Homework 3

PSTAT 131/231

Contents

Classification 1

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.

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.

```
set.seed(123)

library(tidymodels)
library(tidyverse)
library(dplyr)
library(ggplot2)
library(corrr)
library(klaR)
library(MASS)
library(discrim)
library(poissonreg)
tidymodels_prefer()

df <- read_csv("titanic.csv")
df$survived <- as.factor(df$survived)
df$survived <- relevel(df$survived, "Yes")
df$pclass <- as.factor(df$pclass)
head(df)
```

```
## # 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.

```
df_split <- initial_split(df, prop = 0.8, strata = survived)
df_train <- training(df_split)
df_test <- testing(df_split)

dim(df)
```

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

```
0.8 * nrow(df)
```

```
## [1] 712.8
```

```
dim(df_train)
```

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

```
dim(df_test)
```

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

```
colSums(is.na(df_train))
```

```
## passenger_id     survived       pclass         name          sex          age
##            0            0            0            0            0          145
##       sib_sp        parch       ticket         fare        cabin     embarked
##            0            0            0            0          549            0
```

It can be seen that there are missing values in the three columns of age, cabin, and embarked, of which the former two are the majority.

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

Stratified sampling increases the commonality among units in various types by classifying and stratifying, and it is easy to draw representative survey samples. This method is suitable for situations where the overall situation is complex, the differences between units are large, and there are many units.

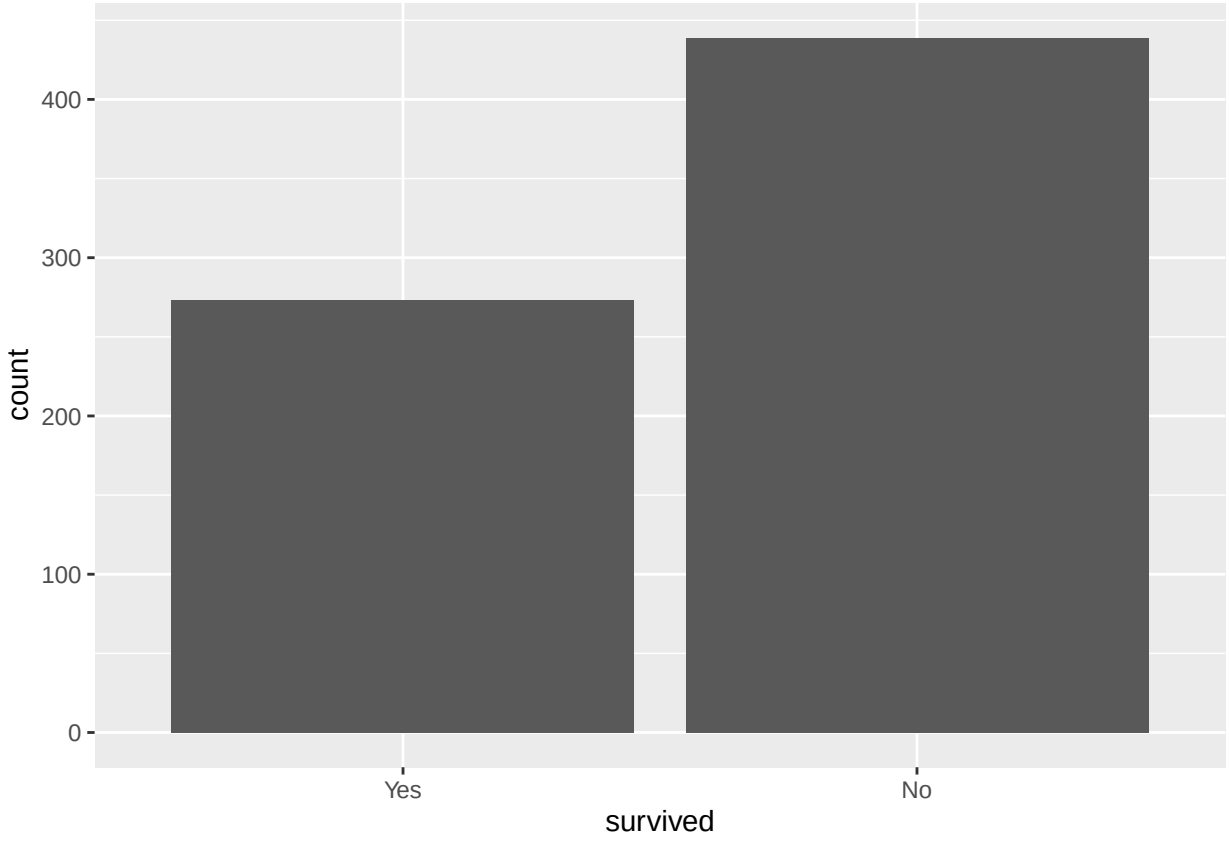
Question 2

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

```
table(df_train$survived)
```

```
##
## Yes  No
## 273 439
```

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



It can be seen that the proportion of survivors is $273/(273+439) = 0.38$, that is, the proportion of non-survivors has around 62%.

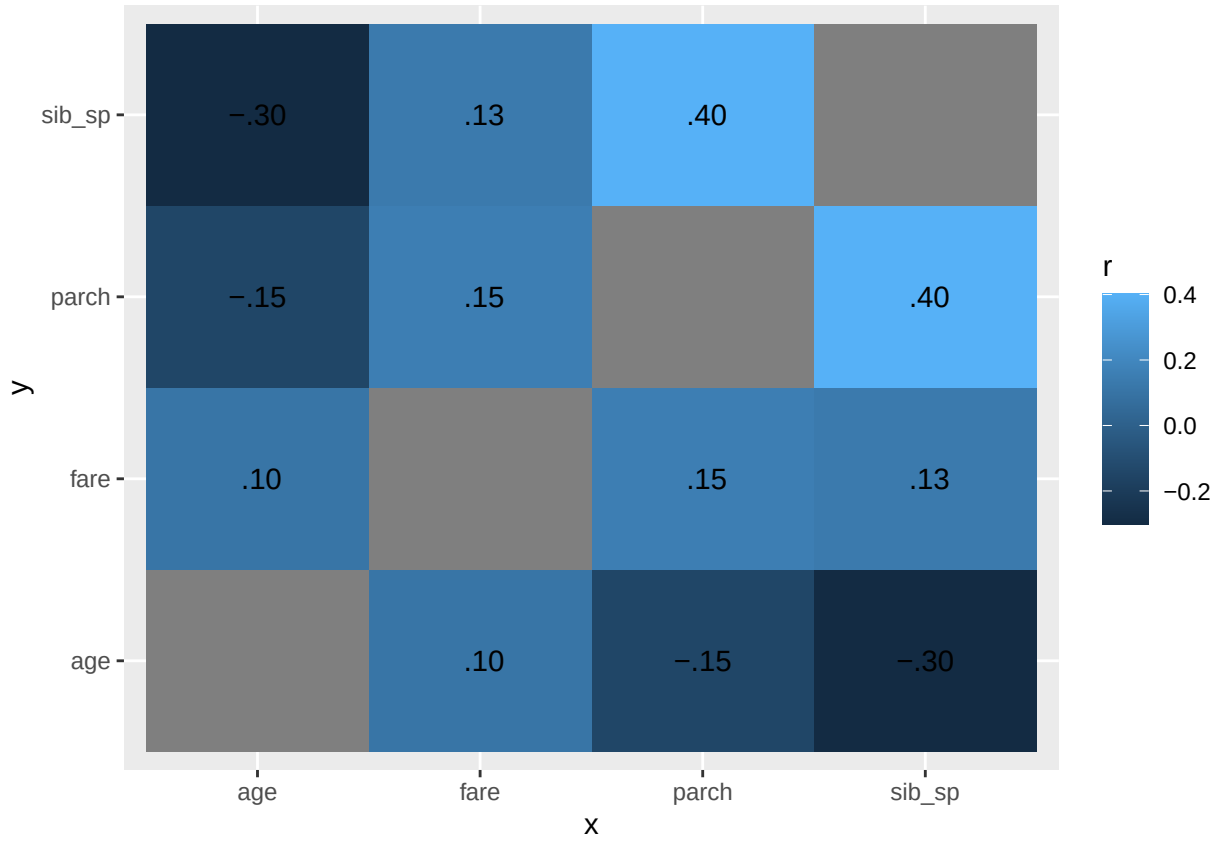
Question 3

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?

```
cor_df <- df_train %>%
  select("age", "sib_sp", "parch", "fare") %>%
  correlate() %>%
  stretch() %>%
  ggplot(aes(x, y, fill = r)) + geom_tile() +
```

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

cor_df
```



passenger id, pclass, name, sex, ticket, and embarked are all discrete/non-numeric variables, and survived is our response variable. Thus, our continuous predictor variables include age, sub sp, parch, and fare. It can be seen that among the continuous variables, only parch and sib_sp have a moderate correlation (<0.5), in fact, these predictors can be considered unrelated to each other. Overall, there is not much correlation between these variables, with the highest correlation being a +0.40 between sib sp and parch, suggesting that as the number of siblings/spouses aboard the titanic increases, the number of parents/children aboard increases moderately as well.

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.

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

imputed_ti
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          6
##
## Operations:
##
## Linear regression imputation for age
## Dummy variables from all_nominal_predictors()
## Interactions with sex_male: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.

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

log_workflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(imputed_ti)

log_fit <- fit(log_workflow, df_train)
```

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_workflow <- workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(imputed_ti)

lda_fit <- fit(lda_workflow, df_train)
```

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_workflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(imputed_ti)

qda_fit <- fit(qda_workflow, df_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`.

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

bayes_workflow <- workflow() %>%
  add_model(bayes_mod) %>%
  add_recipe(imputed_ti)

bayes_fit <- fit(bayes_workflow, df_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?

```
log_pred <- predict(log_fit, new_data = df_train)
log_pred <- bind_cols(log_pred, df_train %>% select(survived))
head(log_pred)
```

```
## # A tibble: 6 x 2
##   .pred_class survived
##   <fct>       <fct>
## 1 No          No
## 2 No          No
## 3 No          No
## 4 No          No
## 5 No          No
## 6 No          No
```

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

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.812
```

```
lda_pred <- predict(lda_fit, new_data = df_train)
lda_pred <- bind_cols(lda_pred, df_train %>% select(survived))
head(lda_pred)
```

```
## # A tibble: 6 x 2
##   .pred_class survived
##   <fct>       <fct>
## 1 No          No
## 2 No          No
## 3 No          No
## 4 No          No
## 5 No          No
## 6 No          No
```

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

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

```
qda_pred <- predict(qda_fit, new_data = df_train)
qda_pred <- bind_cols(qda_pred, df_train %>% select(survived))
head(qda_pred)
```

```
## # A tibble: 6 x 2
##   .pred_class survived
##   <fct>       <fct>
## 1 No          No
## 2 No          No
## 3 No          No
## 4 No          No
## 5 No          No
## 6 No          No
```

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

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.764
```

```
bayes_pred <- predict(bayes_fit, new_data = df_train)
bayes_pred <- bind_cols(bayes_pred, df_train %>% select(survived))
head(bayes_pred)
```

```
## # A tibble: 6 x 2
##   .pred_class survived
##   <fct>       <fct>
## 1 No          No
## 2 No          No
## 3 Yes         No
## 4 No          No
## 5 No          No
## 6 No          No
```

```
bayes_acc <- augment(bayes_fit, new_data = df_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
bayes_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.765
```

It can be seen that the logistic Regression model has the highest accuracy - 0.812 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?

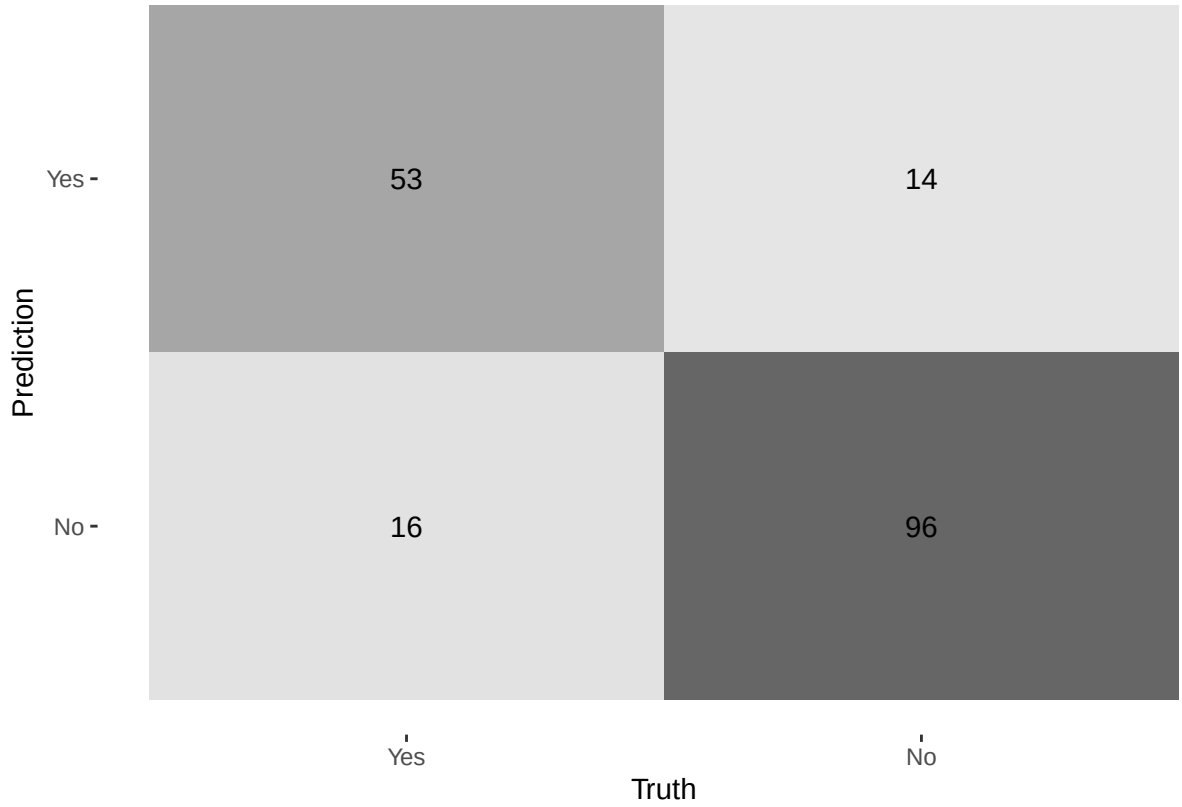
```r
log_fit_test <- fit(log_workflow, df_test)
log_acc_test <- augment(log_fit_test, new_data = df_test) %>%
  accuracy(truth = survived, estimate = .pred_class)
log_acc_test
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.832
```

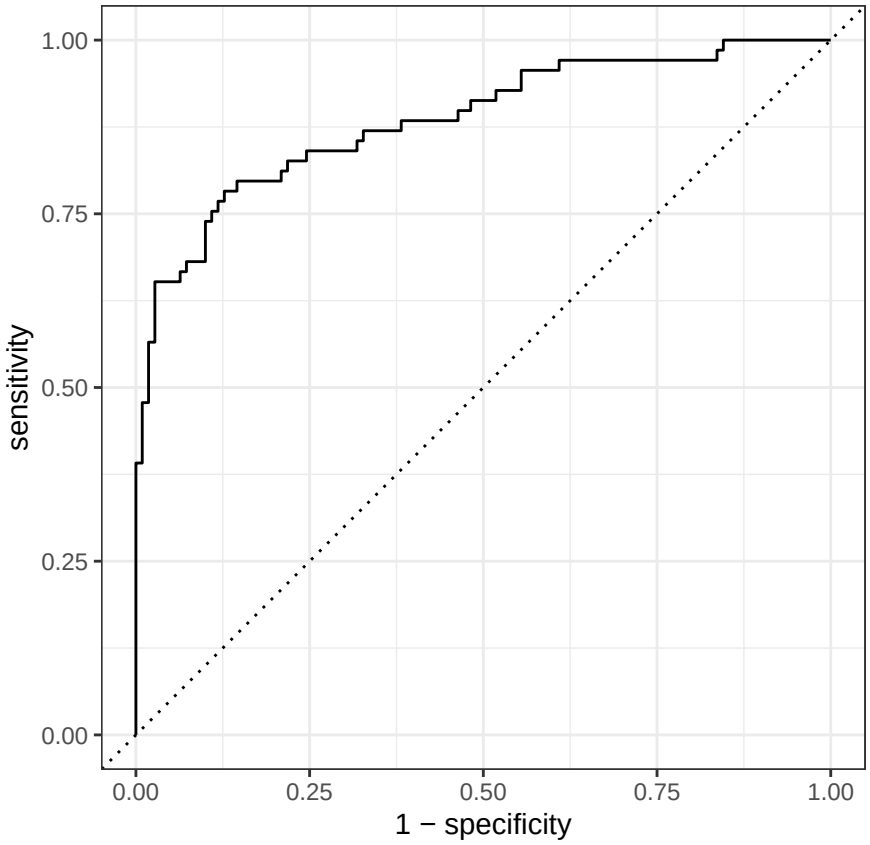
```r
log_confusion_matrix <- augment(log_fit_test, new_data = df_test) %>%
  conf_mat(truth = survived, estimate = .pred_class)
log_confusion_matrix
```

```
##           Truth
## Prediction Yes No
##        Yes  53 14
##        No   16 96
```

```r
log_confusion_matrix %>%
  autoplot(type = "heatmap")
```



```
log_roc <- augment(log_fit_test, new_data = df_test) %>%
  roc_curve(survived, .pred_Yes)
log_roc %>%
  autoplot()
```




```
log_auc <- augment(log_fit_test, new_data = df_test) %>%
  roc_auc(survived, .pred_Yes)
log_auc
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.885
```

The model with the highest training accuracy was logistic regression, so I will use that as my final model. Overall, this is a relatively decent model when looking at the classification accuracy on the test set. The test set performance is slightly stronger than the train set performance, which means that we could probably benefit from more data being used to train. However, because this is a limited dataset , we could do things like cross-validation or changing the train/test split to strengthen our training process.

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 $-\infty$?