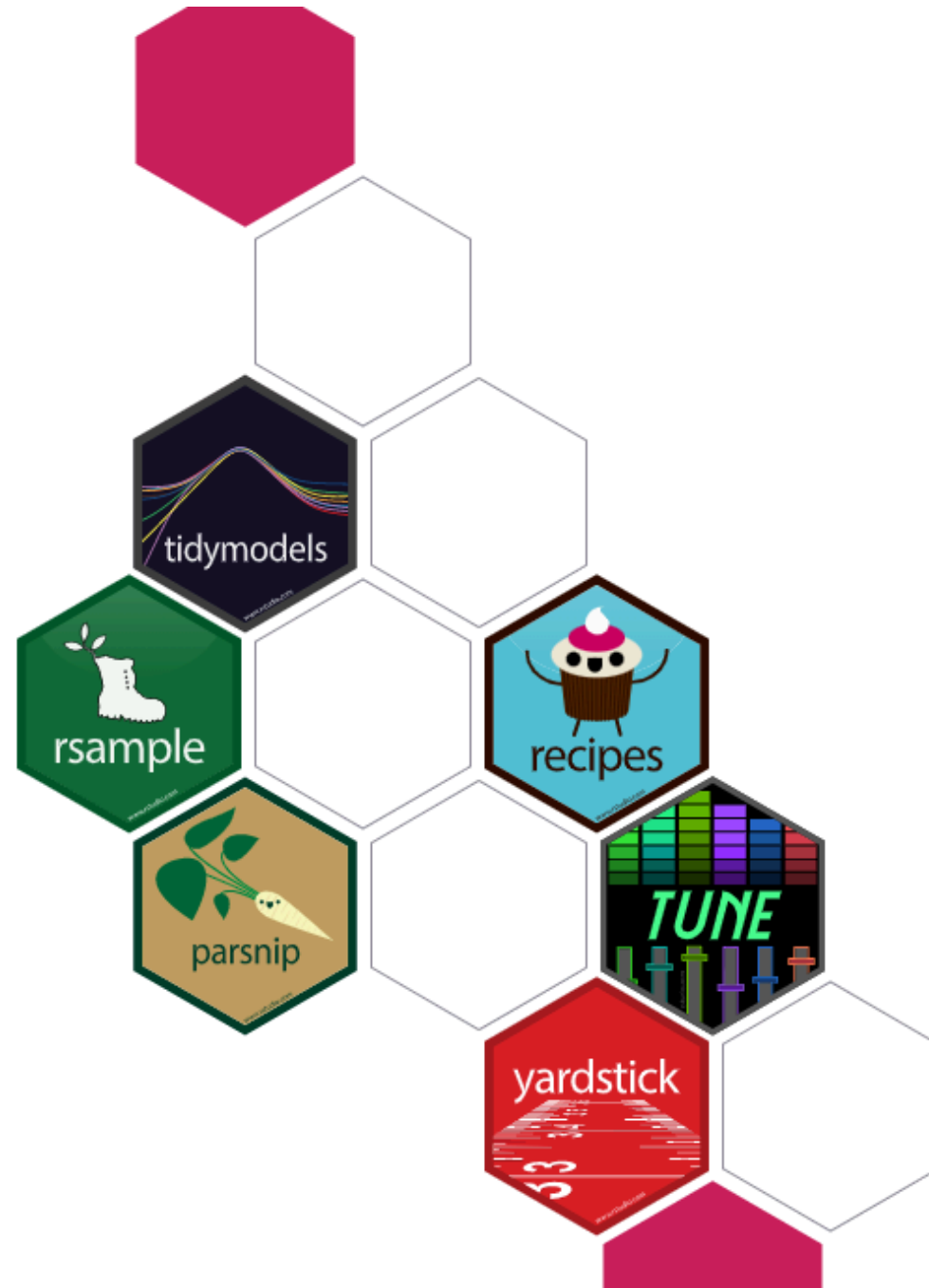


More Tidymodels

Lecture 24

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, 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/T0, Direct, Online_TA, Online_TA, Direct, 0...
$ distribution_channel <fct> TA/T0, Direct, TA/T0, TA/T0, Direct, TA/T0, 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, ...
$ booking_changes <dbl> 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, ...
$ deposit_type <fct> No_Deposit, No_Deposit, No_Deposit, No_Deposit, No_Dep...
$ days_in_waiting_list <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 236, 0, 0, 0...
$ customer_type <fct> Transient-Party, Transient, Transient, Transient, Tran...
$ average_daily_rate <dbl> 80.75, 170.00, 8.00, 81.00, 157.60, 49.09, 289.00, 82...
$ required_car_parking_spaces <fct> none, none, none, none, none, none, none, none, none, none
```

Factors

```
1 map_int(hotels, ~ length(levels(.x))) |> keep(~ .x > 0) |> sort()
```

hotel	children	required_car_parking_spaces
2	2	2
deposit_type	customer_type	meal
3	4	5
distribution_channel	market_segment	reserved_room_type
5	7	9
assigned_room_type	country	
10	155	

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 hotel_split = initial_split(
4   hotels, strata = children
5 )
6
7 hotel_train = training(hotel_split)
8 hotel_test = testing(hotel_split)
```

```
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 lr_recipe = recipe(children ~ ., data = hotel_train) |>
2   step_date(arrival_date, features = c("dow", "month"), label=TRUE) |>
3   step_mutate(
4     season = case_when(
5       arrival_date_month %in% c(12, 1, 2) ~ "Winter",
6       arrival_date_month %in% c(3, 4, 5) ~ "Spring",
7       arrival_date_month %in% c(6, 7, 8) ~ "Summer",
8       arrival_date_month %in% c(9, 10, 11) ~ "Fall"
9     ) |>
10    factor(levels = c("Winter", "Spring", "Summer", "Fall"))
11  ) |>
12  step_rm(arrival_date, arrival_date_month) |>
13  step_rm(country) |>
14  step_unknown(season) |>
15  step_dummy(all_nominal_predictors()) |>
16  step_zv(all_predictors())
```

```

1 lr_recipe |>
2   prep() |>
3   bake(new_data = hotel_train)

```

A tibble: 37,500 × 56

	lead_time <dbl>	stays_in_weekend_nights <dbl>	stays_in_week_nights <dbl>	adults <dbl>	is_repeated_guest <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 51 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>, hotel_Resort_Hotel <dbl>, meal_FB <dbl>,
meal_HB <dbl>, meal_SC <dbl>, meal_Undefined <dbl>, market_segment_Complementary <dbl>,
market_segment_Corporate <dbl>, market_segment_Direct <dbl>, market_segment_Groups <dbl>,
market_segment_Offline_TA.TO <dbl>, market_segment_Online_TA <dbl>, ...

Workflow

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

== Workflow ==

Preprocessor: Recipe

Model: logistic_reg()

— Preprocessor —

7 Recipe Steps

- step_date()
- step_mutate()
- step_rm()
- step_rm()
- step_unknown()
- step_dummy()
- step_zv()

— Model —

Logistic Regression Model Specification (classification)

Computational engine: glm

Fit

```
1 ( lr_fit = lr_work |>  
2   fit(data = hotel_train) )
```

== Workflow [trained] ==

Preprocessor: Recipe

Model: logistic_reg()

— Preprocessor —

7 Recipe Steps

- step_date()
- step_mutate()
- step_rm()
- step_rm()
- step_unknown()
- step_dummy()
- step_zv()

— Model —

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

Coefficients:

(Intercept)
17.121622
lead time

Tidy

```
1 lr_fit |>
2   broom::tidy() |>
3   arrange(p.value) |>
4   print(n=100)
```

A tibble: 56 × 5

	term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
1	average_daily_rate	-1.04e-2	5.14e-4	-20.3	1.40e-91
2	total_of_special_r...	-4.73e-1	2.64e-2	-17.9	9.93e-72
3	assigned_room_type...	-1.19e+0	7.79e-2	-15.3	9.80e-53
4	reserved_room_type...	1.22e+0	8.91e-2	13.7	1.34e-42
5	reserved_room_type...	-2.57e+0	1.92e-1	-13.4	1.05e-40
6	hotel_Resort_Hotel	8.28e-1	6.21e-2	13.3	1.65e-40
7	adults	6.46e-1	4.97e-2	13.0	1.35e-38
8	assigned_room_type...	-1.77e+0	1.45e-1	-12.2	2.63e-34
9	meal_SC	1.28e+0	1.26e-1	10.2	2.78e-24
10	reserved_room_type...	-2.25e+0	2.27e-1	-9.92	3.45e-23
11	booking_changes	-2.29e-1	2.54e-2	-9.03	1.66e-19
12	reserved_room_type...	-1.53e+0	1.79e-1	-8.53	1.42e-17
13	reserved_room_type...	-1.55e+0	1.95e-1	-7.96	1.70e-15
14	reserved_room_type...	-3.38e+0	4.44e-1	-7.62	2.63e-14
15	assigned_room_type...	-2.13e+0	3.01e-1	-7.09	1.30e-12
16	assigned_room_type...	-1.01e+0	1.43e-1	-7.09	1.30e-12
17	assigned_room_type...	-1.06e+0	1.75e-1	-6.08	1.24e- 9
18	assigned_room_type...	-1.20e+0	2.16e-1	-5.54	3.06e- 8
19	lead_time	-1.15e-3	3.07e-4	-3.76	1.73e- 4
20	previous_bookings	3.50e-1	1.07e-1	3.26	1.13e- 3

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 <fct>	.pred_class <fct>	.pred_children <dbl>	.pred_none <dbl>
1	none	none	0.0709	0.929
2	none	none	0.0332	0.967
3	none	none	0.00982	0.990
4	children	children	0.920	0.0798
5	children	none	0.441	0.559
6	children	none	0.115	0.885
7	none	none	0.0656	0.934
8	none	none	0.0931	0.907
9	none	none	0.0220	0.978
10	none	none	0.0539	0.946

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 <fct>	.pred_class <fct>	.pred_children <dbl>	.pred_none <dbl>
1	none	none	0.0194	0.981
2	none	none	0.0221	0.978
3	none	children	0.723	0.277
4	none	none	0.0721	0.928
5	none	none	0.000687	0.999
6	none	none	0.000665	0.999
7	none	none	0.111	0.889
8	none	none	0.0464	0.954
9	none	none	0.0652	0.935
10	none	none	0.103	0.897

i 12,490 more rows

Performance metrics (within-sample)

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

	Truth	
Prediction	children	none
children	1059	418
none	1968	34055

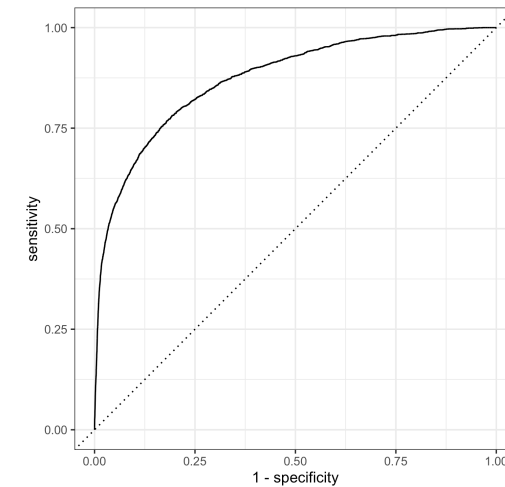
```
1 accuracy(lr_train_perf, children, .pred_class)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>       <dbl>
1 accuracy binary      0.936
```

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

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>       <dbl>
1 precision binary      0.717
```

```
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      0.874
```

Performance metrics (out-of-sample)

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

	Truth	
Prediction	children	none
children	341	134
none	670	11355

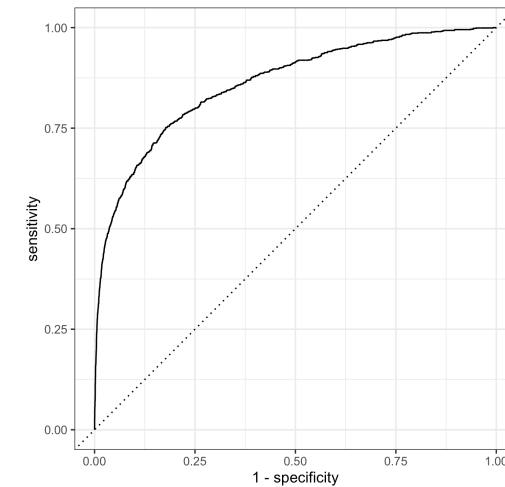
```
1 accuracy(lr_test_perf, children, .pred_class)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>       <dbl>
1 accuracy binary      0.936
```

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

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>       <dbl>
1 precision binary      0.718
```

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

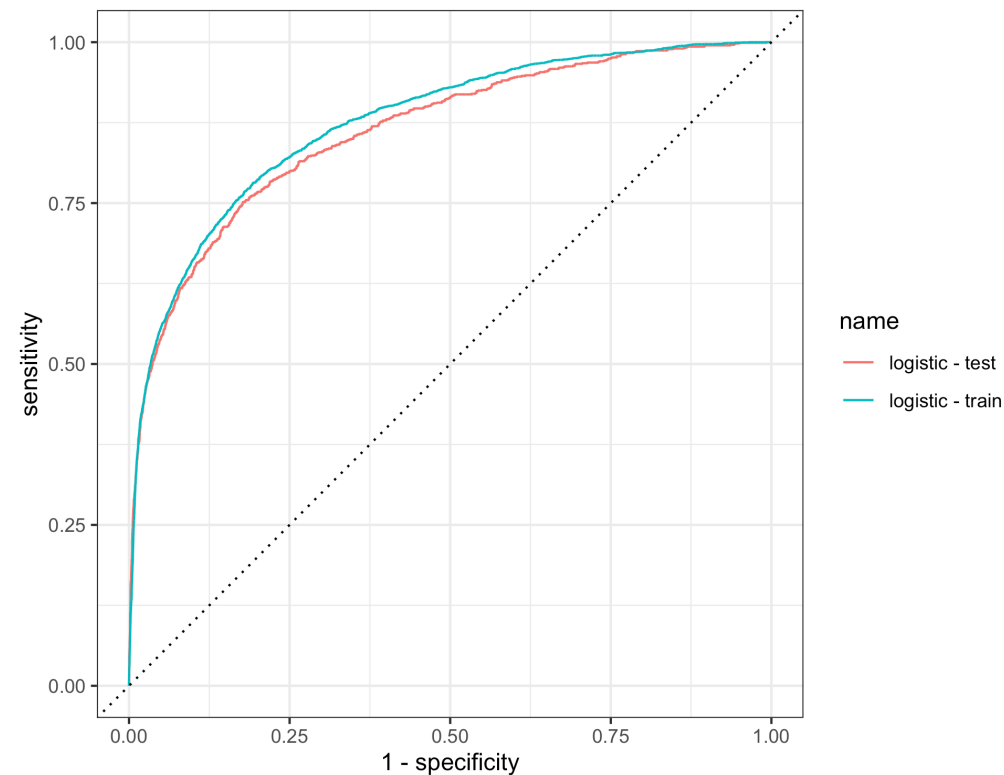


```
1 roc_auc(lr_test_perf, children, .pred_children)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>       <dbl>
1 roc_auc binary      0.861
```


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()
```



Metric sets

```
1 (hotel_metrics = metric_set(accuracy, roc_auc, sensitivity, specificity, brier_class))
```

A metric set, consisting of:

- | | | |
|-------------------|------------------------|---------------------|
| - `accuracy()` | , a class metric | direction: maximize |
| - `roc_auc()` | , a probability metric | direction: maximize |
| - `sensitivity()` | , a class metric | direction: maximize |
| - `specificity()` | , a class metric | direction: maximize |
| - `brier_class()` | , a probability metric | direction: minimize |

```
1 hotel_metrics(lr_train_perf, truth = children, estimate = .pred_class, .pred_children)
```

A tibble: 5 × 3

	.metric <chr>	.estimator <chr>	.estimate <dbl>
1	accuracy	binary	0.936
2	sensitivity	binary	0.350
3	specificity	binary	0.988
4	roc_auc	binary	0.874
5	brier_class	binary	0.0519

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. As such it is necessary 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 × 56

	lead_time <dbl>	stays_in_weekend_nights <dbl>	stays_in_week_nights <dbl>	adults <dbl>	is_repeated_guest <dbl>
1	-0.858	-0.938	-0.767	0.337	-0.213
2	0.160	1.09	1.32	0.337	-0.213
3	-0.146	1.09	-0.245	0.337	-0.213
4	-0.365	-0.938	-0.245	0.337	-0.213
5	-0.267	-0.938	0.278	-3.59	-0.213
6	-0.814	1.09	-0.245	0.337	-0.213
7	0.544	0.0735	-0.245	0.337	-0.213
8	-0.584	-0.938	-0.767	-1.63	-0.213
9	-0.376	-0.938	-0.245	0.337	-0.213
10	3.75	0.0735	-0.767	0.337	-0.213

i 37,490 more rows

i 51 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 <dbl>, hotel_Resort_Hotel <dbl>, meal_FB <dbl>,
meal_HB <dbl>, meal_SC <dbl>, meal_Undefined <dbl>, market_segment_Complementary <dbl>,
market_segment_Corporate <dbl>, market_segment_Direct <dbl>, market_segment_Groups <dbl>,
market_segment_Offline_TA.T0 <dbl>, market_segment_Offline_TA <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 —

8 Recipe Steps

- step_date()
- step_mutate()
- step_rm()
- step_rm()
- step_unknown()
- step_dummy()
- step_zv()
- step_normalize()

— Model —

Logistic Regression Model Specification (classification)

Main Arguments:

```
penalty = tune()
mixture = 1
```

Computational engine: glmnet

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

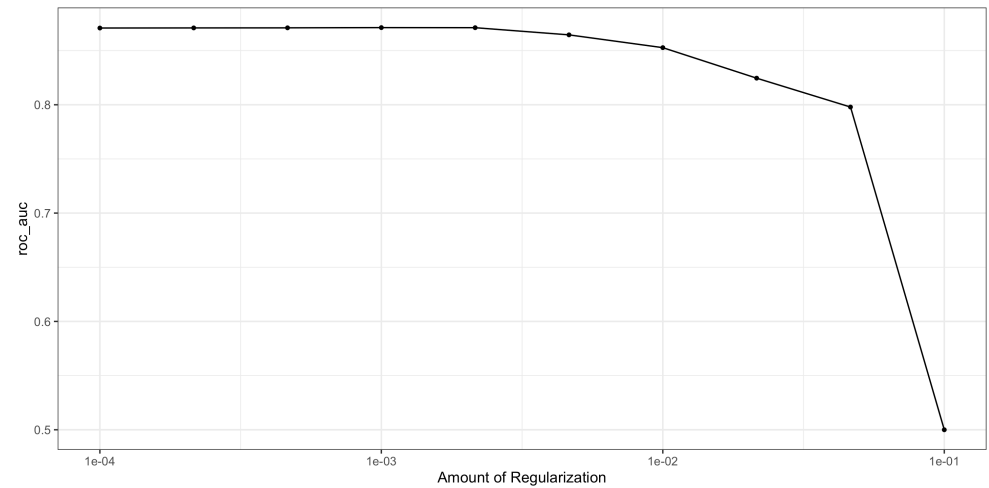
	splits	id	.metrics	.notes	.predictions
	<list>	<chr>	<list>	<list>	<list>
1	<split [30000/7500]>	Fold1	<tibble>	<tibble>	<tibble>
2	<split [30000/7500]>	Fold2	<tibble>	<tibble>	<tibble>
3	<split [30000/7500]>	Fold3	<tibble>	<tibble>	<tibble>
4	<split [30000/7500]>	Fold4	<tibble>	<tibble>	<tibble>
5	<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    0.871     5 0.00412
2 0.000215 roc_auc  binary    0.871     5 0.00419
3 0.000464 roc_auc  binary    0.871     5 0.00419
4 0.001    roc_auc  binary    0.871     5 0.00414
5 0.00215  roc_auc  binary    0.871     5 0.00399
6 0.00464  roc_auc  binary    0.865     5 0.00361
7 0.01     roc_auc  binary    0.853     5 0.00459
8 0.0215   roc_auc  binary    0.825     5 0.00684
9 0.0464   roc_auc  binary    0.798     5 0.00670
10 0.1     roc_auc  binary    0.5       5 0
# i 1 more variable: .config <chr>
```

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



The “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	0.871	5	0.00414	pre0_mod0...
2	0.00215	roc_auc	binary	0.871	5	0.00399	pre0_mod0...
3	0.000464	roc_auc	binary	0.871	5	0.00419	pre0_mod0...
4	0.000215	roc_auc	binary	0.871	5	0.00419	pre0_mod0...
5	0.0001	roc_auc	binary	0.871	5	0.00412	pre0_mod0...

```
1 ( lasso_best = lasso_grid |>
2   select_best() )
```

A tibble: 1 × 2

	penalty	.config
	<dbl>	<chr>
1	0.001	pre0_mod04_post0

Extracting predictions

Since we used `control_grid(save_pred = TRUE)` in `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) )
```

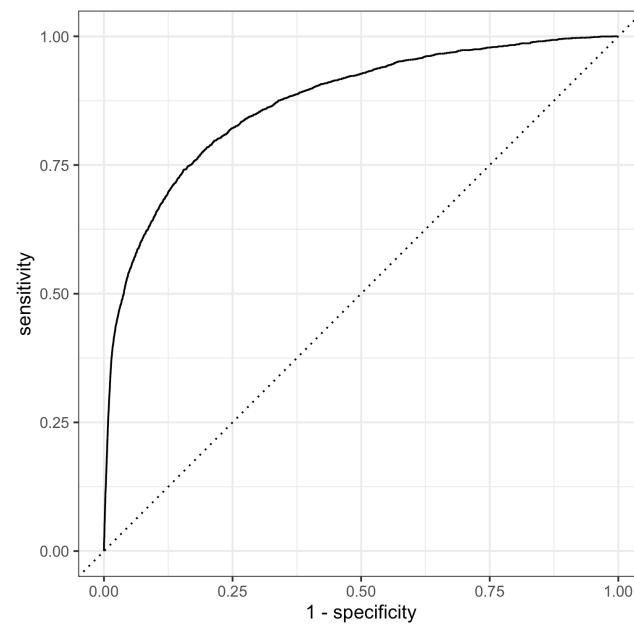
```
# A tibble: 37,500 × 7
```

	.pred_children	.pred_none	id	children	.row	penalty
	<dbl>	<dbl>	<chr>	<fct>	<int>	<dbl>
1	0.0338	0.966	Fold1	none	2	0.001
2	0.0116	0.988	Fold1	none	3	0.001
3	0.0103	0.990	Fold1	none	17	0.001
4	0.0827	0.917	Fold1	children	22	0.001
5	0.839	0.161	Fold1	children	27	0.001
6	0.0357	0.964	Fold1	none	28	0.001
7	0.0103	0.990	Fold1	none	30	0.001
8	0.0432	0.957	Fold1	none	38	0.001
9	0.0204	0.980	Fold1	none	44	0.001
10	0.00831	0.992	Fold1	none	49	0.001

```
# 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      0.871
```

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 —
8 Recipe Steps

- step_date()
- step_mutate()
- step_rm()
- step_rm()
- step_unknown()
- step_dummy()
- step_zv()
- step_normalize()

— Model —

Call: glmnet::glmnet(x = maybe_matrix(x), y = y, family = "binomial", alpha = ~1)

	Df	%Dev	Lambda
1	0	0.00	0.080750
2	1	2.56	0.073580
3	2	5.06	0.067040
4	3	7.45	0.061090
5	3	9.79	0.055660

Test Performance (out-of-sample)

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

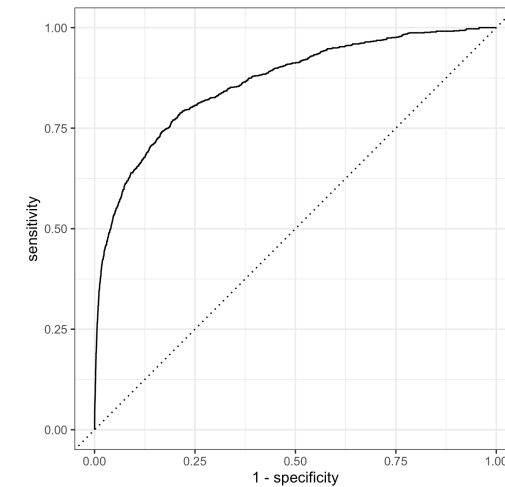
```
1 conf_mat(lasso_test_perf, children, .pred_class)
```

	Truth	
Prediction	children	none
children	331	120
none	680	11369

```
1 hotel_metrics(lasso_test_perf, truth = children, e
```

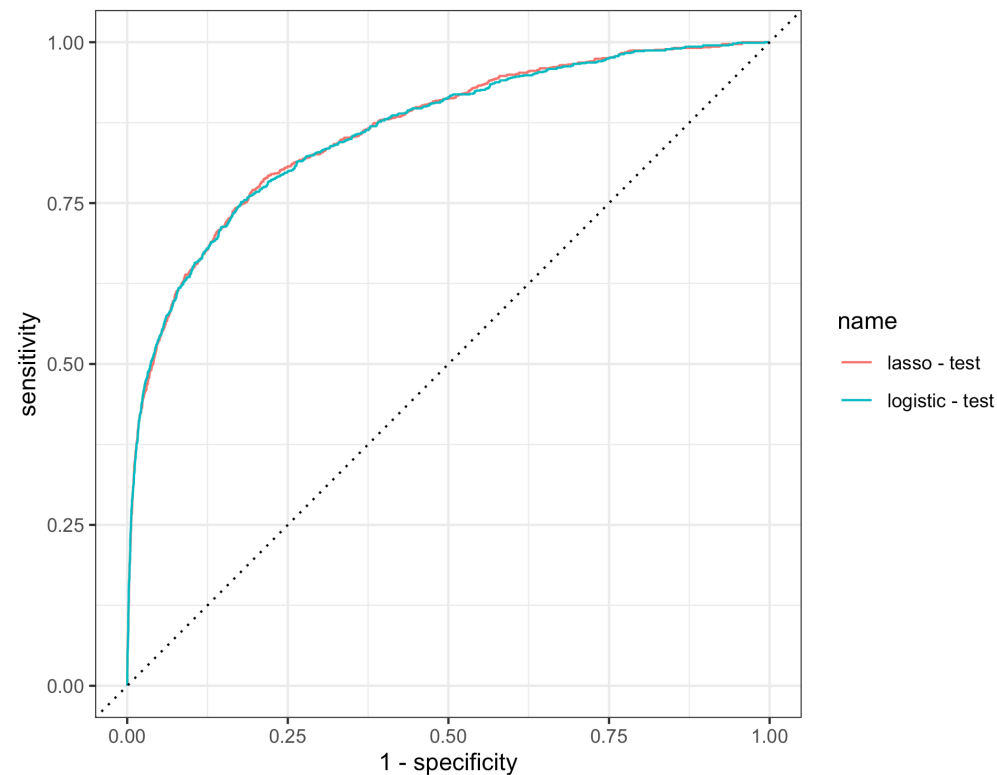
```
# A tibble: 5 × 3
  .metric .estimator .estimate
  <chr>    <chr>      <dbl>
1 accuracy binary      0.936
2 sensitivity binary      0.327
3 specificity binary      0.990
4 roc_auc   binary      0.863
5 brier_class binary      0.0523
```

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



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

Same recipe as before but we skip dummy coding as it is not needed by rpart,

```
1 dt_recipe = recipe(children ~ ., data = hotel_train) |>
2   step_date(arrival_date, features = c("dow", "month"), label=TRUE) |>
3   step_mutate(
4     season = case_when(
5       arrival_date_month %in% c(12, 1, 2) ~ "Winter",
6       arrival_date_month %in% c(3, 4, 5) ~ "Spring",
7       arrival_date_month %in% c(6, 7, 8) ~ "Summer",
8       arrival_date_month %in% c(9, 10, 11) ~ "Fall"
9     ) |>
10    factor(levels = c("Winter", "Spring", "Summer", "Fall"))
11  ) |>
12  step_rm(arrival_date, arrival_date_month) |>
13  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 dt_tune = dt_work |>  
2   tune_grid(  
3     hotel_vf,  
4     grid = dt_grid,  
5     control = control_grid(save_pred = TRUE),  
6     metrics = hotel_metrics  
7   )
```

How many decision tree models were fit by `tune_grid()`?

Tuning results

```
1 dt_tune |>
2   collect_metrics() |>
3   arrange(desc(mean))
```

A tibble: 135 × 9

	cost_complexity	tree_depth	min_n	.metric	.estimator	mean
	<dbl>	<int>	<int>	<chr>	<chr>	<dbl>
1	0.00000000001	8	40	specif...	binary	0.987
2	0.00000316	8	40	specif...	binary	0.987
3	0.00000000001	8	21	specif...	binary	0.987
4	0.00000316	8	21	specif...	binary	0.987
5	0.00000000001	8	2	specif...	binary	0.986
6	0.00000316	8	2	specif...	binary	0.986
7	0.00000000001	15	40	specif...	binary	0.984
8	0.00000316	15	40	specif...	binary	0.984
9	0.1	1	2	specif...	binary	0.979
10	0.1	1	21	specif...	binary	0.979

i 125 more rows

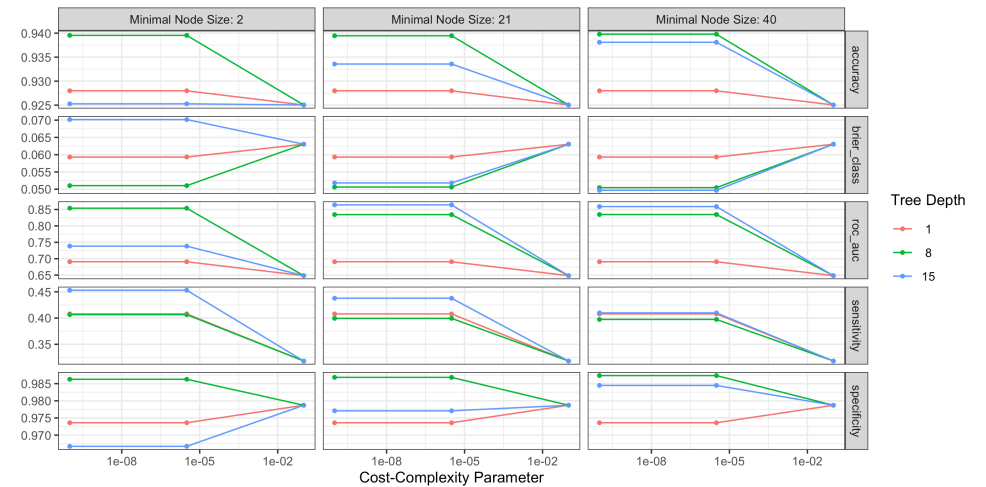
i 3 more variables: n <int>, std_err <dbl>, .config <chr>

“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     2 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 pre0_mod08_post0
```

```
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_mutate()
- step_rm()
- step_rm()

— Model —

n= 37500

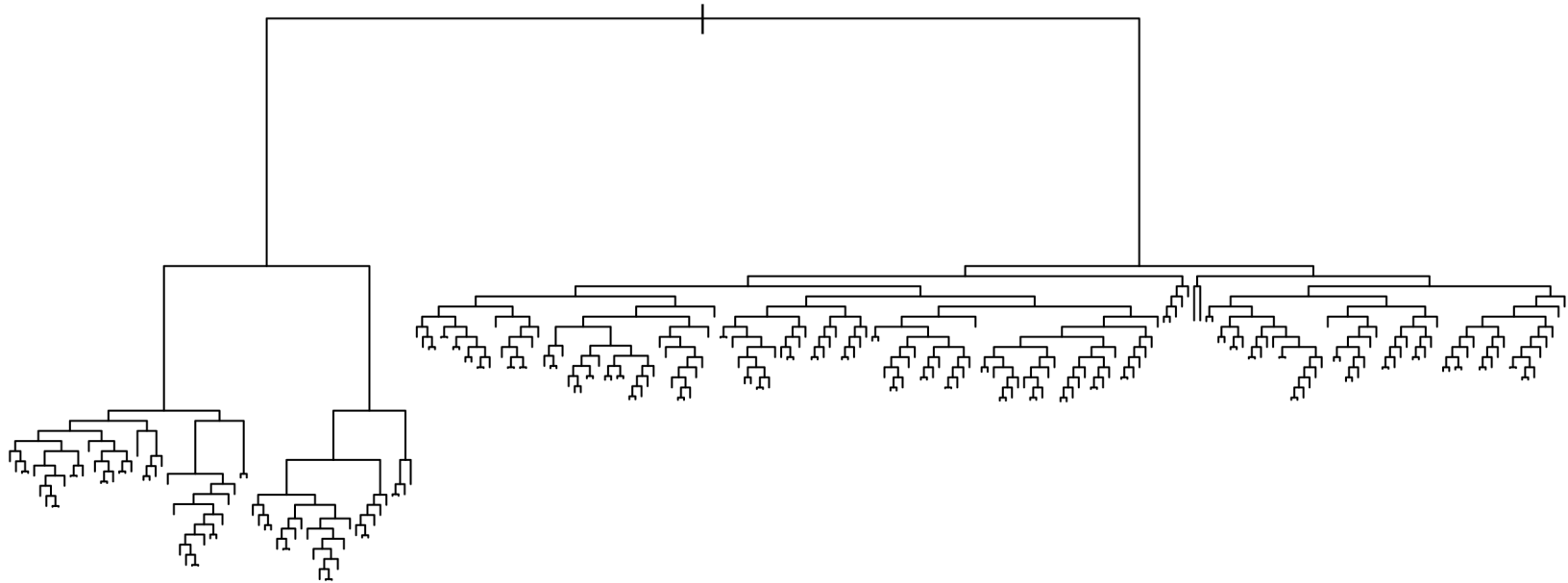
node), split, n, loss, yval, (yprob)

* denotes terminal node

- 1) root 37500 3027 none (0.080720000 0.919280000)
- 2) reserved_room_type=C,F,G,H 2147 910 children (0.576152771 0.423847229)
- 4) market_segment=Online_TA 1218 350 children (0.712643678 0.287356322)
- 8) average_daily_rate>=140.715 890 196 children (0.779775281 0.220224719)
- 16) adults< 2.5 769 139 children (0.819245774 0.180754226)
- 32) booking_changes< 0.5 581 77 children (0.867469880 0.132530120)
- 64) hotel=City_Hotel 363 26 children (0.928374656 0.071625344)
- 128) reserved_room_type=F 311 15 children (0.951768480 0.048231511) *

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"))
```

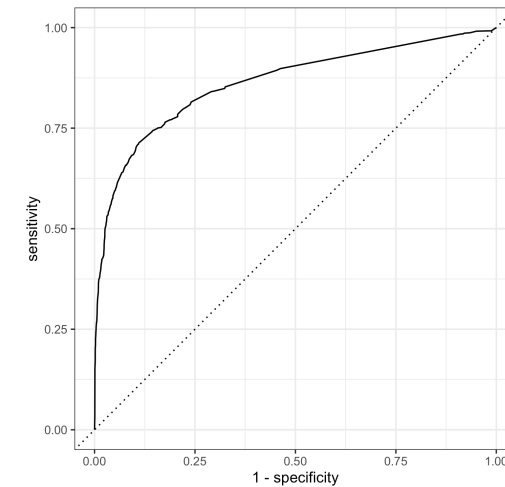
```
1 conf_mat(dt_test_perf, children, .pred_class)
```

	Truth	
Prediction	children	none
children	450	268
none	561	11221

```
1 hotel_metrics(dt_test_perf, truth = children, esti
```

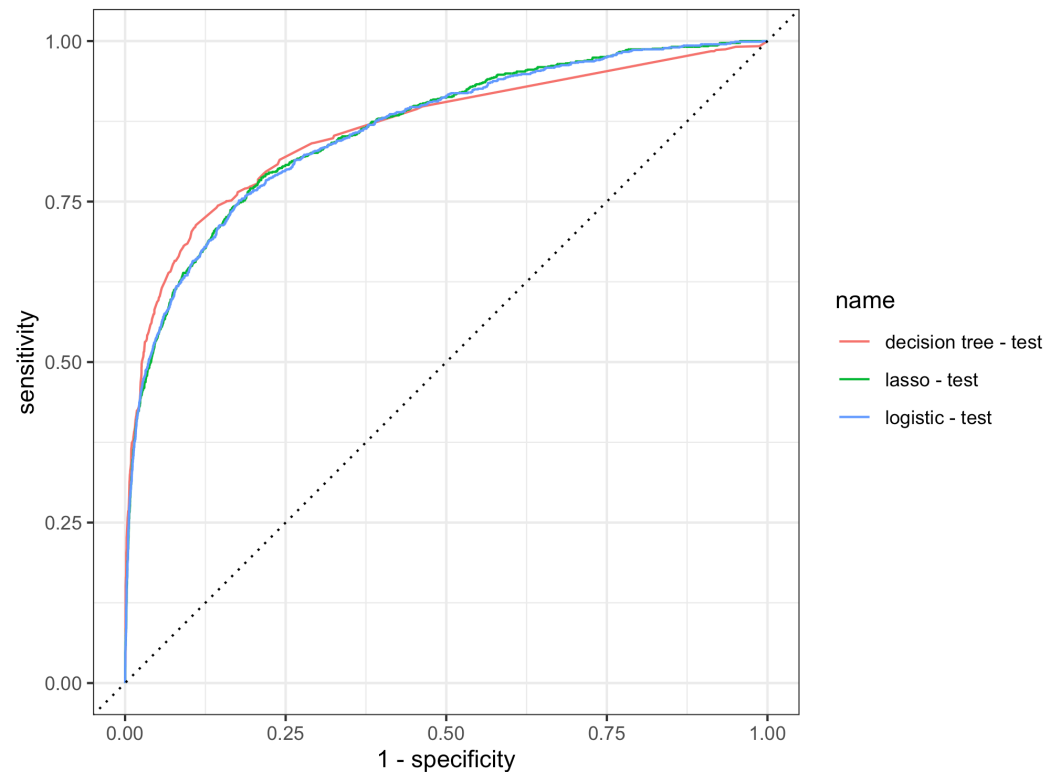
```
# A tibble: 5 × 3
  .metric .estimator .estimate
  <chr>   <chr>       <dbl>
1 accuracy binary      0.934
2 sensitivity binary      0.445
3 specificity binary      0.977
4 roc_auc  binary      0.862
5 brier_class binary      0.0512
```

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



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 = 1) |>  
3   set_mode("classification")
```

Recipe & workflow

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

```
1 rf_recipe = dt_recipe
```

```
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     2 roc_auc binary    0.912     5 0.00445
2     3    31 roc_auc binary    0.911     5 0.00446
3     8    18 roc_auc binary    0.909     5 0.00431
4    10    35 roc_auc binary    0.907     5 0.00416
5    15    23 roc_auc binary    0.903     5 0.00394
6    12     6 roc_auc binary    0.903     5 0.00414
7    17    40 roc_auc binary    0.903     5 0.00402
8    19    10 roc_auc binary    0.900     5 0.00345
9    22    27 roc_auc binary    0.900     5 0.00398
10     1    14 roc_auc binary    0.867     5 0.00384
# i 1 more variable: .config <chr>
```

“Best” parameters

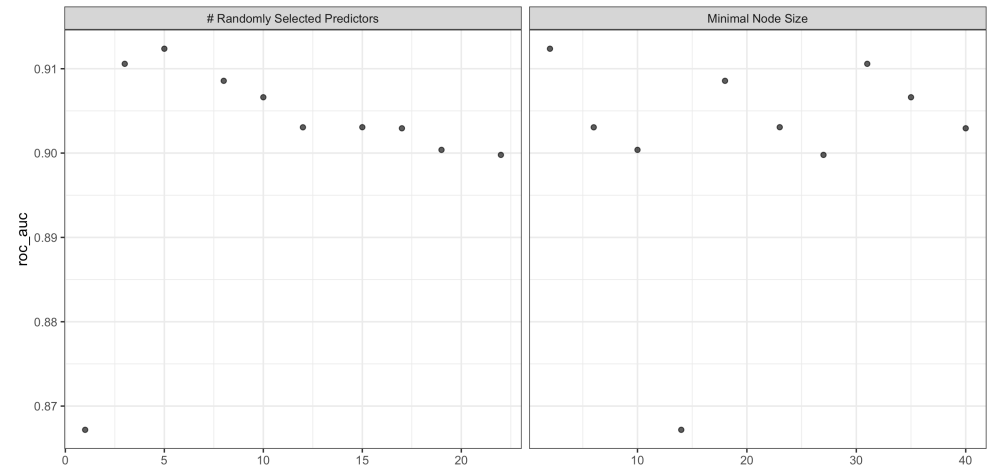
```
1 rf_tune |>  
2   show_best(metric = "roc_auc")
```

A tibble: 5 × 8

	mtry	min_n	.metric	.estimator	mean	n
	<int>	<int>	<chr>	<chr>	<dbl>	<int>
1	5	2	roc_auc	binary	0.912	5
2	3	31	roc_auc	binary	0.911	5
3	8	18	roc_auc	binary	0.909	5
4	10	35	roc_auc	binary	0.907	5
5	15	23	roc_auc	binary	0.903	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_mutate()
- step_rm()
- step_rm()

— Model —

Ranger result

Call:

ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~5L, x), num.trees = ~100, min.node.size =

Type: Probability estimation

Number of trees: 100

Sta 523 - Fall 2025

Sample size: 37500

Number of independent variables: 22
Mtry: 5
Target node size: 2
Variable importance mode: none
Splitrule: qini

Test Performance (out-of-sample)

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

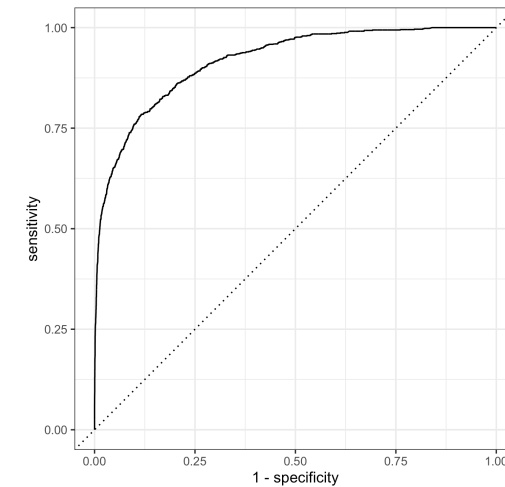
```
1 conf_mat(rf_test_perf, children, .pred_class)
```

	Truth	
Prediction	children	none
children	420	88
none	591	11401

```
1 hotel_metrics(rf_test_perf, truth = children, esti
```

```
# A tibble: 5 × 3
  .metric      .estimator .estimate
  <chr>        <chr>      <dbl>
1 accuracy    binary      0.946
2 sensitivity binary      0.415
3 specificity  binary      0.992
4 roc_auc     binary      0.917
5 brier_class binary      0.0435
```

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



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()
```

