HW 4

PSTAT 131/231 Arthur Starodynov

Contents

```
titanic.data$pclassfac <- as.factor(titanic.data$pclass)
titanic.data$survived <- as.factor(titanic.data$survived)
```

Problem 1)

```
p <- 0.7
strats <- titanic.data$survived

rr <- split(1:length(strats), strats)
idx <- sort(as.numeric(unlist(sapply(rr, function(x) sample(x, length(x) * p)))))

train <- titanic.data[idx, ]
test <- titanic.data[-idx, ]
```

Problem 2)

```
train_fold <- vfold_cv(train, v = 10)
train_fold
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 2
##    splits           id
##    <list>           <chr>
##  1 <split [560/63]> Fold01
##  2 <split [560/63]> Fold02
##  3 <split [560/63]> Fold03
##  4 <split [561/62]> Fold04
##  5 <split [561/62]> Fold05
##  6 <split [561/62]> Fold06
##  7 <split [561/62]> Fold07
##  8 <split [561/62]> Fold08
##  9 <split [561/62]> Fold09
## 10 <split [561/62]> Fold10
```

```
train_control <- trainControl(method = "cv",
                              number = 10)
kfold_train <- train(factor(survived) ~., data = train,
               trControl = train_control,
               na.action = na.exclude)
kfold_train
```

```
## Random Forest
##
## 623 samples
##  12 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 109, 108, 109, 107, 108, 108, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     2   0.7339161  0.0000000
##   158   0.7833916  0.4065385
##   315   0.8174825  0.5300179
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 315.
```

Problem 3) K fold cross Validation is another way to test the skill of a model in machine learning and test its accuracy. What it does is sokit the dataset in k groups, and within each of the groups it will take the group to hold the data and use the remaining groups as training data set so that it could run the model on the training set eventually running through all the other data sets to test to test is skill .Using K fold allows us to run the model on each data set making sure if there is one that does not fit well it can be unfolded and seen and then taken out. It specifies more in depth then just fitting the model on a bigger data set it allows the data set to be thinner. We would use logistic regression with the whole data set.

Problem 4)

```
simple_recipe <- recipe(survived ~
                      pclass+sex+age+sib_sp+parch+fare,
                        data=train) %>%
  step_impute_linear(age) %>%
  step_dummy(all_nominal_predictors(),one_hot = F) %>%
  step_interact(terms = ~starts_with("sex"):fare) %>%
  step_interact(terms = ~ age:fare)
```

```
lgm_model = logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")
lgm_workflow = workflow() %>%
  add_model(lgm_model) %>%
  add_recipe(simple_recipe)
```

```
library(discrim)
dis_model = discrim_linear() %>%
  set_engine("MASS") %>%
  set_mode("classification")
dis_workflow = workflow() %>%
  add_model(dis_model) %>%
  add_recipe(simple_recipe)
```

```
quad_model = discrim_quad() %>%
  set_engine("MASS") %>%
  set_mode("classification")
quad_workflow = workflow() %>%
  add_model(quad_model) %>%
  add_recipe(simple_recipe)
```

we will be fitting 10 folds so 10 datasets along 3 different workflow models.

Problem 5)

```
lgm_tune <- lgm_workflow %>%
  tune_grid(resamples = train_fold)
dis_tune <- dis_workflow %>%
  tune_grid(resamples = train_fold)
quad_tune <- quad_workflow %>%
  tune_grid(resamples = train_fold)
```

Problem 6)

```
lgm_metric <- collect_metrics(lgm_tune)
print(lgm_metric)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.833    10  0.0201 Preprocessor1_Model1
## 2 roc_auc  binary     0.875    10  0.0136 Preprocessor1_Model1
```

```
dis_metric <- collect_metrics(dis_tune)
print(dis_metric)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.811    10  0.0175 Preprocessor1_Model1
## 2 roc_auc  binary     0.876    10  0.0136 Preprocessor1_Model1
```

```
quad_metric <- collect_metrics(quad_tune)
print(quad_metric)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.807    10  0.0152 Preprocessor1_Model1
## 2 roc_auc  binary     0.861    10  0.0121 Preprocessor1_Model1
```

The logistic regression model was the best fitting out of the 3 it is clear due to the mean being the highest as well as having the lowest standard deviation out of all the others meaning that it would be the most accurate.

Problem 7)

```r
lgm_fit <- lgm_workflow %>%
  fit(train)
```

Problem 8)

```r
predict_train_model= bind_cols(predict(lgm_fit, test),
                               test$survived)
colnames(predict_train_model) = c("GLM Predict", "True")
print(accuracy(predict_train_model,
               truth='True', estimate="GLM Predict")$.estimate)
```

```
## [1] 0.7798507
```

The accuracy on the GLM of the test data was lower at 78% compared to the data of the folded parts, meaning that the folded data was able to be predicted better and was more accurate at a mean/accuracy of around 82%