Google Polls

Lavanya N

2/24/2021

In this example, a penalised logistic regression model is trained and tested using data from Google Polls to predict whether an individual wants Obama to win or another candidate to win.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
GooglePoll <- readRDS("GooglePoll.rds")
glimpse(GooglePoll)
```

```
## Rows: 9,483
## Columns: 9
## $ Time_UTC      <dttm> 2012-11-06 10:02:28, 2012-11-06 10:02:28, 2012-11-06...
## $ Gender        <fct> Female, Female, Male, NA, Female, Female, Male, Femal...
## $ Age           <ord> 55-64, 55-64, 35-44, NA, 25-34, 55-64, 25-34, 55-64, ...
## $ Urban_Density <fct> Suburban, Urban, Suburban, Suburban, Urban, Suburban,...
## $ Income        <ord> "25,000-49,999", "25,000-49,999", "0-24,999", "25,000...
## $ Weight        <dbl> 0.4507708, 0.5122865, 1.5133354, NA, 0.9362303, 0.815...
## $ WantToWin     <fct> "Barack Obama / Joe Biden, the Democrats", NA, "Barac...
## $ ThinkWillWin  <fct> "Barack Obama / Joe Biden, the Democrats", "Third par...
## $ Region        <fct> SOUTH, WEST, WEST, SOUTH, MIDWEST, NORTHEAST, WEST, N...
```

```
set.seed(20210220)
```

It is more convenient to rename the outcomes as the level names are long.

```
GooglePoll <- mutate(GooglePoll, Obama = as.factor(WantToWin == levels(WantToWin)[1])) %>% select(-Want'

library(tidymodels)
```

```
## -- Attaching packages --------------------------------- tidymodels 0.1.2.9000 --

## v broom     0.7.2      v recipes   0.1.15
## v dials     0.0.9      v rsample   0.0.8
## v ggplot2   3.3.2      v tibble    3.0.4
## v infer     0.5.3      v tidyr     1.1.2
## v modeldata 0.1.0      v tune      0.1.2
## v parsnip   0.1.4      v workflows 0.2.1
## v purrr     0.3.4      v yardstick 0.0.7

## Warning: package 'broom' was built under R version 4.0.3

## Warning: package 'dials' was built under R version 4.0.3

## Warning: package 'infer' was built under R version 4.0.3

## Warning: package 'modeldata' was built under R version 4.0.3

## Warning: package 'parsnip' was built under R version 4.0.3

## Warning: package 'recipes' was built under R version 4.0.3

## Warning: package 'rsample' was built under R version 4.0.3

## Warning: package 'tibble' was built under R version 4.0.3

## Warning: package 'tune' was built under R version 4.0.3

## Warning: package 'workflows' was built under R version 4.0.3

## Warning: package 'yardstick' was built under R version 4.0.3

## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
```

```
GP_split <- initial_split(GooglePoll, prob = 0.75, strata = Obama)
GP_train <- training(GP_split)
GP_test <- testing(GP_split)
```

```
if (.Platform$OS.type == "windows") {  doParallel::registerDoParallel()} else doMC::registerDoMC(parall
```

Since variables are being compared to one another but have different units, they should be standardised using step_center and step_scale. The penalty is also pre-specified to have a value of 0.001. Missing values are also dealth with.

```
logit_recipe <- recipe(Obama ~ . , data = GP_train) %>%
  step_rm(Time_UTC) %>% step_rm(Weight) %>% # these 2 variables are not meaningful
  step_naomit(Obama) %>%
  step_knnimpute(all_predictors()) %>% # deal with missing values
  step_dummy(all_nominal() & all_predictors()) %>% # convert everything except the outcome to dummy var
  step_center(all_predictors()) %>% step_scale(all_predictors()) %>% prep
```

Instead of specifying the tuning parameters, grid search can be employed to find the optimal parameters. This performs cross-validation to see which value of the tuning parameter yields the best predictions outside the data that are used to obtain the coefficients. In this, the optimal parameter is chosen from 50 possible options.

```
GP_rs <- bootstraps(bake(logit_recipe, new_data = GP_train), times = 50)
tune_spec <- logistic_reg(penalty = tune(), mixture = tune()) %>%
  set_engine("glmnet")
wf <- workflow() %>% add_recipe(logit_recipe) %>%
  remove_formula() %>% add_formula(Obama ~ (.)^2)
```

```
## Warning: The workflow has no formula preprocessor to remove.
```

```
my_grid <- grid_regular(penalty(), mixture(), levels = 10)
results <- tune_grid(wf %>% add_model(tune_spec), resamples = GP_rs, grid = my_grid)
```

The value of the tuning parameter is selected such that the resulting regression model has the highest accuracy.

```
most_accurate <- results %>% select_best("accuracy")
final <- finalize_workflow(wf %>% add_model(tune_spec), most_accurate)
final_fit <- fit(final, data = bake(logit_recipe, new_data = GP_train))
baked <- bake(logit_recipe, new_data = GP_test)
```

The resulting logistic regression model classifies 682 observations correctly as not wanting Obama to win and 972 observations correctly as wanting Obama to win. However, the model classifies 66 observations incorrectly as not wanting Obama to win and 253 observations incorrectly as wanting Obama to win.

```
bind_cols(Obama = baked$Obama, predict(final_fit, new_data = baked)) %>%
  conf_mat(truth = Obama, estimate = .pred_class)
```

```
##           Truth
## Prediction FALSE TRUE
##      FALSE   682   66
##      TRUE    253  972
```

The resulting logistic regression model has an accuracy of 0.838.

```
bind_cols(Obama = baked$Obama, predict(final_fit, new_data = baked)) %>%
  accuracy(truth = Obama, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.838
```