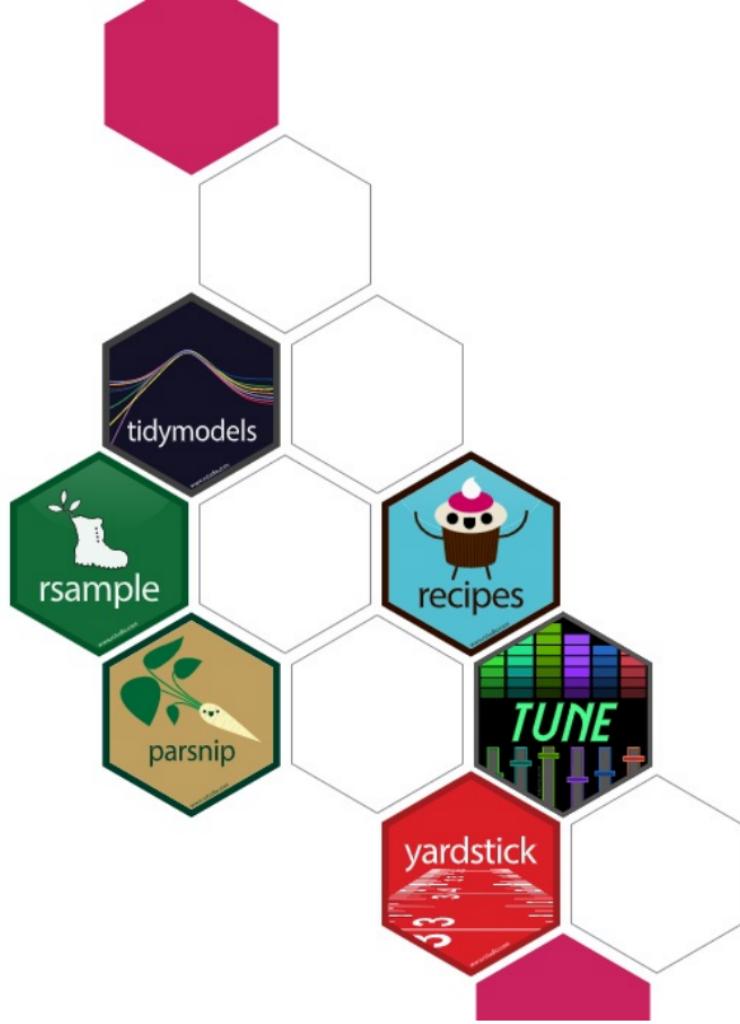


Lec 25 - More Tidymodels

Statistical Programming

Sta 323 | Spring 2022

Dr. Colin Rundel



Hotels Data

Original data from Antonio, Almeida, and Nunes (2019), Data dictionary

```
hotels = read_csv(  
  'https://tidymodels.org/start/case-study/hotels.csv'  
) %>%  
  mutate(  
    across(where(is.character), as.factor)  
)  
  
## # Rows: 50000 Columns: 23  
## — Column specification ——————  
## Delimiter: ","  
## chr (11): hotel, children, meal, country, market_segment...  
## dbl (11): lead_time, stays_in_weekend_nights, stays_in_...  
## date (1): arrival_date  
##  
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

This version of the data is slightly modified from the original data - see [gist](#) for the cleanup steps

The data

```
glimpse(hotels)
```

```
## Rows: 50,000
## Columns: 23
## $ hotel                               <fct> City_Hotel, City_Ho...
## $ lead_time                            <dbl> 217, 2, 95, 143, 13...
## $ stays_in_weekend_nights              <dbl> 1, 0, 2, 2, 1, 2, 0...
## $ stays_in_week_nights                <dbl> 3, 1, 5, 6, 4, 2, 2...
## $ adults                                <dbl> 2, 2, 2, 2, 2, 2, 2...
## $ children                             <fct> none, none, none, n...
## $ meal                                    <fct> BB, BB, BB, HB, HB, ...
## $ country                                <fct> DEU, PRT, GBR, ROU, ...
## $ market_segment                         <fct> Offline_TA/TO, Dire...
## $ distribution_channel                  <fct> TA/TO, Direct, TA/T...
## $ is_repeated_guest                     <dbl> 0, 0, 0, 0, 0, 0, 0...
## $ previous_cancellations               <dbl> 0, 0, 0, 0, 0, 0, 0...
## $ previous_bookings_not_canceled       <dbl> 0, 0, 0, 0, 0, 0, 0...
## $ reserved_room_type                   <fct> A, D, A, A, F, A, C...
## $ assigned_room_type                   <fct> A, K, A, A, F, A, C...
## $ booking_changes                      <dbl> 0, 0, 2, 0, 0, 0, 0...
## $ deposit_type                          <fct> No_Deposit, No_Depo...
## $ days_in_waiting_list                 <dbl> 0, 0, 0, 0, 0, 0, 0...
## $ customer_type                         <fct> Transient-Party, Tr...
## $ average_daily_rate                   <dbl> 80.75, 170.00, 8.00...
```

The model

Our goal is to develop a predictive model that is able to predict whether a booking will include children or not based on the other characteristics of the booking.

```
hotels %>%  
  count(children) %>%  
  mutate(prop = n/sum(n))
```

```
## # A tibble: 2 × 3  
##   children     n    prop  
##   <fct>     <int>  <dbl>  
## 1 children    4038  0.0808  
## 2 none        45962  0.919
```

Clustering the test/train split

```
set.seed(123)

splits = initial_split(hotels, strata = children)

hotel_train = training(splits)
hotel_test = testing(splits)

dim(hotel_train)

## [1] 37500    23

dim(hotel_test)

## [1] 12500    23
```

```
hotel_train %>%
  count(children) %>%
  mutate(prop = n/sum(n))
```

```
## # A tibble: 2 × 3
##   children     n   prop
##   <fct>     <int>  <dbl>
## 1 children  3027  0.0807
## 2 none      34473  0.919
```

```
hotel_test %>%
  count(children) %>%
  mutate(prop = n/sum(n))
```

```
## # A tibble: 2 × 3
##   children     n   prop
##   <fct>     <int>  <dbl>
## 1 children  1011  0.0809
## 2 none      11489  0.919
```

Logistic Regression model

```
show_engines("logistic_reg")  
  
## # A tibble: 7 × 2  
##   engine      mode  
##   <chr>       <chr>  
## 1 glm        classification  
## 2 glmnet     classification  
## 3 LiblineaR  classification  
## 4 spark      classification  
## 5 keras      classification  
## 6 stan       classification  
## 7 brulee     classification  
  
lr_model = logistic_reg() %>%  
  set_engine("glm")  
  
lr_model %>%  
  translate()  
  
## Logistic Regression Model Specification (classification)  
##  
## Computational engine: glm  
##  
## Model fit template:
```

Recipe

```
holidays = c("AllSouls", "AshWednesday", "ChristmasEve", "Easter",
            "ChristmasDay", "GoodFriday", "NewYearsDay", "PalmSunday")

lr_recipe = recipe(children ~ ., data = hotel_train) %>%
  step_date(arrival_date) %>%
  step_holiday(arrival_date, holidays = holidays) %>%
  step_rm(arrival_date) %>%
  step_rm(country) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors())

lr_recipe
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##   outcome          1
## predictor        22
##
## Operations:
##
## Date features from arrival_date
```

```
lr_recipe %>%  
  prep() %>%  
  bake(new_data = hotel_train)  
  
## # A tibble: 37,500 × 76  
##   lead_time stays_in_weekend_nights stays_in_week_n... adults  
##   <dbl>          <dbl>          <dbl>    <dbl>  
## 1     2              0              1      2  
## 2    95              2              5      2  
## 3    67              2              2      2  
## 4    47              0              2      2  
## 5    56              0              3      0  
## 6     6              2              2      2  
## 7   130              1              2      2  
## 8    27              0              1      1  
## 9    46              0              2      2  
## 10   423             1              1      2  
## # ... with 37,490 more rows, and 72 more variables:  
## #   is_repeated_guest <dbl>, previous_cancellations <dbl>,  
## #   previous_bookings_not_canceled <dbl>,  
## #   booking_changes <dbl>, days_in_waiting_list <dbl>,  
## #   average_daily_rate <dbl>,  
## #   total_of_special_requests <dbl>, children <fct>,  
## #   arrival_date_year <dbl>, arrival_date_AllSouls <dbl>, ...
```

Workflow

```
lr_work = workflow() %>%  
  add_model(lr_model) %>%  
  add_recipe(lr_recipe)
```

Fit

```
lr_fit = lr_work %>%  
  fit(data = hotel_train)  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1  
## occurred
```

```
lr_fit
```

```
## == Workflow [trained] =====  
## Preprocessor: Recipe  
## Model: logistic_reg()  
##  
## — Preprocessor -----  
## 6 Recipe Steps  
##  
## • step_date()  
## • step_holiday()  
## • step_rm()  
## • step_rm()  
## • step_dummy()  
## • step_zv()  
##  
## — Model -----  
##
```

Logistic regression predictions

```
lr_perf = lr_fit %>%  
  augment(new_data = hotel_train) %>%  
  select(children, starts_with(".pred"))  
  
lr_perf  
  
## # A tibble: 37,500 × 4  
##   children .pred_class .pred_children .pred_none  
##   <fct>    <fct>          <dbl>        <dbl>  
## 1 none     none         0.0861       0.914  
## 2 none     none         0.0178       0.982  
## 3 none     none         0.0101       0.990  
## 4 children children     0.931        0.0693  
## 5 children none         0.473        0.527  
## 6 children none         0.144        0.856  
## 7 none     none         0.0710       0.929  
## 8 none     none         0.0596       0.940  
## 9 none     none         0.0252       0.975  
## 10 none    none        0.0735       0.926  
## # ... with 37,490 more rows
```

Performance metrics (within-sample)

```
lr_perf %>%  
  conf_mat(children, .pred_class)
```

```
##           Truth  
## Prediction children  none  
##   children      1075   420  
##   none          1952 34053
```

```
lr_perf %>%  
  precision(children, .pred_class)
```

```
## # A tibble: 1 × 3  
##   .metric   .estimator .estimate  
##   <chr>     <chr>        <dbl>  
## 1 precision binary       0.719
```

```
lr_perf %>%  
  roc_auc(children, .pred_children)
```

```
## # A tibble: 1 × 3  
##   .metric .estimator .estimate  
##   <chr>   <chr>        <dbl>  
## 1 roc_auc binary       0.881
```

```
lr_perf %>%  
  yardstick::roc_curve(  
    children,  
    .pred_children  
  ) %>%  
  autoplot()
```

Performance metrics (out-of-sample)

```
lr_test_perf = lr_fit %>%
  augment(new_data = hotel_test) %>%
  select(children, starts_with(".pred"))

lr_test_perf %>%
  conf_mat(children, .pred_class)
```

```
##           Truth
## Prediction children  none
##   children      359   137
##   none          652 11352
```

```
lr_test_perf %>%
  precision(children, .pred_class)
```

```
## # A tibble: 1 × 3
##   .metric   .estimator .estimate
##   <chr>     <chr>        <dbl>
## 1 precision binary      0.724
```

```
lr_test_perf %>%
  roc_auc(children, .pred_children)
```

```
lr_test_perf %>%
  yardstick::roc_curve(
    children,
    .pred_children
  ) %>%
  autoplot()
```

Combining ROC curves

```
lr_test_perf %>%
  yardstick::roc_curve(
    children,
    .pred_children
  )

## # A tibble: 11,521 × 3
##       threshold specificity sensitivity
##           <dbl>      <dbl>      <dbl>
## 1     -Inf          0          1
## 2   2.22e-16        0          1
## 3   1.01e-11       0.0000870    1
## 4   4.96e-11       0.000174    1
## 5   1.16e- 9        0.000261    1
## 6   7.32e- 9        0.000348    1
## 7   8.80e- 9        0.000435    1
## 8   9.29e- 9        0.000522    1
## 9   9.67e- 9        0.000609    1
## 10  1.08e- 8        0.000696    1
## # ... with 11,511 more rows
```

```
lr_roc_train = lr_perf %>%
  yardstick::roc_curve(children, .pred_children)
```

```
bind_rows(
  lr_roc_train,
  lr_roc_test
) %>%
  ggplot(aes(x = 1 - specificity, y = sensitivity))
  geom_path(lwd = 1.5, alpha = 0.8) +
  geom_abline(lty = 3) +
  coord_equal()
```

Lasso

Lasso Model

For this we will be using the `glmnet` package which supports fitting lasso, ridge and elastic net models.

The mixture argument determines the type of model fit with: 1 -> Lasso, 0 -> Ridge, other -> elastic net.

```
lasso_model = logistic_reg(penalty = tune(), mixture = 1) %>%  
  set_engine("glmnet")  
  
lasso_model %>%  
  translate()  
  
## Logistic Regression Model Specification (classification)  
##  
## Main Arguments:  
##   penalty = tune()  
##   mixture = 1  
##  
## Computational engine: glmnet  
##  
## Model fit template:  
## glmnet::glmnet(x = missing_arg(), y = missing_arg(), weights = missing_arg(),
```

Lasso Recipe

Lasso (and Ridge) models are sensitive to the scale of the model features, and so a standard approach is to normalize all features before fitting the model.

```
lasso_recipe = lr_recipe %>%  
  step_normalize(all_predictors())
```

```
lasso_recipe %>%  
  prep() %>%  
  bake(new_data = hotel_train)
```

```
## # A tibble: 37,500 × 76  
##   lead_time stays_in_weekend_nights stays_in_week_n... adults  
##   <dbl>          <dbl>          <dbl>    <dbl>  
## 1 -0.858          -0.938         -0.767  0.337  
## 2  0.160           1.09          1.32   0.337  
## 3 -0.146           1.09         -0.245  0.337  
## 4 -0.365          -0.938         -0.245  0.337  
## 5 -0.267          -0.938          0.278 -3.59  
## 6 -0.814           1.09         -0.245  0.337  
## 7  0.544           0.0735        -0.245  0.337  
## 8 -0.584          -0.938         -0.767 -1.63  
## 9 -0.376          -0.938         -0.245  0.337  
## 10 3.75            0.0735        -0.767  0.337
```

Lasso workflow

```
lasso_work = workflow() %>%  
  add_model(lasso_model) %>%  
  add_recipe(lasso_recipe)
```

k-Folds for tuning

```
hotel_vf = rsample::vfold_cv(hotel_train, v=5, strata = children)
hotel_vf

## # 5-fold cross-validation using stratification
## # A tibble: 5 × 2
##   splits           id
##   <list>          <chr>
## 1 <split [30000/7500]> Fold1
## 2 <split [30000/7500]> Fold2
## 3 <split [30000/7500]> Fold3
## 4 <split [30000/7500]> Fold4
## 5 <split [30000/7500]> Fold5
```

grid search

```
lasso_grid = lasso_work %>%
  tune_grid(
    hotel_vf,
    grid = tibble(
      penalty = 10^seq(-4, -1, length.out = 10)
    ),
    control = control_grid(save_pred = TRUE),
    metrics = metric_set(roc_auc)
  )

lasso_grid

## # Tuning results
## # 5-fold cross-validation using stratification
## # A tibble: 5 × 5
##   splits              id    .metrics .notes   .predictions
##   <list>             <chr> <list>  <list>   <list>
## 1 <split [30000/7500]> Fold1 <tibble> <tibble> <tibble>
## 2 <split [30000/7500]> Fold2 <tibble> <tibble> <tibble>
## 3 <split [30000/7500]> Fold3 <tibble> <tibble> <tibble>
## 4 <split [30000/7500]> Fold4 <tibble> <tibble> <tibble>
## 5 <split [30000/7500]> Fold5 <tibble> <tibble> <tibble>
```

Results

```
lasso_grid %>%  
  collect_metrics()  
  
## # A tibble: 10 × 7  
##   penalty .metric .estimator  mean     n std_  
##       <dbl> <chr>    <chr>    <dbl> <int>   <d  
## 1  0.0001  roc_auc binary    0.877     5  0.00  
## 2  0.000215 roc_auc binary    0.877     5  0.00  
## 3  0.000464 roc_auc binary    0.877     5  0.00  
## 4  0.001   roc_auc binary    0.877     5  0.00  
## 5  0.00215 roc_auc binary    0.877     5  0.00  
## 6  0.00464 roc_auc binary    0.870     5  0.00  
## 7  0.01    roc_auc binary    0.853     5  0.00  
## 8  0.0215  roc_auc binary    0.824     5  0.00  
## 9  0.0464  roc_auc binary    0.797     5  0.00  
## 10 0.1    roc_auc binary    0.5      5  0
```

```
lasso_grid %>%  
  collect_metrics() %>%  
  ggplot(aes(x = penalty, y = mean)) +  
  geom_point() +  
  geom_line() +  
  ylab("Area under the ROC Curve") +  
  scale_x_log10(labels = scales::label_number())
```

"Best" models

```
lasso_grid %>%  
  show_best("roc_auc", n=10)  
  
## # A tibble: 10 × 7  
##   penalty .metric .estimator  mean     n std_err .config  
##   <dbl> <chr>   <chr>    <dbl> <int>  <dbl> <chr>  
## 1 0.001   roc_auc binary    0.877     5 0.00304 Preproce...  
## 2 0.00215  roc_auc binary    0.877     5 0.00263 Preproce...  
## 3 0.000464  roc_auc binary    0.877     5 0.00314 Preproce...  
## 4 0.000215  roc_auc binary    0.877     5 0.00316 Preproce...  
## 5 0.0001   roc_auc binary    0.877     5 0.00318 Preproce...  
## 6 0.00464  roc_auc binary    0.870     5 0.00253 Preproce...  
## 7 0.01    roc_auc binary    0.853     5 0.00249 Preproce...  
## 8 0.0215   roc_auc binary    0.824     5 0.00424 Preproce...  
## 9 0.0464   roc_auc binary    0.797     5 0.00400 Preproce...  
## 10 0.1    roc_auc binary     0.5      5 0       Preproce...
```

"Best" model

```
lasso_best = lasso_grid %>%
  collect_metrics() %>%
  mutate(mean = round(mean, 2)) %>%
  arrange(desc(mean), desc(penalty)) %>%
  slice(1)

lasso_best

## # A tibble: 1 × 7
##   penalty .metric .estimator  mean     n std_err .config
##       <dbl> <chr>    <chr>    <dbl> <int>  <dbl> <chr>
## 1 0.00215 roc_auc binary     0.88      5 0.00263 Preprocess...
```

Extracting predictions

Since we used `control_grid(save_pred = TRUE)` with `tune_grid()` we can recover the predictions for the out-of-sample values for each fold:

```
lasso_train_perf = lasso_grid %>%
  collect_predictions(parameters = lasso_best)
lasso_train_perf

## # A tibble: 37,500 × 7
##   id    .pred_children .pred_none .row penalty children
##   <chr>      <dbl>     <dbl> <int>    <dbl> <fct>
## 1 Fold1       0.366     0.634     5  0.00215 children
## 2 Fold1       0.144     0.856     6  0.00215 children
## 3 Fold1       0.0542    0.946    19  0.00215 none
## 4 Fold1       0.0266    0.973    21  0.00215 none
## 5 Fold1       0.106     0.894    22  0.00215 children
## 6 Fold1       0.0286    0.971    23  0.00215 none
## 7 Fold1       0.0205    0.980    30  0.00215 none
## 8 Fold1       0.0192    0.981    31  0.00215 none
## 9 Fold1       0.0431    0.957    32  0.00215 none
## 10 Fold1      0.0532    0.947   35  0.00215 none
## # ... with 37,490 more rows, and 1 more variable:
## #   .config <chr>
```

Re-fitting

Typically with a tuned model we will refit using the complete test data and the "best" parameter value(s),

```
lasso_work_tuned = update_model(  
  lasso_work,  
  logistic_reg(  
    mixture = 1,  
    penalty = lasso_best$penalty  
  ) %>%  
  set_engine("glmnet")  
)  
  
lasso_fit = lasso_work_tuned %>%  
  fit(data=hotel_train)
```

Test Performance (out-of-sample)

```
lasso_test_perf = lasso_fit %>%
  augment(new_data = hotel_test) %>%
  select(children, starts_with(".pred"))

lasso_test_perf %>%
  conf_mat(children, .pred_class)
```

```
##           Truth
## Prediction children  none
##   children      330   109
##   none          681 11380
```

```
lasso_test_perf %>%
  precision(children, .pred_class)
```

```
## # A tibble: 1 × 3
##   .metric   .estimator .estimate
##   <chr>     <chr>        <dbl>
## 1 precision binary     0.752
```

```
lasso_test_perf %>%
  roc_auc(children, .pred_children)
```

```
lasso_roc = lasso_test_perf %>%
  yardstick::roc_curve(
    children,
    .pred_children
  ) %>%
  mutate(name = "lasso - test")
lasso_roc %>%
  autoplot()
```

Comparing models

Random Forest

Random forest models

```
show_engines("rand_forest")
```

```
## # A tibble: 6 × 2
##   engine      mode
##   <chr>       <chr>
## 1 ranger     classification
## 2 ranger     regression
## 3 randomForest classification
## 4 randomForest regression
## 5 spark      classification
## 6 spark      regression
```

```
rf_model = rand_forest(mtry = tune(), min_n = tune(), trees = 100) %>%
  set_engine("ranger", num.threads = 8) %>%
  set_mode("classification")
```

Recipe & workflow

We skip dummy coding in the recipe as it is not needed by ranger,

```
rf_recipe = recipe(children ~ ., data = hotel_train) %>%  
  step_date(arrival_date) %>%  
  step_holiday(arrival_date, holidays = holidays) %>%  
  step_rm(arrival_date) %>%  
  step_rm(country)
```

```
rf_work = workflow() %>%  
  add_model(rf_model) %>%  
  add_recipe(rf_recipe)
```

Tuning

```
rf_work %>%  
  parameters()  
  
## Warning: `parameters.workflow()` was deprecated  
## Please use `hardhat::extract_parameter_set_dial`  
  
## Collection of 2 parameters for tuning  
##  
##  identifier    type    object  
##        mtry     mtry  nparam[?]  
##        min_n   min_n  nparam[+]  
##  
## Model parameters needing finalization:  
##      # Randomly Selected Predictors ('mtry')  
##  
## See `?dials::finalize` or `?dials::update.parameter`
```

```
rf_grid = rf_work %>%  
  tune_grid(  
    hotel_vf,  
    grid = 10,  
    control = control_grid(save_pred = TRUE),  
    metrics = metric_set(roc_auc)  
  )
```

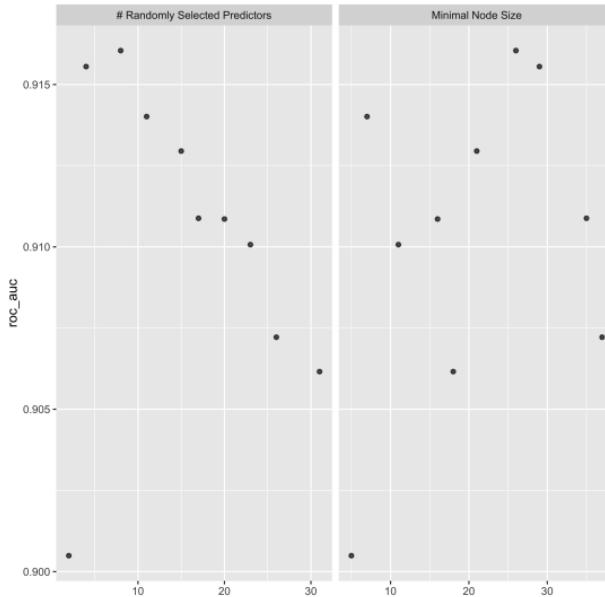
```
## i Creating pre-processing data to finalize unknown p
```

"Best" parameters

```
rf_grid %>%  
  show_best(metric = "roc_auc")
```

```
## # A tibble: 5 × 8  
##   mtry min_n .metric .estimator  mean     n st  
##   <int> <int> <chr>    <chr>     <dbl> <int>  
## 1     8     26 roc_auc binary     0.916     5 0.  
## 2     4     29 roc_auc binary     0.916     5 0.  
## 3    11      7 roc_auc binary     0.914     5 0.  
## 4    15     21 roc_auc binary     0.913     5 0.  
## 5    17     35 roc_auc binary     0.911     5 0.
```

```
autoplot(rf_grid)
```



Refitting

```
(rf_best = rf_grid %>%  
  select_best(metric = "roc_auc"))  
  
## # A tibble: 1 × 3  
##   mtry min_n .config  
##   <int> <int> <chr>  
## 1     8    26 Preprocessor1_Model06  
  
rf_work_tuned = update_model(  
  rf_work,  
  rand_forest(  
    trees=100,  
    mtry = rf_best$mtry,  
    min_n = rf_best$min_n  
  ) %>%  
  set_engine("ranger", num.threads = 8) %>%  
  set_mode("classification")  
)  
  
rf_fit = rf_work_tuned %>%  
  fit(data=hotel_train)
```

Test Performance (out-of-sample)

```
rf_test_perf = rf_fit %>%
  augment(new_data = hotel_test) %>%
  select(children, starts_with(".pred"))

rf_test_perf %>%
  conf_mat(children, .pred_class)
```

```
##          Truth
## Prediction children  none
##   children      402    69
##   none          609 11420
```

```
rf_test_perf %>%
  precision(children, .pred_class)
```

```
## # A tibble: 1 × 3
##   .metric   .estimator .estimate
##   <chr>     <chr>        <dbl>
## 1 precision binary     0.854
```

```
rf_test_perf %>%
  roc_auc(children, .pred_children)
```

```
rf_roc = rf_test_perf %>%
  yardstick::roc_curve(
    children,
    .pred_children
  ) %>%
  mutate(name = "RF - test")
rf_roc %>%
  autoplot()
```

Comparing models