Modeling in the Tidyverse - Comparing Models with Resampling

Max Kuhn (RStudio)

Loading

```r
library(tidymodels)
```

```
## ── Attaching packages ──────────────────────────────────────── tidymodels 0.0.2 ──
```

```
## ✔ broom     0.5.2     ✔ purrr     0.3.2
## ✔ dials     0.0.2     ✔ recipes   0.1.5
## ✔ dplyr     0.8.1     ✔ rsample   0.0.4
## ✔ ggplot2   3.1.0     ✔ tibble    2.1.1
## ✔ infer     0.4.0     ✔ yardstick 0.0.2
## ✔ parsnip   0.0.2
```

```
## ── Conflicts ───────────────────────────────────────────── tidymodels_conflicts() ──
## ✖ purrr::discard() masks scales::discard()
## ✖ dplyr::filter()  masks stats::filter()
## ✖ dplyr::lag()     masks stats::lag()
## ✖ recipes::step()  masks stats::step()
```

Previously

We fit a linear model with a fairly complex recipe and a boosted with minimal feature engineering.

Their resampled RMSE results were somewhat similar: 0.0799 for the linear regression and 0.0713 for boosting.

- These are in $\log_{10}$ units.

Let's get the RMSE values for each model into their own columns.

RMSE Results

```r
rmse_results <-
  bst_splits %>%
  mutate(
    boosting =
      map_dbl(
        boosting_res, ~ .x %>% filter(trees == 20 & .metric == "rmse") %>% pull(.estimate)
      ),
    linear_reg = map_dbl(lm_res, ~ .x %>% filter(.metric == "rmse") %>% pull(.estimate))
  ) %>%
  select(splits, id, boosting, linear_reg)

rmse_results %>% slice(1:5)
```

```
## #  10-fold cross-validation using stratification
## # A tibble: 5 x 4
##   splits           id     boosting linear_reg
## * <list>           <chr>     <dbl>      <dbl>
## 1 <split [2K/222]> Fold01   0.0654     0.0770
## 2 <split [2K/222]> Fold02   0.0681     0.0898
## 3 <split [2K/222]> Fold03   0.0766     0.0746
## 4 <split [2K/222]> Fold04   0.0795     0.0818
## 5 <split [2K/222]> Fold05   0.0738     0.0791
```

Resample-to-Resample Correlation

With resampling methods, it is common to see a correlation in the results across models.

Basically, some resamples are much more likely to produce good results (across models) than others. Similarly, some resamples are more difficult.

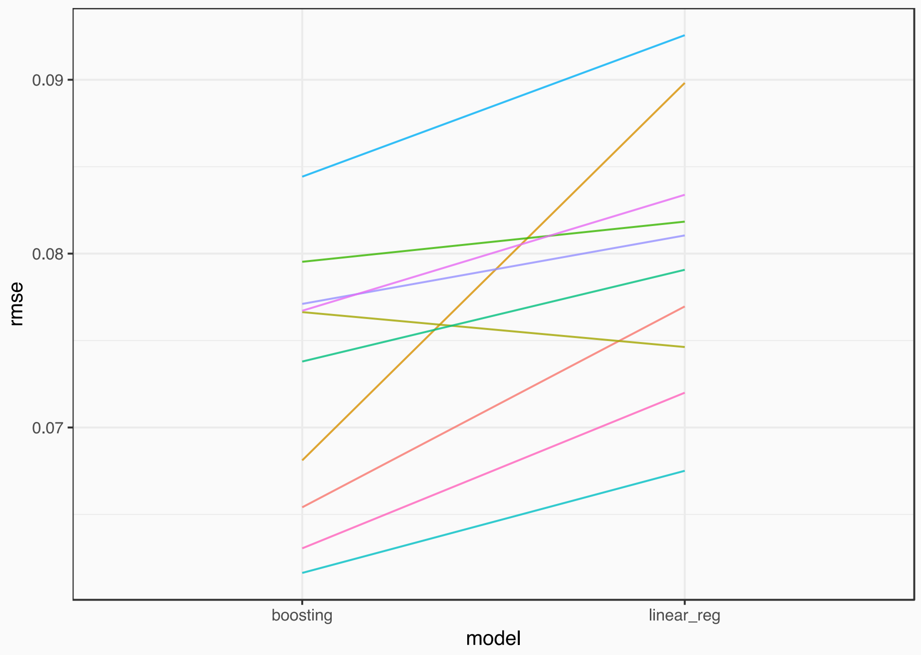
We have 10 *paired* measurements for two models `[*]` whose correlation is very high: 0.6603.

We can view these resampling results as *replicates* of a random process and use them to make *inferential statements* about performance but we have to account for the within-resample correlation.

`[*]` For boosting, there is some *optimization-bias* built into the RMSE estimate. This is a real issue but I have almost always found the amount of bias to be much smaller than the experimental noise.

Resample-to-Resample Correlation

```
rmse_results %>%
  select(-splits) %>%
  gather(model, rmse, -id) %>%
  ggplot() +
  aes(x = model, y = rmse, group = id, color = id) +
  geom_line(alpha = .8) +
  theme(legend.position = "none")
```



Collecting and Analyzing the Resampling Results

We *could* just do a paired t-test:

```
rmse_results %>%
  mutate(diffs = boosting - linear_reg) %>%
  pull(diffs) %>%
  t.test()
```

```
##
##     One Sample t-test
##
## data:  .
## t = -3.6, df = 9, p-value = 0.006
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.011756 -0.002726
## sample estimates:
## mean of x
## -0.007241
```

Noooooo! Down with p-values, right? This is an ASA conference!

The `tidyposterior` package is designed to estimate the relationship between the *outcome metrics* (i.e. RMSE) as a function of the model type (i.e. boosting) in a way that takes into account the resample-to-resample covariances that can occur.

A simple Bayesian linear model is used here for that purpose.

I recommend the book *Statistical Rethinking* if you are new to Bayesian analysis.

Bayesian Hierarchical Linear Model

If we did a basic ANOVA model to compare three models, it might look like:

$$RMSE = b_0 + b_1 m_1 + b_2 m_2$$

where the m_j are indicator variables for the model (boosting, linear regression, etc).

However, there are usually resample-to-resample effects. To account for this, we can make this ANOVA model *specific to a resample*:

$$RMSE_i = b_{i0} + b_{i1} m_1 + b_2 m_{i2}$$

Bayesian Hierarchical Linear Model

We might assume that each

$$RMSE_{ij} \sim N(\beta_{i0} + \beta_{ij}m_{ij}, \sigma^2)$$

and that the b parameters have some multivariate normal distribution with mean β and some covariance matrix. The distribution of the β values, along with a distribution for the variance parameter, are the *prior distributions*.

Bayesian analysis can be used to estimate these parameters. `tidyposterior` uses Stan to fit the model.

There are options to change the assumed distribution of the metric (i.e. gamma instead of normality) or to transform the metric to normality. Different variances per model can also be estimated and the priors can be changed.

Comparing Models using Bayesian Analysis

`tidyposterior::perf_mod` can take the `rset` object as input, configure the Bayesian model, and estimate the parameters.

Since we only have 10 replicates, I'd like to put a slightly wider prior on my intercept parameter to be a bit conservative.

```r
library(tidyposterior)
library(rstanarm) # needed for my prior
rmse_mod <-
  perf_mod(rmse_results,
           seed = 4344, iter = 5000,
           prior_intercept = student_t(df = 1))
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000492 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per trans
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
```

Showing the Posterior Distributions

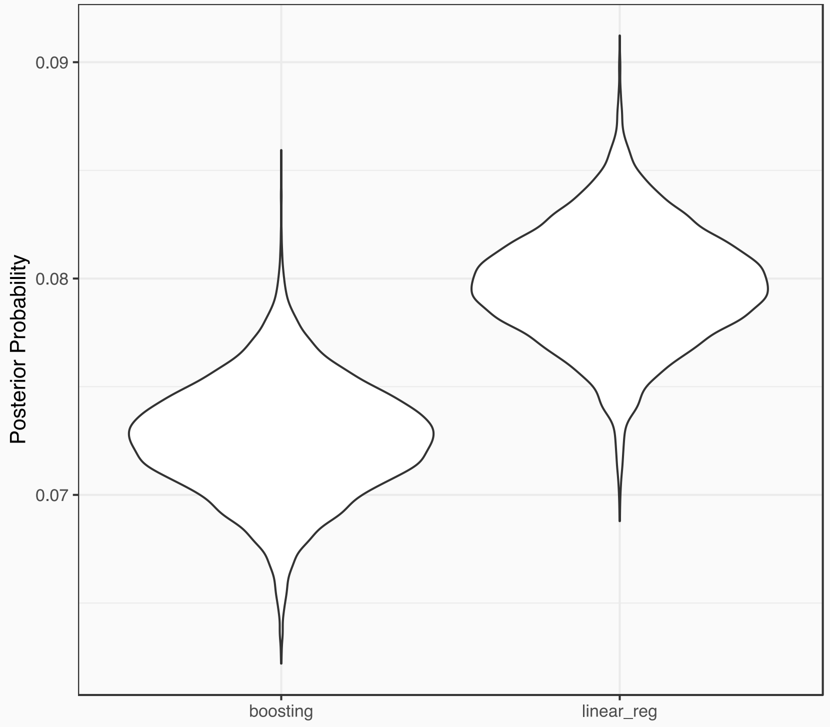
Bayesian analysis can produce probability distributions for the estimated parameters (aka the *posterior distributions*). These can be used to compare models.

```
posteriors <- tidy(rmse_mod, seed = 366784)
summary(posteriors)
```

```
## # A tibble: 2 x 4
##   model         mean  lower  upper
##   <chr>        <dbl>  <dbl>  <dbl>
## 1 boosting   0.0726 0.0683 0.0768
## 2 linear_reg 0.0797 0.0753 0.0840
```

These are 90% *credible intervals*.

```
ggplot(posteriors)
```



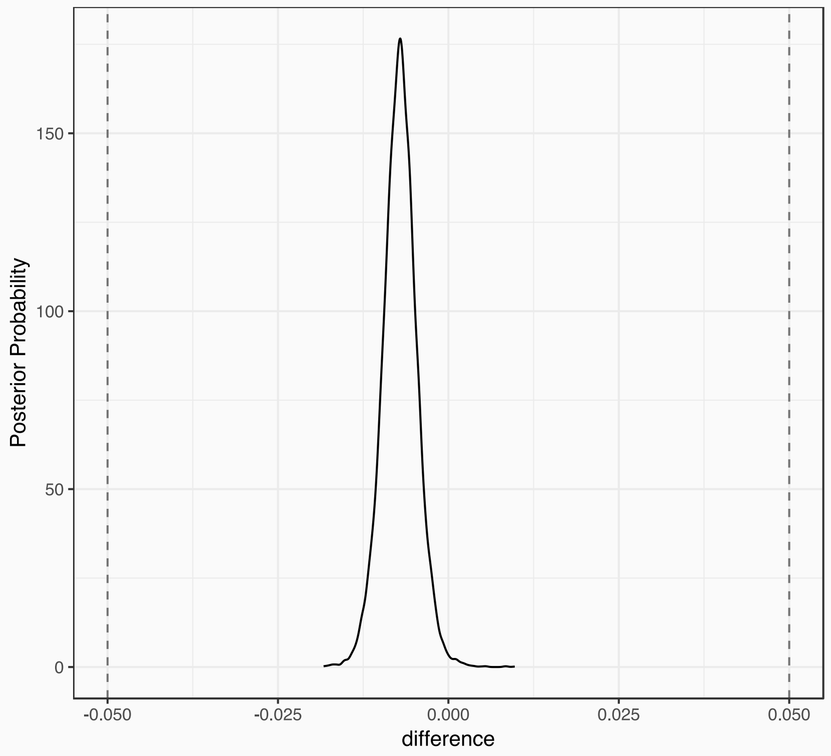
Comparing Models

Once the posteriors for each model are calculated, it is pretty easy to compute the posterior for the differences between models and plot them.

```
differences <- contrast_models(rmse_mod, seed = 2581)
```

If we know the size of a practical difference in RMSE values, this can be included into the analysis to get ROPE estimates (Region of Practical Equivalence).

```
ggplot(differences, size = 0.05)
```



Comparing Models

If we think that 0.05 log units is a real difference, we can assess which models are practically different from one another.

```
summary(differences, size = 0.05)
```

```
## # A tibble: 1 x 9
##   contrast               probability     mean    lower     upper  size pract_neg pract_equiv pract_pos
##   <chr>                        <dbl>    <dbl>    <dbl>     <dbl> <dbl>     <dbl>       <dbl>     <dbl>
## 1 boosting vs linear_reg      0.0059 -0.00712 -0.0112 -0.00300  0.05         0           1         0
```

`probability` is the probability that linear regression is better than boosting.

`pract_neg` is the probability that the difference in RMSE is practically negative based on our thoughts about `size`.