CART 6.0 Feature Matrix

	SE	Pro	ProEX
Enhanced Descriptive Statistics			
Full summary statistics	x	x	x
Brief summary statistics	x	x	x
Stratified summary statistics	x	x	x
Charts and histograms	x	x	x
Improved User Interface			
New setup activity window	x	x	x
Ability to control default settings	x	x	x
Model Building			
Set weighted values for minimum parent/terminal node size	x	x	x
Additional fraction for auto validation	x	x	x
Select/reject predictors directly from variable importance list		x	x
Automated collection of imputation trees **(BATTERY TARGET)**		x	x
Splits			
Forced splits	x	x	x
Automatically test all splitting rules **(BATTERY RULES)**	x	x	x
Linear combination lists		x	x
Constraints and structured trees			x
Cross Validation			
User-controlled cross validation bins	x	x	x
Repeated CV using different random seeds **(BATTERY CVR)**		x	x
Missing Value Analysis			
Automatically add missing value indicators	x	x	x
Allow "missing" as a legal discrete level	x	x	x
Model Evaluation			
Monte carlo testing **(BATTERY MCT)**	x	x	x
"Profit" display: track non-model variables across all nodes		x	x
Train/test consistency: how well do train and test match up across all nodes			x
Hot spot detection: search many trees to find nodes of ultra-high response			x
Additional Summary Reports			
ROC curves and variance of ROC measure	x	x	x
Display learn, test, or pooled results	x	x	x
Gains chart: show perfect model curve	x	x	x
Scoring and Translation			
Multi-tree selection control for scoring and model translation	x	x	x
New model translation formats: Java and PMML	x	x	x
Unsupervised Learning			
Breiman's column scrambler		x	x
Automated Model search: BATTERY			
CV: Varies the number of folds used for cross-validation	x	x	x
MCT: Monte Carlo shuffling of the target variable	x	x	x
MVI: Generates five models with different combinations of missing value settings	x	x	x
RULES: Generates a model for each splitting rule	x	x	x
ATOM: Varies the minimum parent node size		x	x
CVR: Repeats cross validation using different random number seeds		x	x
DEPTH: Varies the depth limit of the tree		x	x
DRAW: Randomly draws the learn sample from the "main" learn sample		x	x
FLIP: Reverses the learn and test samples		x	x
LOVO: Leaves one predictor out of the model each time		x	x
NODES: Varies the maximum number of nodes in the tree		x	x
SAMPLE: Progressively reduces the size of the learn sample		x	x
SUBSAMPLE: Varies the sample size used to find competitor/surrogate splits		x	x
TARGET: Models each variable as a target, using all other variables as predictors		x	x
KEEP: Randomly selects a subset of variables as predictors for each model			x
MINCHILD: Varies the minimum required child node size			x
ONEOFF: Models the target as a function of one predictor at a time			x
PRIORS: Varies the priors for the specified class within a given range			x
SHAVING: Sequentially removes predictors from the model based on their importance			x

info@salford-systems.com • Tel: 619-543-8880 • Fax: 619-543-8888

Enhanced Descriptive Statistics

Our complete set of statistics, including standard summary statistics, quantiles, and detailed tabulations continue to be available for data exploration in a single easy-to-access display. We now also offer an abbreviated version in the traditional *one row per predictor* format.

Also new in CART 6.0 are sub-group statistics based on any segmentation or stratification variable, as well as charts and histograms for visualizing your data.

Improved User Interface

New setup activity window

The activity window offers a quick way to access summary statistics, summary graphs, the model setup dialog, a view of the data, and scoring.

Ability to control default settings

The user can customize and save default settings.

Model Building

Set weighted values for minimum parent/terminal node size

Previous versions of CART have always allowed you to control the size of the smallest terminal node produced. In version 6.0 we now also allow you to control the minimum allowable weighted record counts in any terminal node. A similar control applies to the minimum weighted size a node must have to become a parent node.

Additional fraction for auto validation

Traditionally, CART trees are grown on learn (or training) data and evaluated on test data. Because the test data are used to help select the optimal tree, some practitioners prefer to conduct a further model check by evaluating a performance on a "holdout" portion of the data. We refer to these holdout data as the validation data.

Select/reject predictors directly from variable importance list

Once a model is built, you can easily refine it by managing the variable importance list. Simply highlight the variables you want to keep for the next model and click the "Build New Model" button.

CART Pro and Pro EX provide a higher degree of automation for predictor list refinement (feature extraction) and offer an automated pre-modeling predictor discovery stage. This can be very effective when you are faced with a large number of candidate predictors. In our extensive experiments we have established that automatic predictor discovery frequently improves CART model performance on independent holdout data.

Splits

Forced splits

The user can dictate the splitting variable to be used in the root node or in either of the two child nodes of the root, allowing the user to impose some modest structure on a tree. More specific controls allow the user to specify the split values for both continuous and categorical variables.

Linear combination lists

In CART 6.0 you may specify lists of variables (LC lists) from which any linear combination can be constructed. For example, in a credit risk model you might list credit report variables on one list, core demographics on another list, and current income-related variables on a third list. Time series analysts might create a separate LC list for a variable and all its lagged values. Such LC lists force combinations of variables used in an LC splitter to be of a specific type.

Constraints and structured trees (patent pending)

CART 6.0 offers a powerful mechanism for generating structured trees by allowing you to specify where a variable or group of variables is permitted to appear in the tree. For example, in a marketing model, you might limit consumer-related variables to the top of the tree and product-related variables to the bottom. The resulting model would first split the data into different consumer types and then analyze the product preferences of each group. Constraints could also be used to reflect the natural or causal order of variables within the model structure, or to construct a tree using broad and general predictors at the top and more specific and detailed predictors toward the bottom.

CART allows you to structure your trees in a number of ways. You can specify where a variable can appear in the tree based either on its location in the tree or on the size of the sample arriving at a node. You can also specify as many different regions in the tree as you wish.

Cross Validation

User-controlled cross validation bins

You can create your own partition of the data for the purpose of cross-validation by instructing CART to use a variable you have created for this purpose. This is most useful when there are repeated observations on a behavioral unit such as person or a firm, and it is important to keep all records pertaining to such a unit together (either all records are in the training sample or all in the test sample). User constructed CV bins are also useful in the analysis of time series or geographically correlated data.

Missing Value Analysis

Automatically add missing value indicators

CART has always offered sophisticated high performance missing value handling. In CART 6.0 we introduce a new set of missing value analysis tools for automatic exploration of the optimal handling of your incomplete data. On request, CART 6.0 will automatically add missing value indicator variables (MVIs) to your list of predictors and conduct a variety of analyses using them. MVIs allow formal testing of the core predictive value of knowing that a field is missing. One of the models CART 6.0 will generate for you automatically is a model using only missing value indicators as predictors. In some circumstances such a simple model can be very accurate and it is important to be aware of this predictive power. Other analyses explore the benefits of imposing penalties on variables that are frequently missing.

Allow “missing” as a legal discrete level

For categorical variables, an MVI can be handled either by adding a separate MVI variable or by treating missing as a valid "level." You can experiment to see which works best for your data.

Model Evaluation

"Profit" display: track non-model variables across all nodes

"Profit" variables are any variables the modeler is interested in tracking in the terminal nodes. The "profit" tab on the summary window includes tabular and graphical displays of these variables, showing absolute and average node results and cumulative results based on the ordering of the nodes determined by the original target variable.

Train/test consistency: how well do train and test match up across all nodes

Classic CART trees are evaluated on the basis of overall tree performance. However, many users of CART are more interested in the performance of specific nodes and the degree to which terminal nodes exhibit strongly consistent results across the train and test samples. The TTC report provides new graphical and tabular reports to summarize train/test agreement.

Hot spot detection: search many trees to find nodes of ultra-high response

In many modeling situations, an analyst is looking for "hot spots," regions of modeling space richest in the event of interest. For example, in a fraud detection problem, you might be interested in identifying a set of rules that lead to a high ratio of fraudulent transactions, allowing you to flag future records that fit those rules as almost certainly fraudulent. CART's hot spot detection process is fully automated and is especially effective in processing batteries of models.

Additional Summary Reports

ROC curves and variance of ROC measure

ROC curves have become a preferred way of summarizing the performance of a model, and these are now available for all CART models and ensembles on both train and test data. An estimate of the area under the ROC curve is also produced when cross validation is used to assess model performance.

Display learn, test, or pooled results

Results can be viewed for either the training (learn) data, the test data, or the aggregate created by pooling the learn and test samples.

Gains chart: show perfect model curve

In a gains curve, the performance of a perfect model depends on the balance between the "response" and "non-response" sample sizes. The "perfect model" reference line helps to put the observed gains curve into proper perspective.

Scoring and Translation

Multi-tree selection control for scoring and model translation

In CART 6.0, any tree in the pruning sequence can be used for scoring and model translation.

New model translation formats: Java and PMML

We have added Java and PMML to our existing group of model translation languages. The Predictive Modeling Markup Language (PMML) is a form of XML specifically designed to express the predictive formulas or mechanisms of a data mining model. In CART 6.0 we conform to PMML release 3.0.

Unsupervised Learning

Breiman's column scrambler

We believe that Leo Breiman invented this trick (although we are not entirely sure). We start with the original data and then make a copy. The copy has each of its columns randomly shuffled to destroy its original correlation structure. CART is then used to try to recognize whether a record belongs to the original data or to the shuffled copy. The stronger the correlation structure in the original data, the better CART will do, and the terminal nodes may identify interesting data segments.

Automated Model search: BATTERY

Most modelers conduct a variety of experiments, trying different model control parameters in an effort to find the best settings. This is done for any method that has a number of control settings that can materially affect performance outcomes. In CART 6.0 we have made the process easier yet by packaging our recommended "batteries of models" into batches that the modeler can request with a mouse click.

CV: Runs cross validation with the number of folds set to 5, 10, 20, and 50 CV bins.

MCT: Generates a Monte Carlo test on the significance of the model performance obtained in a given run. The target is first randomly permuted, which destroys any possible dependency of the target on all remaining variables

and should make it very difficult for CART to generate predictive trees. The extent to which trees are still predictive is a measure of potential overfitting.

MVI: Generates five models with different combinations of missing value settings:

1. MVI_No_P – uses regular predictors, missing value indicators, and no missing value penalties
2. No_MVI_No_P – uses regular predictors only (default CART model, no MVIs, no penalties)
3. MVI_only – uses missing value indicators only (no regular predictors, no penalties)
4. MVI_P – uses regular predictors, missing value indicators, and missing value penalties
5. No_MVI_P – uses regular predictors and missing value penalties (no MVIs)

	predictors	MVIs	MV penalties
MVI_No_P	x	x	-
No_MVI_No_P	x	-	-
MVI_Only	-	x	-
MVI_P	x	x	x
No_MVI_P	x	-	x

RULES: Runs each available splitting rule, thus producing six runs for classification or two runs for regression.

ATOM: Varies the required parent node size according to a user-supplied setting.

CVR: Repeats cross validation many times using different random number seeds in order to explore how results might differ as the random partitioning differs.

DEPTH: Varies the depth limit of the tree according to a user-supplied setting.

DRAW: Runs a series of models in which the learn sample is repeatedly drawn (without replacement) from the "main" learn sample. The test sample is not altered. This battery is useful for determining the impact of varying random learn sample selection on ultimate model performance. It is similar in spirit to Battery CVR.

FLIP: Generates two models, reversing the learn/test samples.

LOVO (Leave One Variable Out): Generates a sequence of runs where each run omits one of the variables on the predictor list. Assuming K predictors on the initial list, the battery will produce K models having K-1 predictors each.

NODES: Varies the limit on the tree size in nodes according to a user-supplied setting. It is very similar to battery DEPTH, described above.

SAMPLE: Generates a series of models in which the learn sample is progressively reduced to examine the effect of the learn sample size on error rate. A total of five runs are produced: full train data, ¾ of the train data, ½ of the train data, ¼ of the train data, and ⅛ of the train data.

SUBSAMPLE: Varies the sample size that is used at each node to determine competitor and surrogate splits according to a user-supplied setting. You may list a set of values as well as a repetition factor. Each subsampling size is repeated N times with a different random seed each time.

TARGET: Attempts to model each variable as a target, using all other variables as predictors. The resulting model accuracy indicates the degree of association between the current target and the rest of the variables while the variable importance list shows exactly which variables are involved. This battery can be used to conveniently impute missing values for a group of variables.

KEEP: Randomly selects a specified number of variables from the initial list of predictors (controlled by the KEEP command) and repeats the random selection multiple times. A user has the option of specifying a CORE subset of variables that are always present in each run.

MINCHILD: Varies the required terminal node size according to a user-supplied setting.

ONEOFF: For a given target, this battery develops a one-predictor model for every variable in the KEEP list. Thus if you specify:

```
MODEL Y
KEEP  X1 X2$ X3
```

BATTERY ONEOFF generates three models equivalent to

```
MODEL Y = X1
MODEL Y = X2$
MODEL Y = X3
```

As illustrated, the KEEP list can contain a mixture of continuous and categorical predictors.

PRIORS: Varies the prior probabilities within a specified range in user-supplied increments. By manipulating priors, one could impose different solutions on the sensitivity versus specificity tradeoff as well as control node purity and overall model performance.

SHAVING: Sequentially removes predictors from the model. The following shaving strategies are currently available (assuming K starting variables):

- BOTTOM – removes the least important variables (up to K runs)
- TOP – removes the most important variables (up to K runs)
- ERROR – removes the variable with the least contribution based on the LOVO battery (see above) applied to the current set of variables (up to K(K-1)/2 runs)