**FINAL EXAM**

**By Meet Patel**

**RUID: 222002826**

**NetId: mp1978**

Overview  
This report analyzes the performance of Naive Bayes and Perceptron classifiers using digit and face datasets. The analysis considers various metrics such as accuracy, error, standard deviation, and computational time against the number of training data points. The insights are derived from the graphs.  
  
**Naive Bayes Classifier (Digits Dataset)**

Accuracy vs. Training Data Points:

A graph with numbers and lines

Description automatically generated

* Accuracy improves consistently up to approximately 2000 training data points, peaking at ~76.5%.
* Beyond this threshold, accuracy stabilizes or slightly declines, indicating diminishing returns on additional data.
* This behavior aligns with the Naive Bayes assumption of feature independence, which can limit performance on complex datasets.

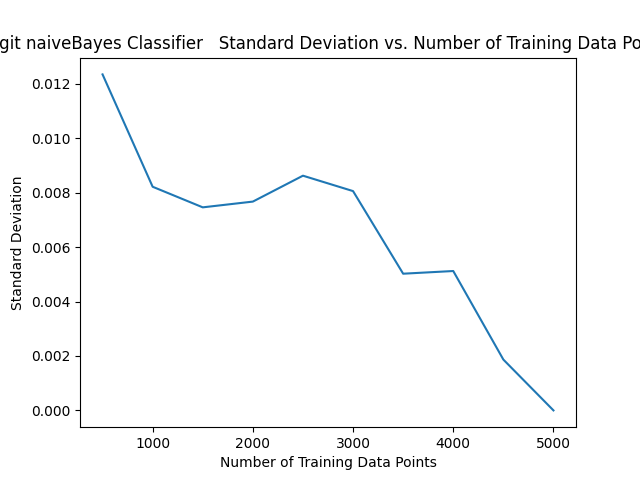
**Error vs. Training Data Points:**

A graph with a line

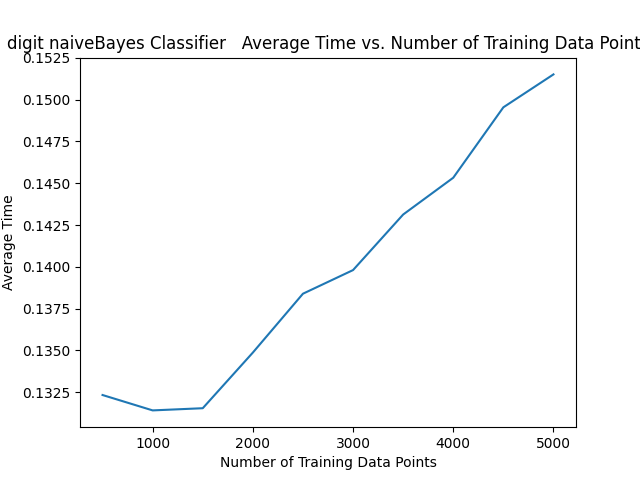
Description automatically generated

* Error decreases significantly with increasing data points, reaching its lowest near 3000 points.
* Slight error increases beyond this point reflect the corresponding minor declines in accuracy.

Standard Deviation vs. Training Data Points:

  
  
Variability in performance decreases as training data size grows, stabilizing after 4000 data points. This is due to consistent class priors and pixel likelihoods computed during training.

Time vs. Training Data Points:

  
  
Computational time increases linearly, influenced by the pre-computation of pixel counts and log-likelihoods during training. The linearity is expected given the operations scale with the number of data points.

**Naive Bayes Classifier (Faces Dataset)**  
  
Accuracy vs. Training Data Points:

A graph with numbers and lines

Description automatically generated

* Accuracy rises steeply with the first 100 training points, reaching ~85%, and then stabilizes.
* This rapid convergence suggests the classifier effectively models simpler datasets with fewer classes.

Error vs. Training Data Points:

A graph with a line

Description automatically generated

* Error decreases rapidly, stabilizing as accuracy plateaus.
* Reflects the model’s efficiency in handling the relatively smaller dataset size.

Standard Deviation vs. Training Data Points:

A graph with numbers and lines

Description automatically generated  
  
Standard deviation diminishes quickly, indicating consistent and stable predictions even with smaller datasets.

Time vs. Training Data Points:

A graph with a line

Description automatically generated

Linear increase in time reflects the classifier’s scalability for datasets with minimal complexity.

**Perceptron Classifier (Digits Dataset)**  
Accuracy vs. Training Data Points:

A graph of a number of training points

Description automatically generated

* Accuracy rises sharply to ~79% with the first 2000 training data points, stabilizing afterward.
* This trend highlights the Perceptron’s ability to learn robust weight vectors from linearly separable data.

Error vs. Training Data Points:

A graph with a line

Description automatically generated

* Error decreases consistently, stabilizing at ~21% as training size increases.
* Reflects the Perceptron’s iterative weight updates effectively minimizing misclassifications.

Standard Deviation vs. Training Data Points:

A graph with a line

Description automatically generated  
  
Declining variability demonstrates increased consistency, attributed to effective learning rates and epoch-based training.

Time vs. Training Data Points:

A graph with a line

Description automatically generated

Time scales linearly due to iterative weight adjustments over multiple epochs and large input sizes.

**Perceptron Classifier (Faces Dataset)**  
  
Accuracy vs. Training Data Points:

A graph with a line

Description automatically generated

* Sharp accuracy increase to ~87% with 100 training data points, followed by stabilization around 85%.
* Highlights the Perceptron’s efficiency in smaller, binary classification tasks.

Error vs. Training Data Points:

A graph with a line

Description automatically generated  
  
Error decreases inversely with accuracy, stabilizing as sufficient data is reached for training.

Standard Deviation vs. Training Data Points:

A graph with numbers and lines

Description automatically generated  
  
Declining standard deviation reflects consistency in predictions as the model converges.

Time vs. Training Data Points:

A graph with a line

Description automatically generated  
  
Linear increase in time is consistent with the Perceptron’s iterative training approach.

**Key Observations**

* Laplace Smoothing:

As you can see in the main file there is fixed value of laplace smoothing as I had tested multiple different value for it in the test file and 0.001 for some reason was performing the best.

* Edge weights:

For edge weights I observed that it didn’t have much impact on the overall accuracies as I had tried a lot of different values for it and also tried some extreme values but it had very minor impact on the overall performance.

* Performance Trends:  
    
  Naive Bayes excels with smaller, simpler datasets but shows diminishing returns with larger, more complex data.  
    
  Perceptron achieves higher accuracy, especially for linearly separable and smaller binary datasets.
* Stability:  
    
  Both classifiers show reduced standard deviation with increasing training data, indicating improved consistency.
* Scalability:  
    
  Computational time scales linearly for both classifiers. Perceptron is slightly more computationally demanding due to iterative weight updates.
* Classifier Suitability:  
    
  Naive Bayes is suitable for tasks where interpretability and simplicity are priorities.  
    
  Perceptron is ideal for tasks requiring higher accuracy and robust handling of larger, more complex datasets.
* Recommendations
  + When to Use Naive Bayes:  
      
    Suitable for datasets with independent features or limited complexity.  
      
    Recommended for scenarios where computational efficiency and interpretability are critical.
  + When to Use Perceptron:  
      
    Effective for high-accuracy tasks, especially with larger datasets and binary classification problems.  
      
    Preferred for applications where robustness to noise and iterative improvements are essential.
* Data Size Considerations:  
    
  Both classifiers show diminishing performance returns beyond optimal training sizes.  
    
  Evaluate trade-offs between computational cost and accuracy gains when expanding training datasets.

This report integrates code-based insights, ensuring a comprehensive analysis of the classifiers’ performance across datasets.