Homework 3

PSTAT 131/231

Contents

Classification

For this assignment, we will be working with part of a Kaggle data set that was the subject of a machine learning competition and is often used for practicing ML models. The goal is classification; specifically, to predict which passengers would survive the Titanic shipwreck.

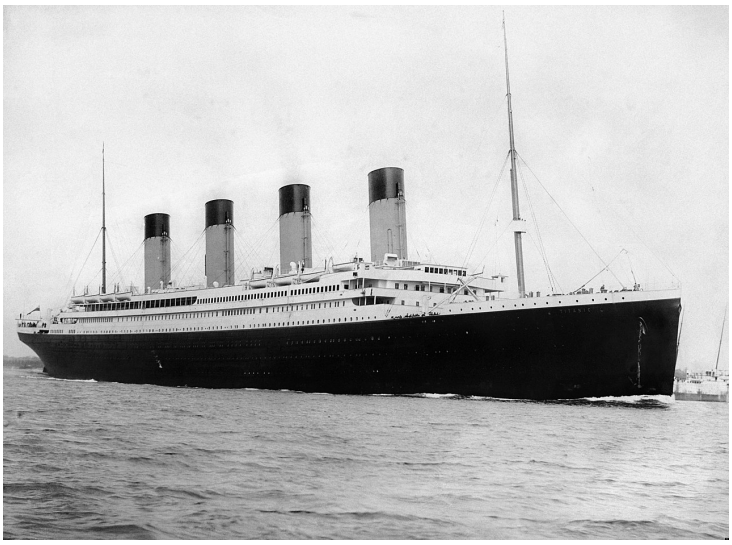


Figure 1: Fig. 1: RMS Titanic departing Southampton on April 10, 1912.

Load the data from `data/titanic.csv` into *R* and familiarize yourself with the variables it contains using the codebook (`data/titanic_codebook.txt`).

Notice that `survived` and `pclass` should be changed to factors. When changing `survived` to a factor, you may want to reorder the factor so that *"Yes"* is the first level.

Make sure you load the `tidyverse` and `tidymodels`!

Remember that you'll need to set a seed at the beginning of the document to reproduce your results.

```
library(tidyverse)
library(tidymodels)
library(rlang)
library(corrr)
```

```
library(klaR)
library(discrim)
library(poissonreg)

#library(ISLR) # For the Smarket data set
#library(ISLR2) # For the Bikeshare data set
tidymodels_prefer()

set.seed(22)

# Load Data
rawData <- read.csv("data/titanic.csv")
head(rawData)
```

```
##   passenger_id survived pclass
## 1            1       No      3
## 2            2      Yes      1
## 3            3      Yes      3
## 4            4      Yes      1
## 5            5       No      3
## 6            6       No      3
##                                                 name    sex age sib_sp parch
## 1                             Braund, Mr. Owen Harris   male  22      1     0
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38      1     0
## 3                              Heikkinen, Miss. Laina female  26      0     0
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35      1     0
## 5                            Allen, Mr. William Henry   male  35      0     0
## 6                                    Moran, Mr. James   male  NA      0     0
##             ticket    fare cabin embarked
## 1        A/5 21171  7.2500  <NA>        S
## 2         PC 17599 71.2833   C85        C
## 3 STON/O2. 3101282  7.9250  <NA>        S
## 4           113803 53.1000  C123        S
## 5           373450  8.0500  <NA>        S
## 6           330877  8.4583  <NA>        Q
```

```
# Copy dataframe
data <- duplicate(rawData, shallow = FALSE)

#reorder factors
data$survived <- factor(data$survived, levels = c("Yes", "No"))
data$pclass <- factor(data$pclass)

levels(data$survived)
```

```
## [1] "Yes" "No"
```

```
levels(data$pclass)
```

```
## [1] "1" "2" "3"
```

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.

```
# Split the data for training/testing
dataSplit <- data %>%
  initial_split(prop = 0.8,
                strata = survived)

dataTrain <- training(dataSplit)
dataTest <- testing(dataSplit)

#Check dimensions
dim(dataTest)
```

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

```
dim(dataTrain)
```

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

```
head(data)
```

```
##   passenger_id survived pclass
## 1            1       No      3
## 2            2      Yes      1
## 3            3      Yes      3
## 4            4      Yes      1
## 5            5       No      3
## 6            6       No      3
##                                                 name    sex age sib_sp parch
## 1                             Braund, Mr. Owen Harris   male  22      1     0
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38      1     0
## 3                              Heikkinen, Miss. Laina female  26      0     0
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35      1     0
## 5                            Allen, Mr. William Henry   male  35      0     0
## 6                                    Moran, Mr. James   male  NA      0     0
##             ticket    fare cabin embarked
## 1        A/5 21171  7.2500  <NA>        S
## 2         PC 17599 71.2833   C85        C
## 3 STON/O2. 3101282  7.9250  <NA>        S
## 4           113803 53.1000  C123        S
## 5           373450  8.0500  <NA>        S
## 6           330877  8.4583  <NA>        Q
```

```
# Return number of survived and did not
count(data, survived)
```

```
##   survived   n
## 1      Yes 342
## 2       No 549
```

```r
#look at the training data, check for missing values
na_count <-sapply(data, function(y) sum(is.na(y)))
na_count
```

```
## passenger_id     survived       pclass         name          sex          age
##            0            0            0            0            0          177
##       sib_sp        parch       ticket         fare        cabin     embarked
##            0            0            0            0          687            2
```

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

We want to use stratified sampling for this data because there is an uneven proportion of passengers who survived and who died. Thus we want the models we train to train on the correct proportion of survivors and non-survivors relative to the total dataset.

Question 2

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

```r
counts <- data.frame(count(dataTrain, survived))
counts
```

```
##   survived   n
## 1      Yes 273
## 2       No 439
```

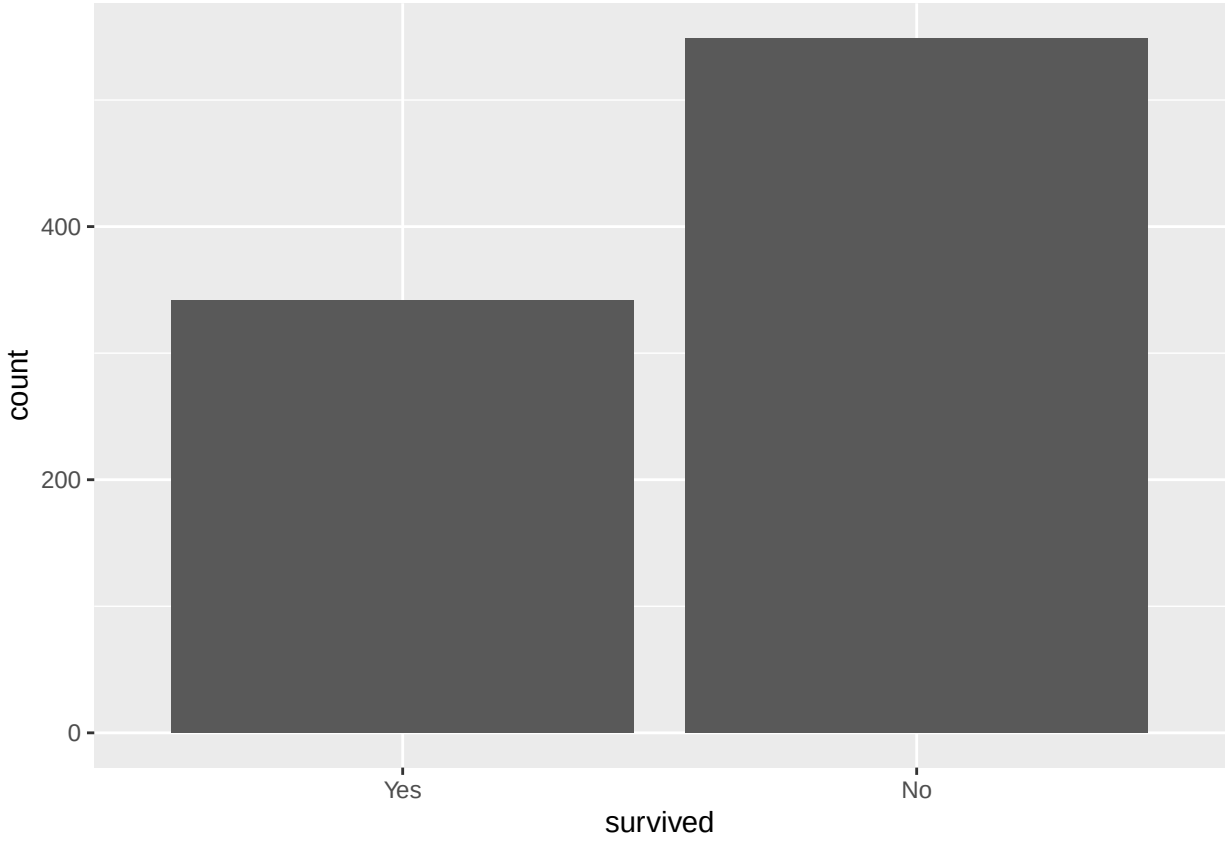
```r
sum(counts$n)
```

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

```r
counts %>% mutate(percentage = n/sum(n))
```

```
##   survived   n percentage
## 1      Yes 273   0.383427
## 2       No 439   0.616573
```

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



Out of 712 recorded passengers on the Titanic, 439 or 61.6% did not survive, while 273 (38.3%) did survive.

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?

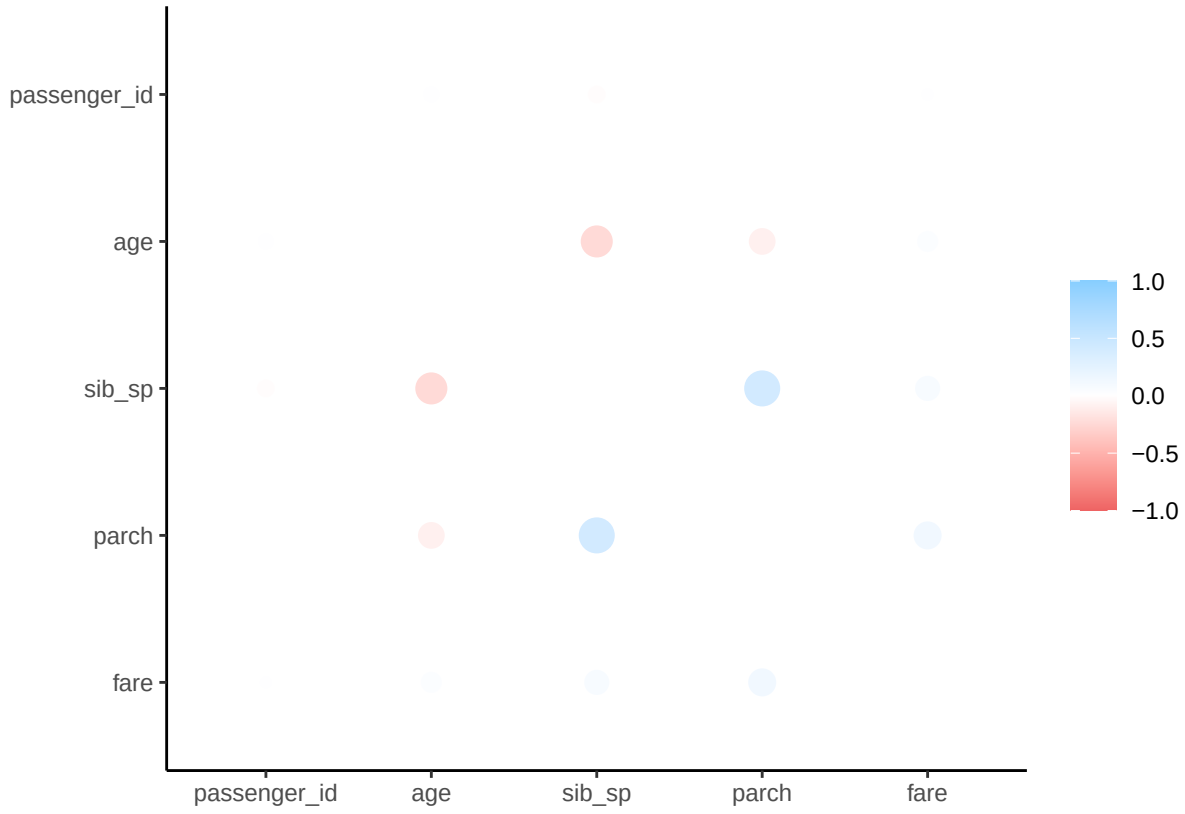
We see the pairs (age, sib_sp) and (age, parch) slightly negatively correlated, and (sib_sp, parch), (fare, sib_sp), and (fare„parch) slightly positively correlated.

```
names(dataTrain)
```

```
##  [1] "passenger_id" "survived"     "pclass"       "name"         "sex"
##  [6] "age"          "sib_sp"       "parch"        "ticket"       "fare"
## [11] "cabin"        "embarked"
```

```
cor_data <- data %>%
  select(-c(survived, pclass, name, sex, ticket, cabin, embarked)) %>%
  correlate()

rplot(cor_data)
```



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.

```
names(dataTrain)
```

```
##  [1] "passenger_id" "survived"     "pclass"       "name"         "sex"
##  [6] "age"          "sib_sp"       "parch"        "ticket"       "fare"
## [11] "cabin"        "embarked"
```

```
titanicRecipe <- recipe(survived ~ pclass +
                          sex +
                          age +
                          sib_sp +
                          parch +
```

```
                            fare, data = dataTrain) %>%
    step_impute_linear(age) %>%
    step_dummy(all_nominal_predictors()) %>%
    step_interact(terms = ~ sex:fare) %>%
    step_interact(terms = ~ 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.

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


log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(titanicRecipe)

log_fit <- fit(log_wkflow, dataTrain)
```

Question 6

Repeat Question 5, but this time 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(titanicRecipe)

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

Question 7

Repeat Question 5, but this time 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(titanicRecipe)
```

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

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`.

```
nb_mod <- naive_Bayes() %>%
  set_mode("classification") %>%
  set_engine("klaR") %>%
  set_args(usekernel = FALSE)

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

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

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: naive_Bayes()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
## * step_interact()
##
## -- Model -----------------------------------------------------------------------
## $apriori
## grouping
##      Yes       No
## 0.383427 0.616573
##
## $tables
## $tables$age
##         [,1]     [,2]
## Yes 28.07076 13.41796
## No  29.17020 13.30665
##
## $tables$sib_sp
##          [,1]      [,2]
## Yes 0.4798535 0.7177866
## No  0.5717540 1.3340378
##
## $tables$parch
##          [,1]      [,2]
## Yes 0.4505495 0.7707483
## No  0.3234624 0.7852352
```

```
##
## $tables$fare
##         [,1]     [,2]
## Yes 49.94506 70.92922
## No  22.57262 33.23062
##
## $tables$pclass_X2
##          [,1]      [,2]
## Yes 0.2490842 0.4332770
## No  0.1822323 0.3864763
##
## $tables$pclass_X3
##          [,1]      [,2]
## Yes 0.3589744 0.4805807
## No  0.6810934 0.4665845
##
## $tables$sex_male
##          [,1]      [,2]
## Yes 0.3260073 0.4696109
## No  0.8451025 0.3622197
##
## $tables$age_x_fare
##         [,1]     [,2]
## Yes 1598.007 2602.336
## No   682.256 1321.477
##
##
## $levels
## [1] "Yes" "No"
##
##
## ...
## and 727 more lines.
```

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.

Used augment() instead of bind_cols()

Which model achieved the highest accuracy on the training data?

The logistic regression achieved the highest accuracy on the training data, with an accuracy of 80.6%. It was closely followed by the quadratic discriminant analysis model which had an accuracy of 80.4%.

```
predict(log_fit, new_data = dataTrain, type = "prob")
```

```
## # A tibble: 712 x 2
##    .pred_Yes .pred_No
##        <dbl>    <dbl>
##  1    0.0901    0.910
##  2    0.0649    0.935
```

```
##  3    0.0956    0.904
##  4    0.315     0.685
##  5    0.143     0.857
##  6    0.780     0.220
##  7    0.0699    0.930
##  8    0.469     0.531
##  9    0.203     0.797
## 10    0.564     0.436
## # ... with 702 more rows
```

```
predict(lda_fit, new_data = dataTrain, type = "prob")
```

```
## # A tibble: 712 x 2
##    .pred_Yes .pred_No
##        <dbl>    <dbl>
##  1    0.0611    0.939
##  2    0.0434    0.957
##  3    0.0635    0.936
##  4    0.257     0.743
##  5    0.0948    0.905
##  6    0.817     0.183
##  7    0.0513    0.949
##  8    0.549     0.451
##  9    0.151     0.849
## 10    0.644     0.356
## # ... with 702 more rows
```

```
predict(qda_fit, new_data = dataTrain, type = "prob")
```

```
## # A tibble: 712 x 2
##     .pred_Yes .pred_No
##         <dbl>    <dbl>
##  1 0.0264        0.974
##  2 0.0175        0.982
##  3 0.0283        0.972
##  4 0.215         0.785
##  5 0.0461        0.954
##  6 0.598         0.402
##  7 0.00000249    1.00
##  8 0.542         0.458
##  9 0.0351        0.965
## 10 0.00375       0.996
## # ... with 702 more rows
```

```
predict(nb_fit, new_data = dataTrain, type = "prob")
```

```
## # A tibble: 712 x 2
##     .pred_Yes .pred_No
##         <dbl>    <dbl>
##  1 0.0233        0.977
##  2 0.0230        0.977
##  3 0.0240        0.976
```

```
##  4 0.339         0.661
##  5 0.0254        0.975
##  6 0.455         0.545
##  7 0.00000729    1.00
##  8 0.403         0.597
##  9 0.194         0.806
## 10 0.0132        0.987
## # ... with 702 more rows
```

```
# assess accuracy of logistic regression model
augment(log_fit, new_data = dataTrain) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##           Truth
## Prediction Yes  No
##        Yes 196  61
##        No   77 378
```

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

log_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.806
```

```
# assess accuracy of lda model
augment(lda_fit, new_data = dataTrain) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##           Truth
## Prediction Yes  No
##        Yes 190  67
##        No   83 372
```

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

lda_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.789
```

```
# assess accuracy of qda model
augment(qda_fit, new_data = dataTrain) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##           Truth
## Prediction Yes  No
##        Yes 197  63
##        No   76 376
```

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

qda_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.805
```

```
# assess accuracy of naive bayes model
augment(nb_fit, new_data = dataTrain) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##           Truth
## Prediction Yes  No
##        Yes 144  37
##        No  129 402
```

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

nb_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.767
```

Question 10

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

The model predicts the survival of persons on the titanic in the testing dataset with an accuracy of 84.9%.

```
multiMetric <- metric_set(accuracy, sensitivity, specificity)

augment(log_fit, new_data = dataTest) %>%
  multiMetric(truth = survived, estimate = .pred_class)
```

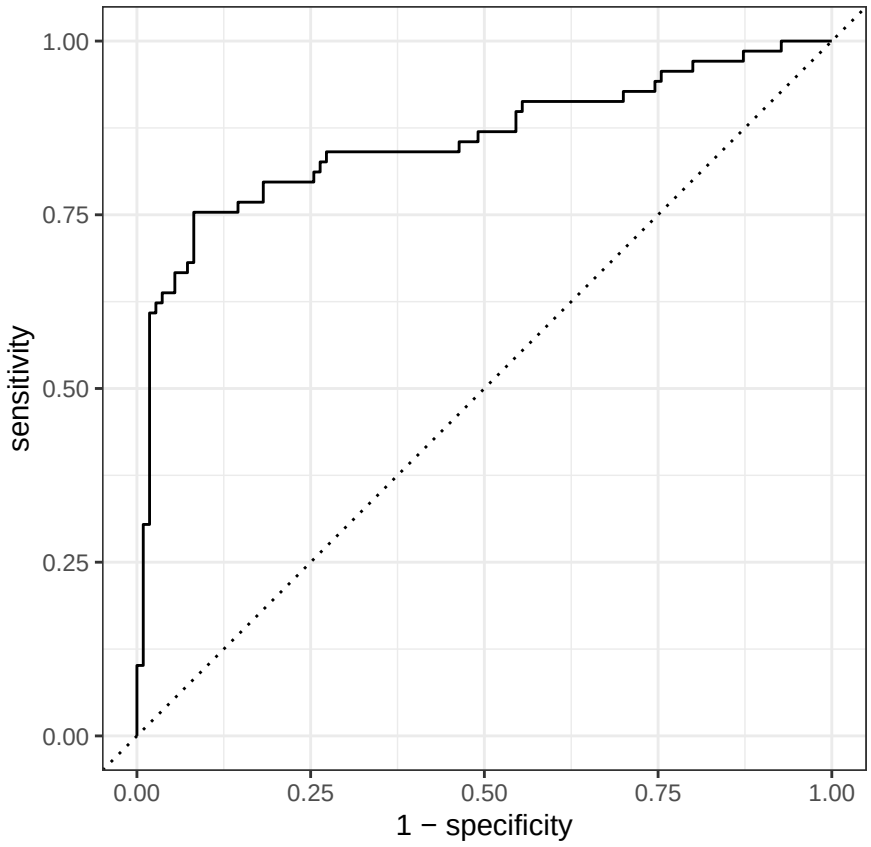
```
## # A tibble: 3 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 accuracy    binary         0.849
## 2 sensitivity binary         0.739
## 3 specificity binary         0.918
```

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

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

```
##           Truth
## Prediction Yes  No
##        Yes  51   9
##        No   18 101
```

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




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

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.857
```

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

*The model performed pretty well. Its training accuracy was 80.6%, and its testing accuracy was 84.9%. This is counterintuitive, because I would believe that the model would be more likely to better describe the training data. The difference may be due to random chance, or perhaps passengers in the test data set were more clear-cut as likely to be survivors than those in the training set.

Required for 231 Students

In a binary classification problem, let p represent the probability of class label 1, which implies that $1 - p$ represents the probability of class label 0. The *logistic function* (also called the "inverse logit") is the cumulative distribution function of the logistic distribution, which maps a real number z to the open interval $(0, 1)$.

Question 11

Given that:

$$p(z) = \frac{e^z}{1 + e^z}$$

Prove that the inverse of a logistic function is indeed the *logit* function:

$$z(p) = ln\left(\frac{p}{1 - p}\right)$$

Question 12

Assume that $z = \beta_0 + \beta_1 x_1$ and $p = logistic(z)$. How do the odds of the outcome change if you increase x_1 by two? Demonstrate this.

Assume now that β_1 is negative. What value does p approach as x_1 approaches ∞? What value does p approach as x_1 approaches $-\infty$?

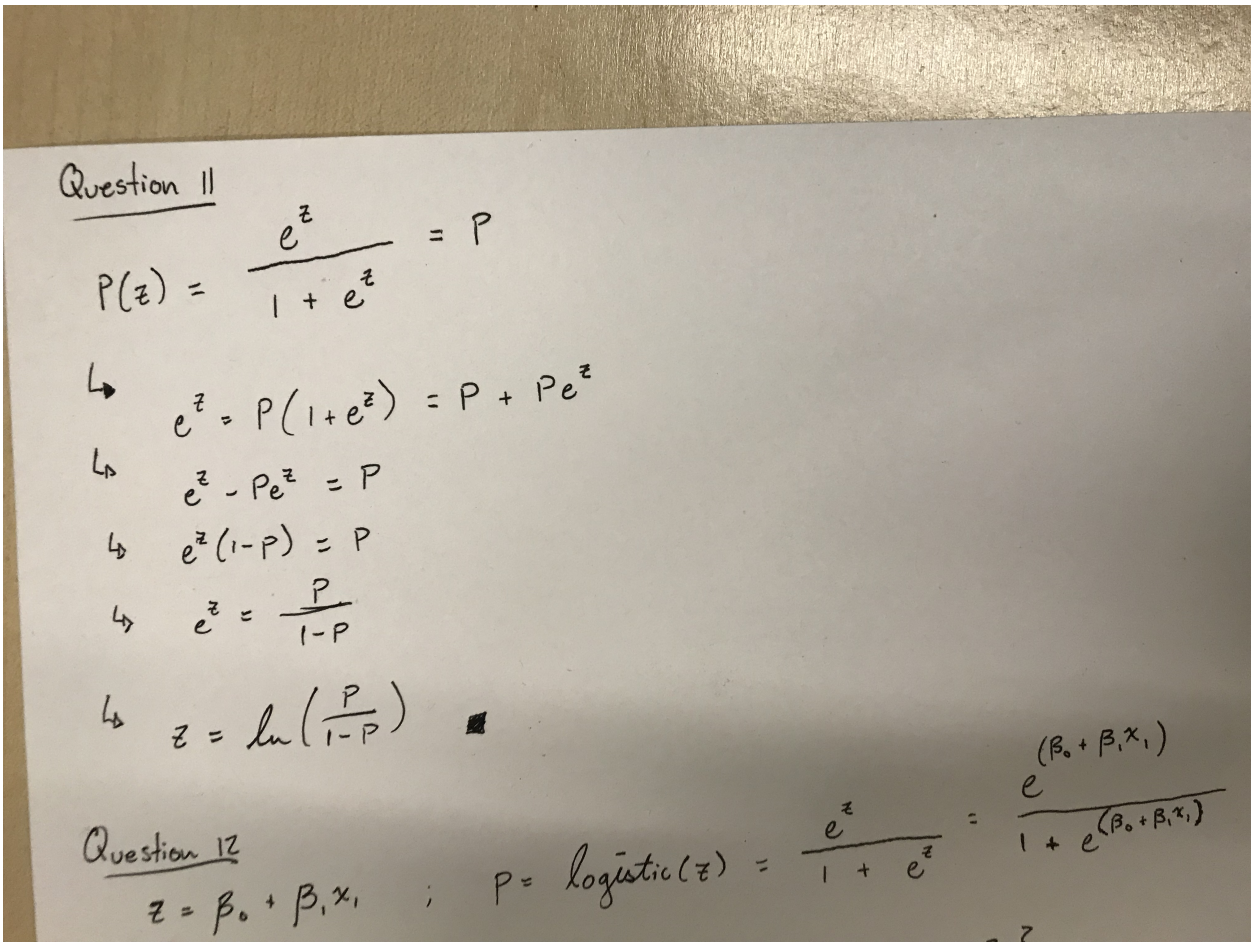


Figure 2: local_image

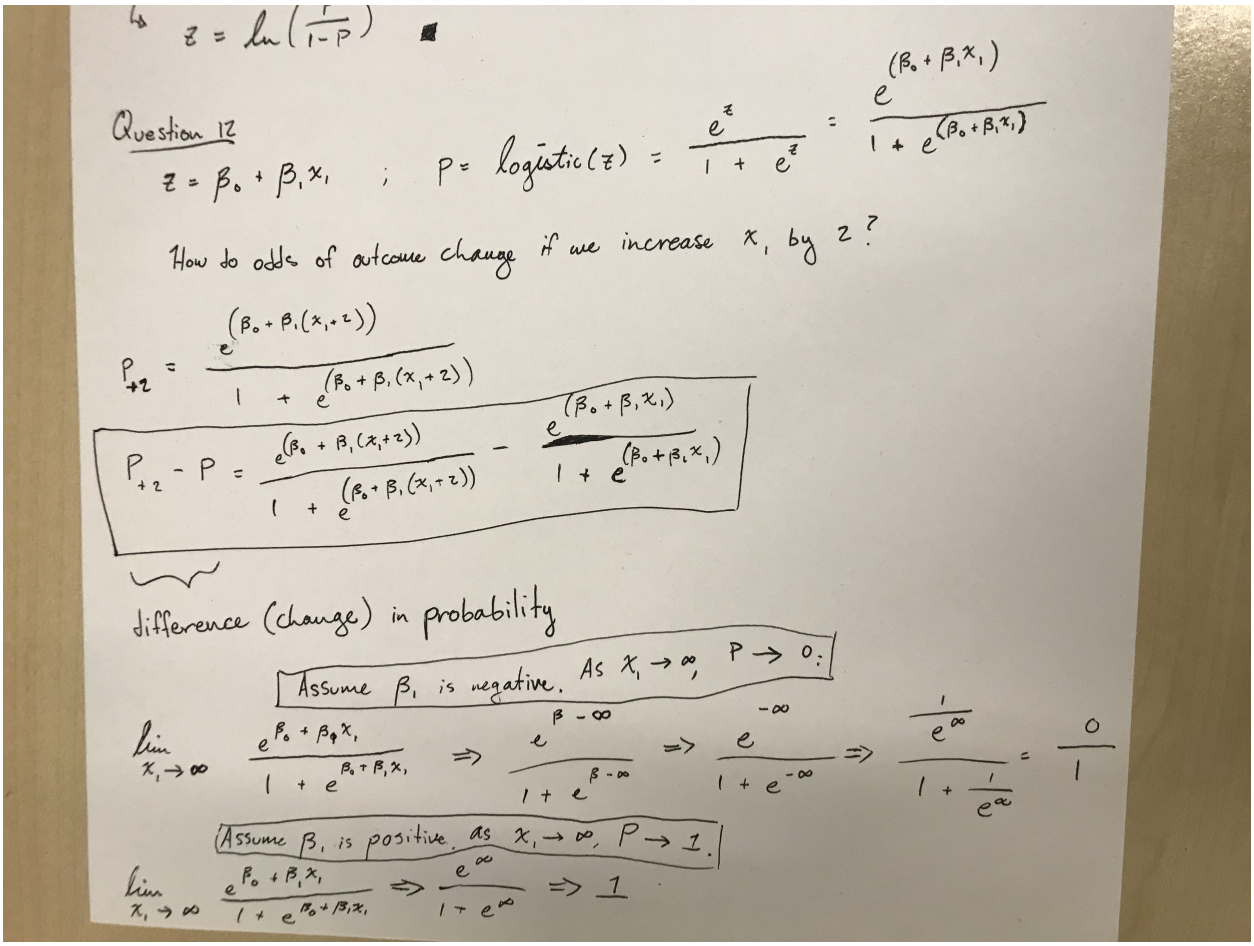


Figure 3: local_image