Activity: Decision Trees

Jay Singhvi

04/24/2024

Install these libraries if necessary (in the console, not in the markup file.) and then execute the statements below to load them into your environment.

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.3
```

Decision Trees (25 TODOs)

Decision trees are non-linear partitioning-type models. They work by partitioning, or splitting, the data into subsets, which are again split into smaller subsets, and so forth until a stopping criterion is met.

The last partitions are called the "leaves", "terminal nodes", or "leaf nodes" of the tree. The tree building algorithm chooses a variable to use as a splitting criterion for each "node" in the tree. The choice of variable is made by assessing some measure of "purity" achieved by the split. Purity means how "mixed" the partitions are in terms of the predicted variable. When built, a tree defines a set of rules, where each sequence of rules define a path from the root node to a terminal node, which provides a prediction.

Decision trees can work with both numeric and categorical data as input and output. Trees that predict numeric values are called "regression trees", and those that predict categorical values are called "classification trees".

In Part 1 you will work with regression trees and in Part 2 you will work with classification trees.

Data Processing The data consists of various statistics observed on major league baseball players in the United States.

"AtBat" "Hits" "HmRun" "Runs" "RBI" "Walks" "Years" "CAtBat" "CHits"
"CHmRun" "CRuns" "CRBI" "CWalks" "League" "Division" "PutOuts" "Assists" "Errors"
"NewLeague" "Salary"

The goal is to predict a player's Salary based on the player's statistics.

1) TODO: Read in the file "baseball.csv" and assign the return to a variable bb.df. Note the size of this dataset. Then, remove all missing values (encoded as NA). Do not print a summary of the entire dataset in this file (you can do that in the console if you want).

```
bb.df = read.csv("baseball.csv")
bb.df = na.omit(bb.df)
```

2) TODO: How many rows were removed?

59

3) TODO: How many predictors are numeric? categorical?

numeric: 17 , categorical: 2

Make any categorical variables factors.

```
bb.df$League = factor(bb.df$League)
bb.df$Division = factor(bb.df$Division)
bb.df$NewLeague = factor(bb.df$NewLeague)
```

4) TODO: Print the summary statistics for the predicted variable: Salary.

```
summary(bb.df$Salary)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    67.5   190.0   425.0   535.9   750.0  2460.0
```

It's always a good idea to explore the data, especially the way the predicted variable is distributed. This is helpful when you are checking the modeling results in your analysis. Getting to know the data is always an important data science activity.

Training and Testing Sets The following code creates subsets for training and testing, with 75% of the data used for training, and 25% held out for testing.

```
RNGversion("4.1.2")
set.seed(123456)

index.train <- createDataPartition(y = bb.df$Salary, p = 0.75, list = FALSE)
train.data <- bb.df[index.train,]
test.data <- bb.df[-index.train,]
```

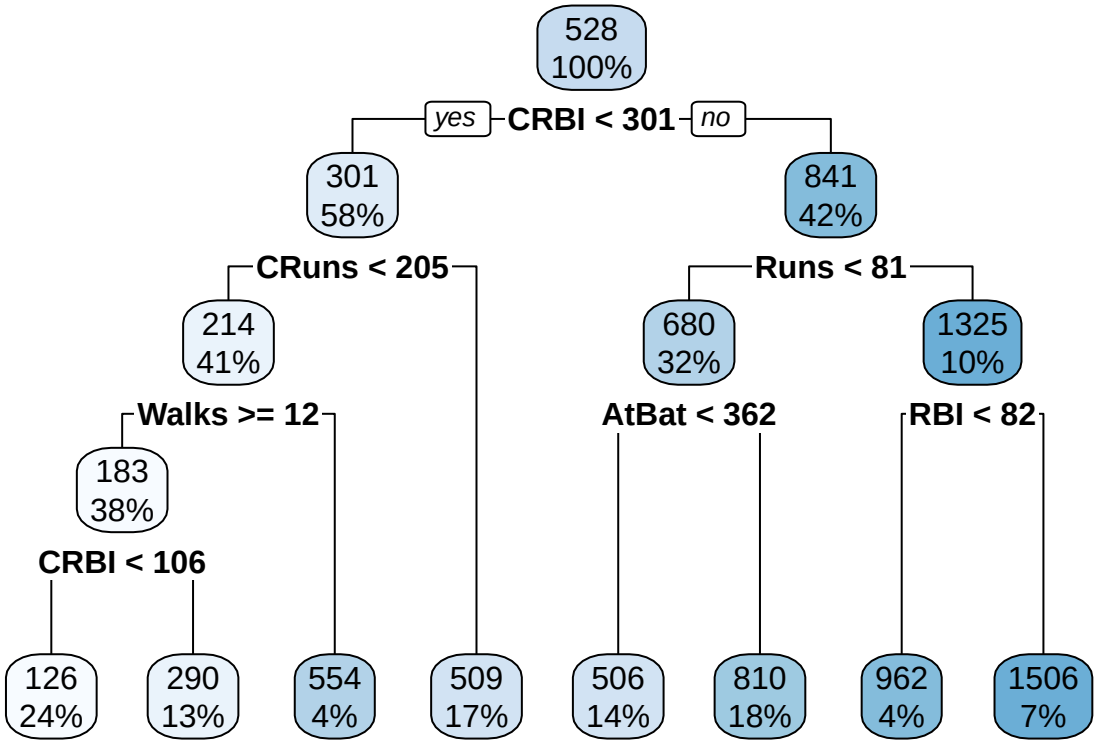
Part 1: Fitting and Evaluating a Regression Tree. In the code chunk below, the rpart function is called to fit a regression tree model to the entire data set. We are using a regression tree because we want to predict Salary, which is a numeric variable.

The parameters used for rpart are: a formula for the model, a "method" parameter with the value "anova", and a "data" parameter, which specifies the data set- bb.df in this case. Note the method="anova" parameter in the rpart function call. This indicates a regression tree, and analysis of variance will be used to make partitions (see ?rpart for more).

```
reg.tree1 <- rpart(Salary ~ .,  method="anova", data=train.data)
```

One advantage of tree models is that they are relatively easy to understand visually (when they are small, that is). The code below uses the rpart.plot function to visualize the tree model that was fitted to the training data. (If necessary, you can resize the Plots window to better see the details in this plot. I suggest routing chunk output to the console, not inline so the plot is easier to see).

```
rpart.plot(reg.tree1)
```



Interpreting a Regression Tree. The plot has some good information about how the model was built. One of the major uses of tree models is for "feature" selection: finding which predictors are most important in contributing to predicting the outcome.

This is useful when a dataset has many possible predictors. Linear regression is another model that is useful in this regard.

5) TODO: List the predictors that the tree model considers important in predicting Salary:

CRBI, CRuns, Runs, Walks, AtBat, RBI

A regression model predicts the average value of a numeric outcome. In the plot, each node reports on the model's prediction for Salary at that level, and then the percent of the data that the node is considering. (Notice that the sum of percentages add to 100% across a level in the tree).

The percent data in nodes decreases as the tree grows down because it is partitioning (splitting) the data at each node- that's why it's called a decision tree: it "decides" on what predictor to use and how to make the split to go to the next level.

6) TODO: What does the root node predict for the average Salary of all players in the training set? This is the prediction using no predictors.

528

7) TODO: How many nodes are in this tree, including the root? How many nodes are leaf (terminal) nodes? What is the percentage of data of the terminal node that contains the most data points? What is the average Salary prediction for this node?

total nodes: 15 , terminal nodes: 8 , highest precentage: 24 %, prediction:126

Tree Rules 8) TODO: What variable is used to perform the initial split of the data at the root of this tree to form the two nodes at the next level?

CRBI

9) TODO: What is the splitting criteria, or rule, used for creating the left node? The right node?

left node: <301 , right node: >301

10) TODO: Starting at the root, list the rules that lead to the terminal node that predicts an average Salary of 810:

CRBI<301 (no), Runs<81 (yes), AtBat<362 (no)

11) TODO: Starting at the root, list the rules that lead to the terminal node that predicts an average Salary of 290:

CRBI<301 (yes), CRuns<205 (yes), Walks>=12 (yes), CRBI<106 (no)

Notice that some variables are used more than once because they are the best at splitting the data at that particular level.

12) TODO: List the variable(s) that are repeated in this tree, if any?

CRBI

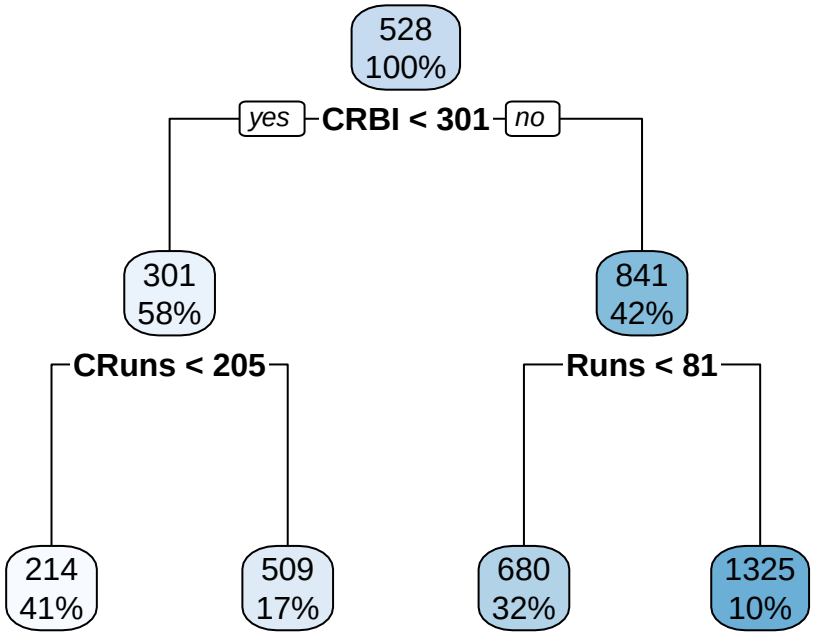
Testing Regression Tree Performance. Now we will test the predictive performance of the tree on the testing set using the predict function. These two error terms are calculated: Root mean squared error, RMSE Mean absolute error, MAE

```
tree1.pred<- predict(reg.tree1, test.data, type="vector")
#calculate errors
rmse1<- sqrt(mean((tree1.pred-test.data$Salary) ^2))
mae1<- mean(abs(tree1.pred-test.data$Salary))
```

Having a result for one regression tree is not terribly meaningful unless we can compare it to another model. Given that decision trees are know for their tendency to overfit, we will create a pruned version of the first tree to compare.

The CP, “complexity parameter”, value is used to control the depth of a tree as it is being built. The code below uses the “control” parameter to set a CP value that will result in a smaller tree:

```
reg.tree2 <- rpart(Salary ~ .,  method="anova", data=train.data, control = rpart.control(cp = 0.05))
rpart.plot(reg.tree2)
```



13) TODO: The height of a tree is defined as the longest path from the root to a leaf node. For example, the height of reg.tree2 is 2. What is the height of reg.tree1?

4

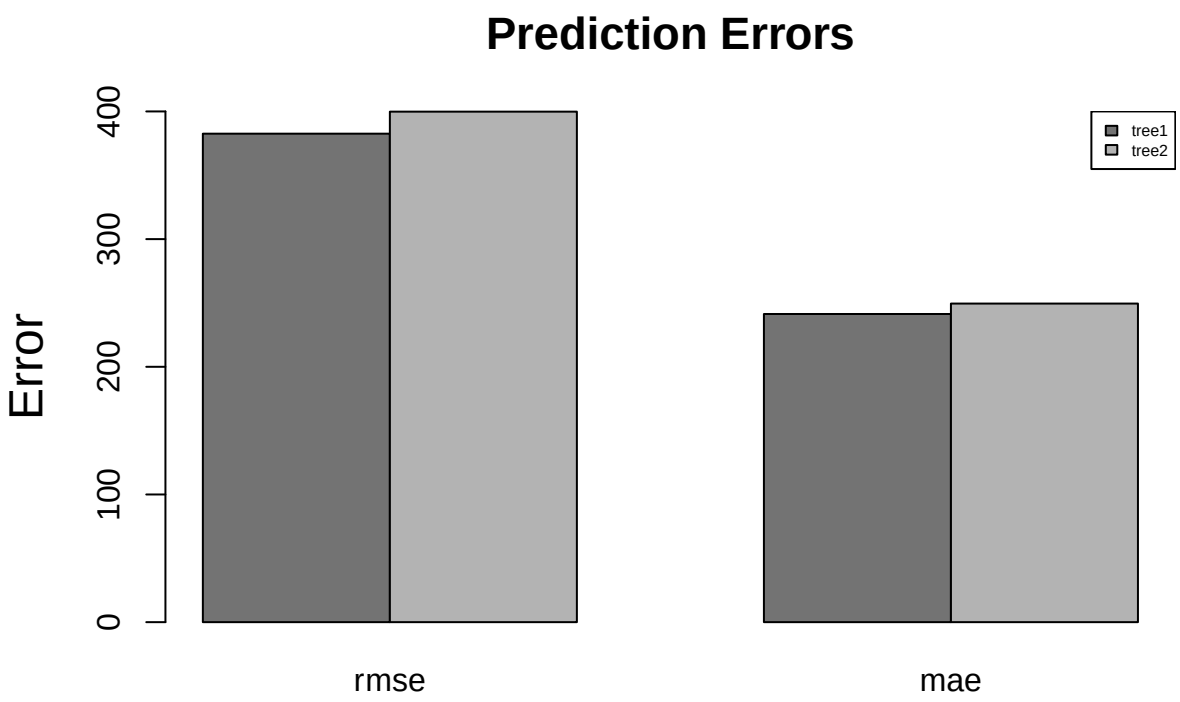
Now predictions for tree2 are obtained and the error terms calculated:

```
tree2.pred<- predict(reg.tree2, test.data, type="vector")
rmse2<- sqrt(mean((tree2.pred-test.data$Salary) ^2))
mae2<- mean(abs(tree2.pred-test.data$Salary))
```

This code plots a bar chart of the two error calculations for each tree.

```
res<-c(rmse1, rmse2, mae1, mae2)
dim(res) <- c(2,2)
colnames(res) <- c("rmse","mae")
colors <- c("grey45", "grey70")
barplot(res, main="Prediction Errors", ylab = "Error", ylim=c(0, 400), cex.lab = 1.5, cex.main = 1.4, b

legend("topright", legend=c("tree1", "tree2"),
       fill=colors, cex=0.5)
```



Summary. Summarize the prediction results for tree1 and the pruned tree2.

14) TODO: Which tree performed better on the test data? Briefly explain why you think one tree model did better in the results.

tree 1 did better than tree 2 because the bar plot shows thta rmse and mae for tree 1 is less compared to tree 2