The data set I used for Homework 3 involved handwriting data, specifically, handwriting for individual numbers. There are sets of 8 (x, y) coordinates for every number written. They define the path the user took to draw the number. They are all normalized. Since there are 8 coordinate pairs, there are 16 columns of data. This would mean that a 16-dimensional graph would be needed to plot the data points (which I certainly cannot visualize). Of course, the number that the user intended to write is in the final column of the data. The link to the data set is below.

I first used Scikit Learn’s Logistic Regression Model (LR) to predict what the user wrote based on a training set. The training set was 70% of the data, while the test set was the remaining 30%. I found that the LR model yielded **93.3899%** accuracy (used .score() method). I then proceeded to adjust the parameters. The most useful changes were to increase the C value (larger values specify weaker regularization), to decrease the tol value (tolerance for stopping criteria), and to change the ‘solver’ method to ‘newton-cg’. It then yielded a **94.0570%** accuracy (used .score() method).

Next, I wrote my script for k Nearest Neighbors (KNN). I wrote it in a very brute-force manner, which meant it was slow (about 25 minutes to calculate predictions for over 3300 test samples on small processor). For each test sample, an array was created that contained the distance from that test sample to every training sample in the data set. Then, the top k closest were selected. The highest-occurring handwritten number among that closest set is the prediction. When I set k to 5, I obtained an accuracy of **99.0297%**. When I set k to 10, I obtained an accuracy of **98.4536%**. This may be because the data already clusters neatly for each number; therefore, there would be no need to go much further than just a few nearest neighbors. I was limited in the different k values I could use because of the poor speed of the algorithm.

The KNN algorithm greatly outperformed LR. This is likely because KNN is generally more suitable for datasets with more than two possibilities (a binary problem). Since I am predicting an output among 10 different possibilities, KNN seems like the more logical choice, intuitively.

**Dataset source:**

<https://ocw.mit.edu/courses/sloan-school-of-management/15-097-prediction-machine-learning-and-statistics-spring-2012/datasets/digits.csv>