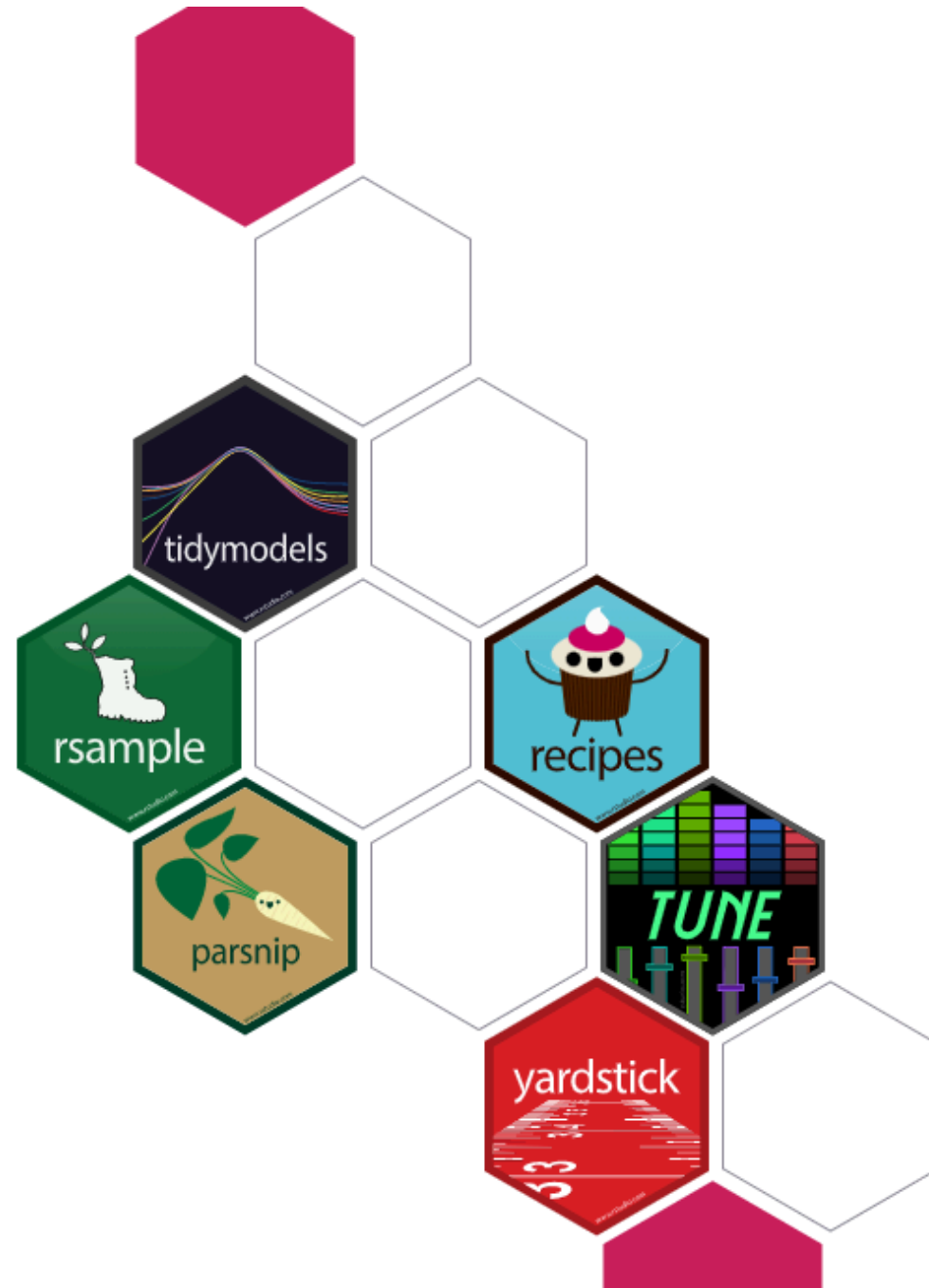


More Tidymodels

Lecture 23

Dr. Colin Rundel



Hotels Data

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

```
1 hotels = read_csv(  
2   'https://tidymodels.org/start/case-study/hotels.csv'  
3 ) |>  
4   mutate(  
5     across(where(is.character), as.factor)  
6   )
```

The data

```
1 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.

```
1 hotels |>
2   count(children) |>
3   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

```
1 set.seed(123)
2
3 splits = initial_split(
4   hotels, strata = children
5 )
6
7 hotel_train = training(splits)
8 hotel_test = testing(splits)
```

```
1 dim(hotel_train)
```

```
[1] 37500    23
```

```
1 dim(hotel_test)
```

```
[1] 12500    23
```

```
1 hotel_train |>
2   count(children) |>
3   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
```

```
1 hotel_test |>
2   count(children) |>
3   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

```
1 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
```

```
1 lr_model = logistic_reg() |>
2   set_engine("glm")
```

```
1 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

```
1 holidays = c("AllSouls", "AshWednesday", "ChristmasEve", "Easter",  
2             "ChristmasDay", "GoodFriday", "NewYearsDay", "PalmSunday")  
3  
4 lr_recipe = recipe(children ~ ., data = hotel_train) |>  
5   step_date(arrival_date) |>  
6   step_holiday(arrival_date, holidays = holidays) |>  
7   step_rm(arrival_date) |>  
8   step_rm(country) |>  
9   step_dummy(all_nominal_predictors()) |>  
10  step_zv(all_predictors())  
11  
12 lr_recipe
```



```

1 lr_recipe |>
2   prep() |>
3   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

```
1 ( lr_work = workflow() |>
2   add_model(lr_model) |>
3   add_recipe(lr_recipe)
4 )
```

== 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

```
1 ( lr_fit = lr_work |>
2   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 = ..v ~ .., family = stats::binomial, data = data)

Tidy

```
1 lr_fit |>
2   broom::tidy()
```

```
# A tibble: 76 × 5
```

	term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <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

```
1 ( lr_train_perf = lr_fit |>
2   augment(new_data = hotel_train) |>
3   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

```
1 ( lr_test_perf = lr_fit |>
2   augment(new_data = hotel_test) |>
3   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)

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

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

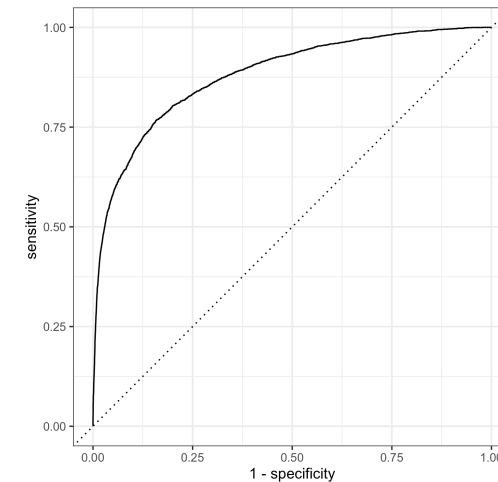
```
1 accuracy(lr_train_perf, children, .pred_class)
```

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

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

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

```
1 yardstick::roc_curve(lr_train_perf, children, .pred_class)
2 autoplot()
```



```
1 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)

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

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

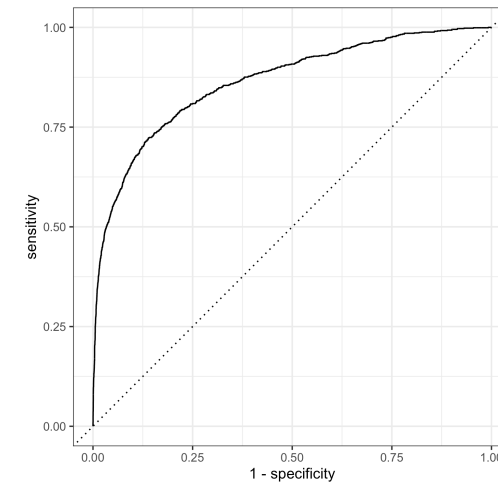
```
1 accuracy(lr_test_perf, children, .pred_class)
```

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

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

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

```
1 yardstick::roc_curve(lr_test_perf, children, .pred_class)
2 autoplot()
```

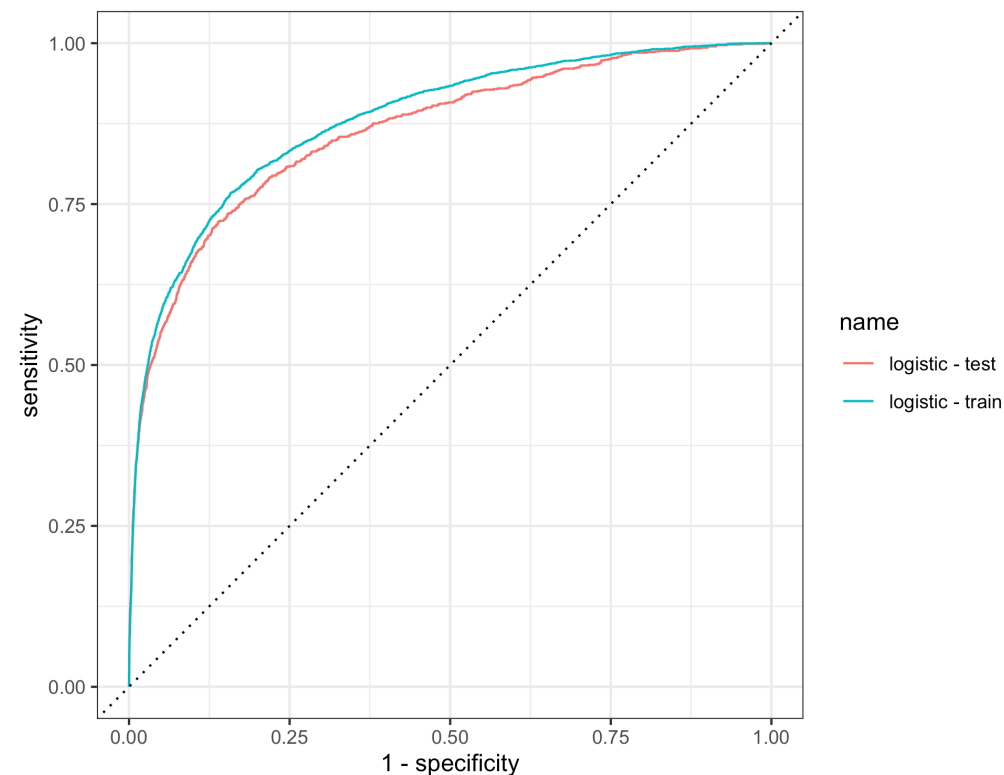


```
1 roc_auc(lr_test_perf, children, .pred_children)
```

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

Combining ROC curves

```
1 bind_rows(  
2   lr_train_perf |> mutate(name = "logistic - train"),  
3   lr_test_perf  |> mutate(name = "logistic - test")  
4 ) |>  
5 group_by(name) |>  
6 yardstick::roc_curve(children, .pred_children) |>  
7 autoplot()
```



Lasso

Lasso Model

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

$$\min_{\beta_0, \beta} \frac{1}{N} \sum_{i=1}^N w_i l(y_i, \beta_0 + \beta^T x_i) + \lambda \left[(1 - \alpha) \|\beta\|_2^2 / 2 + \alpha \|\beta\|_1 \right],$$

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

- `mixture` (α) determines the type of model fit
 - `1` for Lasso,
 - `0` for Ridge,
 - other for elastic net.
- `penalty` (λ) is the penalty term for coefficient size.

```
1 lasso_model |>
2   hardhat::extract_parameter_set_dials()
```

Collection of 1 parameters for tuning

identifier	type	object
penalty	penalty	nparam[+]

```
1 lasso_model |>
2   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.

```
1 lasso_recipe = lr_recipe |>
2   step_normalize(all_predictors())
```

```
1 lasso_recipe |>
2   prep() |>
3   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

```
1 ( lasso_work = workflow() |>
2   add_model(lasso_model) |>
3   add_recipe(lasso_recipe)
4 )
```

== 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

```
1 ( 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

```
1 ( lasso_grid = lasso_work |>
2   tune_grid(
3     hotel_vf,
4     grid = tibble(
5       penalty = 10^seq(-4, -1, length.out = 10)
6     ),
7     control = control_grid(save_pred = TRUE),
8     metrics = metric_set(roc_auc)
9   )
10 )
```

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

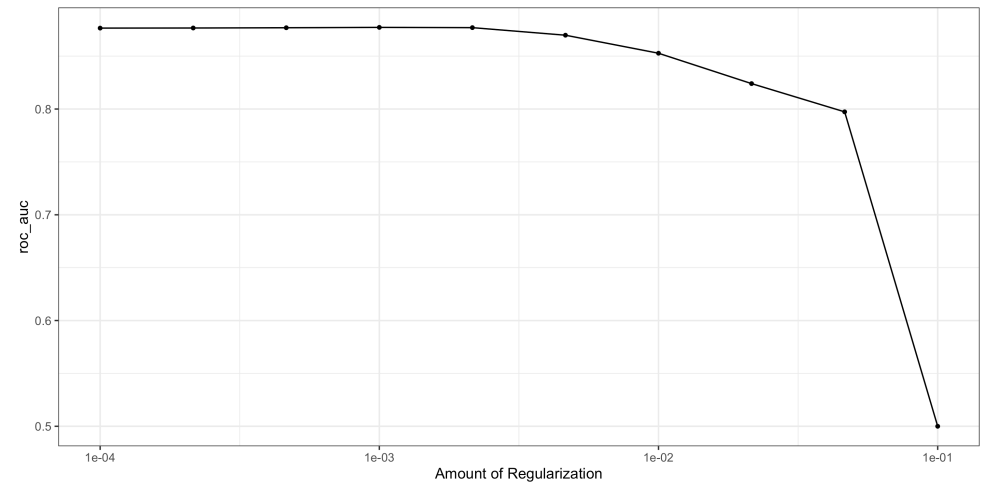
```
1 lasso_grid |>
2   collect_metrics()
```

```
# 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>
```

```
1 lasso_grid |>
2   autoplot()
```



“Best” models

```
1 lasso_grid |>
2   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

```
1 ( lasso_best = lasso_grid |>
2   collect_metrics() |>
3   mutate(mean = round(mean, 2)) |>
4   arrange(desc(mean), desc(penalty)) |>
5   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:

```
1 ( lasso_train_perf = lasso_grid |>
2   collect_predictions(parameters = lasso_best) )
```

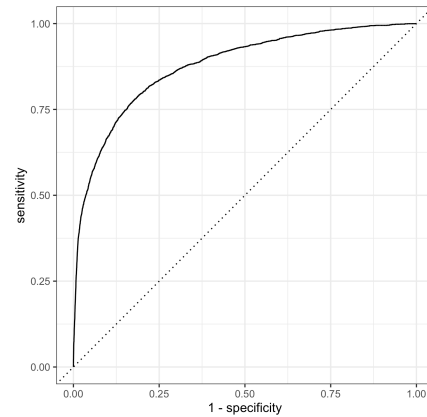
```
# A tibble: 37,500 × 7
```

	.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>
```

```
1 lasso_train_perf |>
2   roc_curve(children, .pred_children) |>
3   autoplot()
```



```
1 lasso_train_perf |>
2   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,

```
1 lasso_work_tuned = finalize_workflow(  
2   lasso_work,  
3   lasso_best  
4 )  
5  
6 ( lasso_fit = lasso_work_tuned |>  
7   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)

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

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

	Truth	
Prediction	children	none
children	330	109
none	681	11380

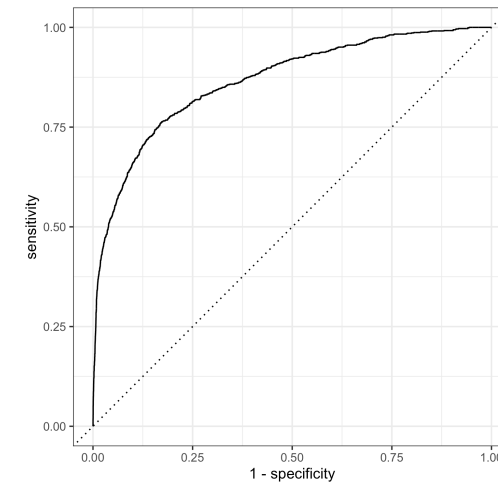
```
1 accuracy(lasso_test_perf, children, .pred_class)
```

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

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

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

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

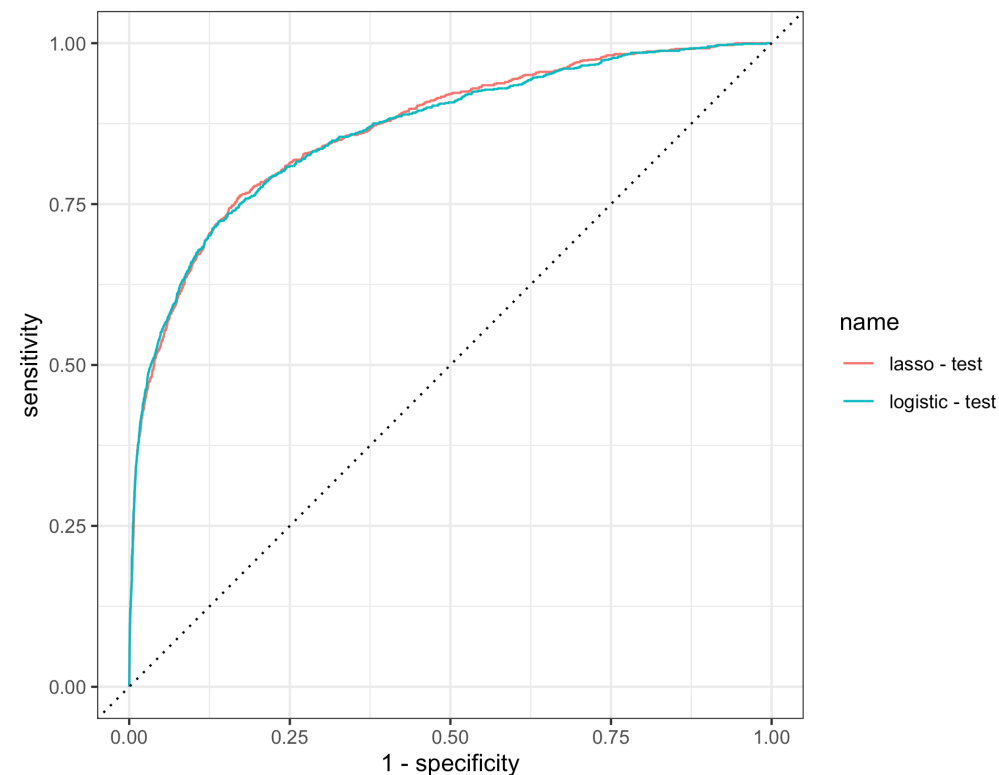


```
1 roc_auc(lasso_test_perf, children, .pred_childre
```

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

Comparing models

```
1 bind_rows(  
2   lr_test_perf |> mutate(name = "logistic - test"),  
3   lasso_test_perf |> mutate(name = "lasso - test")  
4 ) |>  
5 group_by(name) |>  
6 yardstick::roc_curve(children, .pred_children) |>  
7 autoplot()
```



Decision tree

Decision tree models

```
1 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
```

```
1 dt_model = decision_tree(  
2   tree_depth = tune(),  
3   min_n = tune(),  
4   cost_complexity = tune()  
5 ) |>  
6   set_engine("rpart") |>  
7   set_mode("classification")
```

Recipe & workflow

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

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

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

Tuning

```
1 ( dt_grid = grid_regular(  
2   cost_complexity(),  
3   tree_depth(),  
4   min_n(),  
5   levels = 3  
6 ) )
```

```
# 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
```

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

```
1 dt_tune = dt_work |>  
2   tune_grid(  
3     hotel_vf,  
4     grid = dt_grid,  
5     control = control_grid(save_pred = TRUE),  
6     metrics = metric_set(roc_auc)  
7   )
```

How many decision tree models were fit?

Tuning results

```
1 dt_tune |>
2   collect_metrics() |>
3   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

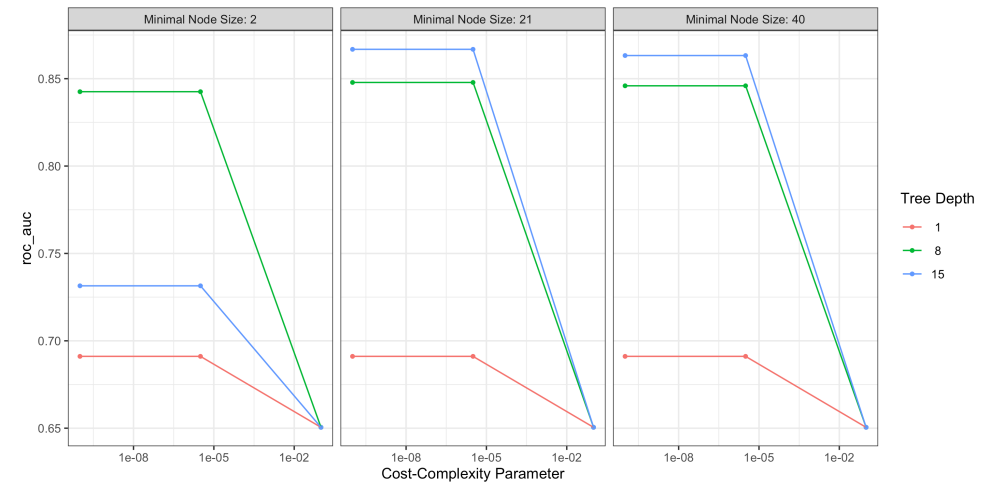
i 3 more variables: n <int>, std_err <dbl>, .config <chr>

“Best” parameters

```
1 dt_tune |>  
2   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>
```

```
1 autoplot(dt_tune)
```



Re-fitting

```
1 (dt_best = dt_tune |>
2   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
```

```
1 dt_work_tuned = finalize_workflow(
2   dt_work,
3   dt_best
4 )
5
6 ( dt_fit = dt_work_tuned |>
7   fit(data=hotel_train))
```

== Workflow [trained] ==

Preprocessor: Recipe

Model: decision_tree()

— Preprocessor —

4 Recipe Steps

- step_date()
- step_holiday()
- step_rm()
- step_rm()

— Model —————

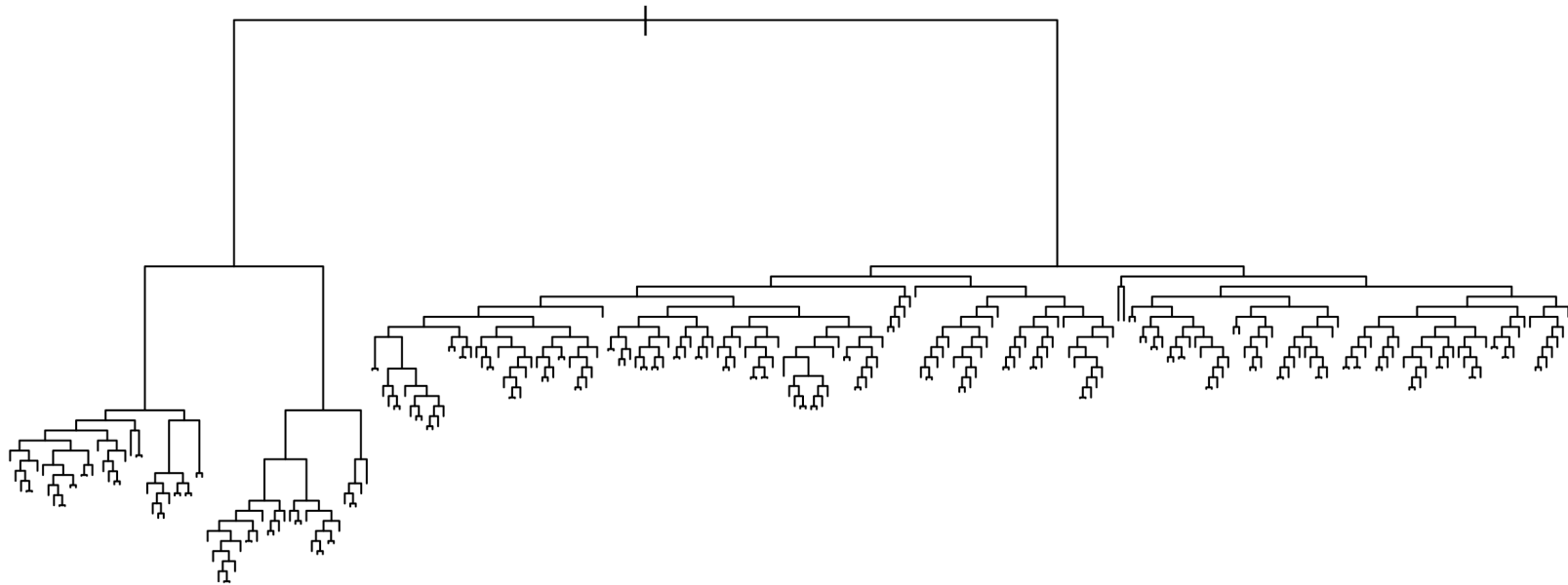
n= 37500

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

· · · · ·

Model extraction

```
1 dt_fit |>  
2   hardhat::extract_fit_engine() |>  
3   plot()
```



Test Performance (out-of-sample)

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

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

	Truth	
Prediction	children	none
children	444	270
none	567	11219

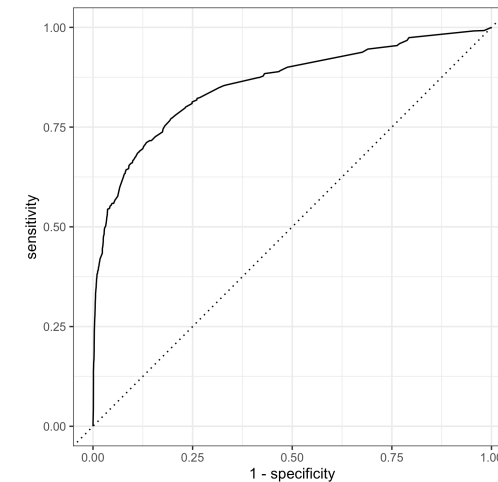
```
1 accuracy(dt_test_perf, children, .pred_class)
```

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

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

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

```
1 yardstick::roc_curve(dt_test_perf, children, .pred_class)
2 autoplot()
```

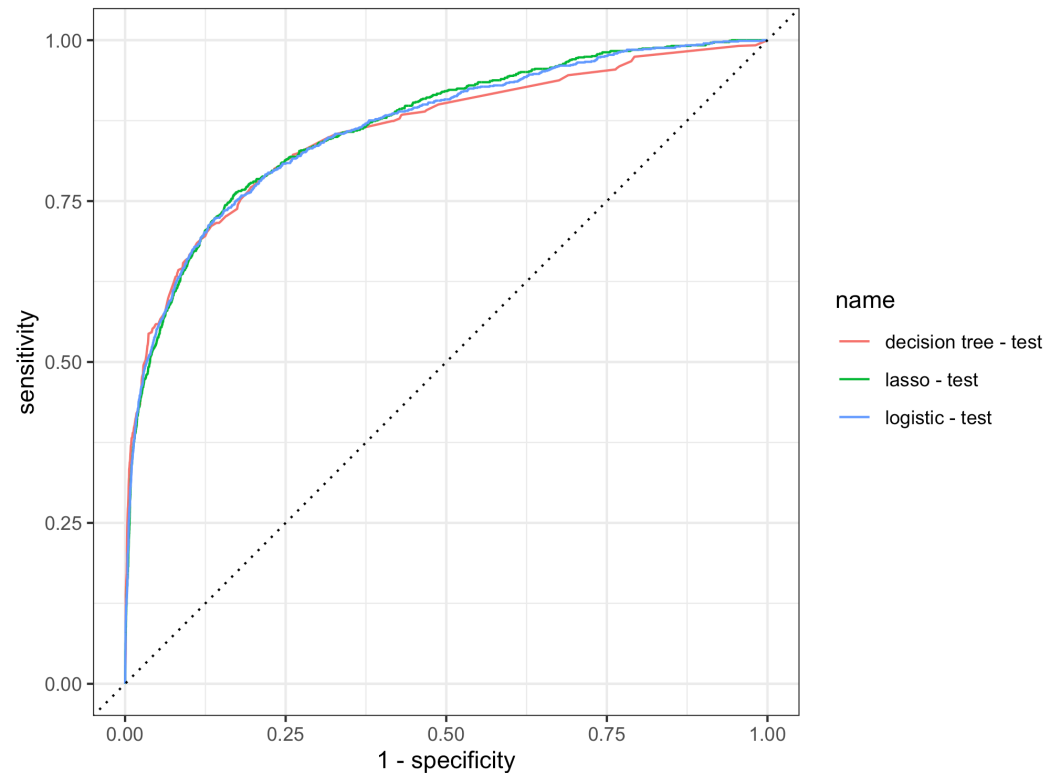


```
1 roc_auc(dt_test_perf, children, .pred_children)
```

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

Comparing models

```
1 bind_rows(  
2   lr_test_perf |> mutate(name = "logistic - test"),  
3   lasso_test_perf |> mutate(name = "lasso - test"),  
4   dt_test_perf |> mutate(name = "decision tree - test")  
5 ) |>  
6   group_by(name) |>  
7   yardstick::roc_curve(children, .pred_children) |>  
8   autoplot()
```



Random Forest

Random forest models

```
1 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

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

Recipe & workflow

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

```
1 rf_recipe = recipe(children ~ ., data = hotel_train) |>
2   step_date(arrival_date) |>
3   step_holiday(arrival_date, holidays = holidays) |>
4   step_rm(arrival_date) |>
5   step_rm(country)
```

```
1 rf_work = workflow() |>
2   add_model(rf_model) |>
3   add_recipe(rf_recipe)
```

Tuning - automatic grid search

```
1 rf_tune = rf_work |>
2   tune_grid(
3     hotel_vf,
4     grid = 10,
5     control = control_grid(save_pred
6     metrics = metric_set(roc_auc)
7   )
```

```
1 rf_tune |>
2   collect_metrics() |>
3   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

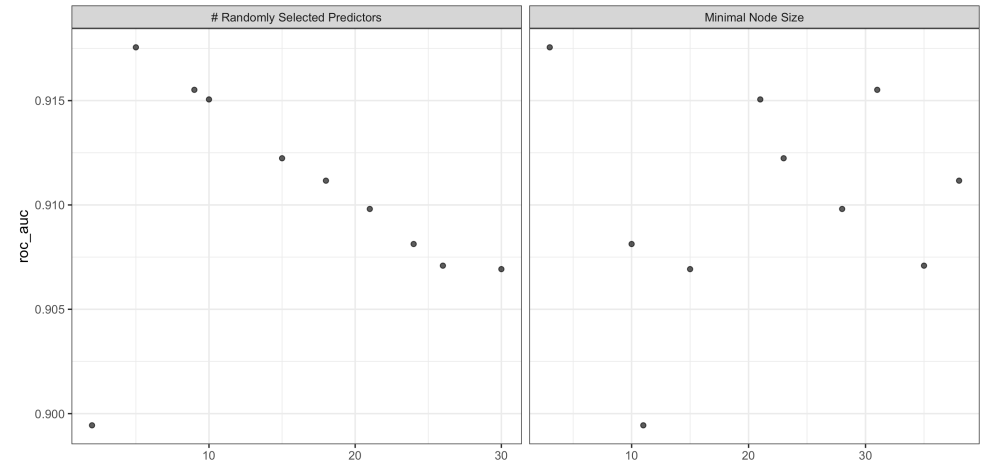
# i 1 more variable: .config <chr>
```

“Best” parameters

```
1 rf_tune |>  
2   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>
```

```
1 autoplot(rf_tune)
```



Re-fitting

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

```
1 rf_work_tuned = finalize_workflow(  
2   rf_work,  
3   rf_best  
4 )  
5  
6 ( rf_fit = rf_work_tuned |>  
7   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)

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

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

	Truth	
Prediction	children	none
children	388	70
none	623	11419

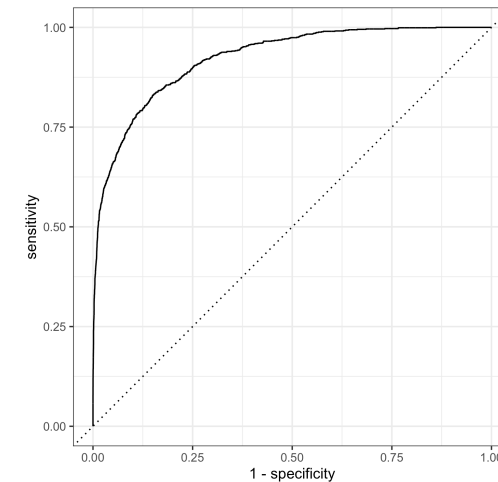
```
1 accuracy(rf_test_perf, children, .pred_class)
```

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

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

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

```
1 yardstick::roc_curve(rf_test_perf, children, .pred_class)
2 autoplot()
```



```
1 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

```
1 bind_rows(  
2   lr_test_perf |> mutate(name = "logistic - test"),  
3   lasso_test_perf |> mutate(name = "lasso - test"),  
4   dt_test_perf |> mutate(name = "decision tree - test"),  
5   rf_test_perf |> mutate(name = "random forest - test")  
6 ) |>  
7   group_by(name) |>  
8   yardstick::roc_curve(children, .pred_children) |>  
9   autoplot()
```

