Although the implementation of both algorithms, perception and naive bayes, provides the basic knowledge a computer needs to determine a given number or face, there seem to be multiple differences when computing. The perception algorithm trains the given training dataset for numbers with features of 4x4 or training dataset for faces with features of 5x5. The algorithm checks the characters in each feature and calculates the score by adding the weight of that feature multiplied by the number of characters versus the number of empty characters. If the prediction is wrong, the algorithm punishes the weight of the given algorithm by adding the number of characters versus the number of empty characters to the actual label weight and subtracting the number of characters versus the number of empty characters from the predicted label weight. This algorithm then iterates multiple times to obtain more accurate weights for testing. The algorithm later uses validation images and labels after training to determine the mean and standard deviation of which values predicted were correct. The naive Bayes algorithm, on the other hand, figures out each probability of a legal label given the image by multiplying the probability of getting the product at each feature, which holds the data using arrays that a specific feature value is at the given feature, by the probability of getting that legal label which was calculated by adding all the times the testing labels was a specific feature and divided it by the total number of testing labels. After finding the probabilities of getting each feature at each legal label to figure out the predicted number or face, the algorithm finds the greatest or max value of the probabilities of features at the legal labels. After each algorithm, These two algorithms go through different methods in order to determine the given datapoint of the training data, which can cause variation among the two. The algorithm later uses validation images and labels after the training session to determine the mean and standard deviation of which values predicted were correct.

One issue that results from the different types of algorithms is the execution time for a given number of data points that are trained. First, since face data only has two legal labels (0, 1), it seems to be faster overall than training for numbers that have legal labels (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). This is because both algorithms would have to train data five times more than face data. The percentage of dataset images being tested also has an influence on the time. As the percentage increases, the time drastically increases. With the already long training times of numbers that are 5 times longer than face training, when calculating 100 percent of the data, the time it takes is then multiplied by 10. When it comes to training through each algorithm, perception takes a greater amount of time than the naive Bayes algorithm since it is iterated multiple times for better accuracy. Since the naive Bayes algorithm would result in the same predicted values due to the same given probabilities, it does not have to be iterated over and over again. That said, the naive Bayes algorithm does not use weights, so the probabilities do not need to be adjusted.

There is a clear difference between the accuracies and standard deviations of both algorithms. Depending on whether we are testing for a face or a number, the face data is going to have a higher general accuracy since there is a 50% chance without training of getting it correct, while for a number, there is a 10% chance without training that it is correct. When training with 10%, the face accuracy results were closer to 50–60% for both algorithms, where Perceptron seemed to be more accurate. When training for numbers, there seems to be a more

accurate result of 60–65% accuracy for both algorithms, where perceptron seems prominent. As the percentage increases, the mean and accuracy both seem to increase rather than fluctuate, while Perceptron is around 10% more accurate than the Naive Bayes algorithm, while the standard deviation of Naive Bayes is greater than the standard deviation of Perceptron. However, there is an issue when the percentage reaches 100%. For faces, the naive Bayes algorithm caps at 72% accuracy, while the perceptron can fluctuate from 70% to 85% (from what I tested). For numbers, the Naive Bayes algorithm caps at 70.4%, while the perceptron can also fluctuate between 70% and 85%, creating a much better number detector. To summarize, the perceptron algorithm seems more prominent when calculating a more accurate number and face depiction than the naive Bayes algorithms due to its ability to train weights as many times as needed for stronger accuracy.

The algorithms that I implemented show the difference between determining images through weights and probabilities. The perceptron algorithm might have higher accuracy, but the training time is much longer than naive bayes. On the other hand, the naive Bayes algorithm will be less accurate but will have much faster training times than the perceptron algorithm. Both algorithms have their own set of qualities, which could help one determine which algorithm is good for training and testing.