hwk 4 - 131

cristian razo

5/3/2022

Question 1

```
tidymodels_prefer()

dt=read.csv("Desktop/homework-4/data/titanic.csv")
dt$survived <- factor(dt$survived,levels=c("Yes","No"))
dt$pclass <- factor(dt$pclass)



set.seed(221)

split <- initial_split(dt, prop = 0.80, strata = survived)
train <- training(split)
test <- testing(split)

sub_train=train[c(2,3,5,6,7,8,10)]
nrow(dt)
```

```
## [1] 891
```

```
nrow(train)
```

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

```
nrow(test)
```

```
## [1] 179
```

By using the length function we can verify that 80% of the data is in the training set and that 20% is in the testing set.I also reorganized the levels to be as (Yes , No) sequence.

Question 2

```
k_folds <- vfold_cv(sub_train, v = 10)
k_folds
```

```
## #  10-fold cross-validation
## # A tibble: 10 × 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

The function vfold_cv allows us to make 10 different combination of a validation set . The function allows us to take turn in what proportion of the data will be in training set and validation set . This allows us to find the best model without having to fit it on our test data set. This method is meant to not spoil our testing set and able fit the best possible model based on the K folds method I described. If we do a k fold on the whole data the method it is known as leave one out cross validation .

Question 4 & 5

```
titanic_recipe=recipe(survived ~ . ,data=sub_train) %>%
  step_impute_linear(age,impute_with = imp_vars(all_predictors()))%>%
  prep(sub_train)%>%
  step_dummy(all_nominal_predictors())
```

I will use the same recipe I did for the previous classifcation home work.

```
tuned_log_reg=logistic_reg()%>%
  set_engine("glm") %>%
  set_mode("classification")




log_wrkflw=workflow() %>%
  add_model(tuned_log_reg) %>%
  add_recipe(titanic_recipe)



log_res <- fit_resamples(log_wrkflw,k_folds)
```

This is my logistic regression workflow and fitted model .

```
lda_model = discrim_linear() %>%
  set_mode('classification')%>%
  set_engine("MASS")



lda_wrkflw=workflow() %>%
  add_model(lda_model) %>%
  add_recipe(titanic_recipe)


lda_res <- fit_resamples(lda_wrkflw,k_folds)
```

This is my Linear discriminant analysis workflow and fitted model .

```
lmda_model=discrim_quad()%>%
  set_engine("MASS") %>%
  set_mode("classification")


lmda_wrkflw=workflow() %>%
  add_model(lmda_model) %>%
  add_recipe(titanic_recipe)


lmda_fit=fit(lmda_wrkflw,sub_train)

lmda_res <- fit_resamples(lmda_wrkflw,k_folds)
```

This is my Linear multiple discriminant analysis workflow and fitted model .

```
nb=naive_Bayes()%>%
  set_engine("klaR") %>%
  set_mode("classification")%>%
  set_args(usekernel=FALSE)


nb_wrkflw=workflow() %>%
  add_model(nb) %>%
  add_recipe(titanic_recipe)


nb_fit=fit(nb_wrkflw,sub_train)

nb_res <- fit_resamples(nb_wrkflw,k_folds)
```

```
## ! Fold01: preprocessor 1/1, model 1/1 (predictions): Numerical 0 probability for a...
```

```
## ! Fold02: preprocessor 1/1, model 1/1 (predictions): Numerical 0 probability for a...
## ! Fold03: preprocessor 1/1, model 1/1 (predictions): Numerical 0 probability for a...
```

```
## ! Fold04: preprocessor 1/1, model 1/1 (predictions): Numerical 0 probability for a...
```

```
## ! Fold05: preprocessor 1/1, model 1/1 (predictions): Numerical 0 probability for a...
```

```
## ! Fold06: preprocessor 1/1, model 1/1 (predictions): Numerical 0 probability for a...
```

```
## ! Fold07: preprocessor 1/1, model 1/1 (predictions): Numerical 0 probability for a...
```

```
## ! Fold08: preprocessor 1/1, model 1/1 (predictions): Numerical 0 probability for a...
```

```
## ! Fold09: preprocessor 1/1, model 1/1 (predictions): Numerical 0 probability for a...
```

```
## ! Fold10: preprocessor 1/1, model 1/1 (predictions): Numerical 0 probability for a...
```

This is my Naive Bayes workflow and fitted model .

Question 6

```
collect_metrics(log_res)
```

```
## # A tibble: 2 × 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.798    10  0.0141 Preprocessor1_Model1
## 2 roc_auc  binary     0.853    10  0.0176 Preprocessor1_Model1
```

```
collect_metrics(lda_res)
```

```
## # A tibble: 2 × 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.798    10  0.0133 Preprocessor1_Model1
## 2 roc_auc  binary     0.854    10  0.0169 Preprocessor1_Model1
```

```
collect_metrics(lmda_res)
```

```
## # A tibble: 2 × 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.802    10  0.0127 Preprocessor1_Model1
## 2 roc_auc  binary     0.846    10  0.0151 Preprocessor1_Model1
```

```
collect_metrics(nb_res)
```

```
## # A tibble: 2 × 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.795    10  0.0147 Preprocessor1_Model1
## 2 roc_auc  binary     0.830    10  0.0110 Preprocessor1_Model1
```

The K fold collected metrics results shows us that LMDA model has the best accuracy and AOC score.The Standard Error is also very small which is what we want because our to minimize it. I will choose LMDA to be my best model and will fit it onto the whole training set.

##Question 7

```
lmda_fit=fit(lmda_wrkflw,sub_train)
```

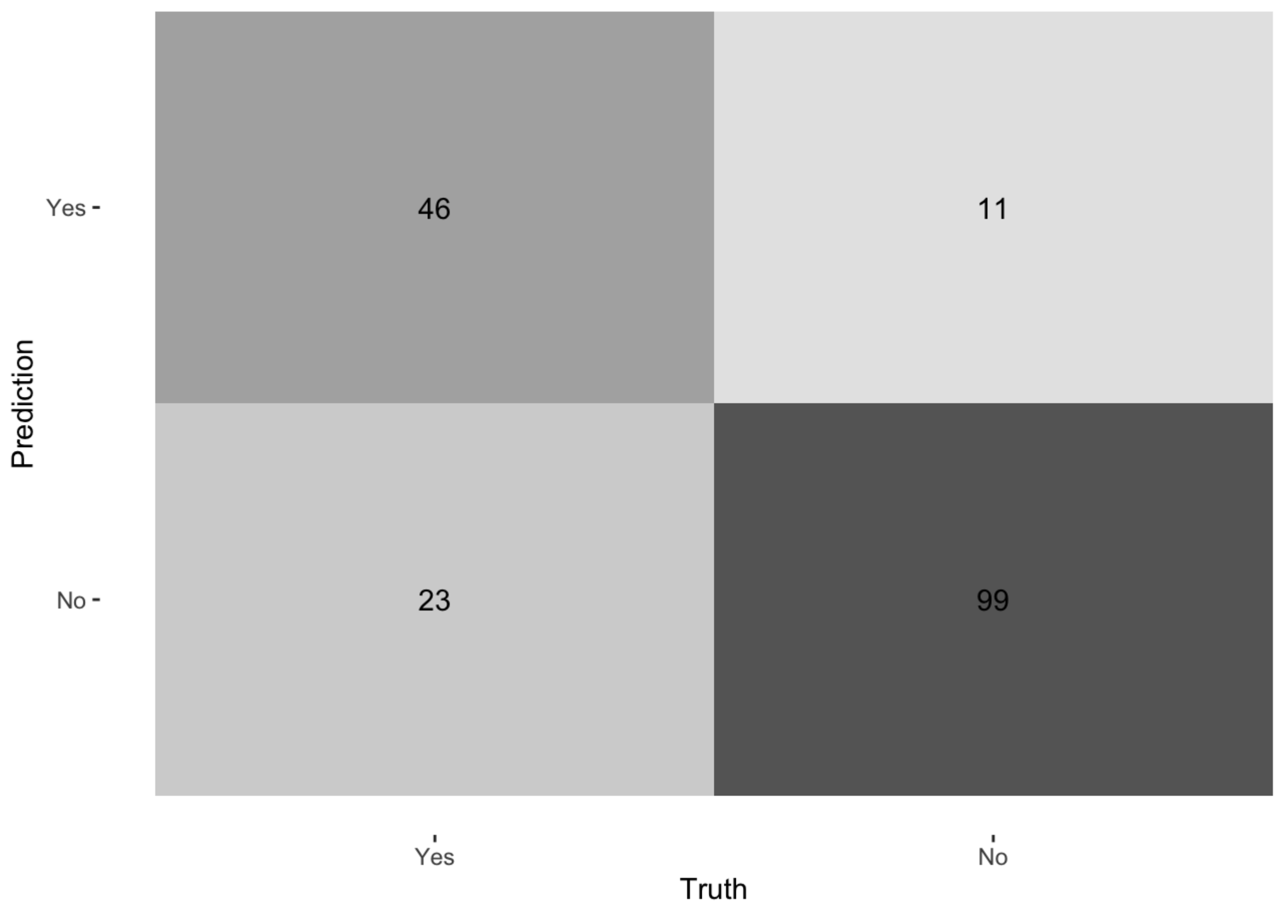
Now that I fitted and trained an LMDA model onto the training set I can know check how well it will the model will fit my testing set .

Question 8

```
sub_test=test[c(2,3,5,6,7,8,10)]
predict(lmda_fit,new_data = sub_test,type='prob')
```

```
## # A tibble: 179 × 2
##    .pred_Yes .pred_No
##        <dbl>    <dbl>
##  1    0.0304  0.970
##  2    0.988   0.0121
##  3    0.567   0.433
##  4    0.0833  0.917
##  5    0.635   0.365
##  6    0.994   0.00641
##  7    0.241   0.759
##  8    0.296   0.704
##  9    0.595   0.405
## 10    0.0451  0.955
## # … with 169 more rows
```

```
augment(lmda_fit,new_data = sub_test)%>%
  conf_mat(truth=survived,estimate=.pred_class)%>%
  autoplot(type='heatmap')
```



```
lmda_test_acc= augment(lmda_fit,new_data = sub_test)%>%
  accuracy(truth=factor(survived), estimate=.pred_class)

lmda_test_acc
```

```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.810
```

The testing accuracy score is higher than K folds accuracy score by .008. Their difference from each other is not that large .This imply that our model trained on the traning set was able to fit the testing set pretty well.