Hotels

GS

4/8/2021

Loading packages

```
library(tidyverse)
library(tidymodels)
library(vip)
theme_set(theme_bw())
```

We will build a model to predict which actual hotel stays included children and/or babies, and which did not.

Loading dataset

```
hotels <-
  read_csv('https://tidymodels.org/start/case-study/hotels.csv') %>%
  mutate_if(is.character, as.factor)
```

Checking data

```
glimpse(hotels)
```

```
Rows: 50,000
Columns: 23
$ hotel                          <fct> City_Hotel, City_Hotel, Resort_Hotel, R~
$ lead_time                      <dbl> 217, 2, 95, 143, 136, 67, 47, 56, 80, 6~
$ stays_in_weekend_nights        <dbl> 1, 0, 2, 2, 1, 2, 0, 0, 0, 2, 1, 0, 1, ~
$ stays_in_week_nights           <dbl> 3, 1, 5, 6, 4, 2, 2, 3, 4, 2, 2, 1, 2, ~
$ adults                         <dbl> 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 1, 2, ~
$ children                       <fct> none, none, none, none, none, none, chi~
$ meal                           <fct> BB, BB, BB, HB, HB, SC, BB, BB, BB, BB,~
$ country                        <fct> DEU, PRT, GBR, ROU, PRT, GBR, ESP, ESP,~
$ market_segment                 <fct> Offline_TA/TO, Direct, Online_TA, Onlin~
$ distribution_channel           <fct> TA/TO, Direct, TA/TO, TA/TO, Direct, TA~
$ is_repeated_guest              <dbl> 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, ~
$ previous_bookings_not_canceled <dbl> 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, ~
$ assigned_room_type             <fct> A, K, A, A, F, A, C, A, D, A, D, D, A, ~
$ booking_changes                <dbl> 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ deposit_type                   <fct> No_Deposit, No_Deposit, No_Deposit, No_~
```

```
$ days_in_waiting_list           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ customer_type                  <fct> Transient-Party, Transient, Transient, ~
$ average_daily_rate             <dbl> 80.75, 170.00, 8.00, 81.00, 157.60, 49.~
$ required_car_parking_spaces    <fct> none, none, none, none, none, none, non~
$ total_of_special_requests      <dbl> 1, 3, 2, 1, 4, 1, 1, 1, 1, 1, 0, 1, 0, ~
$ arrival_date                   <date> 2016-09-01, 2017-08-25, 2016-11-19, 20~
```

```
colSums(is.na(hotels))
```

```
                         hotel                       lead_time
                             0                               0
       stays_in_weekend_nights          stays_in_week_nights
                             0                               0
                        adults                        children
                             0                               0
                          meal                         country
                             0                               0
                market_segment          distribution_channel
                             0                               0
             is_repeated_guest         previous_cancellations
                             0                               0
previous_bookings_not_canceled            reserved_room_type
                             0                               0
             assigned_room_type               booking_changes
                             0                               0
                   deposit_type           days_in_waiting_list
                             0                               0
                  customer_type             average_daily_rate
                             0                               0
   required_car_parking_spaces      total_of_special_requests
                             0                               0
                   arrival_date
                             0
```

Outcome variable 'children' is a factor variable with two levels

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

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

Data splitting

```
set.seed(2021)
hotels_split <- initial_split(hotels,
```

```
                              prop= .75,
                              strata = children)

hotel_train <- training(hotels_split)
hotel_test <- testing(hotels_split)
```

Data resampling (CV)

```
hotel_cv <- vfold_cv(hotel_train, v = 10)
```

Penalized model

```
pen_model <-
  logistic_reg(penalty = tune(), mixture = 1) %>%
  set_engine('glmnet')
```

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

pen_rec <-
  recipe(children ~ ., data = hotel_train) %>%
  step_date(arrival_date) %>%
  step_holiday(arrival_date, holidays = holidays) %>%
  step_rm(arrival_date) %>%
  step_dummy(all_nominal(), -all_outcomes()) %>%
  step_zv(all_predictors()) %>%
  step_normalize(all_predictors())
```

```
pen_wf <-
  workflow() %>%
  add_model(pen_model) %>%
  add_recipe(pen_rec)
```

Grid generation

```
pen_reg_grid <- tibble(penalty = 10^seq(-4, -1, length.out = 30))
```

Run the model

```
pen_res <-
  tune_grid(
    pen_wf,
    resamples = hotel_cv,
```

```
    grid = pen_reg_grid,
    control = control_grid(save_pred = T),
    metrics = metric_set(roc_auc)
  )
pen_res
```

```
# Tuning results
# 10-fold cross-validation
# A tibble: 10 x 5
   splits             id     .metrics          .notes          .predictions
   <list>             <chr>  <list>            <list>          <list>
 1 <split [33750/375~ Fold01 <tibble [30 x ~ <tibble [0 x ~ <tibble [112,500 x ~
 2 <split [33750/375~ Fold02 <tibble [30 x ~ <tibble [0 x ~ <tibble [112,500 x ~
 3 <split [33750/375~ Fold03 <tibble [30 x ~ <tibble [0 x ~ <tibble [112,500 x ~
 4 <split [33750/375~ Fold04 <tibble [30 x ~ <tibble [0 x ~ <tibble [112,500 x ~
 5 <split [33750/375~ Fold05 <tibble [30 x ~ <tibble [0 x ~ <tibble [112,500 x ~
 6 <split [33750/375~ Fold06 <tibble [30 x ~ <tibble [0 x ~ <tibble [112,500 x ~
 7 <split [33750/375~ Fold07 <tibble [30 x ~ <tibble [0 x ~ <tibble [112,500 x ~
 8 <split [33750/375~ Fold08 <tibble [30 x ~ <tibble [0 x ~ <tibble [112,500 x ~
 9 <split [33750/375~ Fold09 <tibble [30 x ~ <tibble [0 x ~ <tibble [112,500 x ~
10 <split [33750/375~ Fold10 <tibble [30 x ~ <tibble [0 x ~ <tibble [112,500 x ~
```

Metrics

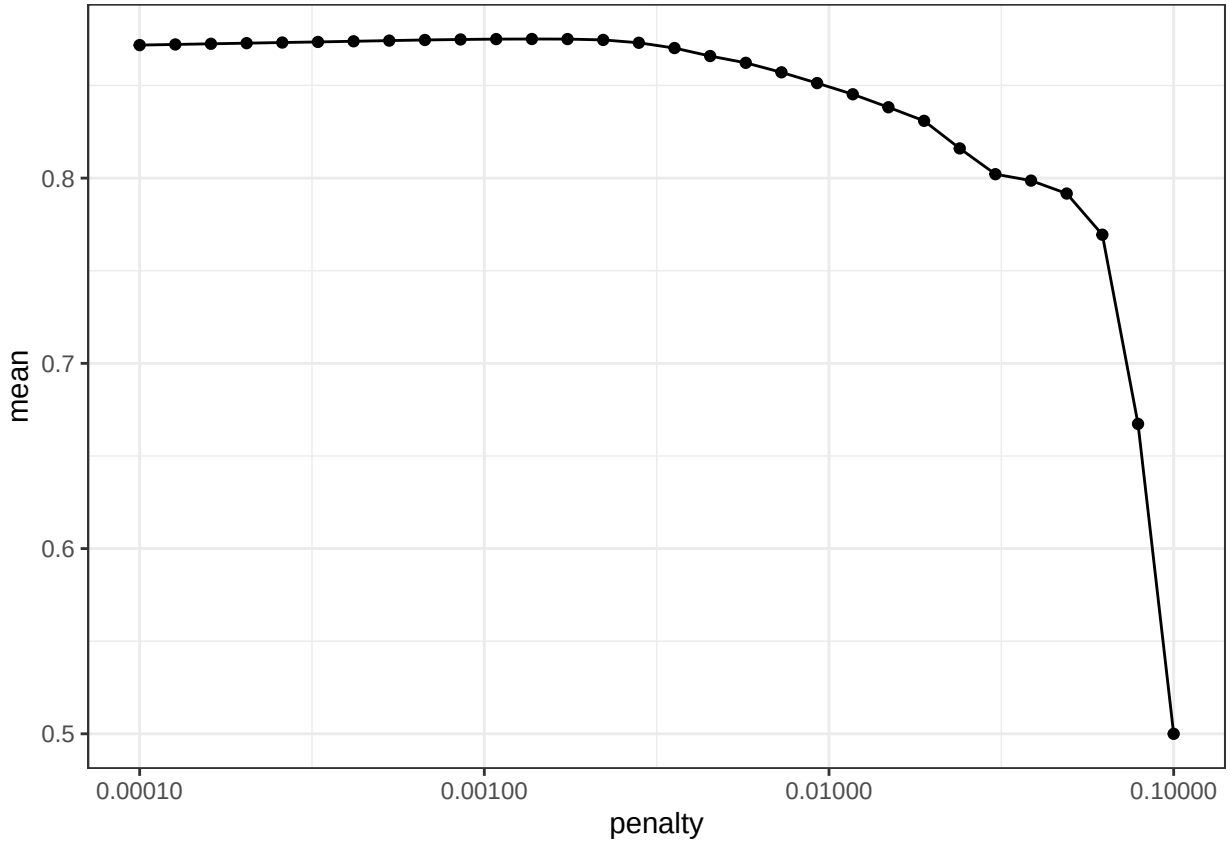
```
pen_metrics <-
  pen_res %>%
  collect_metrics()
pen_metrics
```

```
# A tibble: 30 x 7
    penalty .metric .estimator  mean     n std_err .config
      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
 1 0.0001   roc_auc binary     0.872    10 0.00328 Preprocessor1_Model01
 2 0.000127 roc_auc binary     0.872    10 0.00329 Preprocessor1_Model02
 3 0.000161 roc_auc binary     0.873    10 0.00330 Preprocessor1_Model03
 4 0.000204 roc_auc binary     0.873    10 0.00331 Preprocessor1_Model04
 5 0.000259 roc_auc binary     0.873    10 0.00333 Preprocessor1_Model05
 6 0.000329 roc_auc binary     0.874    10 0.00334 Preprocessor1_Model06
 7 0.000418 roc_auc binary     0.874    10 0.00334 Preprocessor1_Model07
 8 0.000530 roc_auc binary     0.874    10 0.00332 Preprocessor1_Model08
 9 0.000672 roc_auc binary     0.875    10 0.00330 Preprocessor1_Model09
10 0.000853 roc_auc binary     0.875    10 0.00327 Preprocessor1_Model10
# ... with 20 more rows
```

Plots metrics

```
pen_metrics %>%
  ggplot(aes(x = penalty, y= mean)) +
  geom_point() +
```

```
  geom_line() +
  scale_x_log10(labels = scales::label_number())
```



Best penalized models

```
top_models <-
  pen_res %>%
  show_best(metric = 'roc_auc') %>%
  arrange(penalty)

top_models
```

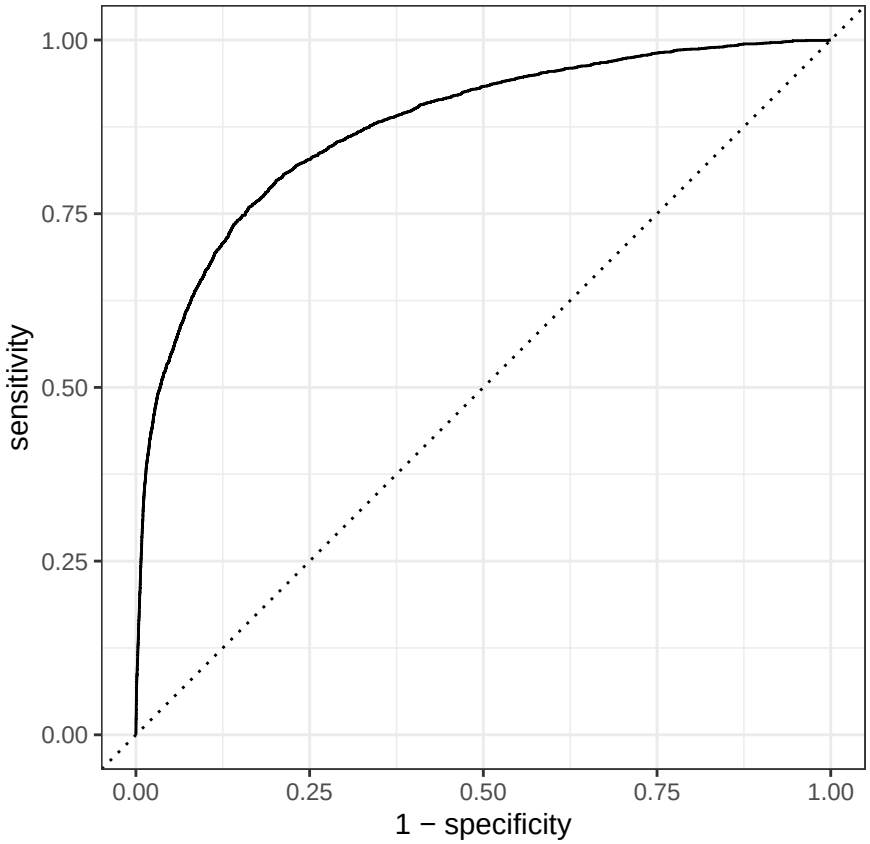
```
# A tibble: 5 x 7
   penalty .metric .estimator  mean     n std_err .config
     <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1 0.000853 roc_auc binary     0.875    10 0.00327 Preprocessor1_Model10
2 0.00108  roc_auc binary     0.875    10 0.00326 Preprocessor1_Model11
3 0.00137  roc_auc binary     0.875    10 0.00326 Preprocessor1_Model12
4 0.00174  roc_auc binary     0.875    10 0.00323 Preprocessor1_Model13
5 0.00221  roc_auc binary     0.875    10 0.00315 Preprocessor1_Model14
```

```
pen_best <- pen_res %>%
  select_best()
pen_best
```

```
# A tibble: 1 x 2
  penalty .config
    <dbl> <chr>
1 0.00137 Preprocessor1_Model12
```

AUC

```
pen_auc <-
  pen_res %>%
  collect_predictions(parameters = pen_best) %>%
  roc_curve(children, .pred_children) %>%
  mutate(model = "Logistic Regression")

autoplot(pen_auc)
```



Random Forest

```
cores <- parallel::detectCores()
cores
```

```
[1] 16
```

RF Model

```
rf_mod <-
  rand_forest(mtry = tune(), min_n = tune(), trees = 500) %>%
  set_engine("ranger", num.threads = cores) %>%
  set_mode("classification")
```

RF Recipe

```
rf_rec <-
  recipe(children ~ ., data = hotel_train) %>%
  step_date(arrival_date) %>%
  step_holiday(arrival_date) %>%
  step_rm(arrival_date)
```

RF workflow

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

```
rf_mod
```

```
Random Forest Model Specification (classification)

Main Arguments:
  mtry = tune()
  trees = 500
  min_n = tune()

Engine-Specific Arguments:
  num.threads = cores

Computational engine: ranger
```

```
rf_wf
```

```
== Workflow ====================================================================
Preprocessor: Recipe
Model: rand_forest()

-- Preprocessor ----------------------------------------------------------------
3 Recipe Steps

* step_date()
* step_holiday()
```

```
* step_rm()

-- Model ---------------------------------------------------------------------
Random Forest Model Specification (classification)

Main Arguments:
  mtry = tune()
  trees = 500
  min_n = tune()

Engine-Specific Arguments:
  num.threads = cores

Computational engine: ranger
```

```
rf_mod %>% parameters
```

```
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.parameters' for more information.
```

Run RF model

```
set.seed(2021)
rf_res <-
  tune_grid(rf_wf,
            resamples = hotel_cv,
            grid = 20,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(roc_auc))
```

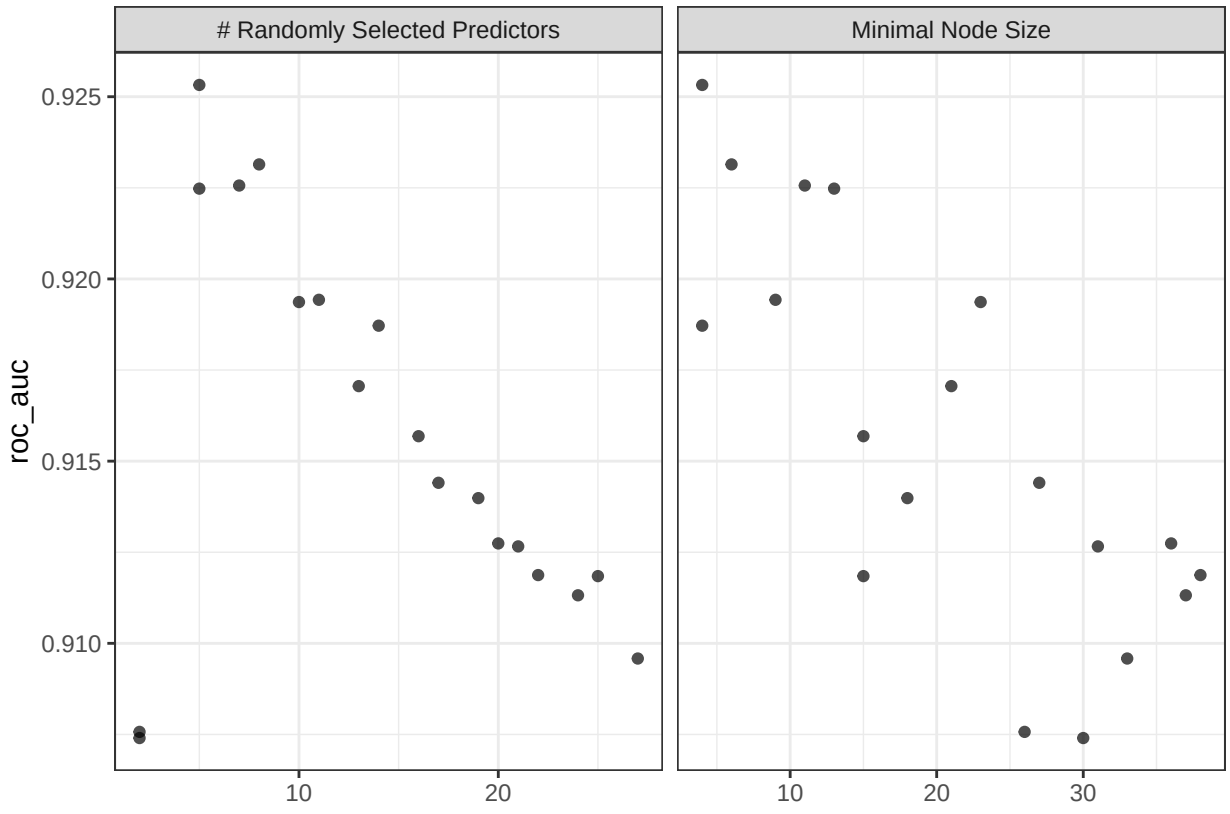
Metrics

```
rf_res %>% show_best(metric = 'roc_auc')
```

```
# A tibble: 5 x 8
   mtry min_n .metric .estimator  mean     n std_err .config
  <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1     5     4 roc_auc binary     0.925    10 0.00152 Preprocessor1_Model19
2     8     6 roc_auc binary     0.923    10 0.00181 Preprocessor1_Model11
3     7    11 roc_auc binary     0.923    10 0.00187 Preprocessor1_Model05
4     5    13 roc_auc binary     0.922    10 0.00155 Preprocessor1_Model18
5    11     9 roc_auc binary     0.919    10 0.00203 Preprocessor1_Model02
```

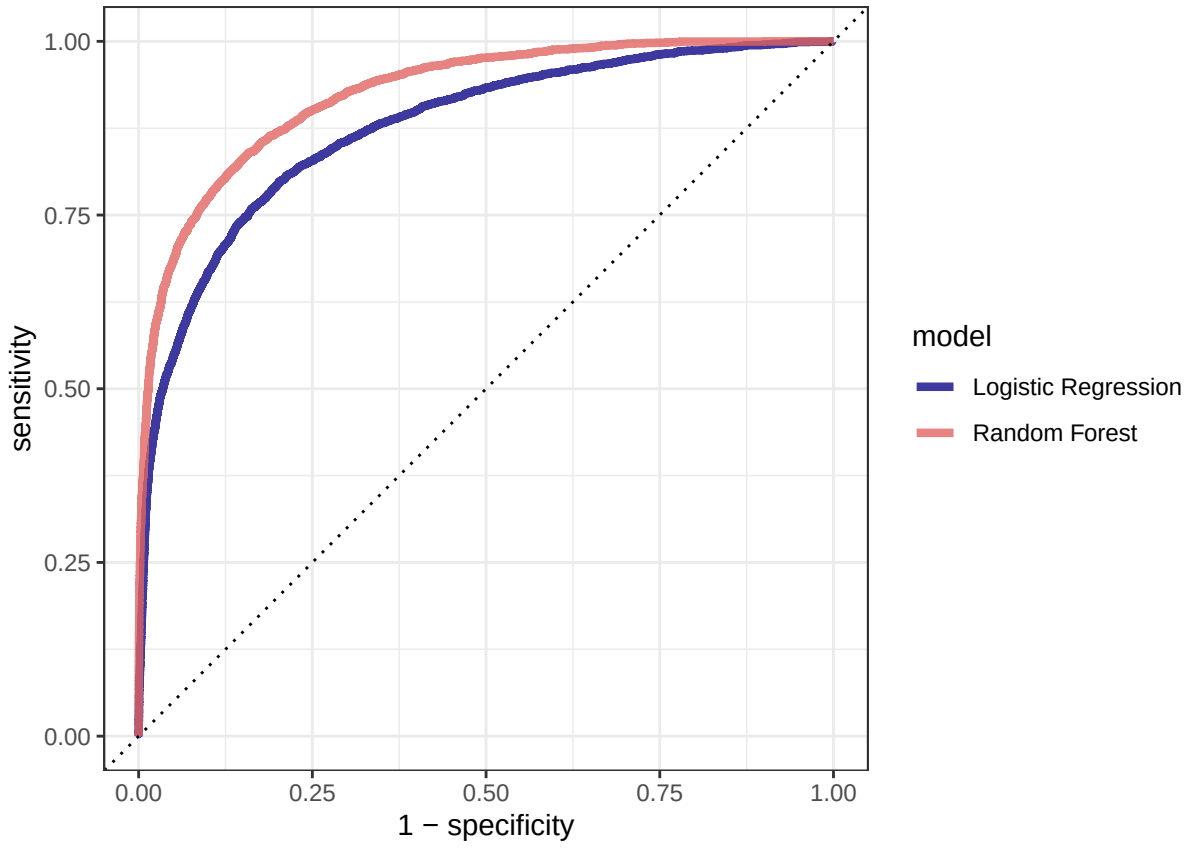
```
rf_best <- rf_res %>% select_best(metric = 'roc_auc')
```

```
autoplot(rf_res)
```




```
rf_auc <-
  rf_res %>%
  collect_predictions(parameters = rf_best) %>%
  roc_curve(children, .pred_children) %>%
  mutate(model = "Random Forest")
```

```
bind_rows(rf_auc, pen_auc) %>%
  ggplot(aes(x = 1 - specificity, y = sensitivity, col = model)) +
  geom_path(lwd = 1.5, alpha = 0.8) +
  geom_abline(lty = 3) +
  coord_equal() +
  scale_color_viridis_d(option = "plasma", end = .6)
```



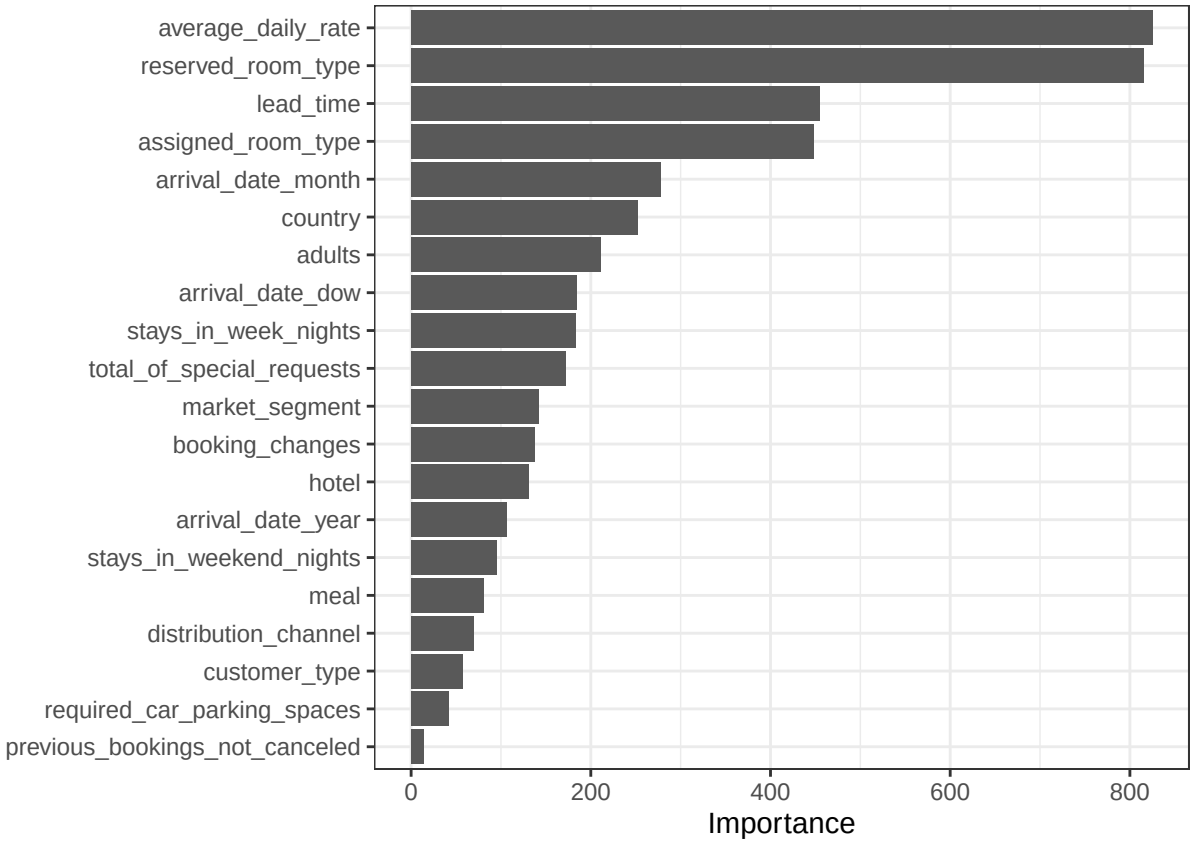

```
last_rf_mod <-
  rand_forest(mtry = 8, min_n = 7, trees = 500) %>%
  set_engine("ranger", num.threads = cores, importance = "impurity") %>%
  set_mode("classification")

# the last workflow
last_rf_workflow <-
  rf_wf %>%
  update_model(last_rf_mod)

# the last fit
set.seed(2021)
last_rf_fit <-
  last_rf_workflow %>%
  last_fit(hotels_split)

last_rf_fit
```

```
# Resampling results
# Manual resampling
# A tibble: 1 x 6
  splits         id          .metrics     .notes       .predictions     .workflow
  <list>         <chr>       <list>       <list>       <list>           <list>
1 <split [37500~ train/test~ <tibble [2 ~ <tibble [0~ <tibble [12,500~ <workflo~
```

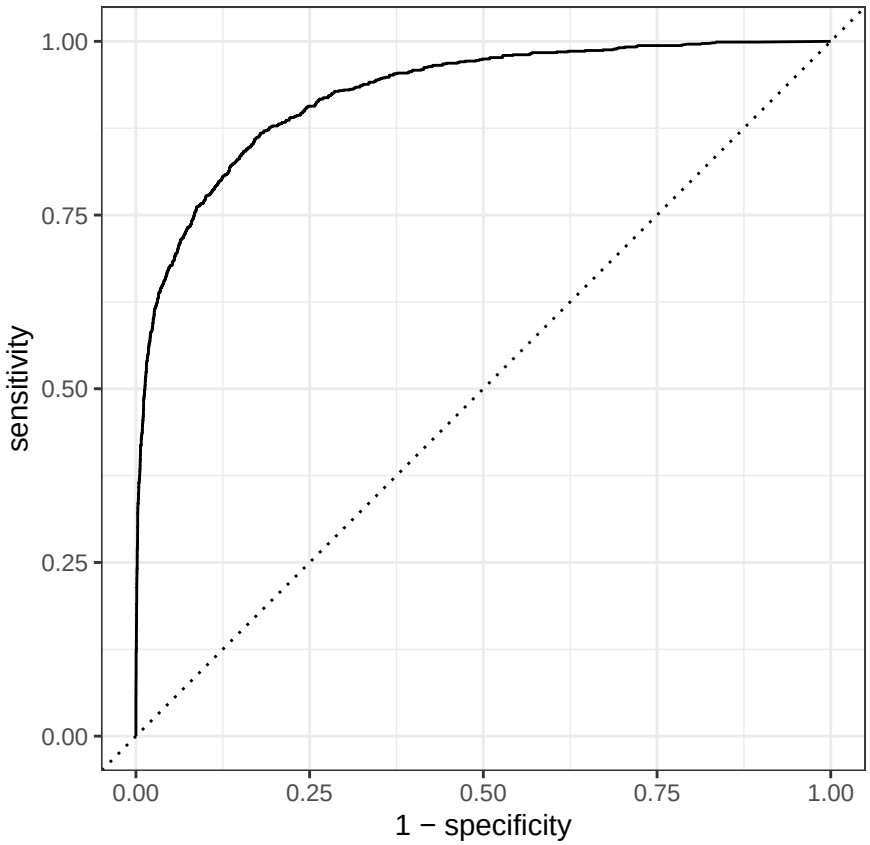
```
last_rf_fit %>%
  collect_metrics()
```

```
# A tibble: 2 x 4
  .metric  .estimator .estimate .config
  <chr>    <chr>          <dbl> <chr>
1 accuracy binary         0.947 Preprocessor1_Model1
2 roc_auc  binary         0.925 Preprocessor1_Model1
```

```
last_rf_fit %>%
  pluck(".workflow", 1) %>%
  pull_workflow_fit() %>%
  vip(num_features = 20)
```




```
last_rf_fit %>%
  collect_predictions() %>%
  roc_curve(children, .pred_children) %>%
  autoplot()
```


1.00
0.75
0.50
0.25
0.00
sensitivity
0.00
0.25
0.50
0.75
1.00
1 − specificity