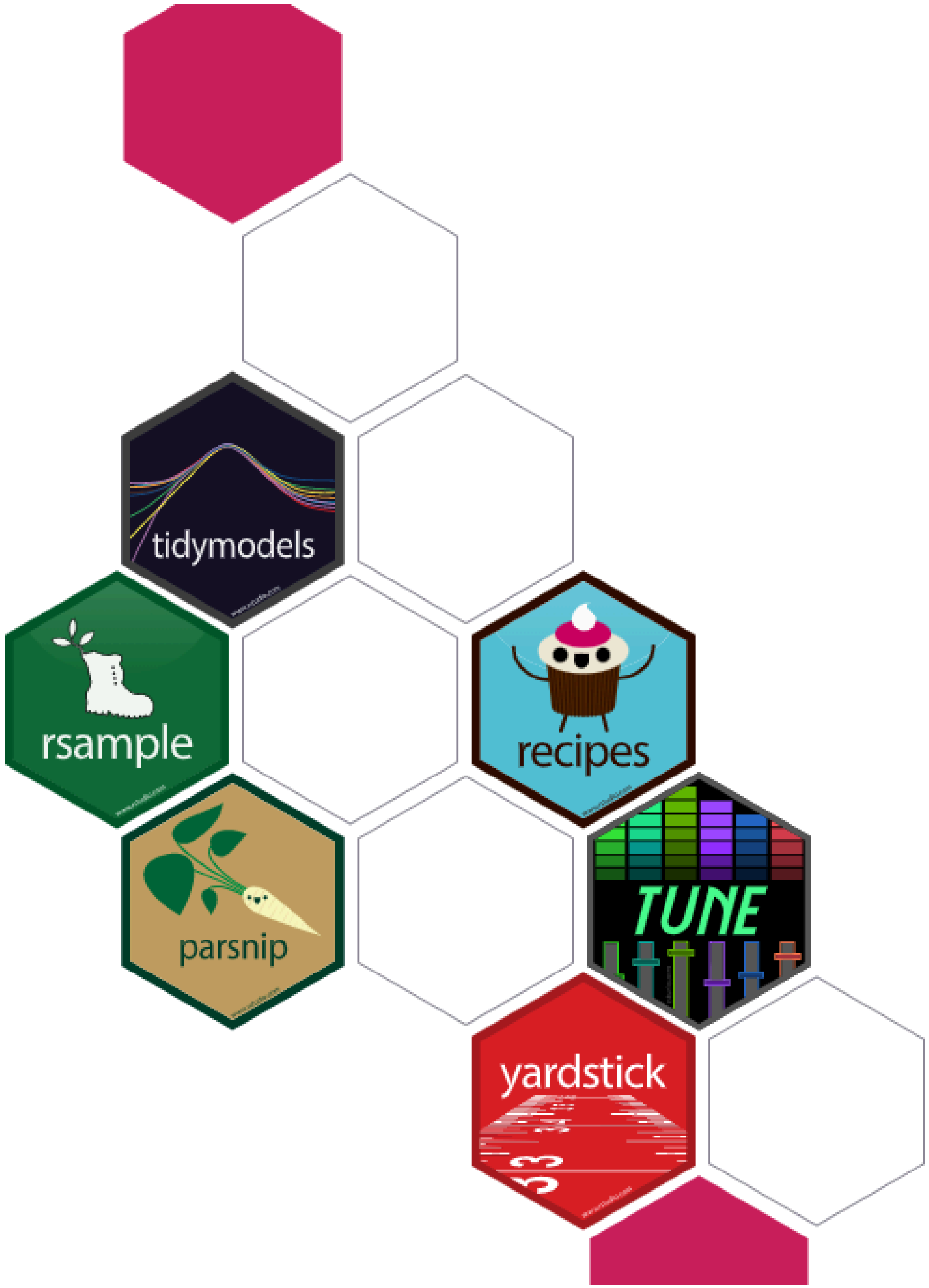
More Tidymodels

Lecture 23

Dr. Colin Rundel


tidymodels
rsample
recipes
parsnip
TUNE
yardstick

Hotels Data

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

```r
hotels = read_csv(
  'https://tidymodels.org/start/case-study/hotels.csv'
) |>
  mutate(
    across(where(is.character), as.factor)
  )
```

The data

```r
glimpse(hotels)
```

```
Rows: 50,000
Columns: 23
$ hotel                          <fct> City_Hotel, City_Hotel, Resort_Hotel, Resort_Hotel, Re…
$ lead_time                      <dbl> 217, 2, 95, 143, 136, 67, 47, 56, 80, 6, 130, 27, 16, …
$ stays_in_weekend_nights        <dbl> 1, 0, 2, 2, 1, 2, 0, 0, 0, 2, 1, 0, 1, 0, 1, 1, 1, 4, …
$ stays_in_week_nights           <dbl> 3, 1, 5, 6, 4, 2, 2, 3, 4, 2, 2, 1, 2, 2, 1, 1, 2, 7, …
$ adults                         <dbl> 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, …
$ children                       <fct> none, none, none, none, none, none, children, children…
$ meal                           <fct> BB, BB, BB, HB, HB, SC, BB, BB, BB, BB, BB, BB, BB, BB…
$ country                        <fct> DEU, PRT, GBR, ROU, PRT, GBR, ESP, ESP, FRA, FRA, FRA,…
$ market_segment                 <fct> Offline_TA/TO, Direct, Online_TA, Online_TA, Direct, O…
$ distribution_channel           <fct> TA/TO, Direct, TA/TO, TA/TO, Direct, TA/TO, Direct, TA…
$ is_repeated_guest              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ previous_cancellations         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ previous_bookings_not_canceled <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ reserved_room_type             <fct> A, D, A, A, F, A, C, B, D, A, A, D, A, D, A, A, D, A, …
$ assigned_room_type             <fct> A, K, A, A, F, A, C, A, D, A, D, D, A, D, A, A, D, A, …
```

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
```

Stratifying the test/train split

```r
set.seed(123)

splits = initial_split(
  hotels, strata = children
)

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

```r
dim(hotel_train)
```

```
[1] 37500    23
```

```r
dim(hotel_test)
```

```
[1] 12500    23
```

```r
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
```

```r
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

```r
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
```

```r
lr_model = logistic_reg() |>
  set_engine("glm")
```

```r
translate(lr_model)
```

```
Logistic Regression Model Specification (classification)

Computational engine: glm

Model fit template:
stats::glm(formula = missing_arg(), data = missing_arg(), weights = missing_arg(),
    family = stats::binomial)
```

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
```

```r
lr_recipe |>
  prep() |>
  bake(new_data = hotel_train)
```

```
# A tibble: 37,500 × 76
   lead_time stays_in_weekend_nights stays_in_week_nights adults is_repeated_guest
       <dbl>                   <dbl>                <dbl>  <dbl>             <dbl>
 1         2                       0                    1      2                 0
 2        95                       2                    5      2                 0
 3        67                       2                    2      2                 0
 4        47                       0                    2      2                 0
 5        56                       0                    3      0                 0
 6         6                       2                    2      2                 0
 7       130                       1                    2      2                 0
 8        27                       0                    1      1                 0
 9        46                       0                    2      2                 0
10       423                       1                    1      2                 0
# ℹ 37,490 more rows
# ℹ 71 more variables: 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 <int>,
```

Workflow

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

```
══ Workflow ════════════════════════════════════════════════
Preprocessor: Recipe
Model: logistic_reg()

── Preprocessor ────────────────────────────────────────────
6 Recipe Steps

• step_date()
• step_holiday()
• step_rm()
• step_rm()
• step_dummy()
• step_zv()

── Model ───────────────────────────────────────────────────
Logistic Regression Model Specification (classification)
```

Fit

```r
( lr_fit = lr_work |>
    fit(data = hotel_train) )
```

```
══ Workflow [trained] ══════════════════════════════════════
Preprocessor: Recipe
Model: logistic_reg()

── Preprocessor ────────────────────────────────────────────
6 Recipe Steps

• step_date()
• step_holiday()
• step_rm()
• step_rm()
• step_dummy()
• step_zv()

── Model ───────────────────────────────────────────────────

Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
```

Tidy

```r
lr_fit |>
  broom::tidy()
```

```
# A tibble: 76 × 5
   term                  estimate std.error statistic  p.value
   <chr>                    <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)           -2.54e+2   2.66e+2    -0.954 3.40e- 1
 2 lead_time             -1.29e-3   3.50e-4    -3.68  2.38e- 4
 3 stays_in_weekend_n…    5.23e-2   3.92e-2     1.33  1.82e- 1
 4 stays_in_week_nigh… -3.43e-2   1.83e-2    -1.88  6.07e- 2
 5 adults                 7.33e-1   5.03e-2    14.6   4.39e-48
 6 is_repeated_guest      3.96e-1   2.15e-1     1.84  6.58e- 2
 7 previous_cancellat…    2.15e-1   5.41e-1     0.397 6.92e- 1
 8 previous_bookings_…    3.73e-1   1.07e-1     3.49  4.86e- 4
 9 booking_changes       -2.40e-1   2.53e-2    -9.49  2.37e-21
10 days_in_waiting_li…    6.42e-3   5.45e-3     1.18  2.39e- 1
# ℹ 66 more rows
```

Logistic regression predictions

```r
( lr_train_perf = lr_fit |>
    augment(new_data = hotel_train) |>
    select(children, starts_with(".pred")) )
```

```
# 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
# ℹ 37,490 more rows
```

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

```
# A tibble: 12,500 × 4
   children .pred_class .pred_children .pred_none
   <fct>    <fct>                <dbl>      <dbl>
 1 none     none              0.00854       0.991
 2 none     none              0.0202        0.980
 3 none     children          0.757         0.243
 4 none     none              0.0373        0.963
 5 none     none              0.000975      0.999
 6 none     none              0.000474      1.00
 7 none     none              0.0736        0.926
 8 none     none              0.0748        0.925
 9 none     none              0.0532        0.947
10 none     none              0.0794        0.921
# ℹ 12,490 more rows
```

Performance metrics (within-sample)

```r
conf_mat(lr_train_perf, children, .pred_class)
```

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

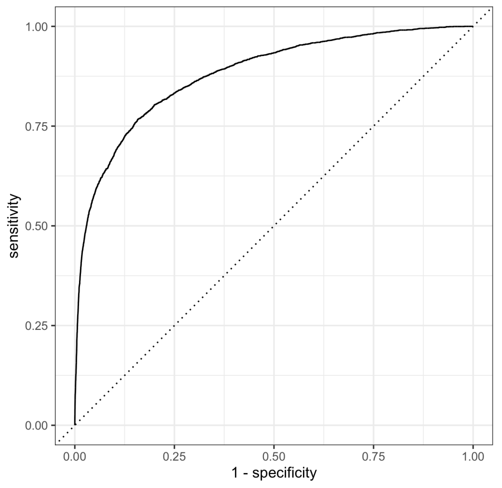
```r
accuracy(lr_train_perf, children, .pred_class)
```

```
# A tibble: 1 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.937
```

```r
precision(lr_train_perf, children, .pred_class)
```

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

```r
yardstick::roc_curve(lr_train_perf, children, .p
  autoplot()
```




```r
roc_auc(lr_train_perf, children, .pred_children)
```

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

Performance metrics (out-of-sample)

```r
conf_mat(lr_test_perf, children, .pred_class)
```

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

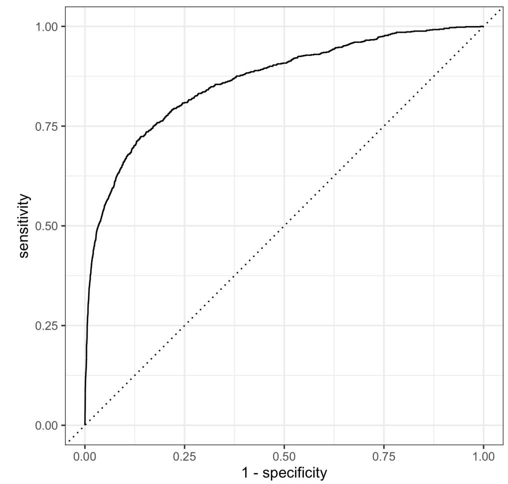
```r
accuracy(lr_test_perf, children, .pred_class)
```

```
# A tibble: 1 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.937
```

```r
precision(lr_test_perf, children, .pred_class)
```

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

```r
yardstick::roc_curve(lr_test_perf, children, .pr
  autoplot()
```

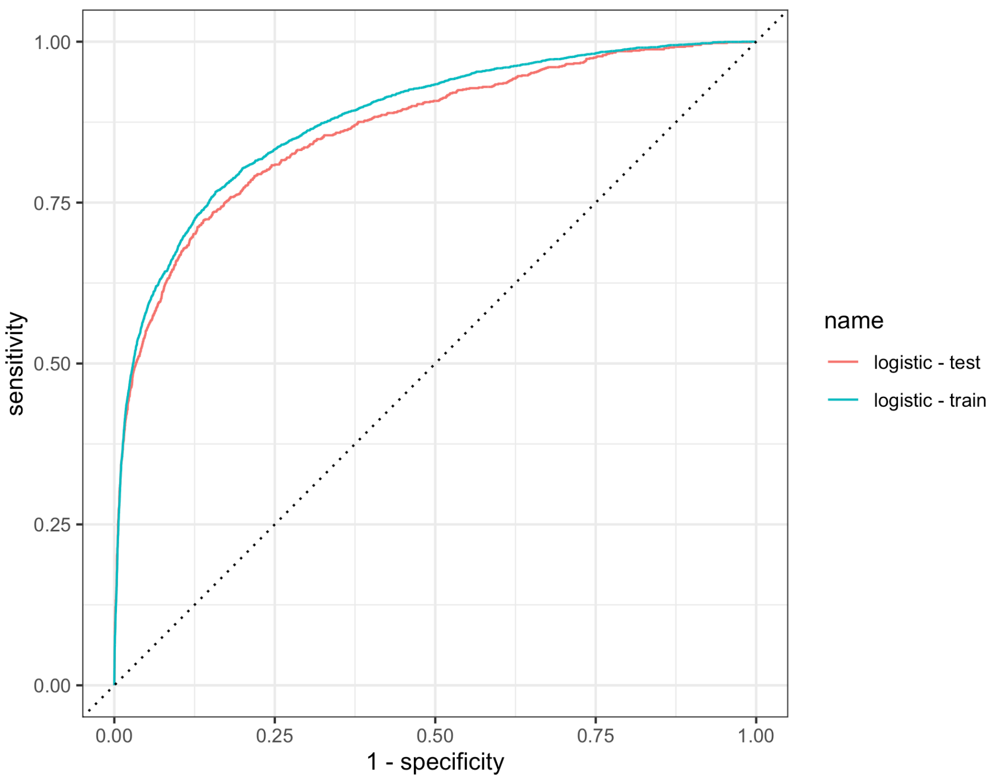



```r
roc_auc(lr_test_perf, children, .pred_children)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.864
```

Combining ROC curves

```r
bind_rows(
  lr_train_perf |> mutate(name = "logistic - train"),
  lr_test_perf |> mutate(name = "logistic - test")
) |>
  group_by(name) |>
  yardstick::roc_curve(children, .pred_children) |>
  autoplot()
```



Lasso

Lasso Model

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

```r
lasso_model = logistic_reg(penalty = tune(), mixture = 1) |>
  set_engine("glmnet")
```

- `mixture` determines the type of model fit
  - `1` for Lasso,
  - `0` for Ridge,
  - other for elastic net.
- `penalty` is λ in the lasso model, scales the penalty for coefficient size.

```r
lasso_model |>
  hardhat::extract_parameter_set_dials()
```

```
Collection of 1 parameters for tuning

 identifier    type    object
    penalty penalty nparam[+]
```

```r
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(),
    alpha = 1, family = "binomial")
```

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_nights
       <dbl>                   <dbl>                <dbl>
 1    -0.858                 -0.938               -0.767
 2     0.160                  1.09                 1.32
 3    -0.146                  1.09                -0.245
 4    -0.365                 -0.938               -0.245
 5    -0.267                 -0.938                0.278
 6    -0.814                  1.09                -0.245
 7     0.544                  0.0735              -0.245
 8    -0.584                 -0.938               -0.767
 9    -0.376                 -0.938               -0.245
10     3.75                   0.0735              -0.767
# ℹ 37,490 more rows
# ℹ 73 more variables: adults <dbl>,
#   is_repeated_guest <dbl>, previous_cancellations <dbl>,
```

Lasso workflow

```r
( lasso_work = workflow() |>
    add_model(lasso_model) |>
    add_recipe(lasso_recipe)
)
```

```
══ Workflow ════════════════════════════════════════════════
Preprocessor: Recipe
Model: logistic_reg()

── Preprocessor ────────────────────────────────────────────
7 Recipe Steps

• step_date()
• step_holiday()
• step_rm()
• step_rm()
• step_dummy()
• step_zv()
• step_normalize()

── Model ───────────────────────────────────────────────────
Logistic Regression Model Specification (classification)
```

v-folds for hyperparameter tuning

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

```
#  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)
    )
)
```

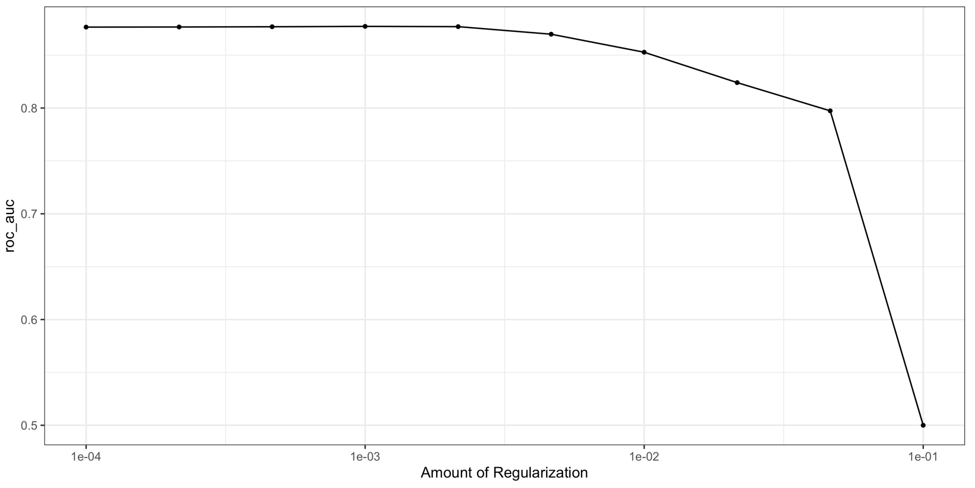
```
# Tuning results
# 5-fold cross-validation using stratification
# A tibble: 5 × 5
  splits               id    .metrics .notes   .predictions
  <list>               <chr> <list>   <list>   <list>
1 <split [30000/7500]> Fold1 <tibble> <tibble> <tibble>
2 <split [30000/7500]> Fold2 <tibble> <tibble> <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_err
      <dbl> <chr>   <chr>      <dbl> <int>   <dbl>
 1 0.0001   roc_auc binary     0.877     5 0.00318
 2 0.000215 roc_auc binary     0.877     5 0.00316
 3 0.000464 roc_auc binary     0.877     5 0.00314
 4 0.001    roc_auc binary     0.877     5 0.00304
 5 0.00215  roc_auc binary     0.877     5 0.00263
 6 0.00464  roc_auc binary     0.870     5 0.00253
 7 0.01     roc_auc binary     0.853     5 0.00249
 8 0.0215   roc_auc binary     0.824     5 0.00424
 9 0.0464   roc_auc binary     0.797     5 0.00400
10 0.1      roc_auc binary     0.5       5 0
# ℹ 1 more variable: .config <chr>
```

```
lasso_grid |>
  autoplot()
```



“Best” models

```r
lasso_grid |>
  show_best(metric = "roc_auc", n=5)
```

```
# A tibble: 5 × 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 Preproces…
2 0.00215  roc_auc binary     0.877     5 0.00263 Preproces…
3 0.000464 roc_auc binary     0.877     5 0.00314 Preproces…
4 0.000215 roc_auc binary     0.877     5 0.00316 Preproces…
5 0.0001   roc_auc binary     0.877     5 0.00318 Preproces…
```

“Best” model

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

```
# 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…
```

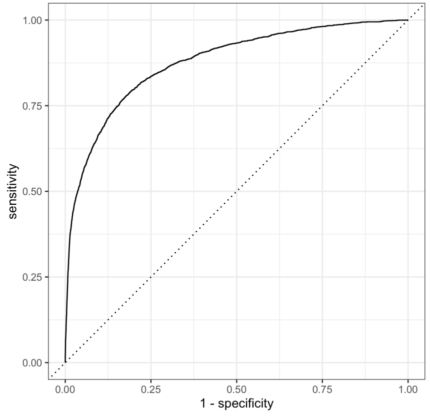
Extracting predictions

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) )
```

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

```r
lasso_train_perf |>
  roc_curve(children, .pred_children) |>
  autoplot()
```




```r
lasso_train_perf |>
  roc_auc(children, .pred_children)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.877
```

Re-fitting

Typically with a tuned model we update the workflow (or model) with the optimal parameter values and then refit using the complete training data,

```
lasso_work_tuned = finalize_workflow(
  lasso_work,
  lasso_best
)

( lasso_fit = lasso_work_tuned |>
    fit(data=hotel_train) )
```

```
══ Workflow [trained] ══════════════════════════════════
Preprocessor: Recipe
Model: logistic_reg()

── Preprocessor ────────────────────────────────────────
7 Recipe Steps

• step_date()
• step_holiday()
• step_rm()
• step_rm()
• step_dummy()
• step_zv()
• step_normalize()
```

Test Performance (out-of-sample)

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

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

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

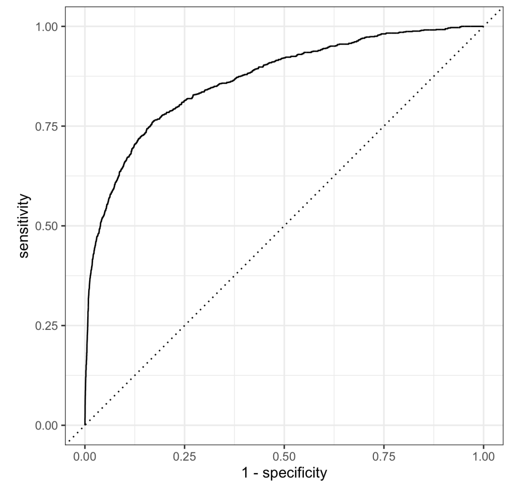
```
accuracy(lasso_test_perf, children, .pred_class)
```

```
# A tibble: 1 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.937
```

```
precision(lasso_test_perf, children, .pred_class
```

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

```
yardstick::roc_curve(lasso_test_perf, children,
  autoplot()
```

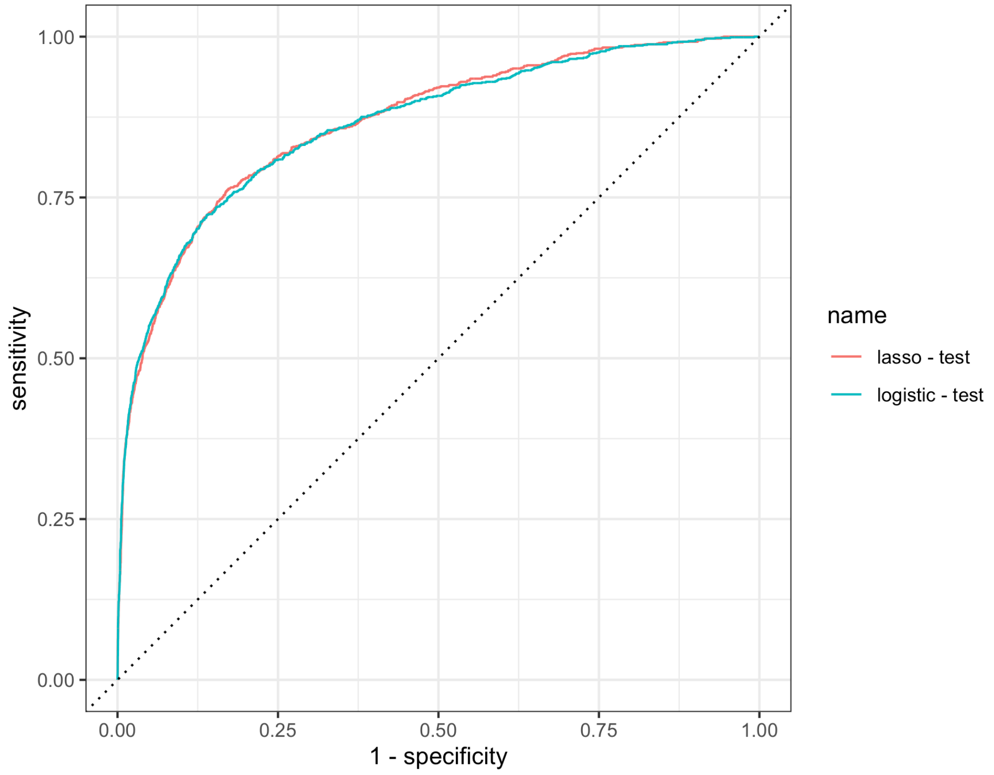



```
roc_auc(lasso_test_perf, children, .pred_childre
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.866
```

Comparing models

```
bind_rows(
  lr_test_perf |> mutate(name = "logistic - test"),
  lasso_test_perf |> mutate(name = "lasso - test")
) |>
  group_by(name) |>
  yardstick::roc_curve(children, .pred_children) |>
  autoplot()
```



Decision tree

Decision tree models

```r
show_engines("decision_tree")
```

```
# A tibble: 5 × 2
  engine mode
  <chr>  <chr>
1 rpart  classification
2 rpart  regression
3 C5.0   classification
4 spark  classification
5 spark  regression
```

```r
dt_model = decision_tree(
  tree_depth = tune(),
  min_n = tune(),
  cost_complexity = tune()
) |>
  set_engine("rpart") |>
  set_mode("classification")
```

Recipe & workflow

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

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

```
dt_work = workflow() |>
  add_model(dt_model) |>
  add_recipe(dt_recipe)
```

Tuning

```r
( dt_grid = grid_regular(
    cost_complexity(),
    tree_depth(),
    min_n(),
    levels = 3
) )
```

```
# A tibble: 27 × 3
   cost_complexity tree_depth min_n
             <dbl>      <int> <int>
 1    0.0000000001          1     2
 2    0.00000316            1     2
 3    0.1                   1     2
 4    0.0000000001          8     2
 5    0.00000316            8     2
 6    0.1                   8     2
 7    0.0000000001         15     2
 8    0.00000316           15     2
 9    0.1                  15     2
10    0.0000000001          1    21
# ℹ 17 more rows
```

```r
doFuture::registerDoFuture()
future::plan(future::multisession, workers = 8)
```

```r
dt_tune = dt_work |>
  tune_grid(
    hotel_vf,
    grid = dt_grid,
    control = control_grid(save_pred = TRUE),
    metrics = metric_set(roc_auc)
  )
```

How many decision tree models were fit?

Tuning results

```
dt_tune |>
  collect_metrics() |>
  arrange(desc(mean))
```

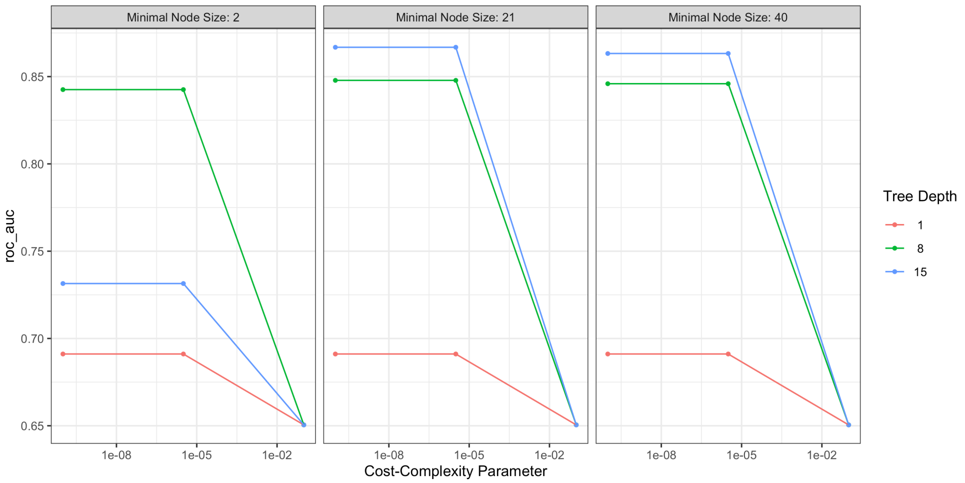
```
# A tibble: 27 × 9
   cost_complexity tree_depth min_n .metric .estimator  mean
             <dbl>      <int> <int> <chr>   <chr>      <dbl>
 1    0.0000000001         15    21 roc_auc binary     0.867
 2    0.00000316           15    21 roc_auc binary     0.867
 3    0.0000000001         15    40 roc_auc binary     0.863
 4    0.00000316           15    40 roc_auc binary     0.863
 5    0.0000000001          8    21 roc_auc binary     0.848
 6    0.00000316            8    21 roc_auc binary     0.848
 7    0.0000000001          8    40 roc_auc binary     0.846
 8    0.00000316            8    40 roc_auc binary     0.846
 9    0.0000000001          8     2 roc_auc binary     0.843
10    0.00000316            8     2 roc_auc binary     0.843
# ℹ 17 more rows
# ℹ 3 more variables: n <int>, std_err <dbl>, .config <chr>
```

“Best” parameters

```r
dt_tune |>
  show_best(metric = "roc_auc")
```

```
# A tibble: 5 × 9
  cost_complexity tree_depth min_n .metric
            <dbl>      <int> <int> <chr>
1    0.0000000001         15    21 roc_auc
2    0.00000316           15    21 roc_auc
3    0.0000000001         15    40 roc_auc
4    0.00000316           15    40 roc_auc
5    0.0000000001          8    21 roc_auc
# ℹ 5 more variables: .estimator <chr>,
#   mean <dbl>, n <int>, std_err <dbl>,
#   .config <chr>
```

```r
autoplot(dt_tune)
```



Re-fitting

```
(dt_best = dt_tune |>
  select_best(metric = "roc_auc"))
```

```
# A tibble: 1 × 4
  cost_complexity tree_depth min_n .config
            <dbl>      <int> <int> <chr>
1    0.0000000001         15    21 Preprocessor1_Model16
```

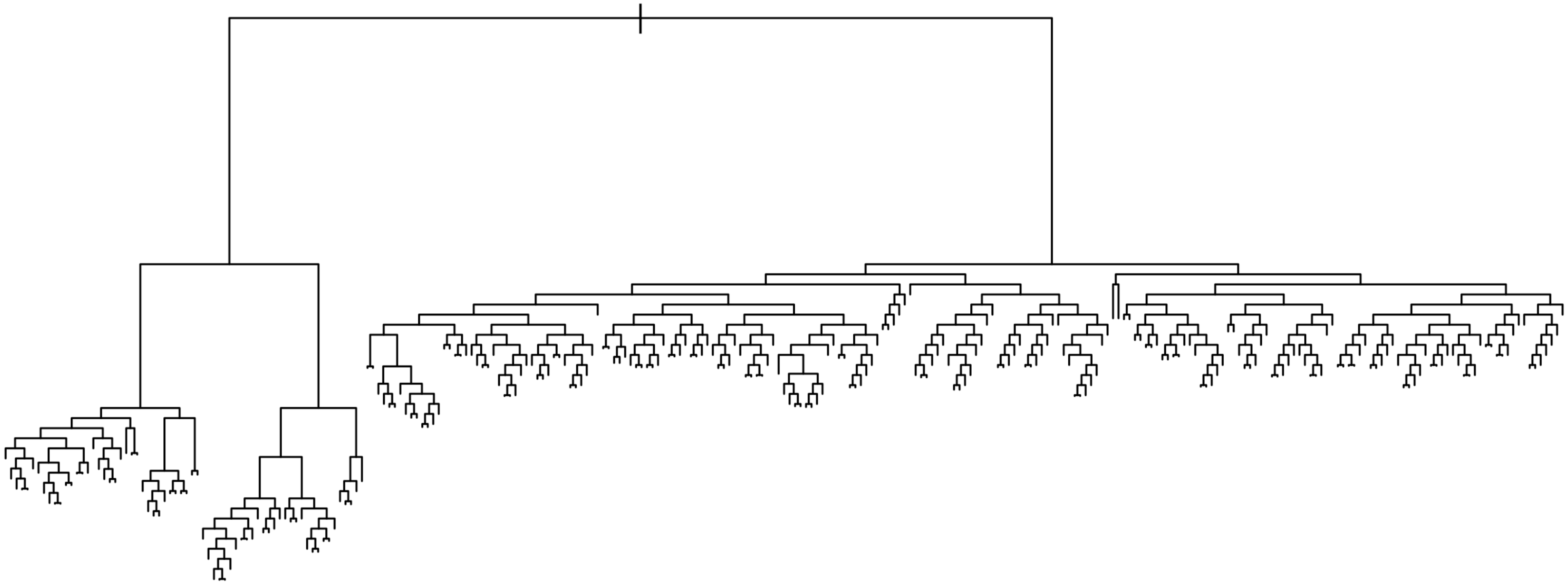
```
dt_work_tuned = finalize_workflow(
  dt_work,
  dt_best
)

( dt_fit = dt_work_tuned |>
    fit(data=hotel_train))
```

```
══ Workflow [trained] ══════════════════════════════════════
Preprocessor: Recipe
Model: decision_tree()

── Preprocessor ────────────────────────────────────────────
4 Recipe Steps

• step_date()
• step_holiday()
• step_rm()
• step_rm()
```

```
── Model ──────────────────────────────────────────────
n= 37500

node), split, n, loss, yval, (yprob)
```

Model extraction

```r
dt_fit |>
  hardhat::extract_fit_engine() |>
  plot()
```



Test Performance (out-of-sample)

```r
dt_test_perf = dt_fit |>
  augment(new_data = hotel_test) |>
  select(children, starts_with(".pred"))
```

```r
conf_mat(dt_test_perf, children, .pred_class)
```

```
          Truth
Prediction children  none
  children      444   270
  none          567 11219
```

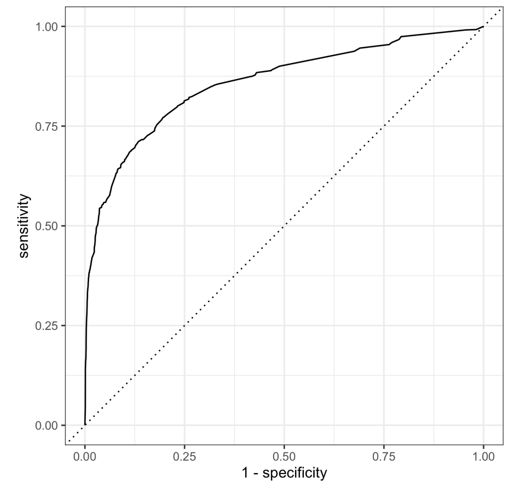
```r
accuracy(dt_test_perf, children, .pred_class)
```

```
# A tibble: 1 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.933
```

```r
precision(dt_test_perf, children, .pred_class)
```

```
# A tibble: 1 × 3
  .metric   .estimator .estimate
  <chr>     <chr>          <dbl>
1 precision binary         0.622
```

```r
yardstick::roc_curve(dt_test_perf, children, .pr
  autoplot()
```

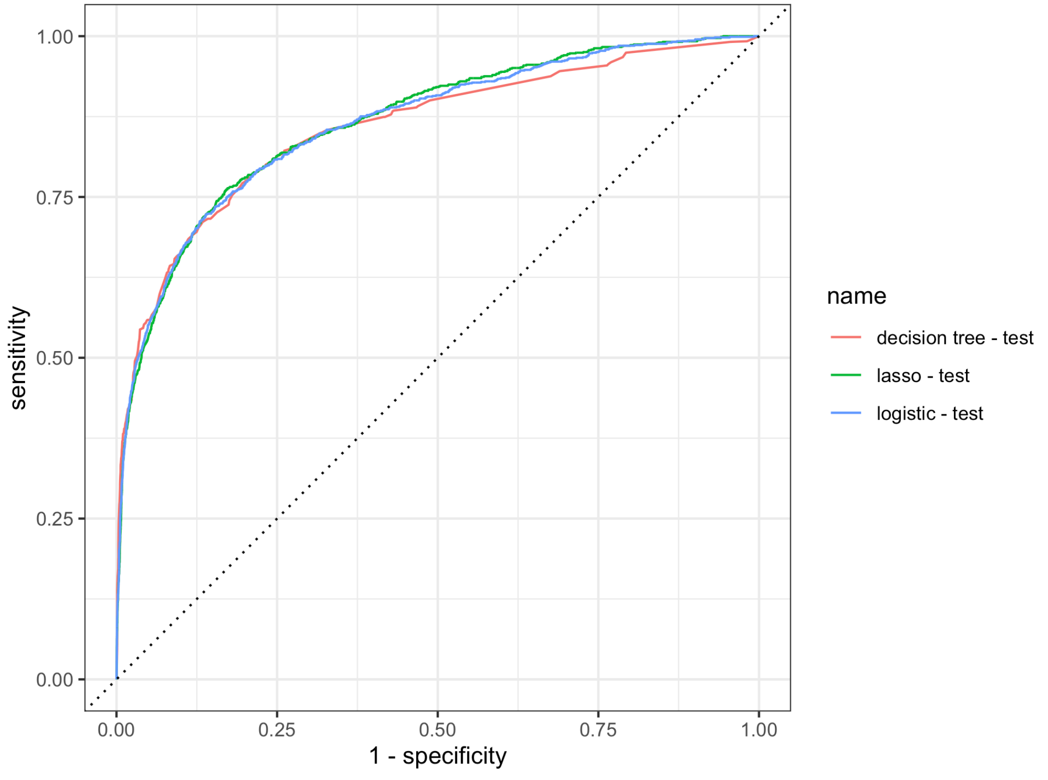



```r
roc_auc(dt_test_perf, children, .pred_children)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.858
```

Comparing models

```
bind_rows(
  lr_test_perf |> mutate(name = "logistic - test"),
  lasso_test_perf |> mutate(name = "lasso - test"),
  dt_test_perf |> mutate(name = "decision tree - test")
) |>
  group_by(name) |>
  yardstick::roc_curve(children, .pred_children) |>
  autoplot()
```



Random Forest

Random forest models

```r
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
```

```r
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 - automatic grid search

```r
rf_tune = rf_work |>
  tune_grid(
    hotel_vf,
    grid = 10,
    control = control_grid(save_pred
    metrics = metric_set(roc_auc)
  )
```

```r
rf_tune |>
  collect_metrics() |>
  arrange(desc(mean))
```

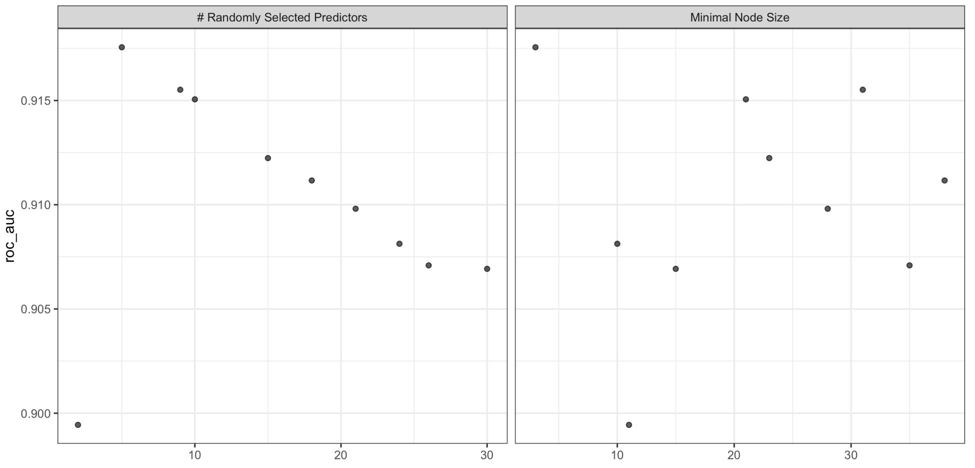
```
# A tibble: 10 × 8
    mtry min_n .metric .estimator  mean     n std_err
   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl>
 1     5     3 roc_auc binary     0.918     5 0.00195
 2     9    31 roc_auc binary     0.916     5 0.00181
 3    10    21 roc_auc binary     0.915     5 0.00183
 4    15    23 roc_auc binary     0.912     5 0.00177
 5    18    38 roc_auc binary     0.911     5 0.00251
 6    21    28 roc_auc binary     0.910     5 0.00198
 7    24    10 roc_auc binary     0.908     5 0.00289
 8    26    35 roc_auc binary     0.907     5 0.00192
 9    30    15 roc_auc binary     0.907     5 0.00237
10     2    11 roc_auc binary     0.899     5 0.00259
# ℹ 1 more variable: .config <chr>
```

"Best" parameters

```r
rf_tune |>
  show_best(metric = "roc_auc")
```

```
# A tibble: 5 × 8
   mtry min_n .metric .estimator  mean     n
  <int> <int> <chr>   <chr>      <dbl> <int>
1     5     3 roc_auc binary     0.918     5
2     9    31 roc_auc binary     0.916     5
3    10    21 roc_auc binary     0.915     5
4    15    23 roc_auc binary     0.912     5
5    18    38 roc_auc binary     0.911     5
# ℹ 2 more variables: std_err <dbl>,
#   .config <chr>
```

```r
autoplot(rf_tune)
```



Re-fitting

```
rf_best = rf_tune |>
  select_best(metric = "roc_auc")
```

```
rf_work_tuned = finalize_workflow(
  rf_work,
  rf_best
)

( rf_fit = rf_work_tuned |>
    fit(data=hotel_train) )
```

```
══ Workflow [trained] ══════════════════════════════════════
Preprocessor: Recipe
Model: rand_forest()

── Preprocessor ────────────────────────────────────────────
4 Recipe Steps

• step_date()
• step_holiday()
• step_rm()
• step_rm()

── Model ───────────────────────────────────────────────────
Ranger result

Call:
```

Test Performance (out-of-sample)

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

```r
conf_mat(rf_test_perf, children, .pred_class)
```

```
          Truth
Prediction children  none
  children      388    70
  none          623 11419
```

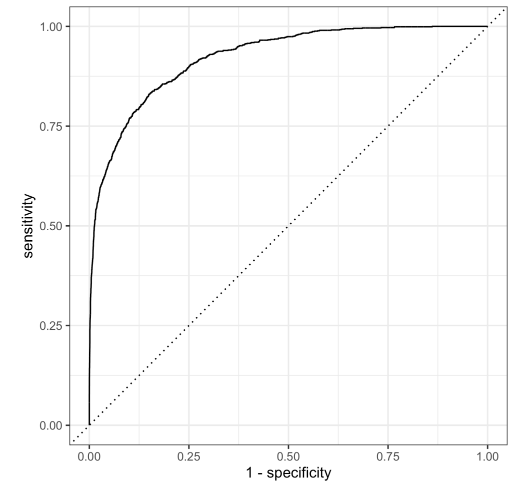
```r
accuracy(rf_test_perf, children, .pred_class)
```

```
# A tibble: 1 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.945
```

```r
precision(rf_test_perf, children, .pred_class)
```

```
# A tibble: 1 × 3
  .metric   .estimator .estimate
  <chr>     <chr>          <dbl>
1 precision binary         0.847
```

```r
yardstick::roc_curve(rf_test_perf, children, .pr
  autoplot()
```

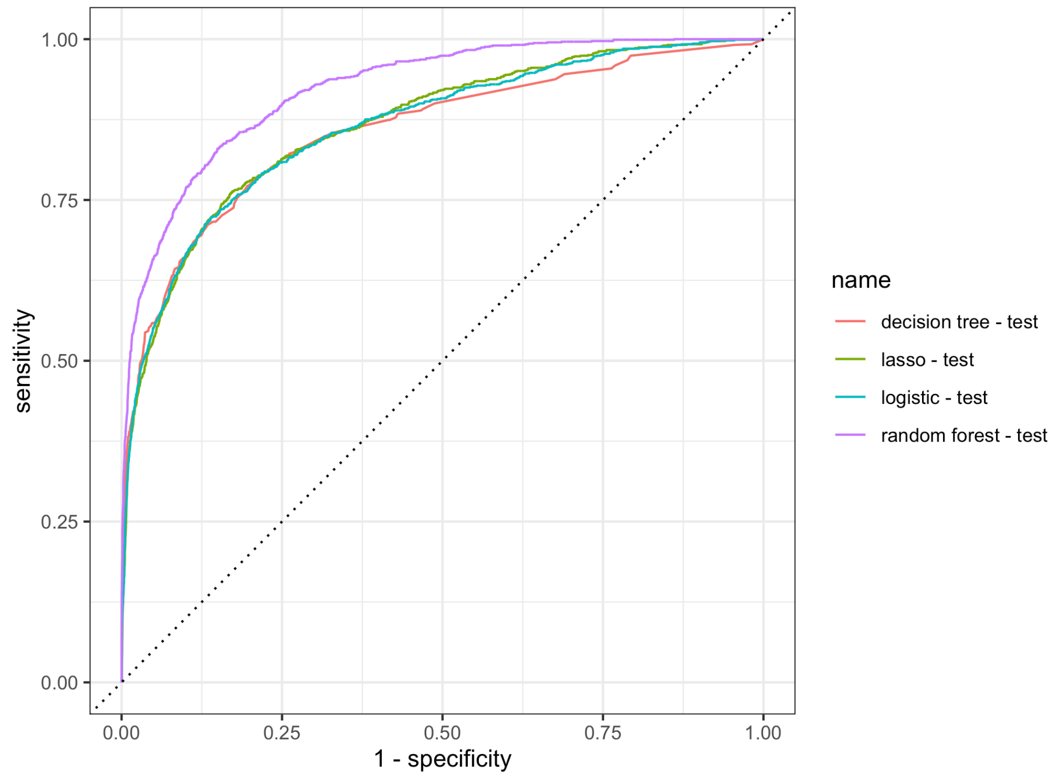



```r
roc_auc(rf_test_perf, children, .pred_children)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.923
```

Comparing models

```r
bind_rows(
  lr_test_perf |> mutate(name = "logistic - test"),
  lasso_test_perf |> mutate(name = "lasso - test"),
  dt_test_perf |> mutate(name = "decision tree - test"),
  rf_test_perf |> mutate(name = "random forest - test")
) |>
  group_by(name) |>
  yardstick::roc_curve(children, .pred_children) |>
  autoplot()
```


sensitivity
1.00
0.75
0.50
0.25
0.00
1 - specificity
0.00
0.25
0.50
0.75
1.00
name
decision tree - test
lasso - test
logistic - test
random forest - test