COMPSCIX 415.2 Homework 9/Final

Sanatan Das

March 29, 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(modelr)
library(randomForest)
```

Bootstrapping (10 points)

Load the Titanic dataset

1. **Follow these steps:**
    - **Load the train.csv dataset into R.**
    - **Convert all character columns into unordered factors.**
    - **Convert the Survived column into an unordered factor because it is loaded as an integer by default.**
    - **Take a glimpse of your data to confirm that all of the columns were converted correctly.**

We will use this same dataset for this entire assignment.

Answer:

```
# load train.csv
train_original = read_csv("C:/view/opt/apps/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()
## )
```

```r
# glimpse train
glimpse(train_original)
```

```
## 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
# copy to a different dataset
train <- data.frame(train_original)
# 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'))
train$Name <- as.factor(train$Name)
train$Sex <- as.factor(train$Sex)
train$Ticket <- as.factor(train$Ticket)
train$Cabin <- as.factor(train$Cabin)
train$Embarked <- as.factor(train$Embarked)
# 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              <fct> Braund, Mr. Owen Harris, Cumings, Mrs. John ...
## $ Sex               <fct> male, female, female, female, male, male, ma...
## $ 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            <fct> A/5 21171, PC 17599, STON/O2. 3101282, 11380...
## $ Fare              <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8....
## $ Cabin             <fct> NA, C85, NA, C123, NA, NA, E46, NA, NA, NA, ...
## $ Embarked          <fct> S, C, S, S, S, Q, S, S, S, C, S, S, S, S, S,...
## $ Survived_Category <chr> "No", "Yes", "Yes", "Yes", "No", "No", "No",...
## $ Pclass_Category   <chr> "Third", "First", "Third", "First", "Third",...
```

Now all the character columns are converted to unordered factors.

Create bootstrap samples (n=100)

2. Use the code below to take 100 bootstrap samples of your data. Confirm that the result is a tibble with a list column of resample objects - each resample object is a bootstrap sample of the titanic dataset.

```
library(tidyverse)
library(modelr)
titanic_boot <- bootstrap(data = ____, n = ___)
```

Answer:

Let's fill in the code.

```
titanic_boot <- bootstrap(data = train, n = 100)
#see the type of titanic_boot
class(titanic_boot)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
# see titanic_boot
titanic_boot
```

```
## # A tibble: 100 x 2
##    strap          .id
##    <list>         <chr>
##  1 <S3: resample> 001
##  2 <S3: resample> 002
##  3 <S3: resample> 003
##  4 <S3: resample> 004
##  5 <S3: resample> 005
##  6 <S3: resample> 006
##  7 <S3: resample> 007
##  8 <S3: resample> 008
##  9 <S3: resample> 009
## 10 <S3: resample> 010
## # ... with 90 more rows
```

```
# see the type of elements
class(titanic_boot$strap[[1]])
```

```
## [1] "resample"
```

We can see that a tibble of 100 rows is created with a list column of resample objects - each resample object is a bootstrap sample of the titanic dataset.

Verify bootstrap samples

3. Confirm that some of your bootstrap samples are in fact bootstrap samples (meaning they should have some rows that are repeated). You can use the *n_distinct()* function from dplyr to see that your samples have different numbers of unique rows. Use the code below to help you extract some of the resample objects from the strap column (which is an R list), convert them to tibbles, and then count distinct rows. Use the code below, no changes necessary.

```
# since the strap column of titanic_boot is a list, we can
# extract the resampled data using the double brackets [[]],
# and just pick out a few of them to compare the number of
# distinct rows
```

```
as.tibble(titanic_boot$strap[[1]]) %>% n_distinct()
as.tibble(titanic_boot$strap[[2]]) %>% n_distinct()
as.tibble(titanic_boot$strap[[3]]) %>% n_distinct()
```

Answer:

Let's run the code.

```
# since the strap column of titanic_boot is a list, we can
# extract the resampled data using the double brackets [[]],
# and just pick out a few of them to compare the number of
# distinct rows

# not distinct
as.tibble(titanic_boot$strap[[1]]) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   891
```

```
# distinct
as.tibble(titanic_boot$strap[[1]]) %>% n_distinct()
```

```
## [1] 552
```

```
# not distinct
as.tibble(titanic_boot$strap[[2]]) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   891
```

```
# distinct
as.tibble(titanic_boot$strap[[2]]) %>% n_distinct()
```

```
## [1] 574
```

```
# not distinct
as.tibble(titanic_boot$strap[[3]]) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   891
```

```
# distinct
as.tibble(titanic_boot$strap[[3]]) %>% n_distinct()
```

```
## [1] 574
```

From above, ew see that every tibble has 891 rows (length of the dataset) but the number distinct rows are less than that. That means some rows are repeated.

Demonstration of Central Limit Theorem

4. Now, let's demonstrate the Central Limit Theorem using the Age column. We'll iterate through all 100 bootstrap samples, take the mean of Age, and collect the results.

- **We will define our own function to pull out the mean of Age from each bootstrap sample and**
- **create our own for loop to iterate through.**

Use the code below and fill in the blanks.

```
age_mean <- function(___) {
data <- as.tibble(___) # convert input data set to a tibble
mean_age <- mean(data$Age, na.rm = TRUE) # take the mean of Age, remove NAs
return(____) # return the mean value of Age from data
}
# loop through the 100 bootstrap samples and use the age_mean()
# function
all_means <- rep(NA, 100)
# start the loop
for(i in 1:___) {
all_means[_] <- age_mean(titanic_boot$strap[[i]])
}
# take a look at some of the means you calculated from your samples
head(all_means)
# convert to a tibble so we can use if for plotting
all_means <- tibble(all_means = all_means)
```

Answer:

Let's fill in the code.

```
age_mean <- function(input) {
data <- as.tibble(input) # convert input data set to a tibble
mean_age <- mean(data$Age, na.rm = TRUE) # take the mean of Age, remove NAs
return(mean_age) # return the mean value of Age from data
}
# loop through the 100 bootstrap samples and use the age_mean()
# function
all_means <- rep(NA, 100)
# start the loop
for(i in 1:100) {
all_means[i] <- age_mean(titanic_boot$strap[[i]])
}
# take a look at some of the means you calculated from your samples
head(all_means)
```

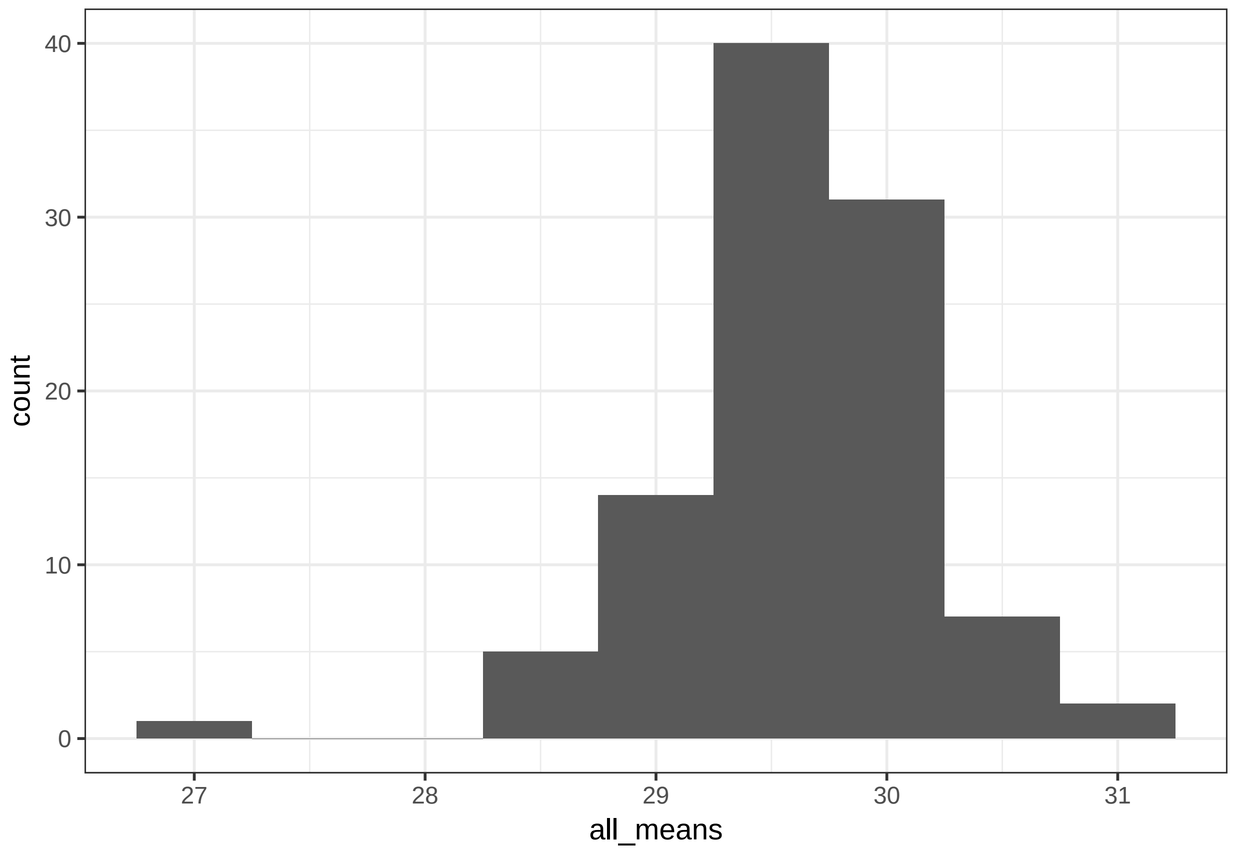
```
## [1] 27.21667 29.68125 29.98714 29.30935 30.29329 29.40264
```

```
# convert to a tibble so we can use if for plotting
all_means <- tibble(all_means = all_means)
```

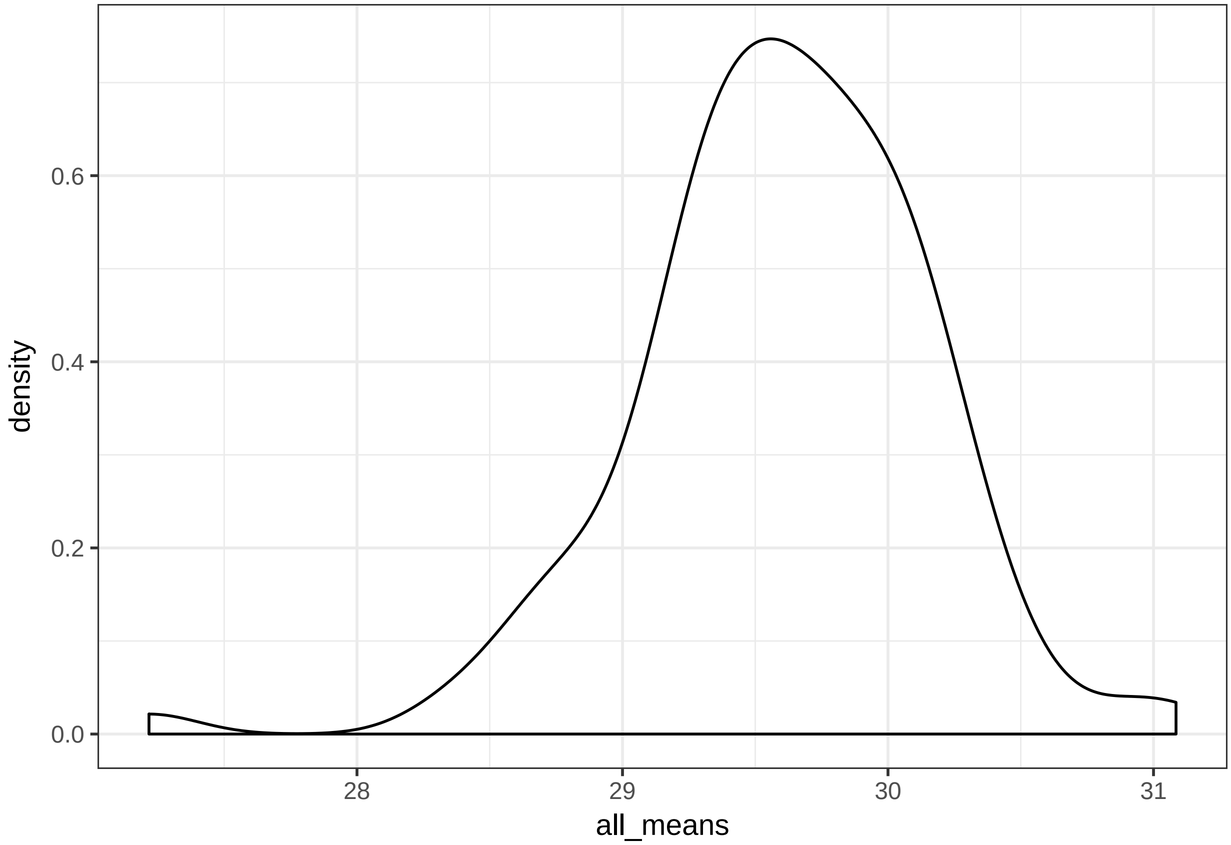
5. Plot a histogram of *all_means.*

The plot is as below.

```
# Histogram
ggplot(data=all_means) +
  geom_histogram(aes(x = all_means), binwidth = 0.5) +
  theme_bw()
```



```
# Density Curve
ggplot(data=all_means) +
  geom_density(aes(x = all_means)) +
  theme_bw()
```



6. Find the standard error of the sample mean of Age using your boostrap sample means. Compare the empirical standard error to the theoretical standard error.

Answer:

First find out the standard error without resampling (theoritical standard error).

```
# theoritical standard error
lm_fit <- lm(Survived ~ Age, data = train)
tidy(lm_fit) %>% select(term, std.error)
```

```
##          term  std.error
## 1 (Intercept) 0.04178795
## 2         Age 0.00126412
```

Now find out bootstrap standard error.

```
# function to take a bootstrap sample,
# fit the linear regression model,
# and return the coefficients
boot_fun <- function(data_samp) {
  lm_samp <- lm(Survived ~ Age, data = data_samp)
  return(tidy(lm_samp)$estimate)
}
# create empty vectors to save the output
intercepts <- rep(NA, 100)
slopes <- rep(NA, 100)
for(i in 1:100) { # loop over the sequence 1,2,3,...1000
  coefs <- boot_fun(titanic_boot$strap[[i]])
  intercepts[i] <- coefs[1] # save the output
  slopes[i] <- coefs[2] # save the output
}
# convert to tibble
all_results <- tibble(intercepts, slopes)
# bootstrap standard error
all_results %>% summarize(se_intercept = sd(intercepts),
                          se_slope = sd(slopes))
```

```
## # A tibble: 1 x 2
##   se_intercept se_slope
##          <dbl>    <dbl>
## 1       0.0410  0.00116
```

From the above result, we can see that the bootstrap standard error is less compared to standard error without resampling (theoritical standard error)

Random forest (10 points)

Split the dataset

On the last homework, we fit a decision tree to the Titanic data set to predict the probability of survival given the features. This week we'll use the random forest and compare our results to the decision tree.

1. Randomly split your data into training and testing using the code below so that we all have the same sets.

```
set.seed(987)
model_data <- resample_partition(____, c(test = 0.3, train = 0.7))
train_set <- as.tibble(model_data$____)
test_set <- as.tibble(model_data$____)
```

Answer:

```
set.seed(987)
train$Survived <- as.factor(train$Survived)
model_data <- resample_partition(train, c(test = 0.3, train = 0.7))
train_set <- as.tibble(model_data$train)
test_set <- as.tibble(model_data$test)

train_set
```

```
## # A tibble: 624 x 14
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare
##  *       <int> <fct>     <int> <fct>  <fct> <dbl> <int> <int> <fct>  <dbl>
##  1           1 0             3 Braun~ male   22.0      1     0 A/5 2~  7.25
##  2           2 1             1 Cumin~ fema~ 38.0      1     0 PC 17~ 71.3
##  3           3 1             3 Heikk~ fema~ 26.0      0     0 STON/~  7.92
##  4           4 1             1 Futre~ fema~ 35.0      1     0 113803 53.1
##  5           5 0             3 Allen~ male   35.0      0     0 373450  8.05
##  6           7 0             1 McCar~ male   54.0      0     0 17463  51.9
##  7           8 0             3 Palss~ male    2.00     3     1 349909 21.1
##  8           9 1             3 Johns~ fema~ 27.0      0     2 347742 11.1
##  9          10 1             2 Nasse~ fema~ 14.0      1     0 237736 30.1
## 10          12 1             1 Bonne~ fema~ 58.0      0     0 113783 26.6
## # ... with 614 more rows, and 4 more variables: Cabin <fct>,
## #   Embarked <fct>, Survived_Category <chr>, Pclass_Category <chr>
```

```
test_set
```

```
## # A tibble: 267 x 14
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare
##  *       <int> <fct>     <int> <fct>  <fct> <dbl> <int> <int> <fct>  <dbl>
##  1           6 0             3 Moran~ male  NA        0     0 330877  8.46
##  2          11 1             3 Sands~ fema~  4.00     1     1 PP 95~ 16.7
##  3          13 0             3 Saund~ male  20.0      0     0 A/5. ~  8.05
##  4          15 0             3 Vestr~ fema~ 14.0      0     0 350406  7.85
##  5          17 0             3 Rice,~ male   2.00     4     1 382652 29.1
##  6          19 0             3 Vande~ fema~ 31.0      1     0 345763 18.0
##  7          23 1             3 "McGo~ fema~ 15.0      0     0 330923  8.03
##  8          26 1             3 Asplu~ fema~ 38.0      1     5 347077 31.4
##  9          30 0             3 Todor~ male  NA        0     0 349216  7.90
## 10          35 0             1 Meyer~ male  28.0      1     0 PC 17~ 82.2
## # ... with 257 more rows, and 4 more variables: Cabin <fct>,
## #   Embarked <fct>, Survived_Category <chr>, Pclass_Category <chr>
```

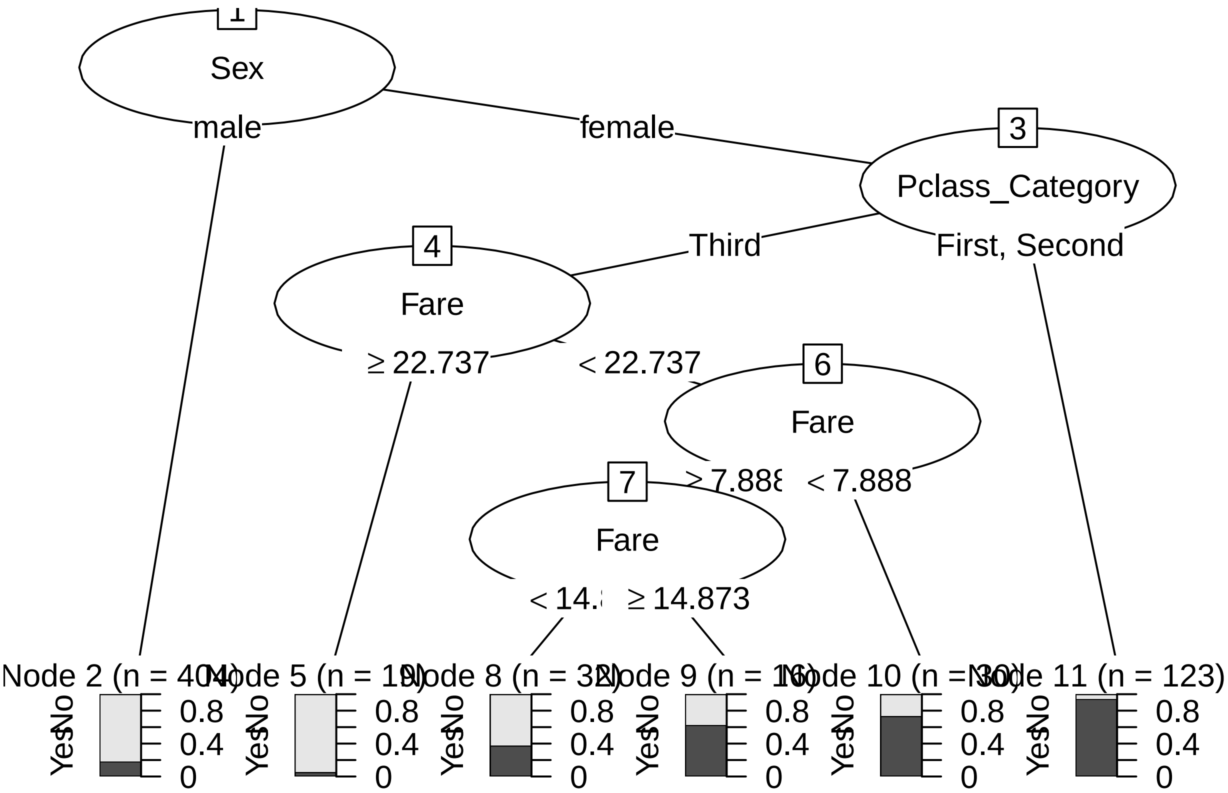
Fit a decision tree

2. Fit a decision tree to *train_set* using the *rpart* package, and using *Pclass*, *Sex*, *Age*, *SibSp*, *Parch*, *Fare*, *Embarked* as the features.

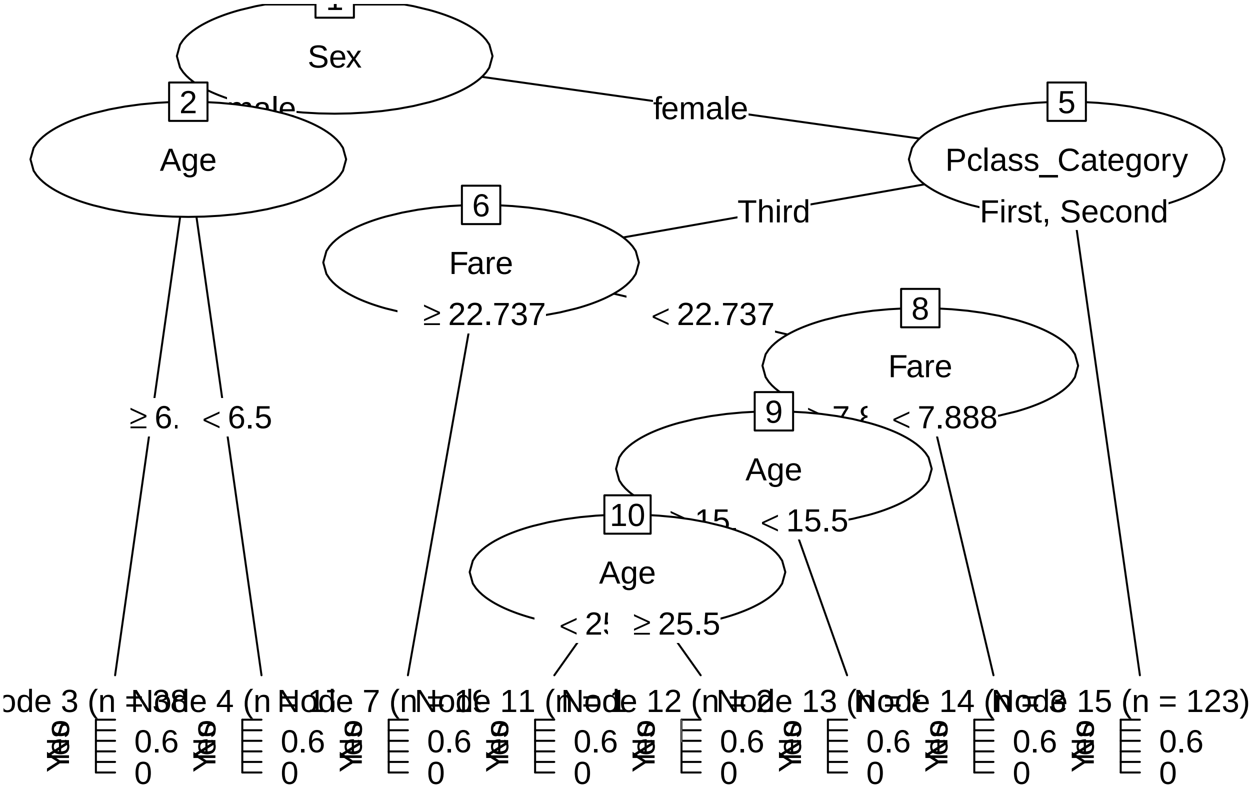
- **Plot the tree using the *partykit* package.**
- **What do you notice about this tree compared to the one from last week which only contained three features?**

Answer:

```
# Last week's tree model
tree_mod_1 <- rpart(Survived_Category ~ Pclass_Category + Sex + Fare, data = train_set)
# plot the tree
plot(as.party(tree_mod_1))
```



```r
# This week's new tree model
tree_mod <- rpart(Survived_Category ~ Pclass_Category + Sex + Age + SibSp + Parch + Fare + Embarked, da
# plot the tree
plot(as.party(tree_mod))
```



Observation: From the above, plots we see that in last week's tree, we had just 3 features and created 6 leaf nodes or categories. This week, we are creating the tree model with 7 features which has created a tree with more levels and 8 leaf nodes. Not much change in Sex=female for First and Second class passengers.

Fit a random forest

3. Fit a random forest to *train_set* using the *randomForest* package, and using *Pclass*, *Sex*, *Age*, *SibSp*, *Parch*, *Fare*, *Embarked* as the features. We'll use 500 trees and sample four features at each split. Use the code below and fill in the blanks.

```
library(randomForest)
rf_mod <- randomForest(____ ~ _____,
                       data = train_set,
                       ntrees = 500,
                       mtry = 4,
                       na.action = na.roughfix)
```

Answer:

Let's fill in the code.

```
rf_mod <- randomForest(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked,
                       data = train_set,
                       ntrees = 500,
                       mtry = 4,
                       na.action = na.roughfix)
```