Tuning

Brooke Anderson

February 25, 2016

```
knitr::opts_knit$set(root.dir = "..") # Reset root directory for analysis
library(lubridate) # To help handle dates
library(dplyr) # Data wrangling
library(ggplot2) # Plotting
library(gridExtra) # To arrange multiple ggplot objects in one graph
library(caret) # machine learning
library(rpart.plot) # prettier tree plot
library(splines) # fitting splines
library(randomForest) # for variable importance plot
```

Read in and clean up the data:

```
train <- read.csv("data/train.csv", as.is = TRUE) # `as.is` so `datetime` comes in as
                                                   # character, not factor
test <- read.csv("data/test.csv", as.is = TRUE)

train <- mutate(train,
                datetime = ymd_hms(datetime),
                year = factor(year(datetime)),
                hour = factor(hour(datetime)),
                month = month(datetime),
                yday = yday(datetime),
                weather = factor(weather, levels = c(1, 2, 3, 4),
                                 labels = c("Clear", "Mist", "Light Precip",
                                            "Heavy Precip")),
                season = factor(season, levels = c(1, 2, 3, 4),
                                labels = c("Spring", "Summer", "Fall", "Winter")),
                workingday = factor(workingday, levels = c(0, 1),
                                    labels = c("Holiday / weekend",
                                               "Working day")))
test  <- mutate(test,
                datetime = ymd_hms(datetime),
                year = factor(year(datetime)),
                hour = factor(hour(datetime)),
                month = month(datetime),
                yday = yday(datetime),
                weather = factor(weather, levels = c(1, 2, 3, 4),
                                 labels = c("Clear", "Mist", "Light Precip",
                                            "Heavy Precip")),
                season = factor(season, levels = c(1, 2, 3, 4),
                                labels = c("Spring", "Summer", "Fall", "Winter")),
                workingday = factor(workingday, levels = c(0, 1),
                                    labels = c("Holiday / weekend",
                                               "Working day")))
```

Tuning

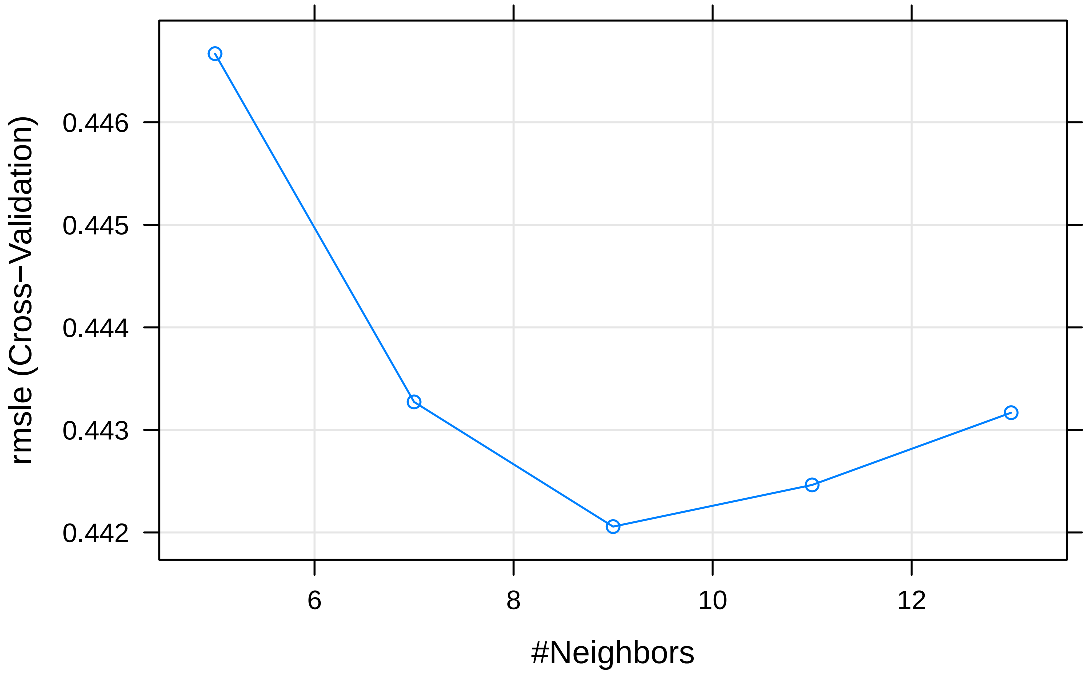
```
rmsle_fun <- function(data, lev = NULL, model = NULL, ...){
  log_p_1 <- log(data$pred + 1)
  log_a_1 <- log(data$obs + 1)
  sle <- (log_p_1 - log_a_1)^2
  rmsle <- sqrt(mean(sle))
  names(rmsle) <- "rmsle"
  return(rmsle)
}

fitControl <- trainControl(method = "cv",
                           number = 5,
                           ## Evaluate performance using
                           ## the following function
                           summaryFunction = rmsle_fun)
```

Tuning a k-NN model

```
set.seed(825)
mod_1 <- train(count ~ temp + hour + workingday + year, data = train,
               method = "knn",
               trControl = fitControl,
               metric = "rmsle",
               maximize = FALSE,
               preProcess = c("center", "scale", "spatialSign"),
               tuneLength = 5)
mod_1
```

```
## k-Nearest Neighbors
##
## 10886 samples
##    15 predictor
##
## Pre-processing: centered, scaled, spatial sign transformation
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 8709, 8708, 8710, 8708, 8709
##
## Resampling results across tuning parameters:
##
##   k   rmsle      rmsle SD
##    5  0.4466694  0.005523422
##    7  0.4432731  0.007505331
##    9  0.4420570  0.008668263
##   11  0.4424635  0.009397534
##   13  0.4431680  0.008070305
##
## rmsle was used to select the optimal model using  the smallest value.
## The final value used for the model was k = 9.
```

```r
plot(mod_1)
```



Check how the model did in the training data and on Kaggle:

```r
rmsle <- function(train_preds, actual_preds){
  log_p_1 <- log(train_preds + 1)
  log_a_1 <- log(actual_preds + 1)
  sle <- (log_p_1 - log_a_1)^2
  rmsle <- sqrt(mean(sle))
  return(rmsle)
}

write_test_preds <- function(test_preds, mod_name){
  out_file <- data.frame(datetime = as.character(test$datetime),
                         count = test_preds)
  out_name <- paste0("test_predictions/", mod_name, ".csv")
  write.csv(out_file, file = out_name, row.names = FALSE)
}
```

```r
train_preds <- predict(mod_1, newdata = train)
rmsle(train_preds, train$count)
```

```
## [1] 0.4080661
```

```r
test_preds <- predict(mod_1, newdata = test)
write_test_preds(test_preds, "knn_tuned")
```

On Kaggle, this got 0.48618.

Fitting one of the GLMs using RMSLE as loss function

```
mod_4 <- train(count ~ year*hour*workingday*season +
               weather + ns(temp, knots = c(30, 35)),
             data = train, family = quasipoisson,
               method = "glm",
               trControl = fitControl,
               metric = "rmsle",
               maximize = FALSE)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
mod_4
```

```
## Generalized Linear Model
##
## 10886 samples
##    15 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 8710, 8709, 8708, 8707, 8710
##
## Resampling results
##
##   rmsle     rmsle SD
##   0.352327  0.006455651
##
##
```

Check how the model did in the training data and on Kaggle:

```
train_preds <- predict(mod_4, newdata = train)
rmsle(train_preds, train$count)
```

```
## [1] 0.3399749
```

```
test_preds <- predict(mod_4, newdata = test)
write_test_preds(test_preds, "glm_rmsle")
```

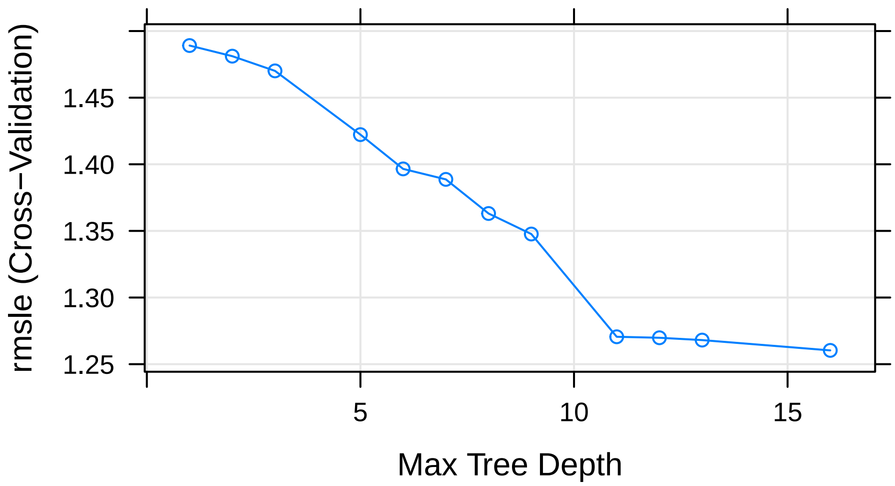
On Kaggle, this got … .

Tuning a regression tree model

```r
set.seed(825)
mod_2 <- train(count ~ season + holiday + workingday + weather +
                 temp + atemp + humidity + windspeed + year + hour +
                 month + yday, data = train,
               method = "rpart2",
               trControl = fitControl,
               metric = "rmsle",
               maximize = FALSE,
               tuneLength = 12)
mod_2
```

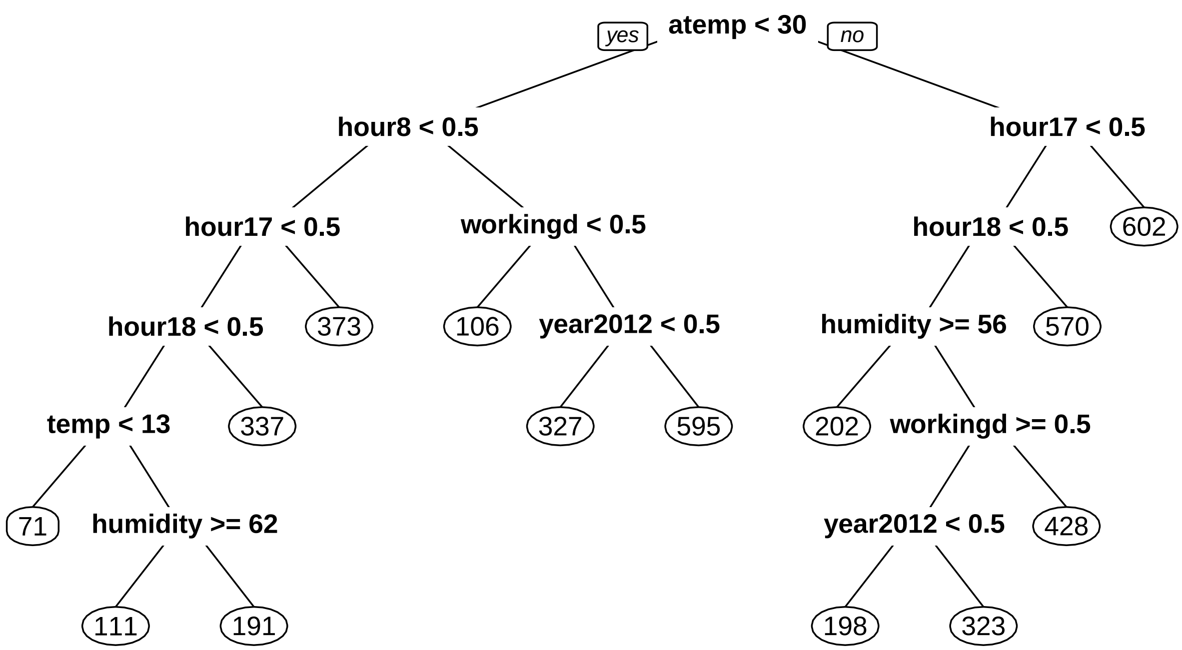
```
## CART
##
## 10886 samples
##    15 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 8709, 8708, 8710, 8708, 8709
##
## Resampling results across tuning parameters:
##
##   maxdepth  rmsle     rmsle SD
##    1        1.489116  0.01295560
##    2        1.481155  0.01218276
##    3        1.470109  0.01731650
##    5        1.422261  0.01068877
##    6        1.396540  0.01145594
##    7        1.388663  0.01240650
##    8        1.363071  0.03375930
##    9        1.347669  0.04134337
##   11        1.270557  0.06958523
##   12        1.269808  0.06913675
##   13        1.268055  0.06800105
##   16        1.260327  0.06899347
##
## rmsle was used to select the optimal model using  the smallest value.
## The final value used for the model was maxdepth = 16.
```

```r
plot(mod_2)
```



Plot the tree using the `prp` function from the `rpart.plot` package to make a prettier tree.

```
prp(mod_2$finalModel)
```



Check how the model did in the training data and on Kaggle:

```
train_preds <- predict(mod_2, newdata = train)
rmsle(train_preds, train$count)
```

```
## [1] 1.290799
```

```
test_preds <- predict(mod_2, newdata = test)
write_test_preds(test_preds, "tree_tuned")
```

On Kaggle, this got … .

Tuning a random forest model

```
set.seed(825)
mod_3 <- train(count ~ season + holiday + workingday + weather +
                 temp + atemp + humidity + windspeed + year + hour +
                 month + yday,
               data = train,
               method = "rf",
               ntree = 10,
               importance = TRUE,
               trControl = fitControl,
               metric = "rmsle",
               maximize = FALSE,
               tuneLength = 5)
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-10
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

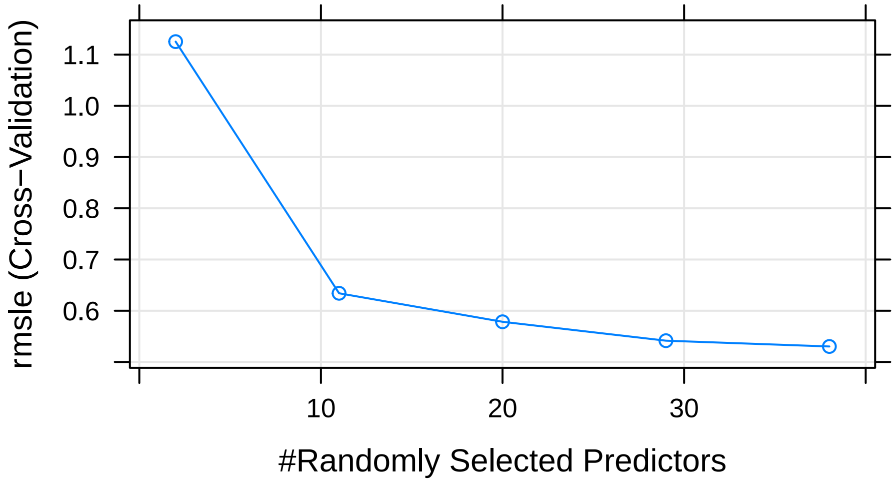
```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
mod_3
```

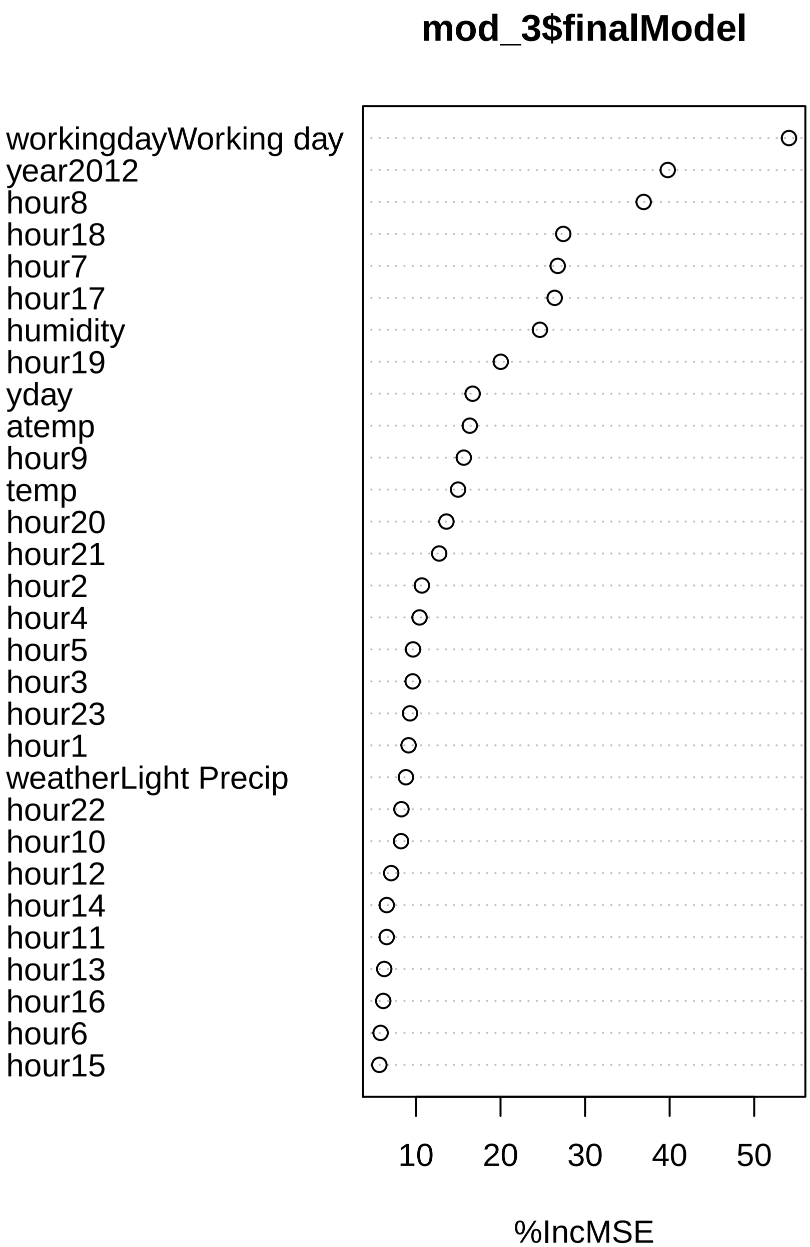
```
## Random Forest
##
## 10886 samples
##    15 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 8709, 8708, 8710, 8708, 8709
##
## Resampling results across tuning parameters:
##
##   mtry  rmsle      rmsle SD
##    2    1.1253169  0.02187762
##   11    0.6340923  0.02064834
##   20    0.5784320  0.01364140
```

```
##    29     0.5415765  0.01075632
##    38     0.5302327  0.01887970
##
## rmsle was used to select the optimal model using  the smallest value.
## The final value used for the model was mtry = 38.
```

```r
plot(mod_3)
```




```r
varImpPlot(mod_3$finalModel, type = 1)
```



Check how the model did in the training data and on Kaggle:

```
train_preds <- predict(mod_3, newdata = train)
rmsle(train_preds, train$count)
```

```
## [1] 0.3011643
```

```
test_preds <- predict(mod_3, newdata = test)
write_test_preds(test_preds, "rf_tuned")
```

On Kaggle, this got … .