Authors

Roaa Fathi Nada	*20210140*
Selsabeel Asim Ali Elbagir	*20210714*
Basma Mahmoud Hashem	*20210091*
Howida Adel Abd El-Halim	*20210446*

Project 1: MNIST

Model Evaluation and Hyperparameter Tuning

	SVM	KNN	GNB

Accuracy	Degree = 3
	C = 0.8: 98.34%
	C = 1.2: 98.34%
	
	C = 1.0
	Degree = 2: 98.35%
	Degree = 4: 98.35%
		K nearest neighbor
		1 : 97.82%
		3 : 97.65%
		5 : 97.58%
			Variable Smoothing:
			1e-5 : 87.88%
			1e-6 : 86.99%
			1e-7 : 86.05%
			1e-8 : 84.87%
			1e-9 : 83.55%

Precision	Degree = 3
	C = 0.8: 0.98
	C = 1.2: 0.98
	
	C = 1.0
	Degree = 2: 0.98
	Degree = 4: 0.98
		K nearest neighbor
		1 : 0.98
		3 : 0.98
		5 : 0.98
			Variable Smoothing:
			1e-5 : 0.89
			1e-6 : 0.88
			1e-7 : 0.88
			1e-8 : 0.87
			1e-9 : 0.86

F1 Score	Degree = 3
	C = 0.8: 0.98
	C = 1.2: 0.98
	
	C = 1.0
	Degree = 2: 0.98
	Degree = 4: 0.98
		K nearest neighbor
		1 : 0.98
		3 : 0.98
		5 : 0.98
			Variable Smoothing:
			1e-5 : 0.88
			1e-6 : 0.87
			1e-7 : 0.86
			1e-8 : 0.85
			1e-9 : 0.84

In our code, three different classification models are evaluated on a dataset using HOG (Histogram of Oriented Gradients) features.
The models include k-Nearest Neighbors (KNN), Support Vector Machine (SVM) with a linear kernel, and Gaussian Naive Bayes.

1. K-Nearest Neighbors (KNN):

KNN is a non-parametric algorithm that classifies a data point based on the majority class among its k-nearest neighbors. In the code, the parameter `k` represents the number of neighbors considered during classification. As `k` increases, the decision boundary becomes smoother, which may result in a more generalized model. In this case, the accuracy decreases slightly with increasing `k`, indicating that a smaller `k` value (e.g., 1) is preferred for this dataset.

2. Support Vector Machine (SVM):

The way that SVM works is that it tries to find a hyperplane that separates classes in the feature space in the best way. We tried multiple different hyperparameters, changing the regularization parameter(C) and the polynomial kernel degree(d). We found that a larger polynomial kernel degree led to a more complex model, which could cause overfitting. And we also found that the more we decreased the c value, a wider margin happened which meant better generalization (but also can misclassify data more often). Overall, our results showed that a moderate C value of 1.0 and a low polynomial degree of 2 achieved the highest accuracy. The precision and F1 score was the same for all 3 models which shows that it was not an appropriate accuracy metric for our models.

3. Gaussian Naive Bayes:

Gaussian Naive Bayes assumes that features are normally distributed within each class. The hyperparameter `var_smoothing` controls the portion of the largest variance of all features added to the variances for calculation stability. The code tests different values for `var_smoothing`. We found that the smallest test value we took for the hyperparameter, 1e-5, gave the best accuracy, showing that a smoother distribution assumption improved the model's performance.

Model Comparison:

Comparing the models, SVM with a linear kernel and K-Nearest Neighbors both achieved high accuracies. However, SVM with a linear kernel and a polynomial degree of 2 did better than others in terms of precision and F1 score in all cases. The KNN model, while accurate, did not perform better than the SVM and Gaussian Naive Bayes model.

Conclusion:

Considering the achieved results, SVM with a linear kernel and a polynomial degree of 2 is recommended for this dataset. It has a good balance between accuracy, precision, and F1 score.

Error Analysis:

The cause of the error percentage may be the critical representation of features; MNIST images are typically represented as matrices of pixel values, and if the chosen feature representation fails to capture relevant patterns, the classifier may make errors. Overfitting, where the model learns the training data too well, and underfitting, where the model is too simple to capture underlying patterns, are both problems we need to be careful of in classification.

The possible cause of the error percentage using the **Gaussian Naive Bayes Model** is that it assumes that pixels in a digit image are independent of each other. However, in reality, pixels in handwritten digits are often related. This can lead to mistakes in classification. Understanding these relationships between pixels is important for making the model better.

Another thing, the MNIST dataset itself is quite diverse, with various handwriting styles and differences in how digits look. The Gaussian Naive Bayes model, despite being simple, might struggle to catch all these details. Using more advanced models like SVM and KNN, or others could be a way to recognize handwritten digits.

Another interesting cause for error I discovered while doing research online, was that we could further improve the way the models extract feature data by labeling “good” and “bad” handwriting. This way, the model does not learn bad handwriting’s human error and wrongly misclassify good handwriting due to it. Separating those two will lead to less wrongly misclassified data.

K-Nearest Neighbors(KNN) Classifier Improvements:

The KNN model achieved high accuracy, but misclassifications may occur due to its sensitivity to local patterns. Some images with similar local patterns might belong to different classes.

Suggestions for Improvement:

1. ***Optimize k-value***: Experiment with different values of k to find the optimal one. Using cross-validation, you can identify the k value that generalizes well to new data.
2. ***Feature Scaling***: Ensure that feature scaling is applied. KNN is sensitive to the scale of features, so normalizing or standardizing the data might improve performance.

Support Vector Machine(SVM) Classifier Improvements:

SVM performed well, but errors may occur when the decision boundary is not well-defined, especially in cases with complex data or noisy features.

Suggestions for Improvement:

1. ***Fine-Tune Hyperparameters***: Try different values of the regularization parameter (C) and the polynomial kernel degree to find the combination that maximizes performance.
2. ***Data Preprocessing***: Check if additional preprocessing steps such as data cleaning or feature engineering can improve SVM performance.

Gaussian Naive Bayes Classifier Improvements:

Gaussian Naive Bayes may struggle with complex relationships between features. It assumes independence between features, which may not hold in some cases.

Suggestions for Improvement:

1. Feature Independence Check: Examine feature correlations. If features are highly correlated, try other models that can handle dependencies between features.