Lab 7: Regression Trees

Sarah Wright

Introduction

In this lab, we will use bike rental data from a bike rental company in Washington D.C. The data set includes information about the total number of bikes rented each day over the course of two years. It also includes information about the season, the day of the week, the weather, and which days were holidays. Our primary interest will be in predicting the number of bike rentals on a particular day (`cnt`) using the other predictors. More details about the data set are included in the file `bike_data_readme.txt`.

Regression Trees

As usual, the first thing that we would like to do is to split our data into a `test` and `train` set. For this data set, we want to make certain that we have an equal representation of days and seasons. So, we will randomly sample to obtain our test and train sets.

1) Download the data contained in the file `bike_data.csv`. Using the `sample` function, randomly select 60% of the rows to create the `train` set and put the rest of the data in the `test` set. **Don't forget to set the seed if you want to be able to reproduce your results.**

```
library(class)
```

```
## Warning: package 'class' was built under R version 4.0.5
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.2
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.2
```

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
## Warning: package 'tibble' was built under R version 4.0.2
```

```
## Warning: package 'tidyr' was built under R version 4.0.2
```

```
## Warning: package 'readr' was built under R version 4.0.2
```

```
## Warning: package 'forcats' was built under R version 4.0.2
```

```
library(ggplot2)
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.0.5
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.0.2
```

```
set.seed(4)

bikeData <- read.csv("/Users/sewii/Documents/CLASSES_Spring2022/Data325_AppliedDataScience/bike_data.csv
bikeData<- bikeData[,-1]
rows =  731

intTrain = 0.60 * rows
train <- sample_n(bikeData,intTrain, replace = FALSE)

intTest = 0.40 * rows
test <- sample_n(bikeData,intTest, replace = FALSE)
```

Before we start building fancy models, it's helpful to understand our data and to have a simple baseline model for comparison.

2) What do you think will be the most important variables in the data set for determining how many bikes will be rented on a given day? Why?

```
#weekday since we are looking at any given day
#cnt since this will tell us how many people are bike searching
```

3) Consider the simple baseline model that always predicts the mean number of bikes rented in the training set as the number of bikes that will be rented on any day. Determine the *root mean square error* of this model when applied to the test set. Save this value as `rmse0` and print it here.

```
#install.packages("Metrics")
library(Metrics)
```

```
## Warning: package 'Metrics' was built under R version 4.0.2
```

```
averageDay <- mean(train$cnt)
rmse0<- rmse(test$cnt, averageDay)
rmse0
```

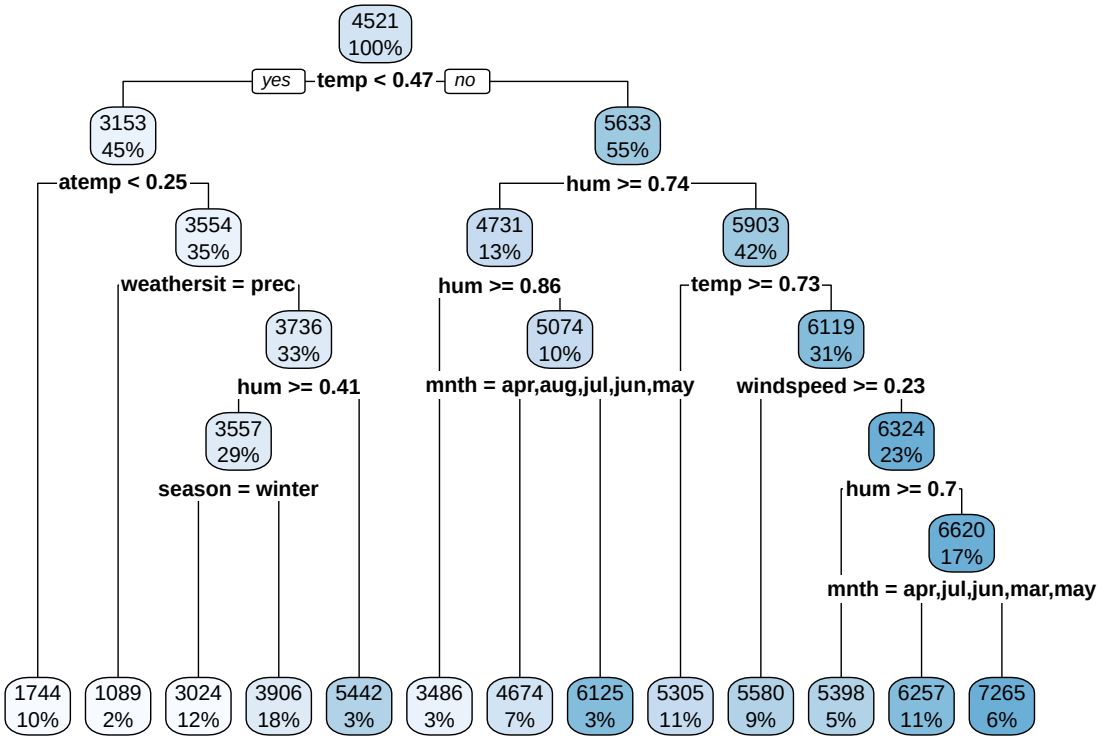
```
## [1] 1912.883
```

Hopefully, we can do better than this simple model using a regression tree. We build a regression tree in much the same way that we build a classification tree. We first construct a series of "splitting rules" in order to divide the observations in the training set. Since our goal is to predict a numerical response variable, we then label each leaf by the mean of the response variable for all of the training observations at that leaf.

To make a prediction for a new observation, we apply the splitting rules to determine which leaf it belongs to. The predicted value of the response variable for the new observation is again just the label of that leaf. The R packages for building regression trees will iteratively determine the splitting rules in order to minimize the error on the training set.

4) Create a regression tree called `reg_tree` that predicts the number of bikes that will be rented on a given day. Use the `rpart` function from the `rpart` package so that you can make a nice plot of your tree. This function will automatically recognize if you are trying to predict a categorical or numerical variable and will build either a classification or regression tree respectively. Plot your tree below. **HINT: Use all variables (except cnt) from the training set to predict the number of bikes that will be rented on a given day. Use the rpart.plot() function with rpart() to plot your tree.**

```
reg_tree <- rpart(cnt~., data = test)
rpart.plot(reg_tree)
```



5) Based on your tree, which variable do you think is most important for determining the number of bikes rented each day? Does this agree with your answer from earlier?

```
#tempeture seems to be the most important variable. I was wrong with my hypothesis at the begining, wee
```

6) In a simple linear model $Y = \beta_1 X + \beta_0$, two different values of X will always yield different values for the response variable Y. Thus, a linear model can make infinitely many predictions depending on the input. How many different predictions are possible from your regression tree model? Why?

```
#there are 13 possible prediction values with the regression tree . This is becasue we have a total of
```

7) Use your classification tree to create a vector of predictions for the number of bikes that are rented each day in the `test` set. How many unique entries are there in this vector? Does this agree with your answer to the previous question?

```
regPredictions <- predict(reg_tree, newdata = test)
length(unique(regPredictions))
```

```
## [1] 13
```

```
#there are 13 unique values in the prediction vector and this agrees with the hypothesis from above tha
```

8) Compute the root mean squared error for your regression tree model applied to `test`. Save this value as `rmse1` and print it here.

```
rmse1<- rmse(test$cnt, regPredictions)
rmse1
```

```
## [1] 1072.386
```

Your model probably did much better than the baseline model. However, one topic that has come up repeatedly in this course is the idea of overfitting.

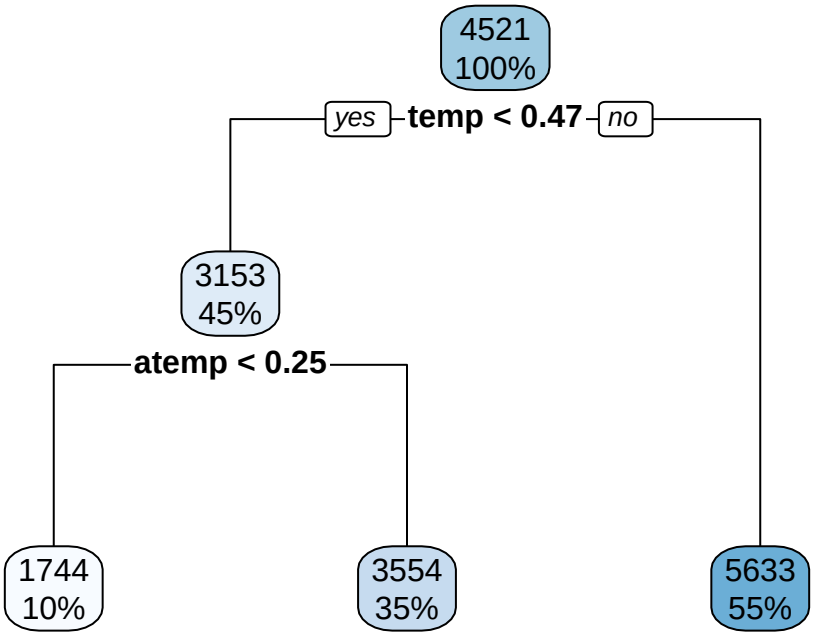
Consider what would happen if when building our regression tree, we just kept splitting each node until we had a gigantic regression tree where each terminal node had exactly one observation from the training data. Then our error on the training set would be zero! Of course, this tree would probably perform very poorly on new data sets.

Our goal is to build a tree that captures a lot of the features of our data, but we don't want a tree so large that we overfit the training data. It can also be nice to have a small tree with just a few splits because they are easy to interpret and show the few features that are the most significant.

We could just keep splitting our data until our accuracy no longer increases by some threshold. However, then we might build a small tree that misses an important split. In practice, it is usually better to build a large tree and then "prune" off nodes. If you want to know more about pruning, Section 8.3.2 in the text provides nice examples with the `tree` package.

9) Experiment with different values of the complexity parameter (`cp`) in the code below to create a pruned version of your tree with just a few terminal nodes (values between 0 and .25 give pretty sensible results).

```
prune_tree <- prune(reg_tree, cp = .05)
rpart.plot(prune_tree)
```



10) In practice, we would want to use *cross-validation* to choose the best cost parameter to make a pruned tree that avoids over-fitting. But, just for fun, let's compute the rmse for our model from Problem #9 on the `test` set and call this `rmse2`. Print this value here. How does it compare to `rmse1` from the large tree model?

```
prune_tree2 <- predict(prune_tree, newdata = test)

rmse2<- rmse(test$cnt, prune_tree2)

rmse0
```

```
## [1] 1912.883
```

```
rmse1
```

```
## [1] 1072.386
```

```
rmse2
```

```
## [1] 1371.75
```

```
#the rmse is much greater than the rmse1 that was predicted , this means that we have a higher residual
```