**57. Boosting eCommerce Sales Predictions with Bagging and Classification Trees**

In this project, I focus on using classification trees to analyze and predict eCommerce sales outcomes. So far, I've discussed regression trees, which are useful when the response variable is quantitative, such as predicting the amount of sales revenue. However, in many cases, I need to predict categorical outcomes, like whether a customer will purchase a product or not, or which category a product belongs to. In such cases, I use classification trees, which are very similar in structure to regression trees, but differ in how they measure performance and determine the best splits in the tree.

In a classification tree, I predict the most commonly occurring class in each terminal node. Unlike regression trees, where the prediction is based on the mean value, in classification trees, I assign each observation to the class with the majority presence in that terminal node. I grow the tree in the same way as I would for a regression tree, but instead of using the residual sum of squares as a criterion for making splits, I need a criterion that is better suited for classification tasks.

One simple approach is to use the classification error rate at each internal node. To compute this, suppose there are K possible classes, and I calculate the proportion of each class within the terminal node. The class I assign to the node will be the one with the highest proportion, and the classification error is simply 1 minus that maximum proportion. This measure gives me the proportion of errors I would make if I chose that class. However, it turns out that this approach can be quite jumpy and noisy, which does not lead to a smooth tree-growing process. For this reason, other measures are often preferable.

One such measure is the Gini index, a measure of variance across the classes. This index calculates the total variability in the region, and it is especially useful in eCommerce when I want to classify products or predict purchase behavior. For example, if I have K different product categories, the Gini index helps determine how "pure" a particular node is in terms of a single dominant category. If the Gini index is very low, it means one category is highly favored, with the rest being minimal. If the node is entirely pure—meaning all observations belong to one category—the Gini index would be zero. Conversely, if all categories are equally distributed, the Gini index would be at its maximum, indicating high impurity.

Another useful criterion is deviance, or cross-entropy, based on the binomial or multinomial log-likelihood. This measure behaves similarly to the Gini index and provides another way to assess the quality of splits in a classification tree. Both the Gini index and deviance are popular choices and yield similar results in most cases.

To illustrate this in an eCommerce context, I analyze a dataset containing customer behavior data, such as browsing history, product categories viewed, and purchase actions. The response variable is whether the customer made a purchase (Yes) or not (No). There are several predictors, such as customer age, gender, time spent on the site, number of products viewed, and previous purchase history.

I run the tree-growing process with cross-validation to find the best classification tree. Initially, I grow a full tree using all the data, which results in a "bushy" tree with many branches. For instance, an early split might occur based on the amount of time a customer spends on the website. Customers who spend more than a certain threshold might be split further based on the number of products they view, indicating different likelihoods of making a purchase.

At each terminal node of the tree, I see classifications of Yes or No. For example, a terminal node might show that most customers in that segment did not make a purchase (No). The tree assigns this classification to any new observation that ends up in this node. Interestingly, there may be cases where two nodes predict the same outcome (e.g., both No), but one is "purer" than the other, meaning it has a higher concentration of one class. The Gini index would help identify these purer nodes.

This full tree is likely too complex for practical use, so I apply cross-validation to prune it back to an optimal size. As seen in the cross-validation results, I compare training error, test error, and validation error to determine the best tree size. I find that a tree size of around six terminal nodes provides a good balance between simplicity and prediction accuracy. The pruned tree represents a smaller subset of the full tree and gives the best classification performance for predicting customer purchase behavior.

I also compare the performance of decision trees with linear models. Decision trees are not always the best choice for every problem. To illustrate this, I consider two scenarios in eCommerce sales prediction. In the first scenario, the best decision boundary between customers who make a purchase and those who don't is linear. Here, a decision tree attempts to partition the space but struggles because it creates "boxy" regions that do not align well with the true linear boundary. As a result, the tree does not perform well, making a steppy approximation of the linear decision boundary.

In contrast, in a scenario where the decision boundary is more naturally segmented into boxy regions—such as when predicting different product categories based on browsing behavior—a linear model would fail to capture the complexity, while a decision tree would excel. The tree can perfectly partition the regions with just a few splits, making it a better choice for such problems.

This demonstrates that some eCommerce problems are more naturally suited to trees, while others are better handled with linear models or other techniques. I think of trees as one tool in my toolbox, to be used when appropriate, alongside simpler linear models and more advanced methods.

To wrap up the section on classification trees, I see both advantages and disadvantages. On the plus side, trees are simple and easy to interpret, especially if they are not too large. This simplicity makes them appealing for non-specialists, such as eCommerce managers, who may prefer a decision-making process that mirrors their own logical steps. For example, the pruned tree for predicting customer purchases might start with a split based on time spent on the site, followed by the number of products viewed, and then by previous purchase history. This step-by-step stratification resembles how some managers might intuitively think about customer behavior.

Additionally, trees can handle categorical predictors without the need to create dummy variables. Even if a categorical variable has multiple levels, the tree can split it into subsets of categories, making it very flexible for eCommerce applications. However, the major downside is that trees generally do not predict as well as more state-of-the-art methods, especially when used in isolation.

To address this, I move beyond single trees and look at ensemble methods that combine multiple trees to improve predictive performance. These methods, such as bagging, boosting, and random forests, build many trees on the same data and combine them to reduce variance and improve accuracy. These advanced methods provide substantial improvements and are an exciting area for further exploration in predicting and optimizing eCommerce sales