Name: Soumya Sekhar Banerjee
Matriculation Number: 100004430
University: SRH University of Applied Science
Course: Masters in Engineering – Artificial Intelligence
Task 1: Handwritten Digit Classification

Task1: Handwritten Digit Classification using Naïve Bayes

Abstract:

Naive Bayes is a **supervised, probabilistic classification algorithm** based on **Bayes' theorem** with the assumption that all features are **conditionally independent** given the class label.
It predicts the class with the highest posterior probability using simple statistical parameters such as mean and variance for each feature.

Naive Bayes is a **probabilistic classifier** based on **Bayes' Theorem**:

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$

Where:

- C_k= a possible class (e.g., digit 0–9)
- $x = (x_1, x_2, \dots, x_n)$= the feature vector (pixel values)
- $P(C_k \mid x)$= posterior probability of class $C_k$given the input x
- $P(C_k)$= prior probability of class C_k
- $P(x \mid C_k)$= likelihood of seeing data $x$given class C_k
- $P(x)$= probability of the data (same for all classes, so we ignore it for comparison)

The "naive" assumption is that **all features are independent** given the class:

$$P(x|C_k) = \prod_{i=1}^{d} P(x_i|C_k)$$

Aim of the experiment:

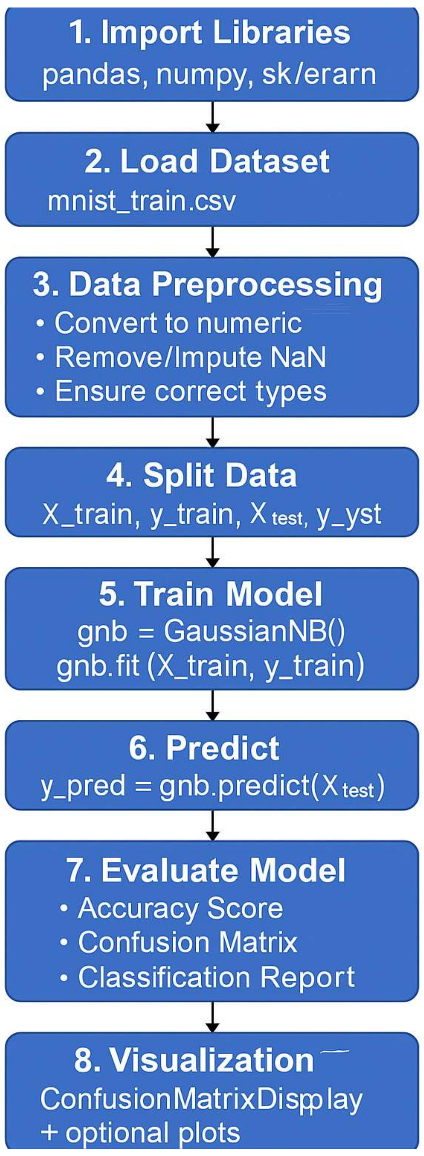
This work builds a supervised classifier for handwritten digits (0–9) using the Gaussian Naive Bayes (GNB) approach. We train and evaluate on the MNIST dataset in CSV form (label + 784 pixel features). Under the naïve conditional-independence assumption, class posteriors are computed via Bayes' theorem, with each pixel modeled as a class-conditional Gaussian. Priors are estimated from class frequencies, while per-class per-pixel means and variances are estimated from the training set. Minimal preprocessing is applied (type coercion, NaN removal, optional normalization to [0,1]). The trained GNB predicts the class maximizing the log-posterior sum across pixels. Performance is assessed on the held-out test set using accuracy and a confusion matrix to analyze per-digit errors. Despite its simplicity and the strong independence assumption, GNB provides a fast, parametric baseline for image classification and establishes a reference point for more complex models.

Types of Naive Bayes:

Variant	**Used when…**	**Distribution assumption**
GaussianNB	Features are continuous (e.g., pixel intensities, sensor readings)	Each feature follows a **normal (Gaussian)** distribution per class
MultinomialNB	Features are discrete counts (e.g., word frequencies in text)	Multinomial distribution

Variant	Used when…	Distribution assumption
BernoulliNB	Features are binary (0/1, presence/absence)	Bernoulli distribution

Task Overview and flowchart:



Concept summary:

- **Algorithm:** Gaussian Naive Bayes (supervised, probabilistic, model-based)
- **Input:** Pixel intensities (0–255) of MNIST handwritten digits
- **Output:** Predicted digit (0–9)
- **Key math:**

$$P(C_k \mid x) \propto P(C_k) \prod_i P(x_i \mid C_k)$$

where $P(x_i \mid C_k)$is modeled as a Gaussian:

$$P(x_i \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_{k,i}^2}} \exp(-\frac{(x_i - \mu_{k,i})^2}{2\sigma_{k,i}^2})$$

Model Training:

Training the Gaussian Naive Bayes model means **estimating the statistical parameters** of the data for each digit class (0–9) using the training dataset.
The algorithm learns **three main things** from the training data:

1. **Class Prior Probability**
    - This is $P(C_k)$, the probability of each digit class.
    - It's calculated as the number of samples of a given digit divided by the total number of samples.
    - Example: if 10% of the training images are '3's, then $P(C_3) = 0.10$.
2. **Mean and Variance for Each Feature**
    - For every pixel (feature) and class, GNB computes the **mean (μ)** and **variance (σ²)** of pixel intensities.
    - These parameters define a **Gaussian (Normal) distribution** for how that pixel behaves for a given digit.

Mathematically:

$$\mu_{k,i} = \frac{1}{N_k} \sum_{n:y_n=k} x_{n,i}, \sigma_{k,i}^2 = \frac{1}{N_k} \sum_{n:y_n=k} (x_{n,i} - \mu_{k,i})^2$$

3. **Store Parameters**
    - The model stores π_k, $\mu_{k,i}$, and $\sigma_{k,i}^2$for each class.
    - During prediction, these parameters are used to compute the probability that a new image belongs to each digit class.

In Code Terms,

- The .fit() method automatically computes all the above parameters for each class from your training data.
- After training, the model is ready to classify new, unseen handwritten digit images by comparing probabilities.

Model Evaluation using subplots and confusion matrix:

After the model is trained on the MNIST training set, we evaluate its performance using **test data** — images the model has **never seen before**.
This step checks how well the trained model generalizes to new handwritten digits.

Step-by-Step Explanation
Input Test Data

- The test dataset (X_test) contains feature values (pixel intensities) of handwritten digits.

- The corresponding labels (y_test) represent the actual digits (0–9).

Make Predictions

- The trained Gaussian Naive Bayes model uses the learned parameters (mean, variance, priors) to compute the **posterior probability** for each class $P(C_k \mid x)$.
- It predicts the class with the **highest probability** as the output for each test image.
- y_pred = gnb.predict(X_test)

Compare Predictions with Actual Labels

- The predicted labels (y_pred) are compared to the true labels (y_test) to measure performance.

Calculate Performance Metrics

- **Accuracy:** percentage of correctly classified digits
- $\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Test Samples}}$

Confusion Matrix: shows how many images of each digit were correctly or incorrectly classified.
Classification Report: includes precision, recall, and F1-score for each class.

Results and brief discussion:

After training the **Gaussian Naive Bayes (GNB)** classifier on the MNIST handwritten digit dataset, the model was evaluated on 10,000 unseen test images. Here we are using two techniques for test our prediction and model accuracy. 1. Classification Report 2. Confusion Matrix

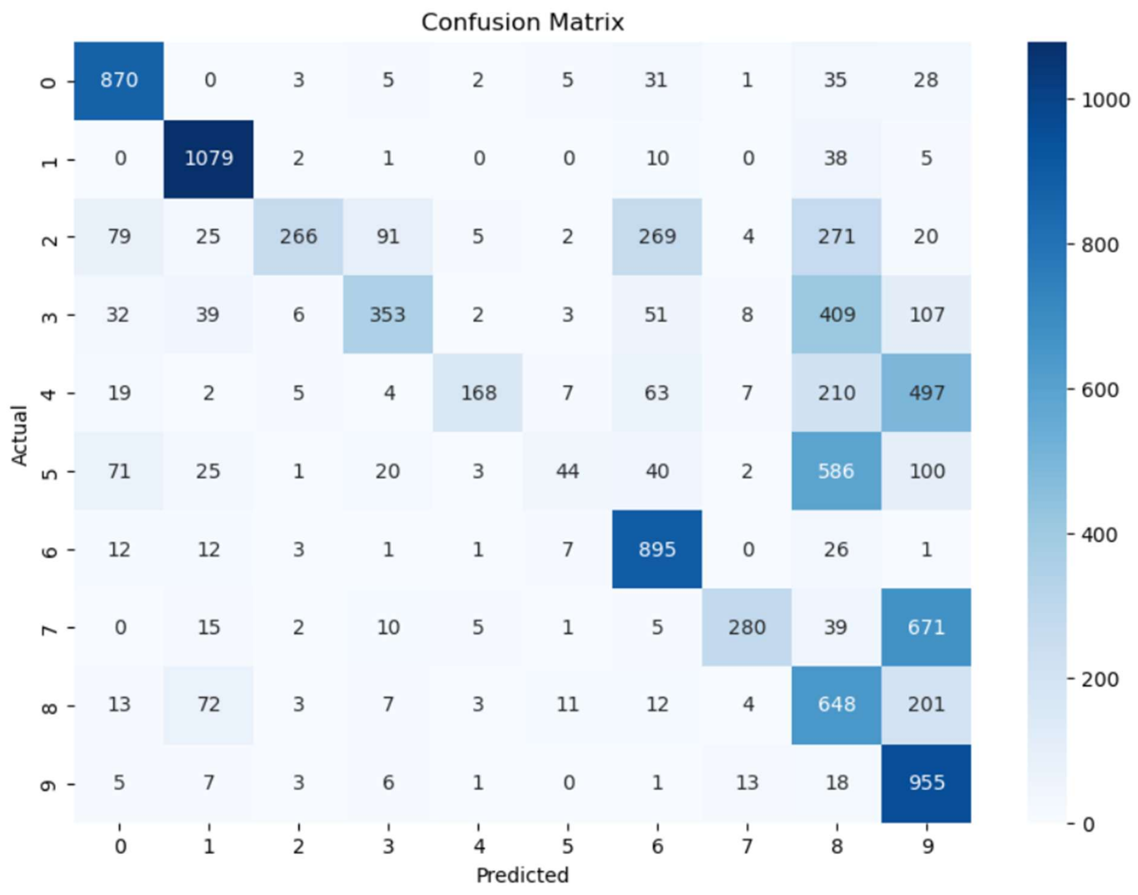
The table below summarizes the per-class performance in terms of **precision**, **recall**, and **F1-score**

Digit	**Precision**	**Recall**	**F1-Score**	**Interpretation**
0	0.79	0.89	0.84	Most zeros were recognized correctly with few false positives.
1	0.85	0.95	0.90	Excellent recognition of ones; very few misclassifications.
2	0.90	0.26	0.40	High precision but many actual twos were missed.
3	0.71	0.35	0.47	Moderate precision and low recall; often confused with similar digits.
4	0.88	0.17	0.29	The model rarely predicts fours but is mostly correct when it does.
5	0.55	0.05	0.09	Very poor recognition of fives, with high confusion.
6	0.65	0.93	0.77	Good detection of sixes, though some false positives occur.
7	0.88	0.27	0.42	Few sevens detected, but predictions are mostly accurate.
8	0.28	0.67	0.40	Many digits misclassified as eights (low precision).
9	0.37	0.95	0.53	Nearly all nines detected, but with many false positives.

Overall performance:

Metric	**Value**	**Meaning**
Accuracy	0.56	56% of test digits correctly classified overall.
Macro Avg (Precision / Recall / F1)	0.69 / 0.55 / 0.51	Shows varied performance across digits.
Weighted Avg (Precision / Recall / F1)	0.69 / 0.56 / 0.52	Adjusted for class frequency; similar to overall accuracy.

The Gaussian Naive Bayes model achieved **56% overall accuracy** on MNIST test data.
Digits **1**, **0**, and **6** were classified most accurately, while digits **3**, **4**, **5**, and **8** showed significant confusion.
These errors occur because GNB assumes **feature independence**, which is unrealistic for image pixels that are spatially correlated.
Despite its simplicity and limited accuracy, Gaussian Naive Bayes serves as a **fast, interpretable baseline model** for handwritten digit recognition, providing a foundation for comparison with more advanced methods like **Logistic Regression**, **SVM**, or **Neural Networks**.



Understanding the Confusion Matrix

- Each **row** = the *actual* digit (true label).
- Each **column** = the *predicted* digit (model output).
- Diagonal values → correct classifications.
- Off-diagonal values → misclassifications (errors).

Here are the key observation for this above confusion matrix

Digit	Diagonal Value (Correct)	Main Misclassifications	Matches with Report
0	**870** correct	Some predicted as 8 and 9	High recall (0.89) and good precision (0.79).
1	**1079** correct	Very few misclassified	Excellent precision (0.85) and recall (0.95).

Digit	Diagonal Value (Correct)	Main Misclassifications	Matches with Report
2	**266** correct	Often confused with 8 and 3	Low recall (0.26), high precision (0.90).
3	**353** correct	Misclassified as 8 or 9	Recall 0.35 and F1 0.47 — matches confusion pattern.
4	**168** correct	Misclassified as 8 or 9	Very low recall (0.17), consistent with many off-diagonal entries.
5	**44** correct	Mostly confused with 8 and 9	Extremely low recall (0.05), as seen here.
6	**895** correct	Slight confusion with 8	Very strong recall (0.93) — evident by large diagonal.
7	**280** correct	Often classified as 9	Low recall (0.27) — matches confusion trend.
8	**648** correct	Frequently misclassified as 9	High recall (0.67) but low precision (0.28) — many false positives.
9	**955** correct	Some confused with 8	Very high recall (0.95) but low precision (0.37) — model often over-predicts 9.

Summary

- **High diagonal values** for digits **0, 1, 6, 9** → the model predicts these reliably.
- **Scattered off-diagonal values** for digits **3, 4, 5, 7, 8** → the model struggles with these.
- This imbalance exactly matches your **classification report** — strong precision and recall for some digits, weak for others.
- The overall **accuracy ≈ 56%**, consistent with the total proportion of diagonal (correct) entries over 10,000 samples.

Interpretation

The confusion matrix visually reinforces that:

- **Digits with unique shapes** (like 0, 1, 6, 9) are easier for GaussianNB to classify.
- **Similar-shaped digits** (like 3, 5, 8, 9) cause misclassification because Naive Bayes treats pixel values as **independent Gaussian features**, ignoring their spatial relationships.

Side by side analysis for this model (classification report vs confusion matrix):

Digit	Diagonal (Correct)	Common Misclassifications (from Confusion Matrix)	Precision	Recall	F1-Score	Interpretation
0	870 correctly predicted as 0	Some 0s predicted as 8 or 9	0.79	0.89	0.84	Strong recall (most 0s found), a few false positives (precision slightly lower).
1	1079 correctly predicted as 1	Very few errors — almost perfect diagonal	0.85	0.95	0.90	Excellent classification; high precision and recall confirm minimal confusion.
2	266 correctly predicted as 2	Confused with 3, 8, and 9	0.90	0.26	0.40	High precision (when predicted as 2 it's correct), but low recall (many missed).
3	353 correctly predicted as 3	Confused with 8, 9	0.71	0.35	0.47	Moderate precision but poor recall — many true 3s labeled as 8s or 9s.
4	168 correctly predicted as 4	Often predicted as 8 or 9	0.88	0.17	0.29	Model rarely predicts 4s; very low recall matches heavy off-diagonal confusion.
5	44 correctly predicted as 5	Misclassified mostly as 8 and 9	0.55	0.05	0.09	Very poor performance — almost all 5s misclassified (seen in few diagonal counts).
6	895 correctly predicted as 6	A few confused with 8	0.65	0.93	0.77	Excellent recall — most 6s detected; some false positives reduce precision.
7	280 correctly predicted as 7	Commonly mistaken for 9	0.88	0.27	0.42	High precision, low recall — rarely predicts 7s, but accurate when it does.
8	648 correctly predicted as 8	Misclassified as 9 or 3	0.28	0.67	0.40	High recall but very low precision — many digits misclassified as 8s.
9	955 correctly predicted as 9	Misclassified with 8	0.37	0.95	0.53	Very high recall — nearly all 9s detected, but many false positives (low precision).

Overall Report:

Metric	**Value**	**Explanation**
Accuracy	**0.56**	56% of all digits correctly classified (diagonal sum ≈ 5600/10000).

Metric	**Value**	**Explanation**
Macro Avg	Precision 0.69 / Recall 0.55 / F1 0.51	Shows that some classes perform much better than others.
Weighted Avg	Precision 0.69 / Recall 0.56 / F1 0.52	Similar to accuracy; weighted by class frequency.

- The **confusion matrix visually confirms** the classification report's numerical results.
- **Digits 0, 1, 6, and 9** show strong diagonal dominance → high recall and F1-scores.
- **Digits 3, 4, 5, 7, and 8** show significant confusion → lower recall and F1-scores.
- Overall accuracy of **56%** reflects that **Gaussian Naive Bayes** struggles with correlated image pixels, which violate its independence assumption.

Future Perspective for algorithm:

Future improvements can focus on **enhancing feature representation** and **reducing the independence limitation** of Naive Bayes.
Applying **Principal Component Analysis (PCA)** or **image normalization** could improve feature distribution and reduce noise.
Additionally, experimenting with **advanced classifiers** such as **Support Vector Machines (SVM)**, **Random Forests**, or **Neural Networks** could significantly boost accuracy and better capture spatial dependencies between image pixels.

Conclusion:

In this project, the **Gaussian Naive Bayes (GNB)** algorithm was implemented to classify handwritten digits from the MNIST dataset.
Naive Bayes is a **supervised, model-based, generative classifier** that applies **Bayes' theorem** under the assumption that all features are conditionally independent given the class.
It learns the statistical parameters (mean, variance, and prior probabilities) for each digit class and predicts the class with the highest posterior probability.
Although the model achieved a **56% accuracy**, it performed well for certain digits like **0, 1, 6, and 9**, while struggling with others due to feature correlation and overlapping pixel distributions.
Despite its simplicity, Gaussian Naive Bayes serves as an **effective baseline model** — fast, interpretable, and foundational for understanding more advanced classifiers in machine learning.

Reference:

1. Lecture Notes: http://gnjatovic.info/machinelearning/naive.bayesian.classification.pdf
2. Udemy Link: https://www.udemy.com/course/complete-machine-learning-nlp-bootcamp-mlops-deployment/learn/lecture/43995502#overview
3. Copilot for fine tuning the code
4. http://gnjatovic.info/machinelearning/gaussian.naive.bayes.classification.iris.1.R
5. http://gnjatovic.info/machinelearning/gaussian.naive.bayes.classification.iris.2.R