Machine Learning - Assignment 01

Jonas Ortner (joortner, 2265527)
Marmee Pandya (mpandya, 1963521)

3. Oktober 2025

3 Experiments on MNIST digits data

3a Accuracy and Error Analysis

We trained a categorical Naive Bayes classifier with symmetric Dirichlet prior on the 60,000 MNIST training images and evaluated on the 10,000 test images. With $\alpha = 2$, the classifier achieved an overall test accuracy of **83.63%**. This means that roughly 8,363 out of 10,000 digits were correctly recognized.

To contextualize this performance, we compared it against a simple *zero-rule* baseline classifier that always predicts the most frequent class. In our training data, digit "1" is the most common with frequency 11.23%. A classifier that always predicts "1" achieves only 11.35% accuracy on the test set. Thus, Naive Bayes improves accuracy by a factor of about seven over this baseline.

A more detailed picture of model performance is obtained from the classification report (precision, recall, F1-score). Precision values across classes range between 0.75 and 0.91. Digits 0 and 7 have the highest precision (0.91), while digits 8 and 9 are lower. Recall varies from 0.67 (digit 5) to 0.97 (digit 1), showing that some classes are more easily detected than others. F1-scores follow the same pattern, with values between 0.72 and 0.91.

The confusion matrix (Figure 1) illustrates class-specific misclassifications. The largest error occurs when digit "5" is misclassified as "3" (128 instances, corresponding to $\sim$14% of all fives). Another frequent confusion is "4" being mistaken for "9" (119 instances). These confusions are not symmetric: a 5 is much more likely to be classified as 3 than vice versa. On the other hand, some digit pairs are rarely confused; for example, 0 is never predicted as 1, reflecting their distinct pixel distributions.

Representative correctly classified examples are shown in Figure 2, while Figure 3 displays misclassified test images. Most errors are human-interpretable, with ambiguous writing styles or overlapping stroke patterns.

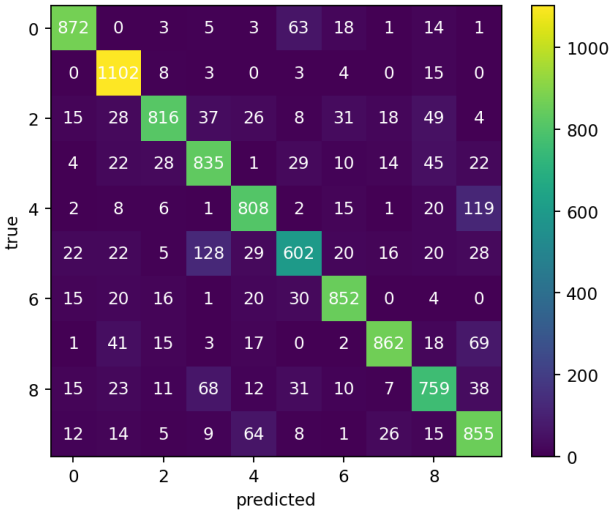

Figure 1: Confusion matrix of Naive Bayes classifier with $\alpha = 2$.

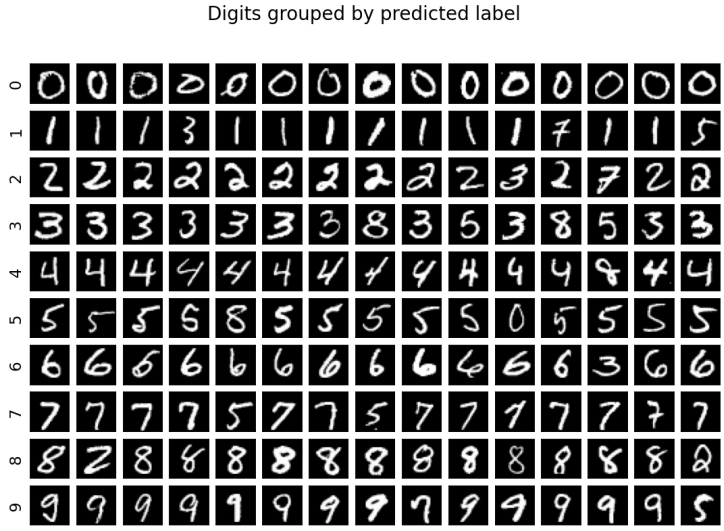

Figure 2: Correctly classified MNIST digits grouped by predicted label.

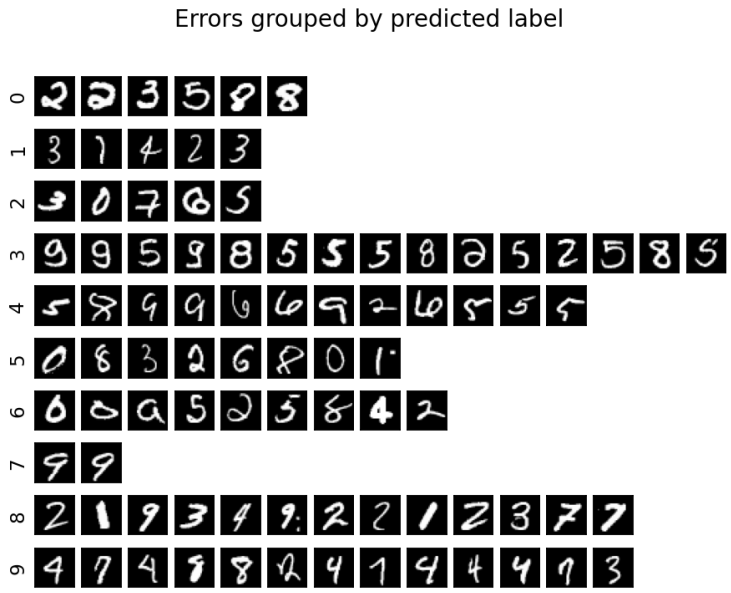

Figure 3: Misclassified MNIST digits grouped by predicted label.

3b Confidence Analysis

Besides predicting labels, the Naive Bayes classifier also produces posterior probabilities, which can be interpreted as model confidence. To investigate the relationship between confidence and accuracy, test examples were ordered by predicted posterior probability.

The cumulative accuracy curve (Figure 4) shows that the top-ranked predictions, with confidence values close to one, are almost always correct. As less confident predictions are added, accuracy gradually declines. This demonstrates that posterior probabilities carry meaningful information about reliability.

Binning predictions by confidence (Figure 5) confirms the trend: predictions in the highest confidence bins achieve near-perfect accuracy, whereas those with lower confidence are much less reliable. Nevertheless, some high-confidence errors remain, illustrating that Naive Bayes can still be overconfident when pixel independence assumptions are violated.

4 Model Selection with Cross-Validation

4a Hyperparameter Search

The Dirichlet prior parameter α controls smoothing of the categorical distributions. To select an appropriate value, we performed 5-fold cross-validation on a balanced sample of the training set, testing $\alpha \in \{1, 2, 5, 10, 50, 100\}$. The results are summarized in Table 1.

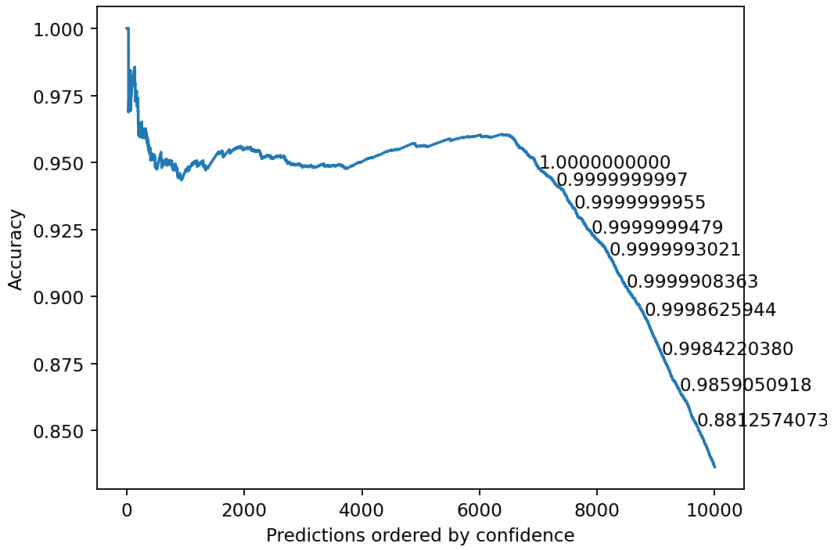

Figure 4: Cumulative accuracy as test examples are added in decreasing order of confidence.

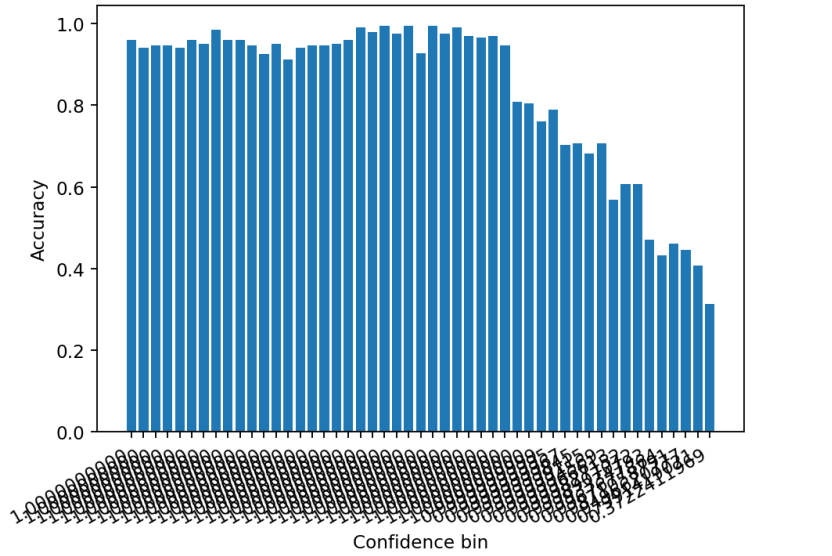

Figure 5: Prediction accuracy across confidence bins. High-confidence predictions are more reliable.

α	Mean CV Accuracy	Std. Deviation
1	0.100	0.027
2	**0.483**	0.127
5	0.365	0.141
10	0.307	0.133
50	0.200	0.104
100	0.181	0.095

Table 1: 5-fold cross-validation accuracies for different α values.

4b Interpretation

Figure 6 visualizes cross-validation accuracy versus α. Both too little smoothing ($\alpha = 1$) and too much smoothing ($\alpha \geq 50$) harm performance. Moderate smoothing ($\alpha = 2$) achieves the best validation accuracy, consistent with the test set evaluation where $\alpha = 2$ reached 83.63%.

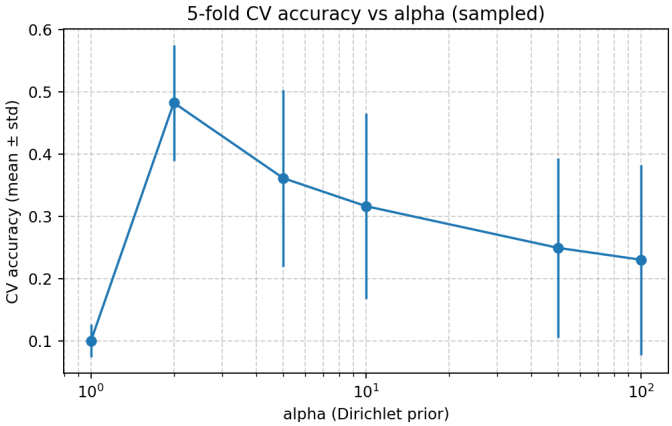

Figure 6: Cross-validation accuracy as a function of α.

5 Generating data

In this task, the Naive Bayes classifier we implemented in Task 2 was used to generate sample images for a given digits. For each of the 256 pixels, given the number y a value for that pixel $X_{i,j}$ was sampled according to the probability distribution modelled by the Naive Bayes classifier.

5.1 Plotting

Figure 7 shows for each digit 10 images created by the Naive Bayes classifier. The most numbers can be recognized. However, the contrasts are high and the image is noisy.

This gets evident when comparing with the plots of the expected values which will be discussed below in paragraph 5.1. $\alpha = 2$ was chosen to avoid the zero-count problem. In contrast, in Figure 8, I chose $\alpha = 100$. One can clearly see that a larger value of α leads to more noise, as the posterior gets smoothened towards a uniform distribution. The probabilities near the edges are increased (compared to the $\alpha = 2$ case) whereas probabilites in the middle are decreased.
Choosing a symmetric Dirichlet prior doesnt match the problem, as the distribution of the coloured pixels is highly comples. Therefore, choosing $\alpha = 2$ in order to avoid the zero-count problem seems to be the most suitable prior.

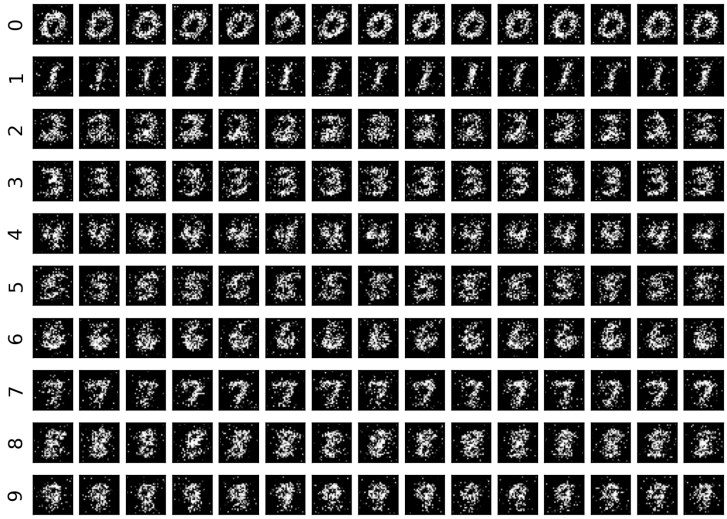

Figure 7: Generating images with sampling for $\alpha = 2$

In contrast to the previous images, in Figure 9a for each pixel X_{ij} the corresponding feature values was not sample according to the distribution suggested by the Naive Bayes classifier, but the Maximum value of the Naive Bayes classifier was chosen. This leads to a deterministic behaviour and a single Maximum Likelihood Image for each class.
Lastly, Figure 9b takes for each X_{ij} the expected value for a given digit. Again, this deterministic procedure leads to a single image for each class. From the perspective of the categorical Naive Bayes classifier this would be forbidden, as for categorical features it doenst make sense to apply aggregation functions, such as taking the mean. As the features indicate brightness values for each pixel, interpreting them as numerical features makes sense. Therefore, we switch our viewing point for the folloing task, so to speak.
What strikes the eye is that the contrasts are decreased, and the images of the digits appear much smoother. Also, there is less noise, but the images are less bright as well. This could be explained by assuming that the original images contain many pixels with

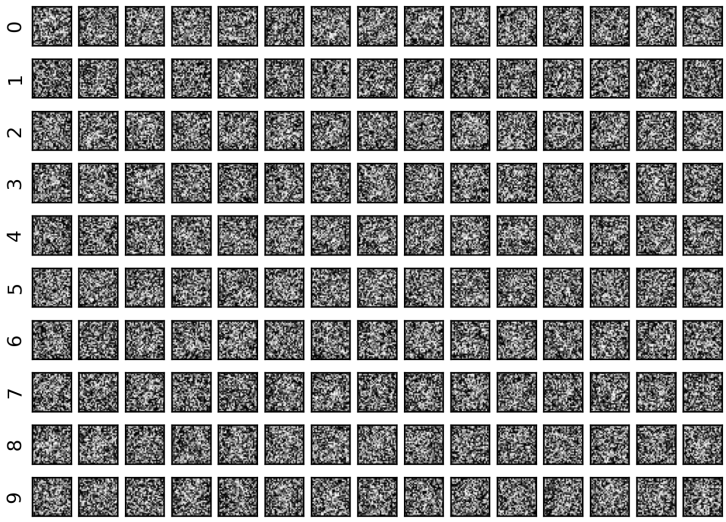

Figure 8: Generating images with sampling for $\alpha = 100$

feature values either close to 0 (black pixel) or 255 (white pixel). The Naive Bayes classifier learns this and outputs larger probabilites for extrem values. The resulting images (both for the sampling and the Maximum Likelihood images) have high contrasts, but also noise. If one chooses the mean for each pixel, boundaries are smoothened, noise vanishes, but the images become darker as well.

(a) Feature vector for $\alpha = 2$

(b) Expected values for $\alpha = 2$

Figure 9: Comparison of feature vector and expected values for $\alpha = 2$

6 Missing Data

With Bayes Rule, the following formula can be derived:

$$p(y|x_{1:D}) = \frac{p(y,\ x_{1:D})}{p(x_{1:D})} = \frac{p(y) \cdot \prod_{j=1}^{D} p(x_j|y)}{\sum_{y \in C} \left(p(y) \cdot \prod_{j=1}^{D} p(x_j|y) \right)} \ . \tag{1}$$

It is useful to calculate the posterior probability of a new datapoint $\vec{x} = (x_1, ..., x_D)^T$ belonging to class y.

The next formula can be derived in a similar fashion:

$$p(y|x_{1:D'}) = \frac{p(y,\ x_{1:D'})}{p(x_{1:D'})} = \frac{p(y) \cdot \prod_{j=1}^{D'} p(x_j|y)}{\sum_{y \in C} \left(p(y) \cdot \prod_{j=1}^{D'} p(x_j|y) \right)}, \ 1 \leq D' < D \ . \tag{2}$$

In can be used, if some features are missing and the datapoint contains only D' features, but one is still interested in the posterior probability of the datapoint belonging to class y.

$$p(x_{D'+1:D}|x_{1:D'}) = \frac{p(x_{1:D})}{p(x_{1:D'})} = \frac{\sum_{y \in C} \left(p(y) \cdot \prod_{j=1}^{D} p(x_j|y) \right)}{\sum_{y \in C} \left(p(y) \cdot \prod_{j=1}^{D'} p(x_j|y) \right)}, \ 1 \leq D' < D \ . \tag{3}$$

In the same case of missing features for a datapoint, the last formula can be used to generate the missing features. Either one could take for each feature the value with the highest probability, or alternatively sample with the corresponding probabilites for each value.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Bachelor-, Master-, Seminar-, oder Projektarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und in der untenstehenden Tabelle angegebenen Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Declaration of Used AI Tools

Tool	Purpose	Where?	Useful?
ChatGPT	Rephrasing	Throughout	+
GPT-4	Code generation	a01_functions.py	+
ChatGPT	Debugging	Throughout	+

Unterschrift
Mannheim, den 03. Oktober 2025