knn

Eric Chu

2024-07-31

Set-Up

If you donwant to knit the PDF but don't want to run the code chunks, set eval = FALSE.

```
knitr::opts_chunk$set(echo = TRUE, eval = FALSE)

set.seed(161)
```

Libraries

```
library(tidyverse)
library(tidymodels)
```

Import Data

```
train <- read_csv('../Data/train_class.csv')
test <- read_csv('../Data/test_class.csv')
```

Examine data

```
# Check number of rows in each dataset
nrow(train)
nrow(test)

# Examine the distribution of the data
train %>%
    summarize(min_total_votes = min(total_votes),
              max_total_votes = max(total_votes),
              mean_total_votes = mean(total_votes),
              sd_total_votes = sd(total_votes))

test %>%
    summarize(min_total_votes = min(total_votes),
              max_total_votes = max(total_votes),
              mean_total_votes = mean(total_votes),
              sd_total_votes = sd(total_votes))


# Create global metrics set
winner_metrics <- metric_set(roc_auc, sens, spec, accuracy)
```

Modify Data

```
# Create copies that will be unaffected by modifications
train_copy <- train
test_copy <- test

# Convert x2013_code into factor
train <- train %>%
  mutate(across('x2013_code', as.factor)) %>%
  select(-'name')

test <- test %>%
  mutate(across('x2013_code', as.factor))
```

Validate Data Using Cross-Folds

```
winner_folds <- vfold_cv(train, v = 10,
                         strata = 'winner')
```

Recipe & Workflow

Set Engines

```
knn_model <- nearest_neighbor() %>%
  set_engine("kknn") %>%
  set_mode("classification")
```

TODO: Create recipe that encodes some features into factors (if necessary), log transforms features, as well as drop unnecessary features ## Create Recipe

```
# This basic recipe log transforms data, unselects id & name, imputes missing data, normalizes data, en
knn_basic_recipe <- recipe(winner ~ ., data = train) %>%
  step_log('total_votes', offset = 1) %>%
  step_rm('id') %>% # Don't use these as predictors
  step_dummy('x2013_code') %>% # Make sure to normalize after encoding
  step_impute_mean(all_numeric()) %>%
  step_normalize(all_numeric()) %>% # if we don't drop id then don't forget to avoid normalizing it wit
  step_interact(terms = ~x0002e:x0003e) %>%
  step_interact(terms = ~ x0005e:x0006e:x0007e:x0008e:x0009e:x0010e:x0011e:x0012e:x0013e:x0014e:x0015e:
  step_interact(terms = ~ x0019e:x0020e:x0021e:x0022e:x0023e:x0024e) %>%
  step_interact(terms = ~ x0025e:x0026e:x0027e:x0029e:x0030e:x0031e) %>%
  step_interact(terms = ~ x0034e:x0035e:x0036e:x0037e:x0038e:x0039e:x0040e:x0041e:x0042e:x0043e:x0044e:
  step_interact(terms = ~ x0058e:x0059e:x0060e:x0061e:x0062e:x0064e:x0065e:x0066e:x0067e:x0068e:x0069e)
  step_interact(terms = ~ x0071e:x0072e:x0073e:x0074e:x0075e) %>%
  step_interact(terms = ~ x0076e:x0077e:x0078e:x0079e:x0080e:x0081e:x0082e:x0083e:x0084e:x0085e) %>%
  step_interact(terms = ~ x0087e:x0088e:x0089e) %>%
  step_interact(terms = ~ c01_001e:c01_002e:c01_003e:c01_004e:c01_005e:c01_006e:c01_007e:c01_008e:c01_0
  step_interact(terms = ~ income_per_cap_2016:income_per_cap_2017:income_per_cap_2018:income_per_cap_20
  step_interact(terms = ~ gdp_2016:gdp_2017:gdp_2018:gdp_2019:gdp_2020)
```

Create Workflow

```
# Basic recipe workflow
knn_basic_wkfl <- workflow() %>%
  add_recipe(knn_basic_recipe) %>%
  add_model(knn_model)
```

Tuning

Create general tuning model

```
# Set up tuning model

# Should we tune other parameters for nearest_neighbor()?
tune_model <- nearest_neighbor(neighbors = tune()) %>%
  set_engine("kknn") %>%
  set_mode("classification")

# Create control grids
con_grid <- control_grid(save_pred = TRUE, save_workflow = TRUE, verbose = TRUE)
```

Create tuning workflow for basic recipe

```
tune_basic_wkfl <- workflow() %>%
  add_recipe(knn_basic_recipe) %>%
  add_model(tune_model)
```

TODO: Create and add control grid with parameters needed for the stack ## Tune the basic model

```
# Tune the basic model with resampled data (this will take a couple of minutes)
tune_basic_rs <- tune_grid(tune_basic_wkfl,
                      resamples = winner_folds,
                      control = con_grid,
                      grid = 20)
```

Acquire optimal k-value

Basic knn model

It appears $n = 13$ is the optimal value for accuracy.

```
tune_basic_results <- tune_basic_rs %>%
  collect_metrics() %>%
  filter(.metric == "accuracy") %>%
  arrange(desc(mean))  # Choose highest accuracy

print(tune_basic_results)

basic_best_k <- select_best(tune_basic_rs, metric = "accuracy")
```

Finalize Model

Basic knn

```
final_basic <- finalize_model(tune_model, basic_best_k)

# Create final  Workflow
final_basic_wkfl <- workflow() %>%
  add_recipe(knn_basic_recipe) %>%
  add_model(final_basic)
```

Fitting Data

Each fit will take a minute or two. ## Fitting the basic wokrflow to resamples

```
knn_basic_rs <- knn_basic_wkfl %>%
  fit_resamples(resamples = winner_folds, metrics = winner_metrics)
```

Fitting the tuned basic workflow to resamples

```
knn_basic_tuned_rs <- final_basic_wkfl %>%
  fit_resamples(resamples = winner_folds, metrics = winner_metrics)
```

Model Assessment

Collect Metrics

```
# Metrics of basic resamples
#knn_basic_rs %>%
#  collect_metrics()

# Metrics of tuned basic model
knn_basic_tuned_rs %>%
  collect_metrics
```

Fit the basic workflow to all of training data

```
knn_basic_fit <- knn_basic_wkfl %>%
  fit(train)

print(knn_basic_fit)
```

Do the same but with the tuned workflow

```
knn_basic_tuned_fit <- final_basic_wkfl %>%
  fit(train)

print(knn_basic_tuned_fit)
```

Get Predictions

Basic knn

```
basic_predictions <- predict(knn_basic_tuned_fit, new_data = test)

# Create new testing data that drops the same features training did
basic_results <- test %>%
  select(id) %>%
  bind_cols(basic_predictions)

write_csv(basic_results, 'basic_knn_tuned_predictions.csv')
```

Create predictions with tuned parameters

```
basic_predictions <- predict(knn_basic_fit, new_data = test)

# Create new testing data that drops the same features training did
basic_results <- test %>%
  select(id) %>%
  bind_cols(basic_predictions)

write_csv(basic_results, 'basic_knn_predictions.csv')
```

Potential Improvements

- For starters, we can tune k as well as increase the number of v folds.
  - While I did tune the number of neighbors, there appears to be other parameters we can tune, type `?nearest_neighbor()` in the terminal to see more.
- There are more ways we can deal with interaction effects between age categories (for instance ages per gender amy interact with general age columns).
- There are probably several other features that should be dropped, especially ones that are essentially duplicates of other features. Lastly, we can probably log transform several other features or maybe even encode other ones.
- Finally, there could be better ways of dealing with incomplete data than imputing mean. We may be better off dropping those rows altogether.
- It also might help to create a confusion matrix especially for comparing model performances.