Model Fitting Part 2

Joe Martin

10/18/2021

Introduction

The goal of this section is to determine whether it is possible to predict nausea based on other symptoms a patient is displaying. This is first tested with all present variables as predictors, then tested against the main predictor of interest, Runny Nose.

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

# I messed this part up and created a new Body Temp variable. This is unnecessary, but I'm leaving it.

set.seed(2)

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

# 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

nausea_rec <- recipe(Nausea ~ ., data = train_data)
nausea_test <- recipe(Nausea ~ ., data = test_data)

#summary(ht_rec)
```

Set up logistic model and create workflow

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

nausea_workflow <-
```

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

Show relationships between predictor variables and outcome

```
nausea_fit <- nausea_workflow %>%
  fit(data = train_data)

nausea_fit %>%
  extract_fit_parsnip() %>%
  tidy()
```

```
## # A tibble: 39 x 5
##    term                 estimate std.error statistic p.value
##    <chr>                   <dbl>     <dbl>     <dbl>   <dbl>
##  1 (Intercept)          -20.1       15.1      -1.33    0.184
##  2 SwollenLymphNodesYes  -0.270      0.240    -1.12    0.262
##  3 ChestCongestionYes     0.205      0.267     0.767   0.443
##  4 ChillsSweatsYes        0.0574     0.353     0.163   0.871
##  5 NasalCongestionYes     0.342      0.306     1.12    0.264
##  6 CoughYNYes            -0.152      0.623    -0.244   0.807
##  7 SneezeYes              0.0453     0.264     0.171   0.864
##  8 FatigueYes             0.599      0.476     1.26    0.208
##  9 SubjectiveFeverYes     0.328      0.292     1.12    0.261
## 10 HeadacheYes            0.439      0.365     1.20    0.229
## # ... with 29 more rows
```

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

```
# use predict to predict if patient has nausea
predict(nausea_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

```
nausea_aug <- augment(nausea_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
```

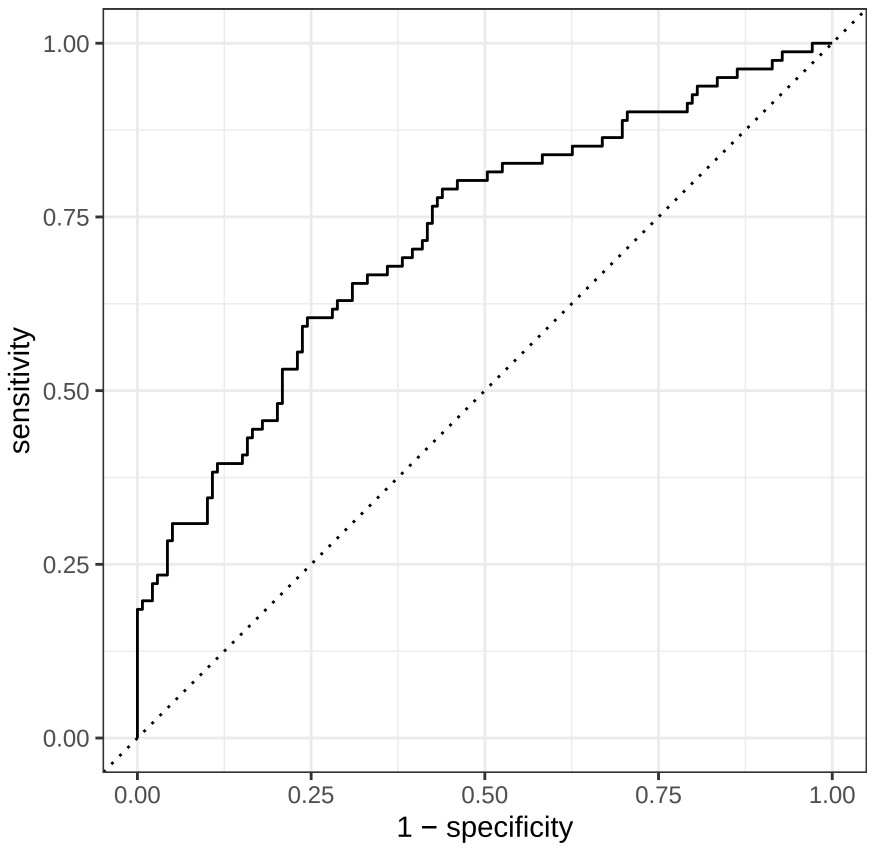
```
nausea_aug %>% select(Nausea, .pred_class, .pred_Yes, .pred_No)
```

```
## # A tibble: 220 x 4
##    Nausea .pred_class .pred_Yes .pred_No
##    <fct>  <fct>           <dbl>    <dbl>
##  1 No     No              0.229   0.771
##  2 Yes    Yes             0.966   0.0342
##  3 Yes    Yes             0.949   0.0514
##  4 No     No              0.224   0.776
##  5 Yes    No              0.243   0.757
##  6 Yes    No              0.142   0.858
##  7 Yes    No              0.215   0.785
##  8 No     No              0.222   0.778
##  9 No     No              0.210   0.790
## 10 Yes    Yes             0.715   0.285
## # ... with 210 more rows
```

The results of the ROC curve the the ROC-AUC value of .72 shows the model is potentially useful for predicting nausea.

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

nausea_aug %>%
  roc_curve(truth = Nausea, .pred_Yes, event_level= "second") %>%
  autoplot()
```



```
nausea_aug %>%
  roc_auc(truth = Nausea, .pred_Yes, event_level= "second")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.722
```

Testing the Main Predictor

Runny Nose

The following model will test whether having a runny nose is a good predictor of nausea. Based on the results testing all variables as predictors of nausea, it is unlikely that having a runny nose will predict nausea (p-value of .699).

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

# Create new recipes
RunnyNose_rec <- recipe(Nausea ~ RunnyNose, data = train_data)
RunnyNose_test <- recipe(Nausea ~ RunnyNose, data = test_data)
```

```
RunnyNose_workflow <-
  workflow() %>%
  add_model(lr_model) %>%
  add_recipe(RunnyNose_rec)
```

```
RunnyNose_fit <- RunnyNose_workflow %>%
  fit(data = train_data)
```

```
RunnyNose_fit %>%
  extract_fit_parsnip() %>%
  tidy()
```

```
## # A tibble: 2 x 5
##   term         estimate std.error statistic  p.value
##   <chr>           <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  -0.661       0.178   -3.72   0.000198
## 2 RunnyNoseYes  0.00462     0.209    0.0221 0.982
```

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

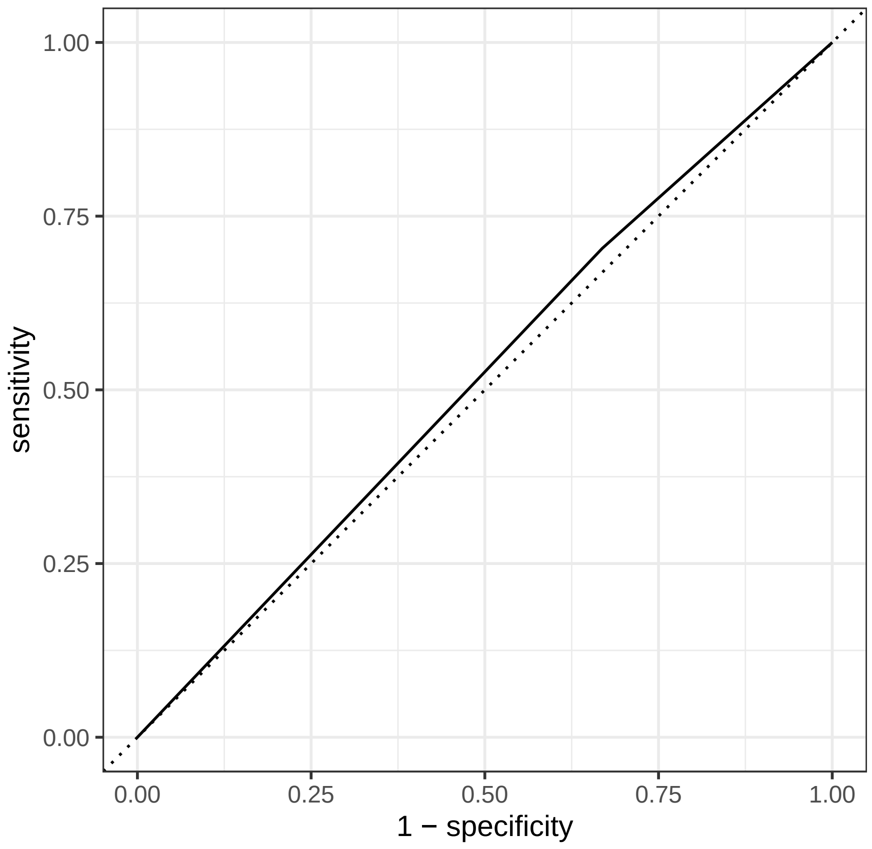
```
RunnyNose_aug <- augment(RunnyNose_fit, test_data)

RunnyNose_aug %>% select(Nausea, .pred_class, .pred_Yes, .pred_No)
```

```
## # A tibble: 220 x 4
##    Nausea .pred_class .pred_Yes .pred_No
##    <fct>  <fct>           <dbl>    <dbl>
##  1 No     No              0.340    0.660
##  2 Yes    No              0.340    0.660
##  3 Yes    No              0.340    0.660
##  4 No     No              0.341    0.659
##  5 Yes    No              0.341    0.659
##  6 Yes    No              0.341    0.659
##  7 Yes    No              0.341    0.659
##  8 No     No              0.341    0.659
##  9 No     No              0.341    0.659
## 10 Yes    No              0.341    0.659
## # ... with 210 more rows
```

Once again, the ROC curve and ROC-AUC value (.52) indicate that RunnyNose might be a good variable for predicting Nausea, but is not strong enough of a model to be significant.

```
RunnyNose_aug %>%
  roc_curve(truth = Nausea, .pred_Yes, event_level= "second") %>%
  autoplot()
```



```
RunnyNose_aug %>%
  roc_auc(truth = Nausea, .pred_Yes, event_level= "second")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.517
```

Analysis Part 2

By Morgan Taylor The goal of this section is to fit the continuous outcome of interest (BodyTemp) to a linear. This is first tested with all present variables as predictors, then tested against the main predictor of interest, Runny Nose. Since the previous section already used the `Tidymodels` infrastructure, we can copy and paste a majority of the code, adjusting the outcome of interest and model type.

We can use the already loaded df, with the same test_data and train_data df.

Testing All Predictors

The following code generates recipes for the training data set, then builds a workflow to fit to a linear model to all predictor variables.

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

bodytemp_rec <- recipe(BodyTemp ~ ., data = train_data)
```

Set up linear model and create workflow for training data

```
linear_model <- linear_reg() %>%
                  set_engine("lm")

bodytemp_workflow <-
  workflow() %>%
  add_model(linear_model) %>%
  add_recipe(bodytemp_rec)
```

Show relationships between predictor variables and outcome

```
bodytemp_fit <- bodytemp_workflow %>%
  fit(data = train_data)

bodytemp_fit %>%
  extract_fit_parsnip() %>%
  tidy()
```

```
## # A tibble: 39 x 5
##    term                 estimate std.error statistic p.value
##    <chr>                   <dbl>     <dbl>     <dbl>   <dbl>
##  1 (Intercept)          100.        0.243    412.     0
##  2 SwollenLymphNodesYes  -0.0476    0.0715    -0.667  0.505
##  3 ChestCongestionYes     0.0225    0.0776     0.290  0.772
##  4 ChillsSweatsYes        0.110     0.0994     1.10   0.270
##  5 NasalCongestionYes    -0.180     0.0869    -2.07   0.0392
##  6 CoughYNYes            -0.232     0.181     -1.29   0.199
##  7 SneezeYes             -0.0246    0.0779    -0.316  0.752
##  8 FatigueYes             0.124     0.122      1.01   0.311
##  9 SubjectiveFeverYes     0.188     0.0837     2.25   0.0252
## 10 HeadacheYes           -0.0674    0.0982    -0.687  0.492
## # ... with 29 more rows
```

The following code predicts a patient's body temperature based on the model built above for the training data set.

```r
# use predict to predict patient body temperature
predict(bodytemp_fit, train_data)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
## # A tibble: 510 x 1
##    .pred
##    <dbl>
##  1  98.5
##  2  98.1
##  3  98.2
##  4  98.3
##  5 100.
##  6 100.
##  7 100.
##  8  98.2
##  9  98.5
## 10  99.8
## # ... with 500 more rows
```

```r
#create df with model predictions and actual measures
bodytemp_aug_train <- augment(bodytemp_fit, train_data)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```r
bodytemp_all_train <- bodytemp_aug_train %>%
                        mutate(data = "train",
                               model = "all")
```

The following code predicts a patient's body temperature based on the model built above for the test data set.

```r
# use predict to predict patient body temperature
predict(bodytemp_fit, test_data)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
## # A tibble: 220 x 1
##    .pred
##    <dbl>
##  1 100.
##  2  99.0
##  3  98.6
##  4  98.1
##  5 101.
##  6  98.3
##  7  98.3
##  8  98.2
##  9 100.
## 10 101.
## # ... with 210 more rows
```

```
#create df with model predictions and actual measures
bodytemp_aug_test <- augment(bodytemp_fit, test_data)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
#fit into df for yardstick evaluation later
bodytemp_all_test <- bodytemp_aug_test %>%
                        mutate(data = "test",
                               model = "all")
```

Testing Only Runny Nose

The following code generates recipes for the training data set, then builds a workflow to fit to a linear model to only the runny nose predictor variable.

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

bodytemp_rec_RN <- recipe(BodyTemp ~ RunnyNose, data = train_data)
```

Create workflow for RunnyNose (We can use the previously defined model)

```
bodytemp_workflow_RN <-
        workflow() %>%
        add_model(linear_model) %>%
        add_recipe(bodytemp_rec_RN)
```

Show relationships between predictor variable and outcome

```
bodytemp_fit_RN <- bodytemp_workflow_RN %>%
  fit(data = train_data)

bodytemp_fit_RN %>%
  extract_fit_parsnip() %>%
  tidy()
```

```
## # A tibble: 2 x 5
##   term         estimate std.error statistic p.value
##   <chr>           <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)    99.2       0.101    980.   0
## 2 RunnyNoseYes   -0.365     0.119     -3.07 0.00224
```

The following code predicts a patient's body temperature based on the model built above for the training data set.

```r
# use predict to predict patient body temperature
predict(bodytemp_fit_RN, train_data)
```

```
## # A tibble: 510 x 1
##    .pred
##    <dbl>
##  1  98.8
##  2  98.8
##  3  98.8
##  4  98.8
##  5  98.8
##  6  98.8
##  7  98.8
##  8  98.8
##  9  98.8
## 10  98.8
## # ... with 500 more rows
```

```r
#create df with model predictions and actual measures
bodytemp_aug_train_RN <- augment(bodytemp_fit_RN, train_data)

bodytemp_RN_train <- bodytemp_aug_train_RN %>%
                        mutate(data = "train",
                               model = "RN")
```

The following code predicts a patient's body temperature based on the model built above for the test data set.

```r
# use predict to predict patient body temperature
predict(bodytemp_fit_RN, test_data)
```

```
## # A tibble: 220 x 1
##    .pred
##    <dbl>
##  1  99.2
##  2  99.2
##  3  99.2
##  4  98.8
##  5  98.8
##  6  98.8
##  7  98.8
##  8  98.8
##  9  98.8
## 10  98.8
## # ... with 210 more rows
```

```r
#create df with model predictions and actual measures
bodytemp_aug_test_RN <- augment(bodytemp_fit_RN, test_data)

#fit into df for yardstick evaluation later
bodytemp_RN_test <- bodytemp_aug_test_RN %>%
                        mutate(data = "test",
                               model = "RN")
```

Linear Model Evaluation

As this is now a linear regression model, we need to use RMSE instead of ROC as the metric. We can combine the predictions into one df and then calculate the RMSE for each model to compare. Inspiration for the coding for this section comes from the TidyModels Yardstick Website

```
#First create comprehensive df with all of the predictions
bodytemp_preds <- bind_rows(bodytemp_all_test,
                            bodytemp_all_train,
                            bodytemp_RN_test,
                            bodytemp_RN_train)

#Use the yardstick package to calculate the metrics on resamples
bodytemp_metrics <- bodytemp_preds %>%
                      dplyr::group_by(data, model) %>%
                      yardstick::metrics(truth = BodyTemp, estimate = .pred)
#for interpretation: rmse = Root Mean Squared Error (RMSE), rsq = R^2, mae = Mean Absolute Error (MAE)

#make a df that just displays RMSE for each model and data (all dplyr functions)
bodytemp_RMSE <- bodytemp_metrics %>%
                    filter(.metric == "rmse") %>%
                    select(-.estimator)

#sort the df to make it more understandable
bodytemp_RMSE[order( bodytemp_RMSE[, 4, 2] ), ]
```

```
## # A tibble: 4 x 4
##   data  model .metric .estimate
##   <chr> <chr> <chr>       <dbl>
## 1 test  all   rmse        0.704
## 2 train all   rmse        0.708
## 3 test  RN    rmse        1.16
## 4 train RN    rmse        1.20
```

Interpreting the Results: RMSE is the standard deviation of the unexplained variance, and has the same units as BodyTemp. A lower RMSE indicates a better fit. Based on the results above, we can conclude that the model that uses all of the predictors is a better fit than the model that only uses RunnyNose as the predictor. Interestingly, in the RN model, the test dataset had a lower RMSE than the training dataset, whereas in the all model, the training dataset had a lower RMSE than the test dataset.