

# Homework 3

## PSTAT 131/231

Aarti Garaye

### Binary 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.

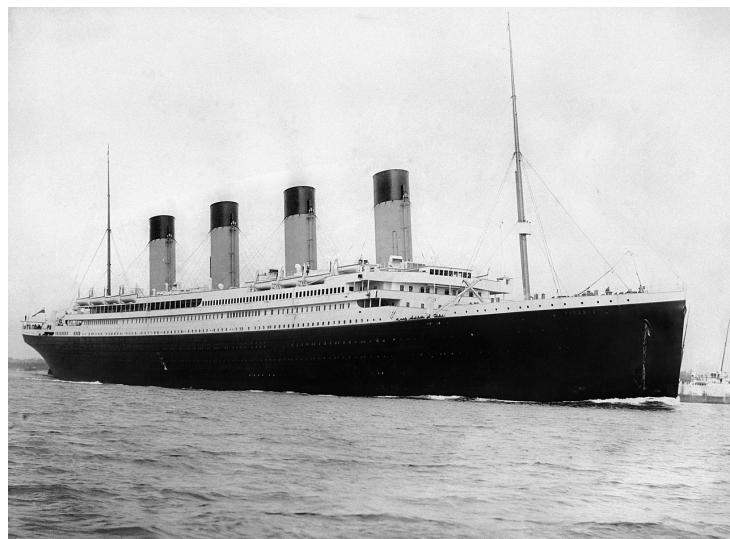


Figure 1: Fig. 1: RMS Titanic departing Southampton on April 10, 1912.

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.*

```
# Setting seed
set.seed(05022025)

# Necessary libraries
library(readr)
library(tidyverse)
library(tidymodels)
library(naniar)
```

```
library(ggplot2)
library(viridis)
library(knitr)
library(dplyr)
library(corrplot)
library(corr)
library(parsnip)
library(workflows)
library(discrim)
library(kknn)
library(yardstick)

# Loading the data in R
titanic <- read_csv("/Users/aarti/Downloads/homework-3/data/titanic.csv",
  col_types = cols(survived = readr::col_factor(levels = c("Yes",
  "No")), pclass = readr::col_factor(levels = c("1",
  "2", "3")), sex = readr::col_factor(levels = c("male",
  "female"))))
```

# Questions for 131

## 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?

**Solution:** To train the data, I would like to use the 70-30 split where 70% of the data is used to train models and 30% for testing. Before that, let's see how many observations the dataset has in total so we can make sure we split the data correctly.

```
nrow(titanic)
```

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

There are 891 observations in total.

```
titanic_split <- initial_split(titanic, prop = 0.70,
                                strata = survived)
```

```
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)
```

```
nrow(titanic_train)
```

```
## [1] 623
```

```
nrow(titanic_test)
```

```
## [1] 268
```

The observations in each, training and testing, are appropriate. Let's check whether there are any missing values in the training dataset.

```
vis_miss(titanic_train)
```

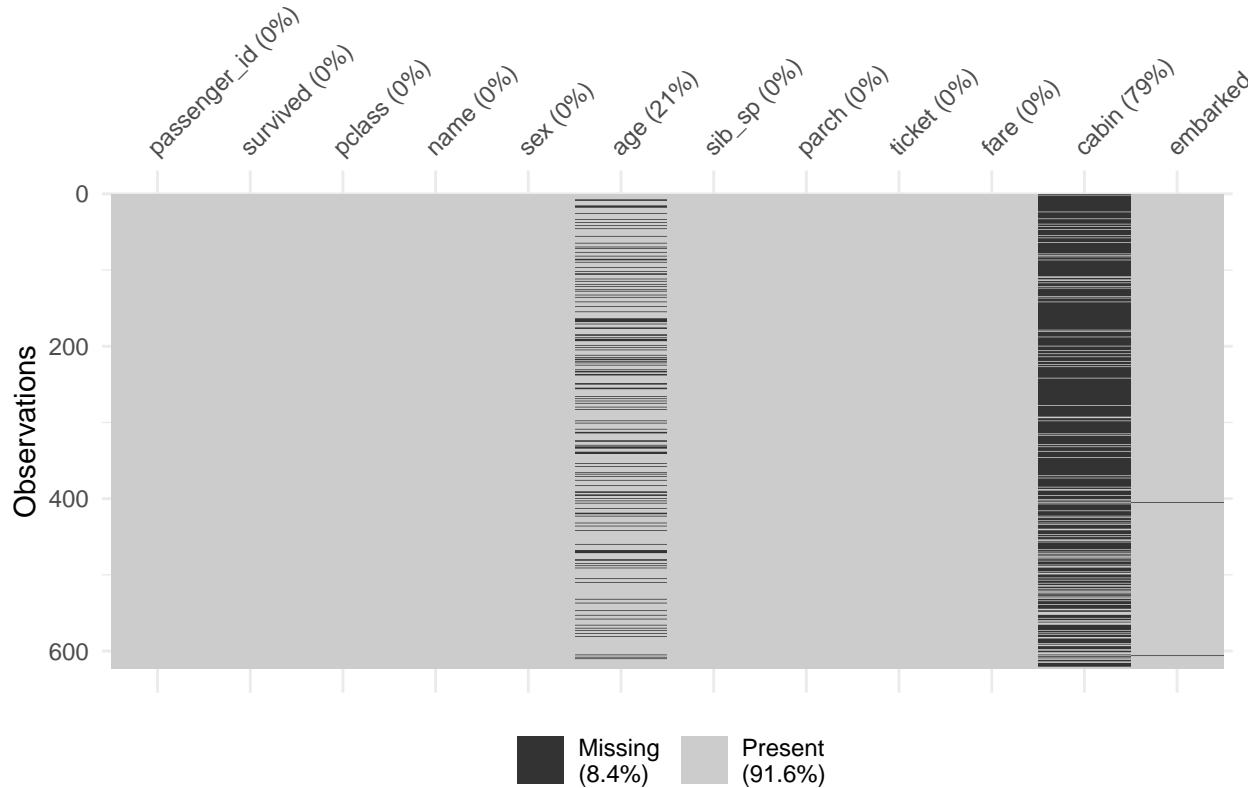


Figure 2: There seems to be around 20 percent of age variable missing from the data and about 76 percent in cabin. We can suspect that age would be more correlated with survival rates. Thus, we should focus on imputing age.

To guess the age of the passenger, most relevant variables would be number of siblings and spouses and number of parents and children.

```
titanic_rec <- recipe(survived ~ ., data = titanic_train) %>%
  step_impute_linear(age, impute_with = imp_vars(sib_sp, parch))

prep(titanic_rec) %>%
  bake(new_data = titanic_train) %>%
  vis_miss()
```

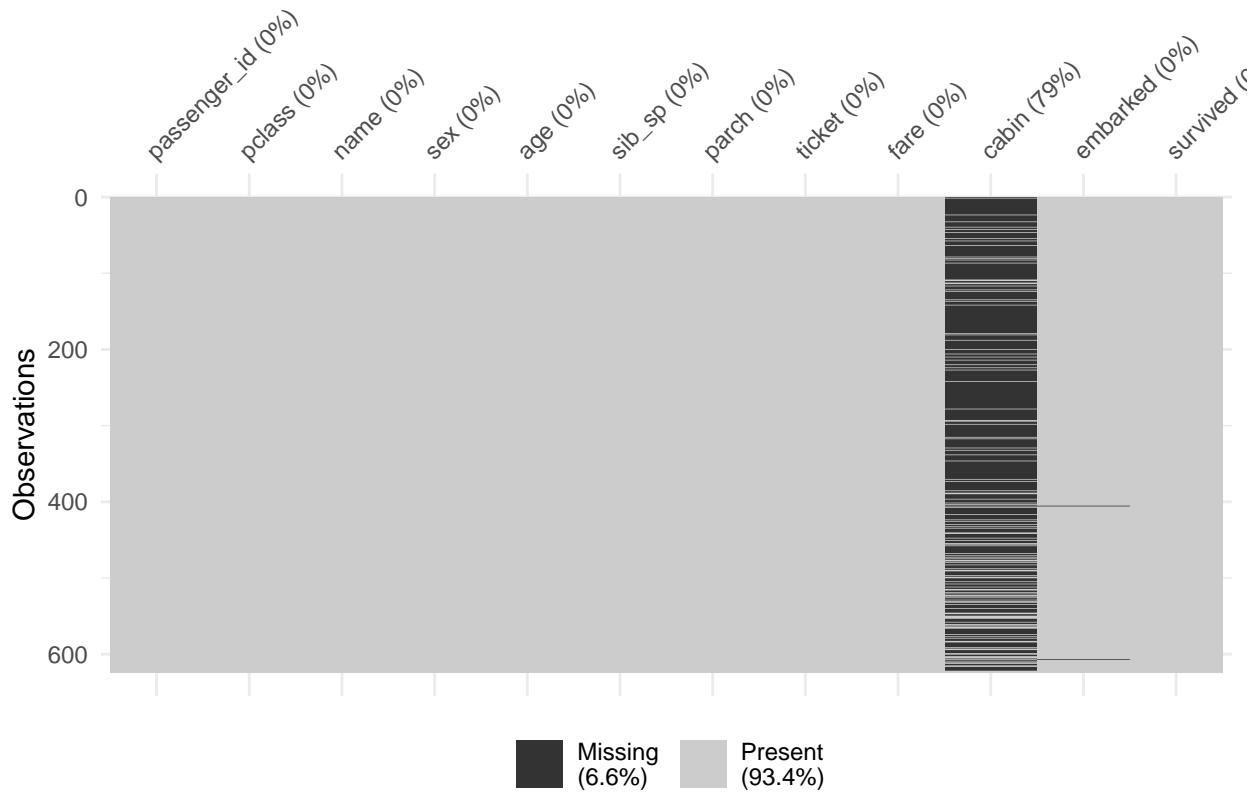


Figure 3: As we can see after imputing age, we don't have any missing values in age.

It's a good idea to use stratified sampling for this data because there might be an overlap between survivors and non-survivors so the training and testing dataset would split without a bias. If we split the data randomly, you might end up with a training or test set that doesn't reflect the imbalance in the survived. In the dataset, about 62% did not survive and 38% did. This imbalance can be represented when we use stratified sampling.

## Question 2

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

Create a percent stacked bar chart (recommend using `ggplot`) with **survived** on the *x*-axis and `fill = sex`. Do you think `sex` will be a good predictor of the outcome?

Create one more percent stacked bar chart of **survived**, this time with `fill = pclass`. Do you think passenger class will be a good predictor of the outcome?

Why do you think it might be more useful to use a percent stacked bar chart as opposed to a traditional stacked bar chart?

**Solution:** To explore the distribution of the **survived** variable, we can look at the bar plot, boxplot, and the density plot.

```
ggplot(titanic_train, aes(x=survived)) +  
  geom_bar() +  
  theme_bw()
```

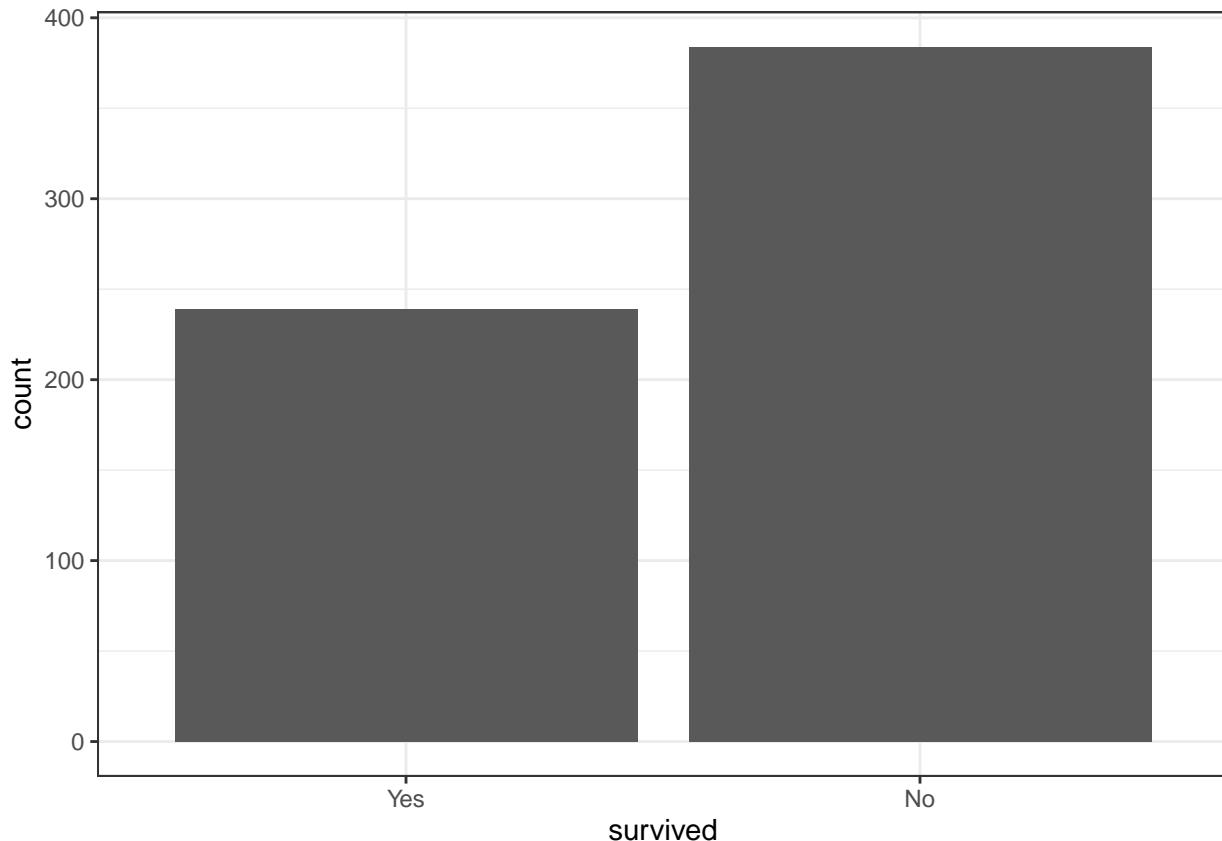


Figure 4: From the graph above it seems that the distribution of our outcome variable is skewed more towards 'no.' This means that there are more non survivors than survivors.

We can generate a table to see the skew more precisely.

```
kable(table(titanic_train$survived),  
      caption = "The table above shows the number of survivors and non survivors.")
```

Table 1: The table above shows the number of survivors and non survivors.

Var1	Freq
Yes	239
No	384

We can see the same information by looking at the percent stacked bar chart but we will explore the relationship between `sex` and `survived`.

```
ggplot(titanic_train, aes(fill = sex, x=survived)) +
  geom_bar(position = "fill") +
  scale_fill_viridis(discrete = T) +
  theme_bw() +
  ggtitle("Proportion of survival based on sex of the passenger") +
  ylab("Proportions")
```

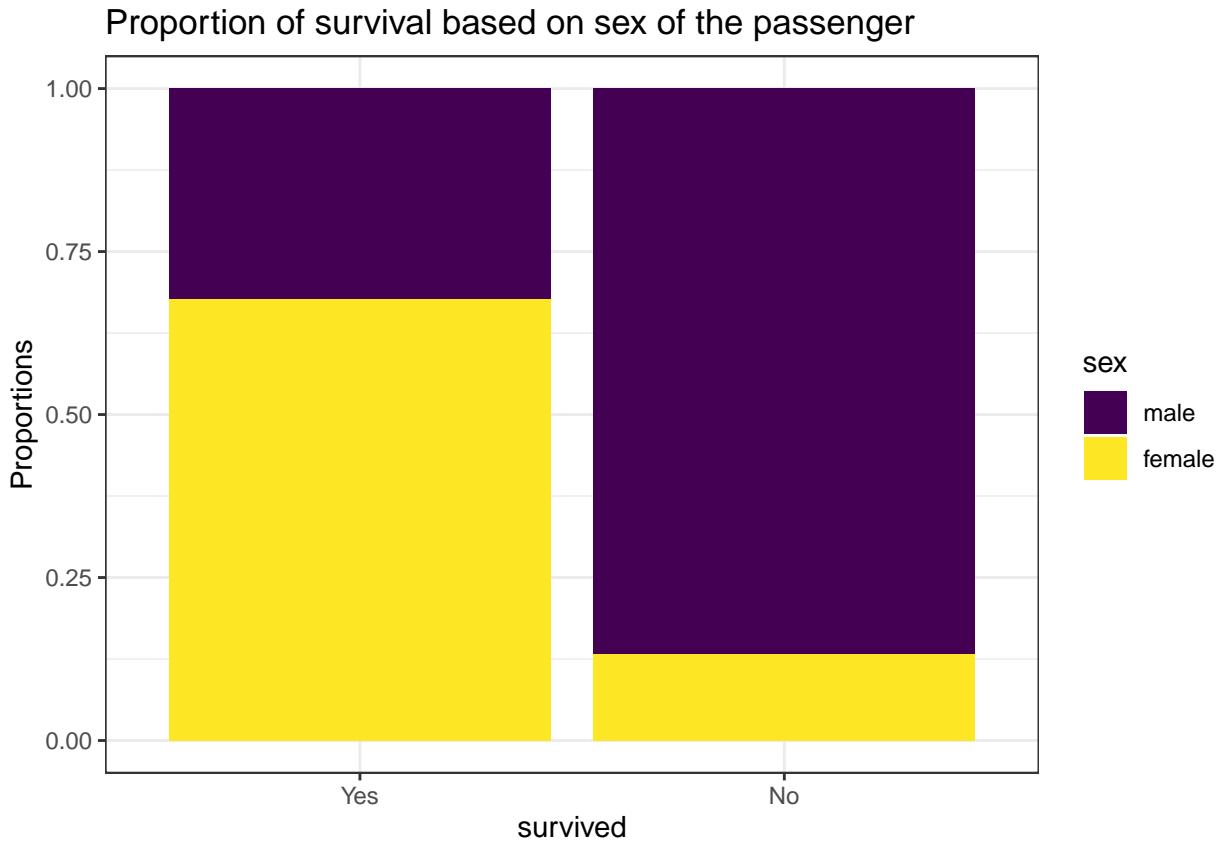


Figure 5: The graph above shows the survival of passengers based on their sex.

From the graph above it looks like more females survived the titanic than males. It's because priority was given to children and ladies when boarding the life boats during evacuation. This shows that `sex` is a pretty good predictor of survival.

Now we can look at a similar percent stacked bar chart and compare `pclass` against the `survive` variable to see whether it's a good predictor of not. In this case, we should expect to have three colors in the bars since there are three classes, 1, 2, and 3.

```

ggplot(titanic_train, aes(fill = pclass, x=survived)) +
  geom_bar(position = "fill") +
  scale_fill_viridis(discrete = T) +
  theme_bw() +
  ggtitle("Proportion of survival based on the ticket class of the passenger") +
  ylab("Proportions")

```

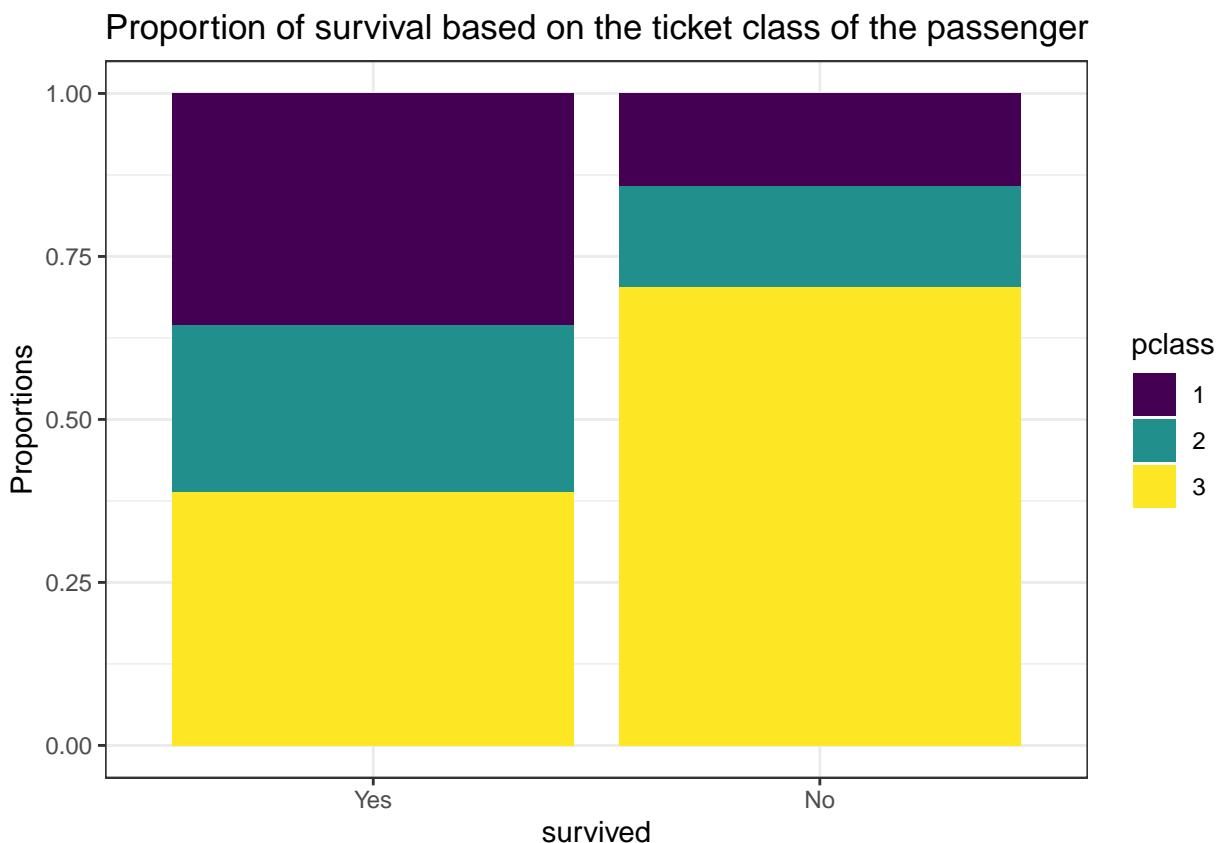


Figure 6: Looking at the graph we can see that not a lot of third class passengers survived the titanic. Whereas, a lot of first class passengers did.

Passenger ticket class is also a good predictor of survival. As we can see from the graph a lot of third class passengers did not survive the crash. Majority of first class passengers, on the other hand, seems to have survived the crash.

I think a stacked percent bar chart is more useful as it allows us to see the percentage of each subgroup that is represented in relation to the outcome variable. It allows to study the evolution of their proportion in the whole.

### Question 3

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

**Solution:** The correlation matrix of all the continuous predictors is useful to check whether the predictors are correlated with each other or not.

```
titanic_train %>%
  select(where(is.numeric)) %>%
  cor(use = "pairwise.complete.obs") %>%
  corrplot(method = "color", type = "lower", diag = FALSE)
```

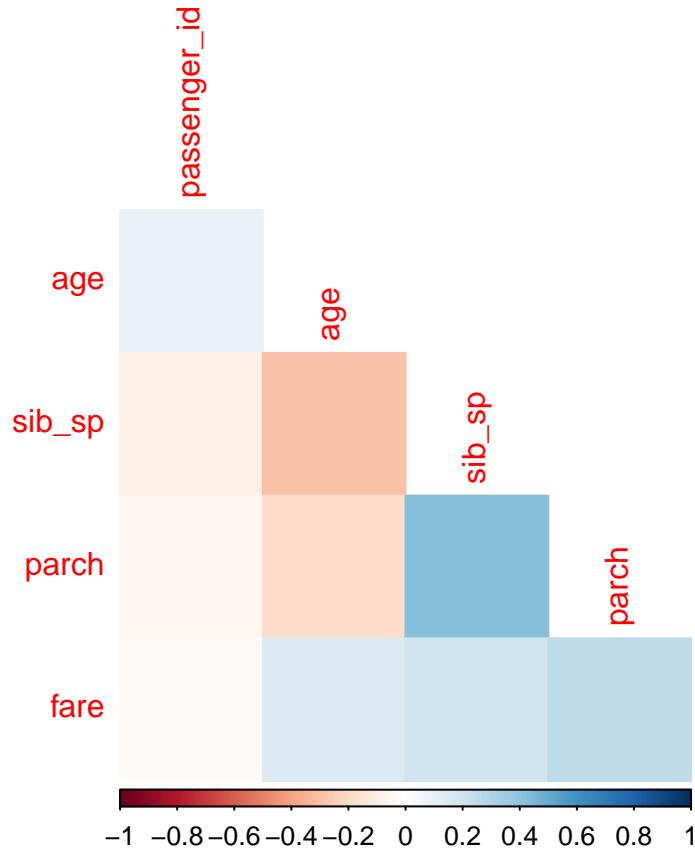


Figure 7: The correlation matrix shows that the predictors are not highly correlated with each other.

As we can see from the above graph, the predictors are not highly correlated with each other. There is some slight positive correlation between number of siblings and spouses and the number of parents and children on board. There is also some slight negative correlation between age and number of siblings and spouses.

Although these are nothing to be worried about since none of them are highly correlated.

#### 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.

**Solution:** The following code creates a recipe predicting the outcome variable **survived** by including the given predictors.

```
titanic_recipe1 <- recipe(survived ~ pclass + sex + age + sib_sp +
                           parch + fare, data = titanic_train) %>%
  step_impute_linear(age, impute_with = imp_vars(sib_sp, parch)) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(terms = ~ sex_female:fare) %>%
  step_interact(terms = ~ age:fare)

prep(titanic_recipe1) %>%
  bake(new_data=titanic_train) %>%
  head() %>%
  kable(caption = "The table shows just a few observations from the recipe.")
```

Table 2: The table shows just a few observations from the recipe.

age	sib_sp	parch	fare	survived	pclass_X2	pclass_X3	sex_female	sex_female_x_fare	age_x_fare
35	0	0	8.0500	No	0	1	0	0	281.750
54	0	0	51.8625	No	0	0	0	0	2800.575
2	3	1	21.0750	No	0	1	0	0	42.150
39	1	5	31.2750	No	0	1	0	0	1219.725
2	4	1	29.1250	No	0	1	0	0	58.250
35	0	0	26.0000	No	1	0	0	0	910.000

Using the training dataset we created the recipe where we dummy coded all the categorical predictors and imputed age so that there are no missing values and added the interaction terms.

## 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.*

**Solution:** Follow the code below

```
# specifying the engine
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

# setting up the workflow
titanic_train_logreg_workflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(titanic_recipe1)

titanic_train_logreg_fit <- fit(titanic_train_logreg_workflow, titanic_train)
```

This just sets up the engine and the workflow and then we saved the fitted model.

## Question 6

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

**Solution:** We will follow the same step but now for LDA.

```
# setting up the engine
lda_model <- discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")

# setting up the workflow
titanic_train_lda_workflow <- workflow() %>%
  add_model(lda_model) %>%
  add_recipe(titanic_recipe1)

titanic_train_lda_fit <- fit(titanic_train_lda_workflow, titanic_train)
```

This just sets up the engine and the workflow and then we saved the fitted LDA model.

## Question 7

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

**Solution:** We will follow the same step but now for QDA.

```
# setting up the engine
qda_model <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

# setting up the workflow
titanic_train_qda_workflow <- workflow() %>%
  add_model(qda_model) %>%
  add_recipe(titanic_recipe1)

titanic_train_qda_fit <- fit(titanic_train_qda_workflow, titanic_train)
```

This just sets up the engine and the workflow and then we saved the fitted QDA model.

## Question 8

**Repeat Question 5**, but this time specify a  $k$ -nearest neighbors model for classification using the "kknn" engine. Choose a value for  $k$  to try.

**Solution:** Follow the code below

```
# setting the engine
knn_model <- nearest_neighbor(neighbors = 10) %>%
  set_engine("kknn") %>%
  set_mode("classification")

# setting the workflow
titanic_train_knn_workflow <- workflow() %>%
  add_model(knn_model) %>%
  add_recipe(titanic_recipe1)

titanic_train_knn_fit <- fit(titanic_train_knn_workflow, titanic_train)
```

The hyperparameter that I chose is 11, mainly because we have two levels in our outcome variable so choosing an odd number is helpful.

### 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 metric of **area under the ROC curve** to assess the performance of each of the four models.

**Solution:** Using the `predict()` and `bind_cols()` we can generate the predictions using each of the four models on our training data set. However, it is important to note that we don't really care about the training data or these metrics, we want the testing metrics to be better.

```
# For logistic regression
titanic_train_logreg_pred <- predict(titanic_train_logreg_fit,
                                         new_data = titanic_train, type = "prob")

titanic_train_logreg_pred <- bind_cols(titanic_train_logreg_pred,
                                         titanic_train %>% select(survived))
kable(head(titanic_train_logreg_pred),
      caption = "The combined prediction for the training data set using
logistic regression.")
```

Table 3: The combined prediction for the training data set using logistic regression.

.pred_Yes	.pred_No	survived
0.0886586	0.9113414	No
0.2492802	0.7507198	No
0.1274543	0.8725457	No
0.0416021	0.9583979	No
0.0789349	0.9210651	No
0.2235293	0.7764707	No

```
# For LDA
titanic_train_lda_pred <- predict(titanic_train_lda_fit,
                                         new_data = titanic_train, type = "prob")

titanic_train_lda_pred <- bind_cols(titanic_train_lda_pred,
                                         titanic_train %>% select(survived))
kable(head(titanic_train_lda_pred),
      caption = "The combined prediction for the training data set using LDA.")
```

Table 4: The combined prediction for the training data set using LDA.

.pred_Yes	.pred_No	survived
0.0528515	0.9471485	No
0.1895933	0.8104067	No
0.0824466	0.9175534	No
0.0270280	0.9729720	No
0.0530751	0.9469249	No
0.1600958	0.8399042	No

```

# For QDA
titanic_train_qda_pred <- predict(titanic_train_qda_fit,
                                    new_data = titanic_train, type = "prob")

titanic_train_qda_pred <- bind_cols(titanic_train_qda_pred,
                                      titanic_train %>% select(survived))
kable(head(titanic_train_qda_pred),
      caption = "The combined prediction for the training data set using QDA.")

```

Table 5: The combined prediction for the training data set using QDA.

.pred_Yes	.pred_No	survived
0.0111724	0.9888276	No
0.0987997	0.9012003	No
0.0007216	0.9992784	No
0.0768918	0.9231082	No
0.0000084	0.9999916	No
0.0571048	0.9428952	No

```

# For KNN
titanic_train_knn_pred <- predict(titanic_train_knn_fit,
                                    new_data = titanic_train, type = "prob")

titanic_train_knn_pred <- bind_cols(titanic_train_knn_pred,
                                      titanic_train %>% select(survived))
kable(head(titanic_train_knn_pred),
      caption = "The combined prediction for the training data set using KNN.")

```

Table 6: The combined prediction for the training data set using KNN.

.pred_Yes	.pred_No	survived
0.0000000	1.0000000	No
0.0844219	0.9155781	No
0.0000000	1.0000000	No
0.1147009	0.8852991	No
0.0000000	1.0000000	No
0.1188331	0.8811669	No

Now we will assess the ROC curves for each of these models

```

titanic_train_logreg_pred %>%
  mutate(survived = factor(survived, levels = c("Yes", "No"))) %>%
  roc_curve(truth = survived, .pred_Yes) %>%
  autoplot()

```

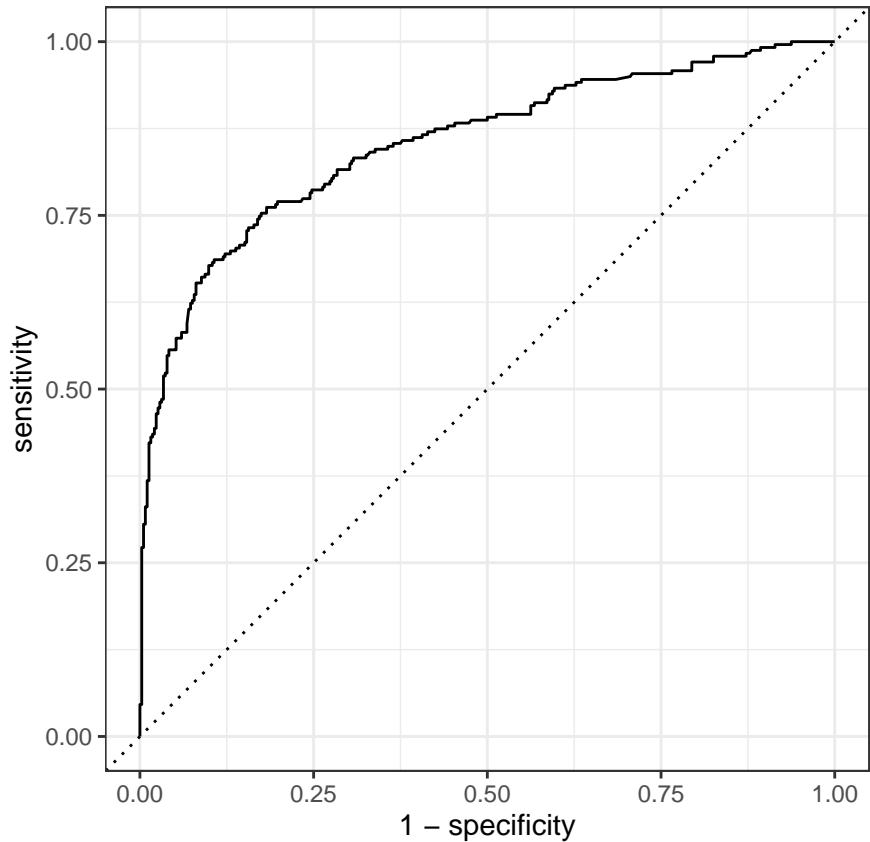


Figure 8: The graph shows the ROC curve for the logistic regression on the training data set. As we can see it's doing better than the random classifier. It looks like the highest point it's somewhere in 0.75 range. However, this is the training ROC curve, we should see how it does on the testing dataset.

```
roc_auc(titanic_train_logreg_pred, truth = survived, .pred_Yes)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>        <dbl>
## 1 roc_auc  binary     0.854
```

Now we will look at the ROC curve for the LDA model.

```
titanic_train_lda_pred %>%
  mutate(survived = factor(survived, levels = c("Yes", "No"))) %>%
  roc_curve(truth = survived, .pred_Yes) %>%
  autoplot()
```

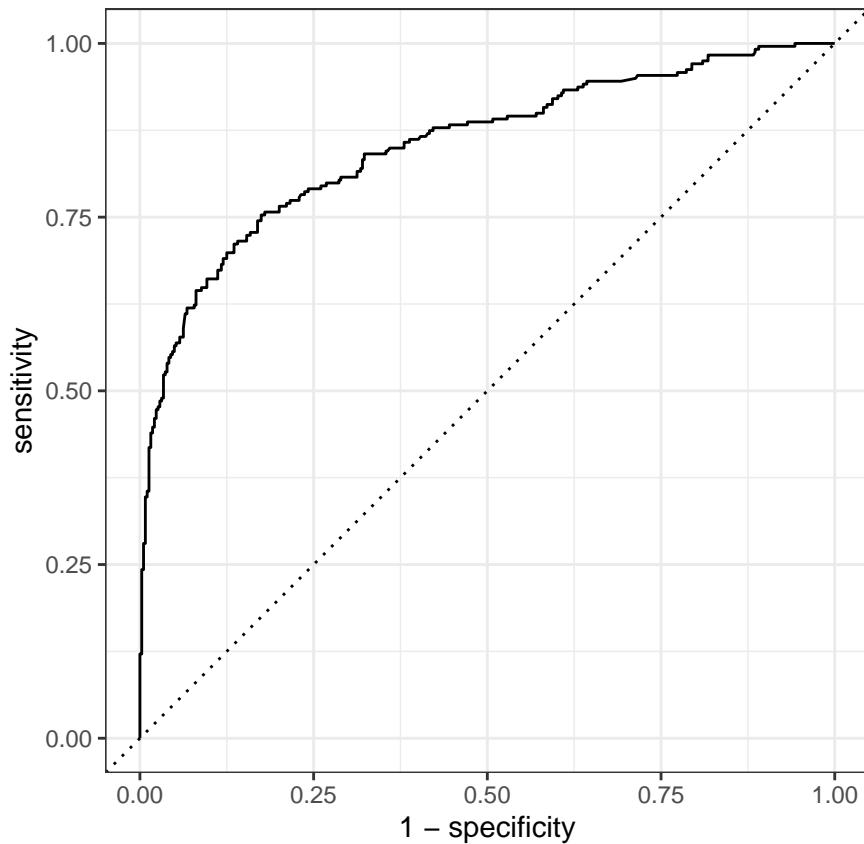


Figure 9: The graph shows the ROC curve for the LDA on the training data set. As we can see it's doing better than the random classifier. It looks like its highest point is somewhere in 0.75 range. However, this is the training ROC curve, we should see how it does on the testing dataset.

```
roc_auc(titanic_train_lda_pred, truth = survived, .pred_Yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>        <dbl>
## 1 roc_auc binary     0.853
```

For the QDA model

```
titanic_train_qda_pred %>%
  mutate(survived = factor(survived, levels = c("Yes", "No"))) %>%
  roc_curve(truth = survived, .pred_Yes) %>%
  autoplot()
```

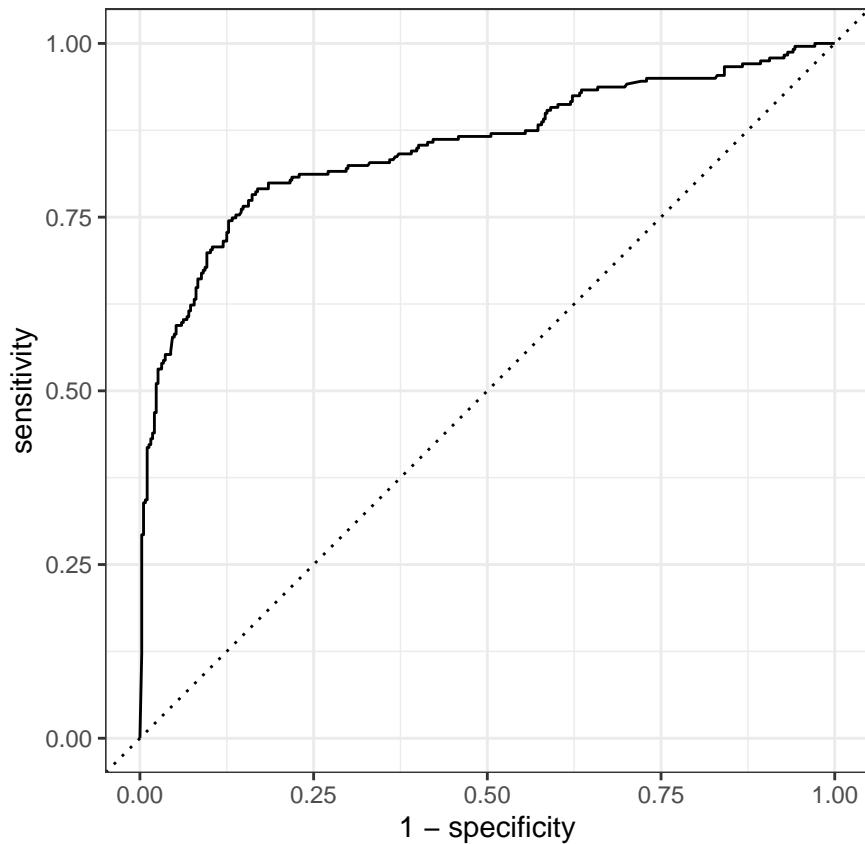


Figure 10: The graph shows the ROC curve for the QDA on the training data set. As we can see it's doing better than the random classifier. It looks like its highest point is somewhere in 0.75 range. However, this is the training ROC curve, we should see how it does on the testing dataset. It looks like the previous two were a little better than this one. We want to get to as close to a right angle as possible.

```
roc_auc(titanic_train_qda_pred, truth = survived, .pred_Yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>        <dbl>
## 1 roc_auc binary     0.852
```

For the KNN

```
titanic_train_knn_pred %>%
  mutate(survived = factor(survived, levels = c("Yes", "No"))) %>%
  roc_curve(truth = survived, .pred_Yes) %>%
  autoplot()
```

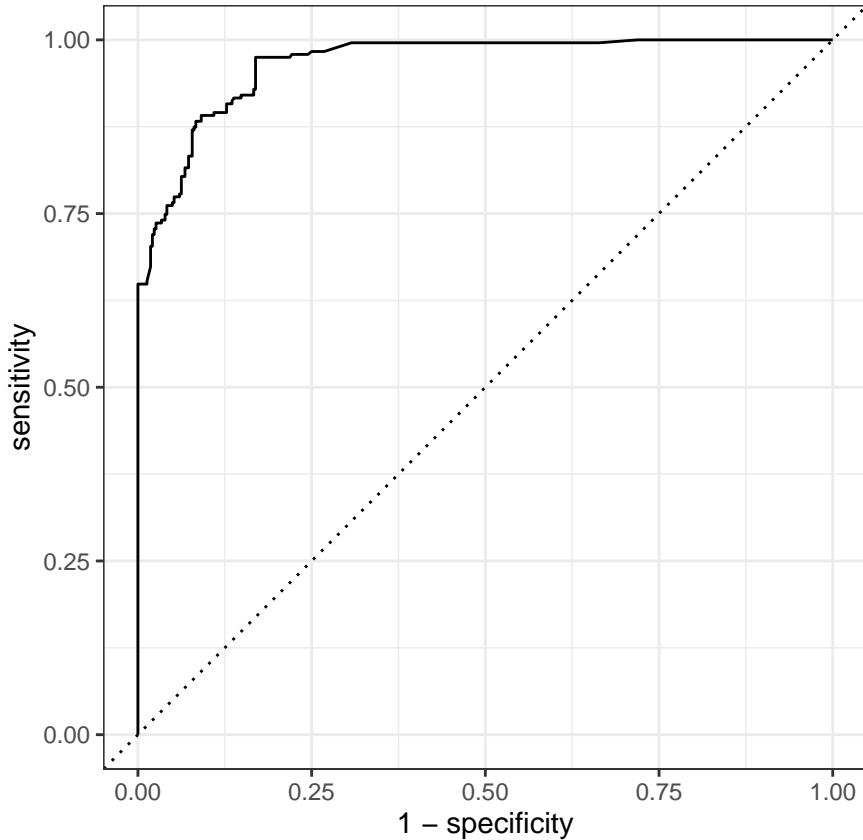


Figure 11: The graph shows the ROC curve for the KNN on the training data set. As we can see it's doing better than the random classifier. It looks like its highest point is somewhere in 0.75 range. However, this is the training ROC curve, we should see how it does on the testing dataset. It looks like this is doing a little better than others. We want to get as close to a right angle as possible.

```
roc_auc(titanic_train_knn_pred, truth = survived, .pred_Yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>        <dbl>
## 1 roc_auc binary     0.967
```

Just from looking at the ROC area under the curve and seeing the function of true positives vs the false positives on the four different models we built, we can tell the best one for the training dataset is the KNN model with  $K = 11$ . The worst one might be the QDA. The KNN model has the area under the curve of 0.967 and the closer we are to 1 the better model it is. The lowest roc area under the curve is the QDA one with 0.852. However, as mentioned several times above, it is important to note that these are the training ROC area under the curve, they don't mean anything. We need to look at the testing data to see how well our models do.

## Question 10

Fit all four models to your **testing** data and report the AUC of each model on the **testing** data. Which model achieved the highest AUC on the **testing** data?

Using your top-performing model, create a confusion matrix and visualize it. Create a plot of its ROC curve.

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

**Solution:** Now we shall fit the the model to the testing data set and see how the ROC area under the curve do for these. First we should fit the four models.

```
# For logistic regression
titanic_test_logreg_pred <- predict(titanic_train_logreg_fit,
                                       new_data = titanic_test, type = "prob")

titanic_test_logreg_pred <- bind_cols(titanic_test_logreg_pred,
                                       titanic_test %>% select(survived))

# For LDA
titanic_test_lda_pred <- predict(titanic_train_lda_fit,
                                   new_data = titanic_test, type = "prob")

titanic_test_lda_pred <- bind_cols(titanic_test_lda_pred,
                                   titanic_test %>% select(survived))

# For QDA
titanic_test_qda_pred <- predict(titanic_train_qda_fit,
                                   new_data = titanic_test, type = "prob")

titanic_test_qda_pred <- bind_cols(titanic_test_qda_pred,
                                   titanic_test %>% select(survived))

# For KNN
titanic_test_knn_pred <- predict(titanic_train_knn_fit,
                                   new_data = titanic_test, type = "prob")

titanic_test_knn_pred <- bind_cols(titanic_test_knn_pred,
                                   titanic_test %>% select(survived))
```

Now we must find the area under the curve. We can plot the roc curves too but the question only wants us to find the area under the curve so let's do that.

```
titanic_test_logreg_auc <- roc_auc(titanic_test_logreg_pred,
                                      truth = survived, .pred_Yes)
titanic_test_lda_auc <- roc_auc(titanic_test_lda_pred,
                                  truth = survived, .pred_Yes)
titanic_test_qda_auc <- roc_auc(titanic_test_qda_pred,
                                  truth = survived, .pred_Yes)
titanic_test_knn_auc <- roc_auc(titanic_test_knn_pred,
                                  truth = survived, .pred_Yes)

titanic_auc_df <- bind_rows(
  titanic_test_logreg_auc %>% mutate(model = "Logistic Regression"),
```

```

titanic_test_lda_auc %>% mutate(model = "LDA"),
titanic_test_qda_auc %>% mutate(model = "QDA"),
titanic_test_knn_auc %>% mutate(model = "KNN")
) %>%
  select(model, .estimate)

kable(titanic_auc_df, caption = "ROC area under the curve for the four
different models on the testing data set. It looks like KNN is slightly
better than the rest. Although other are pretty close to 1 too.")

```

Table 7: ROC area under the curve for the four different models on the testing data set. It looks like KNN is slightly better than the rest. Although other are pretty close to 1 too.

model	.estimate
Logistic Regression	0.8578700
LDA	0.8574581
QDA	0.8619594
KNN	0.8761695

The KNN model with  $K = 11$  appears to do slightly better than the rest of the models in this particular metric. Thus, we will use the KNN model to create our confusion matrix and the ROC curve.

```
titanic_test_knn_class <- predict(titanic_train_knn_fit,
                                    new_data = titanic_test, type = "class")
```

```
titanic_test_knn_results <- bind_cols(
  titanic_test_knn_class,
  titanic_test %>% select(survived)
)
```

```
conf_mat(data = titanic_test_knn_results, truth = survived,
          estimate = .pred_class)
```

```
##           Truth
## Prediction Yes No
##       Yes   76  20
##       No    27 145
```

It looks like it is doing a good job predicting. It predicted  $\frac{76}{(76+27)}$  that is 0.7378641 of the survivals of the titanic passengers correctly and  $\frac{145}{(20+145)}$  which is 0.8787879 correctly. Granted that the data is a little skewed with more non survivals than survivals, the model is doing pretty well classifying whether they survived the crash or not. We can further explore this with the heatmap of the confusion matrix.

```

conf_mat(data = titanic_test_knn_results, truth = survived,
         estimate = .pred_class) %>%
  autoplot(type = 'heatmap')

```

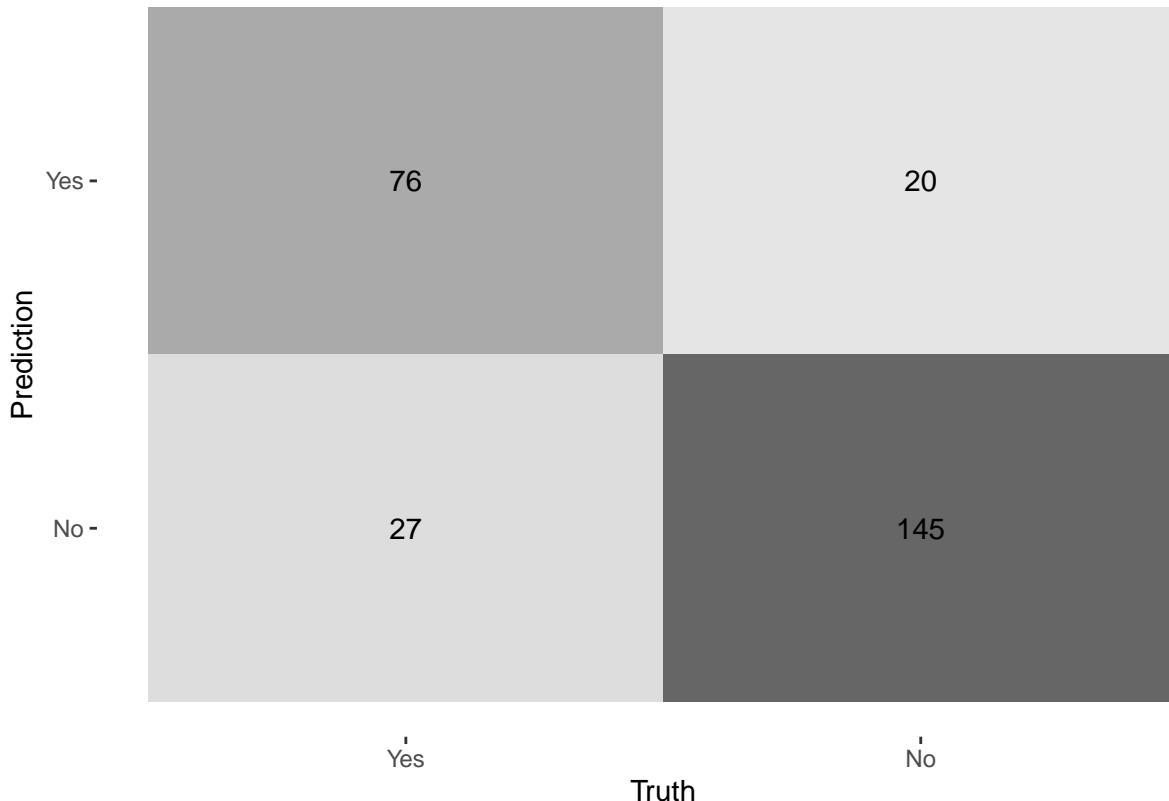


Figure 12: The heatmap of the confusion matrix shows exactly what we saw above but is easier to visualize how many of the predictions were true. It shows the sensitivity, the false negative and positive rates, and the accuracy of our model.

Now, we can look at the ROC curve. A good ROC curve is like a right angle such that the area under that curve is very close to 1. We should expect a somewhat right angle-like shape of the curve.

```
titanic_test_knn_pred %>%
  mutate(survived = factor(survived, levels = c("Yes", "No"))) %>%
  roc_curve(truth = survived, .pred_Yes) %>%
  autoplot()
```

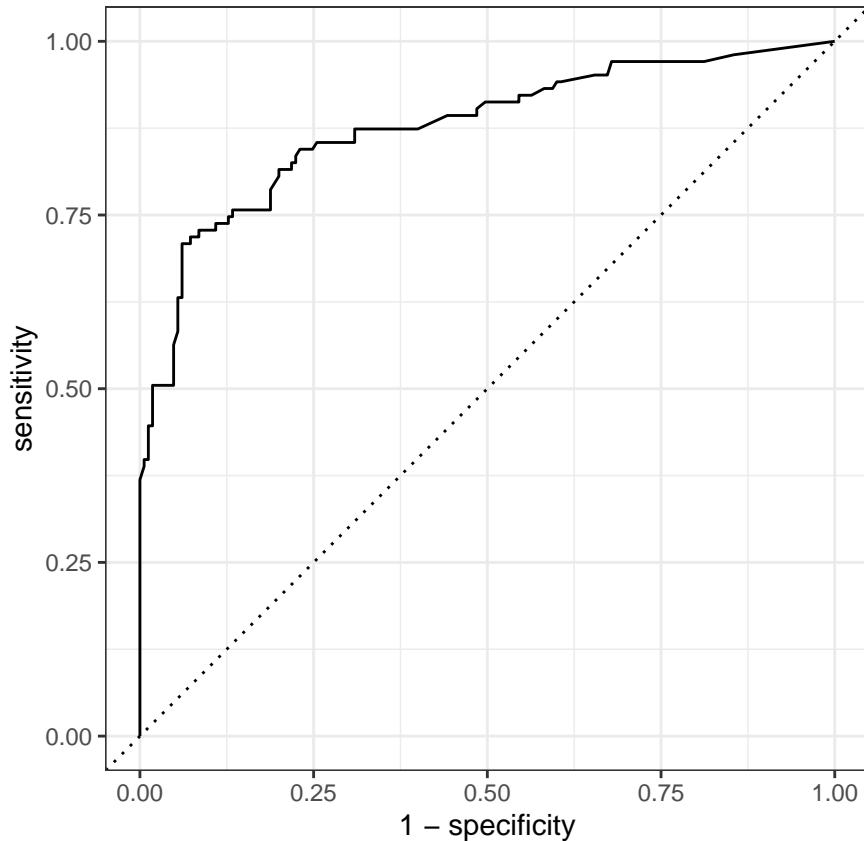


Figure 13: The graph shows the ROC curve for the KNN on the testing data set. As we can see it's doing better than the random classifier. It looks like it's highest point is somewhere in 0.75 range. From the dataframe above we know the area under this curve is 0.8761695.

It looks like it did worse than the training set. We knew that from the area under the curve but the graph just reiterates that. The ROC curve when the model was fit on the training dataset is more of a right angle than this one.

One of the possible reason might be that the model might be overfitting. We know KNN is a non-parametric method that relies heavily on the training set. It's important that there are a lot of similarities in the training and the testing dataset for this model to do well. It performs very well on training data (especially for small k) because it's just checking nearby points — often memorizing patterns. On unseen testing data, those exact neighborhoods may not generalize well, leading to reduced performance. Furthermore, there is the issue of high dimensionality. The titanic dataset had a lot of predictors, KNN does well in lower dimensionality. KNN is therefore, unable to recognize the global boundaries. Furthermore, there was a class imbalance as noted and that is why we did the stratified sampling. This might also cause some issues. KNN is prone to overfitting in higher dimensional problems. Considering all these problems, I would still say the model performed better than expected.

## 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:

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

**Solution:**

*Proof.* Given that the logistic function on terms of  $p(z)$  where  $z \in \mathbb{Z}$ ,

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

to get the inverse we would like to solve for  $p(z)$

Multiplying  $(1 + e^z)$  on both sides,

$$\begin{aligned} p(1 + e^z) &= e^z \\ p + pe^z &= e^z \\ p &= e^z - pe^z \\ p &= e^z(1 - p) \end{aligned}$$

Note, as specified in the question the interval for the inverse function is  $(0, 1)$ . Therefore, we can divide both sides by  $(1 - p)$ . Gives,

$$e^z = \frac{p}{1 - p}$$

Taking the natural log on both sides to solve for  $p$  gives,

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

Thus, the inverse of the logistic function is the indeed the logit function. □

### Question 12

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

Assume now that  $\beta_1$  is negative. What value does  $p$  approach as  $x_1$  approaches  $\infty$ ? What value does  $p$  approach as  $x_1$  approaches  $-\infty$ ? Demonstrate.

**Solution:**

*Proof.* We are to assume that  $z = \beta_0 + \beta_1 x_1$  and  $p = \text{logistic}(z)$ . Thus,

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

From the question above we know the inverse of the above function is

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

Then,

$$e^z = \frac{p}{1 - p}$$

If  $z = \beta_0 + \beta_1 x_1$ . Then,

$$e^{\beta_0 + \beta_1 x_1} = \frac{p}{1 - p}$$

If  $x_1$  increases by 2 then,

$$e^{\beta_0 + \beta_1(x_1 + 2)}$$

which gives,

$$\begin{aligned} e^{\beta_0 + \beta_1(x_1 + 2)} &= e^{\beta_0 + \beta_1 x_1 + 2\beta_1} \\ &= e^{\beta_0 + \beta_1 x_1} \cdot e^{2\beta_1} \\ &= e^z \cdot e^{2\beta_1} \\ &= \left(\frac{p}{1 - p}\right) e^{2\beta_1} \end{aligned}$$

Thus, the odds increase by a factor of  $e^{2\beta_1}$ . □

Now we shall move on to the second part of the question which is asking us to assume that  $\beta_1 < 0$

*Proof.* From the question above and the part we just did, we know

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

Assuming  $\beta_1 < 0$  gives

$$p = \frac{e^{\beta_0 + (-\beta_1 x_1)}}{1 + e^{\beta_0 + (-\beta_1 x_1)}}$$

We want to know what happens to  $p$  as  $x_1 \rightarrow \infty$ . Then, we must take the limit.

$$\begin{aligned} \implies \lim_{x_1 \rightarrow \infty} \left( \frac{e^{\beta_0 + (-\beta_1 x_1)}}{1 + e^{\beta_0 + (-\beta_1 x_1)}} \right) &= \lim_{x_1 \rightarrow \infty} (p) \\ \lim_{x_1 \rightarrow \infty} \left( \frac{e^{\beta_0 - \beta_1 x_1}}{1 + e^{\beta_0 - \beta_1 x_1}} \right) &= \frac{0}{1 + 0} = 0 \end{aligned}$$

Thus, if  $\beta_1 < 0$  and  $x_1 \rightarrow \infty$  then  $p \rightarrow 0$ .

If  $x_1 \rightarrow -\infty$ , we must also take the limit as we did in the above part to know where  $p$  tends to go.

$$\implies \lim_{x_1 \rightarrow -\infty} \left( \frac{e^{\beta_0 + (-\beta_1 x_1)}}{1 + e^{\beta_0 + (-\beta_1 x_1)}} \right) = \lim_{x \rightarrow -\infty} (p)$$

Note, as  $x_1 \rightarrow -\infty$  and  $\beta_1 < 0$  then,  $\beta_1 \cdot x_1 > 0$ . Informally the right hand side limit looks something like this

$$\frac{e^\infty}{1 + e^\infty}$$

Note, that the rate at which both the numerator and the denominator approach  $\infty$  is the same. The  $+1$  is just the  $y$ -intercept in the denominator. They both behave the same way and have the same derivatives. Thus, the ratio tends toward 1. So,

$$\implies \lim_{x_1 \rightarrow -\infty} \left( \frac{e^{\beta_0 + (-\beta_1 x_1)}}{1 + e^{\beta_0 + (-\beta_1 x_1)}} \right) = 1$$

Another way of seeing this clearly is,

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

If we multiply both the numerator and denominator by  $e^{-z}$ , which is essentially just multiplying by 1, we get

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

We know  $z = \beta_0 + \beta_1 x_1$  where  $\beta_1 < 0$ . Substituting gives

$$\frac{1}{e^{-(\beta_0 - \beta_1 x_1)} + 1} = \frac{1}{e^{-\beta_0 + \beta_1 x_1} + 1}$$

Now, taking the limit gives

$$\lim_{x_1 \rightarrow \infty} \left( \frac{1}{e^{-\beta_0 + \beta_1 x_1} + 1} \right) = \frac{1}{0 + 1} = 1$$

Therefore, when  $\beta_1 < 0$  and  $x_1 \rightarrow \infty$ ,  $p \rightarrow 1$ .

□