Model Fitting Part 2

Joe Martin

10/18/2021

Introduction

The goal of this section is to determine whether it is possible to predict a high temperature based on other symptoms a patient is displaying. High Temperature (Yes = 1, No = 0), is the outcome of interest. This is first tested with all present variables as predictors, then tested against the main predictor of interest, Sneeze.

```
# Write code that takes the data and splits it randomly into a train and test that, following for insta

set.seed(2)

df$high_temp <- ifelse(df$BodyTemp > 99, "high_temp","normal_range")
df$high_temp <- factor(df$high_temp)

df <- df %>% select(-BodyTemp)

# Use 70/30 split on data

data_split <- initial_split(df, prop = 7/10)

train_data <- training(data_split)
test_data <- testing(data_split)
```

Testing All Predictors

The following code generates recipes for the training and test sets of data, then builds a workflow to fit to a logistic model to all predictor variables.

```
# Next, following the example in the Create Recipes section of the Get Started tidymodels tutorial, cre

ht_rec <- recipe(high_temp ~ ., data = train_data)
ht_test <- recipe(high_temp ~ ., data = test_data)

#summary(ht_rec)
```

Set up logistic model and create workflow

```
lr_model <- logistic_reg() %>%
  set_engine("glm")

ht_workflow <-
```

```
  workflow() %>%
  add_model(lr_model) %>%
  add_recipe(ht_rec)

ht_workflow
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
##
## -- Model -----------------------------------------------------------------------
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

Show relationships between predictor variables and outcome

```
ht_fit <- ht_workflow %>%
  fit(data = train_data)

ht_fit %>%
  extract_fit_parsnip() %>%
  tidy()
```

```
## # A tibble: 38 x 5
##    term                 estimate std.error statistic p.value
##    <chr>                   <dbl>     <dbl>     <dbl>   <dbl>
##  1 (Intercept)            2.36       0.801     2.94  0.00325
##  2 SwollenLymphNodesYes   0.258      0.224     1.15  0.249
##  3 ChestCongestionYes    -0.0763     0.242    -0.315 0.752
##  4 ChillsSweatsYes       -0.186      0.325    -0.572 0.567
##  5 NasalCongestionYes     0.134      0.264     0.507 0.612
##  6 CoughYNYes            -0.928      0.651    -1.43  0.154
##  7 SneezeYes              0.743      0.243     3.06  0.00222
##  8 FatigueYes            -0.220      0.398    -0.551 0.582
##  9 SubjectiveFeverYes    -0.748      0.273    -2.74  0.00613
## 10 HeadacheYes           -0.350      0.322    -1.09  0.278
## # ... with 28 more rows
```

The following code predicts whether a patient has a high temperature based on the model built above.

```
# use predict to predict if patient has a high temperature
predict(ht_fit, test_data)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

The following code shows the model's predictions for the test data set

```
ht_aug <- augment(ht_fit, test_data)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

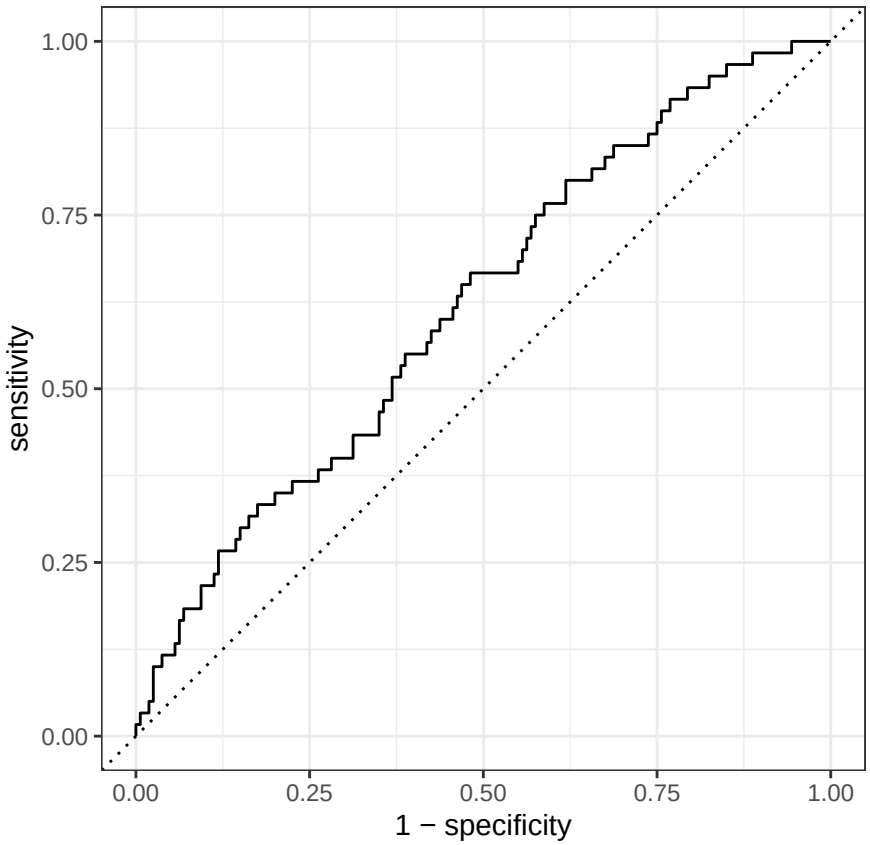
```
ht_aug %>% select(high_temp, .pred_class, .pred_high_temp, .pred_normal_range)
```

```
## # A tibble: 220 x 4
##    high_temp    .pred_class  .pred_high_temp .pred_normal_range
##    <fct>        <fct>                  <dbl>              <dbl>
##  1 high_temp    high_temp              0.832              0.168
##  2 normal_range normal_range           0.378              0.622
##  3 normal_range high_temp              0.730              0.270
##  4 normal_range normal_range           0.137              0.863
##  5 high_temp    normal_range           0.137              0.863
##  6 normal_range normal_range           0.109              0.891
##  7 normal_range normal_range           0.241              0.759
##  8 normal_range normal_range           0.354              0.646
##  9 high_temp    normal_range           0.313              0.687
## 10 high_temp    normal_range           0.354              0.646
## # ... with 210 more rows
```

The results of the ROC curve the the ROC-AUC value of .62 show the model is able to predict high temperature, but is not ideal.

```
#Follow the example in the Use a trained workflow to predict section of the tutorial to look at the pre

ht_aug %>%
  roc_curve(truth = high_temp, .pred_high_temp) %>%
  autoplot()
```




```
ht_aug %>%
  roc_auc(truth = high_temp, .pred_high_temp)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.619
```

Testing the Main Predictor

Sneeze

The predictor found to have one of the strongest relationships to body temperature is the presence of sneezing as a symptom. The following model will test whether sneezing can predict a high temperature.

```
#Let's re-do the fitting but now with a model that only fits the main predictor to the categorical outc

# Create new recipes
sneeze_rec <- recipe(high_temp ~ Sneeze, data = train_data)
sneeze_test <- recipe(high_temp ~ Sneeze, data = test_data)
```

```
sneeze_workflow <-
  workflow() %>%
  add_model(lr_model) %>%
```

```
  add_recipe(sneeze_rec)

sneeze_workflow
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
##
## -- Model -----------------------------------------------------------------------
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

```
sneeze_fit <- sneeze_workflow %>%
  fit(data = train_data)
```

Once again, the p-values suggest that sneezing has a strong association with a high body temperature.

```
sneeze_fit %>%
  extract_fit_parsnip() %>%
  tidy()
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    0.413     0.133      3.10 0.00192
## 2 SneezeYes      0.676     0.192      3.51 0.000442
```

```
# use predict to predict if patient has a high temperature
predict(sneeze_fit, test_data)
```

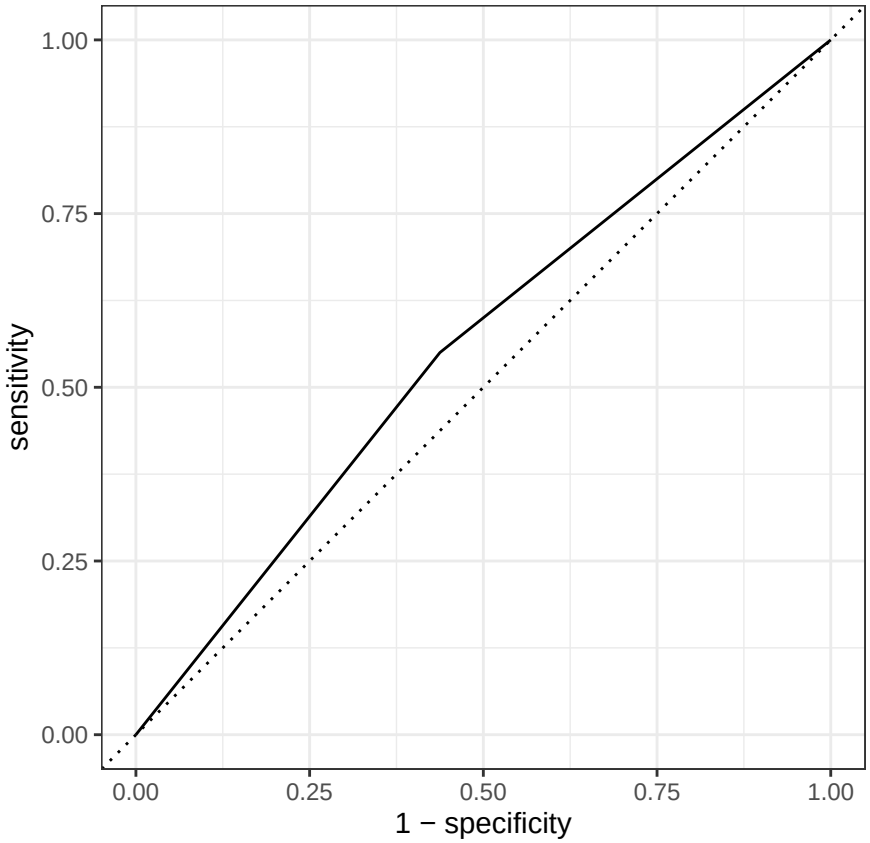
```
sneeze_aug <- augment(sneeze_fit, test_data)

sneeze_aug %>% select(high_temp, .pred_class, .pred_high_temp, .pred_normal_range)
```

```
## # A tibble: 220 x 4
##    high_temp    .pred_class  .pred_high_temp .pred_normal_range
##    <fct>        <fct>                  <dbl>              <dbl>
##  1 high_temp    normal_range           0.398              0.602
##  2 normal_range normal_range           0.398              0.602
##  3 normal_range normal_range           0.398              0.602
##  4 normal_range normal_range           0.398              0.602
##  5 high_temp    normal_range           0.252              0.748
##  6 normal_range normal_range           0.252              0.748
##  7 normal_range normal_range           0.252              0.748
##  8 normal_range normal_range           0.252              0.748
##  9 high_temp    normal_range           0.252              0.748
## 10 high_temp    normal_range           0.252              0.748
## # ... with 210 more rows
```

Although the p-values suggest a strong relationship, the ROC curve and ROC-AUC value (.56), while significant, show this model is not especially strong for predicting whether a patient has a high body temperature.

```
sneeze_aug %>%
  roc_curve(truth = high_temp, .pred_high_temp) %>%
  autoplot()
```




```
sneeze_aug %>%
  roc_auc(truth = high_temp, .pred_high_temp)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.556
```