Machine Learning and Causal Inference.

Trees

Eduardo Fé

05/12/2023

Introduction

Tree-based models partition the data following an “IF-THEN” structure

```
IF Covariate 1 > 0.5, Outcome = 1.3; ELSE...
```

The tree is thus formed by *splits* of the data and *leaves* (or terminal nodes)

Each leave contains a formula to predict the outcome associated with observations satisfying the conditions leading to that leave.

Introduction

A famous dataset attributed to Ronald Fisher (the statistician) and Edgar Anderson (a botanist). It includes 50 samples of three different types of iris flower from the Gaspé peninsula, in North East Canada.

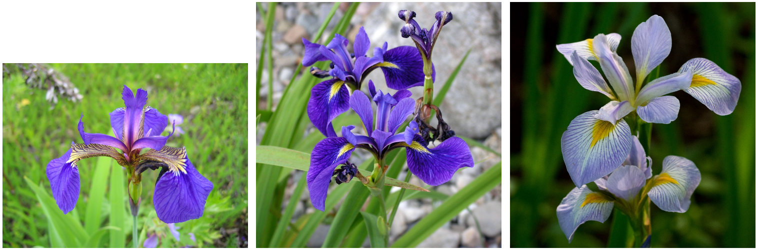


Figure 1: Three different types of Iris flower: Setosa, Versicolor and Virginica.

Introduction

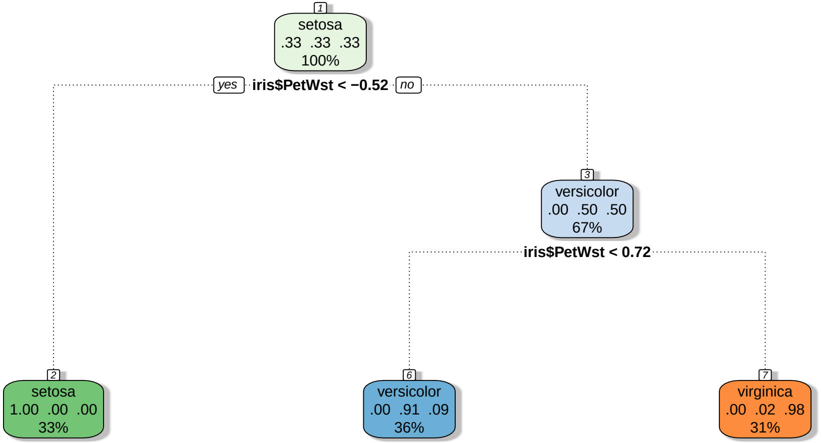


Figure 2: A tree classifying the samples in the iris dataset, by sepal length (SepLsd) and petal width PetWst.

Introduction

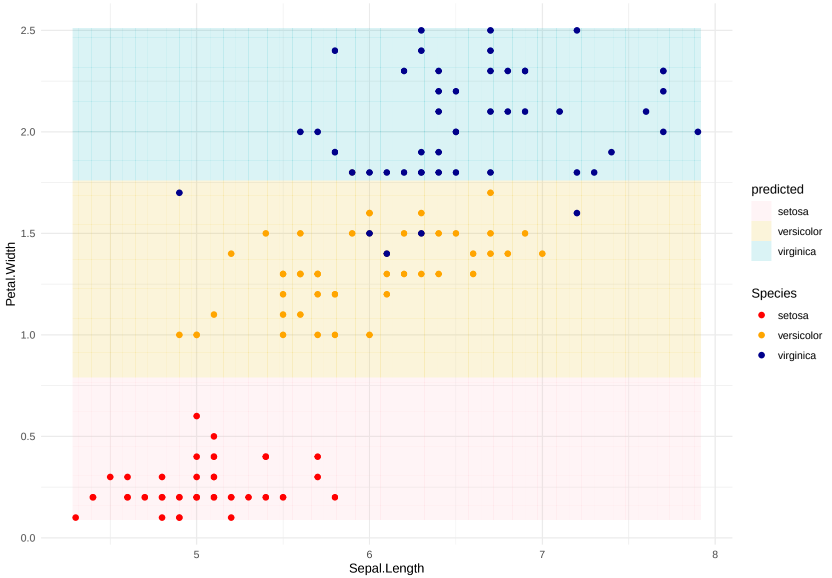
Trees are popular because:

- Results are often easy to interpret
- One does not specify the functional relationship between the outcome of interest and the explanatory variables
- They naturally do covariate selection (any variable not used in splits is considered to be irrelevant)

However they suffer from:

- Model instability (small changes to data can result in very differing tree structures)
- Moderate predictive ability (they split the covariate space in rectangles)

Introduction



Building trees

There are different ways of constructing trees. One of the oldest and most popular algorithms is the *Classification and Regression Tree* (CART), introduced in the book by Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984).

Given an outcome Y and a set of predictors $X_1, \ldots, X_p$, the algorithm is as follows:

- For each predictor, find the value that splits the data into two groups, $\mathcal{S}_1, \mathcal{S}_2$ such that the sum of squared errors are minimised

$$SSE = \sum_{i \in \mathcal{S}_1} (Y_i - \bar{Y}_1)^2 + \sum_{i \in \mathcal{S}_2} (Y_i - \bar{Y}_2)^2 \tag{1}$$

- Select the predictor and split that results in the largest reduction in the SSE.

Building trees

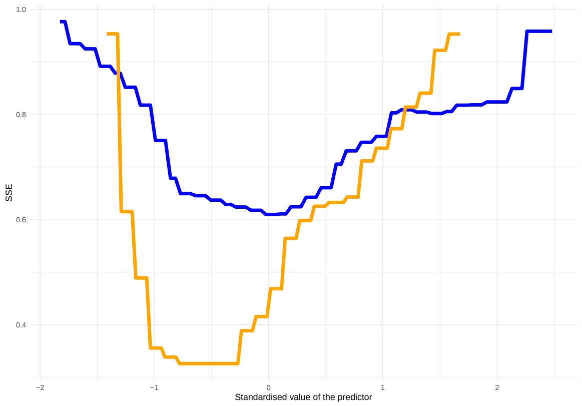


Figure 3: Sum of squared errors given the value of the split, for each predictor in the iris dataset (sepal length -blue- and petal width -in orange

Pruning trees

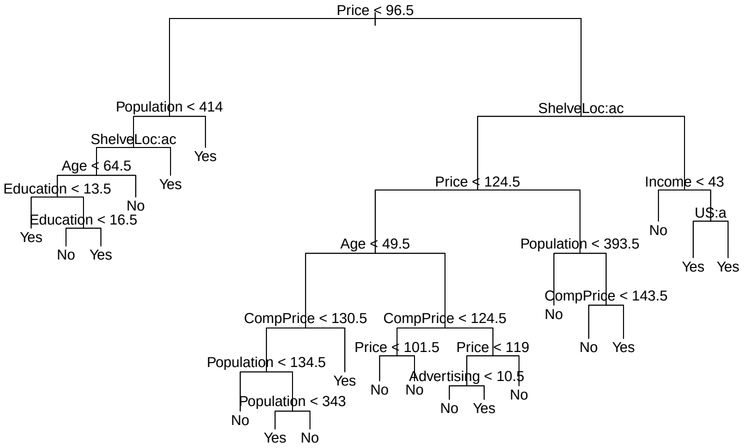


Figure 4: The CART algorithm will fit a training dataset very well, by creating a large tree. However, such detailed trees are likely to perform poorly on a new (test) dataset. The above tree was created using the 'Carseats' dataset in James, Witten, Hastie and Tibshirani's book.

Pruning trees

The previous example is based on the `Carseats` dataset[1]. We split the data into a training and test datasets, each with 200 observations.

The output variable was a dummy variable for "High sales" (sales exceeding a level).

The tree fitted to the training data classified sales level correctly 83% of the time.

When the trained tree was used to predict sales in the test data, accuracy was 71%

Not a disaster, but can we do better?

[1]This data set is include in James, Witten, Hastie and Tibshirani's ISLR package (supporting their book, *An Introduction to Statistical Learning*). The data collects information about child car seat sales in 400 locations.

Pruning trees: Cost Complexity

To find the 'right' size of a tree, we can include a penalty for tree complexity.

Let $\mathcal{T}$ represent the tree and $\#\mathcal{T}$ the number of nodes. Then tree selection can be based on minimising:

$$SSE = \sum_{i \in \mathcal{S}_1} (Y_i - \bar{Y}_1)^2 + \sum_{i \in \mathcal{S}_2} (Y_i - \bar{Y}_2)^2 + c_p \cdot \#\mathcal{T} \qquad (2)$$

where c_p is the complexity parameter.

As c_p increases, smaller trees are preferred.

Pruning trees

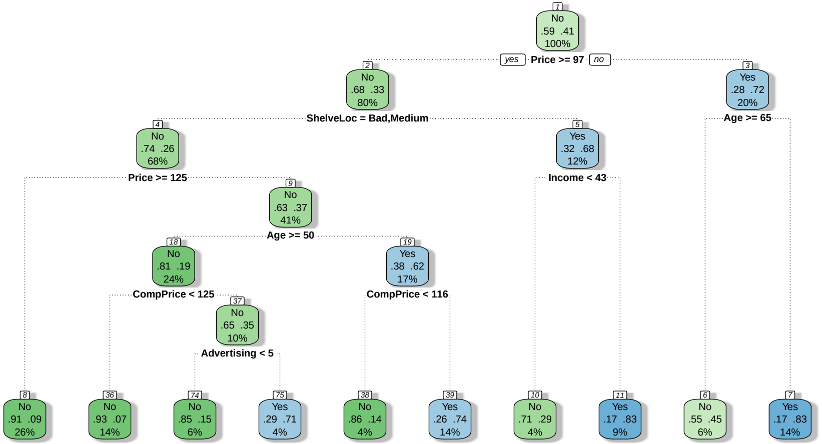


Figure 5: Pruning the 'Carseats' tree. Complexity parameter was set at 0.01

Pruning trees

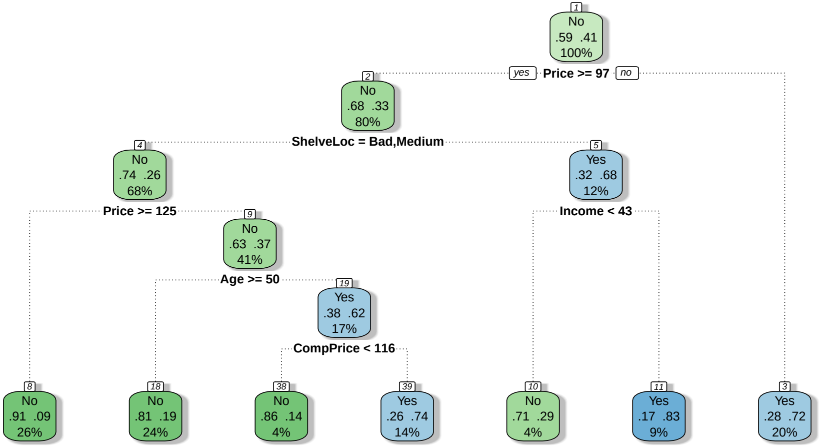


Figure 6: Pruning the ‘Carseats’ tree. Complexity parameter was set at 0.02

Pruning trees

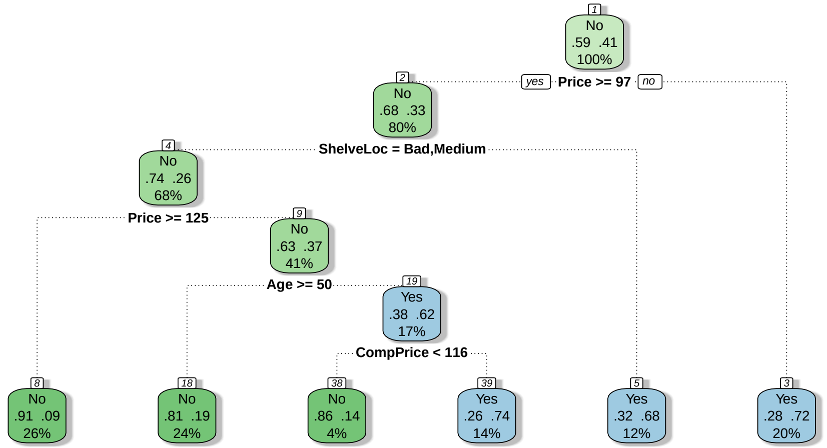


Figure 7: Pruning the 'Carseats' tree. Complexity parameter was set at 0.04

Pruning trees

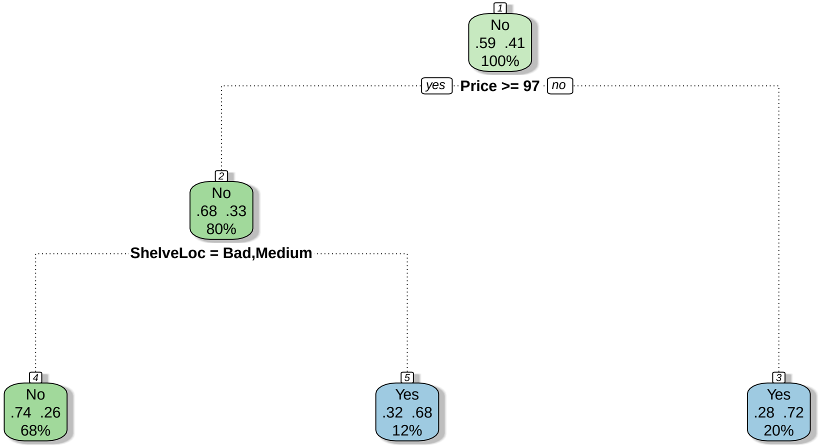


Figure 8: Pruning the ‘Carseats’ tree. Complexity parameter was set at 0.08

Selecting the complexity parameter

We could find a “best” c_p manually. Given a collection of values of c_p, say $\{c_{p,r}\}_{r=1}^{R}$,

- Estimate a tree for each $c_{p,r}$ using a training data set
- Estimate the prediction error in a test data set
- Select the tree yielding a best performance

The problem with this approach is that, if we drew another dataset from the same population (with identical variables, sample size, etc), our results would change.

So we need to take that into account. We do so by using K-Fold Cross-Validation.

Selecting the complexity parameter

Randomly split the data into K groups (folds) of the same size. Let $\mathcal{F}_k$ be the set of observations in fold $k = 1, ..., K$. For $k = 1, ..., K$,

1. Grow a tree using all but fold k.
2. Prune the tree for a sequence of complexity parameters, $\{c_{p,r}\}_{r=1}^{R}$
3. Undertake prediction using fold k and obtain the MSE/Classification error for each tree.
4. Average MSE/Classification error across folds for a given c_p.

Select the c_p resulting in the smallest average MSE/Classification error (and estimate the tree using that c_p).

Trees have high variance.

One of the main pitfalls affecting trees is their inestability.

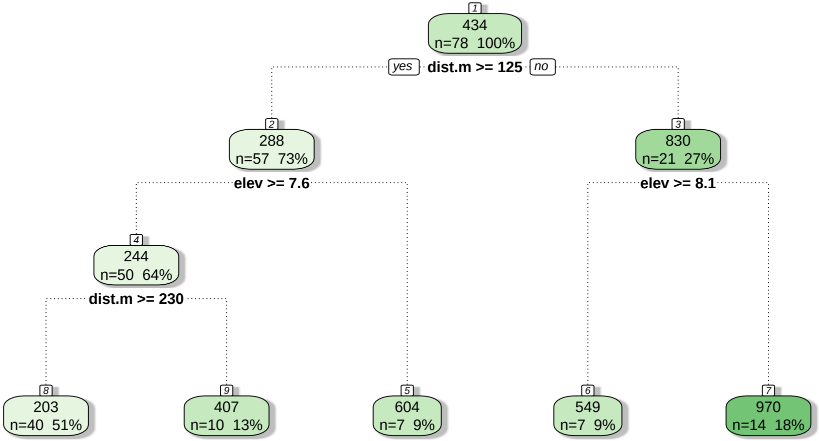
Trees have high variance, which means that small variations to the data set can result in substantive modification of the final tree.

Consider the `meuse` dataset, which contains four heavy metals measured in the top soil in a flood plain. The idea is that the rive (Meuse) carries heavy metals and deposits them close to the bank and in flood-prone areas.

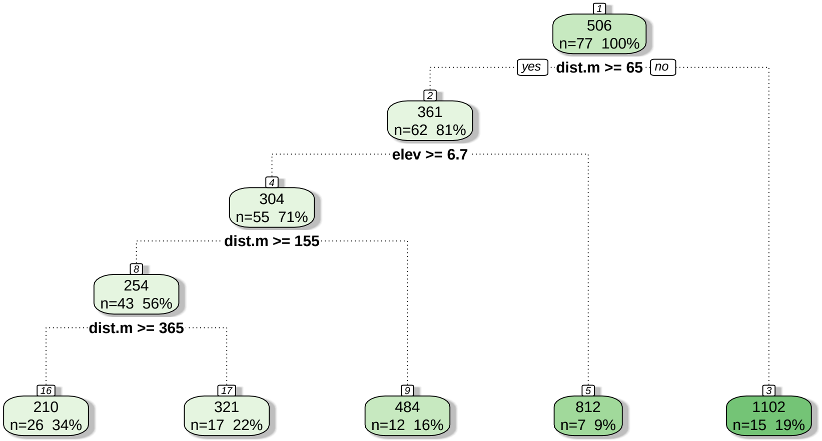
We construct a tree to explain zinc concentration, as a function of the frequency of flooding, distance to the rive and elevation of the area.

We randomly split the 155 observations into 2 groups of about the same size (78 vs 77)

Trees have high variance.



Trees have high variance



Bagging (Out of Bag)

Problem: prediction is based on a sample of one (per unit/individual).

One way of reducing the variance (in general) is to get more data.

E.g. cats have seven lives; to explain cats' consumption of milk, sample them in each of their $t = 1, ..., T = 7$ parallel lives.

- Estimate a tree for each sample and obtain the predictions, for tree $t = 1, 2, ..., T$ $\left\{\hat{Y}_{t,i}\right\}_{i=1}^{n}$ and observation (cat) $i = 1, ..., n$.
- In a "regression" problem, compute the average prediction for each observation[2]

$$\hat{Y}_i = T^{-1} \sum_{t=1}^{T} \hat{Y}_{t,i} \tag{3}$$

We would have a large number of predictions per unit; by averaging them, we reduce the variance.

[2]In a classification use a *majority vote* (the most frequent class predicted for Y_i across all t samples).

Bagging (Out of Bag)

The solution is **The bootstrap**. Given the full sample $\mathcal{S} = \{Y_i, X_{1,i}, ..., X_{p,i}\}_{i=1}^{n}$, the most basic version of the bootstrap

Repeat $b = 1, ..., B$ times:

- Draw, at random and with replacement n elements of $\mathcal{S}$. This yields the b^{th} bootstrap sample $\mathcal{S}_b^*$.
- Estimate a FULL (not-pruned/cross-validated) tree with the bootstrap sample $\mathcal{S}_b^*$ (**small bias** -but large variance)
- Use the estimated tree to predict Y_i for the elements NOT INCLUDED[3] in $\mathcal{S}_b^*$, and obtain the predictions for those observations.

After all B iterations, you will have around $B/3$ predictions per Y_i; call those $\hat{Y}_{b,i}$. Then the final tree is based on the the average of those predictions (or the majority rule in the classification problem).

[3] Since we allow for "replacement" when obtaining a bootstrap sample, one can show that, on average, 1/3 of the sample points in $\mathcal{S}$ are left out. Thus, the bootstrap is doing the job that cross-validation would do otherwise. The bagging method above is known as Out of Bag bagging estimator.

Random Forest

Bagging **substantially** improves the performance of tree-regression/classification.

However it is not totally optimal.

Recall that if you have two random variables X, Y,

$$\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y) + 2 \cdot \text{Cov}(X, Y) \tag{4}$$

If $\text{Cov}(X, Y) > 0$, the variance of the sum is larger than the sum of the variances.

Bagging produces highly correlated trees if, for example, one variable explains a lot of the variation in the outcome.

In that case, all B trees in a Bagging procedure will use that explanatory variable high up the tree, thus creating a positive correlation between trees $b = 1, ..., B$.

Random Forest

Random Forest is a variation of Bagging.

You draw a series of bootstrap samples from the data

However, when constructing the tree in each boostrap sample b, at each split only a random sample m of the p explanatory variables are considered.

This reduces the level of correlation across the predictions produced by each tree and reduces the variance of the final prediction.

The only problem we now face is having to tune a new parameter, namely the number of covariates to consider at each split.

The rule of thumb is $m = \sqrt{p}$.