CS 520: INTRO TO ARTIFICIAL INTELLIGENCE

PROJECT 2

FACE AND DIGIT CLASSIFICATION

December 08, 2022

Manasvini Nittala (Net ID: mn777)

Sahithi Reddy Sakinala (Net ID: ss4362)

Jahnavi Manchala (Net ID: jm2658)

1. NAIVE BAYES

(A) ALGORITHM DESCRIPTION

In Naive Bayes, the feature used was pixel occupancy per region. So first, the algorithm finds the maximum pixel occupancy across all features and all pictures for that training set. With that maximum, it sets up a matrix for each unique data point (faces/not faces, 1,2,3, etc.).

Each row corresponds to a feature, and each column is a number between 0 and count. Where count is nothing but the max value in that column. Next, the algorithm tallies and plots the feature and pixel data received from the training data in their corresponding matrix. So, for example, if the face picture had 9 pixels in region 1. The face matrix would add a +1 to row 1, column 9. Then at the end, it would divide each cell by the total number of pictures to create percentages for each feature and pixel combination.

Lastly, when you test, the algorithm will take the features from the testing image and look up the associated percent across all matrices. Each matrix will keep a score by multiplying each feature percent by each other as given by the testing image feature. Each matrix score is independent of the other. The prediction list will be chosen as the highest percentage of all the matrices.

(B) CHALLENGES FACED

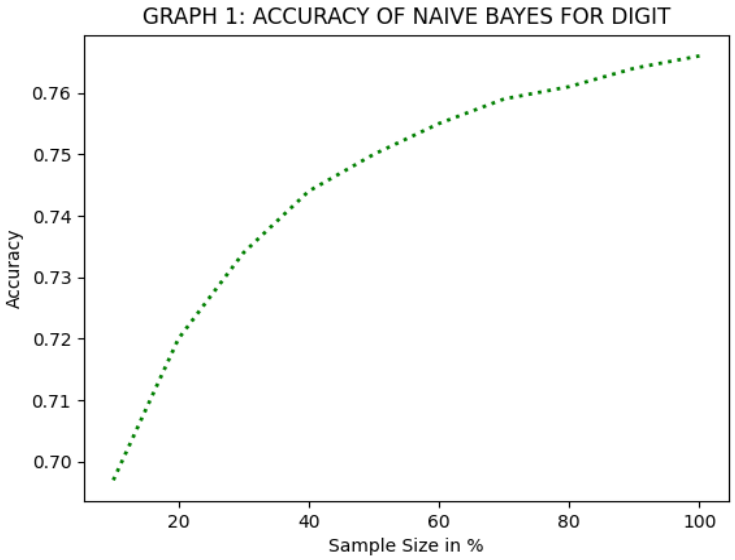
1. Feature Extraction: Finding the right set of features was a bit difficult.
We first took the binary values as pixels but were unable to achieve the desired accuracy, so we split the binary array into grids. We tested by splitting into 5 x 5, 10 x 10, and 20 x 20 grids. We felt that 10 x 10 grids were pretty comfortable to work with.

2. Unused Values: We first filled all our arrays like faces, not faces, and digits with 0s but later realized that they affected the final prediction array as we multiplied the probabilities. We solved it by replacing 0 with 0.01, small enough to be ignored but large enough to uphold the predictions.

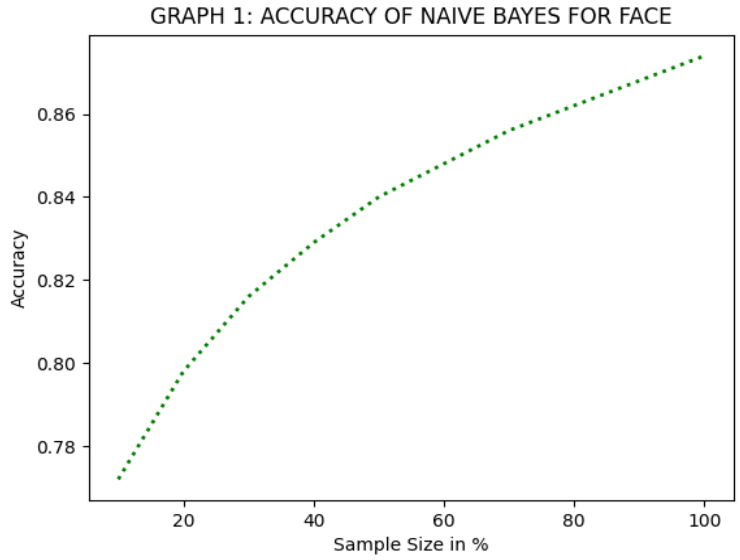
(C) TRAINING AND OBSERVATIONS

Observations for the Naive Bayes algorithm are based on its mean accuracy, standard deviation, prediction error, and time taken over 10% to 100% training samples.

i) Accuracy



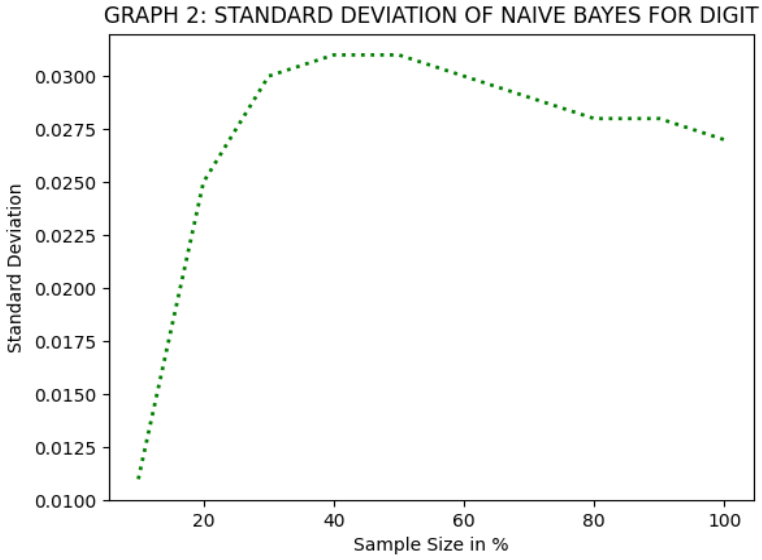
Graph 1: Mean Accuracy for Digit using Naive Bayes



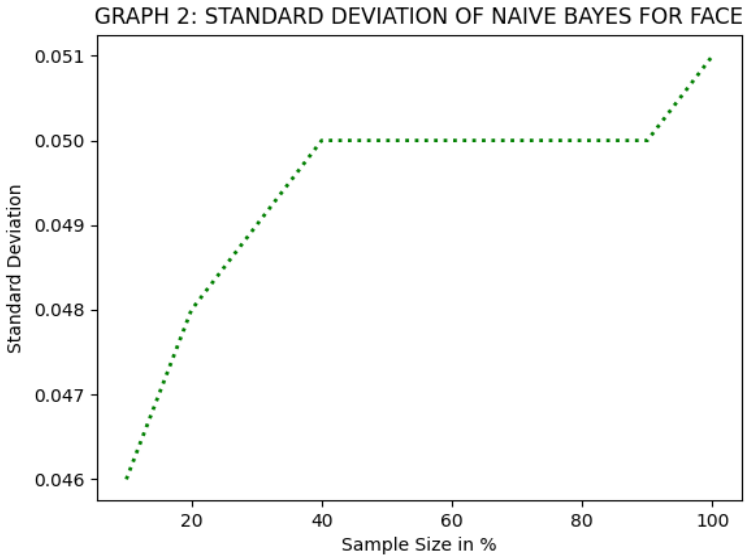
Graph 2: Mean Accuracy for Face using Naive Bayes

• As observed from the graphs, the Accuracy for both face and digit classification is increasing with an increase in the sample size. Maximum Accuracy for Digit was 77% and for Face was 88%

ii) Standard Deviation



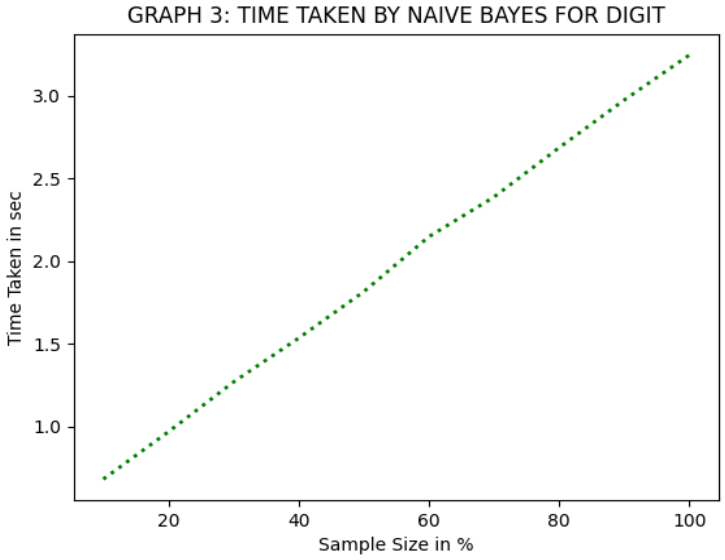
Graph 3: Standard Deviation for Digit using Naive Bayes



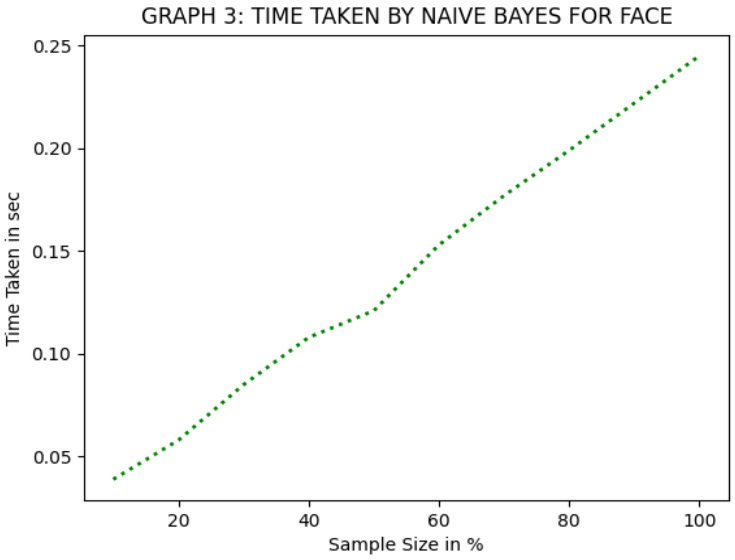
Graph 4: Standard Deviation for Face using Naive Bayes

• As observed from the graphs, the Standard Deviation for both face and digit classification is increasing up to approximately 40% samples but remains almost constant afterward.

iii) Time Taken



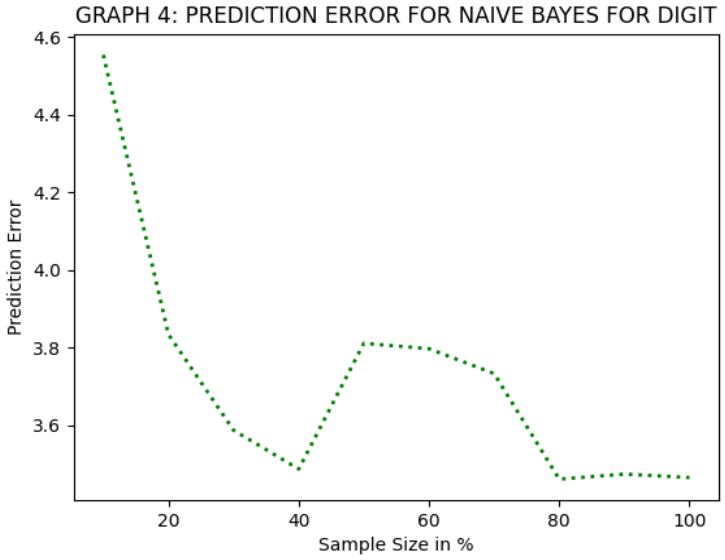
Graph 5: Time Taken for Digit using Naive Bayes



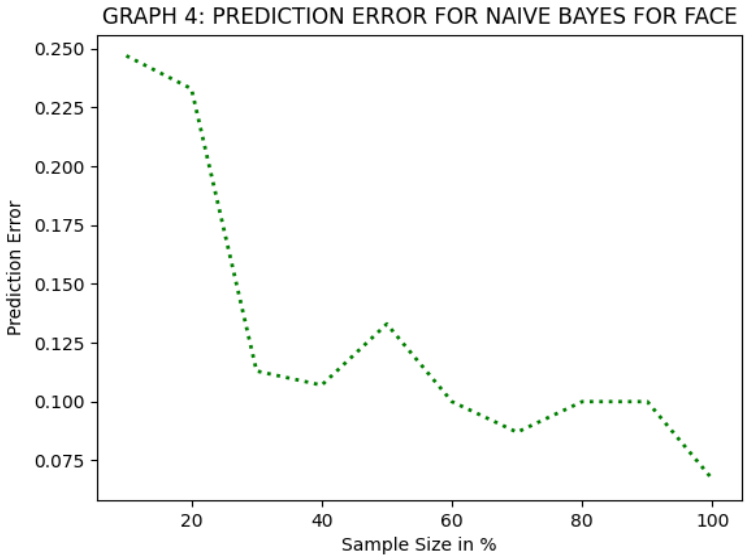
Graph 6: Time Taken for Face using Naive Bayes

• As observed from the graphs, the Time Taken for both face and digit classification has a linear relationship with the sample size. As the sample size increases, the time taken increases.

iv) Prediction Error



Graph 7: Prediction Error for Digit using Naive Bayes



Graph 8: Prediction Error for Face using Naive Bayes

• As observed from the graphs, the prediction error for Digit is the root mean squared error (RMSE) between the actual test labels and the predicted test labels. As the sample size increases, the prediction error decreases. The prediction error for Face is the ratio of wrong predictions to the total number of predictions. For Face, too the prediction error decreases as the sample size increases.

2. PERCEPTRON

(A) ALGORITHM DESCRIPTION

This algorithm begins with initializing random weights for each feature, which We defined as pixel occupancy per region. We split the image into a 10x10 grid, so We have 100 features. Each image has 100 weights. The weight vector is calculated by multiplying each weight by its corresponding feature pixel density. If this resulting number is favorable for faces, then the image predicts that it is a face.

If it is negative, then the prediction is not a face. To validate the correctness of the prediction, the algorithm checks the corresponding training label. If the prediction is correct, the algorithm moves on to the next image and calculates the weight vector again. If the prediction is too high (predicted face and it is not face), then each weight for that image is decremented by its corresponding feature. If the prediction is too low (predicted not face, and it is a face), then each weight is incremented by its corresponding feature. Our algorithm continues to loop through the training data and update the weights until it hits a threshold of 74% correct guesses for digits and 81% for faces.

The only difference between training faces and digits is that for each image, the algorithm calculates the weight vector for the digits 1-9 and chooses the highest one for its prediction. If the prediction is wrong, the weights for the wrong predicted digit are decreased by their respective features, and the weights for the correct digit are increased by their respective features. This process takes quite a long time due to the number of calculations that must be made at each iteration.

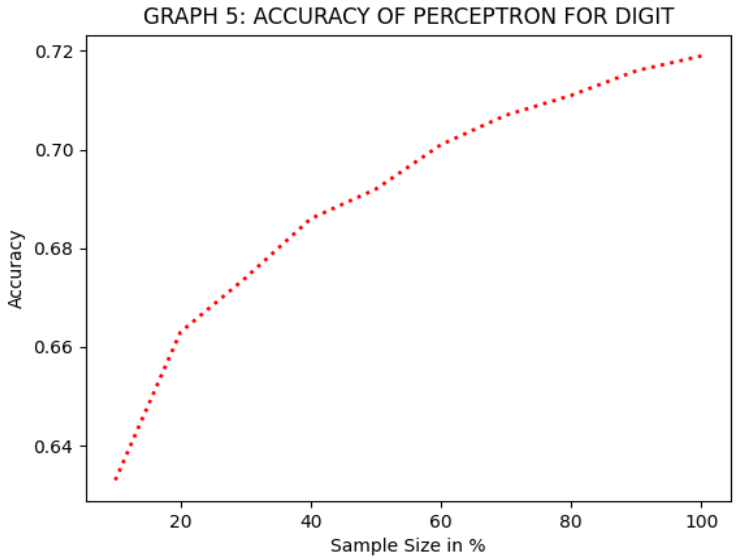
(B) CHALLENGES FACED

The main challenge in implementing perceptron is how much longer it took to train digits rather than faces since guessing a face is a binary decision, while digits are not. Also, calculating each f(x) weight vector for the digits took longer than calculating just one f(x) weight vector for faces. This takes up more run time as well as memory.

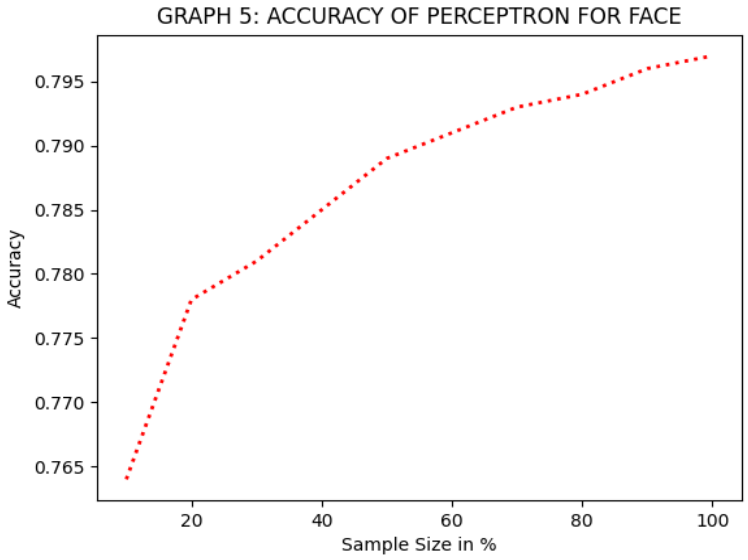
(C) TRAINING AND OBSERVATIONS

Observations for the Naive Bayes algorithm are based on its mean accuracy, standard deviation, prediction error, and time taken over 10% to 100% training samples.

i) Accuracy



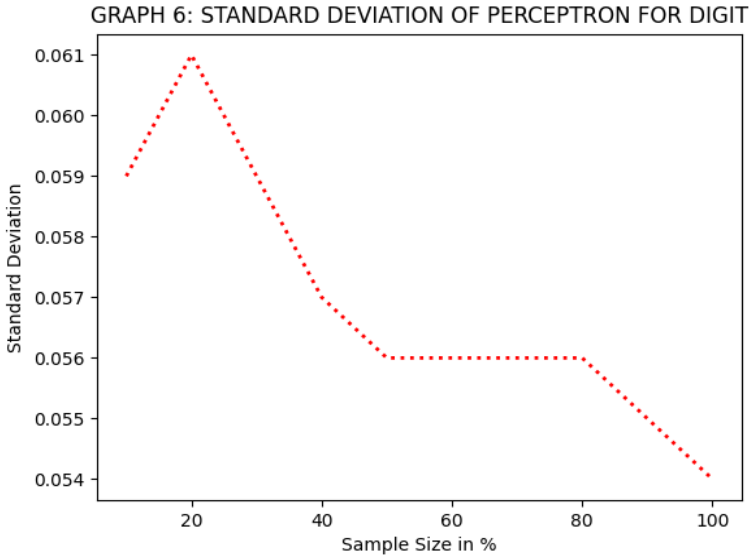
Graph 9: Mean Accuracy for Digit using Perceptron



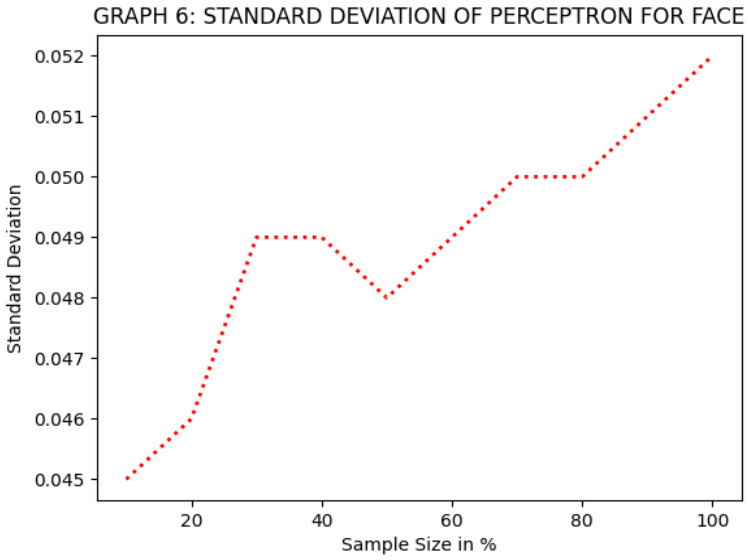
Graph 10: Mean Accuracy for Face using Perceptron

• As observed from the graphs, the Accuracy for both face and digit classification is increasing with an increase in the sample size. Maximum Accuracy for Digit was 72% and for Face was 81%

ii) Standard Deviation



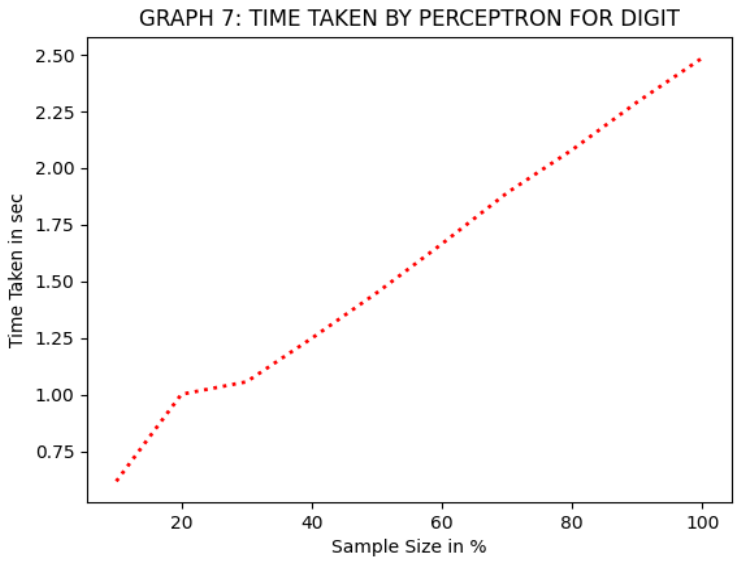
Graph 11: Standard Deviation for Digit using Perceptron



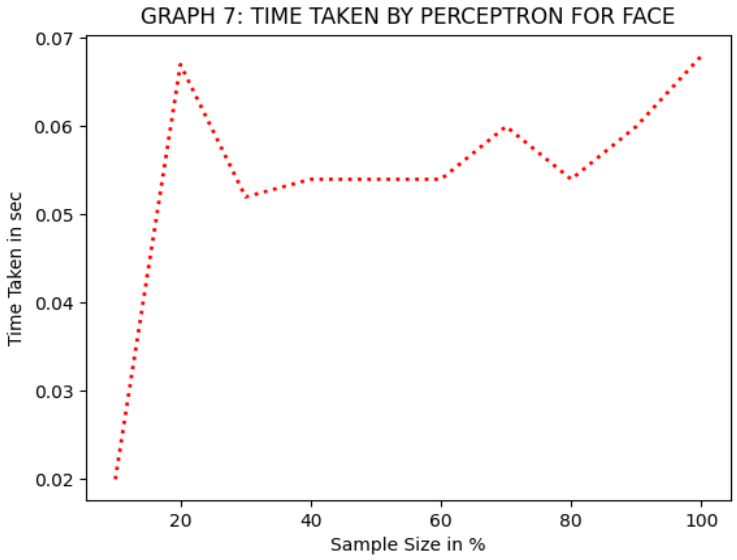
Graph 12: Standard Deviation for Face using Perceptron

• As observed from the graphs, the Standard Deviation for Digit is seeing a decreasing trend as the sample size increases whereas Face is seeing an increasing trend but we should make an observation that these standard deviation values are very very close and can be approximated to one value.

iii) Time Taken



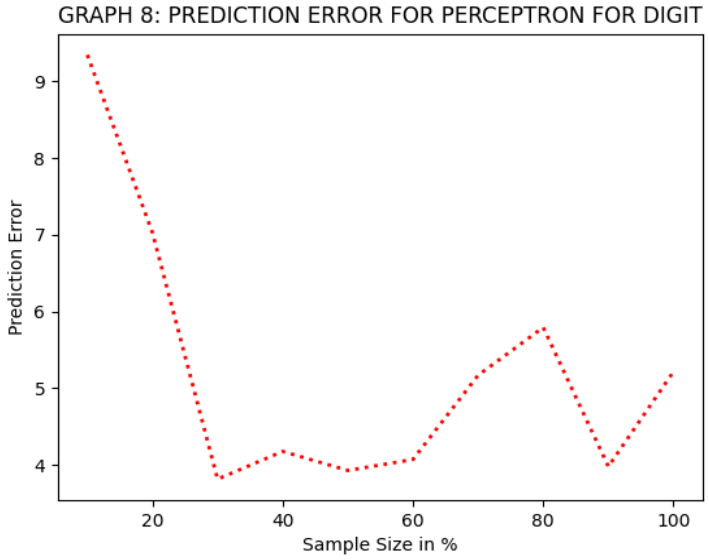
Graph 13: Time Taken for Digit using Perceptron



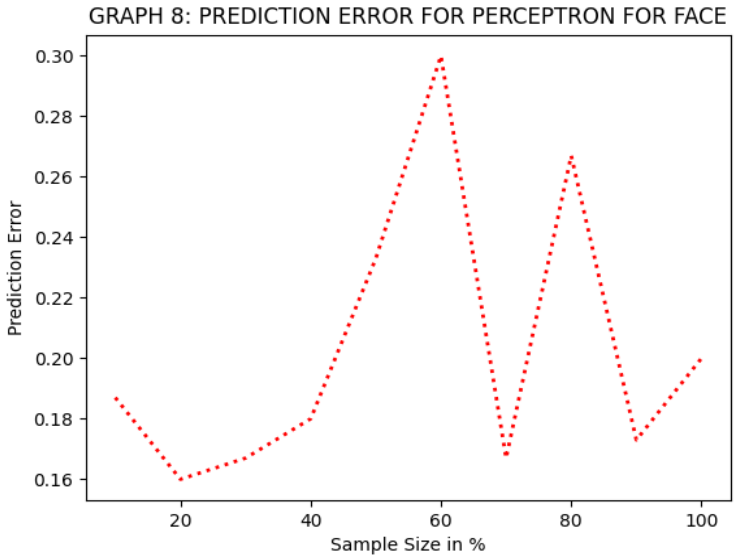
Graph 14: Time Taken for Face using Perceptron

• As observed from the graphs, the Time Taken for both face and digit classification has an almost linear relationship with the sample size. As the sample size increases, the time taken increases.

iv) Prediction Error



Graph 15: Prediction Error for Digit using Perceptron



Graph 16: Prediction Error for Face using Perceptron

• As observed from the graphs, the prediction error for Digit is the root mean squared error (RMSE) between the actual test labels and the predicted test labels. As the sample size increases, the prediction error decreases. The prediction error for Face is the ratio of wrong predictions to the total number of predictions. For Face, many fluctuations are observed, but when trained with 100% training data, the prediction error is the minimum.

3. k NEAREST NEIGHBORS

(A) ALGORITHM DESCRIPTION

This algorithm works by taking the euclidean distance of the features in the testing image against all the features of the images in the training data. So, for example, image 1 of the testing data would be compared against all 5000 images in the training data by finding the euclidean distance for each pairing. Then all that data is saved into an array and sorted from lowest to highest. After that, the first three cells are chosen to vote. The most frequently seen within those three cells is chosen for the predictions. This process repeats for each image in the training set.

(B) CHALLENGES FACED

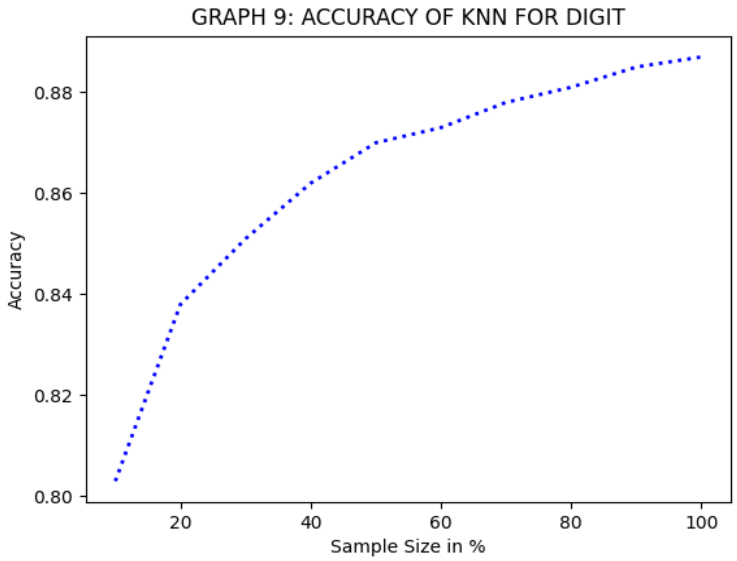
1. Run Time: The main challenge we faced while implementing kNN was the time it took to return predictions. This problem arose with Digit data as there need to be 5000 comparisons made, and we are also iterating it to get the mean values. We have tried other methods instead but found the comparison to be the best way. We have

reduced the run time by decreasing the number of iterations.

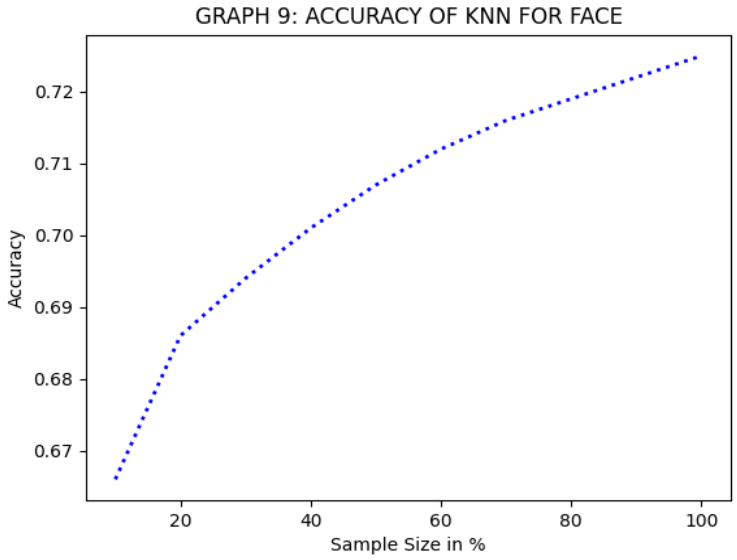
(C) TRAINING AND OBSERVATIONS

Observations for the Naive Bayes algorithm are based on its mean accuracy, standard deviation, prediction error, and time taken over 10% to 100% training samples.

i) Accuracy



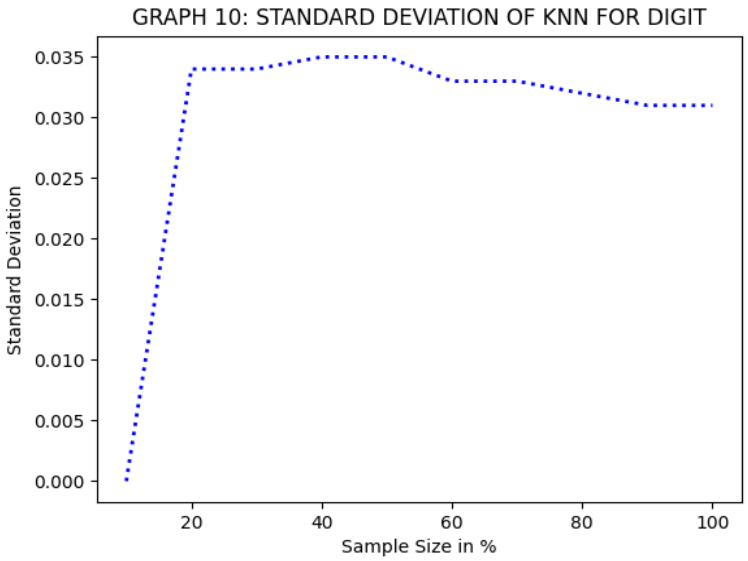
Graph 17: Mean Accuracy for Digit using kNN



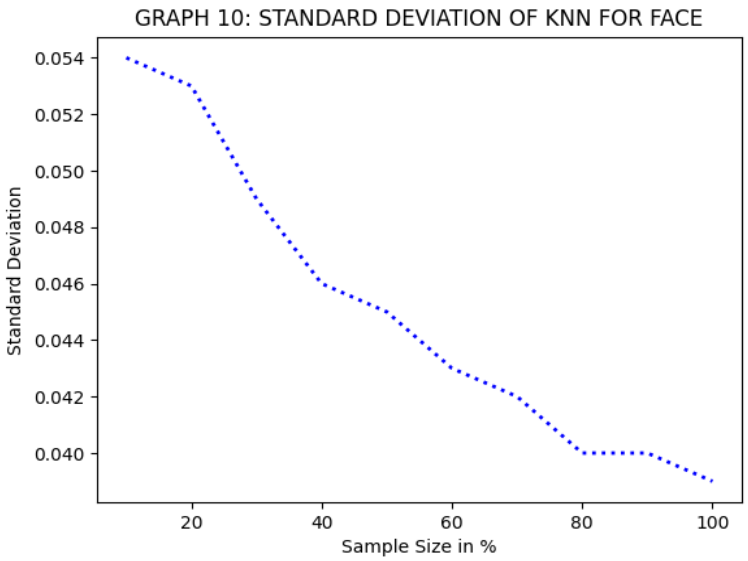
Graph 18: Mean Accuracy for Face using kNN

• As observed from the graphs, the Accuracy for both face and digit classification is increasing with an increase in the sample size. Maximum Accuracy for Digit was 88% and for Face was 74%

ii) Standard Deviation



Graph 19: Standard Deviation for Digit using kNN

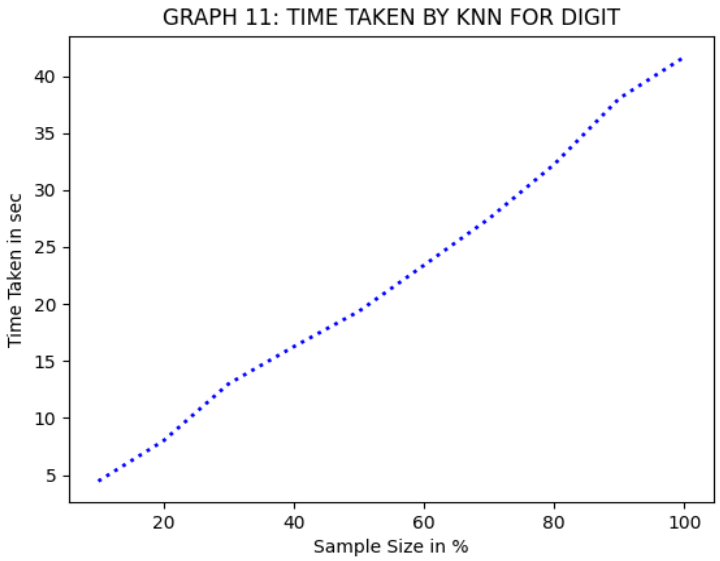


Graph 20: Standard Deviation for Face using kNN

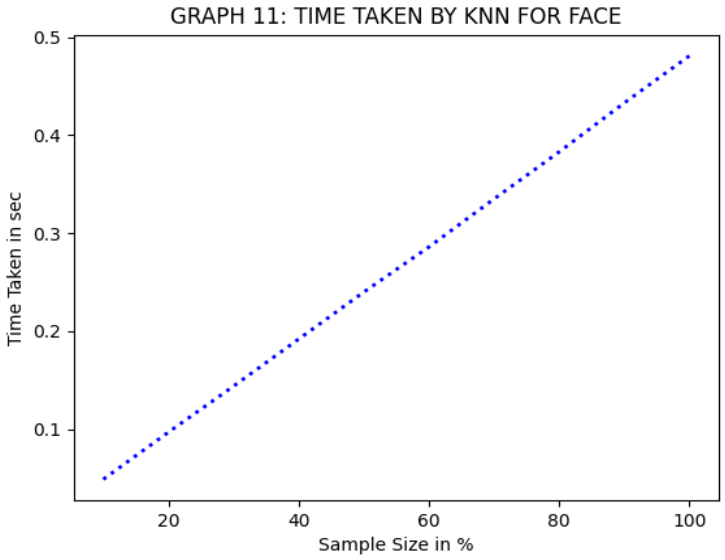
• As observed from the graphs, the Standard Deviation for digit classification increases up to approximately 20% samples but remains almost constant afterward,

but for Face data, there is a decreasing trend observed, whereas the sample size increases the standard deviation is decreasing.

iii) Time Taken



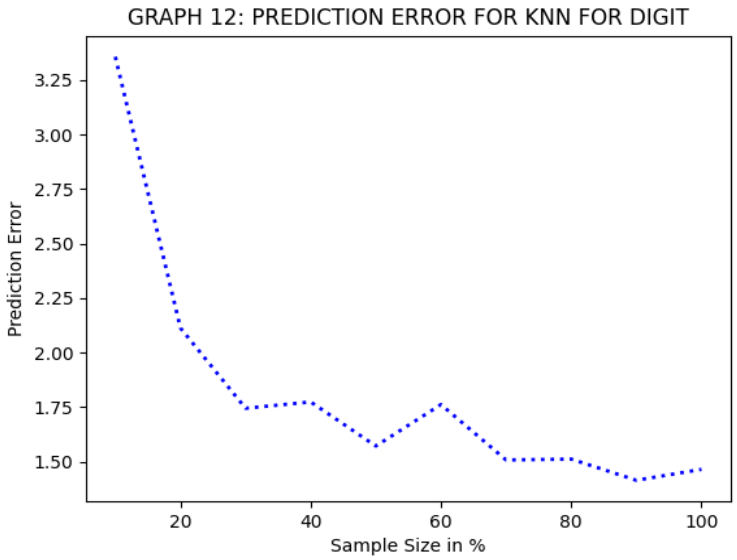
Graph 21: Time Taken for Digit using kNN



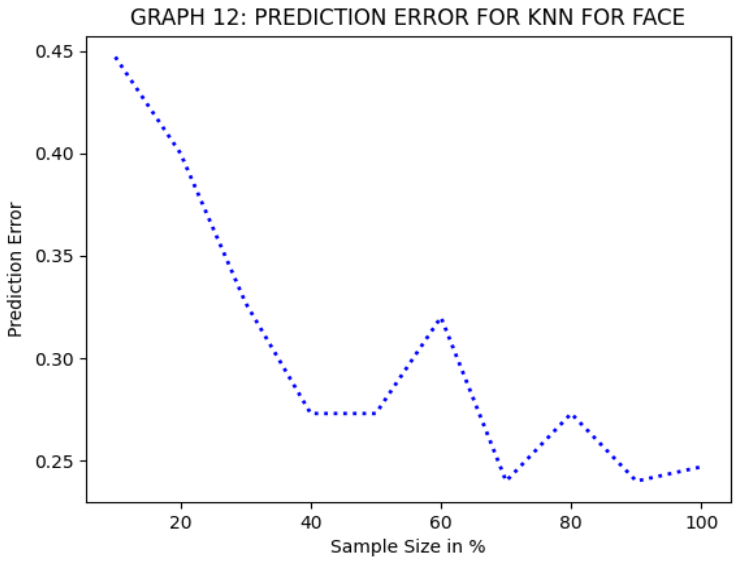
Graph 22: Time Taken for Face using kNN

• As observed from the graphs, the Time Taken for both face and digit classification has a linear relationship with the sample size. As the sample size increases, the time taken increases.

iv) Prediction Error



Graph 23: Prediction Error for Digit using kNN



Graph 24: Prediction Error for Face using kNN

• As observed from the graphs, the prediction error for Digit is the root mean squared error (RMSE) between the actual test labels and the predicted test labels. As the sample size increases, the prediction error decreases. The prediction error for Face is the ratio of wrong predictions to the total number of predictions. For Face, too, the prediction error decreases as the sample size increases.

```
MEAN ACCURACY FOR NAIVE BAYES: [0.768, 0.793, 0.811, 0.825, 0.836, 0.846, 0.854, 0.86, 0.866, 0.873]
MEAN STANDARD DEVIATION FOR NAIVE BAYES [0.06, 0.055, 0.055, 0.055, 0.055, 0.054, 0.054, 0.054, 0.054, 0.055]
TIME TAKEN FOR NAIVE BAYES TRAINING AND TESTING: [0.037, 0.057, 0.073, 0.097, 0.122, 0.208, 0.154, 0.182, 0.201, 0.224]
PREDICTION ERROR FOR NAIVE BAYES FACE: [0.353, 0.14, 0.18, 0.167, 0.113, 0.113, 0.1, 0.08, 0.113, 0.067]
--------------------------------------------------------------------------------

MEAN ACCURACY FOR PERCEPTRON: [0.769, 0.778, 0.781, 0.783, 0.785, 0.786, 0.789, 0.791, 0.793, 0.796]
MEAN STANDARD DEVIATION FOR PERCEPTRON [0.042, 0.047, 0.05, 0.052, 0.052, 0.053, 0.052, 0.053, 0.053, 0.053]
TIME TAKEN FOR PERCEPTRON TRAINING AND TESTING: [0.02, 0.037, 0.032, 0.048, 0.052, 0.049, 0.056, 0.049, 0.062, 0.061]
PREDICTION ERROR FOR PERCEPTRON FACE: [0.247, 0.173, 0.14, 0.147, 0.207, 0.153, 0.193, 0.18, 0.227, 0.227]
--------------------------------------------------------------------------------
MEAN ACCURACY FOR KNN: [0.671, 0.687, 0.695, 0.702, 0.708, 0.713, 0.717, 0.721, 0.724, 0.727]
MEAN STANDARD DEVIATION FOR KNN [0.054, 0.049, 0.046, 0.044, 0.043, 0.042, 0.041, 0.04, 0.039, 0.038]
TIME TAKEN FOR KNN TRAINING AND TESTING: [0.085, 0.093, 0.145, 0.188, 0.232, 0.278, 0.322, 0.368, 0.418, 0.471]
PREDICTION ERROR FOR KNN FACE: [0.253, 0.3, 0.287, 0.28, 0.3, 0.267, 0.253, 0.253, 0.247, 0.247]
--------------------------------------------------------------------------------
```

Figure 1: Output Values for FACE

```
MEAN ACCURACY FOR DIGIT NAIVE BAYES: [0.685, 0.714, 0.73, 0.74, 0.747, 0.753, 0.757, 0.76, 0.762, 0.765]
MEAN STANDARD DEVIATION FOR DIGIT NAIVE BAYES [0.016, 0.032, 0.035, 0.035, 0.035, 0.034, 0.033, 0.032, 0.031, 0.03]
TIME TAKEN FOR NAIVE BAYES DIGIT TRAINING AND TESTING: [0.649, 0.916, 1.186, 1.455, 1.726, 1.995, 2.263, 2.512, 3.182, 3.163]
PREDICTION ERROR FOR NAIVE BAYES FOR DIGIT : [4.371, 3.415, 3.993, 3.758, 3.738, 3.675, 3.536, 3.7, 3.491, 3.465]
--------------------------------------------------------------------------------
MEAN ACCURACY FOR DIGIT PERCEPTRON: [0.621, 0.642, 0.667, 0.677, 0.685, 0.695, 0.702, 0.704, 0.709, 0.712]
MEAN STANDARD DEVIATION FOR DIGIT PERCEPTRON [0.054, 0.048, 0.06, 0.059, 0.059, 0.06, 0.059, 0.058, 0.057, 0.057]
TIME TAKEN FOR PERCEPTRON DIGIT TRAINING AND TESTING: [0.573, 0.769, 0.959, 1.15, 1.346, 1.53, 1.733, 1.924, 2.096, 2.283]
PREDICTION ERROR FOR PERCEPTRON FOR DIGIT : [7.113, 5.876, 3.887, 3.811, 4.174, 3.674, 3.646, 3.55, 3.808, 3.942]
--------------------------------------------------------------------------------
MEAN ACCURACY FOR KNN FOR DIGIT: [0.812, 0.837, 0.85, 0.857, 0.867, 0.872, 0.877, 0.882, 0.885, 0.887]
MEAN STANDARD DEVIATION FOR KNN FOR DIGIT [0.0, 0.025, 0.027, 0.027, 0.031, 0.03, 0.031, 0.031, 0.031, 0.03]
TIME TAKEN FOR KNN FOR DIGIT TRAINING AND TESTING: [3.816, 7.823, 10.937, 14.849, 18.994, 22.893, 26.801, 31.278, 34.24, 40.026]
PREDICTION ERROR FOR KNN FOR DIGIT [2.581, 2.449, 1.948, 1.811, 1.494, 1.685, 1.569, 1.39, 1.576, 1.44]
----------------------------------------
```

Figure 2: Output Values for DIGIT