3.modelling

Joao Lopes

2024-10-04

Contents

1. BUILD A MODEL **3**
1.1. LOOK AT DATA . . . 3
1.2. FIT A MODEL . . . 3
1.3. USE MODEL TO PREDICT . . . 4

2. BUILD A MODEL USING WORKFLOW **6**
2.1. LOOK AT DATA . . . 6
2.2. SPLIT DATA . . . 7
2.3. CREATE WORKFLOW . . . 7
2.4. FIT A MODEL . . . 8
2.5. EVALUATE MODEL . . . 8

3. BUILD VARIOUS MODELS USING WORKFLOW **10**
3.1. LOOK AT DATA . . . 10
3.2. SPLIT DATA . . . 10
3.3. CREATE WORKFLOW AND FIT MODEL 1 . . . 11
3.4. CREATE WORKFLOW AND FIT MODEL 2 . . . 12
3.5. EVALUATE LAST MODEL . . . 14

1. BUILD A MODEL

[from https://www.tidymodels.org/start/models/]

Problem: Predict the price of a diamond.

Formula: price ~ carat + cut

Model: lm()

```
library("skimr")        #function skim() for descriptive statistics
library("GGally")       #function ggpairs() for data visualization
library("hexbin")       #function geom_hex() for data visualization
library("dotwhisker")   #function dwplot() for model coefficients visualization
library("performance")  #function check_model() for linear model assumptions
library("tidymodels")   #collection of packages for modelling
library("tidyverse")    #collection of packages for data analysis
```

1.1. LOOK AT DATA

```
#loading data and making changes
set.seed(123)
diamonds_data <- diamonds |>
  mutate(
    log_price = log(price),
    log_carat = log(carat),
    fct_cut = factor(cut, ordered = FALSE),
    .keep = "used"
  ) |>
  slice_sample(n = 1000)

#inspect data
glimpse(diamonds_data)

#descriptive statistics
skim(diamonds_data)
```

```
#data visualization
diamonds_data |>
  select(-c(carat, price, cut)) |>
  ggpairs(progress = FALSE)
```

```
ggplot(diamonds_data, aes(x = log_carat, y = log_price)) +
  geom_hex() +
  facet_wrap(~fct_cut, ncol = 1)
```

```
ggplot(diamonds_data, aes(x = log_carat, y = log_price, color = fct_cut)) +
  geom_smooth(method = "lm", formula = "y ~ x")
```

1.2. FIT A MODEL

```
#functional form
?linear_reg
```

```
#estimate or train
t1 <- Sys.time()
```

```r
lm_fit <- linear_reg() |>
  fit(log_price ~ log_carat + fct_cut, data = diamonds_data)
Sys.time() - t1

#analyse model
lm_fit
tidy(lm_fit)
summary(lm_fit$fit)$coef

#evaluate model using same data used to fit it (overfitting?)
diamonds_data_aug <- augment(lm_fit, new_data=diamonds_data)
diamonds_data_aug |> rsq(truth = log_price, estimate= .pred)
diamonds_data_aug |> rmse(truth = log_price, estimate= .pred)

#plot results
tidy(lm_fit) |>
  dwplot(
    dot_args = list(size = 2, color = "black"),
    whisker_args = list(color = "black"),
    vline = geom_vline(xintercept = 0, colour = "grey50", linetype = 2))
```

```r
#check assumptions
check_model(lm_fit$fit)
```

1.3. USE MODEL TO PREDICT

```r
#create new points
new_points <- expand.grid(
  log_carat = log(2),
  fct_cut = c("Fair", "Very Good", "Ideal")
  )
new_points

#get prediction
mean_pred <- predict(lm_fit, new_data = new_points)
mean_pred

#get confidence interval
conf_int_pred <- predict(lm_fit,
                         new_data = new_points,
                         type = "conf_int")
conf_int_pred

#plot points
plot_data <-
  new_points |>
  bind_cols(mean_pred) |>
  bind_cols(conf_int_pred)

ggplot(plot_data, aes(x = fct_cut)) +
  geom_point(aes(y = exp(.pred))) +
  geom_errorbar(aes(ymin = exp(.pred_lower),
                    ymax = exp(.pred_upper)),
```

```
                width = .2) +
labs(x = "diamond cut", y = "diamond price")
```

---

2. BUILD A MODEL USING WORKFLOW

[from https://www.tidymodels.org/start/recipes/]

Problem: Predict if a flight is going to arrive late.

Formula: arr_delay ~ .

Model: glm(family = binomial(link = "logit"))

```
library("nycflights13") #collection of datasets
library("skimr")        #function skim() for descriptive statistics
library("GGally")       #function ggpairs() for data visualization
library("tidymodels")   #collection of packages for modelling
library("tidyverse")    #collection of packages for data analysis
```

2.1. LOOK AT DATA

```
#loading data and making changes
flight_data <-
  flights |>
  mutate(
    # Convert the arrival delay to a factor
    arr_delay = ifelse(arr_delay >= 30, "late", "on_time"),
    arr_delay = factor(arr_delay),
    # We will use the date (not date-time) in the recipe below
    date = lubridate::as_date(time_hour)
  ) |>
  # Only retain the specific columns we will use
  select(dep_time, flight, origin, dest, air_time, distance,
         carrier, date, arr_delay, time_hour) |>
  # Exclude missing data
  na.omit() |>
  # For creating models, it is better to have qualitative columns
  # encoded as factors (instead of character strings)
  mutate_if(is.character, as.factor)

#looking at data
glimpse(flight_data)

#descriptive statistcs
skim(flight_data)
```

```
#data visualization
set.seed(123)
flight_data |>
  slice_sample(n=1000) |>
  select(-c(
    time_hour, flight, #ID of the flights
    dest, carrier,     #too many levels
    dep_time, date     #time and data
    )
  ) |>
  select(arr_delay, where(is.factor), where(is.numeric)) |>
  ggpairs(progress = FALSE)
```

```r
flight_data |>
  count(arr_delay) |>
  mutate(prop = n/sum(n))
```

2.2. SPLIT DATA

```r
set.seed(222)
data_split <- initial_split(flight_data, prop = 3/4)

train_data <- training(data_split)
test_data  <- testing(data_split)
```

2.3. CREATE WORKFLOW

```r
#new recipe
flights_rec <-
  recipe(arr_delay ~ ., data = train_data)

#add role to recipe
flights_rec <-
  flights_rec |>
  update_role(flight, time_hour, new_role = "ID")

#add date features
flights_rec <-
  flights_rec |>
  step_date(date, features = c("dow", "month")) |>
  step_holiday(date, holidays = timeDate::listHolidays("US")) |>
  step_rm(date)

#convert categorical variables
flights_rec <-
  flights_rec |>
  step_dummy(all_nominal_predictors())

#remove columns with zero variance
flights_rec <-
  flights_rec |>
  step_zv(all_predictors())

#full recipe
flights_rec <-
  recipe(arr_delay ~ ., data = train_data) |>
  update_role(flight, time_hour, new_role = "ID") |>
  step_date(date, features = c("dow", "month")) |>
  step_holiday(date, holidays = timeDate::listHolidays("US")) |>
  step_rm(date) |>
  step_dummy(all_nominal_predictors()) |>
  step_zv(all_predictors())
```

2.4. FIT A MODEL

```
#build model specification
lr_mod <-
  logistic_reg() |>
  set_engine("glm")
lr_mod

#build workflow
flights_wflow <-
  workflow() |>
  add_model(lr_mod) |>
  add_recipe(flights_rec)
flights_wflow

#fit the model
t1 <- Sys.time()
flights_fit <-
  flights_wflow |>
  fit(data = train_data)
Sys.time() - t1

flights_fit |>
  tidy()
```

2.5. EVALUATE MODEL

```
#predict using augment()
flights_aug <-
  augment(flights_fit, new_data = test_data)
flights_aug |>
  select(arr_delay, time_hour, flight, .pred_class, .pred_on_time) |>
  print(n = 20)

#evaluate model using confusion matrix
lr_cm <- flights_aug |>
  conf_mat(truth=arr_delay, estimate=.pred_class)
lr_cm

summary(lr_cm)

#evaluate model using ROC curve
methods(autoplot)

flights_aug |>
  roc_curve(truth = arr_delay, .pred_late) |>
  autoplot()

flights_aug |>
  roc_auc(truth = arr_delay, .pred_late)

#extract results
flights_fit |>
```

```r
  extract_fit_parsnip() |>
  tidy() |>
  arrange(desc(abs(statistic))) |>
  slice_head(n = 20) |>
  mutate(
    importance = abs(statistic),
    term = reorder(term, importance)
  ) |>
  ggplot(aes(x = importance, y = term)) +
  geom_col()
```

---

3. BUILD VARIOUS MODELS USING WORKFLOW

[from https://www.tidymodels.org/start/case-study/]

Problem: Predict if a booking has children.

Formula: children ~ .

Model 1: glmnet(family = "binomial")

Model 2: ranger(classification = TRUE)

```
library("skimr")       #function skim() for descriptive statistics
library("GGally")      #function ggpairs() for data visualization
library("glmnet")      #function glmnet() to fit GLMs with penalized mL
library("ranger")      #function ranger() to random forest models
library("vip")         #function vip() to plot "vars importance" for ML models
library("tidymodels") #collection of packages for modelling
library("tidyverse")  #collection of packages for data analysis
```

3.1. LOOK AT DATA

```
#loading data and making changes
hotels <-
  read_csv("https://tidymodels.org/start/case-study/hotels.csv") |>
  mutate(across(where(is.character), as.factor))

#inspect data
glimpse(hotels)

#descriptive statistics
skim(hotels)
```

```
#data visualization
set.seed(123)
hotels |>
  slice_sample(n=1000) |>
  select(-c(
    required_car_parking_spaces, customer_type,  #too unbalanced
    meal:assigned_room_type,                     #too many levels
    stays_in_weekend_nights, is_repeated_guest:days_in_waiting_list,
      total_of_special_requests,                 #too many Os
    arrival_date                                 #data
    )
  ) |>
  select(children, where(is.factor), where(is.numeric)) |>
  ggpairs(progress = FALSE)
```

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

3.2. SPLIT DATA

```
set.seed(123)
splits      <- initial_split(hotels, strata = children, prop = 0.75)
```

```
hotel_other <- training(splits)
hotel_test  <- testing(splits)

hotel_other |>
  count(children) |>
  mutate(prop = n/sum(n))

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

set.seed(234)
val_set <- validation_split(hotel_other, strata = children, prop = 0.80)
```

3.3. CREATE WORKFLOW AND FIT MODEL 1

```
#build model
?logistic_reg
```

```
lr_mod <-
  logistic_reg(
    penalty = tune(), #total amount of regularization
    mixture = 1       #choose ridge (0), lasso (1), or elastic models (0 to 1)
  ) |>
  set_engine("glmnet")
lr_mod

#create recipe
holidays <- c("AllSouls", "AshWednesday", "ChristmasEve", "Easter",
              "ChristmasDay", "GoodFriday", "NewYearsDay", "PalmSunday")
lr_recipe <-
  recipe(children ~ ., data = hotel_other) |>
  step_date(arrival_date) |>
  step_holiday(arrival_date, holidays = holidays) |>
  step_rm(arrival_date) |>
  step_dummy(all_nominal_predictors()) |>
  step_zv(all_predictors()) |>
  step_normalize(all_predictors())

#create workflow
lr_workflow <-
  workflow() |>
  add_model(lr_mod) |>
  add_recipe(lr_recipe)
lr_workflow

#tune and fit model
lr_reg_grid <- tibble(penalty = 10^seq(-4, -2, length.out = 20))
lr_reg_grid

t1 <- Sys.time()
lr_res <-
  lr_workflow |>
```

```
  tune_grid(
    val_set,
    grid = lr_reg_grid,
    control = control_grid(save_pred = TRUE),
    metrics = metric_set(roc_auc)
  )
Sys.time() - t1

#select best parameters
lr_res |>
  collect_metrics() |>
  ggplot(aes(x = penalty, y = mean)) +
  geom_point() +
  geom_line() +
  ylab("Area under the ROC Curve") +
  scale_x_log10(labels = scales::label_number())

lr_res |>
  show_best(metric = "roc_auc", n = 5) |>
  arrange(penalty)

lr_best <-
  lr_res |>
  select_best(metric = "roc_auc")
lr_best

#evaluate model using ROC curve
lr_pred <- lr_res |>
  collect_predictions(parameters = lr_best)

lr_auc <-
  lr_pred |>
  roc_curve(children, .pred_children) |>
  mutate(model = "Logistic Regression")
autoplot(lr_auc)

lr_pred |>
  roc_auc(children, .pred_children)
```

3.4. CREATE WORKFLOW AND FIT MODEL 2

```
#build model
?rand_forest
```

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

rf_mod <-
  rand_forest(
    mtry = tune(),  #number of predictors for each split
    min_n = tune(), #minimum number of data points in a node to stop the split
    trees = 1000    #number of trees contained in the ensemble
  ) |>
  set_engine("ranger", num.threads = cores) |>
```

```
  set_mode("classification")
rf_mod

#create recipe
holidays <- c("AllSouls", "AshWednesday", "ChristmasEve", "Easter",
              "ChristmasDay", "GoodFriday", "NewYearsDay", "PalmSunday")
rf_recipe <-
  recipe(children ~ ., data = hotel_other) |>
  step_date(arrival_date) |>
  step_holiday(arrival_date, holidays = holidays) |>
  step_rm(arrival_date)

#create workflow
rf_workflow <-
  workflow() |>
  add_model(rf_mod) |>
  add_recipe(rf_recipe)
rf_workflow

#tune and fit model
set.seed(345)
t1 <- Sys.time()
rf_res <-
  rf_workflow |>
  tune_grid(
    val_set,
    grid = 10,
    control = control_grid(save_pred = TRUE),
    metrics = metric_set(roc_auc)
  )
Sys.time() - t1

#select best parameters
autoplot(rf_res)

rf_res |>
  show_best(metric = "roc_auc", n = 5)

rf_best <-
  rf_res |>
  select_best(metric = "roc_auc")
rf_best

#evaluate model using ROC curve
rf_pred <-
  rf_res |>
  collect_predictions(parameters = rf_best)

rf_auc <-
  rf_pred |>
  roc_curve(children, .pred_children) |>
  mutate(model = "Random Forest")
autoplot(rf_auc)
```

```
rf_pred |> roc_auc(children, .pred_children)
```

3.5. EVALUATE LAST MODEL

```
#compare best models
bind_rows(rf_auc, lr_auc) |>
  ggplot(aes(x = 1 - specificity, y = sensitivity, col = model)) +
  geom_path(lwd = 1.0, alpha = 0.8) +
  geom_abline(lty = 3) +
  coord_equal() +
  theme_bw() +
  theme(legend.position = "bottom")

#build last model
last_rf_mod <-
  rand_forest(mtry = rf_best$mtry, min_n = rf_best$min_n, trees = 1000) |>
  set_engine("ranger", num.threads = cores, importance = "impurity") |>
  set_mode("classification")

#create last workflow
last_rf_workflow <-
  rf_workflow |>
  update_model(last_rf_mod)

#fit last model
set.seed(345)
t1 <- Sys.time()
last_rf_fit <-
  last_rf_workflow |>
  last_fit(split = splits)
Sys.time() - t1

#evaluate model using ROC curve
last_rf_fit |>
  collect_predictions() |>
  roc_curve(children, .pred_children) |>
  autoplot()

last_rf_fit |>
  collect_metrics()

#extract results
last_rf_fit |>
  extract_fit_parsnip() |>
  vip(num_features = 20)
```