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.

Libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.2.0 --
## v broom        1.0.6     v rsample      1.2.1
## v dials        1.2.1     v tune         1.2.1
## v infer        1.0.7     v workflows    1.1.4
## v modeldata    1.4.0     v workflowsets 1.1.0
## v parsnip      1.2.1     v yardstick    1.3.1
## v recipes      1.1.0
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

Import Data

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

```
## Rows: 2331 Columns: 126
## -- Column specification --------------------------------------------------------
```

```
## Delimiter: ","
## chr   (2): winner, name
## 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.
```

```
test <- read_csv('../Data/test_class.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.
```

Examine data

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

```
## [1] 2331
```

```
nrow(test)
```

```
## [1] 780
```

```
# 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))
```

```
## # A tibble: 1 x 4
##   min_total_votes max_total_votes mean_total_votes sd_total_votes
##             <dbl>           <dbl>            <dbl>          <dbl>
## 1              66         2321486           48929.        129039.
```

```
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))
```

```
## # A tibble: 1 x 4
##   min_total_votes max_total_votes mean_total_votes sd_total_votes
##             <dbl>           <dbl>            <dbl>          <dbl>
## 1             302         4264365           56314.        200771.
```

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

Modify Data

```r
# 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

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

Recipe & Workflow

Set Engines

```r
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

```r
# 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

```r
knn_basic_wkfl <- workflow() %>%
  add_recipe(knn_basic_recipe) %>%
```

```
  add_model(knn_model)
```

Fitting Data

Fitting the basic wokrflow to resamples

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

Model Assessment

Collect Metrics

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

```
## # A tibble: 4 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.855    10 0.00721 Preprocessor1_Model1
## 2 roc_auc  binary     0.811    10 0.0118  Preprocessor1_Model1
## 3 sens     binary     0.504    10 0.0242  Preprocessor1_Model1
## 4 spec     binary     0.925    10 0.00632 Preprocessor1_Model1
```

Fit the basic workflow to all of training data

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

print(knn_basic_fit)
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: nearest_neighbor()
##
## -- Preprocessor ----------------------------------------------------------------
## 17 Recipe Steps
##
## * step_log()
## * step_rm()
## * step_dummy()
## * step_impute_mean()
## * step_normalize()
## * step_interact()
## * step_interact()
## * step_interact()
## * step_interact()
## * step_interact()
## * ...
## * and 7 more steps.
##
```

```
## -- Model ----------------------------------------------------------------------
##
## Call:
## kknn::train.kknn(formula = ..y ~ ., data = data, ks = min_rows(5,     data, 5))
##
## Type of response variable: nominal
## Minimal misclassification: 0.1475761
## Best kernel: optimal
## Best k: 5
```

Get Predictions

Basic knn

```
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. Furthermore, 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.