**685.621: Algorithms for Data Science**

**Rachael Kenney**

**Programming Assignment 1 Analysis**

**Due Date: July 1, 2019**

**Dated Turned In: July 1, 2019**

**Programming Assignment 1 Analysis**

For programming assignment 1, I chose to do option 1 from the assignment and implement the sorting algorithm on the iris data set. Because I was more familiar with the plotting packages associated with python, I wrote my assignment in python. This showed to cause more difficulty for me to write the actual sorting algorithm as I was not as familiar with the in depth dataframe manipulations.

Specifically, carrying the dataframe from function to function was difficult. Therefore, I was unable to break apart my sorting function into multiple methods as I had in the homework pseudocode.However, because of the ease of use for the visualization with matplotlib and the learning experience for dataframe manipulation, I would consider the trade-off for using python over java as being worth it.

I chose to create histograms for the visualization portion of this assignment. For the sorting algorithm, we needed to determine which features could split the classes most effectively. In order to determine which feature would be most beneficial to use, I determined that histograms of each feature color coded by each class would best show where those splits existed. From this, it was clear to see that the petal length and petal width were the only features that could split out the setosa class from the other two. However, none of the features could successfully split apart versicolor and virginica.

Because petal length was much closer to splitting apart versicolor and virginica, I chose to only do my sort algorithm on that feature. I had considered doing the sorting algorithm with a combination of petal length and petal width. However, the petal width spread showed a much larger overlap between versicolor and virginica. It was likely that by including petal width in the sorting algorithm, it would have not separated the classes as well. The performance time of the algorithm would also have been doubled, as it would have needed to be run for petal length and then for petal width.

When designing the sorting algorithm for homework 1, the group I was collaborating with chose to create our algorithm based off of sorting analysis done in problem 2 of that same assignment. In that problem, it was determined that a merge sort algorithm can reach Θ(nk + nlg(n/k)) worst case time if insertion sort is used when the recursive function size reaches sub-sized buckets of . This efficiency of speed is not entirely needed for this assignment, as the data set is only of size 150 and there is no variability of input. The input is always random for the petal length feature, but the data is sorted by class.