Classification and Regression Trees 10

It is very difficult to make a vigorous, plausible, and job-risking defense of an estimate that is derived by no quantitative method, supported by little data, and certified chiefly by the hunches of the managers.

—Frederick P. Brooks

Classification and Regression Trees (CART) represents a data-driven, model-based, nonparametric estimation method that implements the define-your-own-model approach. In other words, CART is a method that provides mechanisms for building a custom-specific, nonparametric estimation model based solely on the analysis of measurement project data, called training data.

In general, CART belongs to a wider group of machine learning methods that deal with building decision trees. The most well-known classification tree method is C4.5[1] proposed by Quinlan (1992, 1996).

10.1 Principles

10.1.1 The CART Model

Basically, a decision tree (Fig. 10.1) forms a stepwise partition of the data set on which it was developed. Each node of a tree specifies a condition based on one of the predictor variables. Each branch corresponds to possible values or range of values this variable may take.

The paths through the decision tree leading from the root to terminal nodes represent a collection of decision rules in the form:

$$IF\ (Condition_1\ AND\ Condition_2\ AND\ ...AND\ Condition)\ THEN\ Decision$$

[1] The most recent version of the Quinlan's C4.5 method is called C5.0 and is not revealed to the public. It is implemented in the commercial proprietary software tool named RuleQuest (http://www.rulequest.com/).

A. Trendowicz and R. Jeffery, *Software Project Effort Estimation*,
DOI 10.1007/978-3-319-03629-8_10,

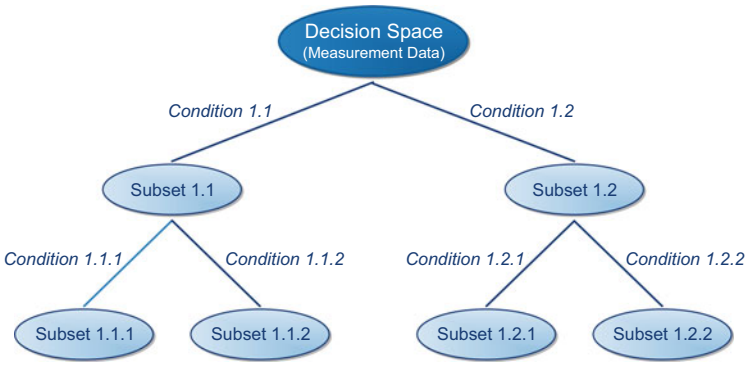

Fig. 10.1 Structure of a decision tree

where decision represents the value of a predicted variable determined on the basis of data in the particular terminal node. For example, in the example tree presented in Fig. 10.1, the example decision rule would be:

```
IF (Condition1.1 AND Condition1.1.2) THEN Subset1.1.2
```

which would indicate that for a new project that meets *Condition1.1* and *Condition1.1.2* on its independent variables (effort drivers), the value of the dependent variable (effort) should be predicted using the historical projects in the *Subset1.1.2*; for example, effort estimate for the new project can be computed as mean over actual effort of projects in the *Subset1.1.2*.

Traditionally, decision tree methods were limited to either regression-type problems or classification-type problems. In the regression-type problem, a decision tree is created for predicting the values of a continuous dependent variable based on the one or more continuous and/or categorical predictor variables. In the classification-type problem, a decision tree is developed for predicting the values of a categorical dependent variable based on the one or more continuous and/or categorical predictor variables.

Proposed by Breiman et al. (1984), the CART method overcomes this limitation and can deal with both, regression and classification problems, meaning it deals with any mixture of continuous and categorical predictor and predicted variables.

10.2 Usage Scenarios

10.2.1 Developing the CART Model

The CART develops classification and regression (C&R) tree models in two phases:

- *Generating Tree*: The tree model is developed based on the training data, which consists of measurement data from already completed (historical) projects. Generating the model involves recursively splitting the data set until certain

stop criteria are satisfied. In CART, univariate binary splits are considered, that is, each split depends on the value of one predictor variable (*univariate*) and results in two partitions[2] (*binary*).

- *Pruning Tree*: In order to avoid overfit, the model resulting from the "Generating tree" phase needs to be adjusted. The tree-generating process typically results in a "maximal" model, meaning a model that is very specific to the characteristics of the training set. In order to generalize the model for projects outside the training set, that is, for predicting new projects, an optimal sub tree of the maximal tree needs to be identified. In the pruning phase, such a sub tree is identified, and the maximal tree is reduced by cutting off its overfitted parts appropriately.

Generating Tree

The CART tree-generation procedure consists of several abstract steps[3]:

1. *Binary Splitting*: Starting with the first variable in the training data set, a variable is split at all of its possible split points. A variable's *split points* are determined by values the variable takes in the training set. For example, for a categorical variable, split points are defined by all categories the variable takes in the training set. At each possible split point of the variable, the project data sample is partitioned into two subsets represented by two nodes creating left and right branches in a tree. A single split is defined by a condition in the form: *IF Variable = Value THEN left branch ELSE right branch*. Notice that in CART, splits can also be based on linear combinations of variables.
2. *Evaluating Split*: Each possible split is evaluated using a "goodness of split" criterion. The *goodness of split* is quantified as reduction of sub node's *impurity* relative to its parent node. The node's impurity refers to the heterogeneity level in the distribution of the predicted variable's values observed in the node.
3. *Selecting Variable Split*: The best split on the variable in terms of the goodness of split (e.g., highest reduction in the partition's impurity) is selected, and the appropriate two subsets (branches) are created in the tree.
4. *Repeating Splitting*: Steps 1–3 are repeated for each of the remaining variables at the root node of the tree, that is, for each remaining variable in the training set.
5. *Ranking Splits*: All of the "best" splits on each variable are ranked according to the reduction of the node's impurity.
6. *Selecting Best Split*: The best split across all variables is selected based on the ranking results.
7. *Assigning Classes*: Tree nodes associated with the best split selected in the previous step are assigned classes according to a rule that minimizes

[2] Although CART generates binary partitions, other approaches have also been proposed for generating decision trees (Porter and Selby 1990).

[3] The detailed description of the procedure for building a C&R tree is beyond the scope of this book and can be found in Breiman et al. (1984).

misclassification costs.[4] In other words, each node resulting from the best split is assigned a decision regarding the value of the predicted variable (class). For each potential value of the predictor variable, a prediction error of the resulting tree model (misclassification cost) is computed. The value that results in the lowest prediction error is then assigned to the node.

8. *Repeat Recursively*: Steps 1–7 are repeated for each nonterminal tree node until the tree is complete, that is, until it cannot be grown any further.

Pruning a Tree

One of the major issues that arises when building C&R tree models is how to avoid their overfit. Overfitting generally occurs when an excessively complex, "maximal" model is generated to adjust it exactly to the characteristics of a specific training set. In the extreme case, a tree model encompasses all variables provided in the training set, and each terminal node includes only one project. Such a "maximal" model, although performing perfectly for projects in the training set, is typically not able to accurately predict projects from beyond the training set, yet similar to training projects. In such a case, it is said that the model does not generalize to situations not represented in the training data. In general, a model is considered to be overfitted relative to a simple one if it is more accurate in predicting known projects but less accurate in predicting new projects.

A major strategy to avoid overfit of a tree model is called *pruning*, and it can be implemented in two ways:

- *In-Process Pruning*: The tree-growing process is stopped in the middle based on a set of pruning criteria. In this approach, the node-splitting process is allowed to continue until certain stop criteria are met. An example instantiation of this idea is to continue the tree-generation process until subsequent splits (1) result only in a very little overall improvement in the tree's prediction performance or (2) contain fewer projects than the user-specified minimum threshold.
- *Post-Process Pruning*: The tree-growing process is allowed to continue until a maximal tree is built and a pruning procedure is applied on it afterward. The CART method actually uses this pruning strategy. At first, CART allows for growing a maximal possible tree. Then it examines sub trees obtained by pruning away branches of the maximal tree in order to select the one that is most likely to perform best on new projects. CART avoids stopping in the middle of the tree-growing process in order not to miss important information that might be discovered at lower levels of the tree.

Building a decision tree is a computationally intensive process and is laborious to perform manually even for small data sets. Therefore, a number of software tools to support building and applying CART exist.

[4] The CART method uses special algorithms for minimizing misclassification costs.

Software Tools Supporting the CART Method

Although the CART method has been revealed to the public (Breiman et al. 1984), "CART" is a trademarked name reserved for the Salford Systems software tool that implements the original CART method (http://www.salford-systems.com/). However, several other data analysis software tools have reimplemented the CART method. Examples include research tools such as CARTX (Srinivasan and Fisher 1995), as well as commercial tools such as Statistica (http://www.statsoft.com/), where it was implemented under the name GC&RT.

10.2.2 Applying the CART Model

A CART model can be applied for a number of purposes related to software effort estimation. On the one hand, a decision tree can be used to predict the effort of a new project. On the other hand, the tree can be analyzed in order to identify the most important factors and their dependencies affecting development effort for the purpose of planning the project and process improvement actions. Let us discuss these aspects of applying CART on a simple example.

Example 10.1. Predicting the Productivity of ESA Projects with CART

Let us consider an example decision tree in Fig. 10.2. It was developed by Briand et al. (1998) for estimating the productivity of software development projects in the context of the European Space Agency (ESA).

Estimating Productivity and Estimating Effort

In principle, it is beneficial to consider development productivity instead of project effort when utilizing CART for learning about the factors influencing a project's performance. In practice, software size is the main determinant of development effort. Therefore, the size will typically (if not always) be identified as the "top" most important factor influencing development effort. Yet, as we might be interested in the influence of other project factors on effort, we need to exclude size from the analysis. One option is to build a decision tree for multiple projects of the same size—yet this is difficult because projects are typically of different sizes. Another option is to build a decision tree for software productivity, which already incorporates effort in relation to size: $Productivity = Size/Effort$. In this case, we are interested in factors that make productivity differ across analyzed development projects. These are the factors that make the effort required for completing software of the same size differ in different project environments.

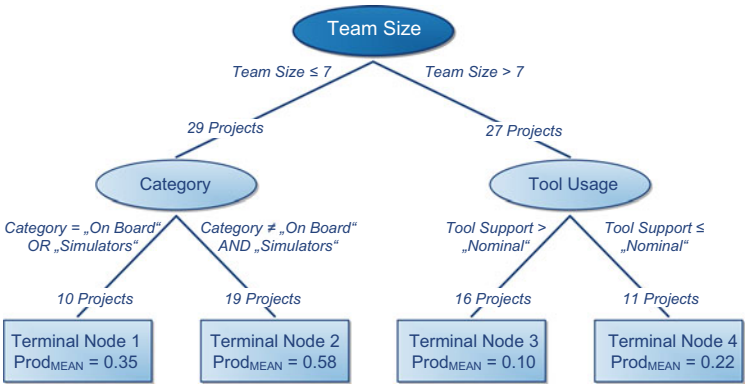

Fig. 10.2 Example decision tree

Each terminal node in the example tree represents the average productivity of projects that are characterized by the path from the root node. Each node represents an individual project factor, upon which *If–Then* decision criteria are defined. The tree represents binary splits, that is, for each node there are two outgoing arcs that represent splitting the underlying project data into two disjunctive sets based on two conditions. Depending on which condition is true for a particular project, the left or the right path is taken.

On the first level of the tree, projects are first split according to their "Team Size." If the team size is lower or equal to seven persons, then these projects are split further according to their *Category*; otherwise, projects are split further according to the "Tool Usage." In the context of ESA, in 29 projects the team size was lower or equal to seven persons, whereas in 27 projects, team size was greater than seven persons.

Following the *Category* branch: Ten projects falling into the "On board systems" or "Simulators" category have an average productivity of 0.35 kLOC/PM (thousand lines of source code per person-month). The remaining 19 projects falling in other categories have an average productivity of 0.58 kLOC/PM.

Following the *Tool Usage* branch: Eleven projects where tool usage is between "Low" and "Nominal" have a predicted average productivity of 0.10 kLOC/PM, whereby "Nominal" tool usage means no tool or basic lower CASE tools. The 16 projects with higher than "Nominal" tool usage—that is, with extensive use of upper CASE, project management, and documentation tools—have predicted an average productivity of 0.22 kLOC/PM.

***Prediction*:** Productivity of a new project can be estimated by classifying the project into one of the tree terminal nodes. For this purpose, one would go through the decision tree starting at its root node and selecting its subsequent branches based on the project's specific factor value, until a terminal node is reached. The mean productivity computed for the historical projects in the node can then be used as the predicted productivity of the new project. Other statistics

such as median, quartiles, or standard deviation can be computed additionally for each node as an indicator of potential uncertainty associated with the mean productivity estimate. Notice that building a decision tree for the purpose of prediction requires that the data on the factors considered in the tree model can be acquired at the beginning of a project—otherwise a new project cannot be estimated because the information on factors required as input for the tree model are not known at the time of estimation.

***Risk Management and Performance Improvement*:** The decision tree allows for drawing some conclusions about the relative importance of factors (predictor variables) covered by the training data set. Factors selected on top levels of the tree hierarchy are typically more important than factors below them, when considered individually. In the example tree (Fig. 10.2), the size of a development team was selected as the most relevant factor influencing development productivity. This information can, for instance, be used to avoid short-term project risks and plan long-term process improvements. For example, large projects should be split into smaller ones in order to ensure optimal team size and, in consequence, optimal development productivity.

Furthermore, decision trees allow, to some extent, for learning about reciprocal dependencies between factors. This is accomplished by analyzing paths in a tree model. Selection of the factor to be considered in a particular node, on a certain tree level, depends on the value of factors considered on the path from the tree root to this node. In the example tree presented in Fig. 10.2, factors considered on the second level of the tree depend on the size of the development team. For projects with a team larger than seven persons, usage of tools plays a more important role than for projects with a team up to seven persons; for the small projects, the category of a project is a more important factor influencing development productivity than tool support. This information can be an additional guide for risk management and improvement activities. For instance, if it is not feasible to split large projects into smaller ones to ensure optimal performance, then one may consider alleviating the negative effect of a large team on productivity by ensuring appropriate tool usage.

***Process Improvement*:** The goodness of a decision tree can be evaluated by comparing predicted against actual project data. For example, after classifying a project into a particular terminal node, one may determine whether a project's productivity varies significantly from the node's mean productivity value, where mean corresponds to a typical productivity for the class of projects represented by this very node. If this is the case, potential reasons for deviation should be investigated. It may, for example, occur that some relevant factor that makes the productivity of the new project differ from historical projects has not been measured so far and, thus, has not been considered in the decision tree. This would call for improving measurement processes and revising the tree model. ■

10.3 Strengths and Weaknesses

In principle, the CART method shares, to a large extent, general characteristics with other data-driven, model-based estimation methods, for example, in that it requires significant amounts of quantitative data. Table 10.1 summarizes the most relevant strengths and weaknesses specific to the CART method.

Table 10.1 Strengths and weaknesses of CART

Aspect	Strengths (+) and weaknesses (–)
Expert involvement	• (+) CART does not require much expert involvement. It requires an expert knowledgeable in using the method and interpreting its outputs • (+) Once a CART model is created, its use for estimation purposes is reduced to providing characteristics of the new project on its input and interpreting output (estimates) it provides • (–) Developing a CART model requires expertise in setting up the method's parameters
Required data	• (–) CART requires quite large amounts of data in order to produce reliable outputs • (–) The more effort drivers that are considered and the more different values they have across historical projects, the more historical data is required by CART to reliably "learn" effort dependencies represented by the data—i.e., to develop a reliable estimation model
Robustness	• (+) CART handles missing data • (+) CART makes no distributional assumptions of any kind, either on dependent or independent variables. No variable in CART is assumed to follow any kind of statistical distribution • (+) The explanatory variables in CART can be a mixture of categorical, interval, and continuous • (+) CART is not at all affected by outliers, collinearities, heteroscedasticity, or distributional error structures that affect parametric estimation methods such as regression analysis. Outliers are isolated into a node and do not have any effect on splitting. Contrary to parametric methods, CART handles and makes use of collinear data • (+) CART is invariant under monotone transformation of independent variables; i.e., the transformation of explanatory variables to logarithms or squares or square roots has no effect on the tree produced • (+) CART deals effectively with multidimensional project data; i.e., from a large number of project characteristics provided on CART's input, it can produce useful results using only a few characteristics that are most relevant from the perspective of project effort
Flexibility	• (+) CART does not require any specific information on its input • (+) Once developed, a CART model can be used multiple times for estimation without the necessity of accessing the base of historical projects • (+) A CART model can be easily rebuilt for a new situation using appropriate project data—under the condition that modeling is supported by a software tool
Complexity	• (–) CART has a complex theoretical basis; it uses elements of statistics and information theory • (–) CART requires setting up a few parameters, the meaning of which needs to be known by the person operating CART

(continued)

Table 10.1 (continued)

Aspect	Strengths (+) and weaknesses (–)
Support level	• (+) The CART method is well documented in multiple sources such as books and articles • (+) CART is supported by several software tools
Reusability	• (+) Estimation models created with CART are reusable for similar situations • (+) The CART model can be easily rebuilt for a new situation using appropriate project data—under the condition that appropriate tool support is available
Predictive power	• (+) CART produces the same outputs when provided with the same inputs • (–) The predictive power of CART depends on the similarity of the estimated project to historical projects on which the CART model was developed • (–) CART cannot estimate a value along a dimension that is outside the range of values encountered in the underlying data set (extrapolation)
Informative power	• (+) CART provides significant insight into project effort dependencies. For example, it has the ability to detect and reveal interactions between effort drivers represented in the input project data • (+) A CART tree allows for making conclusions about the relative importance of effort drivers. Effort drivers selected on top levels of the tree are considered more important than effort drivers below them • (–) Yet, binary trees do not allow for considering multiple factors in a single tree node. It might happen that two factors that are considered as less relevant when considered individually might be very important when considered in combination. This weakness can be solved by applying additional techniques for exploring relationships between effort drivers • (–) CART cannot reveal any effort relationships beyond that represented by available measurement data
Handling uncertainty	• (–) CART provides little support for handling estimation uncertainty. It does not accept uncertain inputs. Estimation uncertainty can be assessed using the distribution of effort across historical projects in the tree leaf in which the estimated project has been classified
Estimation scope	• (+) CART is, principally, applicable to any type of project activities at any granularity level. The only requirement for estimating effort associated with particular project activities is that historical data regarding these very activities are available for building the CART model
Availability	• (+) The availability of the estimates provided by CART depends on the availability of information on effort drivers covered by the CART model. For example, if a CART model requires information on effort drivers that are available late in the development cycle, it cannot be used for estimation before appropriate measurements can be collected. In order to estimate earlier, effort drivers need to be estimated. Yet, as CART does not support uncertain inputs, using estimates as a basis for estimation may lead to highly unreliable effort predictions
Empirical evidence	• (+) CART has been empirically validated in a number of field studies for effort estimation and predictions in other areas

Further Reading

- L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen (1984), *Classification and Regression Trees*. Chapman & Hall.

  This book provides a detailed overview of the theory and methodology of CART.