Pstat 131 Homework 3

Yu Tian

Spring 2022-04-19

Classification

```r
# Read the full titanic data set into R using read_csv()
titanic <- read_csv(file = "titanic.csv")
titanic$survived <- factor(titanic$survived, levels=c("Yes","No"))
titanic$pclass <- factor(titanic$pclass)
titanic %>% head()
```

View Titanic Date

```
## # A tibble: 6 x 12
##   passenger_id survived pclass name  sex     age sib_sp parch ticket  fare cabin
##          <dbl> <fct>    <fct>  <chr> <chr> <dbl>  <dbl> <dbl> <chr>  <dbl> <chr>
## 1            1 No       3      Brau~ male     22      1     0 A/5 2~  7.25 <NA>
## 2            2 Yes      1      Cumi~ fema~    38      1     0 PC 17~ 71.3  C85
## 3            3 Yes      3      Heik~ fema~    26      0     0 STON/~  7.92 <NA>
## 4            4 Yes      1      Futr~ fema~    35      1     0 113803 53.1  C123
## 5            5 No       3      Alle~ male     35      0     0 373450  8.05 <NA>
## 6            6 No       3      Mora~ male     NA      0     0 330877  8.46 <NA>
## # ... with 1 more variable: embarked <chr>
```

Question 1

Split the data, stratifying on the outcome variable, survived. You should choose the proportions to split the data into. Verify that the training and testing data sets have the appropriate number of observations. Take a look at the training data and note any potential issues, such as missing data.

Why is it a good idea to use stratified sampling for this data?

Answer Q1

```r
# set a seed
set.seed(0623)

# split the titanic data into a training set and a testing set.
titanic_split <- initial_split(titanic, prop = 0.80, strata = survived)
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)
dim(titanic)
```

```
## [1] 891  12
```

```r
dim(titanic_train)
```

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

```
dim(titanic_test)
```

```
## [1] 179  12
```

```
# Verify the training and testing data sets have the appropriate number of observations
# the number of observations for all data
a <- nrow(titanic)
a
```

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

```
# the number of observations for training data
b <- nrow(titanic_train)
b
```

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

```
# the number of observations for test data
c <- nrow(titanic_test)
c
```

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

```
# the percentage of observations for training data
b/a
```

```
## [1] 0.7991021
```

```
# the percentage of observations for test data
c/a
```

```
## [1] 0.2008979
```

The probability of training data observations is 0.7991021, which is almost equal to prob=0.80, so the training and testing data sets have the appropriate number of observations

```
# Take a look at the training data and note any potential issues, such as missing data.
sum (is.na(titanic_train))
```

```
## [1] 688
```

We can find that there are 688 missing data in the training data, so they will have some potential effects.

```
# Why is it a good idea to use stratified sampling for this data?
```

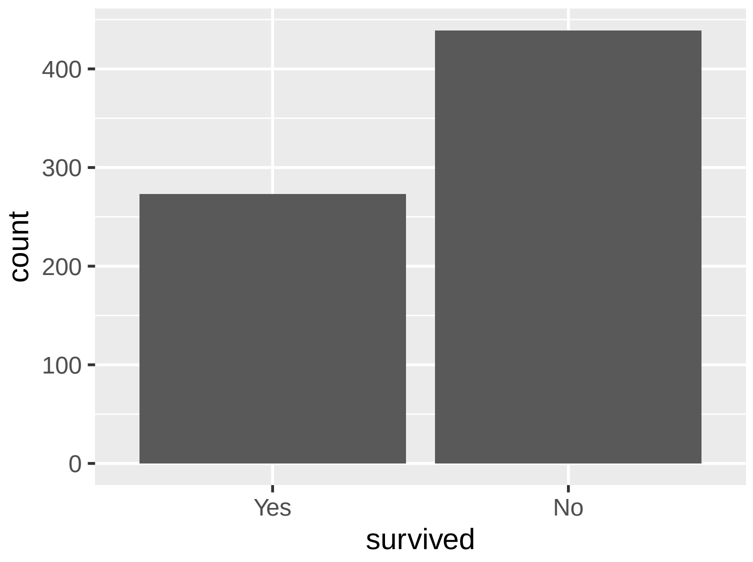
Using stratified sampling us a good idea since the goal is classification to predict which passengers would survive in the Titanic. It allows us to obtain a sample observation that best represents the entire observation being studied. It involves dividing the entire population into homogeneous groups with strata "survived", and it involves the random selection of data from an entire observation, so each possible sample is equally likely to occur. (some definitions of stratified sampling from google)

Question 2

Using the training data set, explore/describe the distribution of the outcome variable survived.

Answer Q2

```
titanic_train %>%
  ggplot(aes(x = survived)) +
  geom_bar()
```



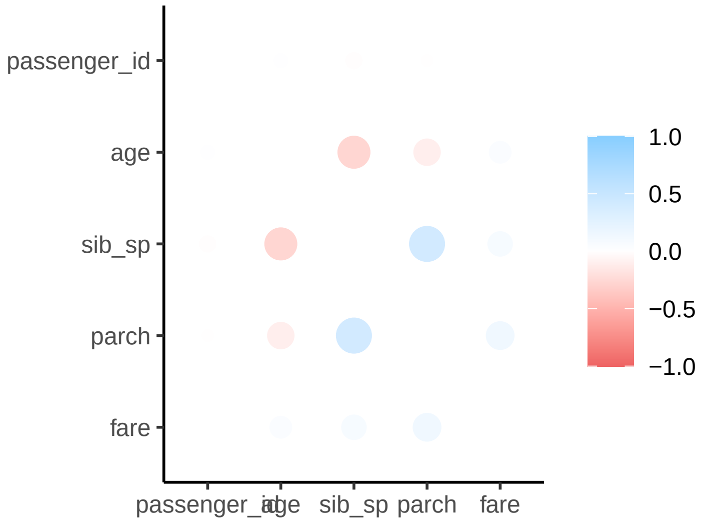
From the histogram above, we can find the number of people who did not survive from the titanic is over 400, which is more than the number of survived people, which is less than 300.

Question 3

Using the training data set, create a correlation matrix of all continuous variables. Create a visualization of the matrix, and describe any patterns you see. Are any predictors correlated with each other? Which ones, and in which direction?

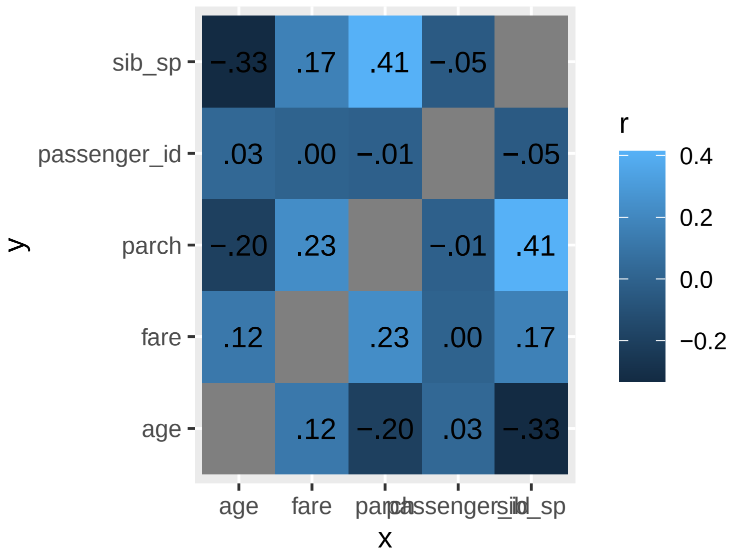
Answer Q3

```
# # create a correlation matrix of all continuous variables. create a visualization of the matrix
cor_titanic <- titanic_train %>%
  select(where(is.numeric)) %>%
  correlate()
rplot(cor_titanic)
```



```
cor_titanic %>%
  stretch() %>%
  ggplot(aes(x, y, fill = r)) +
```

```r
  geom_tile() +
  geom_text(aes(label = as.character(fashion(r))))
```



From the matrix above, we can find that

age and sib_sp has negative correlation with -0.33.

parch and sib_sp has positive correlation with 0.41.

age and parch has slightly negative correlation with -0.20.

fare and parch has slightly positive correlation with 0.23.

Others correlation between predictors (age and fare / age and passenger_id / fare and passenger_id / fare and sib_sp / parch and passenger_id / sib_sp and passenger_id) almost have no correlation. We can find passenger_id almost have no correlation with any predictors.

Question 4

Using the training data, create a recipe predicting the outcome variable survived. Include the following predictors: ticket class, sex, age, number of siblings or spouses aboard, number of parents or children aboard, and passenger fare.

Recall that there were missing values for age. To deal with this, add an imputation step using step_impute_linear(). Next, use step_dummy() to dummy encode categorical predictors. Finally, include interactions between:

Sex and passenger fare, and Age and passenger fare. You'll need to investigate the tidymodels documentation to find the appropriate step functions to use.

Answer Q4

```r
# create a recipe predicting the outcome variable survived
titanic_recipe <- recipe(survived ~ pclass + sex + age + sib_sp + parch + fare,
                         data = titanic_train) %>%
  step_impute_linear(age) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(terms = ~ starts_with("sex"):fare) %>%
  step_interact(terms = ~ age:fare)
titanic_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 5

Specify a logistic regression model for classification using the "glm" engine. Then create a workflow. Add your model and the appropriate recipe. Finally, use fit() to apply your workflow to the training data.

Hint: Make sure to store the results of fit(). You'll need them later on.

Answer Q5

```r
# Specify a logistic regression model for classification using the "glm" engine
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

# create a workflow
log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(titanic_recipe)

# use fit() to apply your workflow to the training data
log_fit <- fit(log_wkflow, titanic_train)
```

Question 6

Repeat Question 5, but this time specify a linear discriminant analysis model for classification using the "MASS" engine.

Answer Q6

```r
# specify a linear discriminant analysis model for classification using the "MASS" engine
lda_mod <- discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")

lda_wkflow <- workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(titanic_recipe)

lda_fit <- fit(lda_wkflow, titanic_train)
```

Question 7

Repeat Question 5, but this time specify a quadratic discriminant analysis model for classification using the “MASS” engine.

Answer Q7

```r
# specify a quadratic discriminant analysis model for classification using the "MASS" engine
qda_mod <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

qda_wkflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(titanic_recipe)

qda_fit <- fit(qda_wkflow, titanic_train)
```

Question 8

Repeat Question 5, but this time specify a naive Bayes model for classification using the “klaR” engine. Set the usekernel argument to FALSE.

Answer Q8

```r
# specify a naive Bayes model for classification using the "klaR" engine.
nb_mod <- naive_Bayes() %>%
  set_mode("classification") %>%
  set_engine("klaR") %>%
  # Set the usekernel argument to FALSE
  set_args(usekernel = FALSE)

nb_wkflow <- workflow() %>%
  add_model(nb_mod) %>%
  add_recipe(titanic_recipe)

nb_fit <- fit(nb_wkflow, titanic_train)
```

Question 9

Now you’ve fit four different models to your training data.

Use predict() and bind_cols() to generate predictions using each of these 4 models and your training data. Then use the accuracy metric to assess the performance of each of the four models.

Which model achieved the highest accuracy on the training data?

Answer Q9

```r
# logistic regression model
predict(log_fit, new_data = titanic_train, type = "prob")
```

```
## # A tibble: 712 x 2
##    .pred_Yes .pred_No
##        <dbl>    <dbl>
##  1    0.0986    0.901
##  2    0.0699    0.930
##  3    0.0985    0.902
##  4    0.281     0.719
##  5    0.102     0.898
##  6    0.163     0.837
##  7    0.0166    0.983
##  8    0.790     0.210
##  9    0.0616    0.938
## 10    0.481     0.519
## # ... with 702 more rows
```

```
log_reg_acc <- augment(log_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
log_reg_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.827
```

```
# linear discriminant analysis model
predict(lda_fit, new_data = titanic_train, type = "prob")
```

```
## # A tibble: 712 x 2
##     .pred_Yes .pred_No
##         <dbl>    <dbl>
##  1   0.0578      0.942
##  2   0.0385      0.961
##  3   0.0548      0.945
##  4   0.213       0.787
##  5   0.0689      0.931
##  6   0.0926      0.907
##  7   0.00920     0.991
##  8   0.853       0.147
##  9   0.0451      0.955
## 10   0.584       0.416
## # ... with 702 more rows
```

```
lda_acc <- augment(lda_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
lda_acc
```

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

```
# quadratic discriminant analysis model
predict(qda_fit, new_data = titanic_train, type = "prob")
```

```
## # A tibble: 712 x 2
##       .pred_Yes .pred_No
##           <dbl>    <dbl>
##  1 0.00616        0.994
##  2 0.00417        0.996
##  3 0.00590        0.994
##  4 0.0579         0.942
##  5 0.000134       1.00
##  6 0.0104         0.990
##  7 0.00614        0.994
##  8 0.631          0.369
##  9 0.000000742    1.00
## 10 0.333          0.667
## # ... with 702 more rows
```

```
qda_acc <- augment(qda_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
qda_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.813
```

```
# a naive Bayes model
predict(nb_fit, new_data = titanic_train, type = "prob")
```

```
## # A tibble: 712 x 2
##     .pred_Yes .pred_No
##         <dbl>    <dbl>
##  1 0.0185        0.981
##  2 0.0178        0.982
##  3 0.0187        0.981
##  4 0.339         0.661
##  5 0.000244      1.00
##  6 0.0214        0.979
##  7 0.00191       0.998
##  8 0.551         0.449
##  9 0.00000233    1.00
## 10 0.451         0.549
## # ... with 702 more rows
```

```
nb_acc <- augment(nb_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
nb_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.802
```

```
# Use predict() and bind_cols() to generate predictions using each of these 4 models and your training
bind_train_data = bind_cols(predict(log_fit, titanic_train),
                            predict(lda_fit, titanic_train),
                            predict(qda_fit, titanic_train),
                            predict(nb_fit, titanic_train))
bind_train_data
```

```
## # A tibble: 712 x 4
##    .pred_class...1 .pred_class...2 .pred_class...3 .pred_class...4
##    <fct>           <fct>           <fct>           <fct>
##  1 No              No              No              No
##  2 No              No              No              No
##  3 No              No              No              No
##  4 No              No              No              No
##  5 No               No               No               No
##  6 No               No               No               No
##  7 No               No               No               No
##  8 Yes              Yes              Yes              Yes
##  9 No               No               No               No
## 10 No               Yes              No               No
## # ... with 702 more rows
```

```r
# compare model performance
# make a table of the accuracy rates from these four models to choose the model that produced the highe
accuracies <- c(log_reg_acc$.estimate, lda_acc$.estimate,
                qda_acc$.estimate, nb_acc$.estimate)
models <- c("Logistic Regression", "LDA", "QDA", "Naive Bayes")
results <- tibble(accuracies = accuracies, models = models)
results %>%
  arrange(-accuracies)
```

```
## # A tibble: 4 x 2
##   accuracies models
##        <dbl> <chr>
## 1      0.827 Logistic Regression
## 2      0.813 QDA
## 3      0.810 LDA
## 4      0.802 Naive Bayes
```

From the table above, we can find that logistic regression model achieved the highest accuracy on the training data.

Question 10

Fit the model with the highest training accuracy to the testing data. Report the accuracy of the model on the testing data.

Again using the testing data, create a confusion matrix and visualize it. Plot an ROC curve and calculate the area under it (AUC).

How did the model perform? Compare its training and testing accuracies. If the values differ, why do you think this is so?

Answer Q10

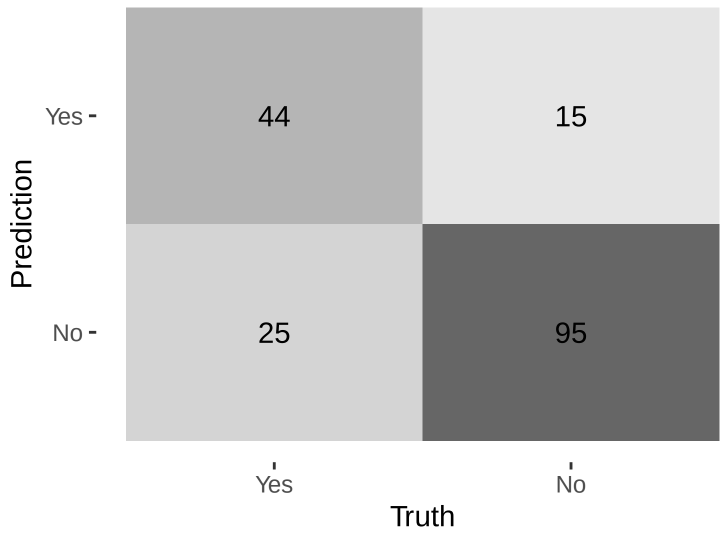
```r
# fit the model with the highest training accuracy to the testing data. Report the accuracy of the mode
predict(log_fit, new_data = titanic_test, type = "prob")
```

```
## # A tibble: 179 x 2
##    .pred_Yes .pred_No
##        <dbl>    <dbl>
##  1    0.548    0.452
##  2    0.913    0.0871
##  3    0.565    0.435
##  4    0.644    0.356
##  5    0.561    0.439
##  6    0.187    0.813
##  7    0.309    0.691
##  8    0.0730   0.927
##  9    0.591    0.409
## 10    0.0989   0.901
## # ... with 169 more rows
```

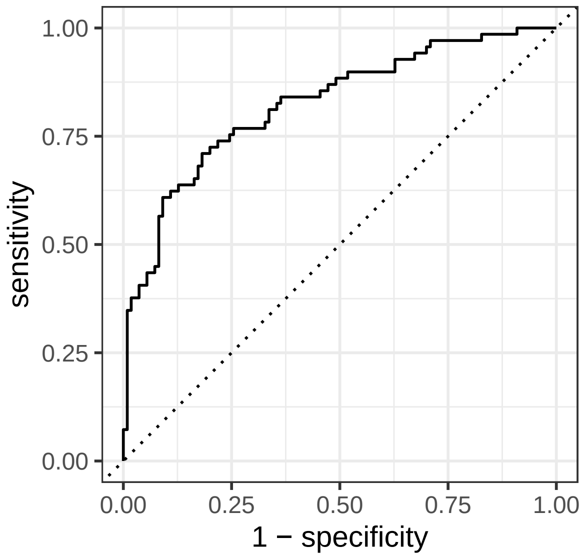
```
log_reg_acc_test <- augment(log_fit, new_data = titanic_test) %>%
  accuracy(truth = survived, estimate = .pred_class)
log_reg_acc_test
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.777
```

```
# using the testing data, create a confusion matrix and visualize it.
augment(log_fit, new_data = titanic_test) %>%
  conf_mat(truth = survived, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



```
# Plot an ROC curve
augment(log_fit, new_data = titanic_test) %>%
  roc_curve(survived, .pred_Yes) %>%
  autoplot()
```



```r
#calculate the area under it (AUC)
augment(log_fit, new_data = titanic_test) %>%
  roc_auc(survived, .pred_Yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.824
```

```r
# How did the model perform? Compare its training and testing accuracies. If the values differ, why do y
```

The training accuracy is 0.827 and the testing roc_accuracy is 0.823 (testing accuracy = 0.7765 is calculated above). Both of them have a high accuracy, so the model perform well to predict on both data sets. They have the slightly difference in the percentage of accuracy, since there are more observations for training data set according to the split the whole data set in the beginning.