COMPSCIX 415.2 Homework 9/Final

Robert Clements

DUE DATE: Apr 6, 2018 @ 12:00PM (NOON)

In this assignment/final...

We will review these topics:

- Bootstrapping
- Sampling distributions and standard error
- Random forests
- Decision trees
- Area under the curve (AUC) and ROC Curves
- Functions and loops
- Plotting with ggplot2

Remember to work on this independently. If you have questions you can reach me on Canvas or through email at robert_clements@berkeley.edu.

Remember to make your document look good, which means you may need to change some settings on the figure sizes and locations, use markdown syntax to create headings or to format your text (use the cheatsheet), and you may want to play with the different themes.

Use complete sentences, and **divide your work in a logical way**. Remember that the whole point of doing reproducible analysis in R Markdown is so that a complete stranger can take your results, *understand them*, and reproduce them.

Remember to save and knit often. Commit when you've completed a big chunk of work or when you are done for the day and will be resuming later.

What to Turn In

For this assignment you have two choices:

You can upload a pdf document (you will have to install latex);
You can upload a standalone html file.

To complete this final you will need...

Access to the internet.
The `tidyverse`, `modelr`, `rpart`, `partykit` and `randomForest` packages installed.
RStudio and git/Github.

To start your assignment

1. Go to File -> Recent Projects

2. Click on the Project that you created during Homework 1. This project should be the one that is already under git version control.

3. RStudio will switch to that project and the Git pane should appear.

4. Go to File -> New File and choose **R Markdown**.

5. Change the title ("COMPSCIX 415.2 Homework 9/Final") and add your name and the date to the YAML header.

6. Save the file in the same folder (or create a subfolder) with your other HW assignments and give it the name **firstname_lastname_final.Rmd**.

7. Knit your document into an html or pdf document.
8. Go to the Git pane and commit both your Rmd and html (or pdf) files by clicking on the checkboxes next to the file names and hitting the Commit button. Write a useful message, and hit the commit button.

Exercises (Total Points - 20)

For these exercises we will use the same Titanic data set from last week. You can download it from Canvas if you don't already have it.

Make sure that you have the `modelr`, `rpart`, `partykit` and `randomForest` packages installed.

Bootstrapping (10 points)

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.

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.

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

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.

```r
# 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()
```

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.

```r
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)
```

5. Plot a histogram of `all_means`.
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.

Recall that the theoretical standard error is given by:

$$SE = \frac{\sigma}{\sqrt{n}}$$

where σ is the standard deviation of `Age` and n is the size of our sample.

Random forest (10 points)

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.

```r
set.seed(987)

model_data <- resample_partition(____, c(test = 0.3, train = 0.7))
```

```r
train_set <- as.tibble(model_data$____)
test_set <- as.tibble(model_data$____)
```

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?

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.

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

4. Compare the performance of the decision tree with the random forest using the `ROCR` package and the `AUC`. Which model performs the best?

Here's some code to get you started.

```r
library(ROCR)
rf_preds <- predict(____, newdata = test_set, type = 'prob')[,2]
tree_preds <- predict(____, newdata = test_set)[,2]

pred_rf <- prediction(predictions = _____, labels = ____)
pred_tree <- prediction(predictions = _____, labels = ____)
```

5. Plot the ROC curves for the decision tree and the random forest above, **on the same plot** with a legend that differentiates and specifies which curve belongs to which model. Use the code below to get you started. Hints:

- You **will** have to modify the `plot_roc()` function to plot the two curves together with different colors and a legend.
- This is easier to do if the data for plotting the two curves are in one tibble. You can combine tibbles using the `bind_rows()` function.

```r
# get the FPR and TPR for the logistic model
# recall that the ROC curve plots the FPR on the x-axis
perf_rf <- performance(pred_rf, measure = 'tpr', x.measure = 'fpr')
perf_rf_tbl <- tibble(perf_rf@x.values[[1]], perf_rf@y.values[[1]])

# Change the names of the columns of the tibble
names(perf_rf_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')
```

```r
# 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)
}
```

6. Answer these questions about the ROC curves:

- which model performs better: decision tree or random forest?
- what is the approximate false positive rate, for both the decision tree and the random forest, if we attain a true positive rate of approximately 0.75? Answers do not need to be exact - just ballpark it by looking at the plots.

Turn in your completed final

This week you should turn in your final by uploading it to Canvas by Friday Apr 6 at Noon.