President_election_prediction

Yifan Jiang

Summer 2023

```
# Load the library
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.1.0 --
```

```
## v broom        1.0.5     v recipes      1.0.7
## v dials        1.2.0     v rsample      1.1.1
## v dplyr        1.1.2     v tibble       3.2.1
## v ggplot2      3.4.3     v tidyr        1.3.0
## v infer        1.0.4     v tune         1.1.1
## v modeldata    1.2.0     v workflows    1.1.3
## v parsnip      1.1.1     v workflowsets 1.0.1
## v purrr        1.0.2     v yardstick    1.2.0
```

```
## -- 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()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.4
## v lubridate 1.9.2     v stringr   1.5.0
```

```
## -- Conflicts ----------------------------------------- tidyverse_conflicts() --
## x readr::col_factor() masks scales::col_factor()
## x purrr::discard()    masks scales::discard()
## x dplyr::filter()     masks stats::filter()
## x stringr::fixed()    masks recipes::fixed()
## x dplyr::lag()        masks stats::lag()
## x readr::spec()       masks yardstick::spec()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
```

```
##
##     slice
```

```
library(stacks)

# Import data
# Remove the predictor "name" since it is only for reference
train <- read_csv("train.csv") %>%
  select(-name)
```

```
## Rows: 2331 Columns: 126
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (1): name
## dbl (125): id, percent_dem, total_votes, x0001e, x0002e, x0003e, x0005e, x00...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
  set.seed(1)

# split the train data further into 80% train set and 20% test set (so we can test the stack model by c
train_split <- initial_split(train, prop = 0.8)
train1 <- training(train_split)
test1 <- testing(train_split)
# Import test data
test <- read_csv("test.csv")
```

```
## Rows: 780 Columns: 124
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (124): id, total_votes, x0001e, x0002e, x0003e, x0005e, x0006e, x0007e, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Fold the training data into a 10-fold cross-validation set. Stratify on `percent_dem`.
train_folds <- vfold_cv(train, v = 10, strata = percent_dem)

# The basic recipe of the models
basic_recipe <-recipe(percent_dem ~ id + x0001e + x0002e + x0003e +
x0022e + x0024e + x0034e + x0035e + x0037e + x0038e + x0039e + x0044e
+ x0052e + x0058e + x0059e + x0060e + x0061e + x0062e + x0057e +
x0071e + x0086e + c01_006e + c01_007e + c01_008e + c01_009e + c01_010e
+ c01_011e + c01_012e + c01_013e + c01_014e + c01_015e +
income_per_cap_2016 + income_per_cap_2017 + income_per_cap_2018 +
income_per_cap_2019 + income_per_cap_2020 + gdp_2016 + gdp_2017 +
gdp_2018 + gdp_2019 + gdp_2020 + x2013_code, data = train) %>%
# convert some columns as ratio and mutate them as new columns
  step_mutate(
  male_perc = x0002e / x0001e,
  female_perc = x0003e / x0001e,
  adult_pop = x0022e - x0024e,
  elder_pop = x0024e,
  white_race_perc = x0037e / x0001e,
  black_race_perc = x0038e / x0001e,
```

```
  american_race_perc = x0039e / x0001e,
  asian_race_perc = x0044e / x0001e,
  hawaiian_race_perc = x0052e / x0001e,
  other_race_perc = x0057e / x0001e,
  hisp_race_perc = x0071e / x0001e,
  mix_wb_perc = x0059e / x0001e,
  mix_wam_perc = x0060e / x0001e,
  mix_wai_perc = x0061e / x0001e,
  mix_bam_perc = x0062e / x0001e,
  high_school = c01_014e - c01_015e,
  bachelor_over = c01_015e,
  high_school_under = c01_006e - c01_014e) %>%
# remove the original columns and the id column
  step_rm(
  x0001e, x0002e, x0003e, x0024e, x0022e, x0034e, x0035e, x0037e ,
  x0038e , x0039e , x0044e , x0052e, x0057e, x0071e , x0059e , x0058e,
  x0060e , x0061e , x0062e , c01_014e , c01_006e, c01_015e) %>%
  step_rm(id) %>%
# remove rows with missing values in percent_dem
# fill in missing values using bagged tree models for predictors
  step_naomit(percent_dem) %>%
  step_impute_bag(all_predictors()) %>%
# remove predictors with low variance
  step_nzv(all_predictors()) %>%
# create dummy variables for categorical variables
  step_dummy(all_nominal()) %>%
# standardize numeric variables
  step_normalize(all_numeric_predictors())
# recipe for spline model
  spline_rec <- basic_recipe %>%
#removing high correlated variables
  step_corr(all_numeric(), -all_outcomes()) %>%
# creating linear combination of numeric features
  step_lincomb(all_numeric(), -all_outcomes()) %>%
# removing categorical variables
  step_rm(all_nominal()) %>%
# box-cox transformation and Yeo Johnson transformation on predictors
  step_bs(all_predictors()) %>%
  step_YeoJohnson(all_predictors())
# recipe for sum model
  svm_rec <- basic_recipe %>%
  step_corr(all_predictors())
# create models
# total of 5 models: spline, random forest, boost tree, sum, knn
# for random forest, boost tree, sum, knn models, we will tune their hyperparameters
spline_model <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

random_forest_model <- rand_forest(min_n = tune(), trees = tune()) %>%
  set_mode("regression") %>%
  set_engine("ranger")
```

```
xgboost_model <- boost_tree(learn_rate = tune(), trees = tune(),
  tree_depth = tune()) %>%
  set_mode("regression") %>%
  set_engine("xgboost")

svm_model <-
  svm_rbf(
  cost = tune("cost"),
  rbf_sigma = tune("sigma")
  ) %>%
    set_engine("kernlab") %>%
    set_mode("regression")

knn_model <- nearest_neighbor(
  mode = "regression",
  neighbors = tune("k")
  ) %>%

set_engine("kknn")
# Create workflows for all the models
spline_wf <- workflow() %>%
  add_model(spline_model) %>%
  add_recipe(spline_rec)
random_forest_wf <- workflow() %>%
  add_model(random_forest_model) %>%
  add_recipe(basic_recipe)
xgboost_wf <- workflow() %>%
  add_model(xgboost_model) %>%
  add_recipe(basic_recipe)
svm_wflow <-
  workflow() %>%
  add_model(svm_model) %>%
  add_recipe(svm_rec)
knn_wflow <-
  workflow() %>%
  add_model(knn_model) %>%
  add_recipe(basic_recipe)
# set up parameter grid for those four models
# random_forest and boost_tree: for each hyperparameters, explore three different values
ranforest_grid <- parameters(random_forest_model) %>%
  grid_regular(levels = 3)
```

```
## Warning: `parameters.model_spec()` was deprecated in tune 0.1.6.9003.
## i Please use `hardhat::extract_parameter_set_dials()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
xgboost_grid <- parameters(xgboost_model) %>%
  grid_regular(levels = 3)
# svm and knn: for each hyperparameters, explore six different values
svm_grid <- grid_regular(parameters(svm_model), levels = 6, filter =c(cost <= 0.05))
knn_grid <- grid_regular(parameters(knn_model), levels = 6)
# fit
```

```
# save predictions and workflow objects during the tuning process
ctrl_grid <- control_grid(save_pred = TRUE, save_workflow = TRUE)
ctrl_res <- control_grid(save_pred = TRUE, save_workflow = TRUE)
metric <- metric_set(rmse) # for evaluating the performance
# Tuning process
spline_res <- spline_wf %>%
  fit_resamples(
  resamples = train_folds,
  metrics = metric,
  control = ctrl_res
  )
```

```
## > A | warning: some 'x' values beyond boundary knots may cause ill-conditioned bases
## There were issues with some computations   A: x1There were issues with some computations   A: x2There
```

```
random_forest_res <- random_forest_wf %>%
  tune_grid(
  resamples = train_folds,
  grid = ranforest_grid,
  metrics = metric,
  control = ctrl_grid
  )
xgboost_res <- tune_grid(
  xgboost_wf,
  resamples = train_folds,
  grid = xgboost_grid,
  metrics = metric,
  control = ctrl_grid
  )
svm_tune <- tune_grid(
  object = svm_wflow,
  resamples = train_folds,
  grid = svm_grid,
  control = ctrl_grid,
  metrics = metric
  )
knn_tune <- tune_grid(
  object = knn_wflow,
  resamples = train_folds,
  grid = knn_grid,
  control = ctrl_grid,
  metrics = metric
  )
# stacks multiple predictive models (the five we have)
ames_stack <- stacks() %>%
  add_candidates(spline_res) %>%
  add_candidates(random_forest_res) %>%
  add_candidates(xgboost_res) %>%
  add_candidates(svm_tune) %>%
  add_candidates(knn_tune)
```

```
## Warning: The inputted `candidates` argument `spline_res` generated notes during
## tuning/resampling. Model stacking may fail due to these issues; see
## `collect_notes()` (`?tune::collect_notes()`) if so.
```

```
# combining predictions from models in stacks
# then fit the blended predictions to create an ensemble model
ames_stack <- ames_stack %>%
blend_predictions() %>%
fit_members()
# make prediction on the test data
# bind the predicted values to the id column in the test data
prediction <- ames_stack %>%
predict(new_data = test) %>%
bind_cols(id = test$id)
# exporting prediction as csv file for Kaggle submission
prediction %>%
rename(percent_dem = .pred) %>%
write_csv("predictions_last2.csv")
```