# Homework 3

PSTAT 131

## Contents



## 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.

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

```
library(tidyverse)
library(tidymodels)
library(discrim) # for lda and qda modeling
library(corrplot) # for creating correlation plots

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

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.

```
titanic_data = titanic_dataset %>%
    mutate(survived = factor(survived, levels = c('Yes', 'No'))) %>%
    mutate(pclass = factor(pclass))
```

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.*

### 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?

```
set.seed(1210)
titanic_split = initial_split(titanic_data, strata = survived, prop = 0.80)
titanic_train = training(titanic_split)
titanic_test = testing(titanic_split)
```

We do notice some missing data for our age variable as well as almost all of our cabin variable data as missing. We should use stratified sampling for this data because we need to make sure that we have a sufficient number of observations from each category, in this case 'survival' (the response). In our original dataset, we have many more reported 'No' observations than 'Yes'. If we don't stratify, then it is possible that we may not get very many 'Yes' observations to work with in our training set (the one we use to predict), or even none at all depending on how uncommon they are. By stratifying, we get the right proportion of 'Yes' and 'No' values in our training data and testing data like our original dataset.

### Question 2

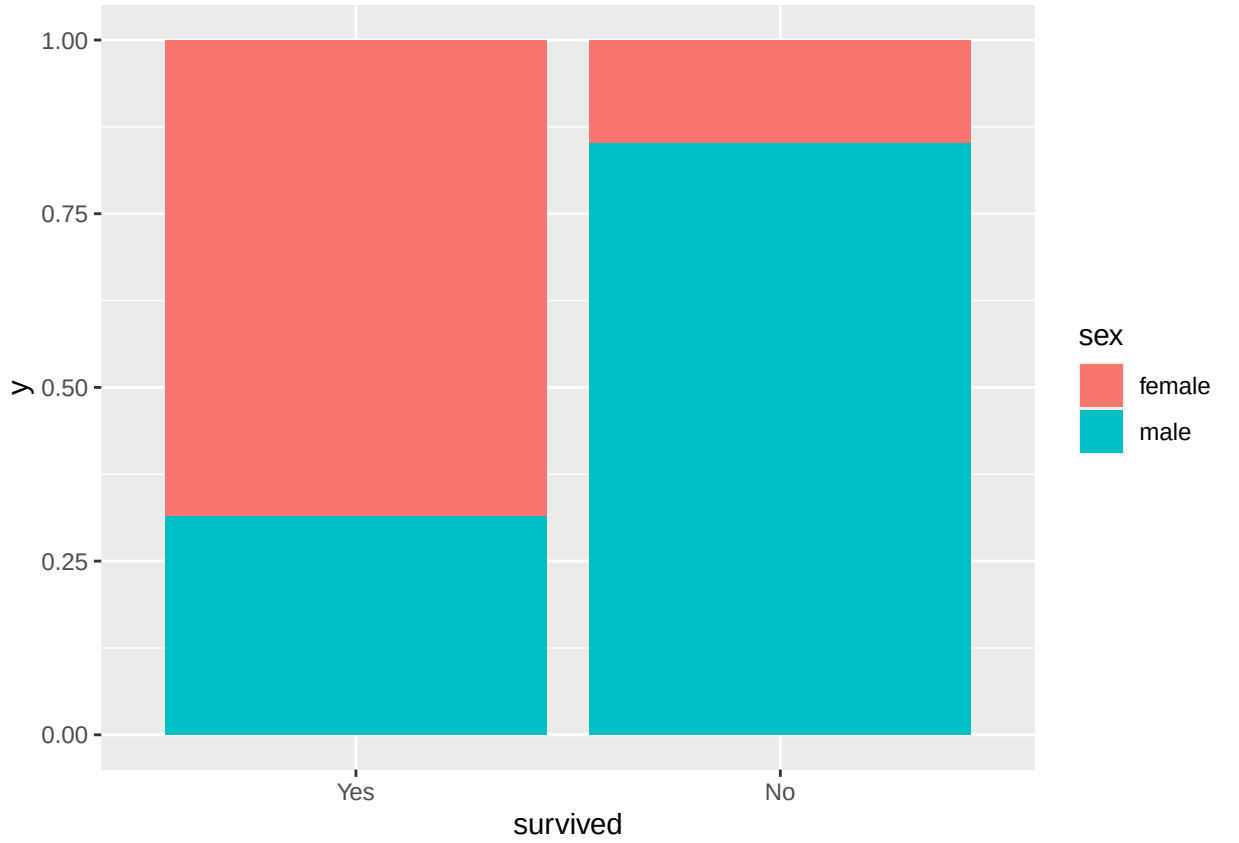
Using the **training** data set, explore/describe the distribution of the outcome variable `survived`.

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?

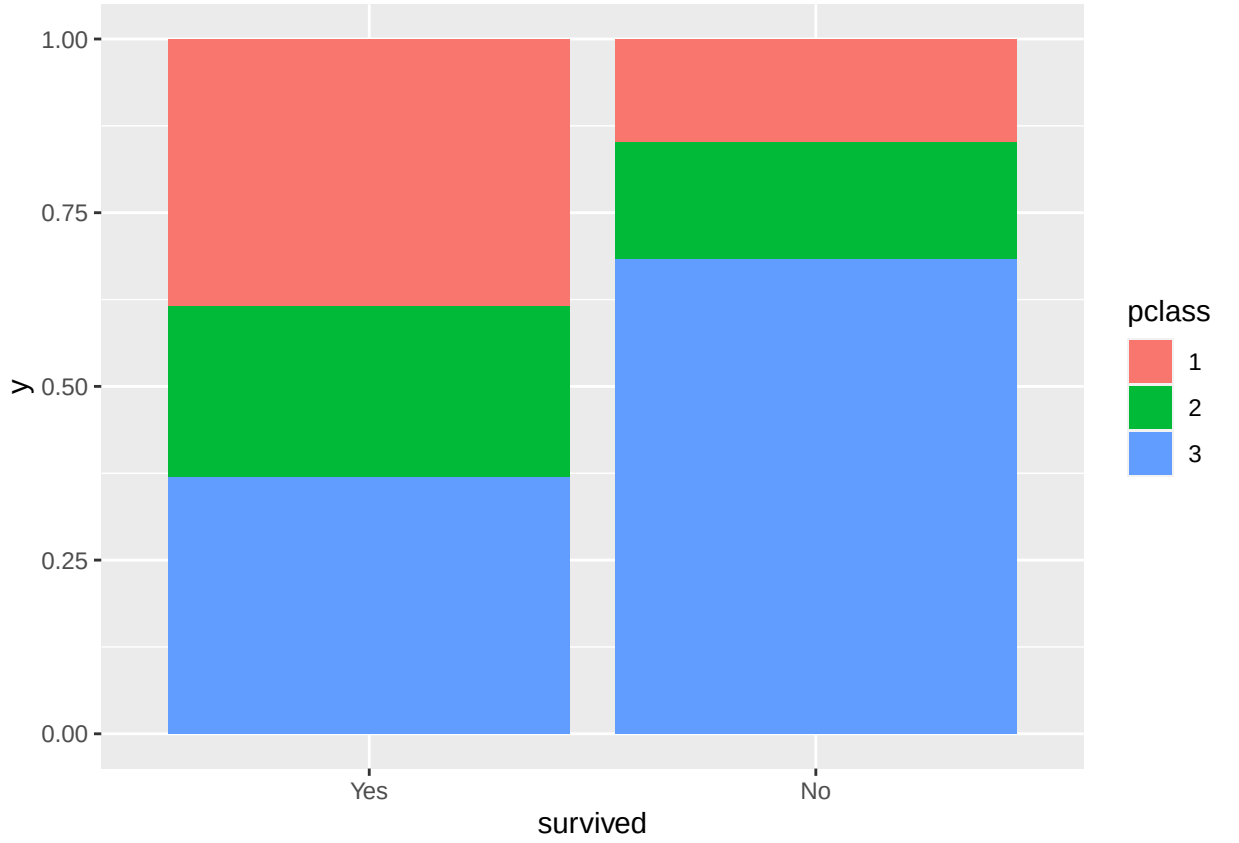
Why do you think it might be more useful to use a percent stacked bar chart as opposed to a traditional stacked bar chart?

```
ggplot(titanic_train, aes(fill=sex, y=1, x=survived)) +
geom_bar(position='fill', stat= 'identity')
```




I think sex will be a good predictor of whether a passenger survived or not. About two thirds of the passengers who survived were females. In addition, about 85% of the passengers who didn't survive were male.

```
ggplot(titanic_train, aes(fill=pclass, y=1, x=survived)) +
geom_bar(position='fill', stat ='identity')
```



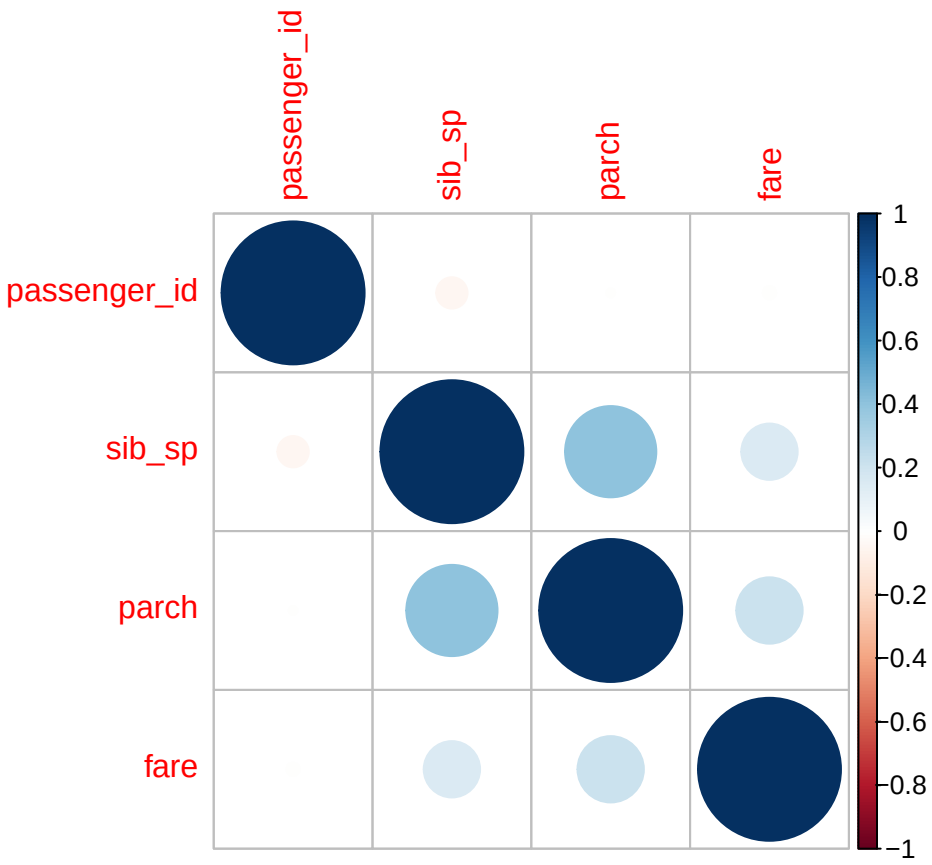

More than one third (~38%) of those survived were in 1st class. This is a lot, especially given that a majority of those on the titanic were in 2nd, or 3rd class. From those who didn't survive, only about 15 percent of them were in first class, and about 69% of them were in third class. So I would say that passenger class would be a good predictor of survival.

Using a percent stacked bar chart can be more beneficial to us, especially in this case because we want to observe what percentage more survived/didn't survive based on a specific category (pclass, sex, etc.), and it will be hard to identify these differences with just a stacked chart as we would just be using numbers to compare differences and for interpretation. In the end, what we really want is a percentage of how many more survived/didn't survive based on specific grouping that we have done above which can't be done easily with just a stacked chart.

### 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?

```
titanic_train %>%
  select(passenger_id, sib_sp, parch, fare) %>%
  cor() %>%
  corrplot()
```




Age had a question mark reported for its correlations due to an insufficient amount of data. There was no negative correlations between the variables, and there was slight positive correlations between some variables but that's about it, nothing very noteworthy. Parch and sib_sp had about a 0.4 positive correlation, and fare had an even lower positive correlation with sib_sp and parch (between 0.1 and 0.2). Passenger ID had basically no correlation with any of the continuous variables, which was what I expected.

### 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.

```
titanic_recipe = recipe(survived ~ pclass + sex + age + sib_sp + parch + fare,
                        data = titanic_train) %>%
  step_impute_linear(age, impute_with = imp_vars(all_predictors())) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(~ starts_with('sex'):fare + 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_model = logistic_reg() %>%
  set_engine('glm') %>%
  set_mode('classification')

titanic_logwflow = workflow() %>%
  add_model(log_reg_model) %>%
  add_recipe(titanic_recipe)

titanic_logfit = fit(titanic_logwflow, titanic_train)

titanic_logfit %>%  # makes our log_fit readable (we can see information about all our predictors)
  tidy()
```

```
## # A tibble: 10 x 5
##    term             estimate std.error statistic  p.value
##    <chr>               <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)     -5.17      0.645       -8.02  1.10e-15
##  2 age              0.0795    0.0129       6.15  7.85e-10
##  3 sib_sp           0.384     0.130        2.96  3.04e- 3
##  4 parch            0.0732    0.130        0.564 5.73e- 1
##  5 fare             0.0115    0.00878      1.31  1.91e- 1
##  6 pclass_X2        1.33      0.360        3.68  2.36e- 4
##  7 pclass_X3        2.69      0.371        7.25  4.21e-13
##  8 sex_male         2.38      0.278        8.58  9.46e-18
##  9 sex_male_x_fare  0.0125    0.00660      1.90  5.79e- 2
## 10 fare_x_age      -0.000624  0.000219    -2.84  4.44e- 3
```

### Question 6

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

```
lda_model = discrim_linear() %>%
  set_mode('classification') %>%
  set_engine('MASS')

titaniclda_wflow = workflow() %>%
  add_model(lda_model) %>%
  add_recipe(titanic_recipe)

titanic_ldafit = fit(titaniclda_wflow, titanic_train)
```

### Question 7

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

```
qda_model = discrim_quad() %>%
  set_mode('classification') %>%
  set_engine('MASS')

titanicqda_wflow = workflow() %>%
  add_model(qda_model) %>%
  add_recipe(titanic_recipe)

titanic_qdafit = fit(titanicqda_wflow, titanic_train)
```

### 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.

```
knn_model = nearest_neighbor(neighbors = 7) %>%
  set_engine('kknn') %>%
  set_mode('classification')

titanic_knn_wflow = workflow() %>%
  add_model(knn_model) %>%
  add_recipe(titanic_recipe)

titanic_knn_fit = fit(titanic_knn_wflow, 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 metric of **area under the ROC curve** to assess the performance of each of the four models.

```
# will use all variables besides survived based on the knn_fit, which uses the recipe formula, not all
titanic_train_logreg = predict(titanic_logfit, new_data = titanic_train %>% select(-survived))
titanic_train_lda = predict(titanic_ldafit, new_data = titanic_train %>% select(-survived))
titanic_train_qda = predict(titanic_qdafit, new_data = titanic_train %>% select(-survived))
titanic_train_knn = predict(titanic_knn_fit, new_data = titanic_train %>% select(-survived))

titanic_train_pred = bind_cols(titanic_train_logreg, titanic_train_lda, titanic_train_qda,
                               titanic_train_knn, titanic_train %>% select(survived))
titanic_train_pred
```

```
## # A tibble: 712 x 5
##    .pred_class...1 .pred_class...2 .pred_class...3 .pred_class...4 survived
##    <fct>           <fct>           <fct>           <fct>           <fct>
##  1 No              No              No              No              No
##  2 No              No              No              No              No
##  3 No              No              No              No              No
##  4 No              No              No              No              No
##  5 No              No              No              No              No
##  6 No              No              No              No              No
```

```
##  7 No              No              No              No              No
##  8 No              Yes             No              No              No
##  9 No              No              No              No              No
## 10 No              No              No              No              No
## # i 702 more rows
```

```
titanic_trainlogauc = augment(titanic_logfit, new_data = titanic_train) %>%
  roc_auc(survived, .pred_Yes)
titanic_trainlogauc
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.861
```

```
titanic_trainldaauc = augment(titanic_ldafit, new_data = titanic_train) %>%
  roc_auc(survived, .pred_Yes)
titanic_trainldaauc
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.859
```

```
titanic_trainqdaauc = augment(titanic_qdafit, new_data = titanic_train) %>%
  roc_auc(survived, .pred_Yes)
titanic_trainqdaauc
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.862
```

```
titanic_trainknnauc = augment(titanic_knn_fit, new_data = titanic_train) %>%
  roc_auc(survived, .pred_Yes)
titanic_trainknnauc
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.978
```

### 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?

```
titanic_testlogauc = augment(titanic_logfit, new_data = titanic_test) %>%
  roc_auc(survived, .pred_Yes)
titanic_testlogauc
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.826
```

```
titanic_testldaauc = augment(titanic_ldafit, new_data = titanic_test) %>%
  roc_auc(survived, .pred_Yes)
titanic_testldaauc
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.826
```

```
titanic_testqdaauc = augment(titanic_qdafit, new_data = titanic_test) %>%
  roc_auc(survived, .pred_Yes)
titanic_testqdaauc
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.864
```

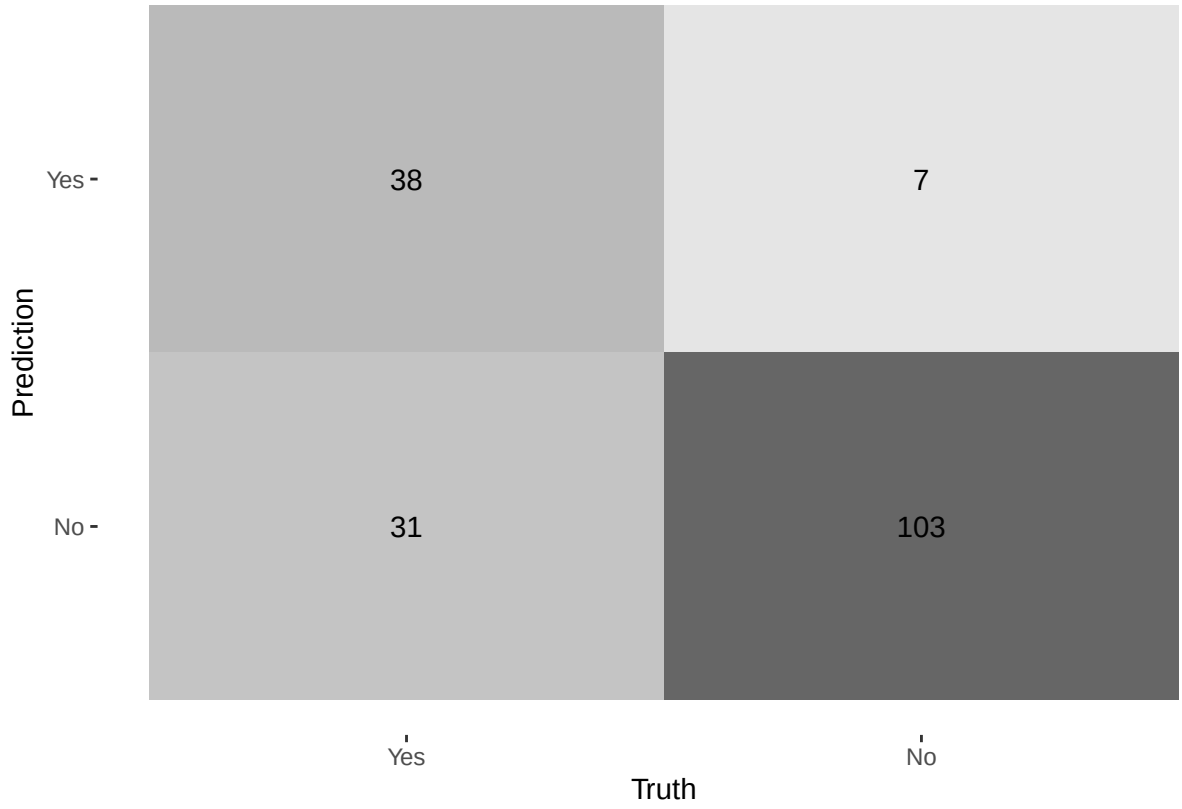
```
titanic_testknnauc = augment(titanic_knn_fit, new_data = titanic_test) %>%
  roc_auc(survived, .pred_Yes)
titanic_testknnauc
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.847
```

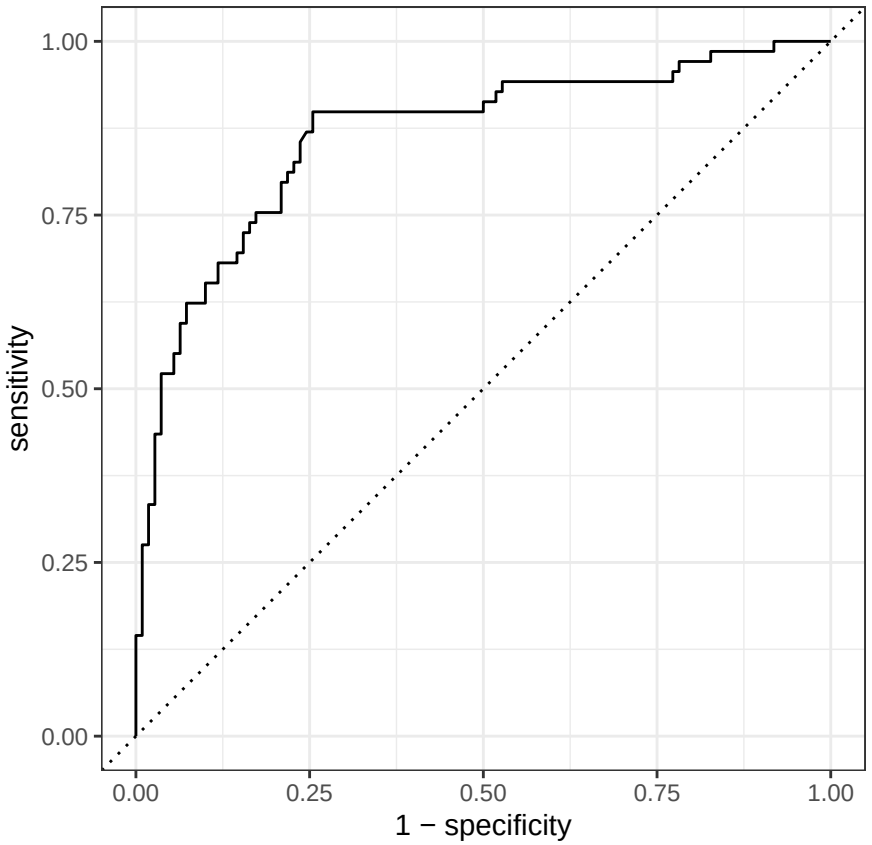
For our testing data, the logistic regression model had an AUC of 0.826, the linear discriminant analysis model also had an AUC of 0.826, the quadratic discriminant analysis model had an AUC of 0.864, and the KNN model had an AUC of 0.847. Our quadratic discriminant analysis model had the highest AUC on the testing data (0.847).

```
# QDA Model is fit to test data (best model)

titanic_testqdamatrix = augment(titanic_qdafit, new_data = titanic_test) %>%
 conf_mat(survived, .pred_class) %>%
  autoplot(type = 'heatmap')
 titanic_testqdamatrix
```




```
titanic_testqdaplot = augment(titanic_qdafit, new_data = titanic_test) %>%
  roc_curve(survived, .pred_Yes) %>%
  autoplot()
titanic_testqdaplot
```




Looking at our confusion matrix, the numbers on the diagonals were the highest (38 and 103; our true positive and negative predictions respectively) which is what we want. However, only 55% of our "Yes" predictions were correct where 94% of our "No" predictions were correct.

Our best model, which was our QDA model performed very well actually. It had a training AUC value of 0.862, and a testing AUC value of 0.864. The testing AUC was greater than our training AUC which is very surprising, as our model uses the training data to learn and make predictions where our training set is data that our model has never seen. So this is quite unusual and unexpected. Some possible reasons: It could be that our model didn't have enough data to work with in our training set to fully learn from, or our testing set was too small so even though we stratified on our response, we were not working with the best testing set.