CSCE/STAT 587 Big Data Analytics Spring, 2023 Decision Tree Homework

Name ______________________________

Note: Submit your solutions to this HW using Teams!

Part 1: Setting up the data.

Follow the steps from **Part 1** of the Take Home Midterm Exam to set up the Titanic dataset.

Since you have already turned in the steps from Part 1 as part of the midterm take home exam, there is no need to turn this R code in a second time.

Part 2: Decision Tree Analysis of the Titanic dataset

0) We are going to explore the effect of using different complexity parameters on the propensity for Decision Trees to overfitting. In this step you set up the first part of a for-loop that will loop a total of 6 times. Included in the body of this loop are steps 1 through described below. Recall that in the case of Decision Trees we can limit the growth of the tree by specifying parameters such as minsplit and cp. The first line in the body of the loop should set the value of cp to 0.008 for the first iteration. For each subsequent iteration, the value of cp should be half of what it was in the previous iteration. A good way to do this would be to use an `“if (Boolean expression)  { statement block} else { statement block}”` construct.
You should also print out the value of cp for each iteration. A nice way to do this is with the following code:

```
print(sprintf("cpval is",cpval, fmt='%s %#.5f'))
```

Here, ***cpval*** is the variable used to hold the current cp value.
This will give you an output that looks like this:

```
[1] "cpval is 0.00800"
```

Your code for Part 2 will look roughly like this:

```
Step 0:  for-loop { # set it up for 6 iterations
        set/update cpval
        display cpval
        Step 1: for-loop {9-fold cross-validation} # similar to midterm except with decision trees
        Step 2: calculate and display class accuracies to the screen
        Step 3: for-loop {9-fold cross-validation} # similar to midterm except with decision trees
        Step 4: calculate and display class accuracies to the screen
} # end of for-loop
```

1) If a for-loop use the rpart() method from the package rpart, train a Decision Tree model for each of the 9 folds of the training data sets and evaluate on the corresponding 9 test sets. Note: when you evaluate the model you will need to include a 3rd parameter in the predict() function. This parameter is **type=’class’**. Review the slides from the Decision Tree Lab. As in Part 2 of the Take Home Midterm Exam, the arguments to rpart() are the independent features (columns 2 through 6 of training set) and the dependent feature (column 1), the feature you are predicting. In addition, use the parameter values cp= cpval (specified in Step 0) and minsplit=2.

2) **Calculate** and **display** the sensitivity and specificity similarly to what you were expected to do in the take home exam. Please feel free to borrow the following print statements to get a nicely formatted output:

    ```
    print(sprintf("Step 2: The sensitivity on the test partitions is",
    sensitivity(confMat)*100,fmt='%s %#.2f'))
    print(sprintf("Step 2: The specificity on the test partitions is",
    specificity(confMat)*100,fmt='%s %#.2f'))
    ```

    This produces the following output:

    ```
    [1] "Step 2: The sensitivity on the test partitions is 88.34"
    [1] "Step 2: The specificity on the test partitions is 67.84"
    ```

3) Repeat step 1. This time however, use the corresponding training data ***instead*** of the test data as the evaluation data, e.g. evaluate the model using **the same training partion to train and evaluate.** In this step we want to explore whether or not the models do better when tested on the same data that was used for training.

4) **Calculate** and **display** the sensitivity and specificity similarly to what you were expected to do in the take home exam. Please feel free to borrow the following print statements to get a nicely formatted output:

    ```
    print(sprintf("Step 4: The sensitivity on the training partitions is",
    sensitivity(confMat)*100,fmt='%s %#.2f'))
    print(sprintf("Step42: The specificity on the training partitions is",
    specificity(confMat)*100,fmt='%s %#.2f'))
    ```

    This produces the following output:

    ```
    [1] "Step 4: The sensitivity on the training partitions is 90.71"
    [1] "Step 4: The specificity on the training partitions is 71.42"
    ```

5) } # end of the for-loop that you started in step 0.

6) Compare the sensitivity and specificity values produced in step 2 with those in step 4 for the various cp values. What lesson can you draw from this comparison?

Turn in: Be sure to turn in your R code for steps 0 - 5 as well as the class accuracies produced in steps 2 and 4 for each cp value. Be sure that you address step 6 by analyzing and comparing the results from steps 2 and 4 for each cp value. Make sure that your R code for Part 2 is clearly documented to indicate that it is for Part 2.