Modeling Run Performance

Joe Martin

10/27/2021

Garmin Data Modeling

The two most obvious primary target variables are average speed (avg_spd) in miles per hour, and average pace (avg_pace_sec) in seconds. A higher average speed and a lower average pace are the desired outcome when measuring performance over time. Reviewing the results of the two preliminary linear regression models, the more desirable variable is average pace, as it has stronger relationships with other variables.

```
# Create preliminary model

prelim_spd <- lm(avg_spd ~ ., df)
summary(prelim_spd)
```

```
##
## Call:
## lm(formula = avg_spd ~ ., data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.187959 -0.029389 -0.000816  0.028251  0.219902
##
## Coefficients: (1 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -6.402e+00  5.019e-01 -12.755  < 2e-16 ***
## distance                    1.031e-02  7.272e-03   1.417 0.157270
## avg_hr                      4.347e-03  1.026e-03   4.236 2.84e-05 ***
## max_hr                     -1.052e-03  6.662e-04  -1.579 0.115059
## avg_run_cadence             4.733e-02  1.259e-03  37.584  < 2e-16 ***
## max_run_cadence             1.806e-04  2.388e-04   0.756 0.449921
## total_ascent               -5.052e-05  7.741e-05  -0.653 0.514341
## total_decent                6.208e-05  7.328e-05   0.847 0.397386
## avg_stride                  5.142e+00  1.294e-01  39.720  < 2e-16 ***
## min_elevation               3.616e-04  1.058e-04   3.418 0.000697 ***
## max_elevation              -2.596e-05  1.096e-04  -0.237 0.812852
## avg_pace_sec               -1.011e-03  3.465e-04  -2.916 0.003745 **
## best_pace_sec              -9.307e-05  1.039e-04  -0.896 0.370737
## `sweat_loss(ml)`           -4.945e-05  1.336e-04  -0.370 0.711397
## aerobic_TE                 -8.122e-02  1.663e-02  -4.885 1.51e-06 ***
## aerobic_fctImpacting       -5.322e-03  9.255e-03  -0.575 0.565576
## aerobic_fctMaintaining      2.091e-02  1.771e-02   1.181 0.238502
## aerobic_fctOverreaching     4.604e-02  1.427e-02   3.225 0.001365 **
## anaerobic_value             1.303e-02  1.097e-02   1.187 0.235838
```

```
## anaerobic_fctMaintaining  -4.067e-03  1.741e-02  -0.234 0.815458
## anaerobic_fctNo Benefit   -4.778e-02  3.440e-02  -1.389 0.165630
## anaerobic_fctSome Benefit -6.501e-02  2.583e-02  -2.517 0.012246 *
## max_spd                   -3.014e-03  2.567e-03  -1.174 0.241070
## short_distanceY            1.336e-02  1.906e-02   0.701 0.483817
## middle_distanceY           1.843e-02  1.414e-02   1.303 0.193186
## long_distanceY                    NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04525 on 392 degrees of freedom
## Multiple R-squared:  0.9978, Adjusted R-squared:  0.9976
## F-statistic:  7263 on 24 and 392 DF,  p-value: < 2.2e-16
```

```
prelim_pace <- lm(avg_pace_sec ~ ., df)
summary(prelim_pace)
```

```
##
## Call:
## lm(formula = avg_pace_sec ~ ., data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.013  -3.313  -0.427   3.011  47.420
##
## Coefficients: (1 not defined because of singularities)
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                1.185e+03  6.190e+01  19.143  < 2e-16 ***
## distance                  -4.443e+00  1.027e+00  -4.326 1.93e-05 ***
## avg_hr                     3.786e-01  1.501e-01   2.522  0.01206 *
## max_hr                    -8.380e-03  9.638e-02  -0.087  0.93075
## avg_run_cadence           -1.787e+00  3.790e-01  -4.715 3.37e-06 ***
## max_run_cadence            5.933e-03  3.445e-02   0.172  0.86337
## total_ascent              -9.885e-03  1.116e-02  -0.886  0.37613
## total_decent               3.999e-03  1.057e-02   0.378  0.70547
## avg_stride                -2.333e+02  4.015e+01  -5.811 1.29e-08 ***
## min_elevation             -2.627e-02  1.542e-02  -1.703  0.08930 .
## max_elevation              2.642e-02  1.575e-02   1.678  0.09417 .
## best_pace_sec              2.566e-02  1.494e-02   1.718  0.08661 .
## `sweat_loss(ml)`           9.787e-02  1.862e-02   5.257 2.41e-07 ***
## aerobic_TE                -8.762e+00  2.429e+00  -3.606  0.00035 ***
## aerobic_fctImpacting      -4.257e+00  1.318e+00  -3.231  0.00134 **
## aerobic_fctMaintaining     5.590e+00  2.542e+00   2.199  0.02849 *
## aerobic_fctOverreaching    5.209e+00  2.069e+00   2.518  0.01220 *
## anaerobic_value           -1.135e+00  1.584e+00  -0.716  0.47435
## anaerobic_fctMaintaining   2.063e+00  2.509e+00   0.822  0.41144
## anaerobic_fctNo Benefit    1.926e-01  4.972e+00   0.039  0.96913
## anaerobic_fctSome Benefit  1.513e+00  3.754e+00   0.403  0.68709
## avg_spd                   -2.101e+01  7.205e+00  -2.916  0.00374 **
## max_spd                    6.393e-02  3.708e-01   0.172  0.86321
## short_distanceY           -5.004e-01  2.750e+00  -0.182  0.85569
## middle_distanceY          -1.782e-01  2.044e+00  -0.087  0.93054
## long_distanceY                    NA         NA      NA       NA
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.525 on 392 degrees of freedom
## Multiple R-squared:  0.9901, Adjusted R-squared:  0.9895
## F-statistic:  1641 on 24 and 392 DF,  p-value: < 2.2e-16
```

The ultimate goal of this model is to utilize data leading up to a performance event. As many races take place on Sunday and the typical long-distance run in this data set takes place on Sunday, the final linear regression model will begin with predicting Sunday performance.

To being predicting run performance, an initial linear regression model will be built below using all available data. Based on the preliminary linear regression above, an aerobic training effect that has a high impact (value between 4 and 4.9) is strongly related to average pace. This variable will be the target variable in the logistic regression that follows.

```
set.seed(456)
# Split data into training and testing sets
df_split <- initial_split(df, prop = 3/4)

train_df <- training(df_split)
test_df <- testing(df_split)

# Create recipe
pace_rec <- recipe(avg_pace_sec ~ ., data = train_df)

summary(pace_rec)
```

```
## # A tibble: 22 x 4
##    variable        type    role      source
##    <chr>           <chr>   <chr>     <chr>
##  1 distance        numeric predictor original
##  2 avg_hr          numeric predictor original
##  3 max_hr          numeric predictor original
##  4 avg_run_cadence numeric predictor original
##  5 max_run_cadence numeric predictor original
##  6 total_ascent    numeric predictor original
##  7 total_decent    numeric predictor original
##  8 avg_stride      numeric predictor original
##  9 min_elevation   numeric predictor original
## 10 max_elevation   numeric predictor original
## # ... with 12 more rows
```

```
lm_pace <- linear_reg() %>%
  set_engine("lm")

pace_wflow <- workflow()%>%
  add_model(lm_pace) %>%
  add_recipe(pace_rec)

pace_fit <- pace_wflow %>%
  fit(data = train_df)

tidy(pace_fit)
```

```
## # A tibble: 26 x 5
##    term              estimate std.error statistic  p.value
##    <chr>                <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)     1176.        74.7      15.7    1.18e-40
##  2 distance          -5.56       2.06     -2.70   7.35e- 3
##  3 avg_hr             0.396      0.189     2.09   3.73e- 2
##  4 max_hr             0.0510     0.120     0.424  6.72e- 1
##  5 avg_run_cadence   -1.71       0.455    -3.76   2.05e- 4
##  6 max_run_cadence   -0.00279    0.0430   -0.0647 9.48e- 1
##  7 total_ascent      -0.0161     0.0147   -1.10   2.74e- 1
##  8 total_decent       0.0106     0.0136    0.777  4.38e- 1
##  9 avg_stride      -238.        48.9      -4.86   1.95e- 6
## 10 min_elevation     -0.0265     0.0203   -1.30   1.94e- 1
## # ... with 16 more rows
```

```
predict(pace_fit, test_df)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
## # A tibble: 105 x 1
##    .pred
##    <dbl>
##  1  450.
##  2  448.
##  3  436.
##  4  440.
##  5  399.
##  6  415.
##  7  428.
##  8  435.
##  9  448.
## 10  434.
## # ... with 95 more rows
```

```
pace_aug <- augment(pace_fit, test_df)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
pace_aug %>% select(avg_pace_sec, .pred)
```

```
## # A tibble: 105 x 2
##    avg_pace_sec .pred
##           <dbl> <dbl>
##  1          447  450.
##  2          449  448.
##  3          432  436.
##  4          438  440.
##  5          391  399.
##  6          414  415.
```

```
##  7          419  428.
##  8          432  435.
##  9          444  448.
## 10          430  434.
## # ... with 95 more rows
```

The R Mean-Squared Error for this model is 5.24. In other words, this model can predict average pace within 5.24 seconds.

```
pace_error <- pace_aug %>%
  rmse(truth = avg_pace_sec, .pred)

pace_error
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        5.24
```

Logistic Regression

All Variables

This first logistic regression is meant to predict whether an activity highly impacts aerobic training. This variable is relevant because it is the highest measure for aerobic conditioning without being over-reaching. In this analysis, calories and other variables related to aerobic training effect were removed.

```
set.seed(456)

df2 <- df %>% mutate(high_impact = ifelse(aerobic_fct == "Highly Impacting", 1,0))
df2$high_impact <- factor(df2$high_impact)

# Split data into training and testing sets
df2_split <- initial_split(df2, prop = 3/4)

train_df2 <- training(df2_split)
test_df2 <- testing(df2_split)

# Create recipe. Use all variables except aerobic_TE and related
aerobic_rec <- recipe(high_impact ~ short_distance + middle_distance + long_distance +
                        max_spd + avg_spd + anaerobic_value + `sweat_loss(ml)` + best_pace_sec +
                        avg_pace_sec + max_elevation + min_elevation + avg_stride +
                        total_decent + total_ascent + max_run_cadence + avg_run_cadence +
                        max_hr + avg_hr + distance, data = train_df2)

summary(aerobic_rec)
```

```
## # A tibble: 20 x 4
##    variable        type    role      source
##    <chr>           <chr>   <chr>     <chr>
##  1 short_distance  nominal predictor original
##  2 middle_distance nominal predictor original
```

```
##  3 long_distance   nominal predictor original
##  4 max_spd         numeric predictor original
##  5 avg_spd         numeric predictor original
##  6 anaerobic_value numeric predictor original
##  7 sweat_loss(ml)  numeric predictor original
##  8 best_pace_sec   numeric predictor original
##  9 avg_pace_sec    numeric predictor original
## 10 max_elevation   numeric predictor original
## 11 min_elevation   numeric predictor original
## 12 avg_stride      numeric predictor original
## 13 total_decent    numeric predictor original
## 14 total_ascent    numeric predictor original
## 15 max_run_cadence numeric predictor original
## 16 avg_run_cadence numeric predictor original
## 17 max_hr          numeric predictor original
## 18 avg_hr          numeric predictor original
## 19 distance        numeric predictor original
## 20 high_impact     nominal outcome   original
```

```
log_reg <- logistic_reg() %>%
  set_engine("glm")

aero_wkfl <- workflow()%>%
  add_model(log_reg) %>%
  add_recipe(aerobic_rec)

aero_fit <- aero_wkfl %>%
  fit(data = train_df2)

tidy(aero_fit)
```

```
## # A tibble: 20 x 5
##    term              estimate std.error statistic  p.value
##    <chr>                <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)      -52.5     143.        -0.366   0.714
##  2 short_distanceY   -0.129     1.62      -0.0796  0.937
##  3 middle_distanceY   1.22      1.07       1.15    0.251
##  4 long_distanceY    NA        NA         NA      NA
##  5 max_spd           -0.769     1.12      -0.686   0.492
##  6 avg_spd          -20.8       6.50      -3.20    0.00138
##  7 anaerobic_value   -1.00      0.536     -1.87    0.0615
##  8 `sweat_loss(ml)`   0.0196    0.0289     0.677   0.498
##  9 best_pace_sec     -0.0201    0.0376    -0.536   0.592
## 10 avg_pace_sec      -0.106     0.120     -0.884   0.377
## 11 max_elevation     -0.0422    0.0158    -2.67    0.00756
## 12 min_elevation      0.0410    0.0135     3.03    0.00245
## 13 avg_stride        87.3      37.8        2.31    0.0209
## 14 total_decent       0.00683   0.00776    0.880   0.379
## 15 total_ascent       0.00178   0.00776    0.229   0.819
## 16 max_run_cadence    0.0432    0.0265     1.63    0.104
## 17 avg_run_cadence    0.727     0.351      2.07    0.0384
## 18 max_hr             0.00227   0.0959     0.0237  0.981
## 19 avg_hr             0.240     0.0828     2.91    0.00366
## 20 distance          -0.519     1.39      -0.375   0.708
```

```
predict(aero_fit, test_df2)
```

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

```
## # A tibble: 105 x 1
##    .pred_class
##    <fct>
##  1 0
##  2 0
##  3 0
##  4 0
##  5 0
##  6 0
##  7 0
##  8 1
##  9 1
## 10 0
## # ... with 95 more rows
```

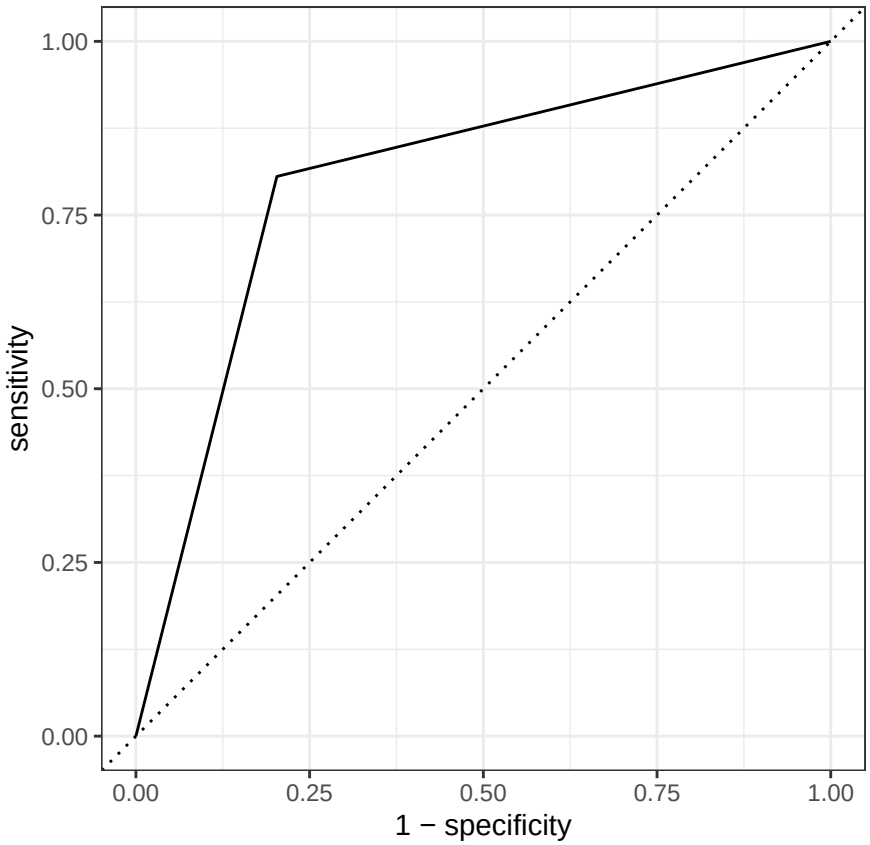
```
aero_aug <- augment(aero_fit, test_df2)
```

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

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

```
aero_aug %>% select(high_impact, .pred_class)
```

```
## # A tibble: 105 x 2
##    high_impact .pred_class
##    <fct>       <fct>
##  1 0           0
##  2 0           0
##  3 0           0
##  4 0           0
##  5 0           0
##  6 0           0
##  7 0           0
##  8 1           1
##  9 1           1
## 10 0           0
## # ... with 95 more rows
```

```
aero_aug$.pred_class <- as.character(aero_aug$.pred_class)
aero_aug$.pred_class <- as.numeric(aero_aug$.pred_class)

aero_aug %>%
  roc_curve(truth = high_impact, .pred_class, event_level="second") %>%
  autoplot()
```




```
aero_aug %>%
  roc_auc(truth = high_impact, .pred_class, event_level="second")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.801
```

Both of these models are acceptable for predicting performance. In the scope of this project, it seems that average pace is the best target variable to choose. When this model is later deployed, being able to predict average pace means being able to predict how well I would perform given a set of recent runs and the temperature and elevation of a location.

These analyses provide a good starting point for building a more complex model that can predict good performance. The possible next step is to use k-fold cross validation to enhance the quality of my training set. In this section, the random forest model will use k-fold cross validation and train with all variables.

```
pacman::p_load(tidymodels, ranger, parallel)

cores <- parallel::detectCores()

set.seed(456)

# Split data into training and testing sets
df_split <- initial_split(df, prop = 3/4)
```

```
train_df <- training(df_split)
test_df <- testing(df_split)

# Create recipe
rf_rec <- recipe(avg_pace_sec ~ ., data = train_df) %>%
          step_dummy(all_nominal_predictors())

folds <- vfold_cv(train_df, v = 10, repeats = 5, strata = avg_pace_sec)

summary(rf_rec)
```

```
## # A tibble: 22 x 4
##    variable        type    role      source
##    <chr>           <chr>   <chr>     <chr>
##  1 distance        numeric predictor original
##  2 avg_hr          numeric predictor original
##  3 max_hr          numeric predictor original
##  4 avg_run_cadence numeric predictor original
##  5 max_run_cadence numeric predictor original
##  6 total_ascent    numeric predictor original
##  7 total_decent    numeric predictor original
##  8 avg_stride      numeric predictor original
##  9 min_elevation   numeric predictor original
## 10 max_elevation   numeric predictor original
## # ... with 12 more rows
```

```
rf_mod <- rand_forest(mtry = tune(), min_n = tune(), trees = 1000) %>%
  set_engine("ranger", num.threads = cores) %>%
  set_mode("regression")

rf_wf <- workflow() %>%
  add_model(rf_mod) %>%
  add_recipe(rf_rec)

rf_res <- rf_wf %>%
  tune_grid(folds,
            grid = 25,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(rmse))
```

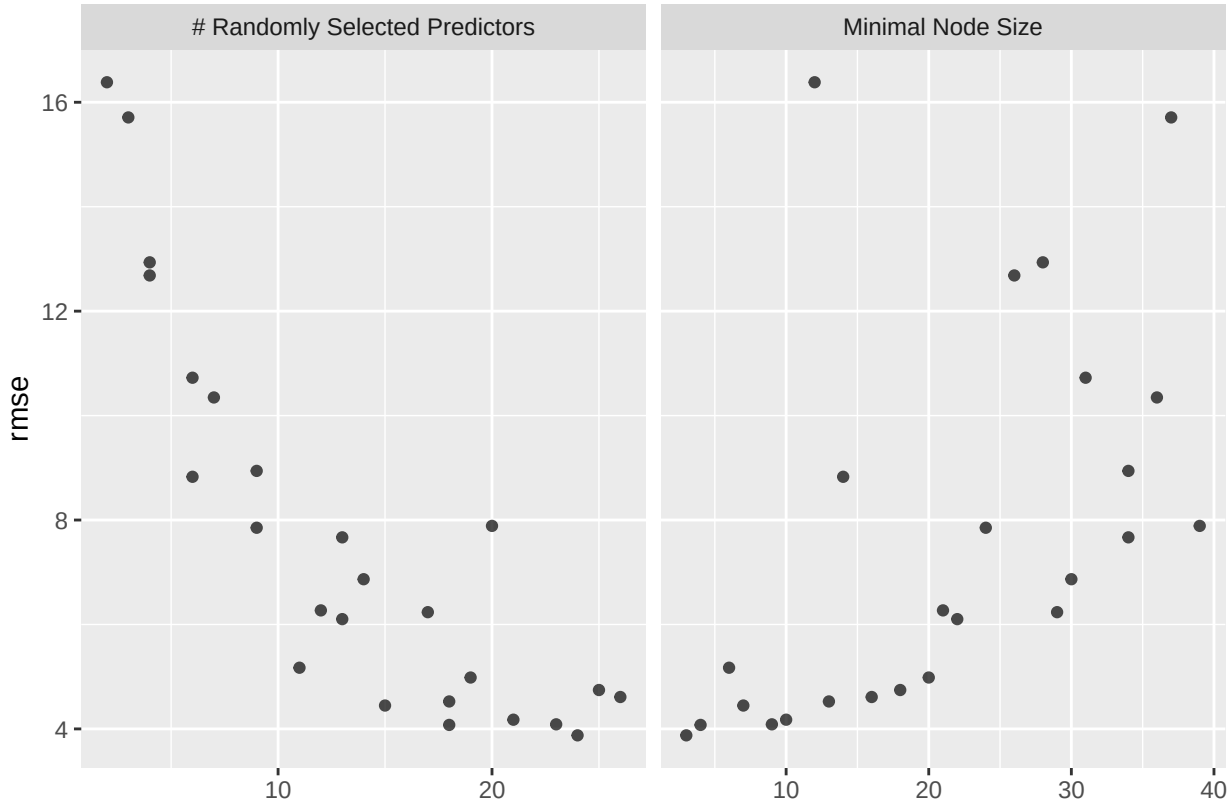
```
## i Creating pre-processing data to finalize unknown parameter: mtry
```

```
rf_res %>%
  show_best(metric = "rmse")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    24     3 rmse    standard    3.88    50   0.301 Preprocessor1_Model01
## 2    18     4 rmse    standard    4.08    50   0.308 Preprocessor1_Model25
## 3    23     9 rmse    standard    4.09    50   0.353 Preprocessor1_Model07
```

```
## 4    21    10 rmse    standard    4.18    50   0.362 Preprocessor1_Model08
## 5    15     7 rmse    standard    4.45    50   0.331 Preprocessor1_Model15
```

```
autoplot(rf_res)
```




```
rf_best <- rf_res %>%
  select_best(metric = "rmse")
rf_res %>% collect_predictions()
```

```
## # A tibble: 39,000 x 8
##    id      id2    .pred  .row  mtry min_n avg_pace_sec .config
##    <chr>   <chr>  <dbl> <int> <int> <int>        <dbl> <chr>
##  1 Repeat1 Fold01  455.    17    24     3          459 Preprocessor1_Model01
##  2 Repeat1 Fold01  485.    19    24     3          485 Preprocessor1_Model01
##  3 Repeat1 Fold01  581.    26    24     3          577 Preprocessor1_Model01
##  4 Repeat1 Fold01  625.    29    24     3          624 Preprocessor1_Model01
##  5 Repeat1 Fold01  441.    33    24     3          436 Preprocessor1_Model01
##  6 Repeat1 Fold01  609.    79    24     3          613 Preprocessor1_Model01
##  7 Repeat1 Fold01  529.    93    24     3          539 Preprocessor1_Model01
##  8 Repeat1 Fold01  561.    98    24     3          563 Preprocessor1_Model01
##  9 Repeat1 Fold01  468.   104    24     3          470 Preprocessor1_Model01
## 10 Repeat1 Fold01  512.   117    24     3          512 Preprocessor1_Model01
## # ... with 38,990 more rows
```

```
final_rf_wf <- rf_wf %>%
  finalize_workflow(rf_best)

final_fit_rf <- final_rf_wf %>%
  last_fit(df_split)

final_fit_rf %>% collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       2.55  Preprocessor1_Model1
## 2 rsq     standard       0.998 Preprocessor1_Model1
```

```
rf_rmse <-
  rf_res %>%
  collect_predictions(parameters = rf_best) %>%
  rmse(avg_pace_sec, .pred) %>%
  mutate(model = "Random Forest")
rf_rmse
```

```
## # A tibble: 1 x 4
##   .metric .estimator .estimate model
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard        4.44 Random Forest
```

This model predicts average pace within 4.4 seconds. In an attempt to improve the accuracy, some variables will be eliminated. This model ran with 21 predictors. Thinking about the purpose of this model (predicting potential race pace), there are features in this dataset which might not be available before the race. For example, speed would be related to the target (average pace) and not necessarily available as a predictor. The same goes for sweat loss. Therefore, these features will be removed.

```
set.seed(456)

#get rid of max_hr,min_elevation, max_elevation, and sweat_loss
df1 <- df %>% select(-max_spd, -avg_spd, -`sweat_loss(ml)`)

# Split data into training and testing sets
df1_split <- initial_split(df1, prop = 3/4)

train_df1 <- training(df1_split)
test_df1 <- testing(df1_split)

# Create recipe
rf_rec <- recipe(avg_pace_sec ~ ., data = train_df1) %>%
          step_dummy(all_nominal_predictors())

folds <- vfold_cv(train_df, v = 10, repeats = 5, strata = avg_hr)

summary(rf_rec)
```

```
## # A tibble: 19 x 4
```

```
##    variable        type    role      source
##    <chr>           <chr>   <chr>     <chr>
##  1 distance        numeric predictor original
##  2 avg_hr          numeric predictor original
##  3 max_hr          numeric predictor original
##  4 avg_run_cadence numeric predictor original
##  5 max_run_cadence numeric predictor original
##  6 total_ascent    numeric predictor original
##  7 total_decent    numeric predictor original
##  8 avg_stride      numeric predictor original
##  9 min_elevation   numeric predictor original
## 10 max_elevation   numeric predictor original
## 11 best_pace_sec   numeric predictor original
## 12 aerobic_TE      numeric predictor original
## 13 aerobic_fct     nominal predictor original
## 14 anaerobic_value numeric predictor original
## 15 anaerobic_fct   nominal predictor original
## 16 short_distance  nominal predictor original
## 17 middle_distance nominal predictor original
## 18 long_distance   nominal predictor original
## 19 avg_pace_sec    numeric outcome   original
```

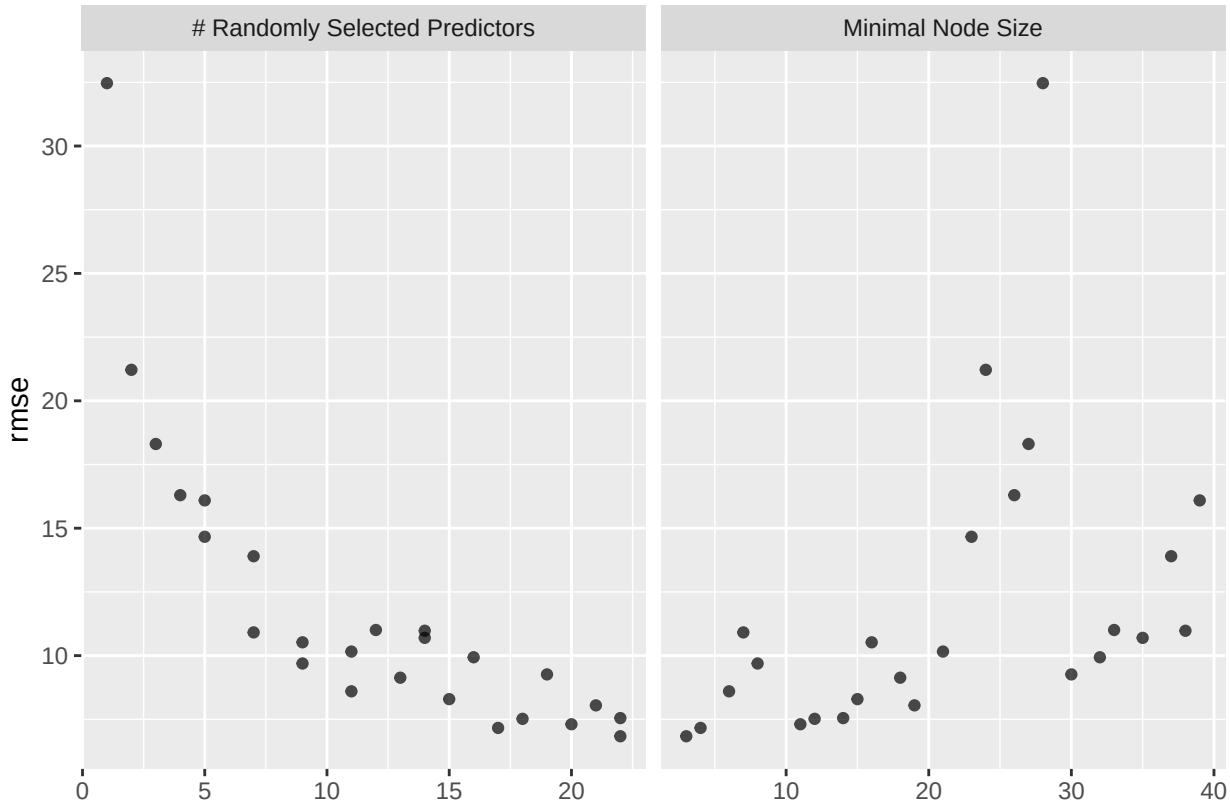
```
rf_mod <- rand_forest(mtry = tune(), min_n = tune(), trees = 1000) %>%
  set_engine("ranger", num.threads = cores) %>%
  set_mode("regression")

rf_wf <- workflow() %>%
  add_model(rf_mod) %>%
  add_recipe(rf_rec)

rf_res <- rf_wf %>%
  tune_grid(folds,
            grid = 25,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(rmse))
```

```
## i Creating pre-processing data to finalize unknown parameter: mtry
```

```
rf_res %>%
  show_best(metric = "rmse")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    22     3 rmse    standard    6.84    50   0.277 Preprocessor1_Model14
## 2    17     4 rmse    standard    7.16    50   0.281 Preprocessor1_Model17
## 3    20    11 rmse    standard    7.31    50   0.334 Preprocessor1_Model12
## 4    18    12 rmse    standard    7.52    50   0.338 Preprocessor1_Model06
## 5    22    14 rmse    standard    7.55    50   0.362 Preprocessor1_Model09
```

```
autoplot(rf_res)
```




```
rf_best <- rf_res %>%
  select_best(metric = "rmse")
rf_res %>% collect_predictions()
```

```
## # A tibble: 39,000 x 8
##    id      id2    .pred  .row  mtry min_n avg_pace_sec .config
##    <chr>   <chr>  <dbl> <int> <int> <int>        <dbl> <chr>
##  1 Repeat1 Fold01  443.    10     7     7          434 Preprocessor1_Model01
##  2 Repeat1 Fold01  494.    15     7     7          490 Preprocessor1_Model01
##  3 Repeat1 Fold01  429.    27     7     7          434 Preprocessor1_Model01
##  4 Repeat1 Fold01  591.    36     7     7          608 Preprocessor1_Model01
##  5 Repeat1 Fold01  424.    37     7     7          404 Preprocessor1_Model01
##  6 Repeat1 Fold01  398.    49     7     7          395 Preprocessor1_Model01
##  7 Repeat1 Fold01  434.    92     7     7          433 Preprocessor1_Model01
##  8 Repeat1 Fold01  489.    96     7     7          494 Preprocessor1_Model01
##  9 Repeat1 Fold01  433.   105     7     7          431 Preprocessor1_Model01
## 10 Repeat1 Fold01  501.   114     7     7          517 Preprocessor1_Model01
## # ... with 38,990 more rows
```

```
final_rf_wf <- rf_wf %>%
  finalize_workflow(rf_best)

final_fit_rf <- final_rf_wf %>%
  last_fit(df1_split)
```

```
final_fit_rf %>% collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       6.45  Preprocessor1_Model1
## 2 rsq     standard       0.987 Preprocessor1_Model1
```

```
rf_rmse <-
  rf_res %>%
  collect_predictions(parameters = rf_best) %>%
  rmse(avg_pace_sec, .pred) %>%
  mutate(model = "Random Forest")
rf_rmse
```

```
## # A tibble: 1 x 4
##   .metric .estimator .estimate model
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard        7.11 Random Forest
```

The accuracy (measured by RMSE) got worse (7.11). Try dropping more variables. This time, ascent, descent, aerobic factors and anaerobic factors.

Try running the model with tuned parameters.

```
set.seed(456)

# Split data into training and testing sets
df1_split <- initial_split(df1, prop = 3/4)

train_df1 <- training(df1_split)
test_df1 <- testing(df1_split)

# Create recipe
rf_rec <- recipe(avg_pace_sec ~ ., data = train_df1) %>%
          step_dummy(all_nominal_predictors())

folds <- vfold_cv(train_df, v = 10, repeats = 5, strata = avg_hr)

summary(rf_rec)
```

```
## # A tibble: 19 x 4
##    variable        type    role      source
##    <chr>           <chr>   <chr>     <chr>
##  1 distance        numeric predictor original
##  2 avg_hr          numeric predictor original
##  3 max_hr          numeric predictor original
##  4 avg_run_cadence numeric predictor original
##  5 max_run_cadence numeric predictor original
##  6 total_ascent    numeric predictor original
##  7 total_decent    numeric predictor original
##  8 avg_stride      numeric predictor original
```

```
##  9 min_elevation   numeric predictor original
## 10 max_elevation   numeric predictor original
## 11 best_pace_sec   numeric predictor original
## 12 aerobic_TE      numeric predictor original
## 13 aerobic_fct     nominal predictor original
## 14 anaerobic_value numeric predictor original
## 15 anaerobic_fct   nominal predictor original
## 16 short_distance  nominal predictor original
## 17 middle_distance nominal predictor original
## 18 long_distance   nominal predictor original
## 19 avg_pace_sec    numeric outcome   original
```

```
rf_mod <- rand_forest(mtry = 7, min_n = 7, trees = 1000) %>%
  set_engine("ranger", num.threads = cores) %>%
  set_mode("regression")

rf_wf <- workflow() %>%
  add_model(rf_mod) %>%
  add_recipe(rf_rec)

rf_res <- rf_wf %>%
  tune_grid(folds,
            grid = 25,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(rmse))
```

```
## Warning: No tuning parameters have been detected, performance will be evaluated
## using the resamples with no tuning. Did you want to [tune()] parameters?
```

```
rf_res %>%
  show_best(metric = "rmse")
```

```
## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard    10.9    50   0.352 Preprocessor1_Model1
```

```
rf_best <- rf_res %>%
  select_best(metric = "rmse")
rf_res %>% collect_predictions()
```

```
## # A tibble: 1,560 x 6
##    id      id2    .pred  .row avg_pace_sec .config
##    <chr>   <chr>  <dbl> <int>        <dbl> <chr>
##  1 Repeat1 Fold01  443.    10          434 Preprocessor1_Model1
##  2 Repeat1 Fold01  494.    15          490 Preprocessor1_Model1
##  3 Repeat1 Fold01  428.    27          434 Preprocessor1_Model1
##  4 Repeat1 Fold01  588.    36          608 Preprocessor1_Model1
##  5 Repeat1 Fold01  424.    37          404 Preprocessor1_Model1
##  6 Repeat1 Fold01  399.    49          395 Preprocessor1_Model1
##  7 Repeat1 Fold01  435.    92          433 Preprocessor1_Model1
##  8 Repeat1 Fold01  490.    96          494 Preprocessor1_Model1
```

```
##  9 Repeat1 Fold01  432.   105          431 Preprocessor1_Model1
## 10 Repeat1 Fold01  502.   114          517 Preprocessor1_Model1
## # ... with 1,550 more rows
```

```
final_rf_wf <- rf_wf %>%
  finalize_workflow(rf_best)

final_fit_rf <- final_rf_wf %>%
  last_fit(df1_split)

final_fit_rf %>% collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard      10.5   Preprocessor1_Model1
## 2 rsq     standard       0.966 Preprocessor1_Model1
```

```
rf_rmse <-
  rf_res %>%
  collect_predictions(parameters = rf_best) %>%
  rmse(avg_pace_sec, .pred) %>%
  mutate(model = "Random Forest")
rf_rmse
```

```
## # A tibble: 1 x 4
##   .metric .estimator .estimate model
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard        11.2 Random Forest
```

This final Random Forest model has a greater RMSE value than the previous one. One of the concerns with the dataset is the greater number of features (21 total predictors avialable). LASSO may be a good option to automate feature selection.

```
set.seed(456)
# Split data into training and testing sets
df_split <- initial_split(df1, prop = 3/4)

train_df <- training(df_split)
test_df <- testing(df_split)

# Create recipe
lasso_rec <- recipe(avg_pace_sec ~ ., data = train_df)

# create folds
folds <- vfold_cv(train_df, v = 10, repeats = 5, strata = avg_hr)

summary(lasso_rec)
```

```
## # A tibble: 19 x 4
##    variable        type    role      source
##    <chr>           <chr>   <chr>     <chr>
```

```
##  1 distance        numeric predictor original
##  2 avg_hr          numeric predictor original
##  3 max_hr          numeric predictor original
##  4 avg_run_cadence numeric predictor original
##  5 max_run_cadence numeric predictor original
##  6 total_ascent    numeric predictor original
##  7 total_decent    numeric predictor original
##  8 avg_stride      numeric predictor original
##  9 min_elevation   numeric predictor original
## 10 max_elevation   numeric predictor original
## 11 best_pace_sec   numeric predictor original
## 12 aerobic_TE      numeric predictor original
## 13 aerobic_fct     nominal predictor original
## 14 anaerobic_value numeric predictor original
## 15 anaerobic_fct   nominal predictor original
## 16 short_distance  nominal predictor original
## 17 middle_distance nominal predictor original
## 18 long_distance   nominal predictor original
## 19 avg_pace_sec    numeric outcome   original
```

```
lasso_mod <- linear_reg(penalty = tune(), mixture = 1) %>%
  set_engine("lm")

lasso_wkfl <- workflow() %>%
  add_model(lasso_mod) %>%
  add_recipe(lasso_rec)
```

```
# create penalty grid for tuning
lasso_grid <- tibble(penalty = 10^seq(-4, -1, length.out = 30))

# lowest penalties
lasso_grid %>% top_n(-5)
```

```
## Selecting by penalty
```

```
## # A tibble: 5 x 1
##    penalty
##      <dbl>
## 1 0.0001
## 2 0.000127
## 3 0.000161
## 4 0.000204
## 5 0.000259
```

```
#highest penalties
lasso_grid %>% top_n(5)
```

```
## Selecting by penalty
```

```
## # A tibble: 5 x 1
##   penalty
##     <dbl>
```

```
## 1  0.0386
## 2  0.0489
## 3  0.0621
## 4  0.0788
## 5  0.1
```

```
lasso_res <- lasso_wkfl %>%
  tune_grid(folds,
            grid = lasso_grid,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(rmse))
```

```
## Warning: No tuning parameters have been detected, performance will be evaluated
## using the resamples with no tuning. Did you want to [tune()] parameters?

## ! Fold01, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold02, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold03, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold04, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold05, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold06, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold07, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold08, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold09, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold10, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold01, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold02, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold03, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold04, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold05, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold06, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold07, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
```

```
## ! Fold08, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold09, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold10, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold01, Repeat3: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold02, Repeat3: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold03, Repeat3: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold04, Repeat3: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold05, Repeat3: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold06, Repeat3: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold07, Repeat3: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold08, Repeat3: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold09, Repeat3: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold10, Repeat3: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold01, Repeat4: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold02, Repeat4: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold03, Repeat4: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold04, Repeat4: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold05, Repeat4: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold06, Repeat4: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold07, Repeat4: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold08, Repeat4: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold09, Repeat4: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold10, Repeat4: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
## ! Fold01, Repeat5: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
```

```
## ! Fold02, Repeat5: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold03, Repeat5: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold04, Repeat5: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold05, Repeat5: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold06, Repeat5: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold07, Repeat5: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold08, Repeat5: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold09, Repeat5: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold10, Repeat5: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
```

```
top_models <- lasso_res %>%
  show_best("rmse")
top_models
```

```
## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard    7.51    50   0.352 Preprocessor1_Model1
```

```
lasso_best <- lasso_res %>%
  select_best("rmse")
lasso_best
```

```
## # A tibble: 1 x 1
##   .config
##   <chr>
## 1 Preprocessor1_Model1
```

```
final_lasso_wf <- lasso_wkfl %>%
  finalize_workflow(lasso_best)

final_lasso_fit <- final_lasso_wf %>%
  last_fit(df_split)
```

```
## ! train/test split: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
```

```
final_lasso_fit %>% collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       5.71  Preprocessor1_Model1
## 2 rsq     standard       0.990 Preprocessor1_Model1
```

The LASSO model does improve the RMSE and is likely the best path forward. The next steps in this project will be to further tune this model.