131 Homework3

Scott Shang (8458655)

April 19, 2022

Question1

```
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   2.1.2     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        0.1.1      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()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```
library(readr)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(discrim)
```

```
##
## Attaching package: 'discrim'

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

```
library(poissonreg)
library(corrr)
library(klaR)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

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

```
tt=read_csv('titanic.csv')
```

```
## Rows: 891 Columns: 12

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): survived, name, sex, ticket, cabin, embarked
## dbl (6): passenger_id, pclass, age, sib_sp, parch, fare
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
tt$survived=factor(tt$survived)
tt$survived=relevel(tt$survived,"Yes")
tt$pclass=factor(tt$pclass)
head(tt)
```

```
## # 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>
```

```
set.seed(1234)
tt_split=initial_split(tt,prop=0.80,strata=survived)
train=training(tt_split)
test=testing(tt_split)

nrow(tt)
```

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

```
nrow(train)+nrow(test)
```

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

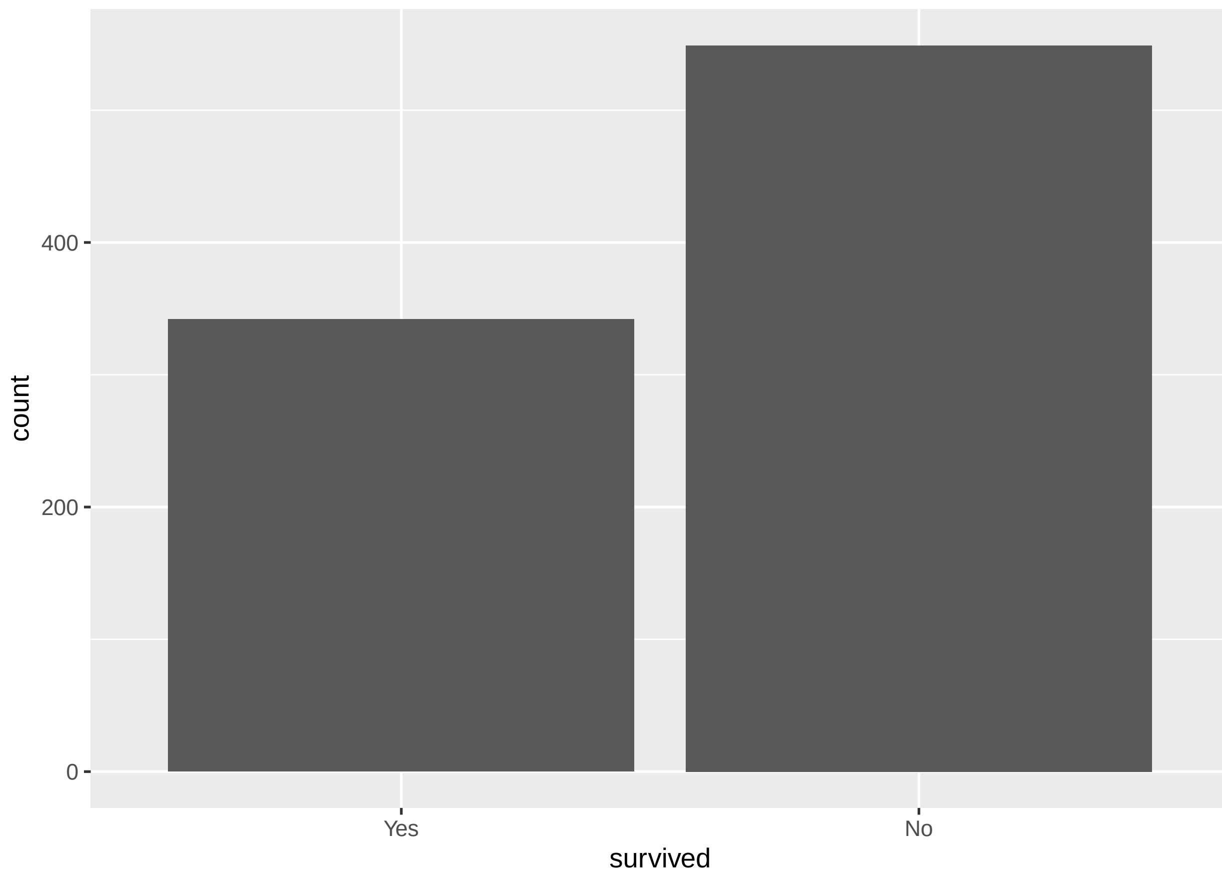
```
colSums(is.na(train))/nrow(train)
```

```
## passenger_id     survived       pclass         name          sex          age
##     0.000000     0.000000     0.000000     0.000000     0.000000     0.191011
##       sib_sp        parch       ticket         fare        cabin     embarked
##     0.000000     0.000000     0.000000     0.000000     0.775281     0.002809
```

We showed that the training and testing data sets have the appropriate number of observations. We saw that we have missing data on predictors: age, cabin, embarked. Specific missing proportion is showed above. We want to use stratified sampling since we are able to stratify our sample by the response variable survive by died or alive and find the difference and relationship between them, and stratified sampling ensures each group of data receive proper representation of each.

Question2

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



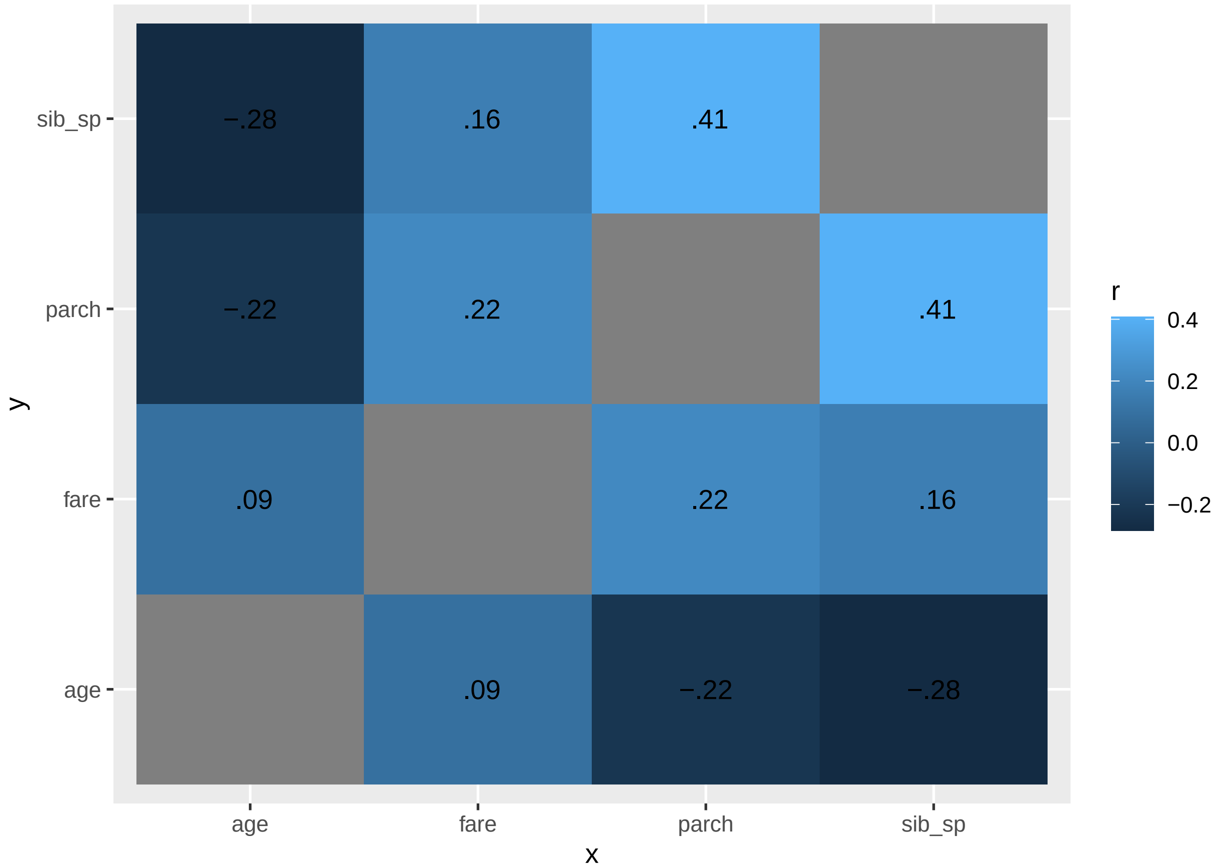
We observe that the outcome variable survived is either Yes or No, with approximately 340 Yes and 570 No. Overall speaking, there are 2/3 more people died comparing to the number of survivor.

Question3

```
cor_tt=train %>%
  dplyr::select("age","sib_sp","parch","fare") %>%
  correlate()
```

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```

```
cor_tt %>%
  stretch() %>%
  ggplot(aes(x, y, fill = r)) +
  geom_tile() +
  geom_text(aes(label = as.character(fashion(r))))
```



We want correlation of continuous variables, which are "age","sib_sp","parch",and "fare". Most correlation are pretty weak. "sib_sp" and "parch" have the strongest one with 0.41 positive correlation, which is not that strong. Then, it's the correlation between "sib_sp" and "age", with -0.28 negative correlation.

Question4

```
tt_recipe=recipe(survived~pclass+sex+age+sib_sp+parch+fare,data=train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_impute_linear(age,impute_with=imp_vars(all_predictors())) %>%
  step_interact(terms=~sex_male:fare+age:fare)
```

Question5

```
log_reg=logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")
log_wf=workflow() %>%
  add_model(log_reg) %>%
  add_recipe(tt_recipe)
log_fit=fit(log_wf,train)
```

Question6

```
lda_mod=discrim_linear() %>%
  set_mode("classification") %>%
```

```
  set_engine("MASS")
lda_wf=workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(tt_recipe)
lda_fit=fit(lda_wf,train)
```

Question7

```
qda_mod=discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")
qda_wf=workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(tt_recipe)
qda_fit=fit(qda_wf,train)
```

Question8

```
nb_mod=naive_Bayes() %>%
  set_mode("classification") %>%
  set_engine("klaR") %>%
  set_args(usekernel=FALSE)
nb_wf=workflow() %>%
  add_model(nb_mod) %>%
  add_recipe(tt_recipe)
nb_fit=fit(nb_wf,train)
```

Question 9

```
predictlog=predict(log_fit,new_data=train)
predictlog=bind_cols(predictlog)
predictlda=predict(lda_fit,new_data=train)
predictlda=bind_cols(predictlda)
predictqda=predict(qda_fit,new_data=train)
predictqda=bind_cols(predictqda)
predictnb=predict(nb_fit,new_data=train)
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 702

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 703

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 704

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 705

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 706

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 707

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 708

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 709

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 710

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 711

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 712
```

```
accuracy=c(log_acc$.estimate,lda_acc$.estimate,             nb_acc$.estimate,qda_acc$.estimate)

mods=c("Logistic Regression","LDA","Naive Bayes","QDA")
results=tibble(accuracy=accuracy,mods=mods)
results %>%
  arrange(-accuracy)
```

```
## # A tibble: 4 x 2
##   accuracy mods
##      <dbl> <chr>
## 1    0.819 Logistic Regression
## 2    0.803 LDA
## 3    0.789 QDA
## 4    0.784 Naive Bayes
```

From the data above, we notice that the logistic regression has the highest accuracy on training data.

Question 10

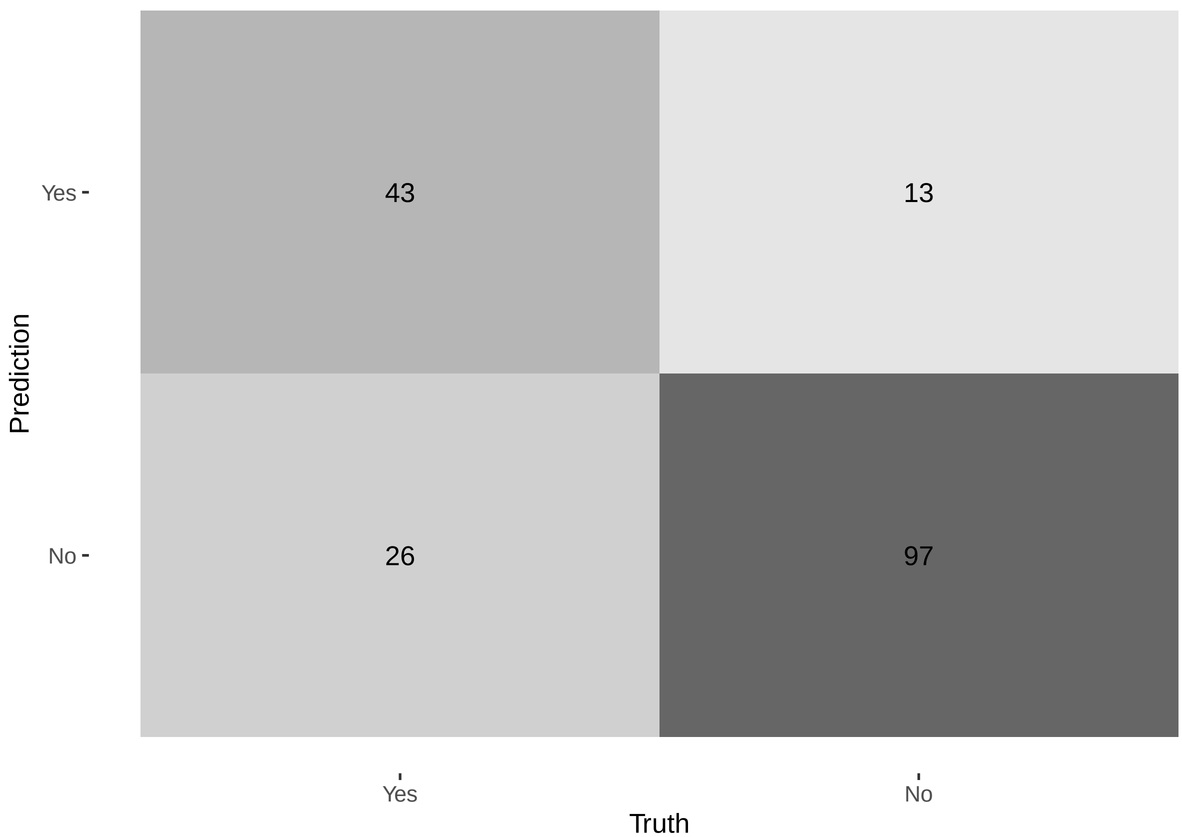
```
new_log_reg=augment(log_fit,new_data=test) %>%
  accuracy(truth=survived,estimate=.pred_class)
new_log_reg
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.782
```

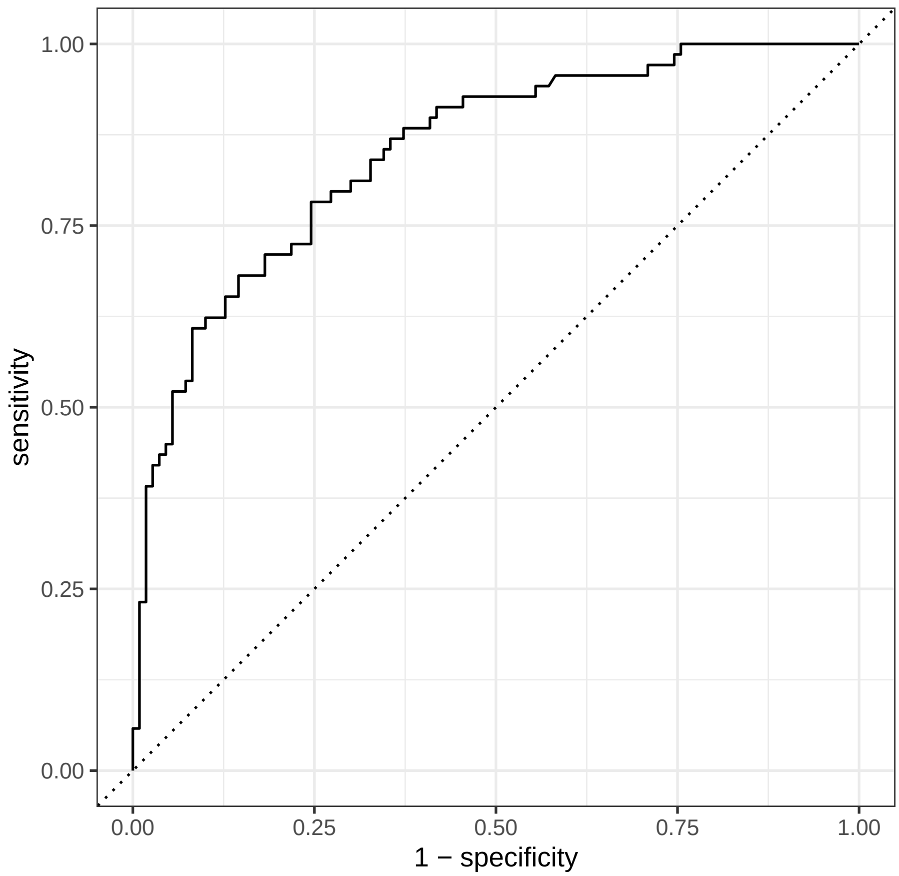
```
augment(log_fit,new_data=test) %>%
  conf_mat(truth=survived,estimate=.pred_class)
```

```
##           Truth
## Prediction Yes No
##        Yes  43 13
##        No   26 97
```

```
augment(log_fit,new_data=test) %>%
  conf_mat(truth=survived,estimate=.pred_class) %>%
  autoplot(type="heatmap")
```



```
augment(log_fit,new_data=test) %>%
  roc_curve(truth=survived,.pred_Yes) %>%
  autoplot()
```




```
augment(log_fit,new_data=test) %>%
  roc_auc(truth=survived,.pred_Yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.850
```

AUC=0.8503

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

I used the logistic regression model. It has 0.8188 accuracy on the training data and 0.7821 accuracy on the testing data, which is slightly worse. We might overfit the model, but overall speaking, it's acceptable.