Tree-Based Methods

Lucas Ben

In this project I'll develope several tree-based regression methods to predict the prices of cars based on their characteristics. The data set is from the UCI Machine Learning Repository. A formatted version of the data was used for this project.

```
set.seed(2002) # for reproducibility
library(caret)
```

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

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

```
data = read.csv("cars_data.csv") # 160x15
```

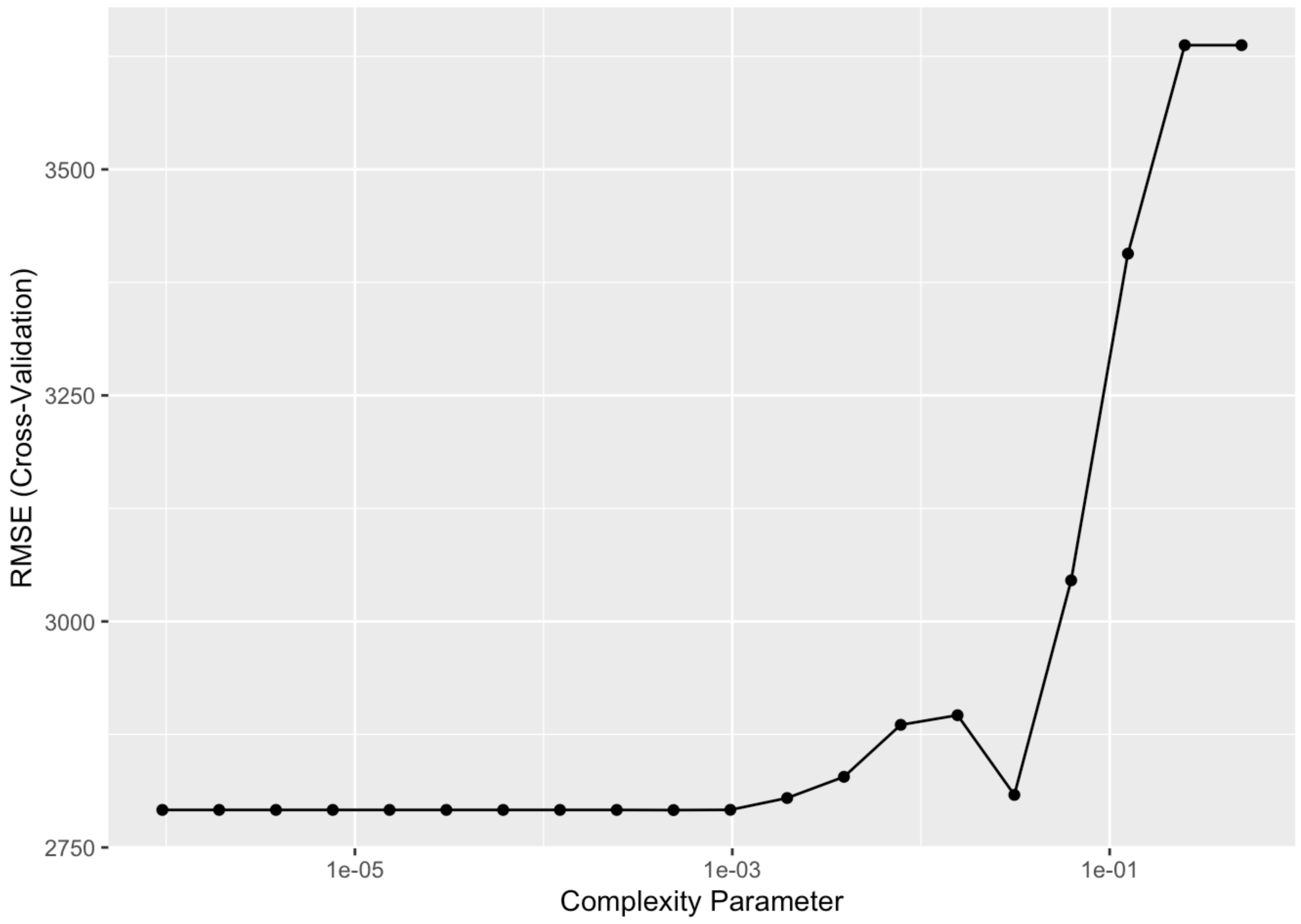
Regression Tree

Here I'm training a regression tree on the data. Also, I'll be testing different values for the tuning parameter *cp*.

cp penalises complexity so if when choosing between two models of varying complexity that have the same predictive performance, it is beneficial to choose the simpler model.

```
ctrl = trainControl(method = "cv", number = 5) # defining model assessment method (5-fol
d CV)
formula = price ~ .
tune_tree = data.frame(cp = 2^((-20):(-1))) # tuning parameters
mod1 = train(
  formula,
  data = data,
  method = "rpart",
  trControl = ctrl,
  tuneGrid = tune_tree
) # training regression tree
```

Plotting the cross-validation RMSE against the complexity parameter, showing the x-axis on the log scale.



Bagging

Training a bagging regression model.

```
mod2 = train(
  formula,
  data = data,
  method = "treebag",
  trControl = ctrl
) # bagging
```

Boosting

Training a boosting regression model. I'll be trying all of the following combinations for the tuning parameters.

mstop ∈ {50, 100, 200} (number of boosting iterations) *maxdepth* ∈ {1,2} (max depth of boosting trees) *nu* ∈$\{2^{-10}$, 2^{-9}, …, 2^{-3}, $2^{-2}\}$

```
# tuning parameters
mstop = c(50, 100, 200)
maxdepth = c(1,2)
nu = 2^((-10):(-2))

tune_bst = expand.grid(mstop = mstop, maxdepth = maxdepth, nu = nu) # grid of all combin
ations
mod3 = train(
  formula,
  data = data,
  method = "bstTree",
  trControl = ctrl,
  tuneGrid = tune_bst
) # boosting
```

Here I'll be calculating the *cross-validation RMSE* from two different model configurations:

(1) mstop = 200, maxdepth = 1, nu = 2^{-5} **(2)** mstop = 200, maxdepth = 2, nu = 2^{-4}

The cross-validation **(1)** 2230 and **(2)** 1955. Choosing the method with the lowest RMSE minimizes prediction error. Therefore **(2)** is better at predicting car prices.

```
res = mod3$results # extracting results for all trained models
# identifying relevant rows of res
ind1 = which(res$maxdepth == 1 & res&nu == 2^(-5) & res$mstop == 200)
ind2 = which(res$maxdepth == 2 & res$nu == 2^(-4) & res$mstop == 200)
res[c(ind1, ind2), 1:4]
```

```
##      maxdepth      nu mstop     RMSE
## 18          1 0.03125   200 2162.303
## NA         NA      NA    NA       NA
## NA.1       NA      NA    NA       NA
## NA.2       NA      NA    NA       NA
## NA.3       NA      NA    NA       NA
## NA.4       NA      NA    NA       NA
## NA.5       NA      NA    NA       NA
## NA.6       NA      NA    NA       NA
## NA.7       NA      NA    NA       NA
## 48          2 0.06250   200 2001.322
```

Random Forests

As I train a random forest model using *caret* with the method **rf**, I'll try every possible value of the variable *mtry* for this data set (from one to the total number of input variables).

```
tune_rf = data.frame(mtry = 1:(ncol(data) - 1)) # tuning parameters
mod4 = train(
  formula,
  data = data,
  method = "rf",
  trControl = ctrl,
  tuneGrid = tune_rf
) # training random forest model
```

Model Comparison

To compare the models I'll rank the four methods by *RMSE*. Based on cross-validation RMSE, the best model is boosting, followed by random forest, bagging, and single tree.

```
mods = list(tree = mod1, bag = mod2, boost = mod3, rf = mod4) # making a list of the mod
els
rmse = sapply(mods, function(mod) min(mod$results$RMSE)) # extracting the lowest RMSE fo
r each model
sort(rmse) # ranks the RMSE from best to worst
```

```
##    boost       rf      bag     tree
## 2001.322 2009.311 2303.146 2791.345
```

Prediction

Here I'll plot the price predicted by the four methods over a range of values of the *horsepower* input variable. I'll do this by creating a data frame with one column for each input variable, where each column is either filled with the mean value of the input variable (variables that are not *horsepower*) or with a regular sequence of length 1000 that covers the range of observed values of *horsepower*.

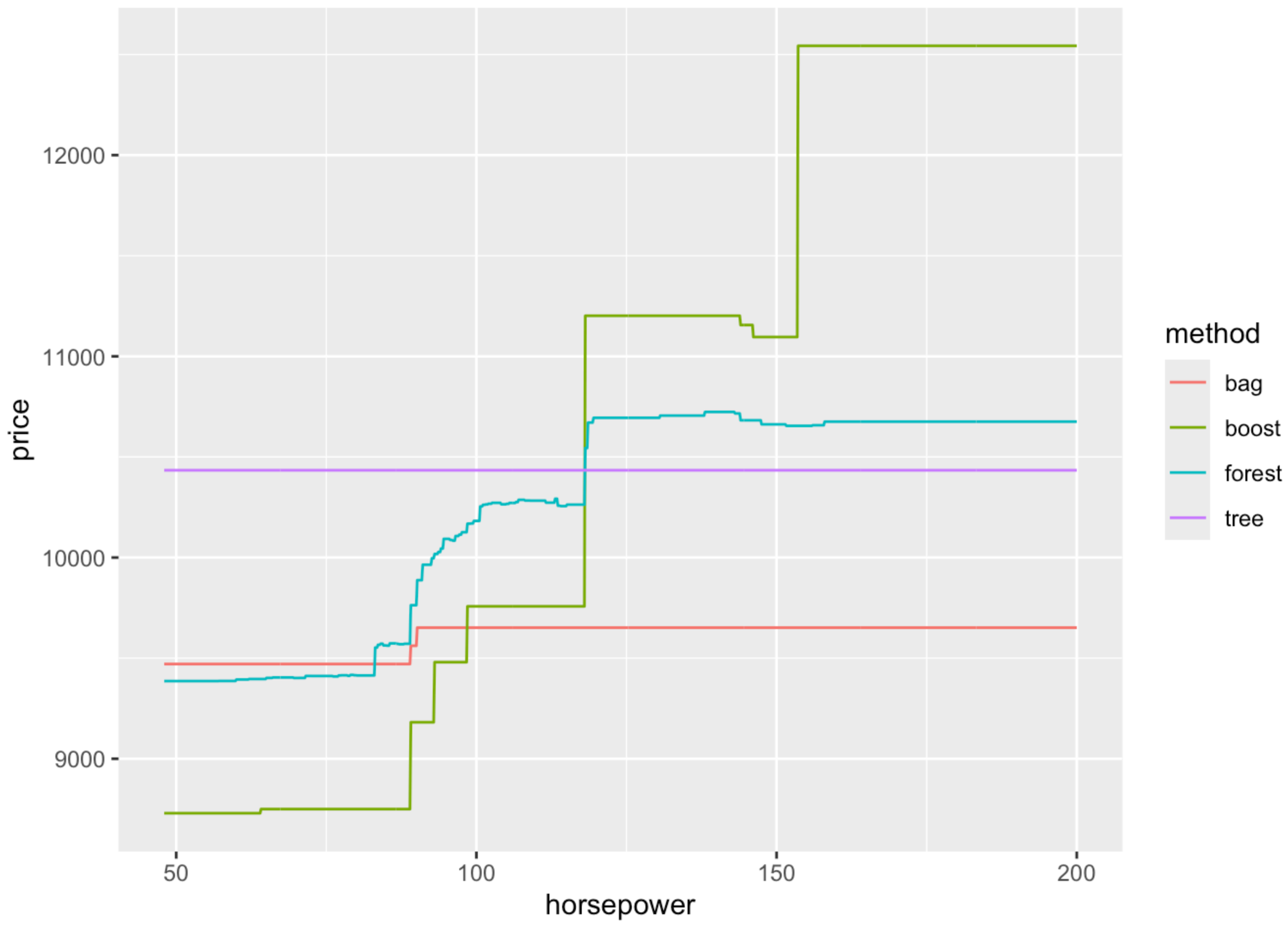
```
ngrid = 1000
var = "horsepower"
# initialising a matrix of input values where each column is filled with the mean value
for the corresponding input
newdata = matrix(NA, nrow = ngrid, ncol = ncol(data))
for(i in 1:ncol(newdata)) {
    newdata[,i] = mean(data[,i])
}
colnames(newdata) = colnames(data)
newdata[,var] = seq(min(data[,var]), max(data[,var]), length = ngrid) # filling the colu
mn for horsepower with a grid of values
newdata = as.data.frame(newdata)
head(newdata)
```

```
##   normalizedLosses wheelBase   length    width   height curbWeight engineSize
## 1            121.3  98.23562 172.3194 65.59625 53.87875    2459.45   119.0938
## 2            121.3  98.23562 172.3194 65.59625 53.87875    2459.45   119.0938
## 3            121.3  98.23562 172.3194 65.59625 53.87875    2459.45   119.0938
## 4            121.3  98.23562 172.3194 65.59625 53.87875    2459.45   119.0938
## 5            121.3  98.23562 172.3194 65.59625 53.87875    2459.45   119.0938
## 6            121.3  98.23562 172.3194 65.59625 53.87875    2459.45   119.0938
##       bore   stroke compressionRatio horsepower peakRpm  cityMpg highwayMpg
## 1 3.298437 3.237312         10.14513   48.00000 5116.25 26.50625   32.06875
## 2 3.298437 3.237312         10.14513   48.15215 5116.25 26.50625   32.06875
## 3 3.298437 3.237312         10.14513   48.30430 5116.25 26.50625   32.06875
## 4 3.298437 3.237312         10.14513   48.45646 5116.25 26.50625   32.06875
## 5 3.298437 3.237312         10.14513   48.60861 5116.25 26.50625   32.06875
## 6 3.298437 3.237312         10.14513   48.76076 5116.25 26.50625   32.06875
##      price
## 1 11427.68
## 2 11427.68
## 3 11427.68
## 4 11427.68
## 5 11427.68
## 6 11427.68
```

Now I’ll predict the prices for *newdata* for the four regression methods and plot predicted price against horsepower.

```
pred1 = predict(mod1, newdata = newdata)
pred2 = predict(mod2, newdata = newdata)
pred3 = predict(mod3, newdata = newdata)
pred4 = predict(mod4, newdata = newdata)

df = data.frame(var = newdata[,var],
                price = c(pred1, pred2, pred3, pred4),
                method = rep(c("tree", "bag", "boost", "forest"), each = ngrid))
ggplot(df, aes(var, price, col = method)) +
  geom_line() +
  labs(x = var)
```


price
horsepower
12000
11000
10000
9000
50
100
150
200
method
bag
boost
forest
tree