Tree-Based Machine Learning Methods in SAS® Viya®

Sharad Saxena

The correct bibliographic citation for this manual is as follows: Saxena, Sharad. 2022. *Tree-Based Machine Learning Methods in SAS® Viya®*. Cary, NC: SAS Institute Inc.

Tree-Based Machine Learning Methods in SAS® Viya®

ISBN 978-1-954846-71-5 (Hardcover)
ISBN 978-1-954846-63-0 (Paperback)
ISBN 978-1-954846-64-7 (Web PDF)
ISBN 978-1-954846-65-4 (EPUB)
ISBN 978-1-954846-66-1 (Kindle)

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

February 2022

Contents

About This Book

Preface

Decision trees are popular machine learning models. They are very intuitive and natural for us humans because in a way, they mimic the way we make decisions in our day-to-day lives. Several decision trees can be combined to produce ensemble models for better predictive performance than using a single decision tree. Decision trees and tree-based ensembles are supervised learning models used for problems involving classification and regression. They are largely used for prediction, but they also have several other uses in the modeling process and beyond.

Why you might want to use tree-based predictive modeling? Tree-based models are designed to handle categorical variables without you needing to create dummy variables. Tree-based models directly consider categorical levels when determining a split. Tree-based models can also handle observations with missing values instead of ignoring an observation with the missing value or being forced to impute. A tree can assign the missing value to one of those branches. This strategy can also be used to score an unknown level that occurs at scoring time but was not seen at training time. Tree-based models are also able to detect nonlinear relationships. Finally, a single decision tree can be interpreted as a set of rules or can easily be visualized in a tree plot.

SAS® Visual Data Mining and Machine Learning software on SAS® Viya® has three tree-based predictive modeling techniques – decision trees, forests, and gradient boosted trees. A decision tree is visually appealing and highly explainable. A forest model is a bagged ensemble of trees and has high predictive power but is harder to interpret than a single tree. A gradient boosting model is a boosted ensemble of trees and is very hard to interpret but has high predictive power. In SAS Visual Data Mining and Machine Learning, it is easy to start building one type of tree-based model and transition to building another. The output is similar so that you can understand the results more easily from one model to the next.

The analytics for all three tree-based predictive modeling techniques can be found in the Decision Tree Action Set in SAS Viya. These techniques can be accessed in the programming interface of SAS Visual Data Mining and Machine Learning when you use the TREESPLIT procedure, the FOREST procedure, or the GRADBOOST procedure. You can access these procedures through the SAS® Studio HTML client. SAS Studio also has tasks that can help you generate code. SAS® Visual Analytics has objects that can help you create these models interactively using a point-and-click interface. The actions in SAS Viya are also available to be called from other clients such as the SAS Visual Analytics client, Java, or Python, and more. For this book, we will primarily consider point-and-click visual interfaces that include mostly the pipelines in Model Studio, a bit of SAS Studio and SAS Visual Statistics. A little exposure of procedures in the SAS programming interface to the SAS Viya actions is also covered in the book.

The book includes discussions of tree-structured predictive models and the methodology for growing, pruning, and assessing decision trees, forests, and gradient boosted trees. You will acquire knowledge not only of how to use tree-based models for classification and regression, and some of their limitations, but also how the respective algorithms that shape them work. Each demonstration introduces a new data concern and then walks you through tweaking the modeling approach, modifying the properties, and changing the hyperparameters, thus building a right tree-based machine learning model. Along the way, you will gain experience making decision trees, forests, and gradient boosted trees that work for you.

The book also explains isolation forest (an unsupervised learning algorithm for anomaly detection), deep forest (an alternative for neural network deep learning), and Poisson and Tweedy gradient boosted regression trees. In addition, many of the auxiliary uses of trees, such as exploratory data analysis, dimension reduction, and missing value imputation are examined and running open source in SAS and SAS in open source is demonstrated.

What Does This Book Cover?

This book covers everything from using a single tree to more advanced bagging and boosting ensemble methods in SAS Viya.

Chapter 1 provides an introduction to tree-structured models and the ones that are available in SAS Viya. Decision trees are powerful predictive and explanatory modeling tools. They are flexible in that they can model interval, ordinal, nominal, and binary targets, and they can accommodate nonlinearities and interactions. They are simple to understand and present. Model Studio is also introduced in this chapter.

Chapter 2 discusses the types of decision trees. The tree can be either a classification tree, which models a categorical target, or a regression tree, which models a continuous target. The model is expressed as a series of if-then statements. For each tree, you specify a target (dependent, response) variable and one or more input (independent, predictor) variables. The input variables for tree models can be categorical or continuous. The initial node is called the root node, and the terminal nodes are called leaves. Partitioning is done repeatedly, starting with the root node, which contains all the data, and continuing to split the data until a stopping criterion is met. At each step, the parent node is split into two or more child nodes by selecting an input variable and a split value for that variable. As with other modeling tools, decision tree models generate scoring code and can be used to generate predictions on new data through the Score node in Model Studio.

Chapter 3 talks about how to grow a decision tree. Recursive partitioning is the method whereby a tree grows from the root node to its maximal size. Splitting criteria are used to assess and compare splits within and between inputs. Those criteria measure reduction in variability in child versus parent nodes. The goal is to reduce variability and thus increase purity. Various measures, such as the Gini index, entropy, and residual sum of squares, can be used to assess candidate splits for each node. The process of building a decision tree begins with growing a large, full tree. The full tree can overfit the training data, resulting in a model that does not adequately generalize to new data. Exhaustive split searches can be computationally expensive. Decision tree algorithms use shortcuts to reduce the split search primarily through restrictions on the number of child nodes (for example, binary splitting as used in CART trees) or using agglomerative clustering (as in CHAID).

Decision tree modeling with interval targets is like modeling with categorical targets. The split search considerations remain the same, and analogous (or equivalent) split criteria are used. Violation of the equal variance assumption from regression and ANOVA can lead to spurious split selection in decision trees. Target transformations, such as the logarithm, might improve split selection. The interactive training facility in SAS Viya enables the user to select variables and split points for model building. Entire models can be built by the user. Alternatively, initial splits can be user-defined, and subsequent splits can be chosen automatically by the tree algorithm.

Chapter 4 covers common advantages and disadvantages of decision trees along with some of their prominent secondary uses. Decision trees can directly incorporate missing values without the need for imputation. Methods include placing missing values in child nodes that lead to the greatest improvement in purity, or in nodes with the most cases. Alternatively, inputs that show highest agreement with primary split variables can be used to assign case membership in child nodes.

Variables' importance can be calculated for inputs involved in modeling, either through a measure of their contributions to impurity reduction or through a measure that compares a given fit statistic (for example, average squared error) for trees with the input included versus a tree in which the input is rendered uninformative.

Compared with other regression and classification methods, decision trees have the advantage of being easy to interpret and visualize, especially when the tree is small. Tree-based methods scale well to large data, and they offer various methods of handling missing values, including surrogate splits. However, decision trees do have limitations. Regression trees fit response surfaces that are constant over rectangular regions of the predictor space, so they often lack the flexibility needed to capture smooth relationships between the predictor variables and the response. Another limitation of decision trees is that small changes in the data can lead to very different splits, and this undermines the interpretability of the model. Random forest models address some of these limitations.

Decision trees have many uses beyond modeling. They are useful as multivariate data exploration tools because of their easily interpreted visual output and because they do not require much in the way of input preparation to use. In some cases, they can be used to aid in identifying interactions to add to regression models. Trees can be used to select important variables for subsequent models such as regression and neural networks because they are resistant to the curse of dimensionality and contain methods to calculate variable importance. Trees can also reduce the number of dimensions for subsequent models by collapsing levels of categorical inputs. Levels that have similar average target values are grouped into the same leaf. Trees can be used to discretize interval inputs to aid subsequent model interpretation or to help accommodate nonlinear patterns between inputs and the target. They might also help identify broad segments within which regression models can be fit. Trees can be used for imputation. Missing values are imputed based on predictive tree models in which variables with missing values are treated as the target and all other variables as model inputs. Tree imputation is available in the Imputation node.

Chapter 5 describes how to tune a decision tree. The “right-sized” tree can be found by using top-down or bottom-up criteria. Top-down criteria limit tree growth. CHAID algorithms tend to feature such criteria, and rely on statistical tests, alpha values, and p-value adjustments (Bonferroni), among others, to restrict the size of the maximal tree. In bottom-up pruning, a large tree is grown and then branches are removed in a backward fashion using some model assessment selection criterion. This strategy originated with cost-complexity pruning as used by CART decision trees. Model selection criteria include accuracy and average square error depending on the type of target variable and are applied to validation data. During bottom-up pruning, tree models are selected based on validation data, if present. Cross validation is a less expensive approach to model validation in which all data is used for both training and validation. Cross validation is available in the Decision Tree node in Model Studio.

When fitting multiple decision trees, you can refer to CART and CHAID methodologies to guide your choice of tree property settings. If you are unsure about what values of hyperparameters to be used for building a decision tree, it is best to use SAS® Viya® autotuning functionality.

Chapter 6 describes ensemble of trees and then focuses on the forest model. Tree-based ensemble models take advantage of the inherent instability of decision trees, whereby small changes in the training data can result in large changes in tree structure due to large numbers of competing splits. Perturb and combine methods generate multiple models by manipulating the distribution of the data or altering the construction method and then averaging the results. Bagging resamples training data with replacement and then fits models to each sample. Boosting adaptively reweights subsequent samples based on mistakes that are made in target classification from previous samples. Ensemble models might help smooth the prediction surface that is generated by individual tree models and thereby improve model performance.

A forest is just what the name implies. It’s a bunch of decision trees – each with a randomly selected subset of the data – all combined into one result. Using a forest helps address the problem of overfitting inherent to an individual decision tree. The forest model in SAS Viya creates a random forest using literally hundreds of decision trees. A forest consists of several decision trees that differ from each other in two ways. First, the training data for a tree is a sample with replacement from all available observations. Second, the input variables that are considered for splitting a node are randomly selected from all available inputs. Among these randomly selected variables, the input variable that is most often associated with the target is used for the splitting rule. In other respects, trees in a forest are trained like standard trees. The training data for an individual tree excludes some of the available data that is used to assess the fit of the model.

The forest model in SAS Viya accepts interval and class target variables. For an interval target, the prediction in a leaf of an individual tree equals the average of the target values among the bagged training observations in that leaf. For a class target, the posterior probability of a target category equals the proportion of that category among the bagged training observations in that leaf. Predictions or posterior probabilities are then averaged across all the trees in the forest. Averaging over trees with different training samples reduces the dependence of the predictions on a training sample. Increasing the number of trees does not increase the risk of overfitting the data and can decrease it. However, if the predictions from different trees are correlated, then increasing the number of trees makes little or no improvement.

Chapter 7 covers some additional forest models. An isolation forest is a specially constructed forest that is used for anomaly detection instead of target prediction. When an isolation forest is created, it writes anomaly scores in the scored data table.

The recently proposed deep forest, which is also termed gcForest (multi-Grained Cascade Forest), is a novel decision tree ensemble approach. This method generates a deep forest ensemble with a cascade structure, which enables gcForest to do representation learning, which can find out the better features by end-to-end training. Its representational learning ability can be further enhanced by multi-grained scanning when the inputs are with high dimensionality, potentially enabling gcForest to be contextual or structural aware. The performance of deep forest claims to be more competitive than that of deep neural network (DNN).

Chapters 8 explains tree-based gradient boosting models. Gradient boosted trees create an ensemble model of weak decision trees in a stage-wise, iterative, sequential manner. Each tree uses a subsample of the data. Gradient boosting algorithms convert weak learners to strong learners. One advantage of gradient boosting is that it can reduce bias and variance in supervised learning. All points begin with the same weight. Points classified correctly are given a lower weight and those classified incorrectly are given a higher weight. Now the model focuses on high weight points and classifies them correctly. However, others that were classified correctly in the first iteration are now misclassified. This process continues for many iterations. In the end, all models are given a weight depending on their accuracy, and the model results are combined into one consolidated result.

Gradient boosting models can be fit in SAS Viya using several ways including the Gradient Boosting node in Model Studio, the GRADBOOST procedure, and so on. A decision tree in a gradient boosting model trains on new training data that are derived from the original training. Using different data to train different trees during the boosting process reduces the correlation of the predictions of the trees, which in turn improves the predictions of the boosting model.

The algorithm samples the original data without replacement to create the training data for an individual tree. It performs the action of sampling multiple times throughout a run, and each set of training data created is referred to as a subsample. In some cases, gradient boosting models can be overtrained and thus perform poorly on validation or test data. One method to combat overtraining in gradient boosting is early stopping. An additional advantage to early stopping is reduced training time in cases where the stopping criterion is met well before the specified maximum number of iterations occurs.

Chapter 9 explores some additional gradient boosting models. Gradient boosting can be used for transfer learning. You can use the auxiliary data to increase the number of observations for training the model. To prevent the model from being overly biased toward the auxiliary data, the gradient boosting algorithm down-weights auxiliary observations that are dissimilar from the training observations.

You can specify the distribution of the objective function in a gradient boosting model. The default distribution is Gaussian, binary, or multinomial for an interval, binary, or nominal target, respectively. The Poisson distribution is appropriate for count data. The Tweedie distribution is useful for modeling total losses in insurance.

Hyperparameter tuning is available in the three tree-based models, decision tree, forest, and gradient boosting, to find the best values for various options. These include the splitting criterion, maximum depth, and number of bins in decision trees; the fraction of training data to sample, maximum tree depth, number of trees, and number of variables to consider for each split in forest; and the L1 and L2 regularization parameters, learning rate, fraction of training data to sample, and number of variables to consider for each split in gradient boosting. There are several objective functions to choose from for the optimization algorithm as well as search methods, including one based on a genetic algorithm.

At the end, a practice case study is provided.

Is This Book for You?

Building representative tree-based machine learning models that generalize well on new data requires careful consideration of both the data used for the model to train, and the assumptions about the various tree-based algorithms. It is important to choose the right options and best hyperparameter values for both the data that you will be modeling and the business problem at hand.

If you want to gain insights about tree-based models and all the nitty-gritty details involved in decision trees, random forests, and gradient-boosted trees, then this book is for you! The discussion can help you quickly get value out of tree-based models from intermediate to advanced level. This book is most suitable for predictive modelers and data analysts who want to build decision trees and ensembles of decision trees using SAS Visual Data Mining and Machine Learning in SAS Viya.

What Are the Prerequisites for This Book?

Before reading this book, you should have the following:

- An understanding of basic statistical concepts. You can gain this knowledge from the course “SAS® Visual Statistics on SAS® Viya®: Interactive Model Building”.
- Familiarity with SAS Visual Data Mining and Machine Learning software. You can gain this knowledge from the course “Machine Learning Using SAS® Viya®”.

What Should You Know about the Examples?

This book includes worked demonstrations and practices for you to follow to gain hands-on experience with SAS Visual Data Mining and Machine Learning software including, but not limited to, Model Studio.

Software Used to Develop the Book’s Content

SAS Visual Data Mining and Machine Learning 8.5 is a comprehensive visual – and programming – interface supports the end-to-end data mining and machine learning process. Analytics team members of all skill levels are empowered to handle all analytics life cycle tasks in a simple, powerful and automated way. You can solve complex analytical problems with a comprehensive visual interface that handles all tasks in the analytics life cycle. SAS Visual Data Mining and Machine Learning, which runs in SAS Viya 3.5, combines data wrangling, exploration, feature engineering, and modern statistical, data mining, and machine learning techniques in a single, scalable in-memory processing environment. The software also includes SAS Visual Statistics 8.5 and SAS Visual Analytics 8.5.

Model Studio is included in SAS Viya. It is an integrated visual environment that provides a suite of analytic data mining tools that enable you to explore and build models. It is part of the Discovery phase of the analytic life cycle. The data mining tools provided in Model Studio enable you to deliver and distribute analytic model data mining champion models, score code, and results.

Example Code and Data

The data sets used in the book’s demonstrations and practices are provided to download.

You can access the example code and data for this book by linking to its author page at https://support.sas.com/saxena. The Solutions to the Case Study can also be found on the author page.

We Want to Hear from You

SAS Press books are written *by* SAS Users *for* SAS Users. We welcome your participation in their development and your feedback on SAS Press books that you are using. Please visit sas.com/books to do the following:

- Sign up to review a book
- Recommend a topic
- Request information about how to become a SAS Press author
- Provide feedback on a book

Do you have questions about a SAS Press book that you are reading? Contact the author through saspress@sas.com or https://support.sas.com/author_feedback.

SAS has many resources to help you find answers and expand your knowledge. If you need additional help, see our list of resources: sas.com/books.

About the Author

Dr. Sharad Saxena is Principal Analytical Training Consultant based at the SAS R&D center in Pune, India. He has been working in the field of statistics and analytics since 2000. He has been doing education consulting in the area of advanced analytics and machine learning across the globe including the UK, USA, Singapore, Italy, Australia, Netherlands, Middle East, China, Philippines, Nigeria, Hong Kong, Malaysia, Indonesia, Mexico, and India for a variety of SAS customers in banking, insurance, retail, government, health, agriculture, and telecommunications. Dr. Saxena earned a bachelor's degree in mathematics with statistics and economics minors, a master's degree in statistics, and a Ph.D. in statistics from the School of Studies in Statistics at Vikram University, India. He has also completed some short-term courses in allied areas from the Indian Statistical Institute, Kolkata, and the Indian Institute of Management, Ahmedabad. He is recipient of the prestigious U.S. Nair Young Statistician Award in 2001–02, and his biography has been included in *Who's Who in Science and Engineering* 2006–2007 for his outstanding contributions in the field of statistics. Prior to joining SAS, he worked as a Product Head–Business Analytics at TimesPro (The Times of India Group) where he started and managed the vocational training program in business analytics. Overall, he has more than two decades of rich experience in research, teaching, training, consulting, writing, and education product design, more than 14 years of which have been with SAS and the remaining in academia as a faculty member with some top-notch institutes in India like the Institute of Management Technology, Ghaziabad; Institute of Management, Nirma University, and more. Dr. Saxena has more than 35 publications including research papers in journals such as the *Journal of Statistical Planning and Inference*, *Communications in Statistics–Theory and Methods*, *Statistica*, *Statistical Papers*, and Vikalpa. He is a co-author of the book titled *Randomness and Optimal Estimation in Data Sampling* published by American Research Press. Apart from various talks delivered at national and international conferences, he has written some case studies, technical notes, white papers, book reviews, and popular columns in newspapers. He is an active member of many professional and academic bodies.

Learn more about this author by visiting his author page at http://support.sas.com/saxena. There you can download free book excerpts, access example code and data, read the latest reviews, get updates, and more.

Chapter 1: Introduction to Tree-Structured Models

Introduction

> "Sometimes you make the right decision, sometimes you make the decision right."
>
> *–Phil McGraw*

A decision tree has many analogies in real life. In decision analysis, a tree can be used to represent decisions and decision making visually and explicitly. As the name suggests, it uses a tree-like model of decisions.

The adjective *decision* in decision trees is a curious one, and misleading. In the 1960s, originators of the tree approach described the splitting rules as decision rules. The terminology remains popular. This is ill-fated because it inhibits the use of ideas and terminology from decision theory. The term decision tree is used in decision theory to depict a series of decisions for choosing alternative activities. You create the tree and specify probabilities and benefits of outcomes of the activities. Software, including SAS, finds the most beneficial path. The project follows a single path and never performs the unchosen activities. The decider follows a path based on a set of criteria.

Decision theory is not about data analysis. The choice of a decision might be made without reference to data. The trees in this book are only about data analysis. A tree is fit to a data set to enable interpretation and prediction of data. An apt name would be *data-splitting trees* that would be used for *supervised learning* also called *predictive modeling*.

In supervised learning, a set of *input* variables (predictors) is used to predict the value of one or more *target* variables (outcome). The mapping of the inputs to the target is a *predictive model*. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the input variables. The data used to estimate a predictive model is a set of *cases* (observations, examples) consisting of values of the inputs and target. The fitted model is typically applied to new cases where the target is unknown.

Decision Tree – What Is It?

There are several tree-structured models that include one or more decision trees. Decision trees are a fundamental machine learning technique that every data scientist should know. Luckily, the construction and implementation of decision trees in SAS Viya is straightforward and easy to produce.

A *decision tree* represents a grouping of the data that is created by applying a series of simple rules. Each rule assigns an observation to a group based on the value of one input. One rule is applied after another, resulting in a hierarchy of groups within groups. The hierarchy is called a *tree*, and each group is called a *node*. The original group contains the entire data set and is called the *root node* of the tree. A node with all its successors forms a branch of the node that created it. The final nodes are called *leaves*. For each leaf, a decision is made and applied to all observations in the leaf. The type of decision depends on the context. In supervised learning, the decision is the predicted value.

You use the decision tree to do one of the following tasks:

- classify observations based on the values of nominal, binary, or ordinal targets
- predict outcomes for interval targets
- predict the appropriate decision when you specify decision alternatives

The tree depicts the first split into groups as branches emanating from a root and subsequent splits as branches emanating from nodes on older branches. Figure 1.1 is an example decision tree predicting a nominal target *Cause of Death* using two binary inputs *Weight Status* and *Smoking Status*. The decision nodes include a bar chart related to the node's sample target values and other details. The leaves of the tree are the final groups, the unsplit nodes. For some perverse reason, trees are always drawn upside down, like an organizational chart. For a tree to be useful, the data in a leaf must be similar with respect to some target measure so that the tree represents the segregation of a mixture of data into purified groups.

Types of Decision Trees

Decision trees are a nonparametric supervised learning method used for both classification and regression tasks. A classification tree models a categorical response, and a regression tree models a continuous response. See Figure 1.2. Both types of trees are called decision trees because the model is expressed as a series of if-then statements. For each type of tree, you specify a response variable

Figure 1.1: A Simple Decision Tree

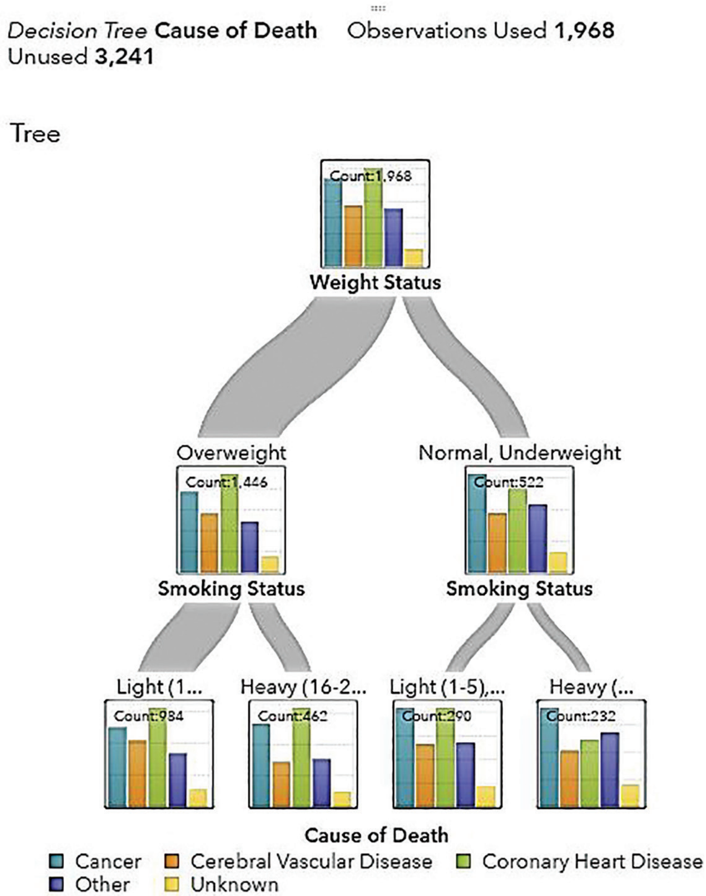
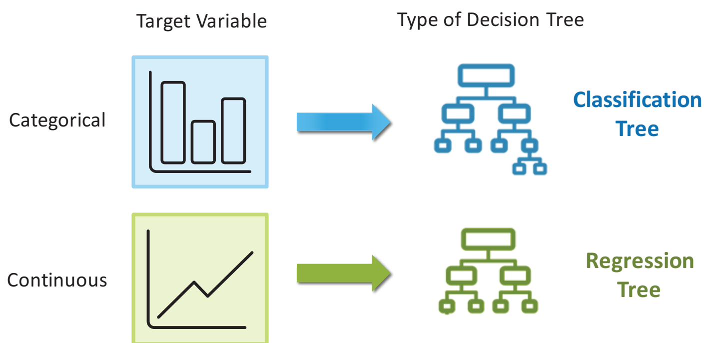


Figure 1.2: Classification and Regression Trees



(also called a target variable), whose values you want to predict, and one or more input variables (called predictor variables), whose values are used to predict the values of the target variable.

The predictor variables for tree models can be categorical or continuous. The set of all combinations of the predictor variables are called the *predictor space*. The model is based on partitioning the predictor space into nonoverlapping groups, which correspond to the leaves of the tree. Partitioning is done repeatedly, starting with the root node, which contains all the data, and continuing until a stopping criterion is met. At each step, the parent node is split into child nodes by selecting a predictor variable and a split value for that variable that minimize the variability according to a specified measure (or the default measure) in the response variable across the child nodes. Various measures, such as the Gini index, entropy, and residual sum of squares, can be used to assess candidate splits for each node. The selected predictor variable and its split value are called the primary *splitting rule*.

Tree-structured models are built from training data for which the response values are known, and these models are subsequently used to score (classify or predict) response values for new data. For classification trees, the most frequent response level of the training observations in a leaf is used to classify observations in that leaf. For regression trees, the average response of the training observations in a leaf is used to predict the response for observations in that leaf. The splitting rules that define the leaves provide the information that is needed to score new data; these rules consist of the primary splitting rules, surrogate rules, and default rules for each node.

The process of building a decision tree begins with growing a large, full tree. The full tree can overfit the training data, resulting in a model that does not adequately generalize to new data. To prevent overfitting, the full tree is often pruned back to a smaller subtree that balances the goals of fitting training data and predicting new data. Two commonly applied approaches for finding the best subtree are cost-complexity pruning and C4.5 pruning.

Compared with other regression and classification methods, tree-structured models have the advantage that they are easy to interpret and visualize, especially when the tree is small. Tree-based methods scale well to large data, and they offer various methods of handling missing values, including surrogate splits.

However, tree-structured models have limitations. Regression tree models fit response surfaces that are constant over rectangular regions of the predictor space, so they often lack the flexibility needed to capture smooth relationships between the predictor variables and the response. Another limitation of tree models is that slight changes in the data can lead to quite different splits, and this undermines the interpretability of the model.

Tree-Based Models in SAS Viya

SAS Viya is a cloud-enabled, analytic run-time environment with several supporting services, including SAS Cloud Analytic Services (CAS). CAS is the in-memory engine on the SAS Viya Platform.

SAS Viya builds tree-based statistical models for classification and regression. You can build three tree-based models in SAS Viya starting from a single tree to more complex ensembles of trees like forest and gradient boosting.

A random forest is just what the name implies. It is a bunch of decision trees – each with a randomly selected subset of the data – all combined into one result. Using a random forest helps address the problem of overfitting inherent to an individual decision tree.

Gradient boosting creates an ensemble model of weak decision trees in a stage-wise, iterative, sequential manner. Gradient boosting algorithms convert weak learners to strong learners. One advantage of gradient boosting is that it can reduce bias and variance in supervised learning.

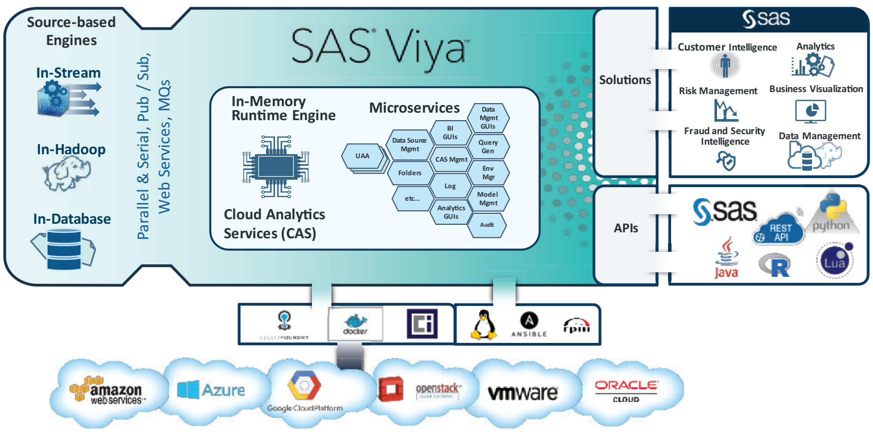
Analytics Platform from SAS

The SAS Analytics Platform is a software foundation that is engineered to address today's business challenges and to generate insights from your data in any computing environment. SAS Viya is the latest extension of the SAS Analytics Platform, which is designed to orchestrate your entire analytic ecosystem, connecting and accelerating all analytics life cycle – from data, to discovery, to deployment. SAS Viya seamlessly scales to data of any size, type, speed, and complexity, and is interoperable with SAS 9. As an integrated part of the SAS Analytics Platform, SAS Viya is a cloud-enabled, in-memory analytics engine.

The SAS Viya Platform architecture is illustrated in Figure 1.3. At the heart of SAS Viya is SAS Cloud Analytic Services (CAS), an in-memory, distributed analytics engine. It uses scalable, high-performance, multi-threaded algorithms to rapidly perform analytical processing on in-memory data of any size.

SAS Viya contains microservices. A microservice is a small service that runs in its own process and communicates with a lightweight mechanism (hypertext transfer protocol (HTTP)). Microservices

Figure 1.3: SAS Viya Platform Architecture



are a series of containers that define all the different analytic life cycle functions, sometimes described as “actions” that fit together in a modular way. The in-memory engine is independent from the microservices and allows for independent scalability.

On the left of Figure 1.3 you see a series of source-based data engines.

SAS Viya has a middle tier implemented on a micro-services architecture, deployed and orchestrated through the industry standard cloud Platform as a Service also known as Cloud Foundry. Through Cloud Foundry, SAS Viya can be deployed, managed, monitored, scaled, and updated. Cloud Foundry enables SAS Viya to support multiple cloud infrastructure allowing customers to deploy SAS in a hybrid cloud environment spanning multiple clouds including the combination of on-premises cloud infrastructure and public cloud infrastructure.

You can choose to use other platforms like Docker and the open container initiative. You can operate on private infrastructure such as OpenStack or VMware, or open infrastructure such as Amazon Web Services, Azure, and so on.

Existing SAS solutions and new ones are being built on SAS Viya. In addition, you can use REST API to include SAS Viya actions in your existing applications. A REST API is an application programming interface that conforms to the constraints of representational state transfer (REST) architectural style and allows for interaction with RESTful web services.

SAS Visual Data Mining and Machine Learning

SAS Visual Data Mining and Machine Learning is a product offering in SAS Viya that contains the underlying CAS actions and SAS procedures for data mining and machine learning applications, and graphical user interface (GUI)-based applications for various levels and types of users.

These applications are as follows:

- *Programming interface:* a collection of CAS action sets and SAS procedures for direct coding or access through tasks in SAS Studio.
- *Interactive modeling interface:* a collection of objects in SAS Visual Analytics for creating models in an interactive manner with automated assessment visualizations.
- *Automated modeling interface:* a pipeline application called Model Studio that enables you to construct automated flows consisting of various nodes for preprocessing and modeling with automated model assessment and comparison and direct model publishing and registration.

Each of these executes the same underlying actions in the CAS execution environment.

> In this book, you primarily explore the Model Studio interface and its integration with other SAS Visual Data Mining and Machine Learning interfaces.

You can use the SAS Visual Data Mining and Machine Learning web client to assemble, configure, build, and compare tree-based models visually and programmatically.

SAS Viya provides two programming run-time servers for processing data that is not performed by the CAS server. Which server is used is determined by your SAS environment. When your SAS environment includes the SAS Viya visual and programming environments, your SAS administrator determines the server. The SAS Workspace Server and the SAS Compute Server support the same SAS code and produce the same results.

There are several interfaces and ways of executing analyses in SAS Viya. This includes the CAS actions, SAS procedures, and visual applications shown in Figure 1.4.

The Decision Tree Action Set

Decision Tree Action Set (Table 1.1) provides actions for modeling and scoring with tree-based models that include decision trees, forests, and gradient boosting.

Figure 1.4: Interfaces and Ways of Executing Analyses in SAS Viya

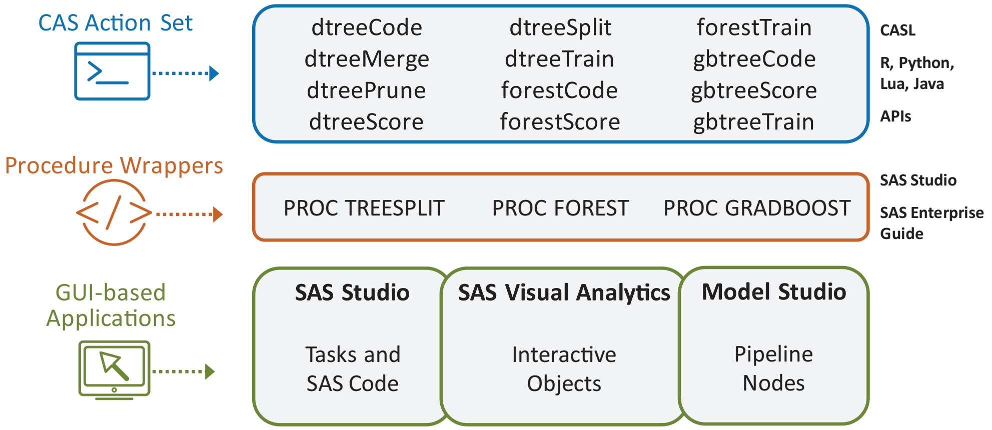


Table 1.1 Decision Tree Action Set

Action Name	Description
dtreeCode	Generates DATA step scoring code from a decision tree model
dtreeMerge	Merges decision tree nodes
dtreePrune	Prunes a decision tree
dtreeScore	Scores a table using a decision tree model
dtreeSplit	Splits decision tree nodes

(*Continued*)

Table 1.1 Decision Tree Action Set

Action Name	Description
dtreeTrain	Trains a decision tree
forestCode	Generates DATA step scoring code from a forest model
forestScore	Scores a table using a forest model
forestTrain	Trains a forest
gbtreeCode	Generates DATA step scoring code from a gradient boosting tree model
gbtreeScore	Scores a table using a gradient boosting tree model
gbtreeTrain	Trains a gradient boosting tree

SAS Viya also supports new analytic methods that can be accessed from SAS and other programming languages that include R, Python, Lua, and Java, as well as public REST APIs.

TREESPLIT, FOREST, and GRADBOOST Procedures

The TREESPLIT procedure builds tree-based statistical models for classification and regression in SAS Viya. The procedure produces a classification tree, which models a categorical response, or a regression tree, which models a continuous response. For each type of tree, you specify a target variable whose values you want PROC TREESPLIT to predict and one or more input variables whose values the procedure uses to predict the values of the target variable.

The following statements and options are available in the TREESPLIT procedure:

```
PROC TREESPLIT <options>;
        AUTOTUNE <options>;
        CLASS variables;
        CODE <options>;
        FREQ variable;
        GROW criterion <options>;
        MODEL response = variable. . .;
        OUTPUT OUT=CAS-libref.data-table output-options;
        PARTITION <partition-options>;
        PRUNE prune-method <(prune-options)>;
        VIICODE <options>;
        WEIGHT variable;
```

The PROC TREESPLIT statement and the MODEL statement are required.

The FOREST procedure creates a predictive model called a forest (which consists of several decision trees) in SAS Viya. The FOREST procedure creates an ensemble of decision trees to predict a single target of either interval or nominal measurement level. An input variable can have an interval or nominal measurement level.

The following statements are available in the FOREST procedure:

```
PROC FOREST <options>;
        AUTOTUNE <options>;
        CODE <options>;
        CROSSVALIDATION <options>;
        GROW criterion;
        ID variables;
        INPUT variables </ LEVEL=NOMINAL | INTERVAL>;
        OUTPUT OUT=CAS-libref.data-table <option>;
        PARTITION partition-option;
        SAVESTATE RSTORE=CAS-libref.data-table;
        TARGET variable </ LEVEL=NOMINAL | INTERVAL>;
        VIICODE <options>;
        WEIGHT variable;
```

The PROC FOREST, INPUT, and TARGET statements are required. The INPUT statement can appear multiple times.

The GRADBOOST procedure creates a predictive model called a *gradient boosting model* in SAS Viya. Based on the boosting method in Hastie, Tibshirani, and Friedman (2001) and Friedman (2001), the GRADBOOST procedure creates a predictive model by fitting a set of additive trees.

The following statements are available in the GRADBOOST procedure:

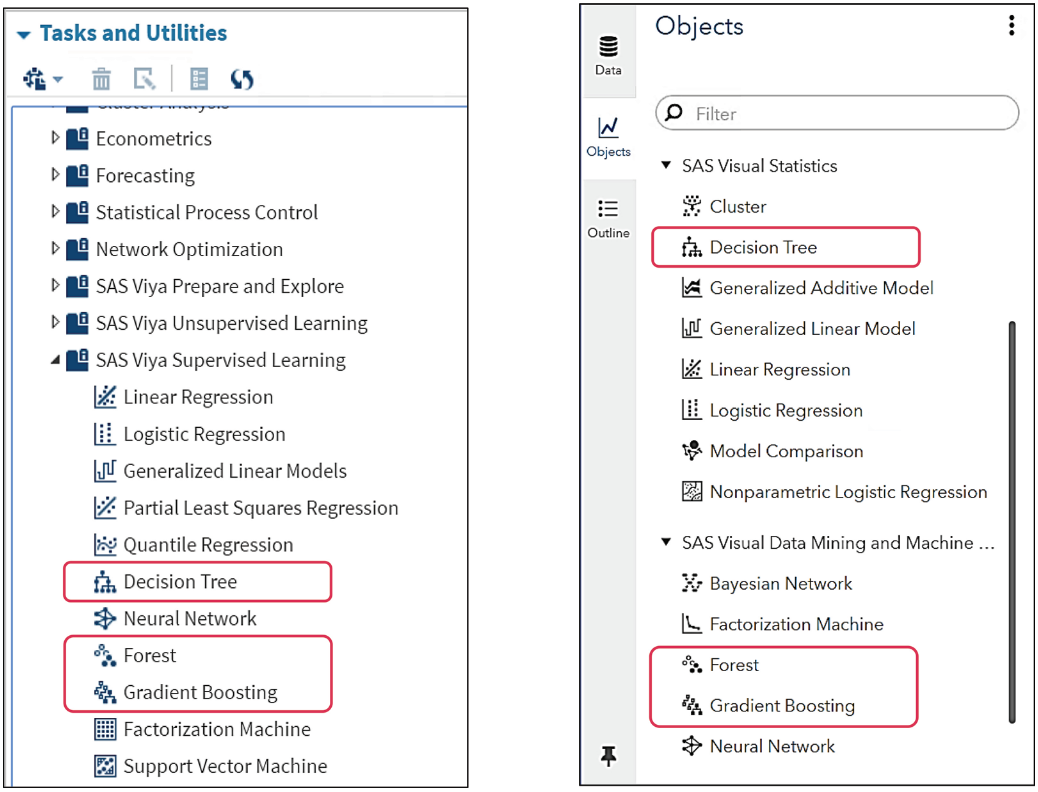
```
PROC GRADBOOST <options>;
        AUTOTUNE <options>;
        CODE <options>;
        CROSSVALIDATION <options>;
        ID variables;
        INPUT variables </ options>;
        OUTPUT OUT=CAS-libref.data-table <option>;
        PARTITION partition-option;
        SAVESTATE RSTORE=CAS-libref.data-table;
        TARGET variable </ LEVEL=NOMINAL | INTERVAL>;
        TRANSFERLEARN variable </ options>;
        VIICODE <options>;
        WEIGHT variable;
```

The PROC GRADBOOST, INPUT, and TARGET statements are required. The INPUT statement can appear multiple times.

Decision Tree, Forest, and Gradient Boosting Tasks and Objects

Shown in Figure 1.5 are SAS Studio tasks (left) and SAS Visual Analytics objects (right) relevant to tree-based models.

Figure 1.5: SAS Studio Tasks and SAS Visual Analytics Objects



SAS Studio is more than just an editor. It is familiar to SAS programmers who just want to write code – no point and click required to start writing in SAS. If you are not familiar with SAS code, SAS Studio includes visual point-and-click tasks that generate code so that you do not have to code. SAS Studio comes with code snippet libraries for frequently used operations, as well as interactive assistance for defining code that works.

> Decision trees are available in SAS Visual Analytics without adding SAS Visual Statistics. However, SAS Visual Statistics does augment the decision tree functionality. The decision tree in SAS Visual Statistics uses a modified version of the C4.5 algorithm. If SAS Visual Data Mining and Machine Learning is licensed at your site (and you have permission), the Forest and Gradient Boosting objects can be accessed under SAS Visual Data Mining and Machine Learning.

SAS Viya enables you to develop, deploy, and manage enterprise-class analytical assets throughout the analytics life cycle (data, discovery, and deployment) with a single platform with the underlying engine called CAS.

SAS Viya delivers a single, consolidated, and centralized analytics environment. Customers no longer need to stitch together different analytic code bases.

It natively supports programming in SAS and access to SAS from other languages such as R, Python, Java, and Lua. This means that data scientists and coders not familiar with SAS can use SAS Viya, but they do not need to learn SAS code.

It supports access to SAS from third-party applications with public REST APIs, so developers can easily include SAS Analytics in their applications.

Regardless of which interface is used, the same CAS actions are applied behind the scenes for the same procedure. This provides important consistency.

A *CAS action*, the smallest unit of functionality in CAS, sends a request to the CAS server. The action parses the arguments of the request, invokes the action function, returns the results, and cleans the resources.

Introducing Model Studio

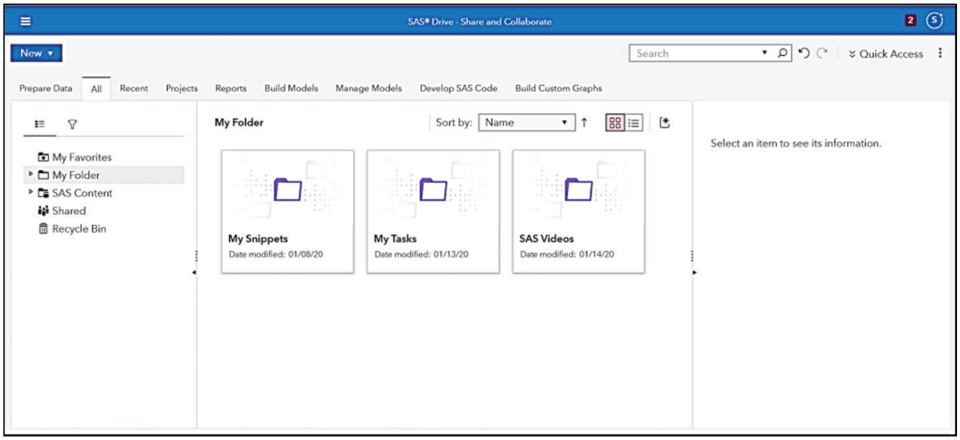
Model Studio enables you to explore ideas and discover insights by preparing data and building models. Model Studio is a central, web-based application that includes a suite of integrated data mining tools. The data mining tools supported in Model Studio are designed to take advantage of the SAS Viya programming and cloud processing environments to deliver and distribute data mining champion models, score code, and results.

Demo 1.1: Model Studio Introductory Flow

In this demonstration, you create a project and define metadata in Model Studio based on the **insurance_part** data set. This data is about a target marketing campaign for a bank that was undertaken to identify segments of customers who are likely to respond to a variable annuity (an insurance product) marketing campaign. The data set contains banking customers and 48 inputs that describe each customer. The 48 input variables represent other product usage in a three-month period and demographics. Two of the inputs are nominally scaled; the others are interval or binary. A binary target variable, **Ins**, indicates whether the customer bought the variable annuity product or not.

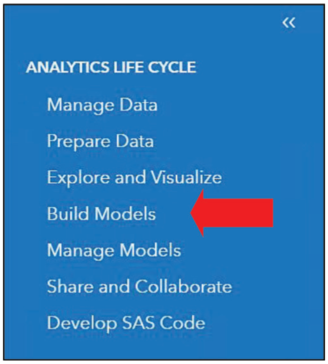
1. From the Windows taskbar, launch Google Chrome. When the browser opens, select **SAS Viya** ⇨ **SAS Drive** from the bookmarks bar or from the link on the page.
2. Log on using your user ID and password.
   Note: Use caution when you enter the user ID and password because values can be case sensitive.
3. Click **Sign In**.

4. Select **Yes** in the Assumable Groups window. The SAS Drive home page appears.

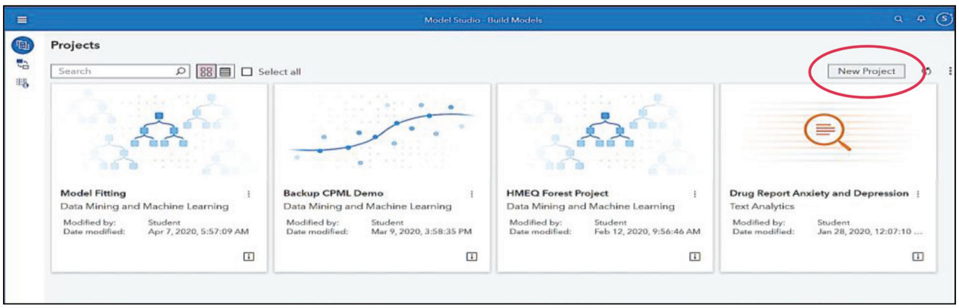


Note: The SAS Drive page on your computer might not have the same tiles as the image above.

5. Click the **Applications** menu (☰) in the upper left corner of the SAS Drive page. Select **Build Models**.

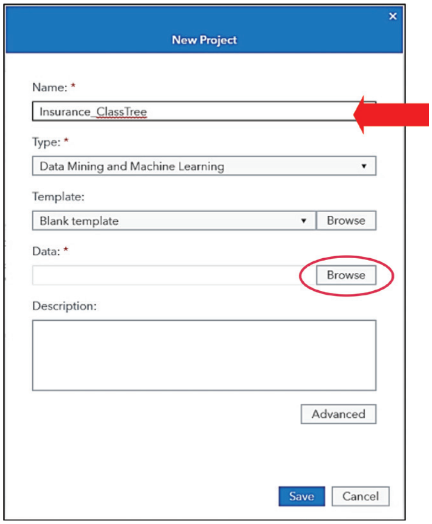


The Model Studio Projects page is now displayed.

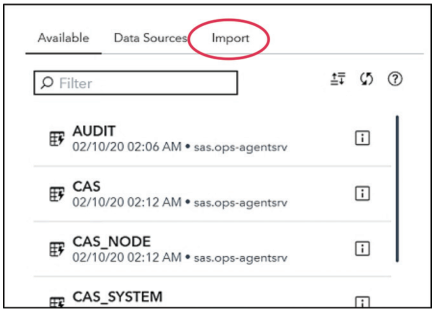


6. Select **New Project** in the upper right corner of the Projects page.

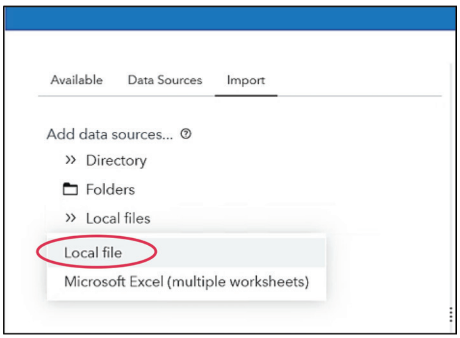
7. Enter **Insurance_ClassTree** as the name in the New Project window. Leave the default Type of **Data Mining and Machine Learning**.
   Select **Browse** in the **Data** field.



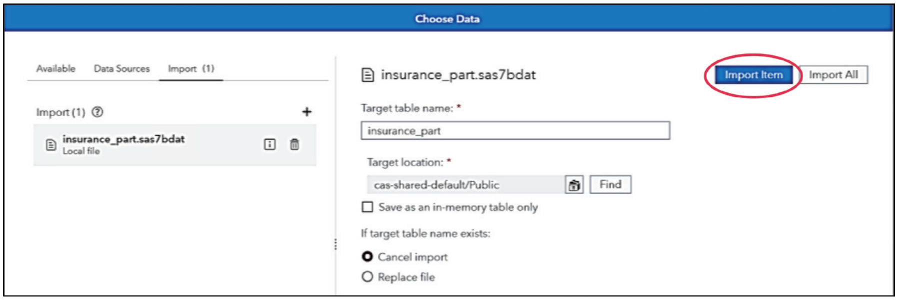
8. Import a SAS data set into CAS.
   a. In the Choose Data window, click **Import**.



   b. Under Import, expand **Local Files** and then select **Local File**.



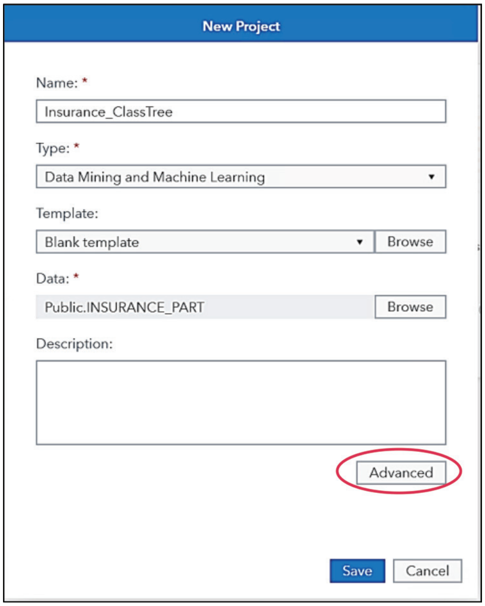
c. Navigate to the data folder.
d. Select the **insurance_part.sas7bdat** table. Click **Open**.
e. Select **Import Item**. Model Studio parses the data set and populates the window with data set configurations.



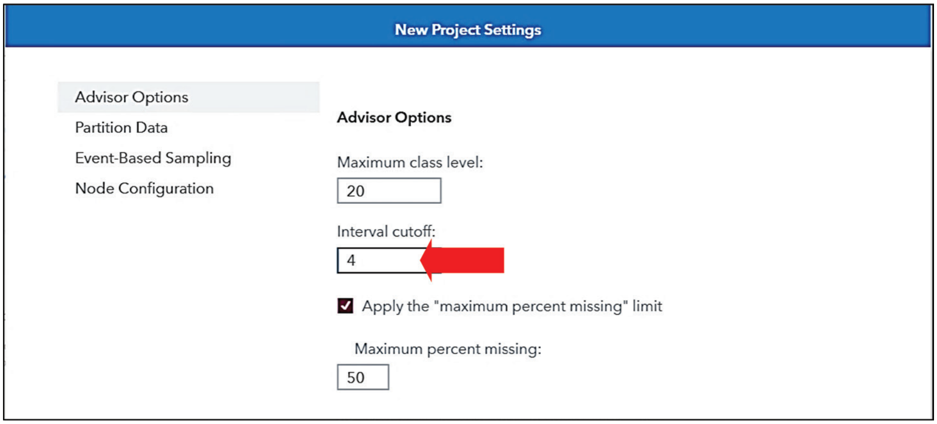
Note: When the data is in memory, it is available for other projects through the Available tab.

f. Click **OK** after the table is imported.

9. Click **Advanced** in the New Project window.

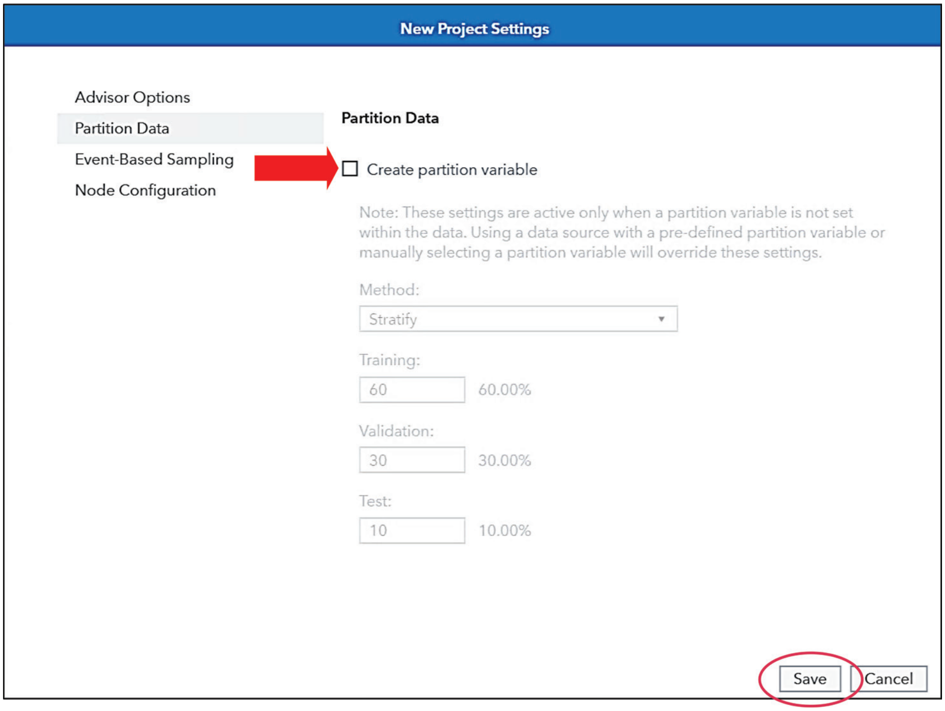


10. The Advanced project settings appear, and **Advisor Options** is one of the selections. Change **Interval cutoff** from 20 to **4**.



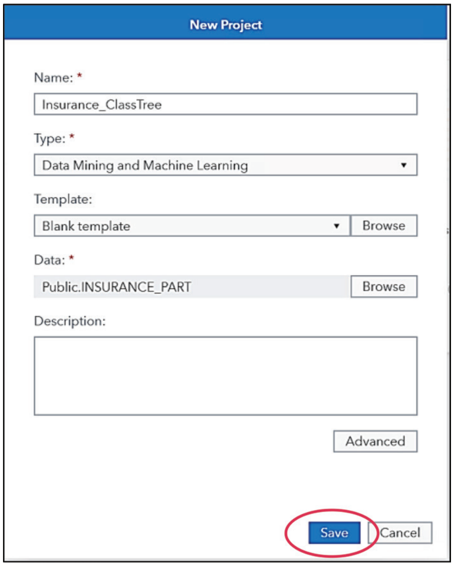
This is the threshold for a numeric input to be assigned as an interval variable, which means that if a numeric input has more than four distinct values or a nominal variable has more levels than four, it will be interval; otherwise, it will be nominal.

11. Click **Partition Data**. Uncheck the **Create partition variable** box as the data is already partitioned into training (70%) and validation (30%).

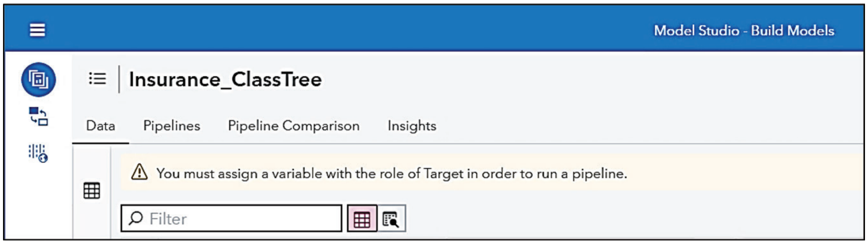


Click **Save** to return to the New Project window.

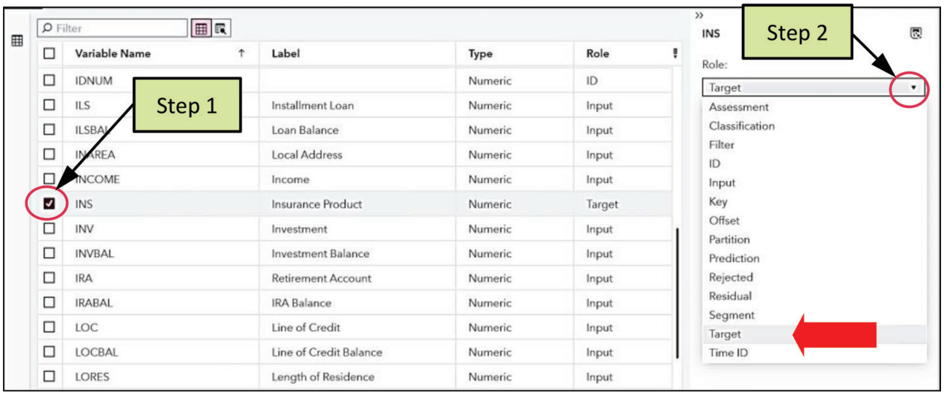
12. Click **Save** again.



When the project is created, you need to assign a target variable to run a pipeline.

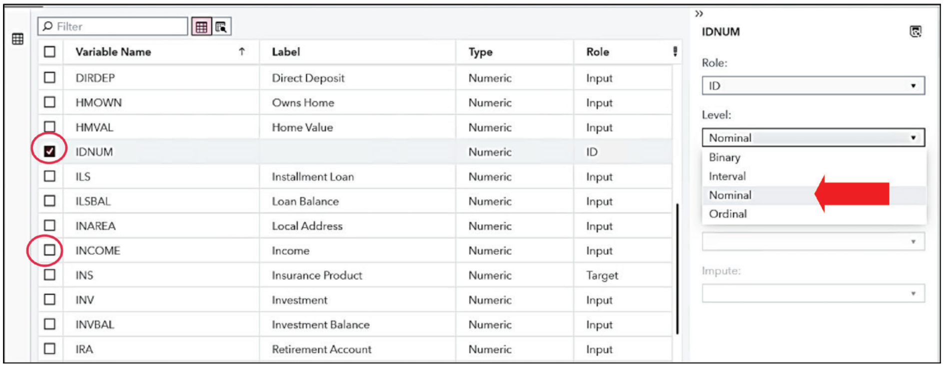


13. In the variables window, scroll down and select **INS** (Step 1). Then in the right pane, select **Target** under the **Role** property (Step 2).



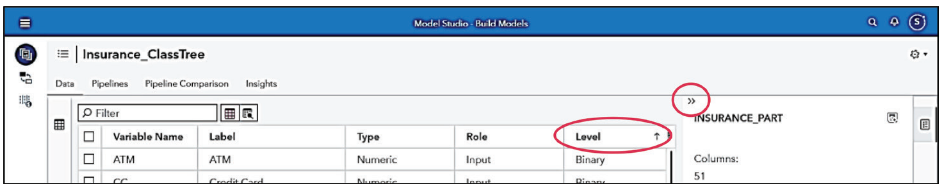
The right pane enables you to specify several properties of the variables, which includes **Role**, **Level**, **Order**, **Transform**, **Impute**, **Lower Limit**, and **Upper Limit**.

14. In the variables window, deselect **INS** and select **IDNUM**. In the right pane, select **Nominal** under the Level property.



IDNUM is a customer identification variable in which each customer has a unique value.

15. Deselect **IDNUM**. Click the **Level** column to sort as per variable roles. Hide the right pane by clicking the **>>** icon to be able to see the additional columns.



Columns' numerical details are shown.

Variable Name	Label	Type	Role	Level	Order	C...	Nu...	Missing	Minimum	Maximum	Mean
PartInd	Partition Indicator	Numeric	Partition	Binary	Default		2	0.0000	0.0000	1.0000	0.7000
ATM	ATM	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.6098
CC	Credit Card	Numeric	Input	Binary	Default		2	12.7344	0.0000	1.0000	0.4804
CD	Certificate of Deposit	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.1259
DDA	Checking Account	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.8144
DIRDEP	Direct Deposit	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.2957
HMOWN	Owns Home	Numeric	Input	Binary	Default		2	17.2444	0.0000	1.0000	0.5445
ILS	Installment Loan	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.0506
INAREA	Local Address	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.9599
INS	Insurance Product	Numeric	Target	Binary	Default		2	0.0000	0.0000	1.0000	0.3464
INV	Investment	Numeric	Input	Binary	Default		2	12.7344	0.0000	1.0000	0.0301
IRA	Retirement Account	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.0524
LOC	Line of Credit	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.0628
MM	Money Market	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.1156
MOVED	Recent Address Change	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.0290
MTG	Mortgage	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.0500
NSF	Number Insufficient Fund	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.0876
SAV	Saving Account	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.4636
SDB	Safety Deposit Box	Numeric	Input	Binary	Default		2	0.0000	0.0000	1.0000	0.1076
BRANCH	Branch of Bank	Character	Input	Nominal	Default		19	0.0000			
IDNUM		Numeric	ID	Nominal	Default		>254	0.0000	1.0000	19,357.0000	9,679.0000
RES	Area Classification	Character	Input	Nominal	Default		3	0.0000			

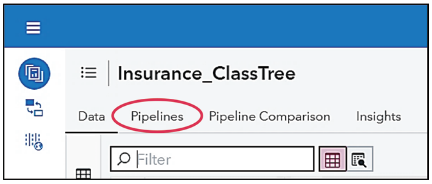
Focus on all the categorical variables first. All the binary variables are lumped together at the top and all the nominal inputs are lumped together at the bottom (scroll down to see). There are 17 binary inputs. Three of them contain missing values (**CC**, **HMOWN**, **INV**). Approximately 35% of the customers bought the insurance product. **BRANCH** has 19 levels, and **RES** has three (**Rural**, **Urban**, **Suburban**). Notice that you have a partition indicator variable, **_PartInd_**.

16. Readjust your scroll bar to see all the interval inputs now.

Variable Name	Label	Type	Role	Level	Order	Comment	Number of Levels	Missing	Minimum	Maximum	Mean
ACCTAGE	Age of Oldest Account	Numeric	Input	Interval	Default		>254	6.4111	0.3000	61.5000	5.8820
AGE	Age	Numeric	Input	Interval	Default		79	19.7241	16.0000	94.0000	47.9001
ATMAMT	ATM Withdrawal Amount	Numeric	Input	Interval	Default		>254	0.0000	0.0000	427,731.2600	1,262.8780
CASHBK	Number Cash Back	Numeric	Input	Interval	Default		4	0.0000	0.0000	4.0000	0.0155
CCBAL	Credit Card Balance	Numeric	Input	Interval	Default		>254	12.7344	-1,903.9900	1,593,806.7400	9,064.0477
CCPURC	Credit Card Purchases	Numeric	Input	Interval	Default		6	12.7344	0.0000	5.0000	0.1513
CDBAL	CD Balance	Numeric	Input	Interval	Default		>254	0.0000	0.0000	386,700.0000	2,431.6061
CHECKS	Number of Checks	Numeric	Input	Interval	Default		39	0.0000	0.0000	49.0000	4.2573
CRSCORE	Credit Score	Numeric	Input	Interval	Default		>254	2.0509	509.0000	820.0000	666.5161
DDABAL	Checking Balance	Numeric	Input	Interval	Default		>254	0.0000	-774.8300	278,093.8300	2,125.5386
DE		Numeric	Input	Interval	Default		>254	0.0000	0.0000	16,000.0000	5,197.1115
DEP	Checking Deposits	Numeric	Input	Interval	Default		19	0.0000	0.0000	28.0000	2.1261
DEPAMT	Amount Deposited	Numeric	Input	Interval	Default		>254	0.0000	0.0000	359,618.4700	2,236.7032
HMVAL	Home Value	Numeric	Input	Interval	Default		173	18.0142	67.0000	754.0000	110.6684
ILSBAL	Loan Balance	Numeric	Input	Interval	Default		>254	0.0000	0.0000	24,634.6300	528.2789
INCOME	Income	Numeric	Input	Interval	Default		192	18.0142	0.0000	233.0000	40.3670
INVBAL	Investment Balance	Numeric	Input	Interval	Default		>254	12.7344	-2,214.9200	8,323,796.0200	1,996.8655
IRABAL	IRA Balance	Numeric	Input	Interval	Default		>254	0.0000	0.0000	285,339.7700	537.9694
LOCBAL	Line of Credit Balance	Numeric	Input	Interval	Default		>254	0.0000	-71.6100	523,147.2400	1,204.2149
LORES	Length of Residence	Numeric	Input	Interval	Default		39	18.0142	0.5000	19.5000	7.0244
MMBAL	Money Market Balance	Numeric	Input	Interval	Default		>254	0.0000	0.0000	68,965.9200	1,884.2266
MMCRED	Money Market Credits	Numeric	Input	Interval	Default		6	0.0000	0.0000	5.0000	0.0574
MTGBAL	Mortgage Balance	Numeric	Input	Interval	Default		>254	0.0000	0.0000	1,628,532.3800	7,626.0502
NSFAMT	Amount NSF	Numeric	Input	Interval	Default		>254	0.0000	0.0000	666.8500	2.3672
PHONE	Number Telephone Banking	Numeric	Input	Interval	Default		19	12.7344	0.0000	30.0000	0.4040
POS	Number Point of Sale	Numeric	Input	Interval	Default		35	12.7344	0.0000	54.0000	1.0747
POSAMT	Amount Point of Sale	Numeric	Input	Interval	Default		>254	12.7344	0.0000	2,608.4300	48.6339
SAVBAL	Saving Balance	Numeric	Input	Interval	Default		>254	0.0000	0.0000	609,587.7200	3,138.5250
TELLER	Teller Visits	Numeric	Input	Interval	Default		26	0.0000	0.0000	25.0000	1.3512

Of the 29 interval variables, 12 contain missing values. Unlike parametric models like regression and neural network, decision trees handle missing values quite well. This is discussed later in the book.

17. Click the **Pipelines** tab in the **Insurance_ClassTree** project.



A blank template pipeline was created with only a Data node.

End of Demonstration

Quiz

1. Model Studio is a central, web-based application that includes a suite of integrated data mining and machine learning tools.
   a. True
   b. False

2. Decision trees can be created for nominal targets as well as for the interval targets.
   a. True
   b. False

3. Which of the following approaches can you use to build decision trees and tree-based models in SAS Viya? (Select all that apply.)
   a. Python, Java, Lua, and R languages
   b. CASL language
   c. SAS procedure wrappers
   d. GUI-based applications like Model Studio and SAS Visual Statistics

Answers

1. True **2.** True **3.** a, b, c, and d.