a. Hypothesis

I expect that insertion sort will be faster than quicksort for lists of up to 100 items large. I imagine that even though both can sort a list of size 100 very quick, quicksort will quickly become faster even a relatively medium sized list. Even though both can likely sort a list of size 1000 in less than a second, I think that quicksort can do it much faster with a list that size and the difference will be noticeable before that. When people refer to insertion sort being faster at “small” sizes of lists, I would think that a list of size 100 would be classified as small because they check efficiency of sizes up to millions.

b. Methods

I am using python to write the insertion and quicksort methods. I have used algorithms for each from sources found on the internet that are reliable and have time complexities equal to the said complexities in the project description. I originally checked times for lists up to size 500 integers but quickly realized that I only needed to check sizes of up to 50 items. In my code I repeated testing with values from 1 to 50 and ran insertion and quicksort on lists of each size. In order to get a better average for each list size I ran insertion sort and quicksort 100 different times on different lists but used random seeds so that both methods tested the same lists. After computing the average time for lists of size 1 to 50 I plotted the results. In order to find a good estimate of where quicksort is better, I learned 3rd Order polynomial regressor models on the quicksort and insertion sort times. I was then able to get a good line for the times of both methods.

c. Results

Below (figure 1) shows the results from my test plotted. The X axis shows the N values (size of the list being tested) and the y axis represents the average time taken in seconds. The insertion sort times are shown in orange and the quicksort times are shown in blue at each value of N. The learned functions from polynomial regression are shown in the red and purple dashed lines.

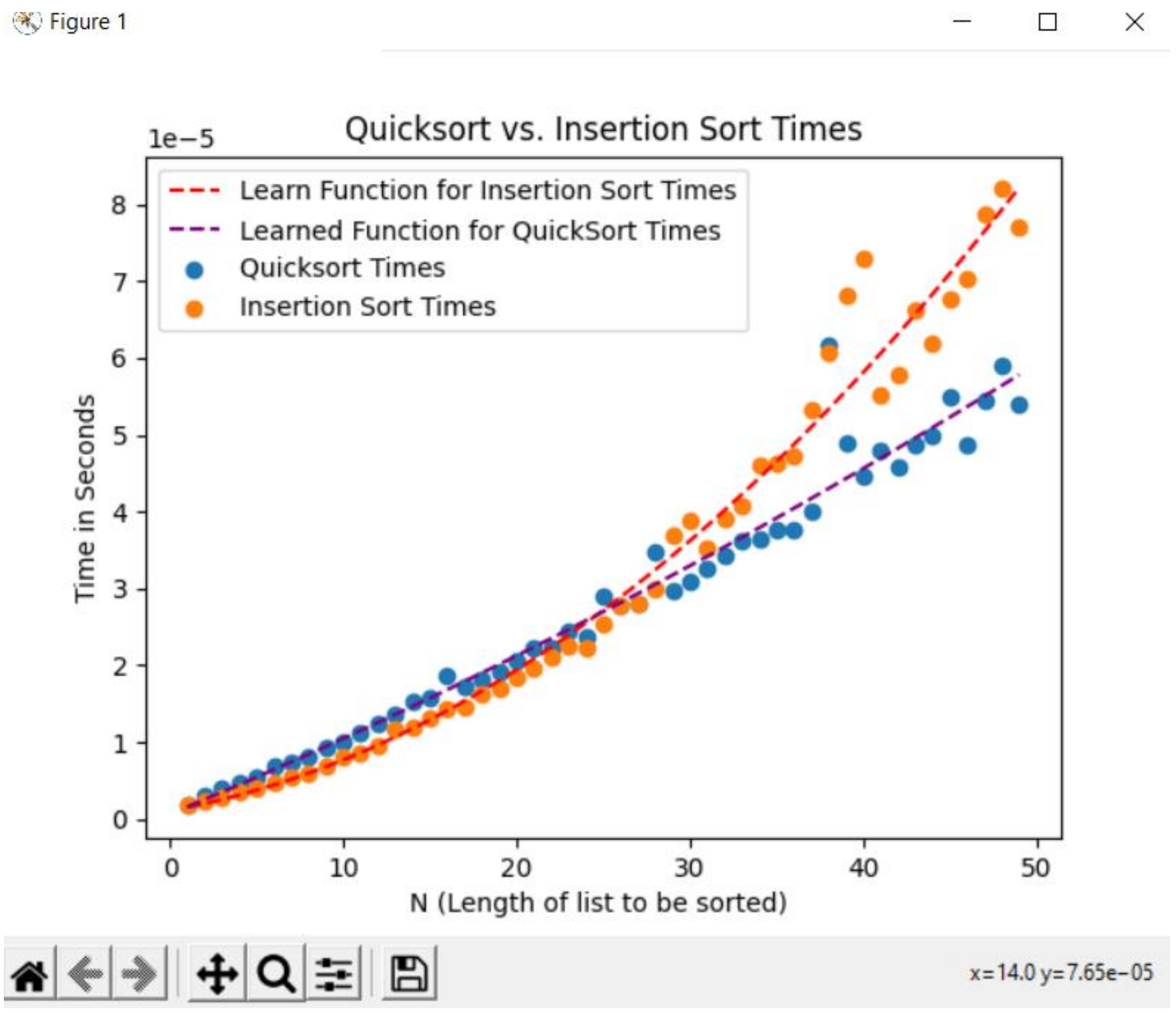

Figure 1

d. Discussion

I was surprised that quicksort was better at such a low value of N. I considered any N value below 50 to be small in testing and I thought that insertion sort would be better. I did find it challenging to find an accurate way to get the point where insertion sort is better. Eventually I chose the 3rd Order polynomial because it best reflects functions that have n^2 and n log(n) complexity. I also did not want to overfit the data using too high of an order when fitting. I also chose to use the crossing point of the two learned functions as the point at which quicksort is better because I felt they best represented the average times. Overall I would say my hypothesis was not correct but a good estimate of what the actual result was.

e. Conclusion

From this experiment I found that Insertion Sort is quicker for list/arrays of size < 25 and Quicksort is better for list/arrays of size > 25.