Project: Face and Digit Classification

Submitted By: Brandon Goldstein (bbg17), Taqiya Ehsan (te137), Swapnil Napuri (spn41)
Date: 3 May 2023

1 Loading Dataset and Feature Extraction

We used the code and methods provided in the project package on the UC Berkeley website to load the dataset. We wrote a program for extracting the features from each image in dataset.

2 Classifiers

2.1 Naive Bayes

Features:

The set of features used for this algorithm are pixel level. Each pixel is independent and could be either 0 (no edge/blank for faces), 1 (gray pixel for faces and + for digits), or 2 (edge or black pixel for faces and # for digits). We followed the documentation and code provided in the UC Berkeley project description. Specifically, we implemented a basic feature extractor, which was suggested by UC Berkeley. For digits, it returns a set of pixel features indicating whether each pixel in the provided datum is white (0) or gray/black (1). For faces, it returns a set of pixel features indicating whether each pixel in the provided datum is an edge (1) or no edge (0).

Algorithm:

To calculate the prior probabilities of each unique data label (Y), we implement the following equation:

$$P(Y = y) = \frac{numberofdatawithlabel(Y=y)intrainingset}{totalnumberoftrainingdata}$$

We store the prior probabilities in a dictionary with the data label Y as the key. We then compute the conditional probability for each feature (f) given label (y):

$$P(F_i = f_i | Y = y) = \frac{c(f_i + k)}{\sum_{f'_i \in (0,1)} c(f'_i, y) + k}$$

Classification:

We use log probability for classifying based on probability as the numbers will be very small. The equation we use is as follows: $logP(y) + \sum_{i=1}^{m} logP(f_i|y)$

Training and Testing Analysis:

We train our model for both datasets with randomly chosen subsets of 10%, 20%, ..., 100% data and record the prediction accuracy after 10 training iterations for each subset and testing on 100 test images.

Face Data:
We can see from these graphs that the prediction accuracy progressively increases as we keep increasing the size of training data from 10% to 100%. Simultaneously, the standard deviation goes down to 0 when 100% of the training data is used. The highest prediction accuracy is 89% when all of the training data is used.

As we would anticipate, the mean training time increases as we increase the size of the training subset.

Digit Data:
As with the previous dataset, we can see that the prediction accuracy progressively increases as we keep increasing the size of training data from 10% to 100%. Simultaneously, the standard deviation goes down to 0 when 100%

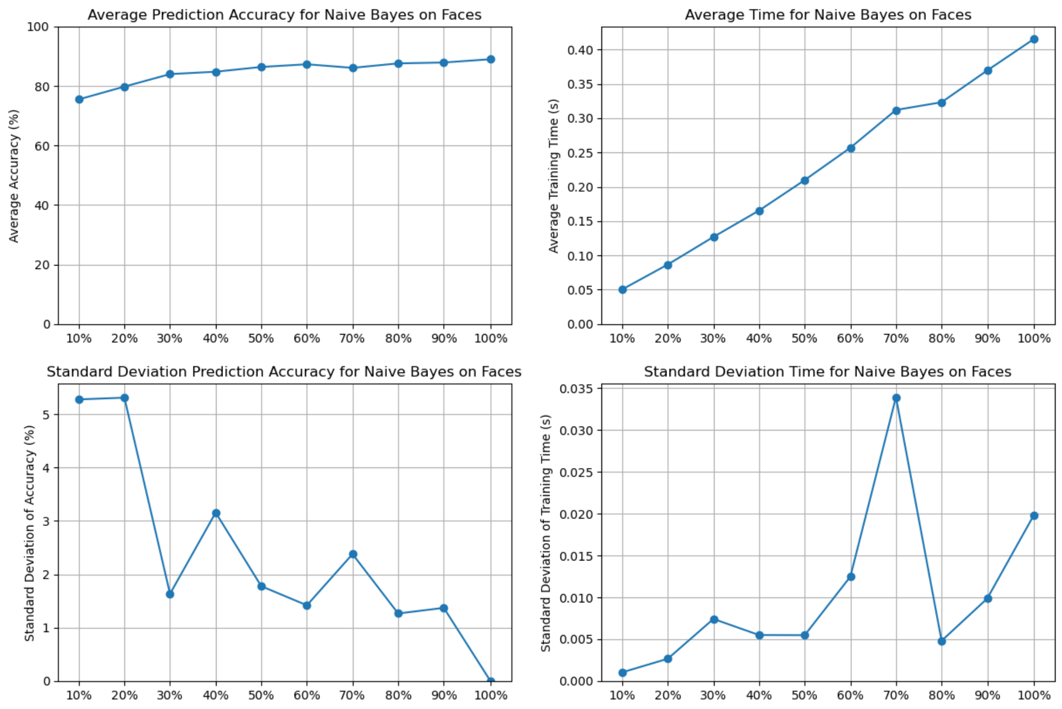

Figure 1: Accuracy and Time for Naive Bayes on Faces

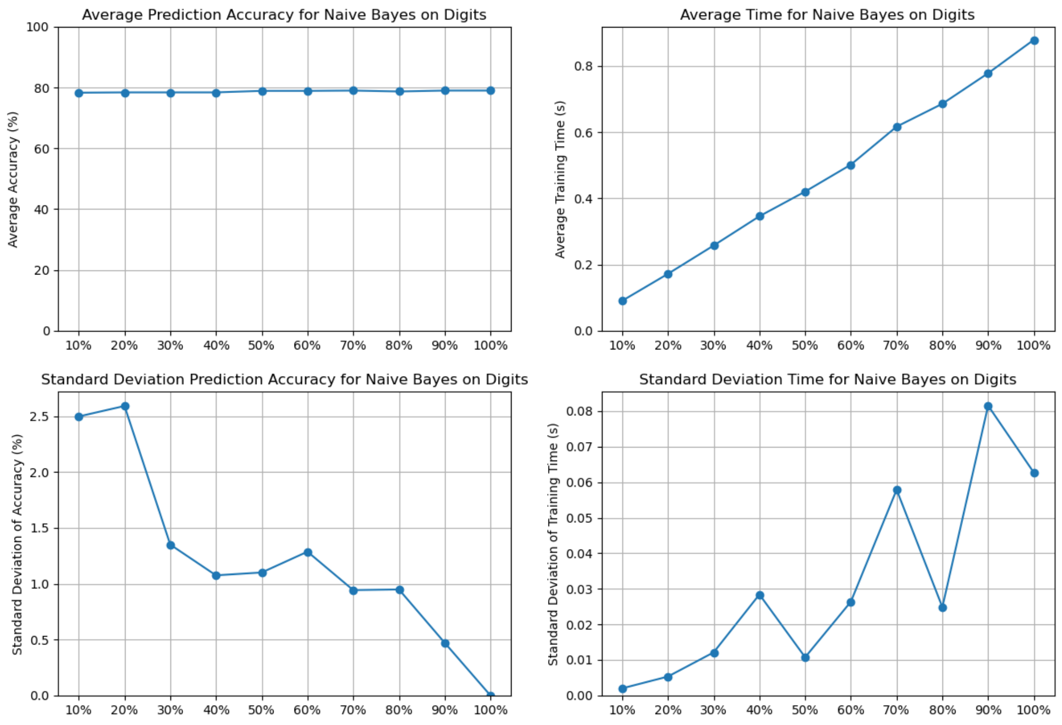

Figure 2: Accuracy and Time for Naive Bayes on Digits

of the training data is used. The highest prediction accuracy is 79% when all of the training data is used.

As before, the mean training time increases as we increase the size of the training subset.

Overall, the Naive Bayes classifier seems to work better on the faces dataset than the digits dataset in terms of prediction accuracy. Our hypothesis for this difference is that for faces it's simply a binary classification – face or not face – whereas for digits it is a multiclass classifier. This makes classification slightly more complicated on the digits dataset, which might explain the lower average prediction accuracy. From the timing point of view, it took less time to train the faces dataset compared to digits because of the size difference between the two – we have more data points in the digits training set than the faces training set.

2.2 Perceptron

Features:

The set of features used for this algorithm are pixel level. Each pixel is independent and could be either 0 (no edge/blank for faces), 1 (gray pixel for faces and + for digits), or 2 (edge or black pixel for faces and # for digits). We followed the documentation and code provided in the UC Berkeley project description. Specifically, we implemented a basic feature extractor, which was suggested by UC Berkeley. For digits, it returns a set of pixel features indicating whether each pixel in the provided datum is white (0) or gray/black (1). For faces, it returns a set of pixel features indicating whether each pixel in the provided datum is an edge (1) or no edge (0).

Algorithm:

We make a guess as to what digit it is or whether it is a face or not. If that guess does not match what it actually is, we adjust the weights array accordingly (as described in the UC Berkeley documentation and the class slides).

Training and Testing Analysis:

We train our model for both datasets with randomly chosen subsets of 10%, 20%, ..., 100% data and record the prediction accuracy after 10 training iterations for each subset.

Face Data:

We can see from these graphs that the prediction accuracy progressively increases as we increase the size of training dataset from 10% to 100%. Simultaneously, the standard deviation trends downward as more training data is used. The highest prediction accuracy is 84.2% when all of the training data is used.

As we would anticipate, the mean training time increases as we increase the size of the training subset.

Digit Data:

We can see from these graphs that the prediction accuracy progressively increases as we increase the size of training dataset from 10% to 100%. Simultaneously, the standard deviation trends downward as more training data is used. The highest prediction accuracy is 81.2% when 90% of the training data is used. This is most likely due to the randomness of the data selection process, but the important idea is that it trends upward.

As before, the mean training time increases as we increase the size of the training subset.

Standard Deviation Issue:

As evident in the graphs, the standard deviation for the average prediction accuracy did not go to 0 even when the size of the face and digit training datasets were 100%. We believe this is a result of the max iterations variable. UC Berkeley defaults the variable to 2, and the plots for this classifier were made with the default in-place. Since the data selected for the training set is picked at random, the order of the observations within the training dataset is also random for every Perceptron run. Accordingly, the order that the weights are changed and compared is not uniform across trials. This is what leads to slightly different prediction accuracies for the trials conducted on the largest training data sets. We theorize that we could realize uniform prediction accuracies if we allowed

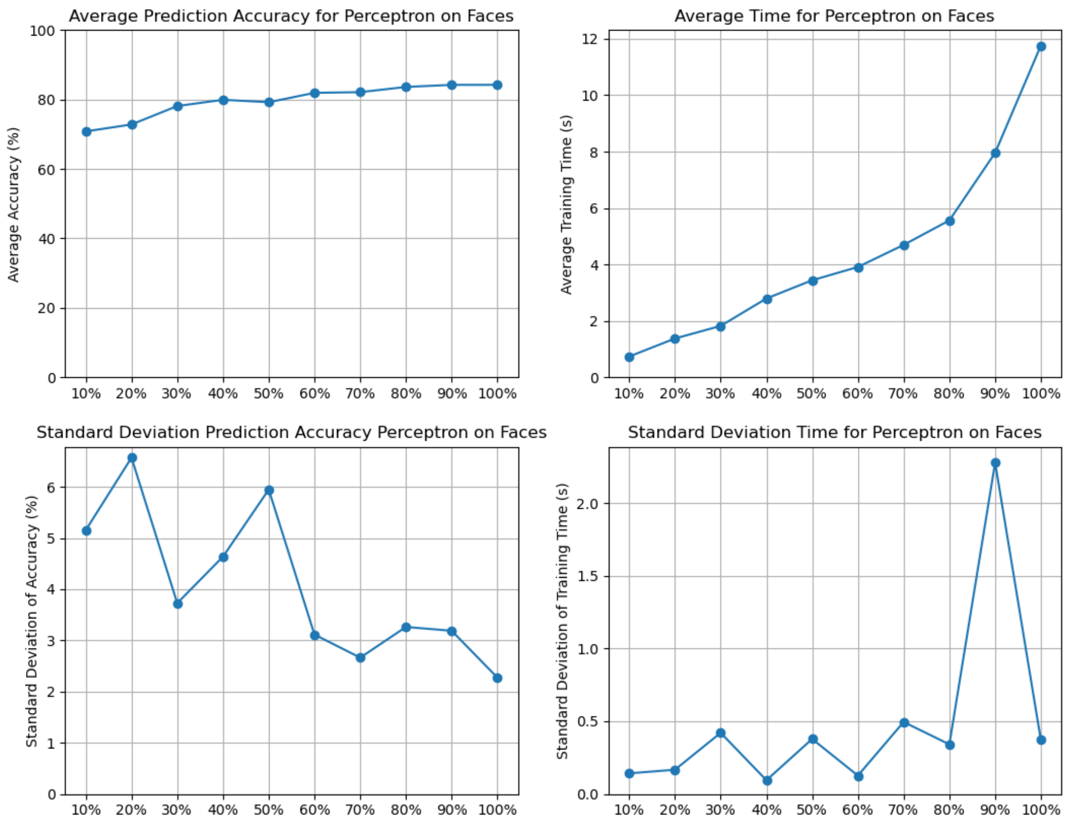

Figure 3: Accuracy and time for Perceptron on Faces

the Perceptron to converge, rather than capping the number of iterations. However, a paper written by Haim Sompolinsky of MIT, titled "Introduction: The Perceptron", suggests that the number of iterations required for convergence is likely large and at least the size of the full training datasets. The computational intensity of this number of iterations motivated us to use the default max iterations variable provided by UC Berkeley.

2.3 Logistic Regression

Features:

The set of features used for this algorithm are pixel level. Each pixel is independent and could be either 0 (no edge/blank for faces), 1 (gray pixel for faces and + for digits), or 2 (edge or black pixel for faces and # for digits). We followed the documentation and code provided in the UC Berkeley project description. Specifically, we implemented a basic feature extractor, which was suggested by UC Berkeley. For digits, it returns a set of pixel features indicating whether each pixel in the provided datum is white (0) or gray/black (1). For faces, it returns a set of pixel features indicating whether each pixel in the provided datum is an edge (1) or no edge (0).

Algorithm:

We used the Logistic Regression Classifier class from the scikit-learn library, as allowed in the project description. According to the documentation, the Logistic Regression Classifier class "implements regularized logistic regression using" one of a set of solvers. We selected the "newton-cg" solver, which applies a modified, linear conjugate gradient to solve the Newton Equations and define the model. This solver was selected over others because it converged in both the binary (used for the Face analysis) and multi-class (used for the digit analysis) cases. The model itself estimates the log-odds of an entity belonging to a certain class by framing the log-odds as a linear combination of the selected features.

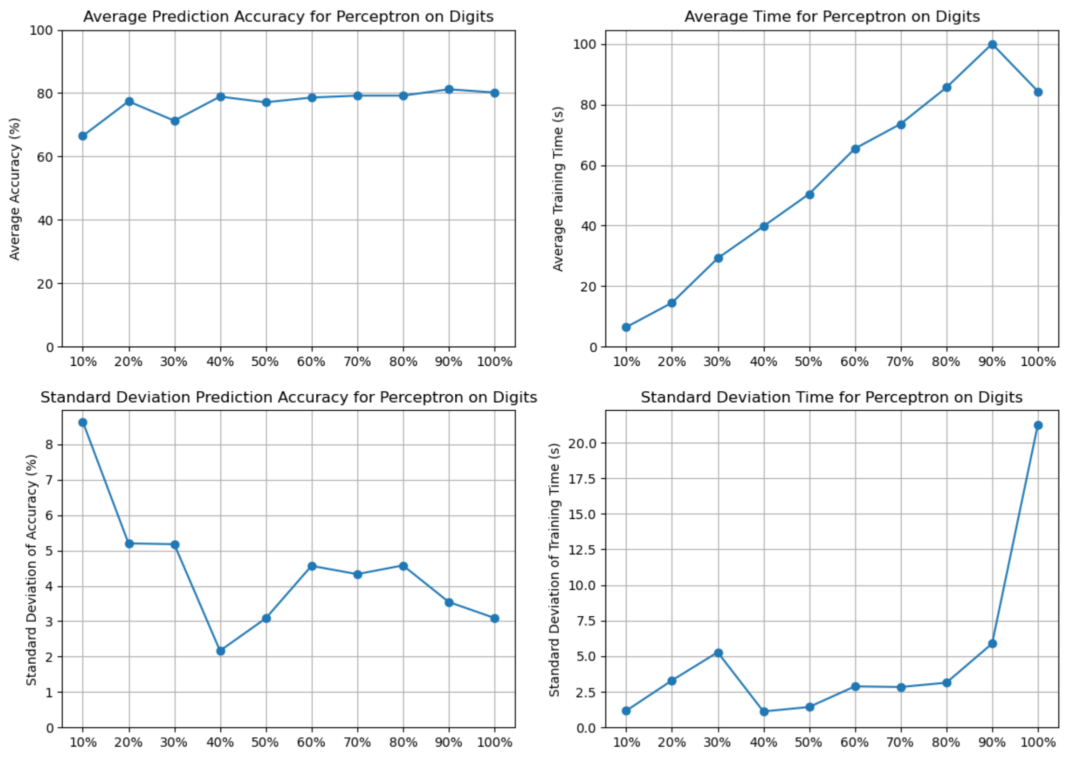

Figure 4: Accuracy and time for Perceptron on Digits

Classification:

As alluded to above, the Logistic Regression Classifier estimates the probability of an entity belonging to a certain class and ascribes a class after viewing the probabilities across the class space.

Training and Testing Analysis:

We train our model for both datasets with randomly chosen subsets of 10%, 20%, ..., 100% data and record the prediction accuracy after 10 training iterations for each subset and testing on 100 test images.

Face Data:
We can see that the prediction accuracy progressively increases as we increase the size of training dataset from 10% to 100%. Simultaneously, the standard deviation goes down to 0 when 100% of the training data is used. The highest prediction accuracy is 92% when 40% or 50% of the training data is used, though the average prediction accuracy when 40% of the training data is used is roughly 86% and the average prediction accuracy when 50% of the training data is used is approximately 89%. When 100% of the training data is used, the average prediction accuracy is 89%.

As we would anticipate, the mean training time increases as we increase the size of the training subset.

Digit Data:
As with the previous dataset, we can see from these graphs that the average prediction accuracy progressively increases as we increase the size of training dataset from 10% to 100%. Simultaneously, the standard deviation goes down to 0 when 100% of the training data is used. The highest prediction accuracy is 93% when 70% of the training data is used, though the average prediction accuracy when 70% of the training data is used is roughly 90%. When 100% of the training data is used, the average prediction accuracy is 92%

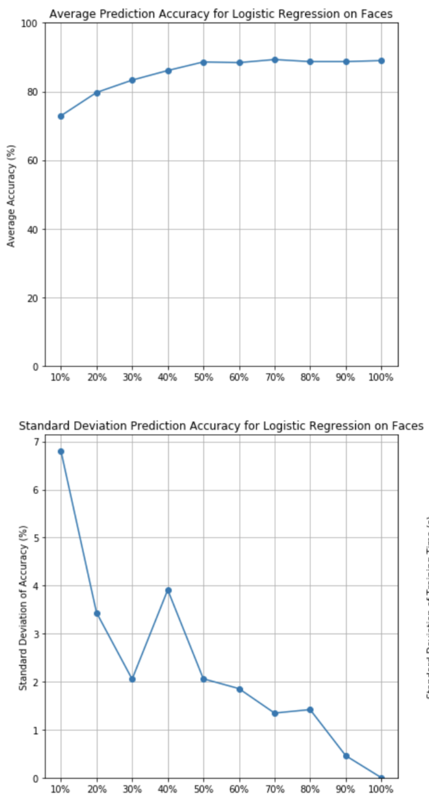

Figure 5: Accuracy and Time for Logistic Regression on Faces

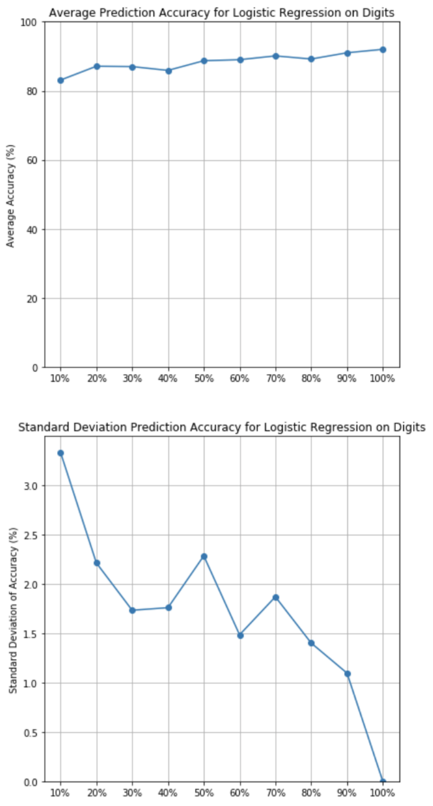

Figure 6: Accuracy and Time for Logistic Regression on Digits

As before, the mean training time increases as we increase the size of the training subset.

Overall, the Logistic Regression classifier seems to work better on the digits dataset than the faces dataset in terms of prediction accuracy. Our hypothesis for this difference is that for faces, different features, such as features that measure distance between certain filled-in pixels, would improve the accuracy score of the Logistic Regression classifier. The diversity of face shapes is the basis for this hypothesis. Whereas, for digits, the lack of diversity in the shape of the digits allows for a high degree of accuracy for the Logistic Regression classifier using the simple features. From the timing point of view, it took significantly more time to train the digits dataset compared to the faces dataset, which, again, we attribute to the the size difference between the two training datasets (in that the digits training dataset has substantially more observations than the faces training dataset).

3 Algorithm Performance Comparison

Altogether, the Logistic Regression classifier performed best (in terms of average prediction accuracy), among the three classifiers, on the digit dataset, but was significantly slower than the Naive Bayes classifier. For the face dataset, the Naive Bayes classifier and the Logistic Regression classifier ultimately achieved the same average prediction accuracy, 89%, but the Naive Bayes classifier was faster again (though slightly in this case). We believe that a change in features might unveil benefits for using the Percepton classifier, which was slower and less accurate than its peers. It should also be noted that the Logistic Regression classifier was optimized through the use of a publicly-available and widely-used library, whereas we built the key components of the Naive Bayes classifier and the Perceptron classifier. All 3 algorithms saw their average prediction accuracies and training times increase as the size of the training datasets grew.