# Forecasting Supreme Court decisions

Tool: Classification And Regression Trees (CART), Random Forests.

The Analytics Edge: Political scientists and legal academics constantly seek to understand what motivates justices. In 2002, a group of academics predicted the decisions of the U.S. Supreme Court using information on past decisions and compared them to the expert judgments of legal academics and professionals (Martin et al., 2004). Using an interpretable analytics technique known as Classification And Regression Trees, the model got 75.0% of the court's affirm/reverse results correct, while experts collectively got 59.1% right. At the individual justice level, the model got 66.7% decisions correct while experts got 67.9%. Using analytics can provide an edge in a traditionally qualitative application. At the individual justice level, the experts got better predictions on the more "predictable" justice votes, while the model did a better job at the more "unpredictable" justice votes. Combining the two approaches—the consistent and unemotional nature of a model with the intuition and knowledge of experts—might be a reasonable approach in this case.

# Overview (Supreme Court of the United States)

The Supreme Court is the highest court in the federal judiciary of the United States. It consists of a chief justice and eight associate justices, which are nominated by the U.S. President and confirmed by the Senate. Once appointed, judges have lifetime tenure, unless they retire, resign, or are removed from office. Each justice has one vote. Because of the importance and nature of their decisions—many high-profile cases expose ideological beliefs—justices are often categorized as conservative, moderates, or liberal based on their votes in previous cases. The supreme court decisions can impact a variety of social, economic, and structural questions. In 2002, some of the important cases were:

1. *Grutter vs Bollinger.* This case dealt with the constitutionality of affirmative action (i.e., the policy of favouring members of a disadvantaged group who suffers from discriminations). This case was upheld by the Supreme Court ruling that the affirmative action policy of the University of Michigan law school was justified. The case was affirmed 5–4.

2. *Lawrence vs Texas.* This case dealt with the right to engage in consensual homosexual sodomy. In a 6–3 ruling, the court reversed and struck down the sodomy law in Texas making same-sex sexual activity legal in the U.S.

Many of the court decisions are hard to predict. Martin et al. (2004) predicted the 2002 term court decisions using simple predictor variables. The model they adopted is indifferent to many of the specific legal and factual aspects of the case that legal experts might use when making predictions. In particular, Martin et al. (2004) used data for the period 1994–2001 (one of the longest periods of time with the same justices) to predict 2002 decisions. The dataset contained the following predictors (see Figure 0.1): (1) the circuit of origin for the case, (2) the issue area of the case, (3) the type of petitioner, (4) the type of respondent, (5) the ideological direction of the lower court ruling, and (6) whether or not the petitioner argued the constitutionality of a law or practice. Further details about the *The Supreme Court Forecasting Project* are available at `http://wusct.wustl.edu`.

Key Question: Is it possible to predict whether the Supreme Court judges will affirm or reverse the lower court decisions (at the individual judge level, and at the overall case level) using simple analytics?

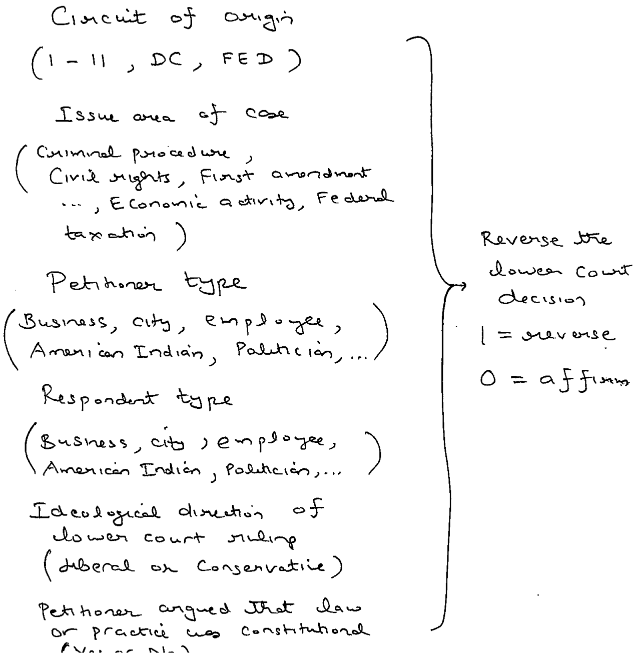
Circuit of origin
(1 - 11, DC, FED)
Issue area of case
(Criminal procedure, Civil rights, First amendment ..., Economic activity, Federal taxation)
Petitioner type
(Business, city, employee, American Indian, Politician, ...)
Respondent type
(Business, city, employee, American Indian, Politician, ...)
Ideological direction of lower court ruling
(Liberal or Conservative)
Petitioner argued that law or practice was constitutional
(Yes or No)
Reverse the lower court decision
1 = reverse
0 = affirm

Figure 0.1: Overview of the data collected by Martin et al. (2004).

## Summary

Data: The data contain information on the cases and corresponding decisions made by the U.S. Supreme Court during the period 1994–2001.

Model: CARTs and Random Forests can be used to forecast the decisions made by the Supreme Court. For each case, the decision is either affirm or reverse, so the model output is a binary variable.

Value and Decision: the model predictions could be used in litigants' decisions to press an appeal and in settlement negotiations.

# Classification And Regression Trees (CART)

## Introduction

Suppose you want to predict a value, or response, $y$ from predictors $x_1, \dots, x_p$. Model such as linear regression or logistic regression are global models in which the single predictive formula is supposed to hold over the entire data region (with linear dependence between dependant and independent variables). One might be able to use nonlinear regression techniques if the data interact in complex, nonlinear ways. However, these models are often less interpretable.

An alternative approach is represented by parametric estimation. The idea is to divide (partition) the prediction space into a number of simple regions, and then use simple rules—such as the mean or mode of the observations—in the space partitioned to make predictions. By partitioning the sub-regions again (recursive partitioning), we can develop regions where the models within regions are very simple. In the example given in Figure 0.2, the result of the partitioning process can be summarized as follows:

Check if $x_1 \leq t_1$:

- If yes, check if $x_2 \leq t_2$;
  - If yes, we are in region $R1$.
  - If no, check if $x_1 \leq t_3$:
    - If yes, we are in region $R_4$;
    - If no, we are in region $R_5$.
- If no, check if $x_2 \leq t_4$:
  - If yes, we are in region $R_2$;
  - If no, we are in region $R_3$.

Another example is illustrated in Figure 0.3. Here, the concept is applied to a classification problem, that is, to predict $x$ or $\circ$ based on the splits and the majority number in each split.

As we shall see, CARTs have three main advantages:

- The model is easily interpretable—even to non-experts;
- The model does not assume a linear relationship between input and output variable, and hence can capture nonlinearities in the data (see the illustration in Figure 0.4);
- During the tree construction process, the model uses only the input variables that are necessary, thereby easing the model selection process.

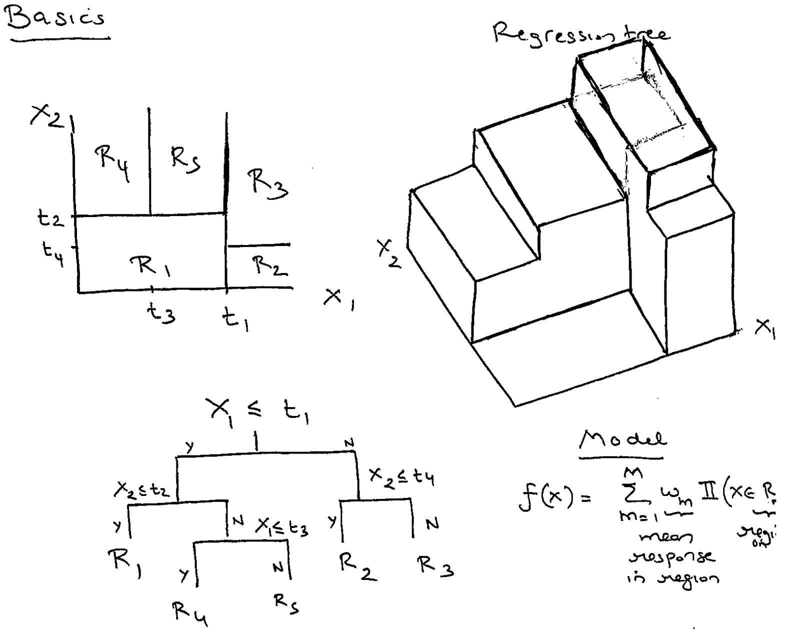

Figure 0.2: CART's recursive partitioning approach for a regression problem.

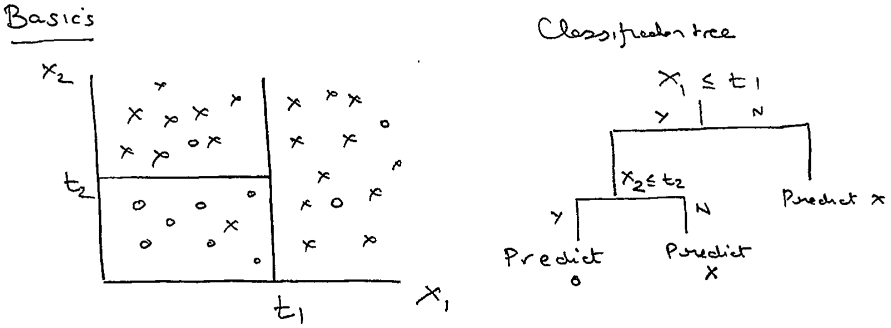

Figure 0.3: CART's recursive partitioning approach for a classification problem.

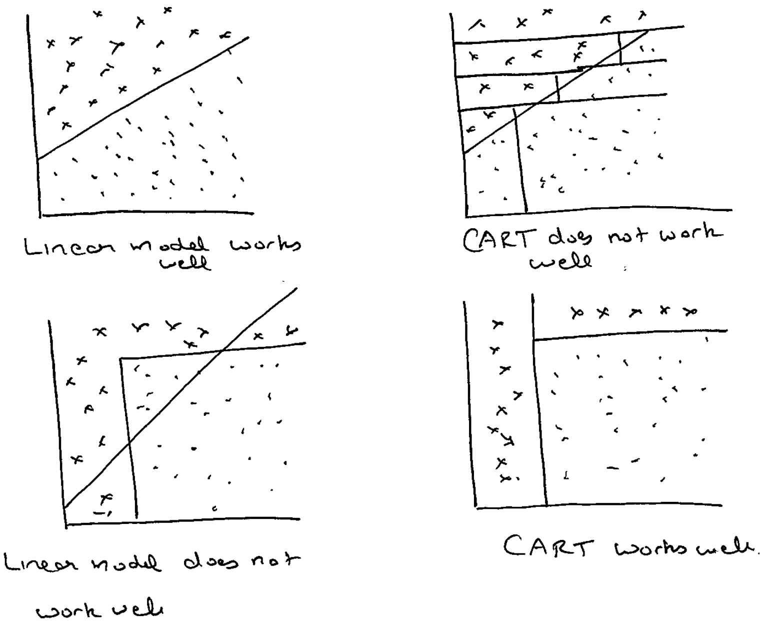

Figure 0.4: Comparison between linear model and CART.

## Learning process (regression)

To create a tree, we want to divide the space of predictor variables $x_1, \ldots, x_p$ into $M$ distinct and non-overlapping regions $R_1, \ldots, R_M$. In CART, these regions are defined by rectangles (boxes), where the cuts are parallel to the axis. From an optimization standpoint, we can define the problem as follows:

$$\min_{R_1,\ldots,R_M} \sum_{m=1}^{M} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2,$$

where $R_1, \ldots, R_M$ are the regions partitioning the space (with $R_i \cap R_j = \emptyset$, $\forall i \neq j$), $y_i$ is the $i$-th observation in $R_m$, and $\hat{y}_{R_m}$ is the mean value of all observations in $R_m$. This is a difficult optimization problem.

We are thus forced to use local search procedures based on heuristics that create reasonable trees in reasonable time. A well-known greedy strategy was introduced by Breiman (1984), a distinguished statistician at UC Berkeley. The strategy he proposed, which is the main building block of CARTs, works as follows:

- Start with all observations in a single region;
- Select the predictor variable $x_j$ and cut point $s$ such that splitting into region $\{x|x_j \geq s\}$ and $\{x|x_j < s\}$ results in the smallest Residual Sum of Squares (RSS):

  $$\min_{1,\ldots,p} \min_{s} \sum_{i: x_{ij} < s} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_{ij} \geq s} (y_i - \hat{y}_{R_2})^2,$$

  where $R_1 = \{x|x_j < s\}$ and $R_2 = \{x|x_j \geq s\}$. Once this problem is solved (by exhaustive search), we find $j^*$ and $s^*$. We use this as the first split.
- Now, we repeat this step for each sub-region, individually. We continue until a stopping criterion is met—such as not having too few observations in a region.

Note that all observations belong to a single sub-region beyond which greedy splits are made at each step without looking necessarily at the best split that might lead to a better tree in future steps (greedy strategy).

## Learning process (classification)

A classification tree is built in almost the same manner as a regression tree, except that the measure used to select predictor variable and cut point (for each decision node) is replaced by a measure of impurity. A split is pure if, after the split, for all branches, all the instances choosing a branch belong to the same class. Some of the most common measures are:

1. Gini Index $= \sum_{k=1}^{K} p_{mk}(1 - p_{mk})$;

where $p_{mk}$ is the proportion of training observations in the $m$-th region from the $k$-th class.

2. Entropy (or Information Gain) $= -\sum_{k=1}^{K} p_{mk} \cdot \log(p_{mk})$.

These two indices take a value near 0, as long as the $p_{mk}$ values are close to 0 or 1 (see Figure 0.5). *rpart*—the most common R package for CARTs—uses the Gini Index for splitting. Another related measure is:

3. Classification error rate $= 1 - \max_{k} p_{mk}$,

which assumes that the class with maximum proportion is to be predicted. Note that, in CART models, splitting on qualitative predictor variables is straightforward too.

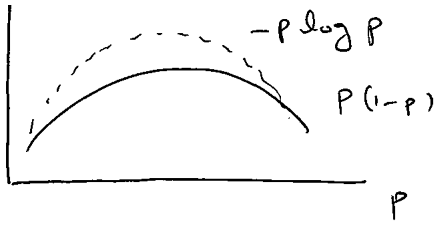

Figure 0.5: Comparison between Gini Index and Entropy.

In some splits for CART models, you might end up with a situation like the one illustrated in Figure 0.6, where the bottom split seems superficial. However, note that such a split might give improved node purity: although the same response is predicted in both leaves ("Yes"), it might be that in one leaf the observations are all in the same class (so you are more sure) while in the other one there might be some impurity (so you are less sure).

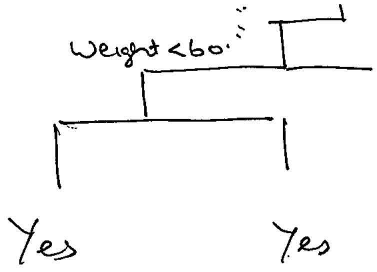

Figure 0.6: Splitting with qualitative predictor variables.

## Pruning

There is a trade-off between the model interpretability and performance on the training set:

- Small tree with fewer splits ⇒ More interpretable (lower variance), but poorer fit in training set (higher bias);
- Larger tree with many splits ⇒ Less interpretable (higher variance), better fit in training set (lower bias).

Yet, what really matters is the model performance on the test set, meaning that we are interested in the bias-variance trade-off (illustrated in Figure 0.7)—where bias is the error introduced by approximating a function $f$ with a simpler function $\hat{f}$, while variance refers to how the estimation would change if we used a different training set. This leads to the following trade-off:

- More complex model $\hat{f}$ ⇒ Low bias and high variance, since the model will capture many features of the data;
- Simpler model $\hat{f}$ ⇒ High bias and low variance.

To control the bias-variance trade-off (or, simply, the model complexity), CARTs use *prepruning* and *pruning*. The former stops the tree construction early on—for example by setting a minimum number of nodes per leaf. The latter grows a large tree and then prunes it back to get a subtree with the objective of getting low test error rates. For a regression tree, pruning can be easily implemented by introducing a cost complexity parameter (*cp*), which measures the trade-off between the fit to the training data and the model complexity. For any value of $\alpha$, we want to find the sub-tree $T \subseteq T_0$ of the original tree ($T_0$) such that:

$$\min_{T \subseteq T_0} \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|,$$

where $|T|$ is the number of terminal nodes in the tree. When $\alpha = 0$, $T = T_0$ (original tree). As $\alpha$ increases, there is a price to be paid for having too many leaf nodes and so we look for smaller sub-trees (similar to LASSO). To find the best value of $\alpha$, one can use cross-validation methods. In doing pruning of trees for classification problems, *rpart* uses the proportion of misclassified instances and number of terminal nodes to do cross-validation.

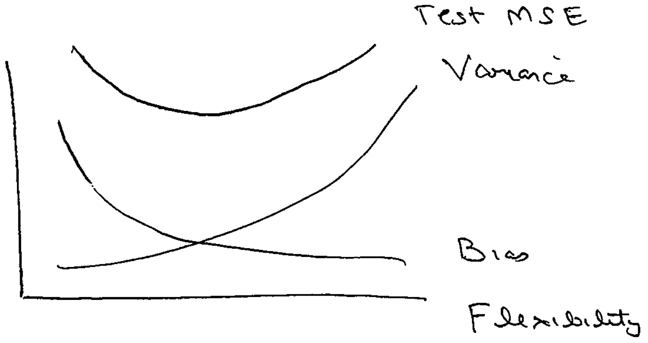

Figure 0.7: Bias-variance trade-off.

## Random Forests

Random Forests is a popular classification and regression technique introduced by Breiman (2001). It builds on a combination of tree predictors (ensemble method) that operates by constructing a multitude of trees during the training phase and asking each tree to output the mode of the classes (most popular class in classification) or mean predictions (in regression). The main advantage is that Random Forests correct for the flaw of decision trees to overfit the data and reduce the variance of the estimator. From a mathematical standpoint, we can interpret the function approximation of Random Forests as follows:

$$f(x) = \sum_{t=1}^{T^*} \frac{f_t(x)}{T^*},$$

where $T^*$ is the number of trees in the ensemble, and each $f_t(x)$ is a CART that trains on a subset of the data that is chosen randomly with replacement (this is known as *bootstrapping*). One of the challenges with this approach is that, if we use the same algorithm, then the predictions are highly correlated. This can be improved by learning trees on randomly chosen subsets of input variables. Note that Random Forests provide less interpretability but often better predictive power than CARTs. Naturally, the larger the number of trees the larger the time needed to build the ensemble (the same concept applies to the node sizes).

In a typical implementation, a random sample of $m = \sqrt{p}$ predictors are considered at each split of the tree and the split is made based only on one of this candidate variables. This helps de-correlate the trees chosen by the Random Forest method and reduces the variance.

(Rough) algorithm for Random Forests. For a selected number of trees $T^*$:

- Sample observations (with replacement) to create a subset of data;
- Build a tree. At each node:
  - For a chosen value of $m$, select only $m$ random predictor variables from the entire set;
  - Find the best among $m$ predictors;
  - At the next node choose another $m$ random predictors and repeat.

Notes:

- Bagging helps ensure that the average of many trees is not sensitive to noise in training set as just a single tree may be.
- Using subsets of predictors randomly helps reduce the strong correlation that might arise if a few predictors are very strong, which kept getting chosen.

Typically, the value of $m$ is equal to $\sqrt{p}$ in classification and $p/3$ in regression.

## References

Breiman, L. (1984). *Classification and regression trees.* Chapman & Hall/CRC.

Breiman, L. (2001). Random forests. *Machine learning 45*(1), 5–32.

Martin, A. D., K. M. Quinn, T. W. Ruger, and P. T. Kim (2004). Competing approaches to predicting supreme court decision making. *Perspectives on Politics 2*(4), 761–767.