Building better training data to predict children in hotel bookings

Suggested answers

APPLICATION EXERCISE ANSWERS

MODIFIED
September 12, 2024

Your Turn 1

Unscramble! You have all the steps from our `knn_rec` - your challenge is to *unscramble* them into the right order!

Save the result as `knn_rec`

```
step_normalize(all_numeric())

recipe(children ~ ., data = hotels)

step_rm(arrival_date)

step_date(arrival_date)

step_downsample(children)

step_holiday(arrival_date, holidays = holidays)

step_dummy(all_nominal_predictors())

step_zv(all_predictors())
```

Answer:

```
knn_rec <- recipe(children ~ ., data = hotels) |>
  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_numeric()) |>
  step_downsample(children)
knn_rec
```

```
── Recipe ──────────────────────────────────────────────────────────────────────


── Inputs

Number of variables by role

outcome:    1
predictor: 21


── Operations

• Date features from: arrival_date

• Holiday features from: arrival_date

• Variables removed: arrival_date

• Dummy variables from: all_nominal_predictors()

• Zero variance filter on: all_predictors()

• Centering and scaling for: all_numeric()

• Down-sampling based on: children
```

Your Turn 2

Fill in the blanks to make a workflow that combines `knn_rec` and with `knn_mod`.

```
knn_wf <- ______ |>
  ______(knn_rec) |>
  ______(knn_mod)
knn_wf
```

Answer:

```
knn_wf <- workflow() |>
  add_recipe(knn_rec) |>
  add_model(knn_mod)
knn_wf
```

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

── Preprocessor ────────────────────────────────────────────────────────────────
```

```
7 Recipe Steps

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

── Model ───────────────────────────────────────────────────────────────────────
K-Nearest Neighbor Model Specification (classification)

Computational engine: kknn
```

Your Turn 3

Edit the code chunk below to fit the entire `knn_wflow` instead of just `knn_mod`.

```
set.seed(100)
knn_mod |>
  fit_resamples(children ~ .,
                resamples = hotels_folds,
                # print progress of model fitting
                control = control_resamples(verbose = TRUE)) |>
  collect_metrics()
```

Answer:

```
set.seed(100)
knn_wf |>
  fit_resamples(resamples = hotels_folds,
                control = control_resamples(verbose = TRUE)) |>
  collect_metrics()
```

```
i Fold01: preprocessor 1/1

✓ Fold01: preprocessor 1/1

i Fold01: preprocessor 1/1, model 1/1

✓ Fold01: preprocessor 1/1, model 1/1

i Fold01: preprocessor 1/1, model 1/1 (extracts)

i Fold01: preprocessor 1/1, model 1/1 (predictions)

i Fold02: preprocessor 1/1
```

```
✓ Fold02: preprocessor 1/1

i Fold02: preprocessor 1/1, model 1/1

✓ Fold02: preprocessor 1/1, model 1/1

i Fold02: preprocessor 1/1, model 1/1 (extracts)

i Fold02: preprocessor 1/1, model 1/1 (predictions)

i Fold03: preprocessor 1/1

✓ Fold03: preprocessor 1/1

i Fold03: preprocessor 1/1, model 1/1

✓ Fold03: preprocessor 1/1, model 1/1

i Fold03: preprocessor 1/1, model 1/1 (extracts)

i Fold03: preprocessor 1/1, model 1/1 (predictions)

i Fold04: preprocessor 1/1

✓ Fold04: preprocessor 1/1

i Fold04: preprocessor 1/1, model 1/1

✓ Fold04: preprocessor 1/1, model 1/1

i Fold04: preprocessor 1/1, model 1/1 (extracts)

i Fold04: preprocessor 1/1, model 1/1 (predictions)

i Fold05: preprocessor 1/1

✓ Fold05: preprocessor 1/1

i Fold05: preprocessor 1/1, model 1/1

✓ Fold05: preprocessor 1/1, model 1/1

i Fold05: preprocessor 1/1, model 1/1 (extracts)

i Fold05: preprocessor 1/1, model 1/1 (predictions)

i Fold06: preprocessor 1/1

✓ Fold06: preprocessor 1/1

i Fold06: preprocessor 1/1, model 1/1
```

```
✓ Fold06: preprocessor 1/1, model 1/1

i Fold06: preprocessor 1/1, model 1/1 (extracts)

i Fold06: preprocessor 1/1, model 1/1 (predictions)

i Fold07: preprocessor 1/1

✓ Fold07: preprocessor 1/1

i Fold07: preprocessor 1/1, model 1/1

✓ Fold07: preprocessor 1/1, model 1/1

i Fold07: preprocessor 1/1, model 1/1 (extracts)

i Fold07: preprocessor 1/1, model 1/1 (predictions)

i Fold08: preprocessor 1/1

✓ Fold08: preprocessor 1/1

i Fold08: preprocessor 1/1, model 1/1

✓ Fold08: preprocessor 1/1, model 1/1

i Fold08: preprocessor 1/1, model 1/1 (extracts)

i Fold08: preprocessor 1/1, model 1/1 (predictions)

i Fold09: preprocessor 1/1

✓ Fold09: preprocessor 1/1

i Fold09: preprocessor 1/1, model 1/1

✓ Fold09: preprocessor 1/1, model 1/1

i Fold09: preprocessor 1/1, model 1/1 (extracts)

i Fold09: preprocessor 1/1, model 1/1 (predictions)

i Fold10: preprocessor 1/1

✓ Fold10: preprocessor 1/1

i Fold10: preprocessor 1/1, model 1/1

✓ Fold10: preprocessor 1/1, model 1/1

i Fold10: preprocessor 1/1, model 1/1 (extracts)
```

```
i Fold10: preprocessor 1/1, model 1/1 (predictions)

# A tibble: 3 × 6
  .metric     .estimator  mean     n std_err .config
  <chr>       <chr>      <dbl> <int>   <dbl> <chr>
1 accuracy    binary     0.739    10 0.00192 Preprocessor1_Model1
2 brier_class binary     0.175    10 0.00143 Preprocessor1_Model1
3 roc_auc     binary     0.830    10 0.00352 Preprocessor1_Model1
```

Your Turn 4

Turns out, the same `knn_rec` recipe can also be used to fit a penalized logistic regression model using the lasso. Let's try it out!

```
plr_mod <- logistic_reg(penalty = .01, mixture = 1) |>
  set_engine("glmnet") |>
  set_mode("classification")

plr_mod |>
  translate()
```

```
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = 0.01
  mixture = 1

Computational engine: glmnet

Model fit template:
glmnet::glmnet(x = missing_arg(), y = missing_arg(), weights = missing_arg(),
    alpha = 1, family = "binomial")
```

Answer:

```
glmnet_wf <- knn_wf |>
  update_model(plr_mod)

glmnet_wf |>
  fit_resamples(resamples = hotels_folds) |>
  collect_metrics()
```

```
# A tibble: 3 × 6
  .metric     .estimator  mean     n  std_err .config
  <chr>       <chr>      <dbl> <int>    <dbl> <chr>
1 accuracy    binary     0.828    10 0.00215  Preprocessor1_Model1
2 brier_class binary     0.139    10 0.000876 Preprocessor1_Model1
3 roc_auc     binary     0.873    10 0.00209  Preprocessor1_Model1
```

Acknowledgments

- Materials derived from Tidymodels, Virtually: An Introduction to Machine Learning with Tidymodels by Allison Hill.
- Dataset and some modeling steps derived from A predictive modeling case study and licensed under a Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA) License.

Session information