Regression Trees

Niclas Frederic Sturm

22-10-2019

Introduction

This document contains the code for the seminar "Statistical Learning and Econometrics", with a focus on tree-based models. Four models will be used for modeling: A simple decision tree, a bagged model, boosting as well as a random forest. To accomplish a good benchmarking estimated, we will use two datasets; one for regression, the other for classification. We will first import the UCI Communities and Crime Dataset as the regression dataset.

```
library(rpart) # Loading required packages
library(rpart.plot)
library(rsample)
library(dplyr)
library(readr)
library(naniar)
library(Metrics)
library(ipred)
library(purrr)
library(ranger)
library(gbm)
library(ggplot2)
library(doParallel)
library(foreach)
library(tidyr)
library(stringr)

set.seed(1997) # Set seed for reproducibility
setwd("/Users/nfsturm/Documents/STATLEARN/forestranger")

crime_data <- read_csv("crimedata.csv", col_names = TRUE)
drop_cols <- c("communityname", "state", "countyCode", "communityCode", "fold", "murdPerPop", "rapesPerI
crime_data <- crime_data %>%
  select(-drop_cols) %>%
  replace_with_na_all(condition = ~.x == "?")

drop_vars <- miss_var_summary(crime_data)
drop_vars <- drop_vars %>%
  filter(pct_miss > 10) %>%
  select(variable) %>%
  pull()

crime_data <- crime_data %>%
  select(-drop_vars)

crime_data <- drop_na(crime_data)
crime_data <- map_at(crime_data, .at = 1:103, .f = as.numeric)
crime_data <- as_tibble(crime_data)
```

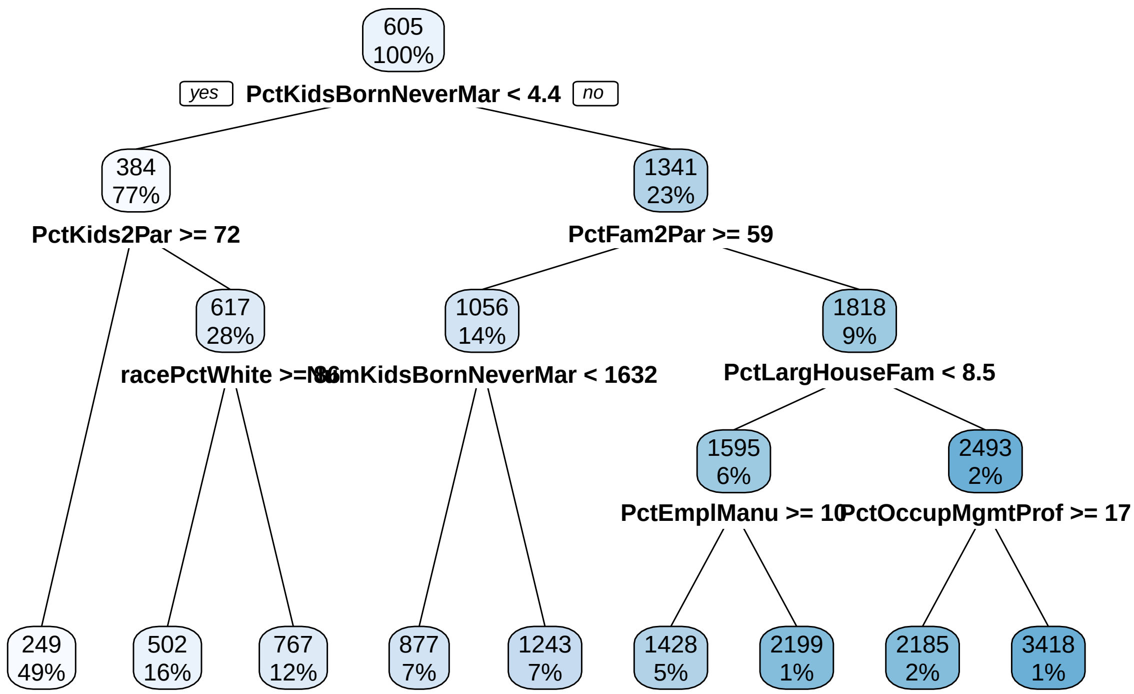
We will randomly split the data into a training and testing dataset.

```
crime_splits <- initial_split(crime_data)
train <- training(crime_splits)
test <- testing(crime_splits)
```

Now, we will fit a regression tree using recursive partitioning. The response variable is "ViolentCrimesPerPop", i.e. the number of violent crimes per 100,000 inhabitants. All predictors are used.

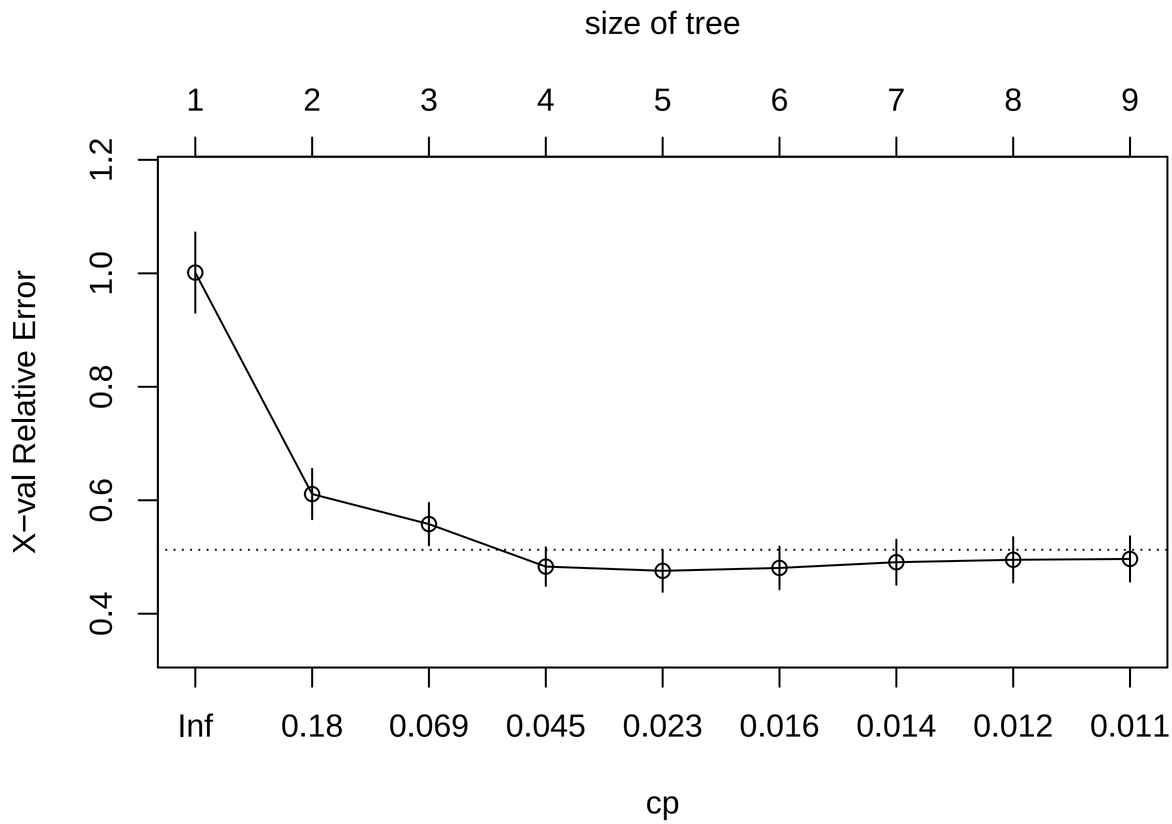
Decision Trees

```
tree_crime <- rpart(ViolentCrimesPerPop~., method = "anova", data = train, control = rpart.control(xval
rpart.plot(tree_crime, branch = 0.04, tweak = 1.2)
```



The regression tree results in a tree that predicts nine distinct values. Although the tree is not very complex, we could investigate how the complexity parameter λ influences the relative error of the model.

```
plotcp(tree_crime)
```



Inspecting the graphic, it becomes apparent that the x-val relative error (crossvalidated $1 - R^2$) is minimized at a value of around 0.01.

Bootstrap aggregating

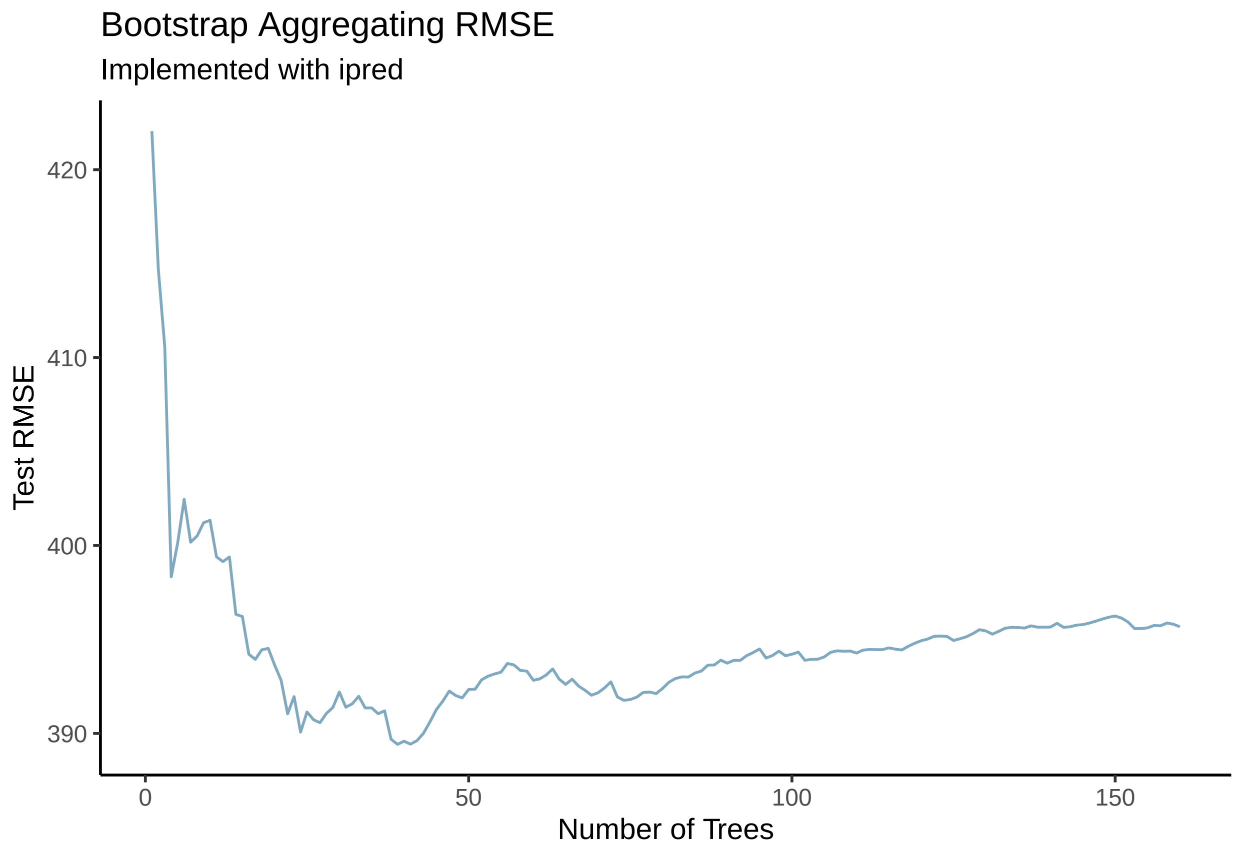
Following this simple regression tree, we will estimate a model using bootstrap aggregation. To visualize how the number of trees fit influences prediction accuracy, we will run a process that creates 160 trees and fits and one model per tree. This model will then predict values of the test set.

```
nr_cores <- detectCores() - 2
cl <- makeCluster(nr_cores) # Use the number of cores available minus 2
registerDoParallel(cl) # Activate parallel backend

# Fit trees in parallel and compute predictions on the test set
predictions <- foreach(
  icount(160),
  .packages = "rpart",
  .combine = cbind
   ) %dopar% {
    # Create bootstrapped copy of the training set
    index <- sample(nrow(train), replace = TRUE)
    train_boot <- train[index, ]

    # Grow tree on top of bootstrapped copy
    bagged_tree <- rpart(
      ViolentCrimesPerPop ~ .,
      control = rpart.control(minsplit = 2, cp = 0.05),
```

```
      data = train_boot
      )

    predict(bagged_tree, newdata = test)
   }
# stopCluster(cl)

predictions <- as_tibble(predictions)

predictions_df <- predictions %>%
  mutate(instance = 1:n(), actual = test$ViolentCrimesPerPop)

predictions_df2 <- gather(predictions_df, nr_tree, predicted, -c(instance, actual)) %>%
  mutate(nr_tree = str_extract(nr_tree, '\\d+'))

predictions_df2$nr_tree <- as.numeric(predictions_df2$nr_tree)

tree_prep <- predictions_df2 %>%
  arrange(instance, nr_tree) %>%
  group_by(instance) %>%
  mutate(avg_prediction = cummean(predicted)) %>%
  group_by(nr_tree) %>%
  summarize(RMSE = rmse(actual, avg_prediction))
```

We will plot the bagged model.

```
ggplot(tree_prep, aes(nr_tree, RMSE)) + geom_line(col = "#7fa9c1") + theme_classic() + labs(title = "Boo
```



Boosting

Next, a gradient boosting machine will be used. We will the popular "gbm" package to build a model.

```
boost_model <- gbm(ViolentCrimesPerPop~., distribution = "gaussian", data = train, n.trees = 2000, shrin
best <- which.min(boost_model$cv.error)
sqrt(boost_model$cv.error[best])
```

```
## [1] 388.1402
```

The best tree grown by boosting achieves a cross-validated RMSE of 383.

Random Forests

We conclude the analysis of regression trees with a random forest. To this end we will be using the "ranger" package, a fast implementation. Again, ten-fold cross-validated will be used to evaluate the model.

```
cv_split <- vfold_cv(train, v = 10)

cv_data <- cv_split %>%
  mutate(train = map(splits, ~training(.x)),
         validation = map(splits, ~testing(.x)))

cv_tune <- cv_data %>%
  crossing(mtry = 1:15) %>%
  mutate(model = map2(train, mtry, .f = ~ranger(formula = ViolentCrimesPerPop~.,
  data = .x, mtry = .y)))

cv_tune <- cv_tune %>%
  mutate(validation_actual = map(validation, ~.x$ViolentCrimesPerPop)) %>%
  mutate(validation_predicted = map2(model, validation, ~predict(.x, .y)$predictions)) %>%
  mutate(validation_rmse = map2_dbl(validation_actual, validation_predicted,
                                    ~rmse(actual = .x, predicted =.y)))

cv_select <- cv_tune %>%
  select(mtry, model, validation_rmse) %>%
  group_by(mtry) %>%
  summarize(mean.rmse = mean(validation_rmse))

cv_select
```

```
## # A tibble: 15 x 2
##     mtry mean.rmse
##    <int>     <dbl>
##  1     1      400.
##  2     2      386.
##  3     3      380.
##  4     4      378.
##  5     5      375.
##  6     6      376.
##  7     7      375.
##  8     8      376.
##  9     9      377.
## 10    10      375.
## 11    11      376.
```

```
## 12    12       376.
## 13    13       375.
## 14    14       378.
## 15    15       377.
```

The results show that the minimum RMSE is achieved for a number of five variables that are considered at each split (mtry = 5). This is the model, for which we will calculate the RMSE on the test data.

```r
ranger_model <- cv_tune$model[[5]]
actual <- test$ViolentCrimesPerPop
predicted <- predict(ranger_model, test)
predicted <- predicted$predictions
rmse(actual, predicted)
```

```
## [1] 338.1287
```

We will now plot mtry against the cross-validated RMSE metric.

```r
ggplot(cv_select, aes(x = mtry, y = mean.rmse)) + geom_line(col = "#7fa9c1") + geom_point(col = "#7fa9c
```