problem: Select *classification* if the goal is to predict which class or group a case belongs to. Select *regression* if the goal is to predict the quantitative value of the output variable. To fit a regression model, the output variable must be numeric. To fit a classification model, the output variable may be numeric or non-numeric.

Categorical Features: names of columns containing categorical features to be used in predicting the output. These columns may be either numeric or non-numeric. If numeric, the values will be treated as distinct levels in the same manner as non-numeric labels.

When fitting supervised learning models, categorical factors are converted to 0-1 numeric columns using “one-hot encoding” which creates columns similar to dummy variables used when fitting statistical models.

Quantitative Features: names of columns containing quantitative features to be used in predicting the output. These columns must be numeric and will be treated as continuous variables.

Weights: name of an optional column containing weights to be applied to each row in the datasheet. This permits some cases to have more influence than others in constructing the model.

Select: optional Boolean column or expression identifying the cases (rows of the Databook) to be included in the analysis.

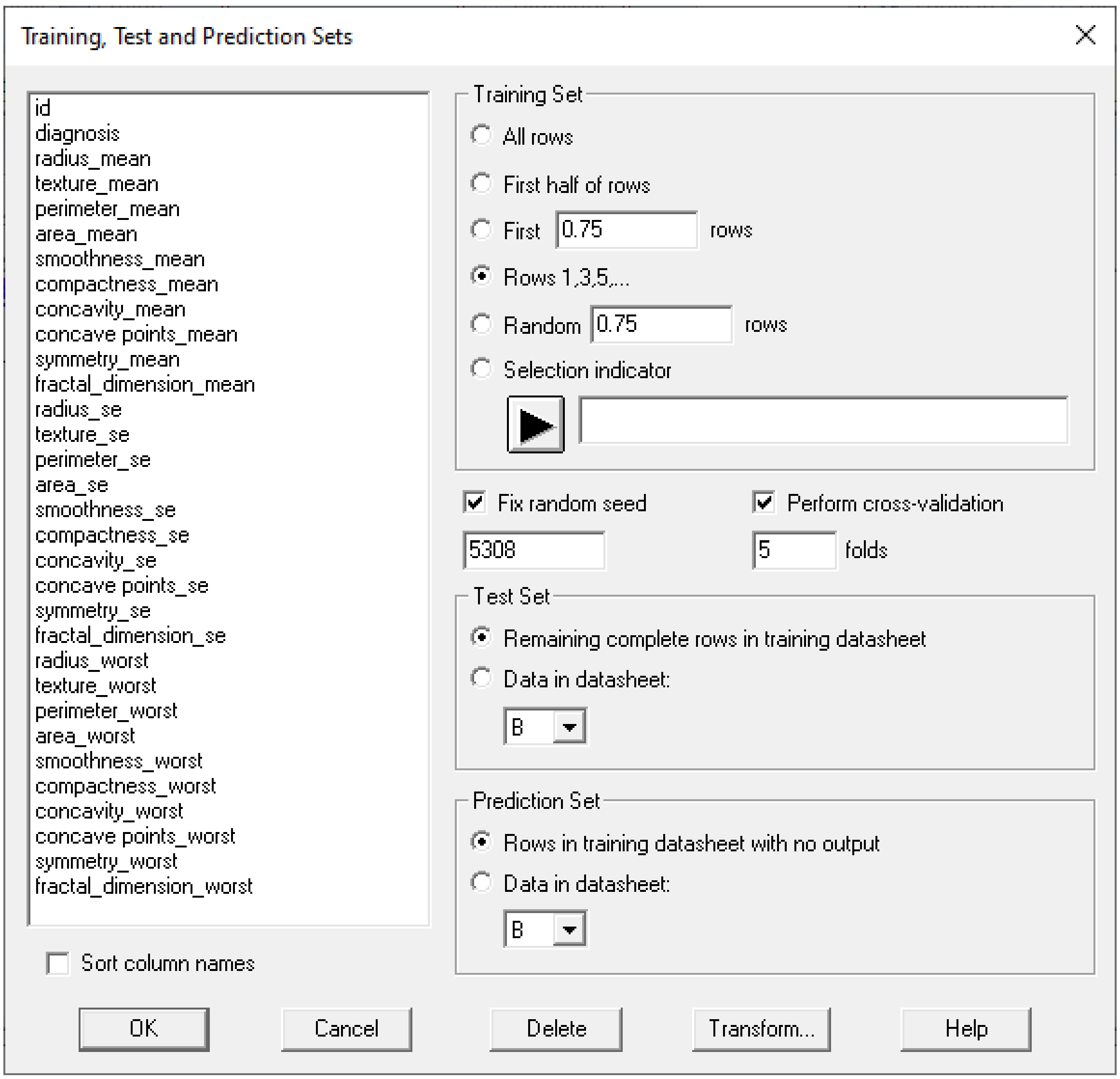
For the breast sample data, the output variable to be predicted is *diagnosis*. 30 quantitative features will be considered as potential predictors.

Step 2: Define Training, Test and Prediction Sets

Cases are typically divided into three sets:

1. A *training* set which is used to construct the model.
2. A *test* set for which the actual output classification or value is known, which can be used to validate the model.
3. A *prediction* set for which the actual output classification or value is not known but for which predictions are desired.

When the button labeled *Step 2* is pressed, the following dialog box is displayed:



Specify the following information:

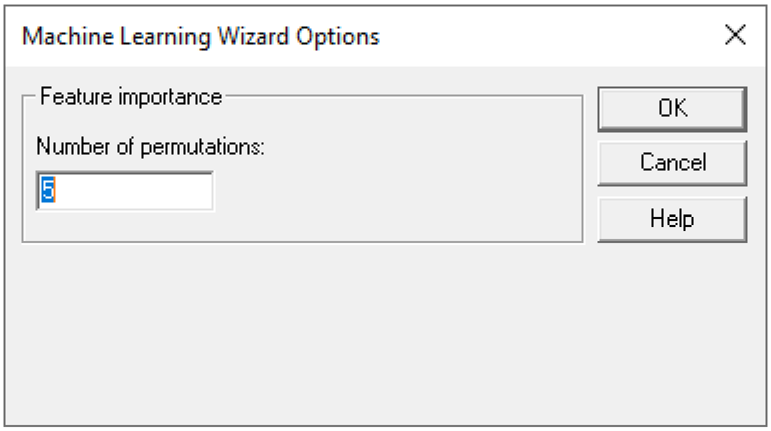
- **Training set:** these cases will be used to train (develop) the models. You may use:
    - *All rows* – all of the rows in the datasheet.
    - *First half of rows* – the topmost half of the rows in the datasheet.
    - *First _____ rows* – specify the number or fraction of rows at the top of the datasheet. If the value entered is less than 1, it is assumed to represent the fraction of rows to be used. As an example, entering 0.75 indicates that the first 75% of the rows should be used.
    - *Rows 1,3,5,…* – use every other row beginning with row 1.
    - *Random _____ rows* – specify the number or fraction of rows to be randomly selected from all rows in the datasheet. If the value entered is less than 1, it is assumed to represent the fraction of rows to be used. As an example, entering 0.75 indicates that a randomly selected 75% of the rows should be used.
    - *Selection indicator* – enter a Statgraphics expression to specify which rows should be used. Any row resulting in something other than 0 will be included in the training set.
- **Fix random seed** – if checked, the random number generator will be seeded with the specified value. This allows you to obtain identical results when repeating the training process more than once.
- **Perform cross-validation** – if checked, cross-validation will be performed on the machine learning models to assess their likely performance on data that were not used to train the model. For example, entering "5" in the *folds* field requests 5-fold cross-validation. This instructs the program to divide the training data into 5 folds, each containing 20% of the cases. 5 models are then trained, each using all but one fold. The models are then used to predict the 20% of the cases that were withheld and the average score across all folds is calculated.
- **Test set:** these cases will be used to test the trained models. You may use:
    - *Remaining complete rows in training datasheet* – uses all rows in the datasheet from which the training data were obtained that were not used to train the model.
    - *Data in datasheet _____*: indicates a different datasheet that contains the test data. The column names in this datasheet must be the same names as those in the training datasheet.
- **Prediction set:** these cases contain cases for which the output is not known but a prediction is desired. You may select:

- *Rows in training datasheet with no output* – uses all rows in the datasheet from which the training data were obtained that do not contain an entry for the output variable.

- *Data in datasheet* _____: indicates a different datasheet that contains the cases to be predicted. The column names in this datasheet must be the same as those in the training datasheet.

Step 3: Set Options

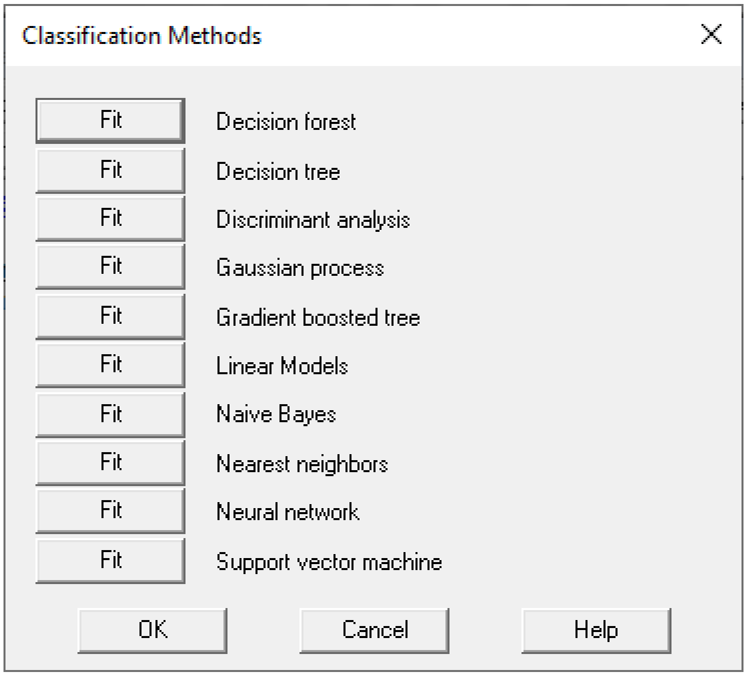
The button labeled *Step 3: Set options* lets you override options that will be applied to all predictive models generated by the wizard.



- **Number of permutations:** number of times that features are randomly shuffled when determining feature importance. Permutation feature performance is defined to be the average reduction in model score when a particular feature is randomly shuffled within its column.

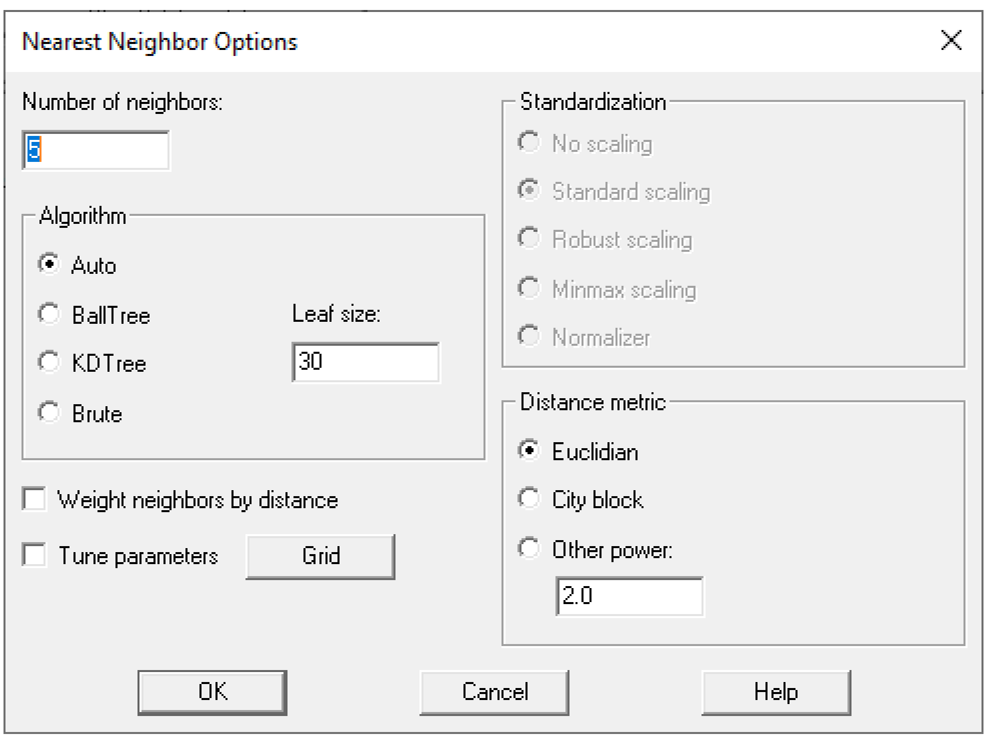
Step 4: Fit Models

The fourth step in using the wizard is to fit one or more models to the data. When the button labeled *Step 4: Fit models* is pressed, the following dialog box is displayed:



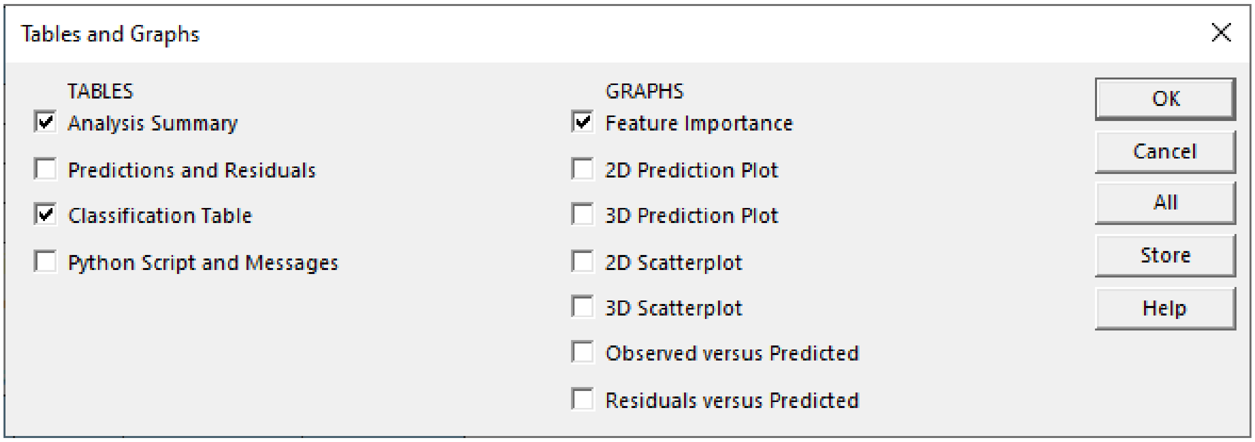
Pressing any button will launch the indicated procedure. Step 4 may be executed multiple times to generate more than one model.

As an example, pressing *Fit* next to *Nearest neighbors* invokes the Scikit-Learn *KNeighborsClassifier* or *KNeighborsRegressor* procedure. The first dialog box to be displayed is the *Analysis Options* dialog box for that procedure:

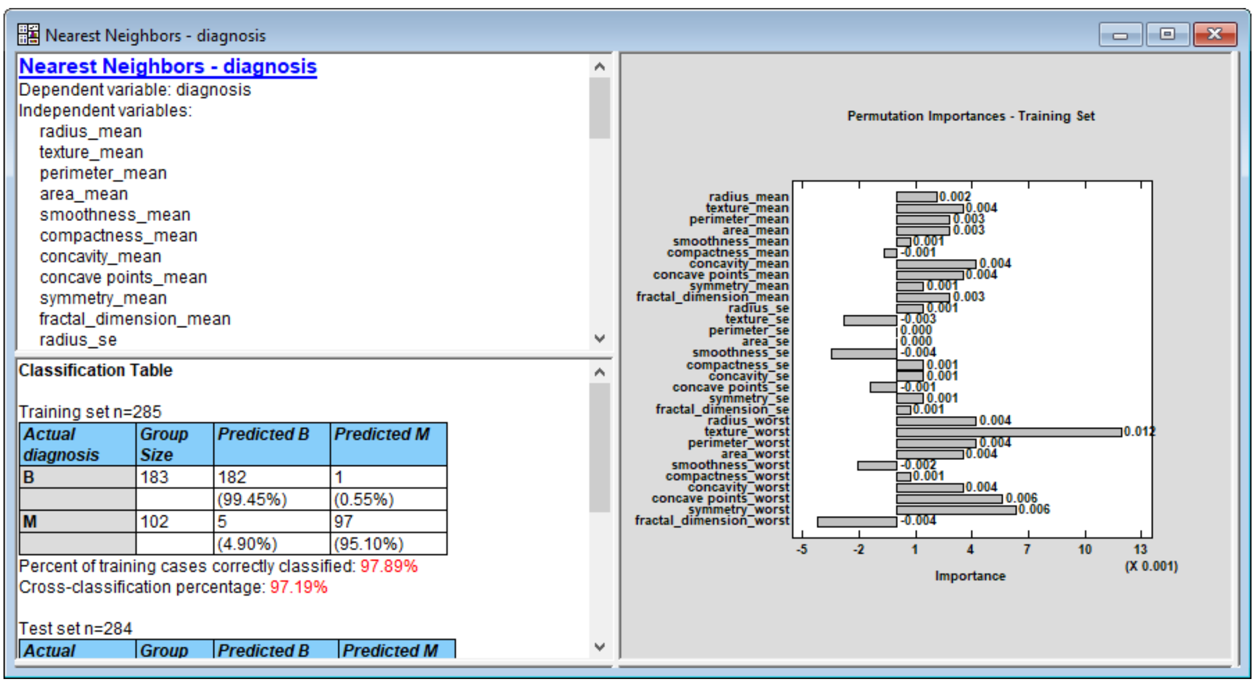


Select the options you wish to use for that model and press *OK*. Note: the options for each method are described in separate PDF documents.

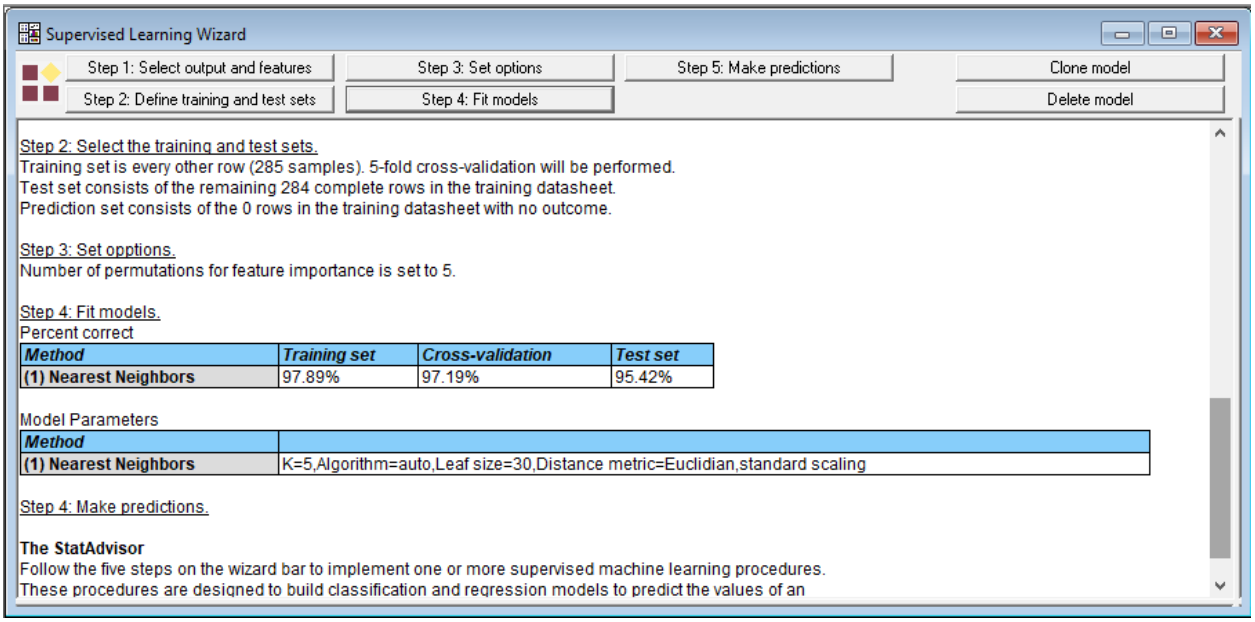
Next, select the tables and graphs you wish to create for that model:



Then press *OK* and a new window will open showing the results of fitting the selected procedure:



At the same time, information about the results will be added to the Wizard's window:


Supervised Learning Wizard
Step 1: Select output and features
Step 2: Define training and test sets
Step 3: Set options
Step 4: Fit models
Step 5: Make predictions
Clone model
Delete model

Step 2: Select the training and test sets.
Training set is every other row (285 samples). 5-fold cross-validation will be performed.
Test set consists of the remaining 284 complete rows in the training datasheet.
Prediction set consists of the 0 rows in the training datasheet with no outcome.

Step 3: Set opptions.
Number of permutations for feature importance is set to 5.

Step 4: Fit models.
Percent correct

Method	Training set	Cross-validation	Test set
(1) Nearest Neighbors	97.89%	97.19%	95.42%

Model Parameters

Method	
(1) Nearest Neighbors	K=5,Algorithm=auto,Leaf size=30,Distance metric=Euclidian,standard scaling

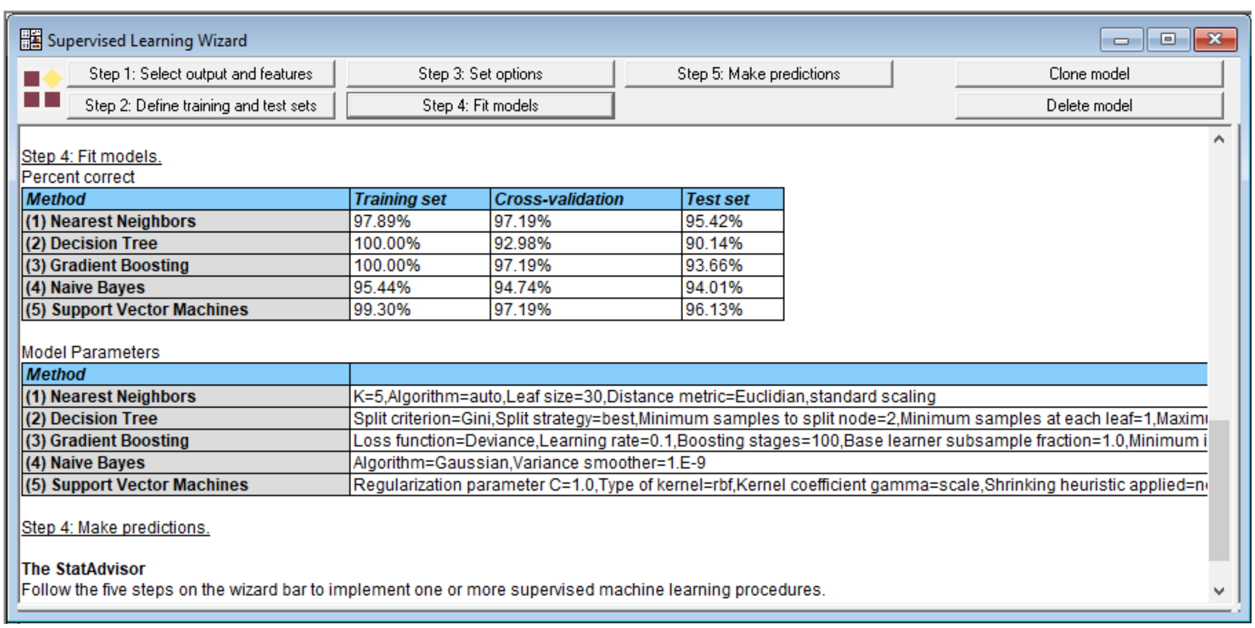
Step 4: Make predictions.

The StatAdvisor
Follow the five steps on the wizard bar to implement one or more supervised machine learning procedures.
These procedures are designed to build classification and regression models to predict the values of an

In this case, the model correctly predicted 97.89% of the cases in the training set. In the cross-validation study, the percent correctly predicted equaled 97.19%. More importantly, the nearest neighbors method predicted 95.42% of the test cases correctly even though they were not used to fit the model.

Model Comparisons

Pressing the Step 4 button and pressing *Fit* again will generate additional windows with output from other procedures (or the same procedure with different options). For example, the table below shows the results of trying 5 methods with their default options:


Supervised Learning Wizard
Step 1: Select output and features
Step 2: Define training and test sets
Step 3: Set options
Step 4: Fit models
Step 5: Make predictions
Clone model
Delete model

Step 4: Fit models.
Percent correct

Method	Training set	Cross-validation	Test set
(1) Nearest Neighbors	97.89%	97.19%	95.42%
(2) Decision Tree	100.00%	92.98%	90.14%
(3) Gradient Boosting	100.00%	97.19%	93.66%
(4) Naive Bayes	95.44%	94.74%	94.01%
(5) Support Vector Machines	99.30%	97.19%	96.13%

Model Parameters

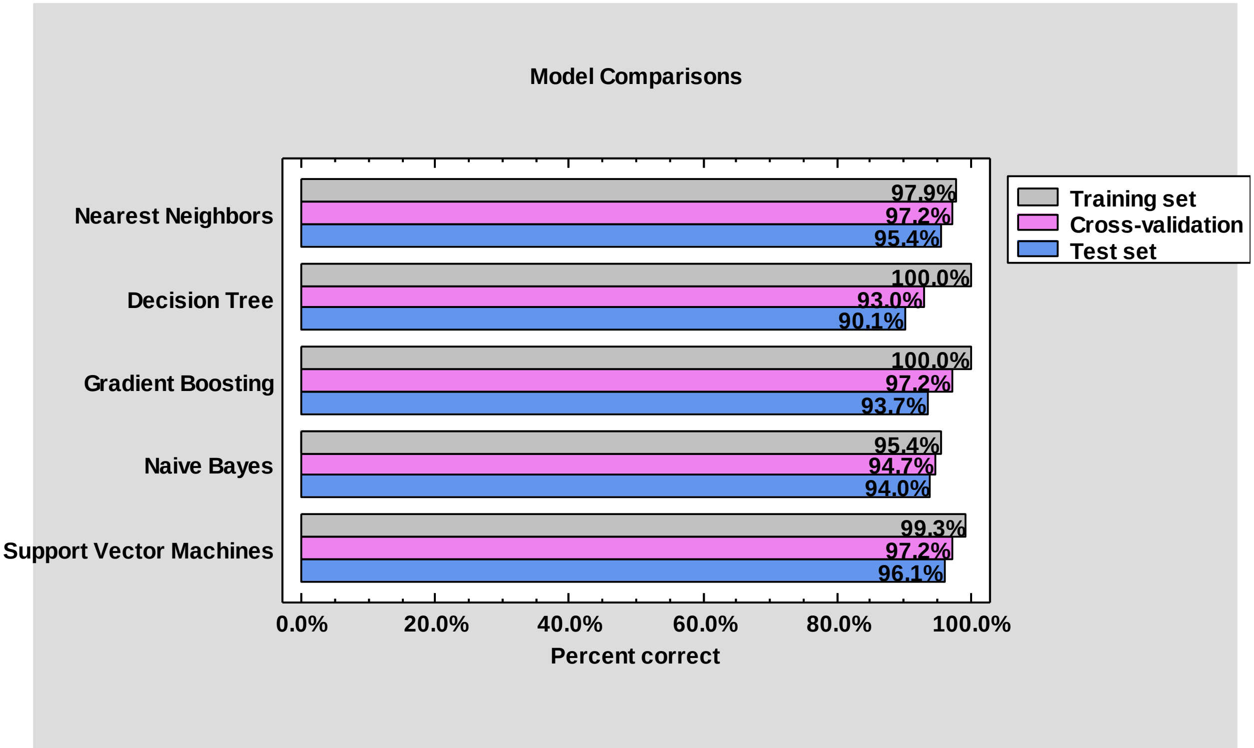
Method	
(1) Nearest Neighbors	K=5,Algorithm=auto,Leaf size=30,Distance metric=Euclidian,standard scaling
(2) Decision Tree	Split criterion=Gini,Split strategy=best,Minimum samples to split node=2,Minimum samples at each leaf=1,Maxim
(3) Gradient Boosting	Loss function=Deviance,Learning rate=0.1,Boosting stages=100,Base learner subsample fraction=1.0,Minimum i
(4) Naive Bayes	Algorithm=Gaussian,Variance smoother=1.E-9
(5) Support Vector Machines	Regularization parameter C=1.0,Type of kernel=rbf,Kernel coefficient gamma=scale,Shrinking heuristic applied=n

Step 4: Make predictions.

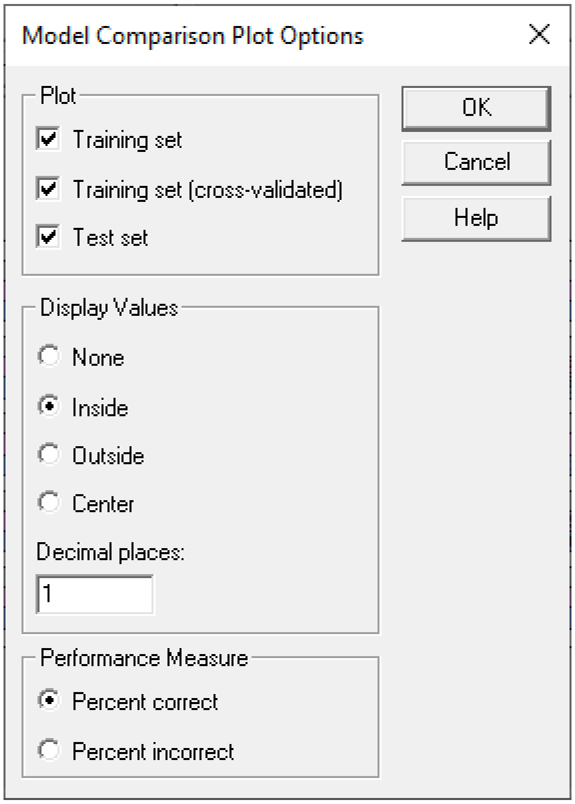
The StatAdvisor
Follow the five steps on the wizard bar to implement one or more supervised machine learning procedures.

Of the 5 methods, the *Support Vector Machines* gave the highest correct percentage on the test data.

The wizard will also create a plot showing the performance of the models:



<u>Pane Options</u>



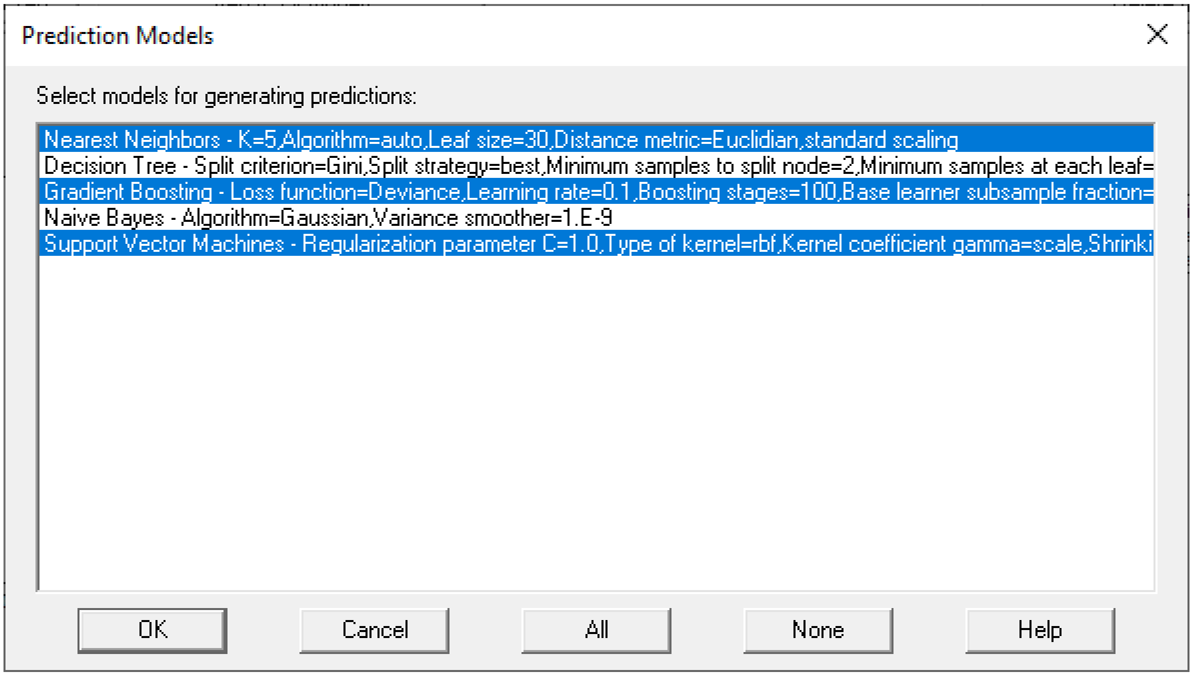
- **Plot** – specify the sets for which the results should be plotted.

- **Display Values** – specify the location with respect to the bars where the values should be plotted and the number of decimal places to display.

- **Performance Measure** – the performance measure to plot.

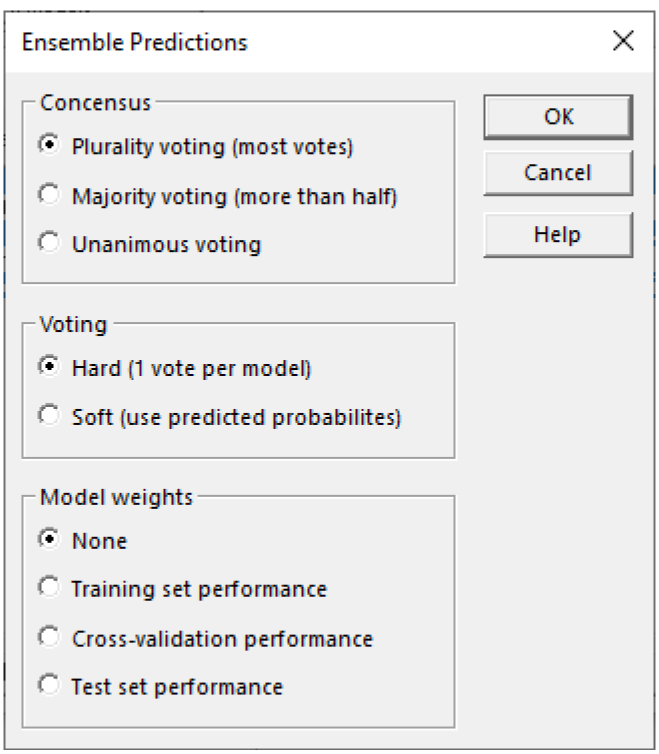
Step 5: Make Predictions

The final step in using the wizard is to make predictions for cases in the prediction set. These are typically cases for which the actual value of the output variable is not known. When the *Step 5: Make predictions* button is pressed, a dialog box is displayed showing the models that have been fit:



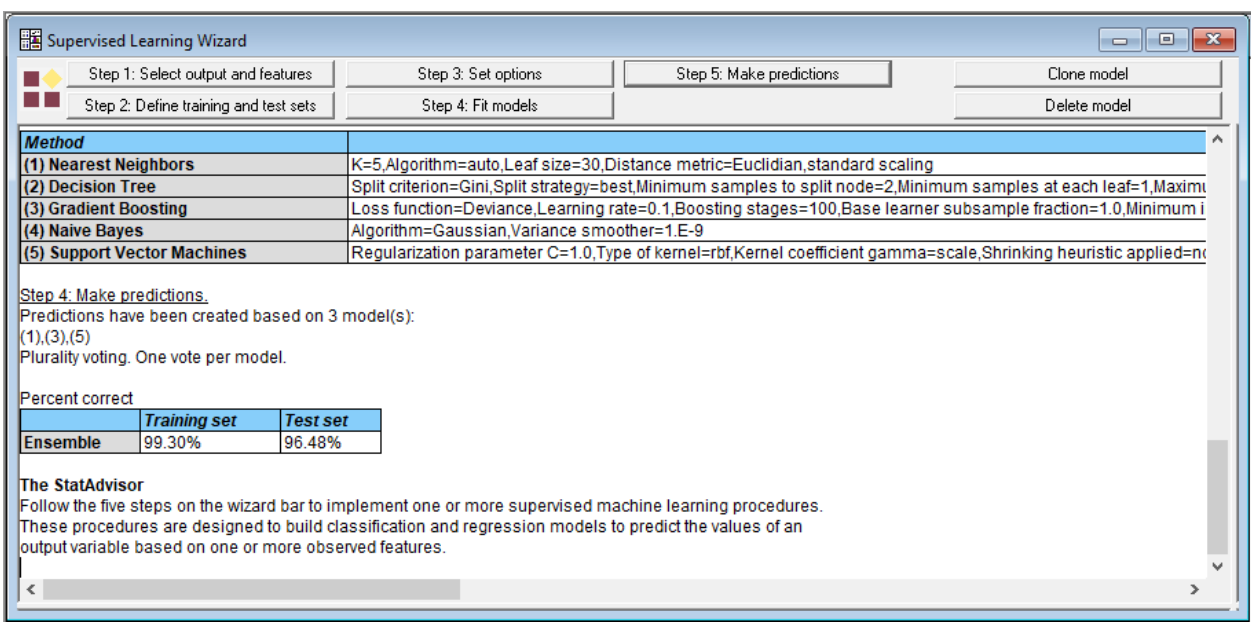
Select one or more models. To avoid ties, it is often best to select an odd number of models.

When you press *OK*, a second dialog box will be displayed to specify how predictions of the models should be combined:



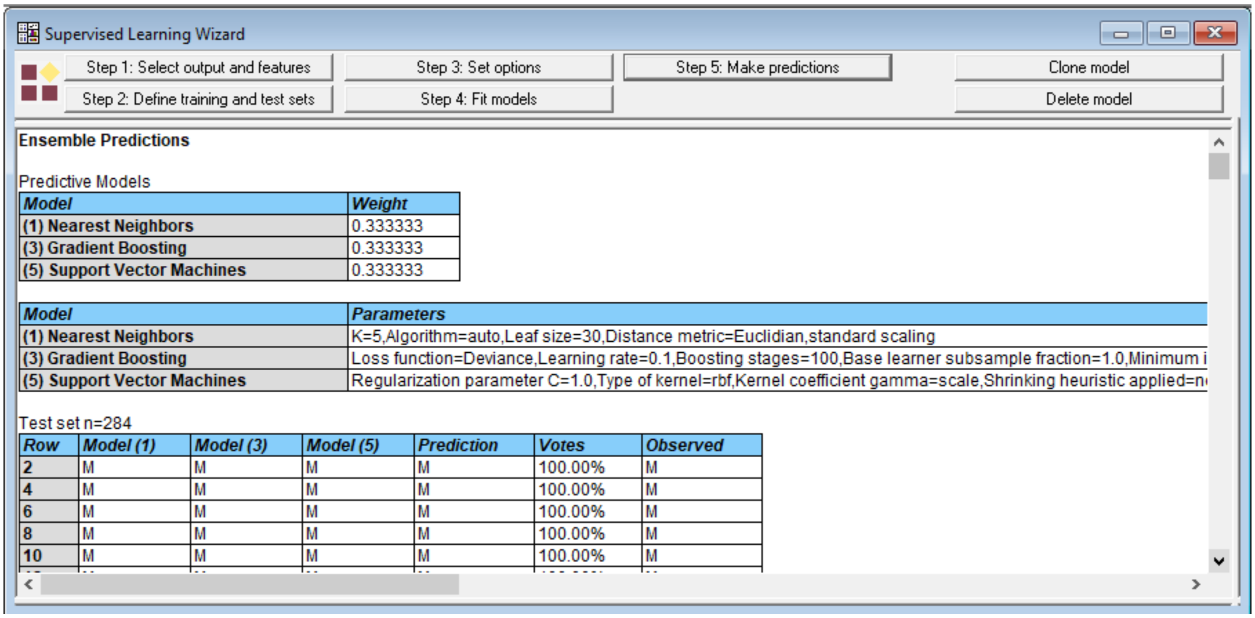
- **Consensus** – determines how decisions will be made to make a prediction for each selected case.
    - **Plurality voting** – the predicted category is the one receiving the most votes. Predictions are made whether or not the leading category has more than 50% of the votes.
    - **Majority voting** – the predicted category is the one receiving the most votes. Predictions are made only if the leading category has more than 50% of the votes.
    - **Unanimous voting** – the predicted category is the one receiving the most votes. Predictions are made only if the leading category receives all of the votes.
- **Voting –** specifies how votes are allocated.
    - **Hard –** each model gets a single vote as determined in their respective analysis windows.
    - **Soft –** each model allocates its vote proportionately based on the estimated probabilities determined for each category. Note that for some methods, the category with the highest probability may not be the category selected as the prediction in the corresponding analysis window.
- **Model weights –** how much importance is given to each model's vote.
    - **None –** each model is treated as equally important.
    - **Training set performance –** models with better performance on the training set are weighted more when tallying the votes.
    - **Cross-validation performance –** models with better cross-validation are weighted more when tallying the votes.
    - **Test set performance –** models with better performance on the test set are weighted more when tallying the votes.

Pressing OK on the second box causes the program to generate predictions for each observation. The ensemble predictions are summarized in the main Wizard window:



In this case, the combination of the 3 selected models correctly predicted 96.48% of the cases in the test set, which is better than any single model by itself.

You can see the predictions made for each observation by selecting the *Essemble Predictions* from the list of tables and graphs provided in the main wizard window:



Model	Weight
(1) Nearest Neighbors	0.333333
(3) Gradient Boosting	0.333333
(5) Support Vector Machines	0.333333

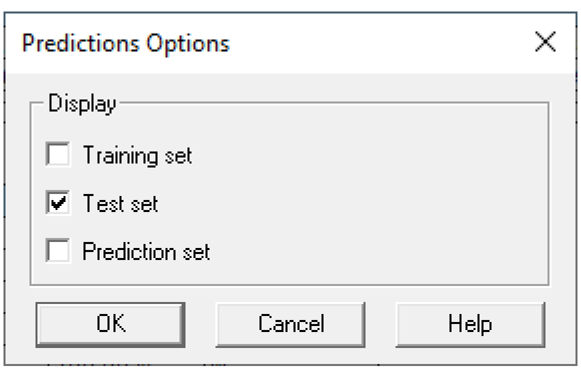
Model	Parameters
(1) Nearest Neighbors	K=5,Algorithm=auto,Leaf size=30,Distance metric=Euclidian,standard scaling
(3) Gradient Boosting	Loss function=Deviance,Learning rate=0.1,Boosting stages=100,Base learner subsample fraction=1.0,Minimum i
(5) Support Vector Machines	Regularization parameter C=1.0,Type of kernel=rbf,Kernel coefficient gamma=scale,Shrinking heuristic applied=n

Row	Model (1)	Model (3)	Model (5)	Prediction	Votes	Observed
2	M	M	M	M	100.00%	M
4	M	M	M	M	100.00%	M
6	M	M	M	M	100.00%	M
8	M	M	M	M	100.00%	M
10	M	M	M	M	100.00%	M

The table shows:

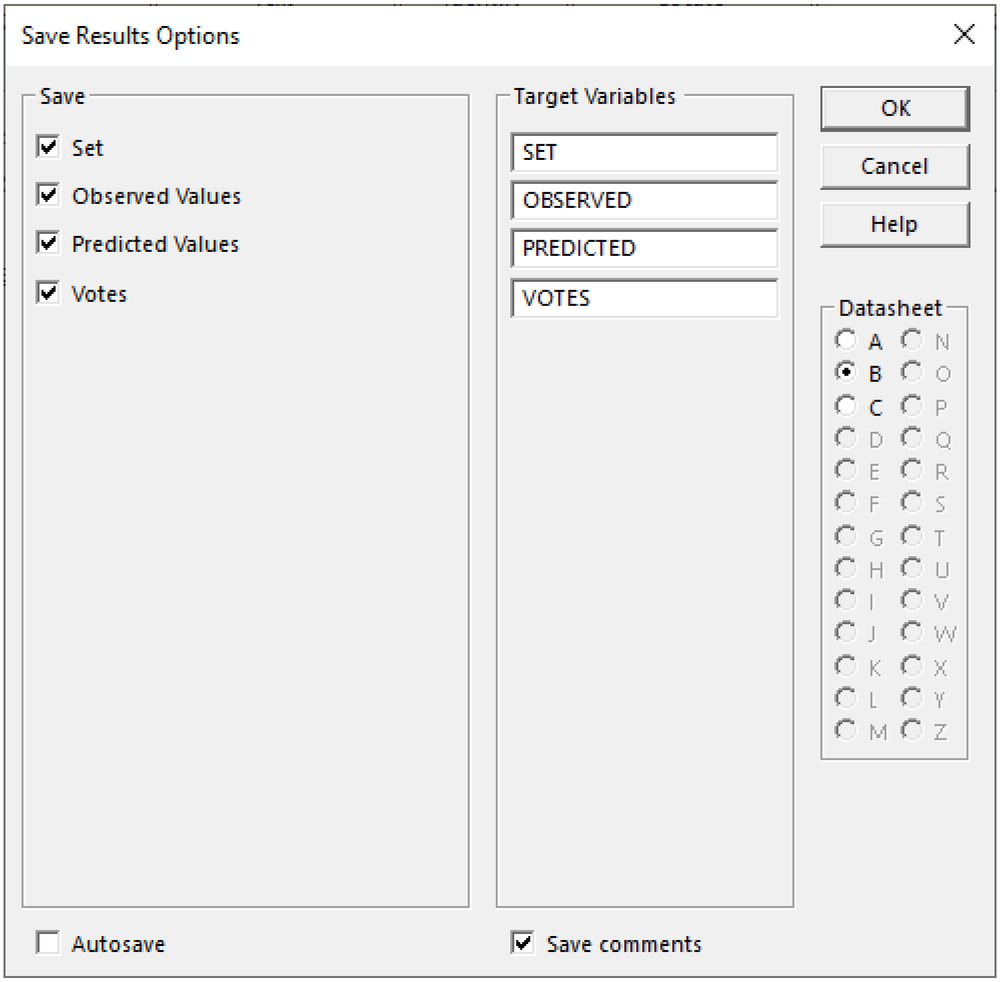
1. *Model (#)* - the predictions made by each of the selected models.
2. *Prediction* – the ensemble predictions made by combining the models.
3. *Votes* – the percentage of votes received by the category corresponding to the ensemble predictions.
4. *Observed* – the actual category (for the training and test sets only).

You may control the sets displayed by selecting *Pane Options:*



<u>Saving Results</u>

Once predictions are made, those predictions may be saved in the DataBook. To do so, select *Results* in the *Output* section of the ribbon bar. This will display the following dialog box:



To save results, select:

- **Save:** select the items to be saved.
- **Target Variables:** enter names for the columns to be created.
- **Datasheet:** the datasheet into which the results will be saved.
- **Autosave:** if checked, the results will be saved automatically each time a saved StatFolio is loaded.
- **Save comments:** if checked, comments for each column will be saved in the second line of the datasheet header.

Press *OK* to save the selected information in the Databook.

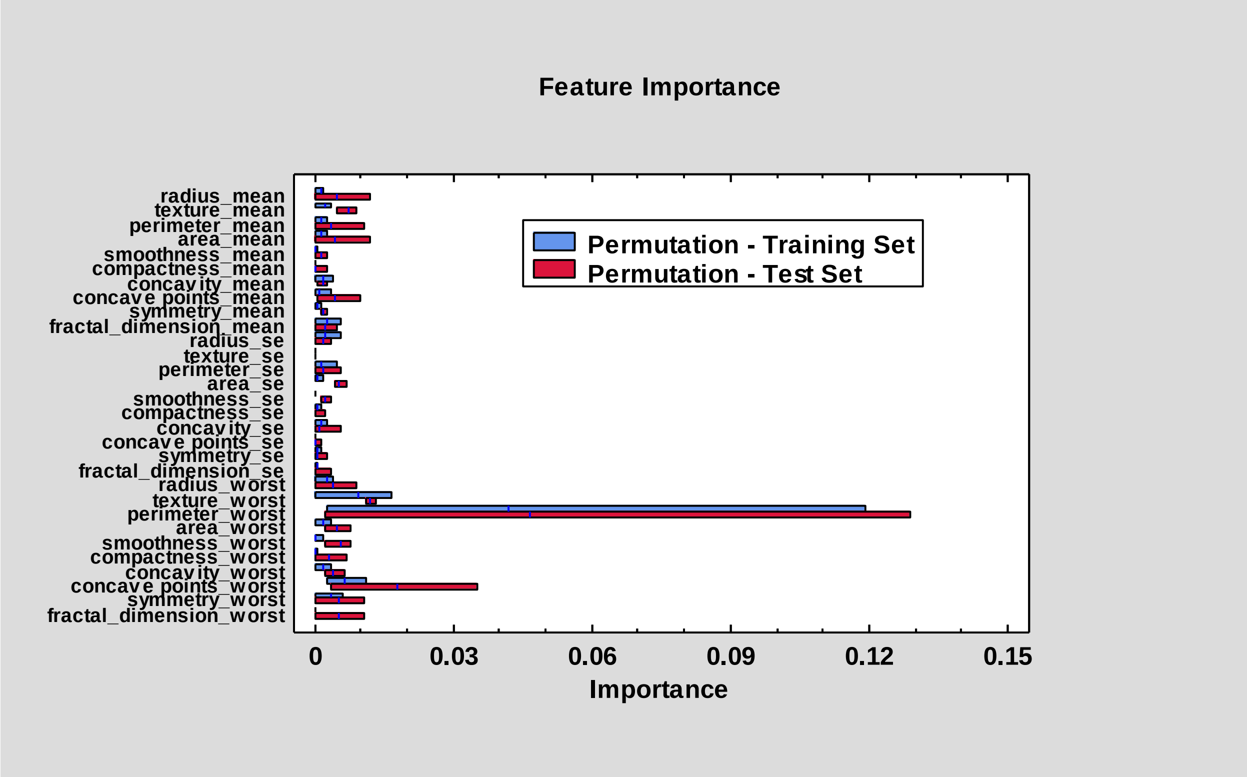
<untitled>

	SET	OBSERVED	PREDICTED	VOTES
	Set	Observed Values	Predicted Values	Votes
	Character	Character	Character	Percentage
1	Training	M	M	100%
2	Test	M	M	100%
3	Training	M	M	100%
4	Test	M	M	100%
5	Training	M	M	100%
6	Test	M	M	100%
7	Training	M	M	100%
8	Test	M	M	100%
9	Training	M	M	100%
10	Test	M	M	100%
11	Training	M	M	100%
12	Test	M	M	100%
13	Training	M	M	100%

breast cancer | B

Feature Importance

The wizard will also create a plot to display the range of feature importances calculated by each model that has been included in the ensemble predictions. It creates a graph such as that shown below:

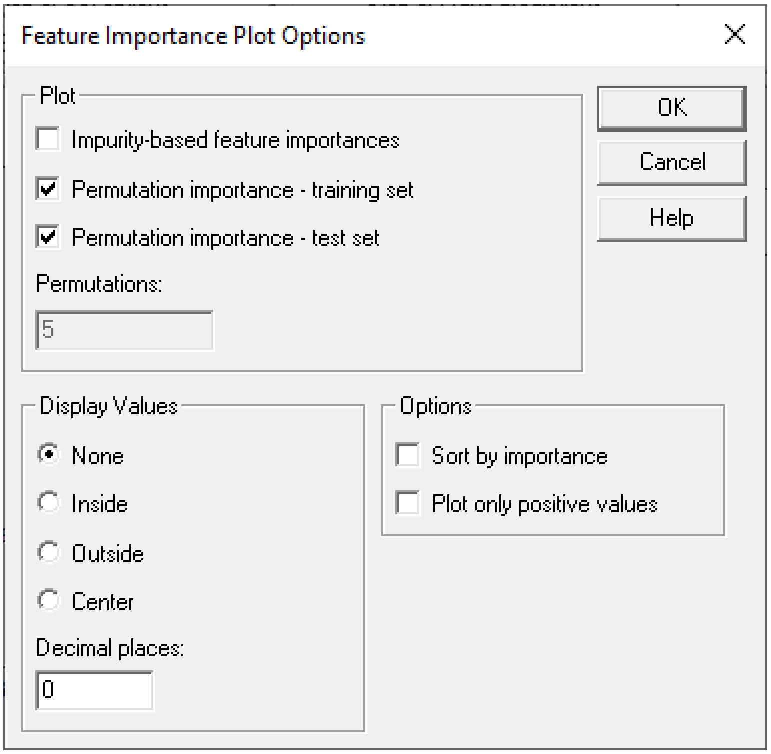


Feature importance is defined as the average reduction in model score caused by randomly shuffling the values in a particular feature column. In the above plot, there are 2 bars for each feature, one showing its importance when applied to the training set and the second showing the importance when applied to the test set. The bar ranges from the smallest feature importance amongst the models selected to the largest importance amongst the models selected. There is also a vertical line through each bar, located at the weighted average of feature importance across all of the predictive models, using the *model weights* defined during *Step 5*.

In the above plot, the most important feature is clearly *perimeter_worst*. *Concave_points_worst* was also important, particularly when the model was applied to the test set.

Pane Options

Various features of the plot may be changed using *Pane Options*:



- **Plot:** indicates which feature importancer should be plotted. In addition to the permutation importance, the *Gradient Boosting* procedure also generates a feature importance estimate based on the reduction of model inaccuracies attributable to a factor during model estimation.

- **Permutations:** the number of times the features are shuffled (set during *Step 3*).

- **Display values:** values to display next to each bar.

- **Sort by importance:** if checked, the features will be plotted in order of importance.

- **Plot only positive values:** if checked, only features for which the importance is greater than 0 for at least one model will be plotted.

Classification Table

This table shows how well the ensemble of predictive models performs in classifying the observations:

Classification Table

Training set n=285

Actual diagnosis	Group Size	Predicted B	Predicted M
B	183	183	0
		(100.00%)	(0.00%)
M	102	2	100
		(1.96%)	(98.04%)

Percent of training cases correctly classified: 99.30%

Test set n=284

Actual diagnosis	Group Size	Predicted B	Predicted M
B	174	173	1
		(99.43%)	(0.57%)
M	110	9	101
		(8.18%)	(91.82%)

Percent of test cases correctly classified: 96.48%

It shows:

- **Actual diagnosis:** There is a row for each level of the output variable.
- **Group Size:** the number of cases in the training and test sets that fall in the each class.
- **Predicted:** the number of cases in the training and test sets that were predicted to fall in each class.

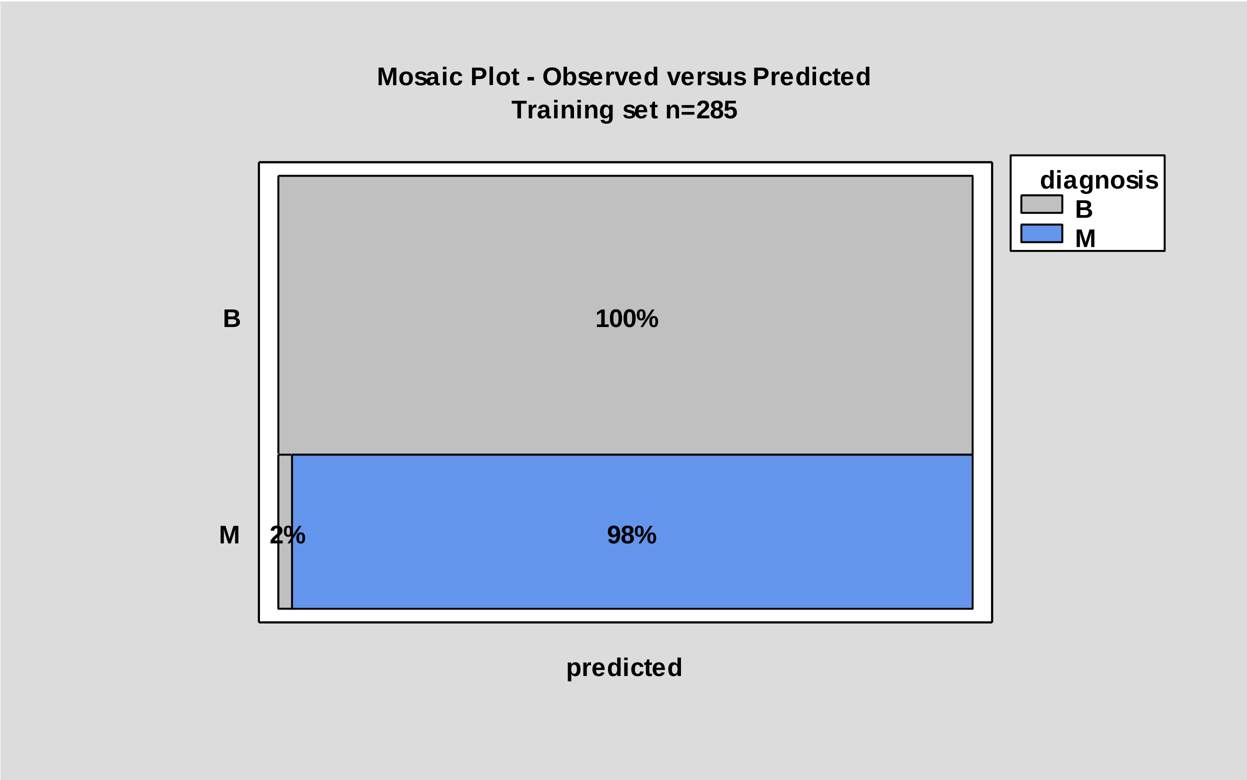
The percentage of correctly classified observations is displayed for each level of the output variable and for all of the observations combined.

For example, there were 183 cases with *benign* masses in the training set. All were correctly predicted to be *benign*. There were 102 cases with *malignant* masses in the training set. All but 2 were correctly predicted to be *malignant*. Overall, of the 285 cases in the training set, 99.3% were correctly classified.

A separate table is displayed for any cases in the test set. As expected, performance in the test set was not as good as in the training set.

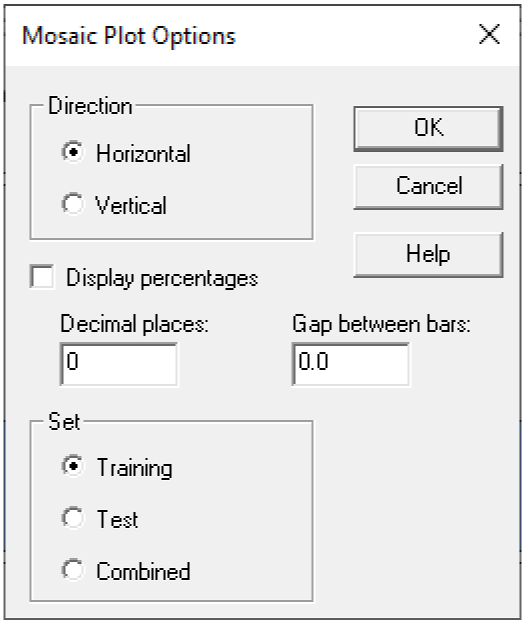
Observed versus Predicted

For a classification model, this selection displays a mosaic plot comparing the observed values of the output variable to the ensemble predictions:



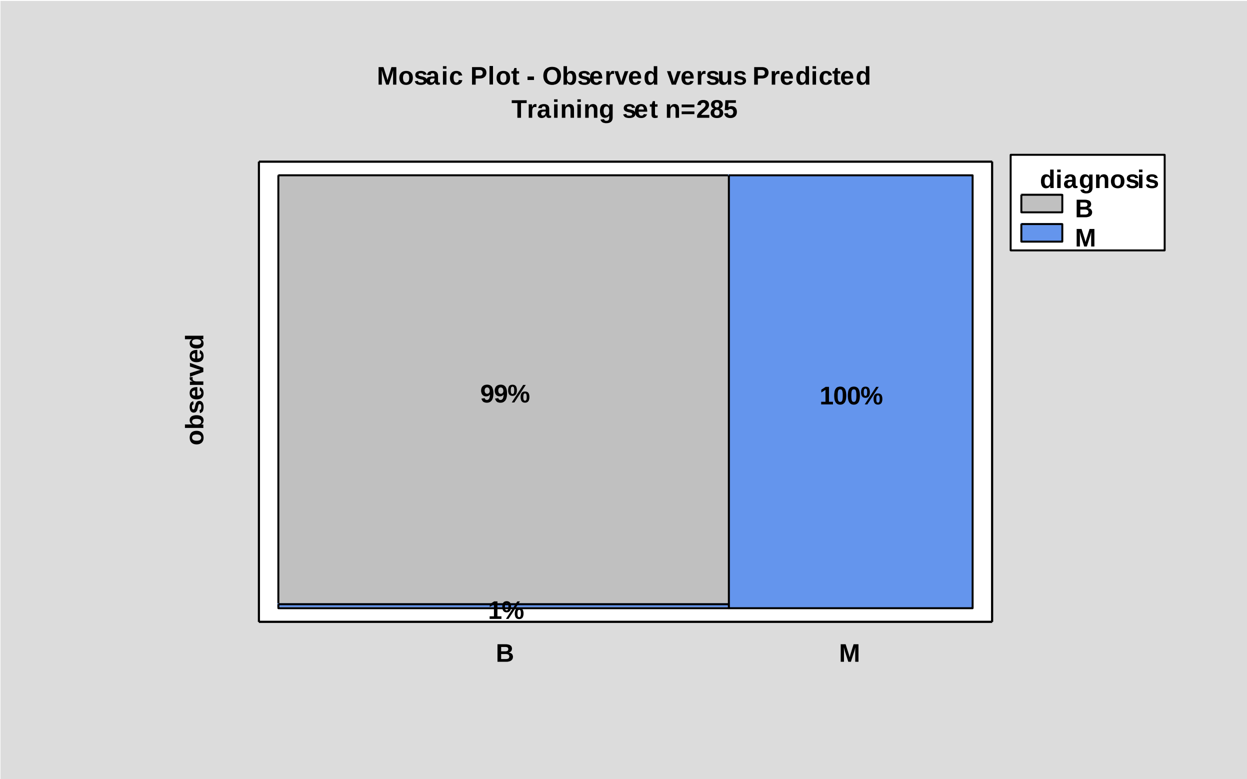
In horizontal format, the width of each bar in the vertical direction is proportional to the number of cases in the selected set that were observed to belong to each class. The length of the bars in the horizontal direction shows the proportional breakdown of predicted values among those observed cases. For example, the above plot shows that a majority of the observed tumors were benign. All benign tumors were correctly predicted to be benign, but about 2% of the malignant tumors were also predicted to be benign.

<u>Plot Options</u>



- **Direction:** orientation of the bars.
- **Display percentage:** whether the plot should display the percentage corresponding to each bar.
- **Decimal places:** number of decimals places to include in the displayed percentages.
- **Gap between bars:** fractional space to use to separate each bar.
- **Set:** set of points to be displayed on the plot.

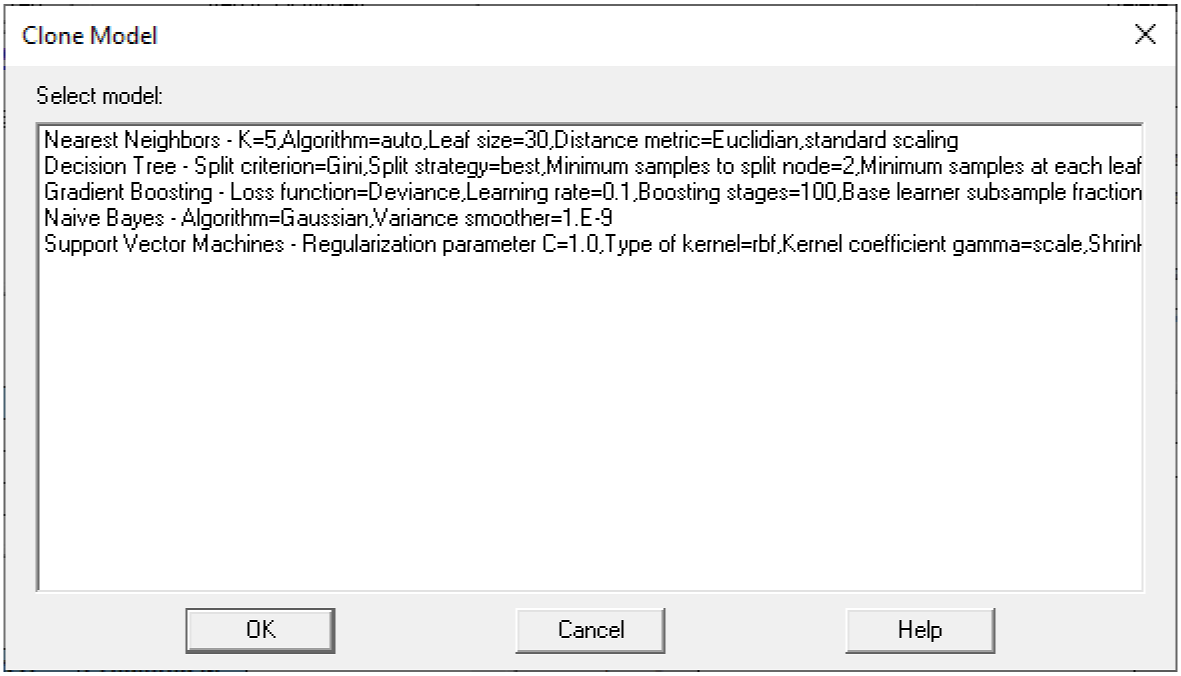
When displayed vertically, the plot vtakes the following form:



The width of the bars in the horizontal direction is proportional to the perentage of cases that were predicted to fall in each class. The length of the bars in the vertical direction shows the proportional breakdown of each predicted set according to its observed value. The above plot shows that about 1% of all cases predicted to be benign were actually malignant, while all cases predicted to be malignant were observed to be malignant.

Cloning Models

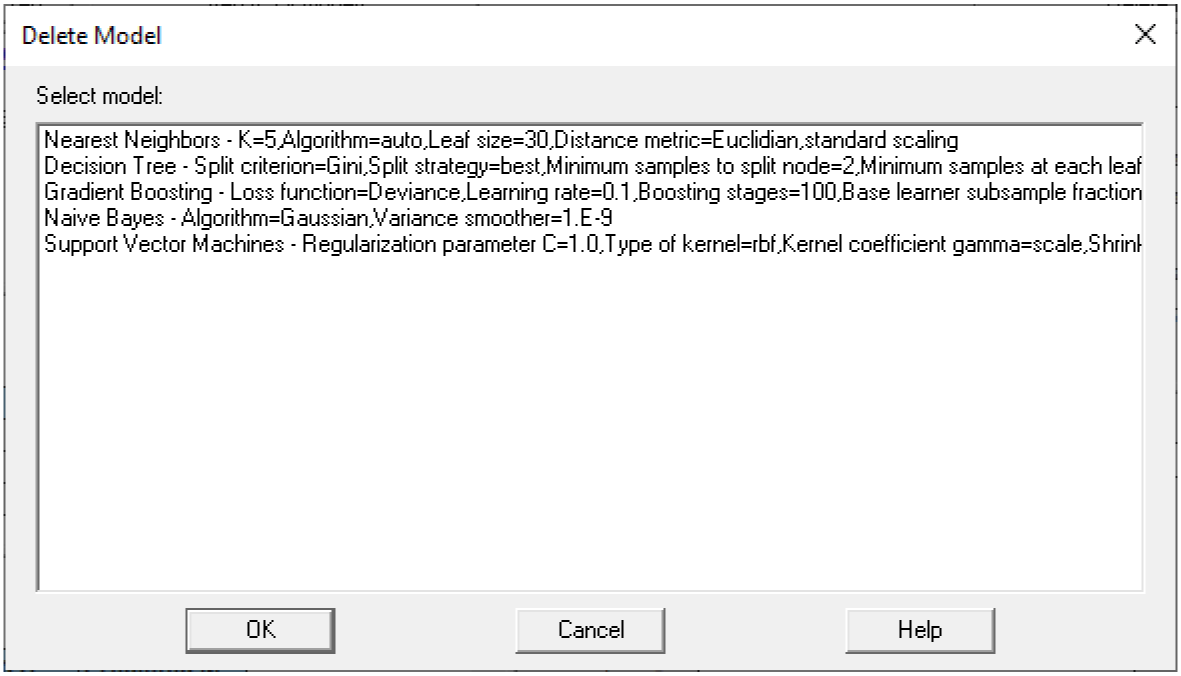
A copy of any fitted model can be added to the wizard by pressing the *Clone model* button on the wizard toolbar. This will display a dialog box showing all of the current models:



Click on the model that you wish to clone and press *OK*. This will create a new analysis window with the same model as that selected. You can then go to the new analysis window, select *Analysis Options*, and change the parameters of the model. This is designed to let you experiment with values of the model parameters without changing the original model.

Deleting Models

Any model that has been fit by the wizard can be removed by pressing *Delete model* on the wizard toolbar. This will display a dialog box showing all of the current models:



Click on the model that you wish to delete and press *OK*. This will permanently delete the analysis window selected and remove the model from the wizard. If the model had previously been selected for use in creating ensemble predictions, those predictions will be recalculated without that model.

Example 2: Regression Models

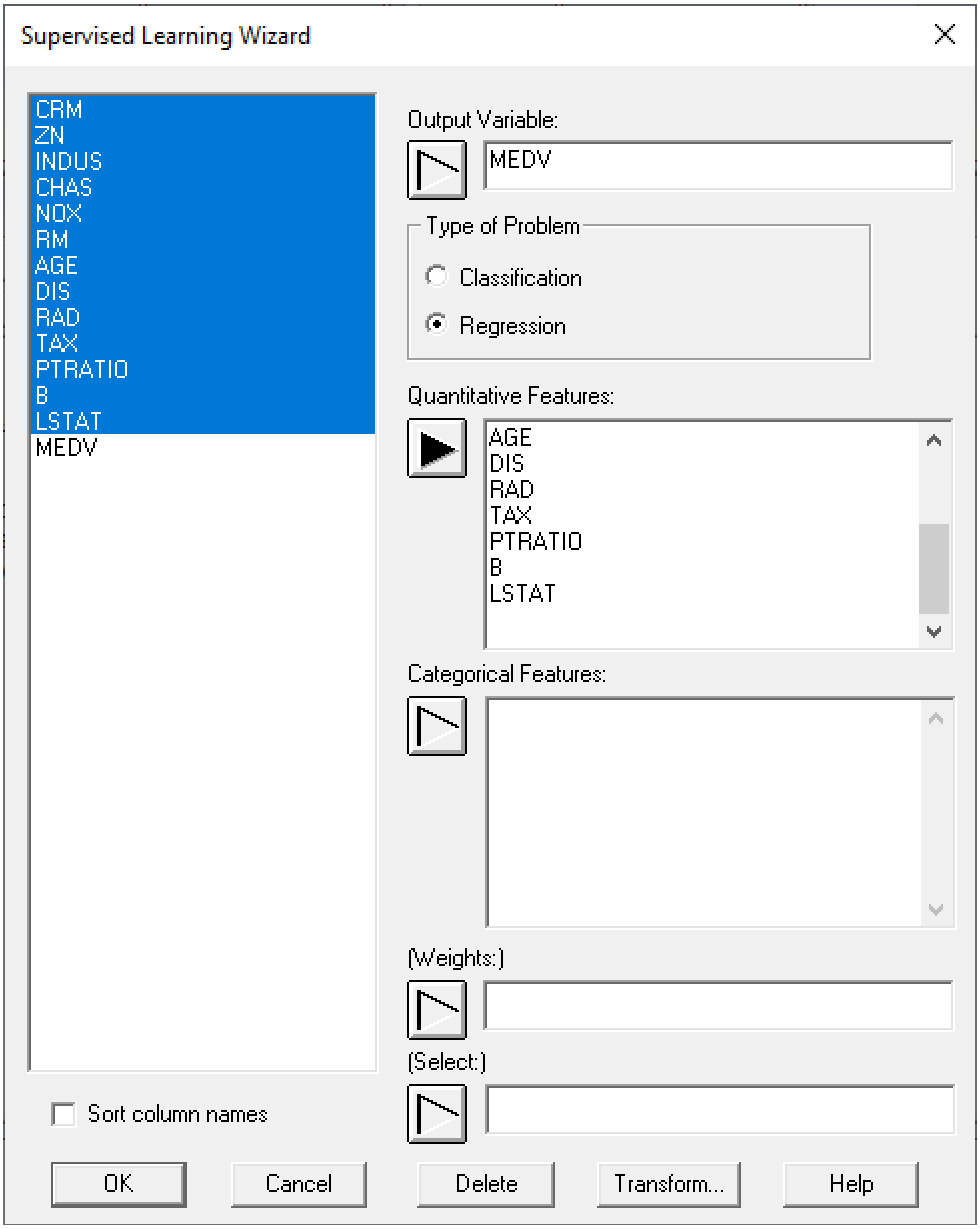
The second example considers the problem of predicting housing prices. As an example, consider the widely studied data describing housing prices in communities around Boston. It was first published by Harrison, D. and Rubinfeld, D.L. in an article titled "Hedonic prices and the demand for clean air" in the Journal of Evironmental Economics and Management.

The data consist of information about 506 communities. The columns in the file *boston house prices.sgd* are:

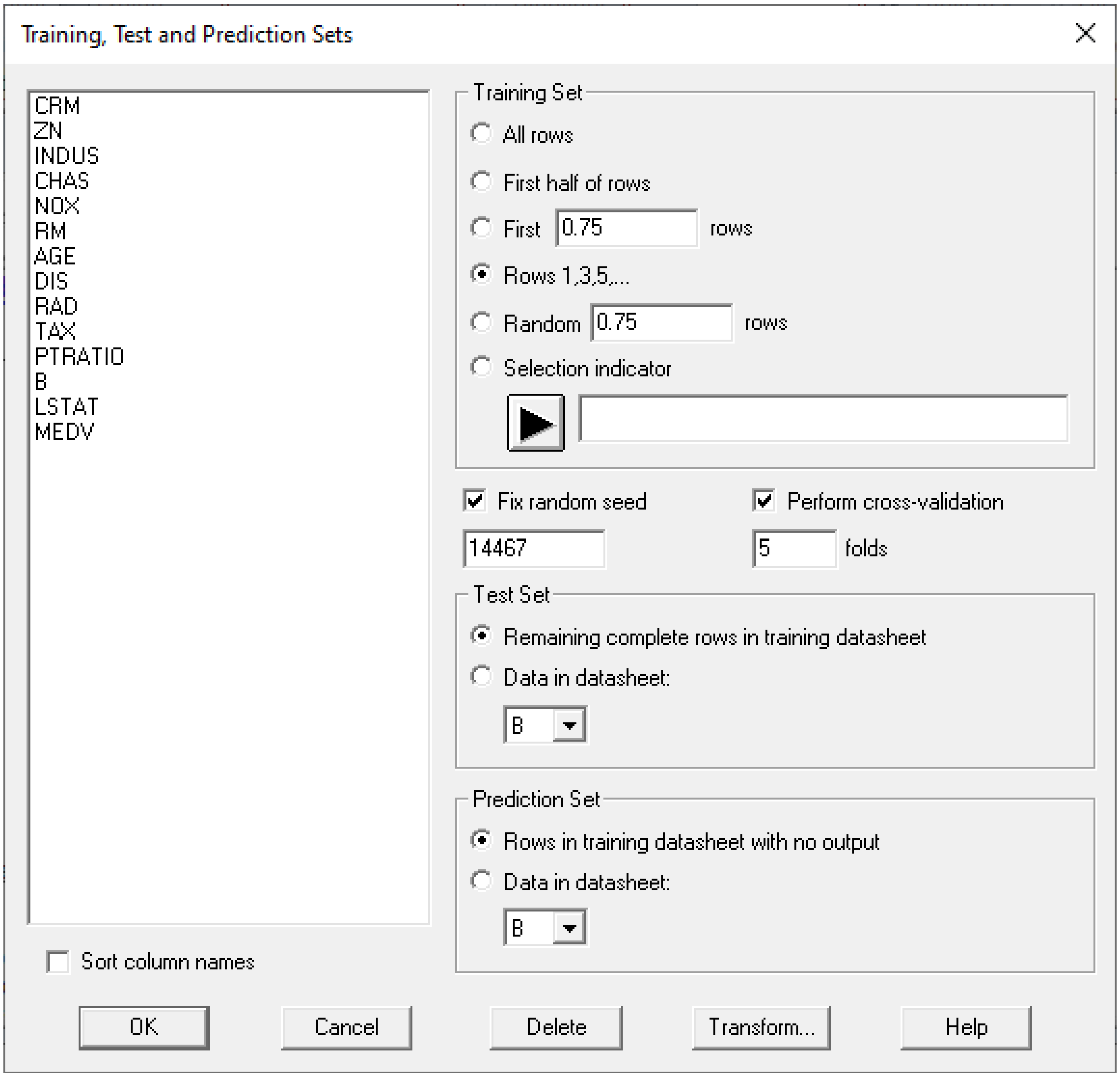
```
CRIM     per capita crime rate by town
ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS    proportion of non-retail business acres per town
CHAS     Charles River dummy var. (= 1 if tract bounds river; 0 otherwise)
NOX      nitric oxides concentration (parts per 10 million)
RM       average number of rooms per dwelling
AGE      proportion of owner-occupied units built prior to 1940
DIS      weighted distances to five Boston employment centers
RAD      index of accessibility to radial highways
TAX      full-value property-tax rate per $10,000
PTRATIO  pupil-teacher ratio by town
B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
LSTAT    % lower status of the population
MEDV     Median value of owner-occupied homes in $1000's
```

The goal is to build a predictive model for MEDV give information on the other features.

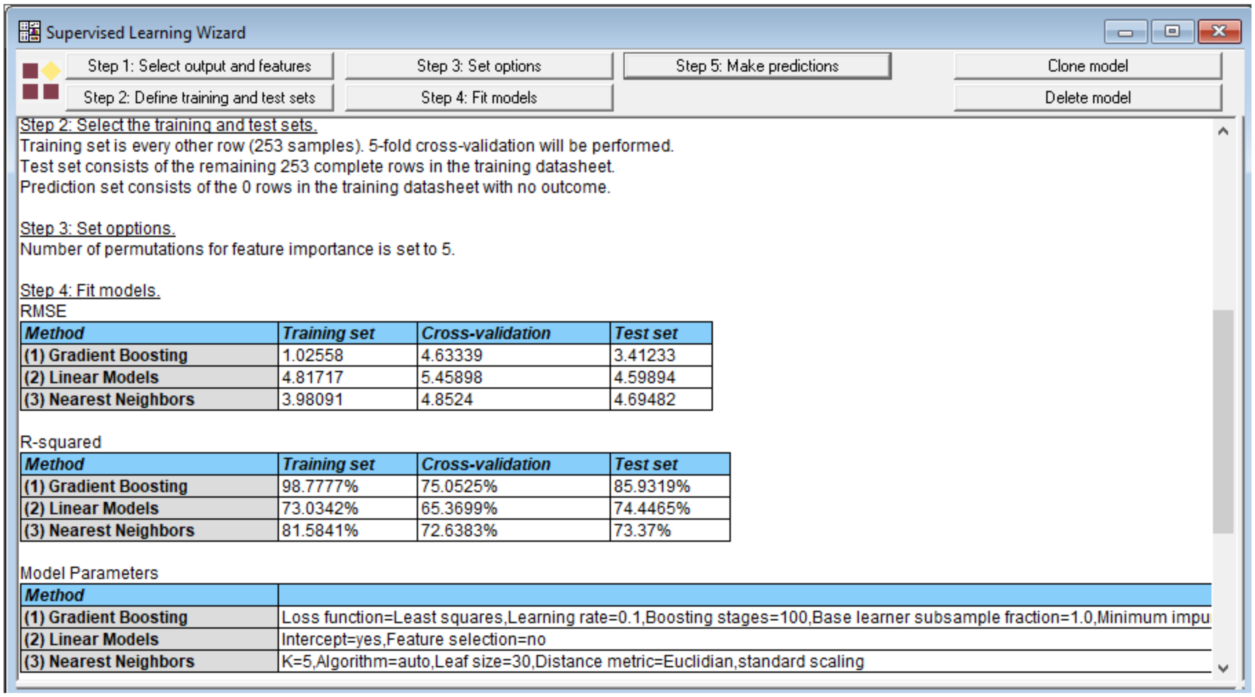
The data input dialog box generated by the wizard in Step 1 is shown below:



Definition of the training and test sets in Step 2 is as follows:



Using 3 different methods to generate predictive models results in the following:



Supervised Learning Wizard

Step 1: Select output and features | Step 3: Set options | Step 5: Make predictions | Clone model

Step 2: Define training and test sets | Step 4: Fit models | Delete model

Step 2: Select the training and test sets.
Training set is every other row (253 samples). 5-fold cross-validation will be performed.
Test set consists of the remaining 253 complete rows in the training datasheet.
Prediction set consists of the 0 rows in the training datasheet with no outcome.

Step 3: Set opptions.
Number of permutations for feature importance is set to 5.

Step 4: Fit models.
RMSE

Method	Training set	Cross-validation	Test set
(1) Gradient Boosting	1.02558	4.63339	3.41233
(2) Linear Models	4.81717	5.45898	4.59894
(3) Nearest Neighbors	3.98091	4.8524	4.69482

R-squared

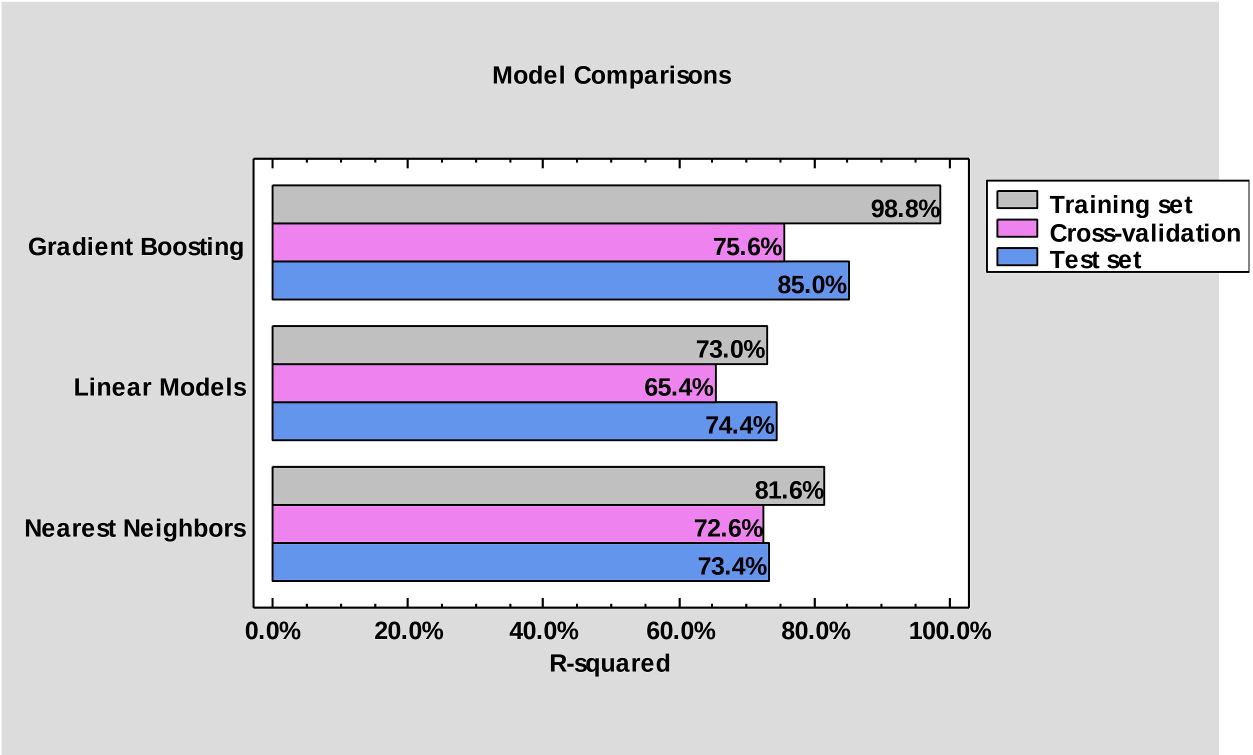
Method	Training set	Cross-validation	Test set
(1) Gradient Boosting	98.7777%	75.0525%	85.9319%
(2) Linear Models	73.0342%	65.3699%	74.4465%
(3) Nearest Neighbors	81.5841%	72.6383%	73.37%

Model Parameters

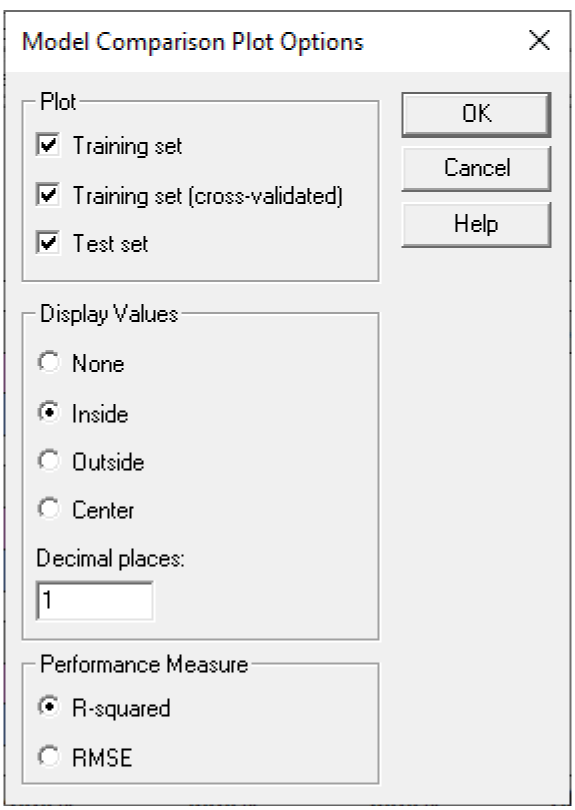
Method	
(1) Gradient Boosting	Loss function=Least squares,Learning rate=0.1,Boosting stages=100,Base learner subsample fraction=1.0,Minimum impu
(2) Linear Models	Intercept=yes,Feature selection=no
(3) Nearest Neighbors	K=5,Algorithm=auto,Leaf size=30,Distance metric=Euclidian,standard scaling

An importance difference between classification and regression models is the performance measure. For regression models, the wizard displays two statistics: the RMSE (root mean squared error) and R-squared. Smaller values of RMSE and larger values of R-squared are preferable. Of the 3 methods used, it appears that gradient boosting does the best.

In the *Model Comparions Plot*, you may choose to plot either the RMSE or R-Squared.



Pane Options



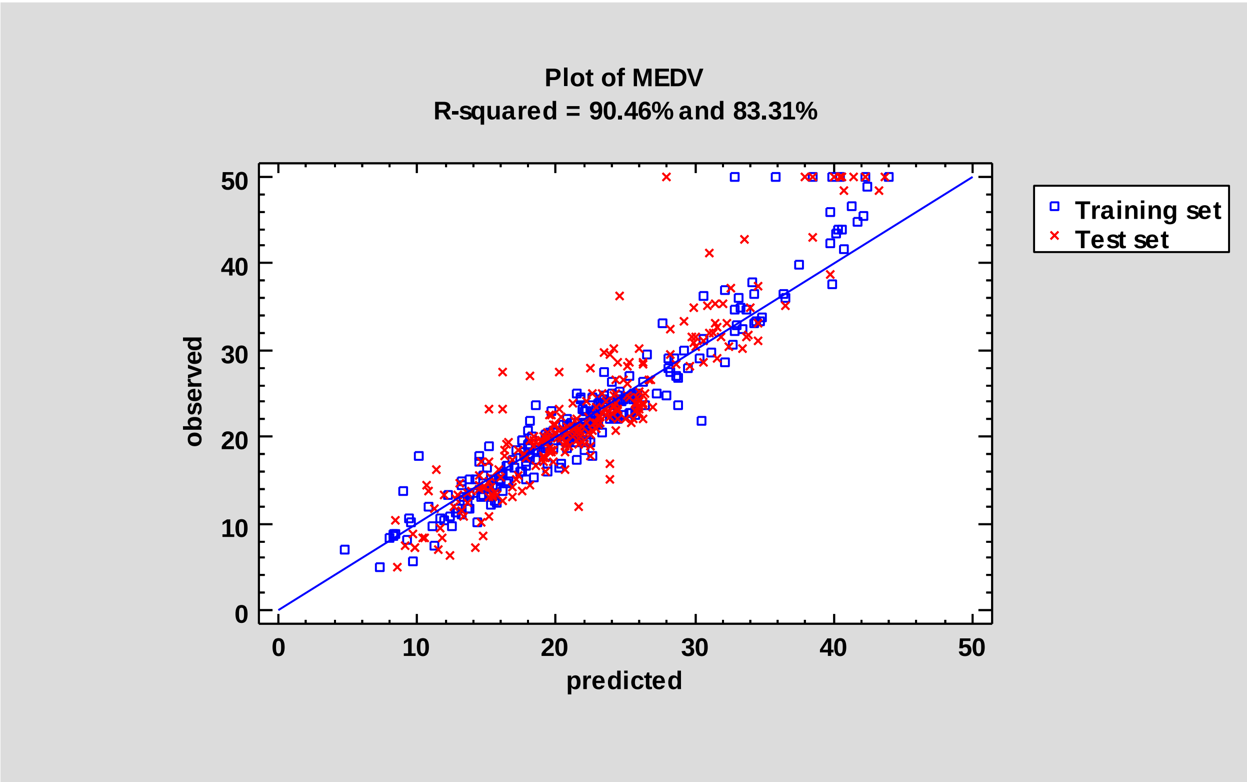
- **Plot** – specify the sets for which the data should be plotted.

- **Display Values** – specify the location with respect to the bars where the values should be plotted and the number of decimal places to display.

- **Performance Measure** – the statistic to plot.

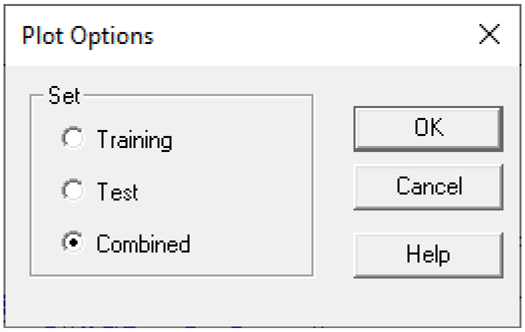
Observed versus Predicted

For a regression model, this plot shows the observed values in the training and/or test sets compared with the ensemble predictions:



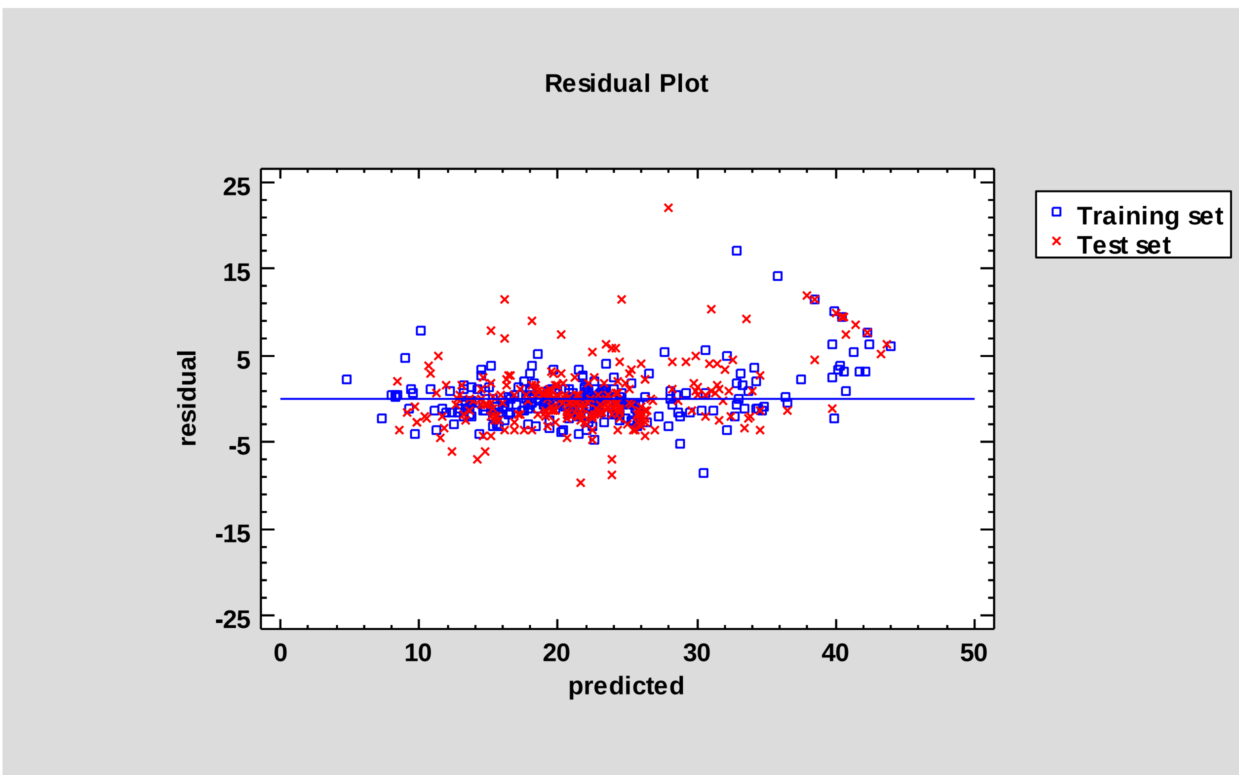
The heading of the plot indicates that the R-squared statistic based on a linear regression between the observed and predicted values is 90.46% for the training set and 83.31% for the test set.

You may use *Pane Options* to select the sets to plot:



Residuals versus Predicted

This plot shows the residual values in the training and/or test sets. Residuals are defined as the observed values minus the ensemble predictions:



References

Breast cancer data:

Wolberg, W.H., Street, W.N. and Mangasarian, O.L., University of Wisconsin, 1995. https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic)

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Housing price data:

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic prices and the demand for clean air', J. Environ. Economics & Management, vol.5, 81-102, 1978. http://lib.stat.cmu.edu/datasets/boston

Muller, A.C. and Guido, S. Introduction to Machine Lerning with Python. O'Reilly, 2017.

Scikit-Learn. https://scikit-learn.org/stable/

