COMPSCIX 415.2 Homework 8

Robert Clements

DUE DATE: Mar 27, 2018 @ 6:15PM

In this assignment. . .

. . . we will work with logistic regression, decision trees, and classification evaluation metrics.

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.

Prerequisite

Basic R Markdown knowledge
R and RStudio
tidyverse, broom, rpart, partykit and ROCR packages installed
Access to internet
git and Github

Exercises

Here are the exercises for you to complete.

Go to the Titanic competition on Kaggle here: https://www.kaggle.com/c/titanic

Read the Overview Description, Evaluation, and the Data sections.

Download the train.csv file either from the Kaggle website, or from Canvas (in the homework section). You **do not** need to download the test.csv file, we will be creating our own test set. To download from Kaggle you will probably have to sign up for an account, so I leave that decision up to you.

Next, download the `rpart`, `partykit`, and `ROCR` packages. We will be using `rpart` for fitting decision trees, `partykit` for visualizing the trees, and `ROCR` for evaluating them.

Answer these questions:

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.

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.

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.

```
library(broom)

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

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

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))
```

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.

```
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.

```
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)
```

(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?

(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
```

Exercise 6 (OPTIONAL - won't be graded)

Feel free to play around with logistic regression and classification trees. Add some other features and see how the model results change. Test the model on `test_set` to compare the ROC curves. Bonus points (still won't be graded) for plotting all ROC curves on the same plot.

Turn in your completed assignment

Commit your changes with the comment "finished assignment 8" and push your R Markdown file and your html or pdf file to Github.

This week you should turn in your assignment by uploading it to Canvas.