Lab 05: Candy Competition

Recipes + Model comparison

Stats is 'Fun': Dav King, Thomas Barker, Luke Thomas, Harry Liu

2022-10-25

Setup

Load packages and data:

```r
library(tidyverse)
library(tidymodels)
library(fivethirtyeight)
library(knitr)
```

[Select this page for the "Workflow & formatting" in Gradescope.]

Exercises

Exercise 1

```r
set.seed(1)
div <- initial_split(candy_rankings, prop = 0.8)
candy_train <- training(div)
candy_test <- testing(div)
```

Exercise 2

```
# This line of code creates a new recipe, with all other variables from
# candy_train predicting winpercent in some unspecified regression model. It
# stores this recipe in a variable named candy_rec.
candy_rec <- recipe(winpercent ~ ., data = candy_train) |>
# This line of code updates the role of competitorname from a predictor to an
# identification variable - it remains in our data as a way to identify the
# candy bar, but this term no longer predicts an individual observation's
# winpercent.
  update_role(competitorname, new_role = "ID") |>
# This line of code takes a continuous variable sugarpercent and turns it into
# a factor variable with four levels: sugarpercent = [0, 0.25],
# (0.25, 0.5], (0.5, 0.75], and (0.75, 1].
  step_cut(sugarpercent, breaks = c(0, 0.25, 0.5, 0.75,1)) |>
# This line of code uses dplyr to transform pricepercent - taking the original
# value of pricepercent, multiplying it by 100, and re-assigning that variable
# back to pricepercent. In other words, it modifies pricepercent in place,
# multiplying it by 100.
  step_mutate(pricepercent = pricepercent * 100) |>
# This line of code takes all of our nominal predictors, i.e., every predictor
# that is not a continuous number but rather some sort of factor (or logical),
# and turns them into traditional dummy variables (replacing all instances of
# false with 0 and true with 1, creating an indicator function).
  step_dummy(all_nominal_predictors()) |>
# This line of code includes in our model the interaction effect between a
# candy bar's price percentile compared to the total dataset and whether or not
# the candy bar contains chocolate, by multiplying the two values together. This
# will be included in our final model output and show us the amount that the
# slope of the relationship between pricepercent and winpercent differs based on
# a candy bar's chocolate content.
  step_interact(terms =~ pricepercent:chocolate) |>
# This line of code removes all of the variables listed below from our recipe.
# They are no longer predictors in the model, nor are they contained anywhere
# in the recipe after this line.
  step_rm(fruity, caramel, peanutyalmondy, nougat, hard, bar, pluribus,
          crispedricewafer) |>
# This line of code removes any predictors that have only a single value (i.e.,
# zero variance). This leaves us with only variables that differ at some point
# in the data and thus could meaningfully predict anything.
  step_zv(all_predictors())
```

...

Exercise 3

```
candy_rec %>%
  prep() %>%
  bake(candy_train) %>%
  glimpse()
```

```
Rows: 68
Columns: 8
$ competitorname              <fct> Sour Patch Tricksters, Milky Way Simply C~
$ chocolate                   <lgl> FALSE, TRUE, TRUE, TRUE, TRUE, FALSE, FAL~
$ pricepercent                <dbl> 11.6, 86.0, 86.0, 65.1, 76.7, 51.1, 27.9,~
$ winpercent                  <dbl> 52.82595, 64.35334, 66.97173, 66.57458, 7~
$ sugarpercent_X.0.25.0.5.    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,~
$ sugarpercent_X.0.5.0.75.    <dbl> 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1,~
$ sugarpercent_X.0.75.1.      <dbl> 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,~
$ pricepercent_x_chocolateTRUE <dbl> 0.0, 86.0, 86.0, 65.1, 76.7, 0.0, 0.0, 0.~
```

There will be six terms in the model produced by this recipe.

…

Exercise 4

```r
candy_spec <- linear_reg() |>
  set_engine("lm")

candy_wflow <- workflow() |>
  add_model(candy_spec) |>
  add_recipe(candy_rec)

candy_fit <- candy_wflow |>
  fit(data = candy_train)

tidy(candy_fit)|>
  kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	43.763	3.621	12.085	0.000
chocolateTRUE	11.783	6.948	1.696	0.095
pricepercent	-0.026	0.088	-0.290	0.772
sugarpercent_X.0.25.0.5.	-2.890	4.679	-0.618	0.539
sugarpercent_X.0.5.0.75.	0.968	4.452	0.218	0.829
sugarpercent_X.0.75.1.	3.435	4.864	0.706	0.483
pricepercent_x_chocolateTRUE	0.112	0.118	0.949	0.346

Exercise 5

- Intercept: For a candy that doesn't contain chocolate that is in between the 0th and 25th percentile of sugar and at the 0th percentile of price, we expect the win percent to be, on average, 43.763%.
- For a candy with a sweetness score between the 75th and 100th percentile, we expect the win percent to be approximately 3.435% higher, on average, compared to candies with a sweetness score not between the 75th and 100th percentile, holding all else constant.
- The effect of price percent on win percent differs by 0.112% when the candy contains chocolate, on average, compared to when it doesn't contain chocolate, holding all else constant.

Exercise 6

```r
candy_spec2 <- linear_reg() |>
  set_engine("lm")

candy_rec2 <- recipe(winpercent ~ chocolate + pricepercent + crispedricewafer +
                       pluribus + sugarpercent, data = candy_train) |>
  step_cut(sugarpercent, breaks = c(0, 0.25, 0.5, 0.75,1)) |>
  step_mutate(pricepercent = pricepercent * 100) |>
  step_interact(terms =~ pricepercent:pluribus) |>
  step_zv(all_predictors())

candy_wflow2 <- workflow() |>
  add_model(candy_spec2) |>
  add_recipe(candy_rec2)

candy_fit2 <- candy_wflow2 |>
  fit(data = candy_train)

tidy(candy_fit2) |>
  kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	41.161	5.206	7.907	0.000
chocolateTRUE	16.125	3.679	4.383	0.000
pricepercent	0.070	0.095	0.735	0.465
crispedricewaferTRUE	2.575	5.390	0.478	0.635
pluribusTRUE	2.529	6.049	0.418	0.677
sugarpercent(0.25,0.5]	-2.928	4.734	-0.619	0.539
sugarpercent(0.5,0.75]	0.010	4.510	0.002	0.998
sugarpercent(0.75,1]	4.802	5.062	0.949	0.347
pricepercent_x_pluribusTRUE	-0.102	0.106	-0.967	0.338

Exercise 7

```
glance(candy_fit) |>
  select(r.squared, adj.r.squared, AIC, BIC)
```

```
# A tibble: 1 x 4
  r.squared adj.r.squared   AIC   BIC
      <dbl>         <dbl> <dbl> <dbl>
1     0.438         0.383  532.  549.
```

```
glance(candy_fit2) |>
  select(r.squared, adj.r.squared, AIC, BIC)
```

```
# A tibble: 1 x 4
  r.squared adj.r.squared   AIC   BIC
      <dbl>         <dbl> <dbl> <dbl>
1     0.447         0.372  535.  557.
```

Exercise 8

According to R-squared, we want to choose model 2. A higher R-squared indicates that a higher percentage of the variation in win percentage can be explained by the fitted model. As $0.438 < 0.447$, we want to choose the model with the higher R-squared, which is model 2.

According to adjusted R-squared, we want to choose model 1. As $0.382 > 0.372$, model 1 has a higher adjusted R-squared score, and we want to choose the model with the higher adjusted R-squared. Although before the adjustment, R-squared is bigger for model 2, we take into account the amount of variables used to fit the model in prevention of over-fitting (penalize model 2 for using too many variables).

According to AIC, we want to choose model 1. As $531.6 < 534.6$, model 1 has the smaller AIC (lower is better as that indicates lower prediction error, taking into account a penalty for models that include too many predictors).

According to BIC, we want to choose model 1. As $549.4 < 556.8$, model 1 has the smaller BIC (lower is better as that indicates lower prediction error, taking into account a fairly harsh penalty for models that include too many predictors).

Exercise 9

```
candy_test_pred <- predict(candy_fit, candy_test) |>
  bind_cols(candy_test)

rmse(candy_test_pred, truth = winpercent, estimate = .pred)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        11.5
```

```
candy_test_pred2 <- predict(candy_fit2, candy_test) |>
  bind_cols(candy_test)

rmse(candy_test_pred2, truth = winpercent, estimate = .pred)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        11.3
```

Model 1 has a RMSE of 11.517, whereas model 2 has a RMSE of 11.257. We would choose model 2 because it has a lower RMSE. A lower RMSE indicates lower error, on average, in predicting win percentage with the predictor variables. The predictive performance of model 2 on the testing data is better.

Exercise 10

- Which model do you choose after taking into account all the model evaluation statistics in Exercises 9 and 10? Briefly explain your response.
  - I would choose model 1 because it gives a better (higher) adjusted R-squared, along with a better (lower) AIC and BIC score. When choosing a model, the adjusted R-squared is more important than the R-squared because it gives a more comprehensive reflection on the model's ability to explain the variation in win percentage taking into account the complexity of the model (the number of variables used to make the prediction). Taking into account of the model complexity is important because it would help prevent choosing a model that is over-fitted. Also, the AIC and BIC scores also indicates that model 1 should be a better choice. With model 1 having a lower AIC/BIC score, AIC and BIC both reflect that model 1 fits the data better given into consideration the number of parameters each respective model take in.
- Use the model you selected to describe what generally makes a good candy, i.e., one with a high win percentage.
  - Generally, a good candy according to model 1 is a candy that contains chocolate, with high sugar content (above 0.5 percentile, preferably above 0.75 percentile), and a candy that is more pricey compared to the others in the set.