PSTAT131 - HW4

Xilong Li (3467966)

2022-05-02

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.2.0 --

## v broom        0.7.12     v rsample      0.1.1
## v dials        0.1.0      v tune         0.2.0
## v infer        1.0.0      v workflows    0.2.6
## v modeldata    0.1.1      v workflowsets 0.2.1
## v parsnip      0.2.1      v yardstick    0.0.9
## v recipes      0.2.0

## -- 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()
## * Search for functions across packages at https://www.tidymodels.org/find/
```

```
library(discrim)
```

```
##
## Attaching package: 'discrim'
## The following object is masked from 'package:dials':
##
##     smoothness
```

```
library(poissonreg)
library(corrr)
library(klaR) # for naive bayes
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
tidymodels_prefer()
Titanic <- read.csv("titanic.csv")
Titanic$survived <- factor(Titanic$survived,levels = c("Yes","No"))
Titanic$pclass <- as.character(Titanic$pclass)
Titanic$pclass <- as.factor(Titanic$pclass)
```

##Question 1:

```
set.seed(2216)

titan_split <- initial_split(Titanic, prop = 0.80,
                             strata = survived)
titan_train <- training(titan_split)
titan_test <- testing(titan_split)

titan_split
```

```
## <Analysis/Assess/Total>
## <712/179/891>
```

```
dim(titan_train)
```

```
## [1] 712  12
```

```
titan_recipe <- recipe(survived ~
                         pclass +
                         sex +
                         age +
                         sib_sp +
                         parch +
                         fare,
                       data = titan_train) %>%
  step_impute_linear(age) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(~ starts_with("sex"):fare) %>%
  step_interact(~ age:fare)

titan_recipe
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          6
##
## Operations:
##
## Linear regression imputation for age
## Dummy variables from all_nominal_predictors()
## Interactions with starts_with("sex"):fare
## Interactions with age:fare
```

##Question 2:

```
titan_folds <- vfold_cv(titan_train, v = 10)
titan_folds
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 2
##    splits           id
##    <list>           <chr>
##  1 <split [640/72]> Fold01
##  2 <split [640/72]> Fold02
##  3 <split [641/71]> Fold03
##  4 <split [641/71]> Fold04
##  5 <split [641/71]> Fold05
##  6 <split [641/71]> Fold06
##  7 <split [641/71]> Fold07
##  8 <split [641/71]> Fold08
##  9 <split [641/71]> Fold09
## 10 <split [641/71]> Fold10
```

##Question 3: 1) We use k-fold cross-validation so that we can hold out a subset of the training observations from the fitting process, and apply the learned model to those held out observations.
2) We use k-fold cross-validation method so that it results in a less biased model, "because it ensures that every observation from the original dataset has the chance of appearing in training and test set." (cited from: https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f#:~:text=K-Folds%20Cross%20Validation%3A&text=Because%20it%20ensures%20that%20every,have%20a%20limited%20input%20data
3) If we use the entire training dataset, we would use the boot approach. strap ##Question 4:

```
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")
```

```r
log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(titan_recipe)
```

```r
lda_mod <- discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")

lda_wkflow <- workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(titan_recipe)
```

```r
qda_mod <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

qda_wkflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(titan_recipe)
```

Therefore, 30 models in total will be fitted to the data, because there are 10 folds for each of 3 different models.

##Question 5:

```r
# Fit the logistic regression model
log_fit <-
  log_wkflow %>%
  fit_resamples(titan_folds)

# Fit the LDA model
lda_fit <-
  lda_wkflow %>%
  fit_resamples(titan_folds)

# Fit the QDA model
qda_fit <-
  qda_wkflow %>%
  fit_resamples(titan_folds)
```

##Question 6:

```r
# See the result of logistic regression model
collect_metrics(log_fit)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.810    10  0.0166 Preprocessor1_Model1
## 2 roc_auc  binary     0.843    10  0.0212 Preprocessor1_Model1
```

```r
# See the result of LDA model
collect_metrics(lda_fit)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.789    10  0.0213 Preprocessor1_Model1
## 2 roc_auc  binary     0.845    10  0.0212 Preprocessor1_Model1
```

```r
# See the result of QDA model
collect_metrics(qda_fit)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.782    10  0.0130 Preprocessor1_Model1
## 2 roc_auc  binary     0.843    10  0.0194 Preprocessor1_Model1
```

##Question 7: Since the logistic regression model has the highest accuracy, I will use the logistic regression model as my final model.

```r
final_fit <- fit(log_wkflow, titan_train)
```

##Question 8:

```r
predict(final_fit, new_data = titan_test, type = "prob")
```

```
## # A tibble: 179 x 2
##    .pred_Yes .pred_No
##        <dbl>    <dbl>
##  1    0.914    0.0864
##  2    0.117    0.883
##  3    0.0873   0.913
##  4    0.568    0.432
##  5    0.264    0.736
##  6    0.245    0.755
##  7    0.0510   0.949
##  8    0.169    0.831
##  9    0.0352   0.965
## 10    0.775    0.225
## # ... with 169 more rows
```

```r
augment(final_fit, new_data = titan_test) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

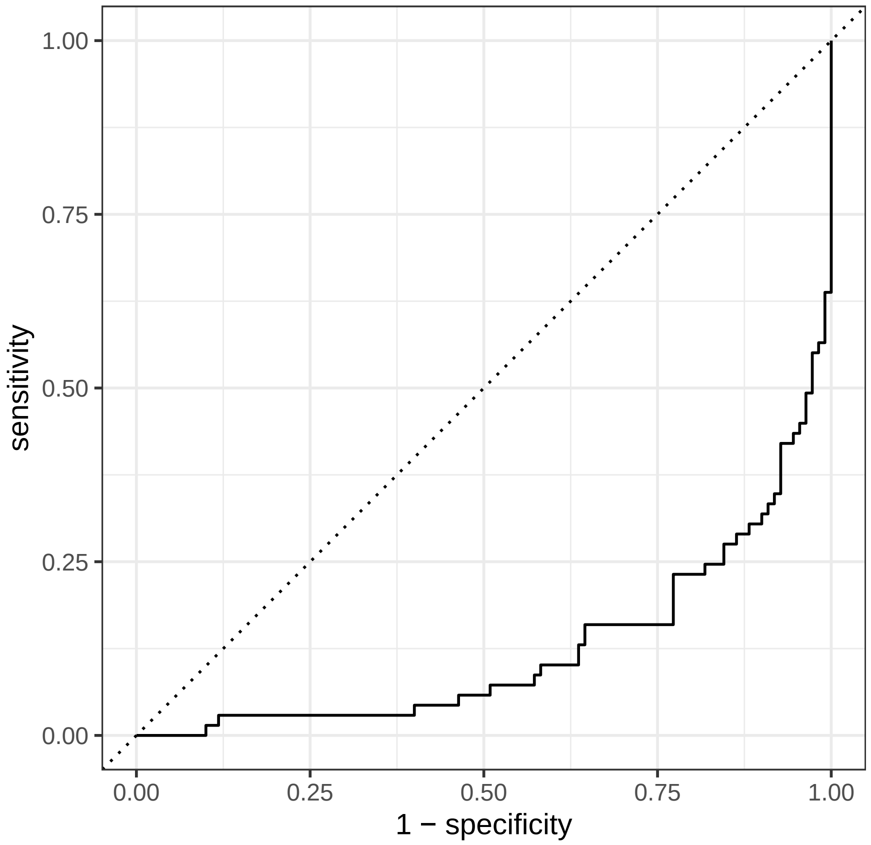
```
##           Truth
## Prediction Yes No
##        Yes  50 16
##        No   19 94
```

```r
multi_metric <- metric_set(accuracy, sensitivity, specificity)

augment(final_fit, new_data = titan_test) %>%
  multi_metric(truth = survived, estimate = .pred_class)
```

```
## # A tibble: 3 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 accuracy    binary         0.804
## 2 sensitivity binary         0.725
## 3 specificity binary         0.855
```

```r
augment(final_fit, new_data = titan_test) %>%
  roc_curve(survived, .pred_No) %>%
  autoplot()
```



Therefore, as the accuracy shown above, the accuracy generated based on the entire training dataset is very close to the average accuracy generated based on the folds.
Thus, overall, the model fits well in predicting the response parameter "survived".