COMPSCIX 415.2 Homework 8

Sanatan Das

March 24, 2018

Contents

Code and Documents Git Repository

All the work can be found in the below Git repository location:

https://github.com/sanatanonline/compscix-415-2-assignments

Load packages (prerequisites to run the code in this document)

```
library(tidyverse)
library(broom)
library(rpart)
library(partykit)
library(ROCR)
```

Analysis of Titanic dataset

Load data and glimpse it

Exercise 1

Load the train.csv dataset into R. How many observations and columns are there? Convert the target variable to a factor because it will be loaded into R as an integer by default.

Answer:

```
# load train.csv
train = read_csv("/opt/apps/code/git/compscix-415-2-assignments/titanic/train.csv")
```

```
## Parsed with column specification:
## cols(
##   PassengerId = col_integer(),
##   Survived = col_integer(),
##   Pclass = col_integer(),
##   Name = col_character(),
##   Sex = col_character(),
##   Age = col_double(),
##   SibSp = col_integer(),
##   Parch = col_integer(),
##   Ticket = col_character(),
##   Fare = col_double(),
##   Cabin = col_character(),
##   Embarked = col_character()
## )
```

```
# glimpse train
glimpse(train)
```

```
## Observations: 891
## Variables: 12
## $ PassengerId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,...
## $ Survived    <int> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0,...
## $ Pclass      <int> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3,...
## $ Name        <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bra...
## $ Sex         <chr> "male", "female", "female", "female", "male", "mal...
## $ Age         <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, ...
## $ SibSp       <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4,...
## $ Parch       <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1,...
## $ Ticket      <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "1138...
## $ Fare        <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, ...
## $ Cabin       <chr> NA, "C85", NA, "C123", NA, NA, "E46", NA, NA, NA, ...
## $ Embarked    <chr> "S", "C", "S", "S", "S", "Q", "S", "S", "S", "C", ...
```

```r
# Add the factors
train <- train %>%
  mutate(Survived_Category = case_when(Survived == 1 ~ 'Yes',
                                       Survived == 0 ~ 'No'))%>%
  mutate(Pclass_Category = case_when(Pclass == 1 ~ 'First',
                                     Pclass == 2 ~ 'Second',
                                     Pclass == 3 ~ 'Third'))
# glimpse train again
glimpse(train)
```

```
## Observations: 891
## Variables: 14
## $ PassengerId       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 1...
## $ Survived          <int> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,...
## $ Pclass            <int> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3,...
## $ Name              <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. Jo...
## $ Sex               <chr> "male", "female", "female", "female", "male"...
## $ Age               <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58...
## $ SibSp             <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0,...
## $ Parch             <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0,...
## $ Ticket            <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282",...
## $ Fare              <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8....
## $ Cabin             <chr> NA, "C85", NA, "C123", NA, NA, "E46", NA, NA...
## $ Embarked          <chr> "S", "C", "S", "S", "S", "Q", "S", "S", "S",...
## $ Survived_Category <chr> "No", "Yes", "Yes", "Yes", "No", "No", "No",...
## $ Pclass_Category   <chr> "Third", "First", "Third", "First", "Third",...
```

Split the data to *training set* and *test set*

Exercise 2

Our first step is to randomly split the data into train and test datasets. We will use a 70/30 split, and use the random seed of 29283 so that we all should get the same training and test set.

Answer:

```r
set.seed(29283)
# Let's create our training set using sample_frac.
train_set <- train %>% sample_frac(0.7)
# Print train set
train_set
```

```
## # A tibble: 624 x 14
##    PassengerId Survived Pclass Name  Sex      Age SibSp Parch Ticket   Fare
##          <int>    <int>  <int> <chr> <chr> <dbl> <int> <int> <chr>   <dbl>
##  1          13        0      3 Saun~ male   20.0     0     0 A/5. ~   8.05
## 2          389        0       3 Sadl~ male   NA       0     0 367655   7.73
## 3           74        0       3 Chro~ male   26.0     1     0 2680    14.5
## 4          351        0       3 Odah~ male   23.0     0     0 7267     9.22
## 5          867        1       2 Dura~ fema~  27.0     1     0 SC/PA~  13.9
## 6          712        0       1 Klab~ male   NA       0     0 113028  26.6
## 7          768        0       3 Mang~ fema~  30.5     0     0 364850   7.75
## 8          803        1       1 Cart~ male   11.0     1     2 113760 120
## 9           71        0       2 Jenk~ male   32.0     0     0 C.A. ~  10.5
## 10         684        0       3 Good~ male   14.0     5     2 CA 21~  46.9
## # ... with 614 more rows, and 4 more variables: Cabin <chr>,
## #   Embarked <chr>, Survived_Category <chr>, Pclass_Category <chr>
```

```r
# let's create our testing set using the PassangerId column. Fill in the blanks.
test_set <- train %>% filter(!(train$PassengerId %in% train_set$PassengerId))
# Print test set
test_set
```

```
## # A tibble: 267 x 14
##    PassengerId Survived Pclass Name  Sex     Age SibSp Parch Ticket   Fare
##          <int>    <int>  <int> <chr> <chr> <dbl> <int> <int> <chr>   <dbl>
## 1            1        0      3 Brau~ male  22.0      1     0 A/5 2~   7.25
## 2            2        1      1 Cumi~ fema~ 38.0      1     0 PC 17~  71.3
## 3            5        0      3 Alle~ male  35.0      0     0 373450   8.05
## 4           12        1      1 Bonn~ fema~ 58.0      0     0 113783  26.6
## 5           19        0      3 Vand~ fema~ 31.0      1     0 345763  18.0
## 6           23        1      3 "McG~ fema~ 15.0      0     0 330923   8.03
## 7           25        0      3 Pals~ fema~  8.00     3     1 349909  21.1
## 8           28        0      1 Fort~ male  19.0      3     2 19950  263
## 9           30        0      3 Todo~ male  NA        0     0 349216   7.90
## 10          33        1      3 Glyn~ fema~ NA        0     0 335677   7.75
## # ... with 257 more rows, and 4 more variables: Cabin <chr>,
## #   Embarked <chr>, Survived_Category <chr>, Pclass_Category <chr>
```

EDA and Logistic Regression

Exercise 3

Our target is called *Survived*. First, fit a logistic regression model using *Pclass*, *Sex*, *Fare* as your three features. Fit the model using the glm() function.

Ask yourself these questions before fitting the model:

- **What kind of relationship will these features have with the probability of survival?**
- **Are these good features, given the problem we are trying to solve?**

After fitting the model, output the coefficients using the broom package and answer these questions:

- **How would you interpret the coefficients?**
- **Are the features significant?**

Use the code below and fill in the blanks.

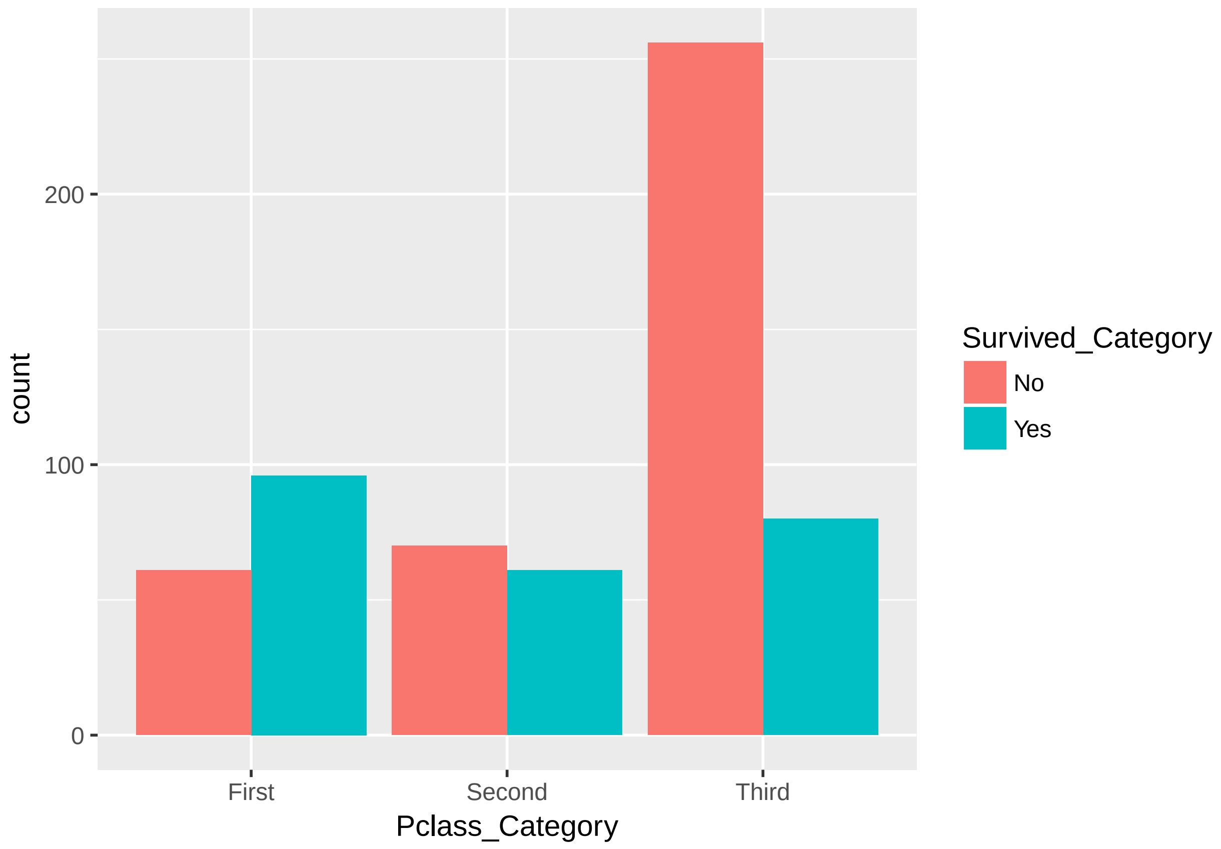
```r
library(broom)
# Fit a model with intercept only
mod_1 <- glm(_______ ~ _____, data = _____, family = 'binomial')
```

```
# take a look at the features and coefficients
tidy(____)
```

Answer:

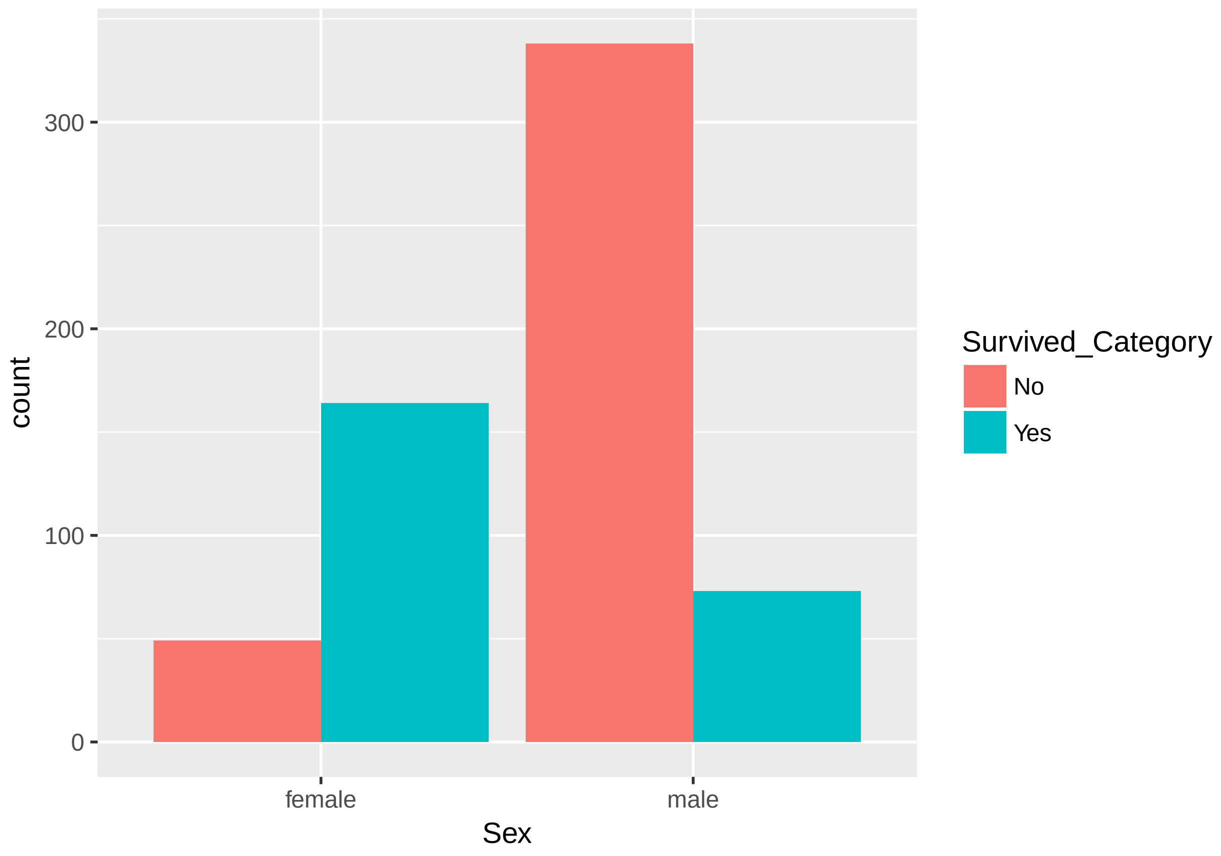
Let's plot the relationships:

```
# Plot for Pclass
ggplot(train_set, aes(Pclass_Category, ..count..)) +
  geom_bar(aes(fill = Survived_Category), position = "dodge")
```



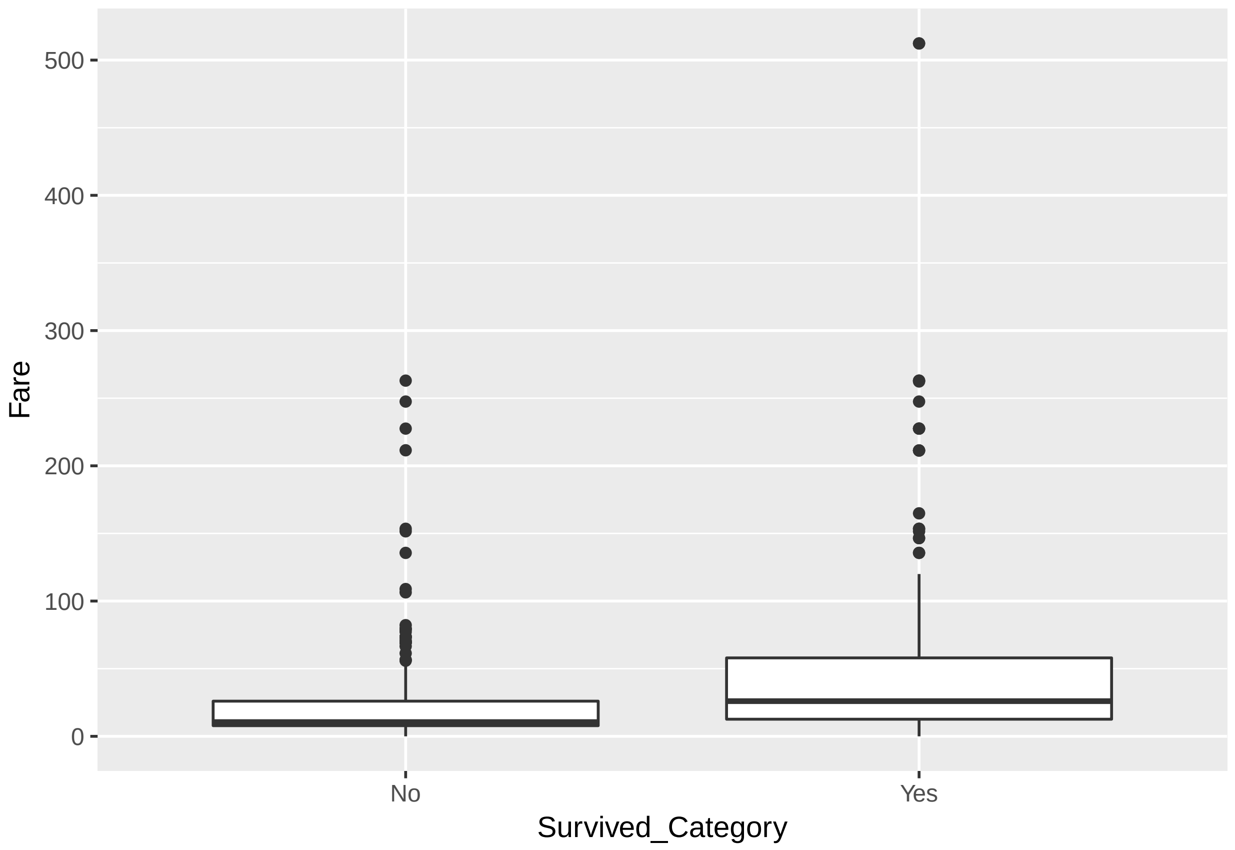
It looks like more percentage of passengers survived in the first class and second class than third class.

```
# Plot for Sex
ggplot(train_set, aes(Sex, ..count..)) +
  geom_bar(aes(fill = Survived_Category), position = "dodge")
```



It looks like more female passengers got survived compared to male passengers.

```r
# Plot for Fare
train_set %>% ggplot(aes(x = Survived_Category, y = Fare)) +
  geom_boxplot()
```



People who paid more, survived (may be they are first class and second class passengers)

Now, let's create the model.

```r
# Fit a model with intercept only
mod_1 <- glm(Survived ~ Pclass_Category + Sex + Fare, data = train_set, family = 'binomial')
# take a look at the features and coefficients
tidy(mod_1)
```

```
##                   term     estimate   std.error   statistic      p.value
## 1          (Intercept)  2.184273582 0.328033129   6.6586981 2.762637e-11
## 2 Pclass_CategorySecond -0.633237694 0.321525515  -1.9694788 4.889813e-02
## 3  Pclass_CategoryThird -1.701756611 0.298007147  -5.7104557 1.126741e-08
## 4              Sexmale -2.834542001 0.229213455 -12.3663857 3.973000e-35
## 5                 Fare  0.002266955 0.002360593   0.9603331 3.368876e-01
```

Review of Coefficients

- **Pclass : -0.875841345** - It's negative, means higher class (number is lower) people are more probable to survive.
- **Sex-male : -2.840421241** - It's negative, means male passengers have less likelihood to survive.
- **Fare : 0.001846965** - It's positive with value tends to zero. That means people paid more have more probability to survive. Although looks like not a very significant feature.

Significance of the Features - From the coefficient values, it looks like Pclass and Sex is more significant feature to predict. Fare looks like less significant.

Decision Tree

Exercise 4

Now, let's fit a model using a classification tree, using the same features and plot the final decision tree. Use the code below for help.

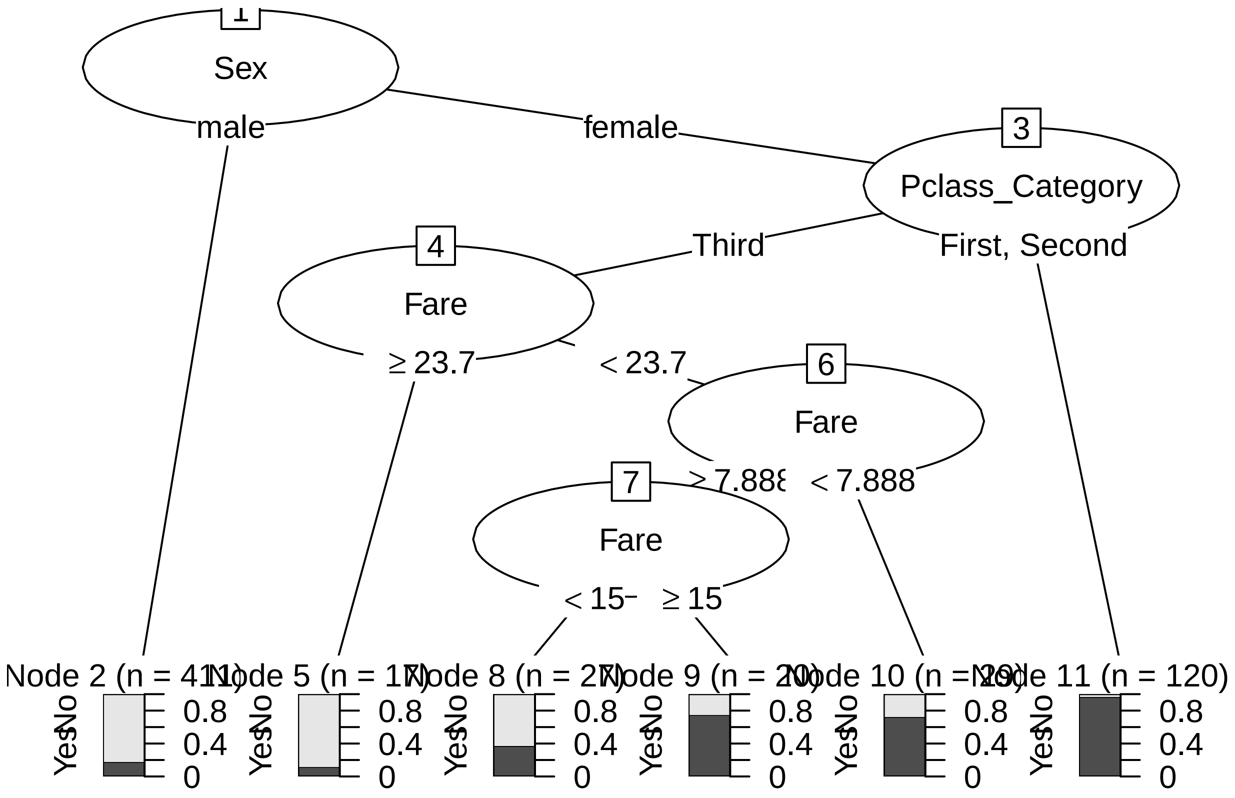
Answer these questions:

- **Describe in words one path a Titanic passenger might take down the tree. (Hint: look at your tree, choose a path from the top to a terminal node, and describe the path like this - a male passenger who paid a fare > 30 and was in first class has a high probability of survival)**
- **Does anything surprise you about the fitted tree?**

```
library(rpart)
library(partykit)
tree_mod <- rpart(______ ~ ______, data = _____)
plot(as.party(tree_mod))
```

Answer:

```
tree_mod <- rpart(Survived_Category ~ Pclass_Category + Sex + Fare, data = train_set)
plot(as.party(tree_mod))
```



In the above decision tree, we see there are 6 leaf nodes. We can describe the paths as below (for example):

- **Node 2:** - A male passenger has less probability(~20%) of survival.
- **Node 5:** - A female passenger from third Pclass and paid fare >= 23.7 has less probability(~ <20%) of survival.

- **Node 5:** - A female passenger from third Pclass and paid fare <15 and >=7.8 has ~40% probability of survival.

On the above tree plot, the feature Sex is a strong predictor compared to the other two.

Model Evaluation

Exercise 5

Evaluate both the logistic regression model and classification tree on the test_set. First, use the predict() function to get the model predictions for the testing set. Use the code below for help.

- **It may seem odd that we are using the same predict() function to make predictions for two completely different types of models (logistic regression and classification tree). This is a feature of R that there are many generic functions that you can apply to different R objects. Depending on the class of the object that is passed to the generic function, R will know which definition of the generic function to use on that object.**

```r
test_logit <- predict(mod_1, newdata = test_set, type = 'response')
test_tree <- predict(tree_mod, newdata = test_set)[,2]
```

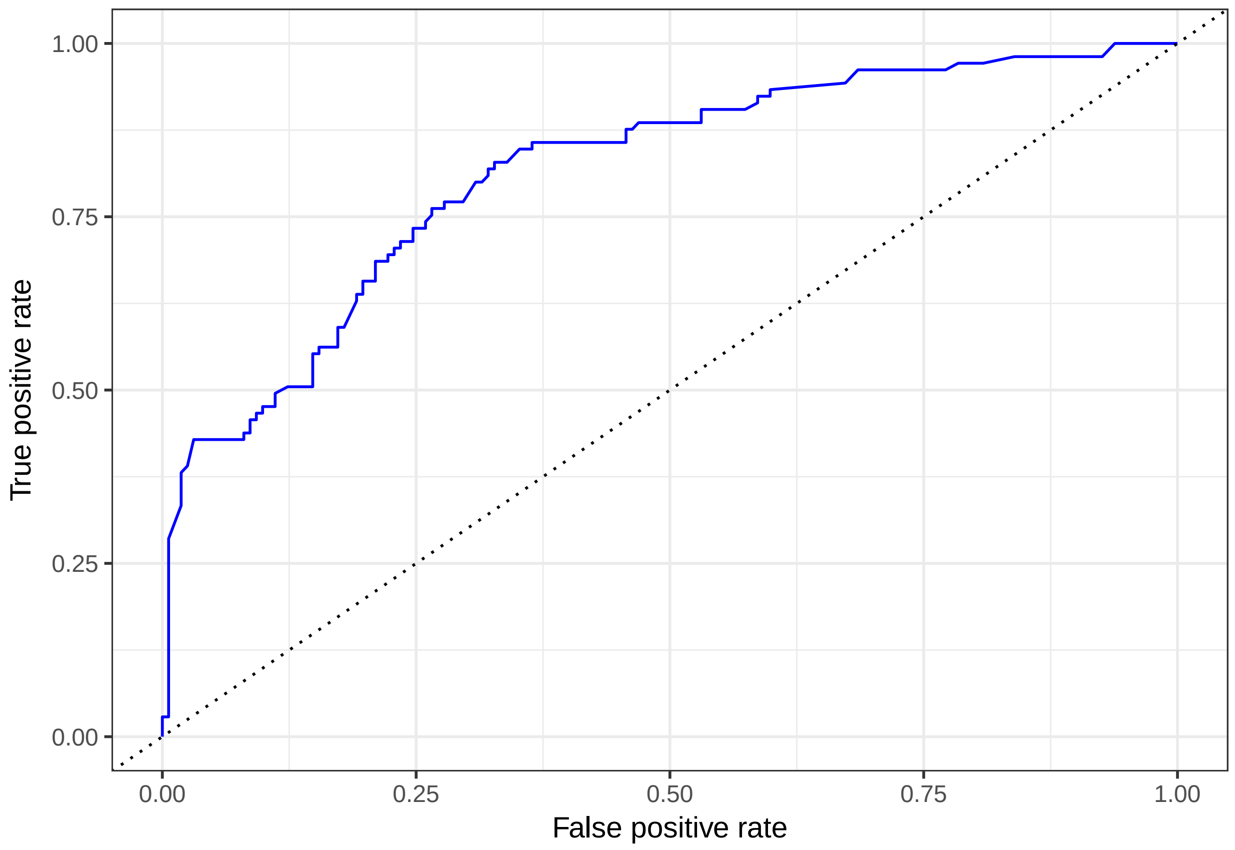
(a) Next, we will plot the ROC curves from both models using the code below. Don't just copy and paste the code. Go through it line by line and see what it is doing. Recall that predictions from your decision tree are given as a two column matrix.

```r
library(ROCR)
# create the prediction objects for both models
pred_logit <- prediction(predictions = test_logit, labels = test_set$Survived)
pred_tree <- prediction(predictions = test_tree, labels = test_set$Survived)
# get the FPR and TPR for the logistic model
# recall that the ROC curve plots the FPR on the x-axis
perf_logit <- performance(pred_logit, measure = 'tpr', x.measure = 'fpr')
perf_logit_tbl <- tibble(perf_logit@x.values[[1]], perf_logit@y.values[[1]])
# Change the names of the columns of the tibble
names(perf_logit_tbl) <- c('fpr', 'tpr')
# get the FPR and TPR for the tree model
perf_tree <- performance(pred_tree, measure = 'tpr', x.measure = 'fpr')
perf_tree_tbl <- tibble(perf_tree@x.values[[1]], perf_tree@y.values[[1]])
# Change the names of the columns of the tibble
names(perf_tree_tbl) <- c('fpr', 'tpr')
# Plotting function for plotting a nice ROC curve using ggplot
plot_roc <- function(perf_tbl) {
p <- ggplot(data = perf_tbl, aes(x = fpr, y = tpr)) +
geom_line(color = 'blue') +
geom_abline(intercept = 0, slope = 1, lty = 3) +
labs(x = 'False positive rate', y = 'True positive rate') +
theme_bw()
return(p)
}
# Create the ROC curves using the function we created above
plot_roc(perf_logit_tbl)
plot_roc(perf_tree_tbl)
```

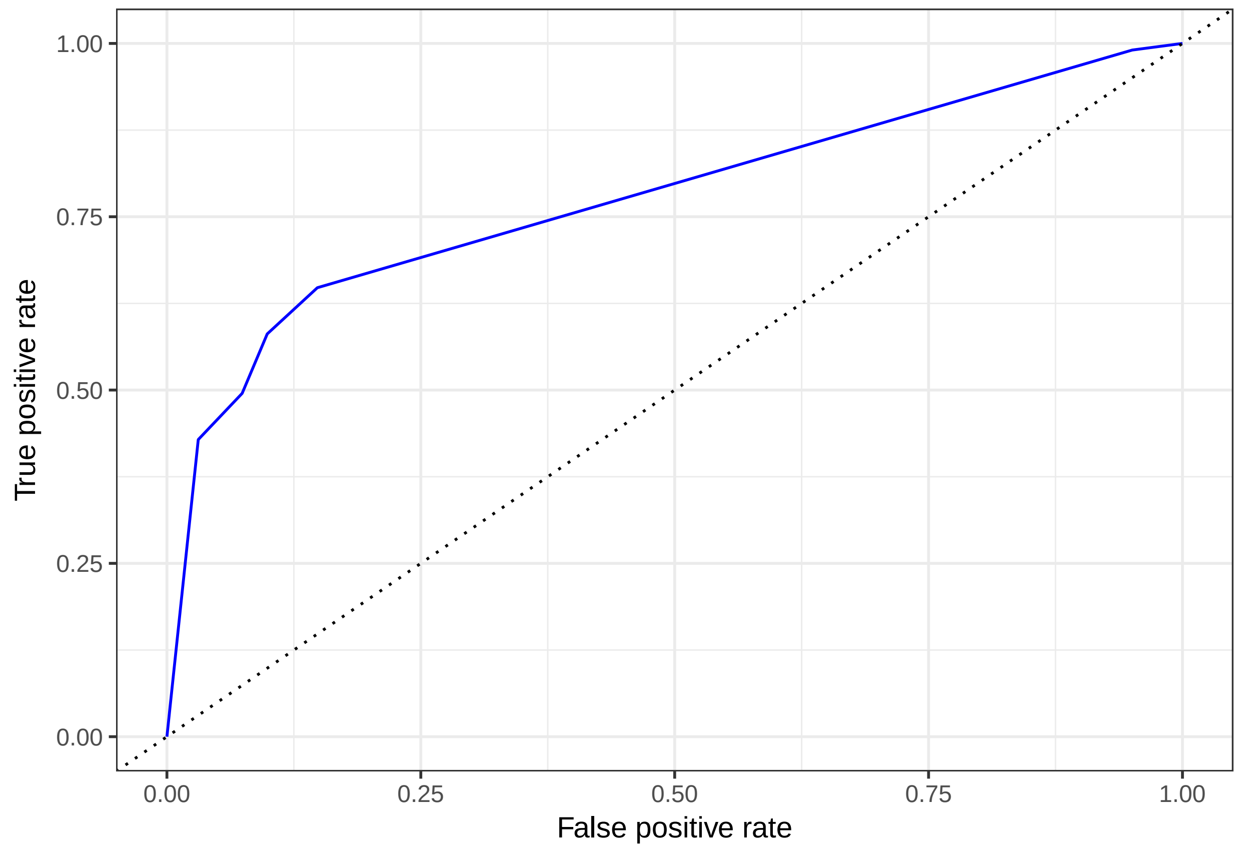
Answer:

Let's execute the above code.

```r
test_logit <- predict(mod_1, newdata = test_set, type = 'response')
test_tree <- predict(tree_mod, newdata = test_set)[,2]
# create the prediction objects for both models
pred_logit <- prediction(predictions = test_logit, labels = test_set$Survived)
pred_tree <- prediction(predictions = test_tree, labels = test_set$Survived_Category)
# get the FPR and TPR for the logistic model
# recall that the ROC curve plots the FPR on the x-axis
perf_logit <- performance(pred_logit, measure = 'tpr', x.measure = 'fpr')
perf_logit_tbl <- tibble(perf_logit@x.values[[1]], perf_logit@y.values[[1]])
# Change the names of the columns of the tibble
names(perf_logit_tbl) <- c('fpr', 'tpr')
# get the FPR and TPR for the tree model
perf_tree <- performance(pred_tree, measure = 'tpr', x.measure = 'fpr')
perf_tree_tbl <- tibble(perf_tree@x.values[[1]], perf_tree@y.values[[1]])
# Change the names of the columns of the tibble
names(perf_tree_tbl) <- c('fpr', 'tpr')
# Plotting function for plotting a nice ROC curve using ggplot
plot_roc <- function(perf_tbl) {
p <- ggplot(data = perf_tbl, aes(x = fpr, y = tpr)) +
geom_line(color = 'blue') +
geom_abline(intercept = 0, slope = 1, lty = 3) +
labs(x = 'False positive rate', y = 'True positive rate') +
theme_bw()
return(p)
}
# Create the ROC curves using the function we created above
plot_roc(perf_logit_tbl)
```



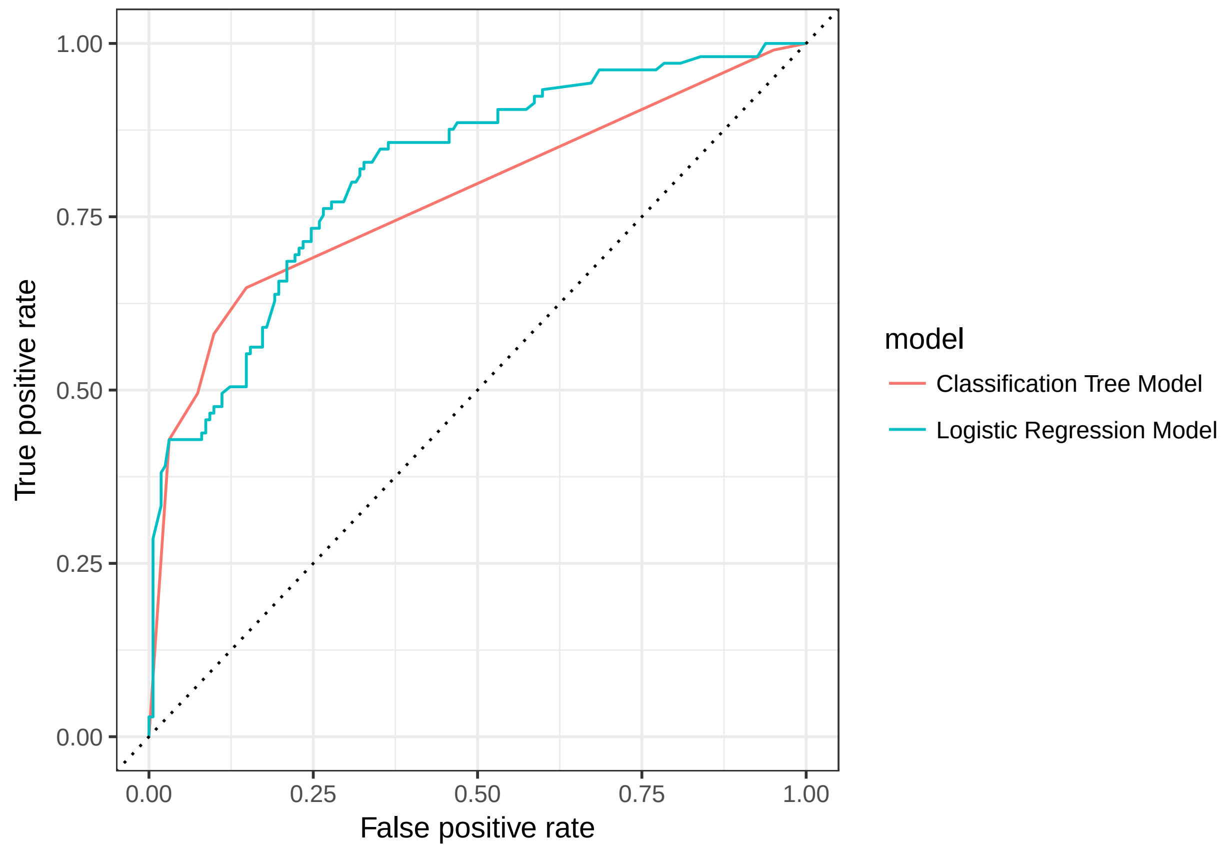
```
plot_roc(perf_tree_tbl)
```



```r
# Plotting function for plotting multiple ROC in a single plot
plot_multi_roc <- function(perf_tbl) {
p <- ggplot(data = perf_tbl, aes(x = fpr, y = tpr, group=model, colour=model)) +
geom_line() +
geom_abline(intercept = 0, slope = 1, lty = 3) +
labs(x = 'False positive rate', y = 'True positive rate') +
theme_bw()
return(p)
}
# Create ROC in a single plot for both the model
t1 <- perf_logit_tbl %>% mutate(model = 'Logistic Regression Model')
t2 <- perf_tree_tbl %>% mutate(model = 'Classification Tree Model')
t3 <- t1 %>% full_join(t2)
```

```
## Joining, by = c("fpr", "tpr", "model")
```

```r
plot_multi_roc(t3)
```



(b) Now, use the performance() function to calculate the area under the curve (AUC) for both ROC curves. Check ?performance for help on plugging in the right measure argument.

```
# calculate the AUC
auc_logit <- performance(______, measure = ____)
auc_tree <- performance(_____, measure = ____)
# extract the AUC value
auc_logit@y.values[[1]]
auc_tree@y.values[[1]]
```

What do you notice about the ROC curves and the AUC values? Are the models performing well? Is the logistic regression model doing better, worse, or about the same as the classification tree?

Answer:

Let's fill in the blanks.

```
# calculate the AUC
auc_logit <- performance(pred_logit, measure = 'auc')
auc_tree <- performance(pred_tree, measure = 'auc')
# extract the AUC value
auc_logit@y.values[[1]]
```

```
## [1] 0.8147854
```

```
auc_tree@y.values[[1]]
```

```
## [1] 0.776602
```

Conclusion: Both the ROC curves and AUC values looks fine (not bad). The AUC values for both are >

0.5, means both the model is performing well.

The AUC value is higher for logistic regression model, so, it is doing better than the classification tree model.

(c) Lastly, pick a probability cutoff by looking at the ROC curves. You pick, there's no right answer (but there is a wrong answer - make sure to pick something between 0 and 1). Using that probability cutoff, create the confusion matrix for each model by following these steps:

1. Pick a cutoff value.

2. Append the predicted probability values from each model (you created these at the beginning of Exercise 5) to your test_set tibble using mutate().

3. Create a new column for the predicted class from each model using mutate() and case_when(). Your new predicted class columns can have two possible values: yes or no which represents whether or not the passenger is predicted to have survived or not given the predicted probability.

4. You should now have 4 extra columns added to your test_set tibble, two columns of predicted probabilities, and two columns of the predicted categories based on your probability cutoff.

5. Now create the table using the code below:

```
test_set %>% count(_____, Survived) %>% spread(Survived, n)
test_set %>% count(_____, Survived) %>% spread(Survived, n)
```

If you choose a cutoff of 0.25, your confusion tables should look like this:

```
## Logistic model:
## # A tibble: 2 x 3

##   class_logit `0` `1`
## * <chr> <int> <int>
## 1 No 112 21
## 2 Yes 50 84

## Classification tree model:
## # A tibble: 2 x 3

## class_tree `0` `1`
## * <chr> <int> <int>
## 1 No 138 37
## 2 Yes 24 68
```

Answer:

Let's choose the cutoff 0.25 and create the confusion matrix.

```
# Mutate the probability and category for logistic regression predictions
test_set_logit_result <- test_set %>% mutate(preds_prob1 = test_logit) %>%
  mutate(preds_cat1 = case_when(preds_prob1 < .25 ~ 'No',
                                preds_prob1 >= .25 ~ 'Yes'))
# See the count
test_set_logit_result %>% count(preds_cat1)
```

```
## # A tibble: 2 x 2
##   preds_cat1     n
##   <chr>      <int>
## 1 No           133
## 2 Yes          134
```

```
# Mutate the probability and category for classification tree predictions
test_set_tree_result <- test_set %>% mutate(preds_prob2 = test_tree) %>%
  mutate(preds_cat2 = case_when(preds_prob2 < .25 ~ 'No',
                                preds_prob2 >= .25 ~ 'Yes'))
# See the count
test_set_tree_result %>% count(preds_cat2)
```

```
## # A tibble: 2 x 2
##   preds_cat2     n
##   <chr>      <int>
## 1 No           175
## 2 Yes           92
```

```
# Confusion matrix for Logistic model
test_set_logit_result %>% count(preds_cat1, Survived) %>% spread(Survived, n)
```

```
## # A tibble: 2 x 3
##   preds_cat1   `0`   `1`
##   <chr>      <int> <int>
## 1 No           112    21
## 2 Yes           50    84
```

```
# Confusion matrix for Classification tree model
test_set_tree_result %>% count(preds_cat2, Survived) %>% spread(Survived, n)
```

```
## # A tibble: 2 x 3
##   preds_cat2   `0`   `1`
##   <chr>      <int> <int>
## 1 No           138    37
## 2 Yes           24    68
```

It matches with the provided matrix.

Determine threshold values from ROC curve (additional)

```
# Create cutoffs matrix for Logistic Regression model
logit_cutoffs <- data.frame(cut=perf_logit@alpha.values[[1]], fpr=perf_logit@x.values[[1]], tpr=perf_log
logit_cutoffs <- logit_cutoffs[order(logit_cutoffs$tpr, decreasing = TRUE),]
head(subset(logit_cutoffs, fpr<0.2))
```

```
##          cut       fpr       tpr
## 74 0.6222020 0.1975309 0.6571429
## 73 0.6222131 0.1975309 0.6476190
## 71 0.6224706 0.1913580 0.6380952
## 72 0.6223730 0.1975309 0.6380952
## 70 0.6224795 0.1913580 0.6285714
## 68 0.6225217 0.1728395 0.5904762
```

```
# Create cutoffs matrix for Classification Tree model
tree_cutoffs <- data.frame(cut=perf_tree@alpha.values[[1]], fpr=perf_tree@x.values[[1]], tpr=perf_tree@
tree_cutoffs <- tree_cutoffs[order(tree_cutoffs$tpr, decreasing = TRUE),]
head(subset(tree_cutoffs, fpr<0.2))
```

```
##         cut        fpr       tpr
## 5 0.3703704 0.14814815 0.6476190
## 4 0.7241379 0.09876543 0.5809524
## 3 0.7500000 0.07407407 0.4952381
## 2 0.9666667 0.03086420 0.4285714
## 1       Inf 0.00000000 0.0000000
```