SAS Statistics Research and Applications
Paper #2022-02

Introducing Bayesian Additive Regression Trees in SAS®Viya®

Michael Lamm, SAS Institute Inc., Cary, NC

Bayesian additive regression trees (BART) are a flexible regression technique that uses a sum-of-trees ensemble to approximate the conditional mean of a response variable. As a predictive modeling tool, BART models have many desirable features. In particular, the default BART model fit tends to generalize well, typically requires little or no hyperparameter tuning, and enables you to assess uncertainty in the model predictions on the basis of posterior sample variability. In this paper, we provide an overview of BART models, compare them to other sum-of-trees ensembles, discuss computational considerations, and demonstrate how you can train and score BART models by using the new BART procedure and Bayesian Additive Regression Trees action set in SAS®Visual Statistics software.

Introduction

Modeling an outcome, or target, variable is a common task for statisticians and data scientists. Bayesian additive regression trees (BART) provide an approach to outcome modeling that performs well when used as a purely predictive technique, and they can also be successfully used in causal analysis as a tool for studying the relationship between an outcome and treatment variables.

As a purely predictive model, the BART model has many desirable features (Chipman, George, and McCulloch 2010). By using a sum-of-trees ensemble, BART models inherit many of the desirable properties of tree-based modeling techniques: namely, they are nonparametric, they can incorporate a mix of continuous and categorical predictors, they do not require the explicit modeling of interaction terms, and they can naturally handle missing values in the predictor variables. BART models also have a number of desirable properties that are more specific to the models' Bayesian nature. In particular, these models use a prior distribution that is typically effective at limiting overfitting to the training data. The default specifications of a BART prior tend to perform well in terms of model fit, and this performance does not change significantly if there are small changes in the prior parameters. For this reason, a popular feature of these models is that they usually do not require hyperparameter tuning. Moreover, you can assess uncertainty in the BART model predictions from a Bayesian perspective by examining variability in the predictions from the samples of the ensemble through the use of credible intervals. In addition to performing well as purely predictive models, BART models have also formed the basis of top-performing methods in causal inference competitions (Dorie et al. 2019).

Starting in the 2022.1.1 release of SAS Visual Statistics, you can fit BART models of normally distributed response variables either by using the BART procedure or by using the `bartGauss` action in the Bayesian Additive Regression Trees (`bart`) action set. A BART model that you train using either the procedure or the action can be saved as an analytic store. With the analytic store, you can score new observations by using the ASTORE procedure, the `bartScore` action in the `bart` action set, or the `score` action in the `astore` action set. In addition to scoring new observations, you can use the `bartScoreMargin` action to compute predictive margins by using a saved model. When the appropriate set of causal assumptions are satisfied, the computation of predictive margins corresponds to a regression-based approach to estimating potential outcomes means in a counterfactual framework.

This paper provides an introduction to using these new BART-based modeling tools and is organized as follows. First, the section "BART Model for Normal Data" provides an overview of BART models. The section "BART Models in SAS Visual Statistics" discusses the new implementation of BART modeling in SAS Visual Statistics, computational considerations, and how BART models compare to other tree-based modeling methods in SAS®Viya®. "Example 1: Comparing Model Fit" illustrates how you can train a BART model and use it to score new data, and it compares the BART model performance to that of other tree-based methods. "Example 2: Computing Predictive Margins" shows how to use BART models to estimate causal effects.

BART Model for Normal Data

For a response y that is assumed to follow a normal distribution,

$$y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$$

a Bayesian additive regression trees (BART) model uses a sum-of-trees ensemble to approximate $f(\mathbf{x})$, the conditional mean of y given the vector of predictors $\mathbf{x}$.

As described by Chipman, George, and McCulloch (2010), the development of the BART model was motivated by the performance of sum-of-trees ensembles, and in particular by the weak learner property that is used in gradient boosting algorithms. BART models take a Bayesian approach to creating an ensemble in which the individual trees are subject to a weak learner property that limits the contribution of any one tree to the ensemble prediction. For these models, the weak learner property is created through the use of a well-chosen regularization prior. A BART model is fit by using a Bayesian backfitting Markov chain Monte Carlo (MCMC) algorithm. This algorithm uses a form of Gibbs sampling to produce samples of the sum-of-trees ensemble by successively modifying each tree in the ensemble. The final fitted BART model consists of many samples of the ensemble that you save for prediction.

The subsections that follow provide details about the components of a BART model. "BART Prior Distribution" provides a summary of the BART prior distribution and how the prior creates the weak learner property of the ensemble. "Sampling the Posterior" summarizes the process by which you obtain the posterior samples of the sum-of-trees ensemble.

BART Prior Distribution

The following definitions describe components of the BART prior distribution:

- m denotes the number of trees in the ensemble.
- T_j denotes the structure of the jth tree.
- M_j denotes the set of terminal node, or leaf, parameters in the jth tree.
- b_j denotes the number of terminal nodes in the jth tree.
- $\mu_{\ell j}$ denotes the terminal node parameter associated with the ℓth terminal node in the jth tree.

Prior distributions of this form assume independence between the tree structures and conditionally independent terminal node parameters, given the tree structure. The priors for the tree structure $p\left(T_j\right)$, the terminal node parameters $p\left(\mu_{\ell j} \mid T_j\right)$, and the variance term $p\left(\sigma^2\right)$ are chosen to form a regularization prior that limits the contribution of a single tree to the model fit. The use of a regularization prior enforces the so-called weak learner property on the individual trees (Chipman, George, and McCulloch 2010). In particular, to create the weak learner property, the prior is chosen to place greater prior probability on simpler tree structures and to applying shrinkage to the leaf parameters through the choice of their prior mean and variance.

The tree prior places greater prior probability on simpler tree structures, primarily through how it determines the prior probability that a node will be split. Based on the recommendations of Chipman, George, and McCulloch (1998), the probability of a node being split is determined by a power function of the node's depth from the root node. This function depends on the choice of a base-splitting probability and the function's degree, which is chosen to assign higher prior probability to smaller trees. For splitting a node, the distribution for sampling the splitting rule is determined by two components: first, a distribution for selecting which variable to use for splitting the node; and second, a distribution for selecting the splitting criterion, given the selected variable. The choice of variable to split on and the splitting rule are both sampled from a uniform distribution.

The leaf parameters $p\left(\mu_{\ell j} \mid T_j\right)$ use a normal prior that has mean μ_μ and variance σ_μ^2. Given the assumed form of the BART prior, the conditional mean of the response, given the predictor variables, $f(\mathbf{x}) = E\left[Y \mid \mathbf{X}\right]$, is the sum of m independent and identically distributed (iid) normal random variables and has mean $m\mu_\mu$ and variance $m\sigma_\mu^2$. As described by Chipman, George, and McCulloch (2010), the parameters μ_μ and σ_μ^2 are chosen to assign high probability to values within the range of the training data.

For a specified value of a constant $k > 0$, the values μ_μ and σ_μ^2 are determined by considering a linear transformation of the response variable that results in a range of $(-0.5, 0.5)$. After such a transformation, the leaf parameter prior is given by a normal prior that has mean 0 and variance $\frac{0.5}{k\sqrt{m}}$. As noted by Chipman, George, and McCulloch (2010), these choices of parameter values limit the contribution of a single tree and result in a decreasing prior variance for $\mu_{\ell j}$ as the number of trees in the ensemble m increases.

The prior for the variance $p\left(\sigma^2\right)$ is chosen to follow a scaled inverse chi-square distribution that has degrees of freedom υ and scale parameter λ. As described by Chipman, George, and McCulloch (2010), the choice of parameters υ and λ can be informed by the training data.

Sampling the Posterior

Given a training data set, a Bayesian backfitting MCMC algorithm is used to draw samples from the posterior distribution (Chipman, George, and McCulloch 2010). This algorithm is a form of Gibbs sampling that can be described as follows. Let $\mathbf{y}$ denote the vector of training data response values, and let $\mathbf{T}_{-j}$ denote the tree structures for the $m-1$ trees in the ensemble, excluding the jth tree. Similarly, let $\mathbf{M}_{-j}$ denote the set of all leaf parameters, excluding the jth tree parameters. Each iteration of the sampling algorithms takes m consecutive draws, $j = 1, \ldots, m$, from

$$T_j, M_j \mid \mathbf{T}_{-j}, \mathbf{M}_{-j}, \mathbf{y}, \sigma^2$$

followed by a draw from

$$\sigma^2 \mid T_1, M_1, \ldots, T_m, M_m, \mathbf{y}$$

As described by Chipman, George, and McCulloch (2010), the draws for the tree structures T_j and leaf parameters M_j are simplified by observing that their conditional distribution depends on $\mathbf{T}_{-j}$, $\mathbf{M}_{-j}$, and $\mathbf{y}$ only through the partial residuals for the fit, excluding the jth tree. Moreover, the choice of the conjugate normal prior for the leaf parameters allows for the sampling of T_j and M_j to be carried out in two steps. Let $\mathbf{r}_j$ denote the partial residuals that are based on the fit, excluding the jth tree. You then obtain the samples for T_j and M_j by taking successive draws from

$$\begin{gathered} T_j \mid \mathbf{r}_j, \sigma^2 \\ M_j \mid T_j, \mathbf{r}_j, \sigma^2 \end{gathered}$$

The draws for the tree structure T_j are obtained by using the Metropolis-Hastings sampling algorithm described by Chipman, George, and McCulloch (1998). This algorithm considers sampling from four operations that modify the tree structure—namely, pruning a pair of terminal nodes, splitting a terminal node, changing the splitting rule of an internal node, or swapping a splitting rule between a parent node and a child node. As described in Pratola et al. (2014), although the changing and swapping operations significantly improve the fit of single-tree models, these operations have less impact on the sum-of-trees-ensemble BART models. Limiting the tree modifications to only pruning and splitting operations was observed to have a limited effect on model fit while providing a significant computational benefit. For BART models that are fit in SAS Visual Statistics, only the splitting and pruning operations are considered, and for splitting operations a uniform distribution is used to sample both the splitting variable and the splitting criterion.

Let T_{new} denote the sampled modification to the tree structure T_j, and let $q(\cdot \mid \cdot)$ denote the proposal density. The proposed modification is then accepted with the probability

$$\alpha = \min\left\{ \frac{q(T_j \mid T_{\text{new}}) p\left(\mathbf{r}_j \mid T_{\text{new}}, \sigma^2\right) p\left(T_{\text{new}}\right)}{q\left(T_{\text{new}} \mid T_j\right) p\left(\mathbf{r}_j \mid T_j, \sigma^2\right) p\left(T_j\right)}, 1 \right\}$$

After the proposed operation is accepted or rejected, a draw is taken for the set of leaf parameters M_j, and the partial residuals $\mathbf{r}_{j+1}$ are then updated for sampling the $j+1$th tree structure T_{j+1} and the $j+1$th set of leaf parameters M_{j+1}.

To begin the sampling, the ensemble is initialized with m single-node trees, or stumps. After an initial burn-in period, the main simulation loop is conducted, from which samples of the ensemble are saved for prediction. You can apply a thinning rate to the main simulation and save so that only portion of the simulated samples are saved for prediction and for calculating posterior statistics. After the main simulation loop finishes, you can obtain predictions from the fitted BART model by averaging the predictions from each sample of the ensemble that was saved, and you can assess uncertainty in the model predictions on the basis of the posterior sample variability.

BART Models in SAS Visual Statistics

Starting in the 2022.1.1 release of SAS Visual Statistics, you can fit BART models of normally distributed response data either by using the BART procedure or by using the `bartGauss` action in the `bart` action set. When you use either the procedure or the action, the default BART model is trained by using a 200-tree ensemble, 100 burn-in iterations, and a main simulation loop of 1,000 iterations performed without thinning. The default prior that is used to train a BART model in SAS Visual Statistics largely follows the recommendations of Chipman, George, and McCulloch (2010). For the reasons described by Pratola et al. (2014), only the splitting and pruning operations are considered when you sample tree-modifying operations. For more information about the options that you can use to train a BART model in SAS Visual Statistics, see the documentation of PROC BART and the `bart` action set.

You can use an analytic store save a BART model that is fit using either the BART procedure or the `bartGauss` action. You can use the saved model to score new observations by using the ASTORE procedure, the `bartScore` action in the `bart` action set, or the `score` action in the `astore` action set. When scoring new observations through any of these methods, you can also request equal-tail credible intervals for the average prediction and the individual predictions from each saved sample of the ensemble. An example of using a saved BART model to score new observations is provided in "Example 1: Comparing Model Fit."

In addition to scoring new observations, you can also use a saved BART model to compute predictive margins by using the `bartScoreMargin` action. The predictive margins are obtained by fixing the value of one or more of the input variables and averaging the predicted values over the distribution of the covariates in an input data table. In cases where the proper causal assumptions hold (Hernán and Robins 2020), the predictive margin can correspond to an estimate of a potential outcome mean, and a difference in predictive margins can represent a causal effect estimate. In general, predictive margins can always be interpreted as average predictions that are marginal to the covariate distribution in a scoring data set. An example of using the `bartScoreMargin` action to compute predictive margins that are assumed to have a causal interpretation is provided in "Example 2: Computing Predictive Margins."

The remainder of this section discusses important computational considerations of working with BART models and how BART models compare to other predictive models that use tree-based ensembles.

Computational Considerations

Training a BART model is a computationally intensive process. For each posterior sample, the modifications to each tree are sampled sequentially, and each tree modification requires at least one pass through the data. When you train a BART model in SAS Visual Statistics, there are options that you can use to attempt to minimize the time it takes to train the model. However, these options can require a significant amount of memory and might not be suitable for very large data sets.

One option is to ensure that the training data are stored in memory when you train the model. Storing the training data in memory can reduce the time it takes to access data during the sampling of tree modifications. This option requires memory allocations that are in total proportional to the number of observations times the size of an input row. By default, the data are not stored in memory during model training.

A second option is to store a mapping of each observation to terminal nodes, or leaves, in memory during model training. Storing a mapping of each observation to the terminal nodes that it corresponds to removes the need to route an observation through a tree when you are sampling updates to a tree's structure and its leaf parameters. This option requires memory allocations that are in total proportional to the number of observations times the number of trees. By default, the observation-to-leaf mapping is not stored in memory during model training.

When you train a BART model on a cluster of machines, you can run multiple chains and divide the MCMC samples across worker nodes. In addition to enabling multiple chains to be run in parallel, distributed mode can reduce the time it takes for communications across the cluster of machines. This mode requires distributing the training data to every worker so that each worker running a separate chain has a full copy of the training data. This option does not apply to single-machine mode. By default, distributed mode is used when the model is trained on a cluster of machines. You can also specify the number of distributed chains to use, up to the number of available worker nodes in the cluster, or you can specify that a single chain be used with each worker node that is assigned a portion of the training data. Note that the fit of a BART model depends on the number of chains that are run.

Scoring new observations by using a saved BART model requires routing observations through each tree for all the saved samples of the ensemble. For the default BART model, this is 200,000 trees in total, so scoring new observations can also be time-consuming for large data sets. When you are interested in using a BART model only for prediction and not for assessing uncertainty in the model predictions by using equal-tail credible intervals, consider reducing the number of samples of the ensemble that are saved for prediction.

BART Models Compared to Gradient Boosting and Random Forest Models

Gradient boosting models (Friedman 2001) and random forest models (Breiman 1996, 2001) are other popular predictive models that use ensembles of decisions trees. In SAS Viya, you can use the GRADBOOST and FOREST procedures in SAS®Viya: Machine Learning software to fit these types of models. Although BART models, gradient boosting models, and random forest models all use tree-based ensembles, they differ substantially in terms of what you must specify to train the model, how you train the model, and how you obtain predictions from the model.

When you train either a gradient boosting or random forest model, the fitted model consists of a single ensemble of trees. By contrast, the fitted BART model consists of the posterior samples of a sum-of-trees ensemble that you saved for prediction. Moreover, you produce the samples of the sum-of-trees ensemble in a BART model by successively sampling modifications to each tree in the ensemble. This differs from how you train gradient boosting or random forest models, for which trees are fitted and added to the ensemble individually. For gradient boosting models, the trees are added sequentially, and the target value that you use to train a new tree is obtained by evaluating the gradient of a loss function, given the current ensemble fit. For a random forest model, you train each tree on a subset of the training data that is taken with replacement, and the individual trees can be trained in parallel.

To control the process of training the individual trees in either a gradient boosting or random forest model, you must specify a number of hyperparameters. For example, when using either model, you must prespecify the maximum depth of a tree. You can tune these hyperparameters manually, or you can use an automated process to search for the best configuration of parameter values. Tuning the hyperparameters is important, because gradient boosting and random forest models build trees in a greedy fashion, and the hyperparameters are needed to limit overfitting to the training data by applying various forms of regularization. In contrast, BART models do not use hyperparameters to impose regularization by setting constraints on tree structure such as the maximum tree depth. Instead, BART models use a stochastic search algorithm and a regularization prior to limit overfitting of the training data. BART prior parameters typically do not require hyperparameter tuning, because the default BART model tends to perform well in terms of model fit, and this performance does not change significantly if there are small changes in the prior parameters.

Obtaining predictions from all three types of models requires routing new observations through all the trees that make up the fitted model. For a gradient boosting model, the predictions from the individual trees are combined by taking their summation and multiplying by the model learning rate, or step size. For a random forest model, the predictions from each tree in the ensemble are averaged to obtain the overall prediction. For a BART model, a prediction from each sample of the ensemble is obtained by summing the predictions from each tree in that sample. By averaging the predictions from all the saved posterior samples of the ensemble, you then obtain the overall model prediction. Using a BART model to score new observations is a more time-consuming process because the model consists of many samples of the ensemble; this differs from using fitted gradient boosting and random forest models, which consist of a single ensemble. Note that many commonly used interpretable model-agnostic explanation methods score data by using a fitted model. For this reason, computing these quantities for BART models can be time-consuming. However, a benefit of using the larger BART models is that you can naturally assess uncertainty in the model predictions from a Bayesian perspective by examining the variability of the posterior samples.

For more information about comparing BART models to gradient boosting and random forest models, see “Example 1: Comparing Model Fit.”

Example 1: Comparing Model Fit

This example uses simulated data to compare the fit of predictive models that are produced by using different tree-based methods. In particular, it compares tree-based ensembles that are fit by using the BART, GRADBOOST, and FOREST procedures. All three procedures were developed specifically for SAS Viya and require the input data to be in a SAS®Cloud Analytic Services (CAS) data table accessible in your CAS session.

The following statements assume that your CAS engine libref is named mycas and generate the data table inputData, which consists of 10,000 observations on a continuous response variable (y) and 40 continuous variables (x1–x40):

```
data mycas.inputData / single =yes;
   drop  j w1-w40;

   array x{40};
   array w{40};
   call streaminit(6524);
   pi=constant("pi");

   do i=1 to 10000;
      u = rand("Uniform");
      do j=1 to dim(x);
         w{j} = rand("Uniform");
         x{j} = (w{j} + u)/2;
      end;

      f1 = sin(pi * x1 * x2 );
      f2 = (x3-0.5)**2;
      f3 = x4;
      f4 = x5;
      fb = 10*f1 +20*f2+10*f3+5*f4;

      y = fb +  rand("Normal");
      output;
   end;

run;
```

For the simulation, only the variables x1–x5 affect the outcome y, and the variables x1–x40 are all positively correlated.

The following statements fit a Bayesian additive regression trees (BART) model to these data:

```
   proc bart data=mycas.inputData seed=9181 trainInMem mapInMem;
      model y = x1-x40;
      store mycas.bartFit;
   run;
```

PROC BART uses traditional modeling syntax, and the MODEL statement is required. You specify the response variable and predictor variables in this statement. In this example, because the input data are not large, the TRAININMEM and MAPINMEM options are specified to store the data and elements of the model in memory when the model is trained. The mycas.bartFit analytic store contains a representation of the model that you can use to score new observations. The fit statistics for the fitted model are displayed in the “Fit Statistics” table in Figure 1.

Figure 1: BART Model Fit Statistics

The BART Procedure

Fit Statistics	
Average Square Error	0.97290

Note that for this example, the model is fit in single-machine mode and all the posterior samples are generated from a single chain. When you train a BART model on a cluster of machines, the default is to run multiple parallel chains and divide the MCMC samples across the worker nodes. Because of the random sampling involved in generating the posterior samples, the fit of a BART model depends on how many chains are run. For reproducibility, you can specify the number of parallel chains to use, up to the number of available worker nodes, or you can specify that a single chain be run, with each worker assigned only a portion of the training data.

For comparison, the following programs fit models to the same data table by using the GRADBOOST and FOREST procedures, respectively:

```
proc gradboost data=mycas.inputData seed=9181;
   autotune;
   target y;
   input x1-x40;
   store mycas.gbFit;
run;

proc forest data=mycas.inputData seed=9181;
   autotune;
   target y;
   input x1-x40;
   store mycas.rfFit;
run;
```

When you use the GRADBOOST and FOREST procedures, the response, or target, variable is specified in the TARGET statement and the predictor variables are specified in the INPUT statement. Training gradient boosting and random forest models can involve the selection of many hyperparameters. In this example, the AUTOTUNE statement is specified in order to perform an automated search for a good combination of hyperparameter values. The models that the GRADBOOST and FOREST procedures fit are saved in the analytic stores mycas.gbFit and mycas.rfFit, respectively. The fit statistics for the gradient boosting and random forest models are displayed in Figure 2 and Figure 3, respectively.

Figure 2: Gradient Boosting Model Fit Statistics

Fit Statistics	
Number of Trees	**Training Average Square Error**
1	24.283
2	21.233
3	18.608
4	16.379
5	14.478
6	12.789
.	.
.	.
.	.
.	.
145	0.926
146	0.925
147	0.924
148	0.922
149	0.921
150	0.919

Figure 3: Random Forest Model Fit Statistics

Fit Statistics		
Number of Trees	**OOB Average Square Error**	**Training Average Square Error**
1	1.975	1.975
2	1.236	1.236
3	0.977	0.977
4	0.849	0.849
5	0.779	0.779
6	0.724	0.724
.	.	.
.	.	.
.	.	.
.	.	.
80	0.522	0.522
81	0.522	0.522
82	0.522	0.522
83	0.522	0.522
84	0.522	0.522
85	0.522	0.522

Note that the average square error (ASE) of the training data does not accurately indicate how the models will generalize to new data because of the possibility of overfitting the model to the training data. To obtain a fair comparison of the fits, a new data set of test observations is simulated in the following program, using the same data-generating process that produced the training data:

```
data mycas.toScoreData / single =yes;
   drop  j w1-w40;
   array x{40};
   array w{40};
   call streaminit(1972);
   pi=constant("pi");
   do i=1 to 1000;
      u = rand("Uniform");
      do j=1 to dim(x);
         w{j} = rand("Uniform");
         x{j} = (w{j} + u)/2;
      end;
      f1 = sin(pi * x1 * x2 );
      f2 = (x3-0.5)**2;
      f3 = x4;
      f4 = x5;
      fb = 10*f1 +20*f2+10*f3+5*f4;
      y = fb +  rand("Normal");
      output;
   end;
run;
```

The following statements use PROC CAS to invoke the `bartScore` action to score the new data by using the saved BART model, and to invoke the `score` action in the `astore` action set to score the new data by using the saved gradient boosting and random forest models. Note that you could also use the `score` action to obtain predictions by using the saved BART model. After you score the data by using the gradient boosting and random forest models, the `alterTable` action in the `table` action set is used to change the name of the variable that contains the predicted response values and to remove a column from the output data tables.

```
proc cas;
   action bart.bartScore /
      table    = {name="toScoreData"}
      restore  = {name="bartFit"}
      casOut   = {name="scoredBARTData", replace=TRUE}
      pred     = "bartPred",
      copyVars = {"i" "y"};
   run;

   action astore.score /
      table       = {name="toScoreData"}
      rstore      = {name="gbFit"}
      casOut      = {name="scoredGBData", replace=TRUE}
      copyVars    = {"i" "y"};
   run;

   action table.alterTable /
      name = "scoredGBData"
      columns = {{ name="P_y" rename="gbPred"}
                 { name="_WARN_" drop=TRUE}};
   run;

   action astore.score /
      table       = {name="toScoreData"}
      rstore      = {name="rfFit"}
      casOut      = {name="scoredRFData", replace=TRUE}
      copyVars    = {"i" "y"};
   run;

   action table.alterTable /
      name = "scoredRFData"
      columns = {{ name="P_y" rename="rfPred"}
                 { name="_WARN_" drop=TRUE}};
   run;
quit;
```

The following statements combine the three data tables into a SAS data set and compute the test data ASE for the three different models:

```
data fitCheck;
   merge mycas.scoredBARTData mycas.scoredGBData
         mycas.scoredRFData;
   by i;
   bartSqErr = (y - bartPred)**2;
   gbSqErr   = (y - gbPred)**2;
   rfSqErr   = (y - rfPred)**2;
run;

proc means data=fitCheck mean;
   var bartSqErr gbSqErr rfSqErr;
run;
```

Figure 4 shows the ASE for the test data. In this example, the BART model provides the best fit for the test data. Moreover, the ASE values from the BART model for the training and test data are both close to one, which is the expected value from an oracle model. For these data, the BART model also has the smallest absolute difference between training and test data ASE among the three models.

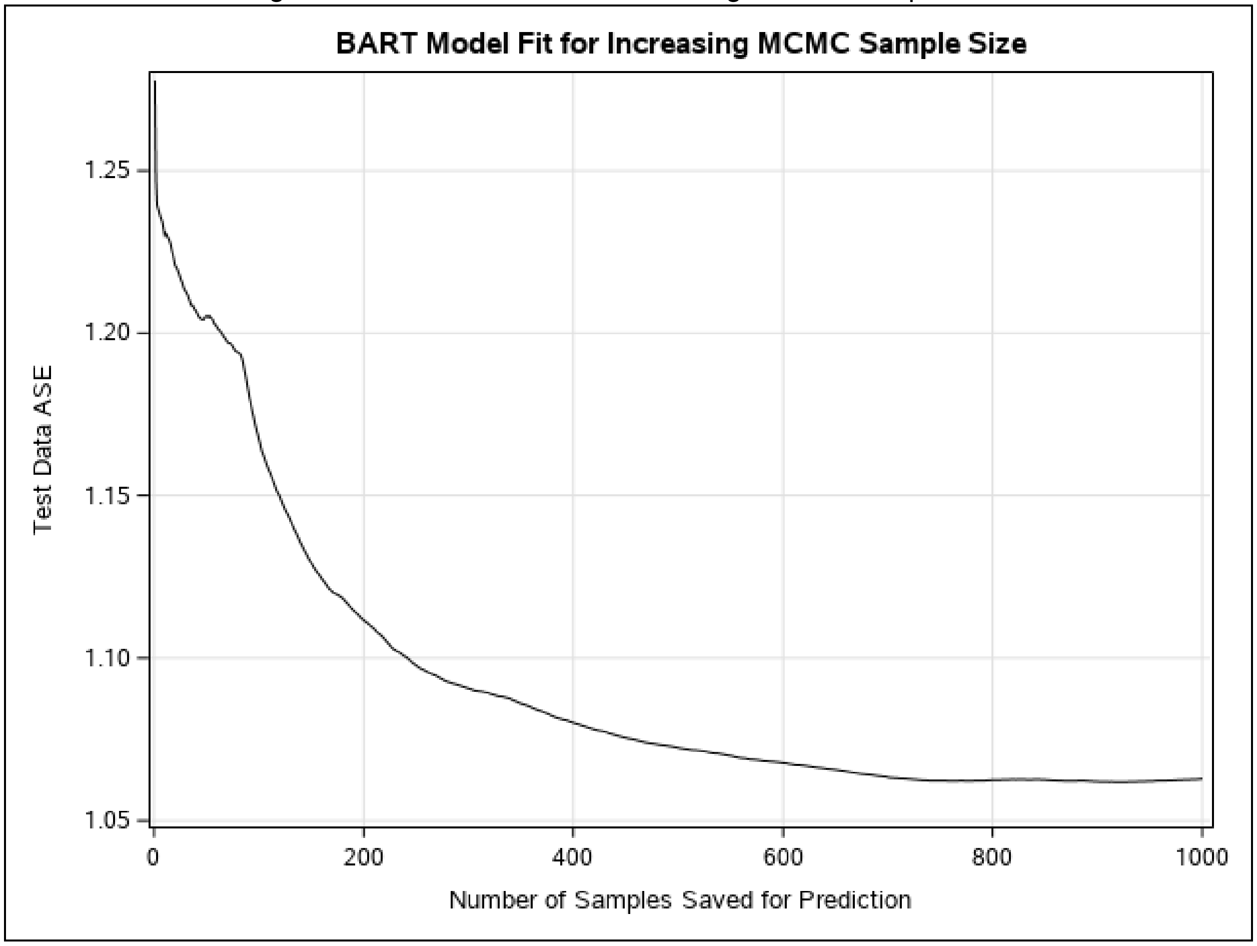
Figure 4: ASE for the Scored Data

The MEANS Procedure

Variable	Mean
bartSqErr	1.0628698
gbSqErr	1.2037748
rfSqErr	1.5316406

In this example, the fitted BART model is used only to obtain point estimates for predicted values, and equal-tail credible intervals are not used to assess uncertainty in the predictions. Although a large number of posterior samples are required if you want to assess uncertainty in BART model predictions, a BART model that saves fewer samples of the ensemble for prediction can in some cases provide a comparable fit, as measured by statistics like the ASE. For example, Figure 5 shows how the ASE for the test data changes as the number of samples of the ensemble that are saved for prediction increases.

Figure 5: Test Data ASE for Increasing MCMC Sample Size



This example demonstrates how you use BART models for causal effect estimation, in particular through the use of the `bartScoreMargin` action. It is important to note that the predictive margins computed by the `bartScoreMargin` action do not always have a causal interpretation. In general, the predictive margins are interpreted as covariate-adjusted marginal means that are obtained by fixing a covariate level and averaging the predictions over the distribution of the remaining covariates in an input data set. Attributing a causal interpretation to these estimates is appropriate only when the data that you are analyzing and the model that you are using satisfy the required causal assumptions. When you define a causal effect within a counterfactual framework, the required assumptions would include that the data capture a well-defined causal question. You must also assume that the variables that are not fixed in the predictive margin computations represent a valid adjustment set for studying the causal relationship between the outcome and the designated treatment variable that you intervene on. When these assumptions are satisfied, the predictive margin computations correspond to an approach to effect estimation that is based on regression, or response surface methodology. For more information about the definition of causal effects in a counterfactual framework and the required assumptions, see the "Overview of Causal Analysis" chapter in the SAS/STAT®software documentation and references therein.

Example 2: Computing Predictive Margins

The data that this example analyzes are contained in the SmokingWeight data set that is used in documentation examples for the CAUSALTRT procedure in SAS/STAT. The data are a subset of the NHANES 1 Epidemiologic Follow-Up Study (NHEFS) used by Hernán and Robins (2020). Like the PROC CAUSALTRT documentation examples, this example investigates the effect that quitting smoking has on individuals' change in weight over a 10-year period. In analyzing these data, it is important to take into account how observations that have missing values in the predictor values are handled. Because PROC CAUSALTRT uses generalized linear models to fit models for the treatment and outcome variables, it excludes all observations that have missing values for a covariate. This contrasts with the tree-based models that are fit by the BART procedure, which can incorporate rules for handling missing values into node-splitting rules and therefore provide predictions for all observations regardless of whether a covariate value is missing. In this example, to compare more directly to the estimates obtained by using PROC CAUSALTRT, in creating a CAS data table from the SmokingWeight data set, observations that would not be used by PROC CAUSALTRT are removed.

To study the effect of interest, this example fits an outcome model by using PROC BART, saves the model in an analytic store, and then uses PROC CAS to invoke the `bartScoreMargin` action to compute the difference in predictive margins defined by different levels of the treatment variable Quit.

To begin, the following statements invoke the BART procedure to fit a model for the outcome variable Change:

```
proc bart data=mycas.smokingWeight seed=1976;
   class Sex Race Education Exercise Activity Quit;
   model Change = Quit Sex Education Exercise Activity
                  YearsSmoke PerDay Age;
   store mycas.store1;
run;
```

In this example, the outcome model includes a number of categorical predictors, including the designated treatment variable Quit, that are listed in the CLASS statement. The fitted model is saved in an analytic store named mycas.store1.

To compute the predictive margins, the following statements first use PROC CAS to invoke the `bartScoreMargin` action. The `margins` parameter defines predictive margins that intervene on the input variables. Observations in an input data table are scored according to the specified interventions, and the predictive margins are obtained by taking the mean of these predictions. In this case, two predictive margins are defined for interventions on the designated treatment variable Quit. The predictive margin named "Cessation" sets the value of Quit to 1, which corresponds to a subject quitting smoking, and the predictive margin named "No Cessation" sets the value of Quit to 0, which corresponds to a subject not quitting. These names are used in the `differences` parameter to request the difference between these predictive margins.

```
proc cas;
   action bart.bartScoreMargin /
   table = {name="smokingWeight"}
   restore = {name="store1"}
   margins= {
      { name="Cessation",   at={{var="Quit" value="1"}}}
      { name="No Cessation",at={{var="Quit" value="0"}}}
   }
   differences = {
      { label="Cessation Difference"
        refMargin="No Cessation"
        evtMargin="Cessation"}
   };
   run;
quit;
```

If you make the proper causal assumptions, then because the data table that this example uses to compute the predictive margin is the same as the training data, the predictive margin estimates would correspond to potential outcome mean estimates and their difference would provide an estimate of the average treatment effect (ATE). To estimate a conditional effect within some subpopulation, a data table other than the training data can be used to compute the predictive margins. In particular, to estimate the average treatment effect for the treated, or ATT, you can compute the predictive margins by using only the observations that are in the designated treatment condition. In general, you can use the `bartScoreMargin` action to score predictive margins that intervene on more than one variable, and the predictions can be computed for an input data table other than the training data.

Figure 6 shows the predictive margin estimates, their difference, and the 95% equal-tail credible intervals for this example. The difference estimate of about 3.42 kilograms is comparable to the ATE estimate that is obtained in the PROC CAUSALTRT documentation. If you are interested in a function of the predictive margins other than the difference, you can use the `casOut` parameter to create an output data table on the server that contains the predictive margin estimate from each sample of the ensemble saved for prediction. You can then apply the function of interest to these output data and compute the corresponding credible interval.

Figure 6: Quit Predictive Margins and Their Difference

Results from bart.bartScoreMargin

Predictive Margins

Description	**Estimate**	**95% Equal-Tail Interval**	
Cessation	5.19341	4.41642	5.95147
No Cessation	1.77156	1.35216	2.19989

As discussed in Dorie et al. (2019), a number of modifications to the BART models have been proposed when they are used to estimate causal effects. A simple modification inspired by Hahn, Murray, and Carvalho (2020) is to add to the BART model a covariate that contains predicted propensity score values; the propensity score is the conditional probability of being assigned to the designated treatment condition. As described by Hahn, Murray, and Carvalho, including the propensity score as a predictor can help reduce the bias from regularization and improve estimation of a total effect. Moreover, it was observed by Dorie et al. (2019) that BART models that are fit using the estimated propensity score values as a predictor produce credible intervals that have better coverage rates for the treatment effect. To demonstrate this approach, the following statements use the GAMSELECT procedure in SAS Visual Statistics to fit a model for the treatment variable Quit:

```
proc gamselect data=mycas.smokingWeight;
   class Sex Race Education Exercise Activity Quit;
   model Quit(Event='1') = param(Sex Race Education Exercise
         Activity) spline(Age) spline(PerDay) spline(YearsSmoke);
   selection method=boosting;
   output out=mycas.SmokingWeightPS pred=pScore
          copyVars=(Quit Sex Race Education Exercise Activity
                    Change Age PerDay YearsSmoke);
run;
```

PROC GAMSELECT fits and performs model selection for generalized additive models. In this example, the classification variables are included in the model as parametric effects and listed in the PARAM option. The SPLINE option specifies nonparametric spline effects that are constructed from continuous variables. A univariate spline term is constructed here for each continuous predictor. The boosting selection method is used to select and fit the treatment model for this example. The output data table SmokingWeightPS contains the predicted treatment probabilities and the variables that are needed to train the BART model.

The following statements invoke the BART procedure to fit a new model for the outcome variable that incorporates the predicted propensity score values, as well as to compute the Quit predictive margins by using the new model:

```
proc bart data=mycas.SmokingWeightPS seed=1976;
   class Sex Race Education Exercise Activity Quit ;
   model Change = Quit Sex Education Exercise Activity
                  YearsSmoke PerDay Age pScore;
   store mycas.store2;
run;

proc cas;
   action bart.bartScoreMargin /
   table = {name="smokingWeightPS"}
   restore = {name="store2"}
   margins= {{ name="Cessation",   at={{var="Quit" value="1"}}}
             { name="No Cessation",at={{var="Quit" value="0"}}}}
   differences = {{ label="Cessation Difference"
                    refMargin="No Cessation"
                    evtMargin="Cessation"} };
   run;
quit;
```

Figure 7 shows the predictive margin estimates and their difference obtained by using the updated model. For this example, there is little difference in the potential outcome mean and ATE estimates obtained using either model.

Figure 7: Predictive Margins and Their Difference with Propensity Score Predictor

Results from bart.bartScoreMargin

Predictive Margins

Description	Estimate	95% Equal-Tail Interval	
Cessation	5.23025	4.46545	5.97486
No Cessation	1.77635	1.35233	2.22522

Summary

Starting in the 2022.1.1 release of SAS Visual Statistics, you can fit BART models of normally distributed response variables either by using the BART procedure or by using the `bartGauss` action in the Bayesian Additive Regression Trees action set. A BART model that is fit using either the procedure or the action can be saved as an analytic store and used with two additional actions in the `bart` action set. You can use the `bartScore` action to score new observations by using a fitted model, and you can use the `bartScoreMargin` action to compute predictive margins.

BART models provide a powerful and easy-to-use tool for statisticians and data scientists who want to model an outcome variable. By using a sum-of-trees ensemble to approximate the conditional mean of a random variable, these models can incorporate a mix of continuous and categorical predictors without requiring the explicit modeling of interaction terms, and they can naturally incorporate rules for handling missing predictor values. Moreover, BART models are user-friendly because of the typically robust performance of the default BART prior and the good generalization of predictions from the default BART model. As a result, these models typically do not require hyperparameter tuning, and as shown in "Example 1: Comparing Model Fit," the default BART model fit is usually comparable to the fit of other tree-based methods that use hyperparameter tuning.

Unlike other tree-based predictive models that you obtain by using gradient boosting or random forests, BART models consist of many posterior samples of a sum-of-trees ensemble instead of a single ensemble. As a result, a BART model is typically much larger than a model that is obtained by using gradient boosting or random forests. A downside of the larger BART model is that it can be a more time-consuming process both to train a BART model and to score new observations by using the fitted model. The larger amount of time required to score observations can have an impact on the explainability of a BART model, because many commonly used model-agnostic explanation methods involve scoring data by using a fitted model. However, a benefit of using a BART model is that you can use a Bayesian perspective to assess uncertainty in the model predictions by assessing variability in the posterior samples. As a result, BART models can be well suited for situations where assessing uncertainty in the model predictions might be of greater importance than explaining how predictions are obtained from the model. As demonstrated in "Example 2: Computing Predictive Margins," an application where this is the case, and one for which BART models have been observed to perform quite well, is the estimation of causal effects by using a regression-based, or response-surface-based, approach.

References

Breiman, L. (1996). "Bagging Predictors." *Machine Learning* 24:123–140.

Breiman, L. (2001). "Random Forests." *Machine Learning* 45:5–32.

Chipman, H. A., George, E. I., and McCulloch, R. E. (1998). "Bayesian CART Model Search." *Journal of the American Statistical Association* 93:935–948. https://doi.org/10.2307/2669832.

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010). "BART: Bayesian Additive Regression Trees." *Annals of Applied Statistics* 4:266–298. https://doi.org/10.1214/09-AOAS285.

Dorie, V., Hill, J., Shalit, U., Scott, M., and Cervone, D. (2019). "Automated versus Do-It-Yourself Methods for Causal Inference: Lessons Learned from a Data Analysis Competition." *Statistical Science* 34:43–68. https://doi.org/10.1214/18-STS667.

Friedman, J. H. (2001). "Greedy Function Approximation: A Gradient Boosting Machine." *Annals of Statistics* 29:1189–1232.

Hahn, P. R., Murray, J. S., and Carvalho, C. M. (2020). "Bayesian Regression Tree Models for Causal Inference: Regularization, Confounding, and Heterogeneous Effects." *Bayesian Analysis* 15:965–1056. With discussion. https://doi.org/10.1214/19-BA1195.

Hernán, M. A., and Robins, J. M. (2020). *Causal Inference: What If*. Boca Raton, FL: Chapman & Hall/CRC.

Pratola, M. T., Chipman, H. A., Gattiker, J. R., Higdon, D. M., McCulloch, R. E., and Rust, W. N. (2014). "Parallel Bayesian Additive Regression Trees." *Journal of Computational and Graphical Statistics* 23:830–852. https://doi.org/10.1080/10618600.2013.841584.