CS313 E Term Project Bootstrapping

Madeline Boss (mrb5727) and Tariq Ali (taa2536)

1. Our project goals will be to create a sampling distribution of bootstrapped samples from a defined sample given by a user. Bootstrapping refers to the statistical process of creating synthetic samples from an original sample with replacement, which is then used to conduct statistical tests. If needed, more information can be found about bootstrapping here. This project will create n amount of samples that the user requests, and then be able to pull information about descriptive statistics surrounding the sampling distribution (mean, median, range, standard error), along with a confidence interval (of the user's choosing), and the ability to test another sample against the confidence interval (testing statistical significance).
2. We're not using any datasets.
3. There will be two non-main classes and one function determined outside the classes. There will also be one dictionary outside of any function or class.
    a. The first class will be Node, which will create nodes that have children (left child and right child) and contain numeric data. The numeric data will be the mean of a bootstrap sample and the standard deviation of a bootstrap sample.
    b. The second class will be a modified Binary Search Tree (BST) that contains the normal properties of a BST, along with the ability to conduct statistical tests (find confidence interval, test significance of another sample, etc). The BST will be sorted by mean values. Classes in this include a class that can add values to the BST, calculate mean, calculate median, calculate range, return a sorted list by mean, return a list of standard deviations (sorted by their mean values), find

standard error for the BST, and calculate confidence interval of BST. These classes will include a BST insert algorithm (add to BST function) and a BST walk-through algorithm (to get a sorted mean list).

c. The extra function will be a function that can create a bootstrap sample from an original sample. This will be done by a for loop that pulls random index values from the original sample, pulls the value from the list, and then calculates the mean and standard deviation and returns it to the user. These results will then be used to create Nodes to go into the BST.

d. The dictionary outside of any class/function will be a dictionary of z* values according to the Central Limit Theory. These will include the most notable Confidence Intervals such as 90%, 95%, 97.5%, etc, and their respective z* values.

e. Other notes: All of the user input will be through a mainline logic that will ask the user if they want to do certain tasks (which will then be used for testing the program). Depending on time constraints other features (such as a binary search algorithm to find the next best z* values, etc) may be implemented. However, they are not currently planned.

4. We will use the random library to make the samples, and the math library to use the sqrt function (necessary for the standard deviation function).
5. Note all these testing samples will have a predefined random seed used since bootstrap sampling is an inherently random process. One test will check if the bootstrap function outputs the correct values for mean and standard deviation. Another test will check if the BST can perform typical BST functions normally (adding a Node, returning a correctly

sorted list, finding mean, finding median, and finding range). One last test case will test if the statistical tests run as expected (confidence interval, if a value is in a confidence interval, etc). Notably this last test will check that the hand-calculated version of a confidence interval is relatively similar to the computer output (i.e compare the bootstrap method in Python to De Moivre's equation, which in theory should be relatively the same)

6. Currently, we expect it to work mostly fine with some limitations. First, we expect there to be time issues due to how high the number of bootstrap samples can be (10,000 is an average amount on statistical applications such as R), and plan to figure out the time run limit for our program. Secondly, we recognize we can only add so many z* values to our z* dictionary. We are currently unsure how to fix this issue, but we plan to either limit what confidence interval can be examined or apply a binary search algorithm to give the closest possible confidence interval z* value.