Chapter 5
Tree-Based Forecasting Methods

Abstract The past two chapters have provided the necessary technical background for a consideration of statistical procedures that can be especially effective in criminal justice forecasting. The joint probability distribution model, data partitioning, and asymmetric costs should now be familiar. These features combine to make tree-based methods the fundamental building blocks for the machine learning procedures discussed. The main focus is random forests. Stochastic gradient boosting and Bayesian trees are discussed briefly as worth competitors to random forests.

5.1 Introduction

We will be emphasizing forecasting methods that are “tree-based.” This means that classification trees are a key component of the procedures. Classification trees have a long history (Breiman et al., 1984), and over the years have proven very effective in identifying complicated associations between a response variable and a set of predictors. However, used by themselves, classification trees can be very unstable, and are usually not a good stand-alone method. Moreover, the recursive nature of the partitioning and implicit reliance on step functions can badly misrepresent additive, smooth relationships such as those commonly assumed with convention linear regression. Still, if the inherent responsiveness of classification trees can be exploited with far greater stability and with a means to better summarize simple smooth functions, one may approach the best of all possible worlds: low bias, low variance, and sensible approximations of key relationships.
5.2 Splitting the Data

We have already discussed recursive partitioning, decision boundaries, and classification trees in a nontechnical manner. It is now time to add some formal details. To begin, how does the procedure decide which partitions to construct?

A good place to start is with the need to subset the data into two groups using the available predictors. The goal is to identify good break points. For a single quantitative predictor with \( m \) values, there are \( m - 1 \) splits that leave the order of the values unchanged. Therefore, \( m - 1 \) splits on that variable need to be evaluated. For example, if there are 50 distinct ages, there are 49 possible splits that maintain the existing order. The same logic holds for a single ordinal predictor. For a categorical predictor, order does not matter. Consequently, a categorical variable with \( K \) categories has \( (2^k - 1) \) possible splits. For example, if there are 5 countries of origin, there are 15 possible splits.

Recursive partitioning begins with all of the training data in the “root node.” All possible splits for all available predictors are examined, and the “best” single split over all available predictors is selected. The chosen split is better than the best split of any other predictor, and the data are partitioned accordingly. The same procedure is applied to all subsequent partitions until all observations have been placed in a terminal node. A predictor can be chosen more than once as the partitions are determined. Because the final partitions do not overlap, each case can only be in one terminal node.

What is meant by “best”? The goal is to have as little heterogeneity within a node as possible. The best split, therefore, is the one that reduces heterogeneity the most. Getting to a formal definition requires a few steps.

Using the reasoning in Hastie et al., (2009:Section 9.2.3), for any node \( m \), defining partition \( P_m \), having \( N_m \) observations, one can estimate

\[
\hat{p}_{mk} = \frac{1}{N_m} \sum_{i \in P_m} I(y_i = k),
\]

which is the proportion of observations in class \( k \) in node \( m \), and \( I \) denotes an indicator variable equal to “1” when \( y_i = k \) and “0” otherwise. One can then classify, as before, by the largest proportion.

But there are a very large number possible partitions implying different proportions for each class. What now? The proportion of observations in class \( k \) becomes the argument in a function to characterize the heterogeneity of a node. There are three popular options for this function.

\[
\text{Misclassification Error : } \frac{1}{N_m} \sum_{i \in P_m} I(y_i \neq k) \quad (5.2)
\]

\[
\text{Gini Index : } \sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} \quad (5.3)
\]