ANALYSIS OF MNIST DATASET USING THE REGULARIZED MULTINOMIAL LOGIT MODEL AND SUPPORT VECTOR MACHINES

Sotiris Zenios 8657785

Nikolas Stavrou 0906581

Andreas Alexandrou 9503978

Utrecht University

INFOMPRDL: Pattern Recognition and Deep Learning

Table of Contents

Introduction

Purpose of the Study

This study delves into the MNIST dataset, a very popular dataset in the domain of machine learning. The MNIST dataset, originally consisting of 70,000 images of handwritten digits, serves as a benchmark for evaluating machine learning models in the domain of image processing and classification. The primary aim is to perform a quality analysis of this dataset and different classification methods and derive insightful conclusions.

Overview of the Approach

We start with an initial exploratory analysis where we investigate the dataset's characteristics, including the identification of any non-contributing variables and an assessment of class distribution. Statistical metrics such as mean, standard deviation, and 'total ink'—the aggregate of pixel intensities—were computed for each digit class. These metrics provided foundational insights into the characteristics of the dataset. Furthering the feature engineering process, an innovative approach was employed by segmenting each image into four quadrants. The 'total ink' for each quadrant was calculated, aiming to encapsulate the spatial distribution of pixel intensities within each digit.

Two machine learning algorithms were central to this analysis – the regularized multinomial logit model employing the LASSO (Least Absolute Shrinkage and Selection Operator) penalty, and Support Vector Machines (SVMs). For each model, we focus on fine-tuning and identifying the best hyperparameters to optimize their performance in digit classification using cross validation. We then compare these models in terms of effectiveness and efficiency, and we employ statistical tests, with a focus on identifying any notable differences in their accuracy levels, offering a comprehensive view of their applicability to the MNIST dataset.

This report details the methodologies adopted from data preprocessing to model evaluation. It aims to provide a clear, reproducible account of the findings, underlining the functions and settings used in the Python implementation. This approach aligns with the principles of empirical research, ensuring the transparency and reproducibility of the analysis.

Description and Exploratory Analysis of the Data

Data Overview

We use a lightly altered MNIST dataset which comprises of 42,000 images of handwritten digits, each depicted in a grayscale 28x28 pixel format. This translates into 784-pixel values per image, with each pixel value serving as an individual feature representing the intensity at a specific location in the image, with their values ranging from 0 to 255.

Exploratory Data Analysis

Redundant Features: Initial analysis revealed the presence of redundant features in the dataset. Several pixel columns, such as pixel0, pixel1, pixel2, etc., displayed a standard deviation of zero, an indication that is consistent across all images. These columns provide no variability or distinguishing information and do not contribute to the modeling process.

Class Distribution: The target variable 'label' represents digit classes (0-9). The class distribution was found to be relatively well-balanced. The most common class is '1', constituting approximately 11.15% of the dataset, indicating a slight but firm class imbalance.

Majority Class Baseline: A baseline model, predicting only the majority class ('1'), would achieve an accuracy of about 11.15%. This baseline serves as a benchmark, with the expectation being that any effective model should significantly outperform this accuracy.

Experimental Setup

Feature Engineering

Preliminary Model Analysis Using 'Ink' Feature

The initial phase of the experimental setup involved the exploration of a simple yet intuitive feature - the total amount of 'ink' used in each digit, calculated as the sum of pixel values. This feature was utilized to fit a multinomial logistic regression model, aiming to evaluate its power in digit classification.

Model Performance: The accuracies obtained for each digit class varied significantly, reflecting the limited capability of the 'ink' feature in isolation. More notably, classes like '1' showed higher accuracy, while others, particularly '4', '5', '6', '8', showed notably lower accuracies, with '8' recording the worst one. This inconsistency highlighted the feature's weakness in capturing the spatial complexities inherent in different digit classes.

Introduction of Quadrant Density Features

To enhance the feature set and potentially improve model performance, a new approach was adopted by segmenting each digit image into four quarters and calculating the pixel densities in each quarter. This method aimed to capture more detailed information about the spatial distribution of pixel intensities.

Feature Extraction Process: Each image in the dataset was split into four parts, resulting in four new features per image (14x14), each representing the total ink in one of the four quadrants.

Model Training with Quadrant Features: A logistic regression model was then trained using these quadrant density features. The features were scaled using the scale function to ensure uniformity in feature magnitude.

Confusion Matrix and Accuracy: The performance of the logistic regression model, when trained on quadrant density features, was evaluated. The confusion matrix and accuracy metrics were used to assess the model's effectiveness in correctly classifying each digit class.

Combination of both features

To enhance the model's predictive capability, a combination of the 'ink' feature and the quadrant density features was employed. This integration aimed to leverage both the overall ink usage and the spatial distribution of ink across the digit images. The 'ink' feature provides a global measure of pixel intensity, while the quadrant density features offer a more granular insight into the spatial characteristics of the digits.

Model Training and Evaluation: The multinomial logit model was trained using this combined feature set, comprising the total 'ink' feature and the four quadrant density measures. Prior to training, all features were scaled to ensure consistency in magnitude and influence.

Analysis of Model Performance: Surprisingly, the combination of these features did not yield an improvement in the model's overall accuracy. This outcome suggests a potential overlap in the information conveyed by the 'ink' and quadrant density features. Both sets of features essentially derive their values from pixel intensities, albeit at different granularities. The 'ink' feature summarizes the total pixel intensity of the entire image, while the quadrant features do so across four different parts of the image.

Implications and Interpretation: The lack of improvement in accuracy indicates that the quadrant density features might already encompass the information provided by the total 'ink' feature. In essence, the total ink used in a digit can be inferred from the ink densities in the four quadrants. Therefore, the addition of the total 'ink' feature does not introduce new discriminative information useful for classification. This finding underscores the importance of not only creating features but also

understanding the interrelationships and redundancies among them. It highlights that more features do not necessarily translate to better performance, especially when they are correlated or provide overlapping information.

Data Preparation

Data Sampling

As part of our analytical process, we focused on the utilization of the raw pixel values of the MNIST dataset images as individual features. This approach differs from our previous feature engineering methods, such as the 'ink' and quadrant density features, and instead relies on the inherent pixel data of each image.

The MNIST dataset images are 28x28 pixels, resulting in a total of 784 features per image, with each pixel representing an individual feature. This high-dimensional feature space provides a detailed representation of each digit, capturing intricate variations in handwriting and style.

From the dataset, a random sample of 5,000 digit images was drawn for the purposes of training and model selection. This sampling process was executed with the train_test_split function from the sklearn library, employing a stratified approach (stratify=y) to maintain the class distribution of the digits.

We employed 5-fold cross-validation within this sample to facilitate robust model selection and hyperparameter tuning. Cross-validation, a standard practice in machine learning, helps in assessing the model's generalization capability and mitigates the risk of overfitting.

After finalizing the models and their respective hyperparameters, we shifted our focus to evaluating the performance of these models on the unseen data. The remaining 37,000 digit images in the MNIST dataset, which were not included in our initial sample, were used as a test set. This extensive test set allowed us to estimate the error of the models and evaluate their performance.

Feature Reduction

As an important preprocessing step, images were resized from their original 28x28 pixel resolution to a 14x14 pixel resolution using the resize function from the OpenCV library [4]. This reduced the feature dimensions from 784 to 196 per image. Therefore, greatly reducing the complexity and execution time of the training procedure.

Model Selection and Hyperparameter Tuning with raw pixel values

Regularized multinomial logit model (LASSO)

Model Implementation: The regularized multinomial logit model using the LASSO penalty (L1), was implemented using the LogisticRegression function from sklearn [5]. This model was selected to assess its effectiveness in the context of digit classification within the MNIST dataset.

Hyperparameter Tuning: Hyperparameter tuning focused on the regularization strength (C), which is a crucial factor in LASSO regularization. This tuning plays a role in controlling the balance between model complexity and the degree of regularization, thus impacting the model's ability to generalize while retaining essential features. We used LogisticRegressionCV with 5-fold cross validation on the training data to identify the best C value [3].

Support Vector Machine

Model Implementation: As a comparative model, SVM, was implemented using the SVC function from sklearn [6]. Initial experiments with SVM involved exploring various kernel functions and hyperparameters to understand their influence on the model's performance.

Hyperparameter Tuning: A comprehensive approach was undertaken for hyperparameter tuning in SVM. The primary hyperparameters included mostly the regularization parameter (C), the kernel type, and the kernel coefficient (gamma). The exploration encompassed multiple kernel

functions, including Radial Basis Function (RBF), Polynomial (Poly), Linear, and Sigmoid. For the RBF and the Sigmoid kernel, both C and gamma were systematically varied, while the Polynomial kernel also involved tuning the degree of the polynomial. For the linear kernel only the C hyperparameter was used.

A grid search technique, using GridSearchCV from sklearn, was employed to evaluate the model's performance across a predefined parameter grid [7]. This grid search assessed a plethora of combinations of C, gamma, and kernel types. The most influential hyperparameters were found to be C and gamma, particularly for the RBF kernel. These parameters were instrumental in defining the SVM's decision boundaries and balancing the bias-variance trade-off. These values were selected based on their ability to maximize the mean test score from the cross-validation data, indicating their effectiveness in enhancing the SVM's generalization capabilities.

Analysis of Experimental Results

Feature Analysis

Ink Feature

The experiment's first step examined the ‘ink’ feature. This was employed to train a multinomial logistic regression model to assess its effectiveness in classifying digits. The model using the ‘ink’ feature achieved an accuracy of 22%.

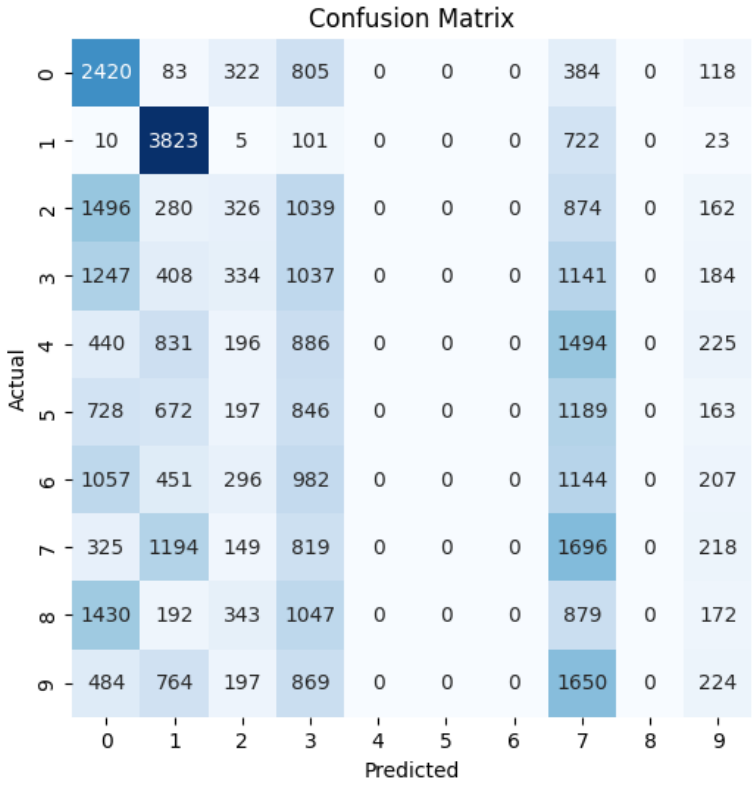

Figure 1: Confusion Matrix using Logistic Regression Model with 'ink' as the only feature.

As seen in Figure 1, the confusion matrix illustrates how well the model can distinguish between the different digit classes. Focus was given on the misclassifications between the digits '1' and '8', '3' and '8'. The confusion matrix indicates that the model is relatively adept at identifying the digit '1', with a high rate of correct classifications (3823 out of 4184 instances). This suggests that for digits like '1', which should have a lower pixel intensity, 'ink cost' is a more distinguishing feature. However, the model's performance is not uniformly effective among other digits.

A significant challenge is evident with the digits '4', '5', '6', '8', where the model's accuracy is extremely low at 0 correct classifications, highlighting a critical shortcoming in using 'ink cost' as the sole feature. For example, the digit '8' often gets confused with '0', '3', and '7', suggesting that these digits

have similar total pixel intensities. Similarly, the model also struggles significantly with the digit ‘3’, exhibiting a high rate of misclassification as various other digits. This points to the inability of 'ink cost' in distinguishing digits that share similar ink costs but differ in shape and structural characteristics.

These observations underline a broader limitation of relying solely on a single feature like 'ink cost' for complex classification tasks such as digit recognition. While this feature offers some level of discriminatory power for certain simpler digits, it is not sufficient in accurately classifying more complex ones. This limitation suggests that a more effective procedure in incorporating additional features that capture the shape and spatial distribution.

Quadrant Density Feature

The new approach was adopted by segmenting each digit image into four quarters and calculating the pixel densities in each quarter. A logistic regression model was subsequently developed, utilizing the quadrant density features. To maintain consistency in feature magnitude across the model, these features were normalized using the scale function. The model using the Quadrant Density feature achieved an accuracy of 46%.

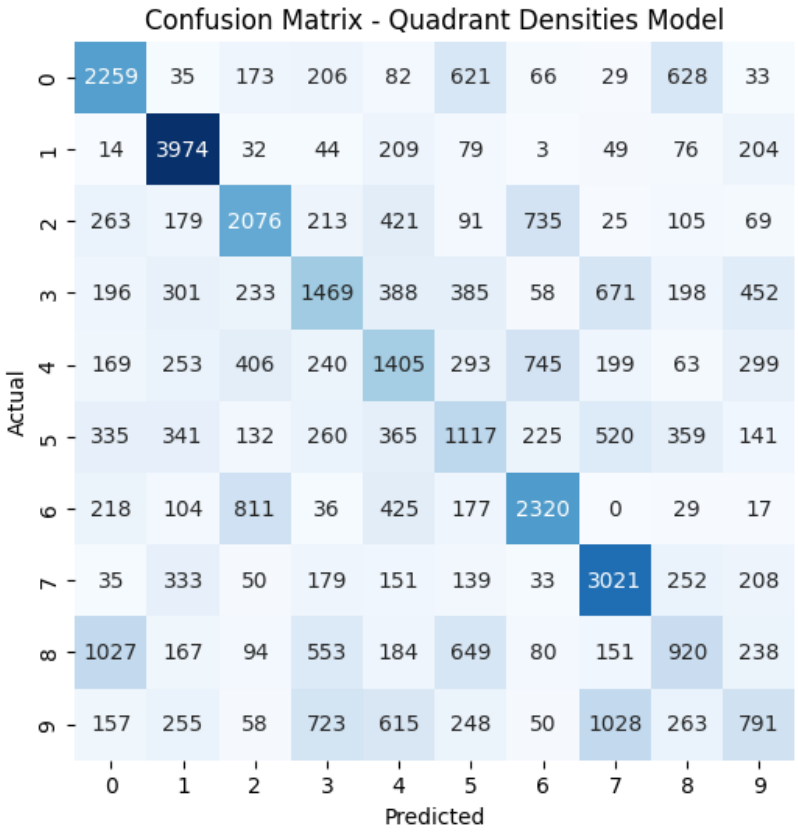

Figure 2: Confusion Matrix – Quadrant Density feature method

Comparing the model using this new approach with the simplified total ink feature, notable improvements in certain areas are observed, along with persistent challenges in others as seen in the Confusion Matrix in Figure 2.

The Quadrant Density feature method provides a more distinct set of features compared to the total ink model, capturing not just the amount of ink used but its distribution across the digit in four non overlapping areas. This added detail seems to improve the classification for digits like '0' and '1', which have distinct ink patterns that are well represented by quadrant densities.

For instance, the total ink model struggled significantly with the digit '8', often confusing it with '0', '3', and '7' due to similar total ink costs. While the Quadrant Densities feature method still

misclassifies '8' frequently, it appears to make fewer mistakes among these specific digits, showing a better understanding of their spatial distribution.

Despite these improvements, the model above continues to face difficulties with complex digits like '8', as well as with distinguishing between certain pairs like '2' and '6', '3' and '9'. These challenges point to the limitations of using Quadrant Density feature alone, as they still do not capture the intricate structures and subtleties that differentiate these digits.

Combination of Features

The quadrant densities feature was used in addition to the total amount of ink. The confusion matrix of the two features combined resulted in the exact same results as seen in Figure 2. This is due to the fact that the total amount of ink is directly derived from the sum of those 4 values, therefore making it insignificant in the training phase.

Comparison of Features

When comparing the two features, it's clear that the Quadrant Density feature is an advancement over the total ‘ink’ feature, as it incorporates more information about the shape of the digits as shown by the difference in overall accuracy which is 46% in comparison to 22% when using the ‘ink’ feature. However, the fact that both features struggle with similar digits indicates that additional, more intricate features are required to effectively distinguish between all ten digits.

In conclusion, while the use of quadrant densities as features provides a more detailed representation of the digits compared to total ‘ink’, and subsequently improves classification for some digits, it is not a complete solution. The comparison underlines the need for a more comprehensive approach to feature selection in digit classification, using more features to capture the full complexity of the digit shapes and arrangements.

Models Performance using raw pixel values

Regularized multinomial logit model

In our analysis of the MNIST dataset, one of the models used for the classification of the dataset is the regularized multinomial logit model. The dataset comprised 42,000 samples, of which 5,000 were utilized for training and the remaining 37,000 for testing. This division ensured a robust evaluation of the model's performance on unseen data. The multinomial logit model used the LASSO penalty (L1) and the 'liblinear' solver [5].

The training phase involved an essential step of hyperparameter tuning, where 5-fold cross-validation was implemented to identify the most effective hyperparameters [3]. The optimal C value, determined to be 0.00625, was chosen based on its ability to balance the model complexity and the training accuracy after testing 50 different C values from 10^{-4} to 10^{4} following a logarithmic scale, achieving the highest mean cross validation score. This value indicates a moderate level of regularization, preventing overfitting while maintaining a good fit to the training data.

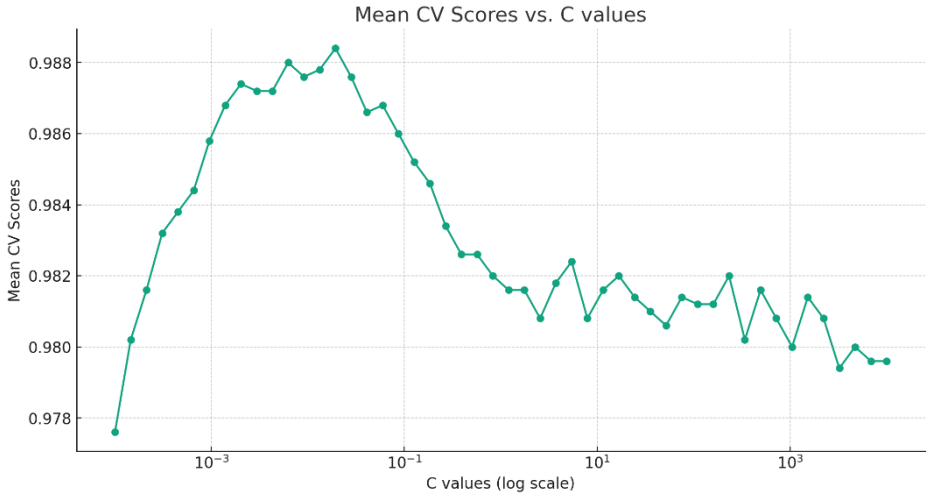

Figure 3: Mean CV Scores vs. C values

Upon evaluating the model's performance on the test dataset, it achieved an accuracy of 89,98%. The insights from the model's performance were further deepened by analyzing the confusion matrix as seen in Figure 4. This analysis revealed specific patterns in the model's predictions, such as certain digits that were more prone to misclassification than others (e.g., digit 8). The confusion matrix provided a detailed view of the model's strengths and weaknesses in classifying each digit, highlighting areas where the model could be improved.

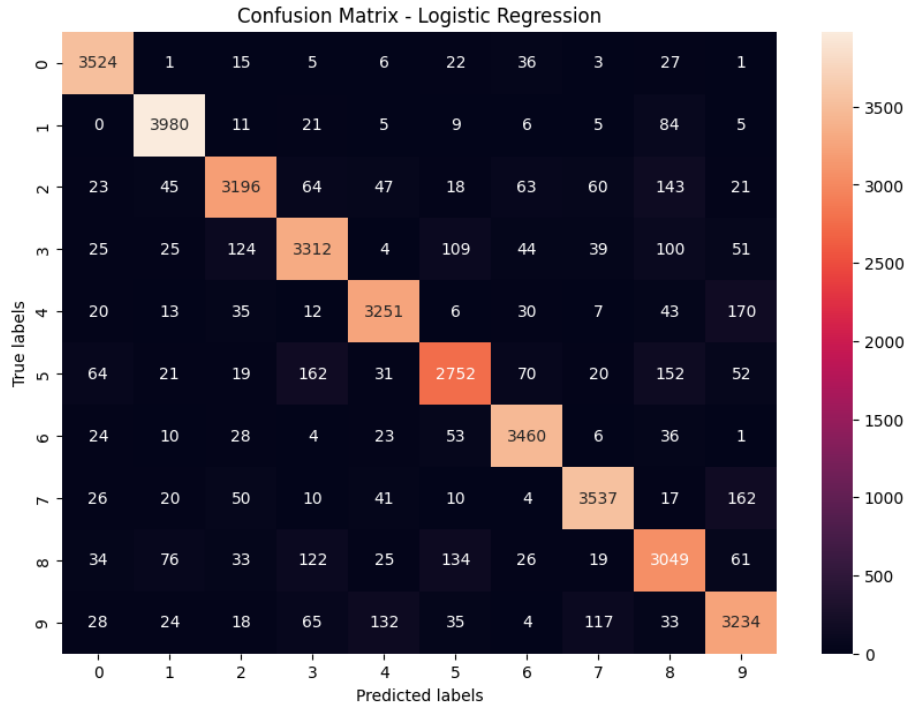

Figure 4: Confusion Matrix – Logistic Regression

In conclusion, the Regularized Multinomial Logistic Regression model demonstrated strong performance on the MNIST dataset, with its high accuracy and the detailed insights provided by the confusion matrix. Through the analysis, the best C hyperparameter was obtained achieving the high accuracy of 89,98% on the model when evaluated on the test data.

Support Vector Machine

A Support Vector Machine (SVM) was employed with 4 different kernels. This model was tasked with the same objective of classifying handwritten digits. The dataset, consistent with the one used for the logistic regression model, consisted of 42,000 samples, split into 5,000 for training and 37,000 for testing. This division was crucial in assessing the model's ability to generalize to new, unseen data.

For the SVM with an RBF kernel, the training phase was critically dependent on the optimization of hyperparameters. Through the process of 5-fold cross-validation, the optimal hyperparameters for our model were identified. The parameter C, a regularization parameter in SVM, was found to be 24.42. This value signifies an optimal trade-off between model complexity and training data fit, ensuring a good generalization capability without overfitting. Alongside C, the gamma parameter was fine-tuned, which controls the influence of individual training samples on the decision boundary. The optimal gamma was found to be approximately 1e-06, indicating a very precise and localized influence of the training samples.

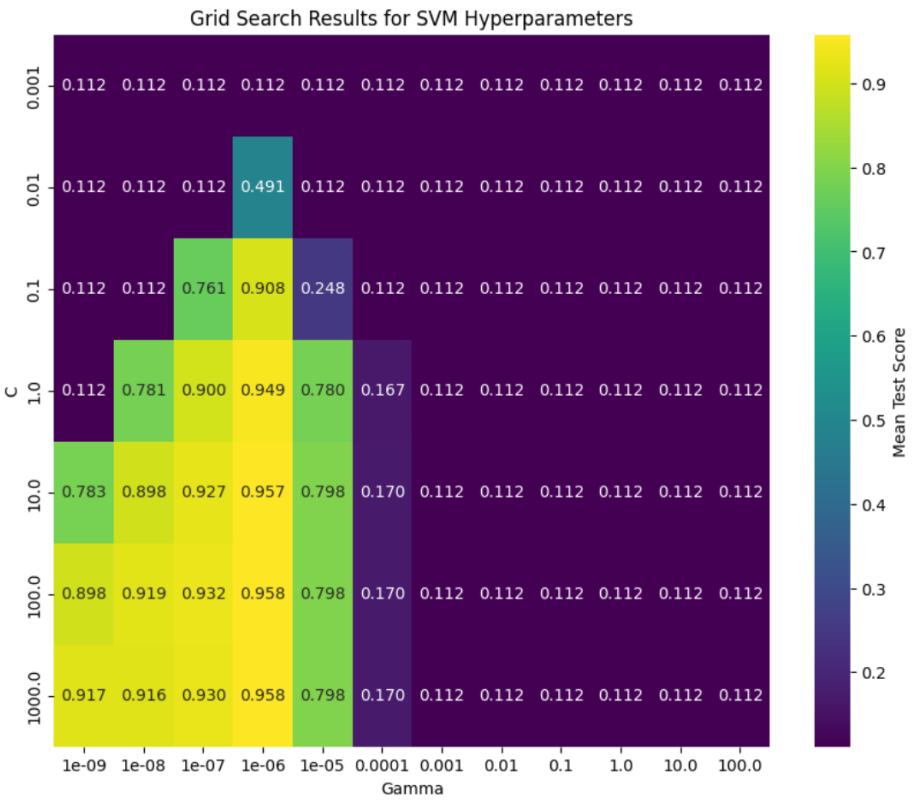

Figure 5: Grid search results for SVM (RBF) hyperparameters 'C' and 'Gamma'

Evaluating the SVM model with an RBF kernel on the test dataset revealed an impressive accuracy of 95.87% that outperformed the other kernels, as shown in Table 1. This high accuracy rate is particularly notable given the diverse range of handwriting styles present in the MNIST dataset. It suggests that the model, with its RBF kernel, is remarkably efficient in distinguishing between the different digits.

Table 1

Support Vector Machine Results

Kernel/Hyperparameters	C	Gamma	Degree	Accuracy Scores on Test set
RBF	23.4	1e-6	-	0.96
Poly	0.1	1e-6	2	0.93
Sigmoid	0.1	1e-6	-	0.86
Linear	1e-6	-	-	0.91

Further insights into the model's performance are gained through the analysis of the confusion matrix as seen in Figure 6. The confusion matrix unveils details about the classification accuracy across different digits. It highlights specific instances where the model excels or underperforms that might be prone to misclassification, providing a deeper understanding of the model's strengths and weaknesses. In the model, it is observed that there isn't a specific digit class that underperforms since the misclassifications are mostly distributed equally across all digit classes.

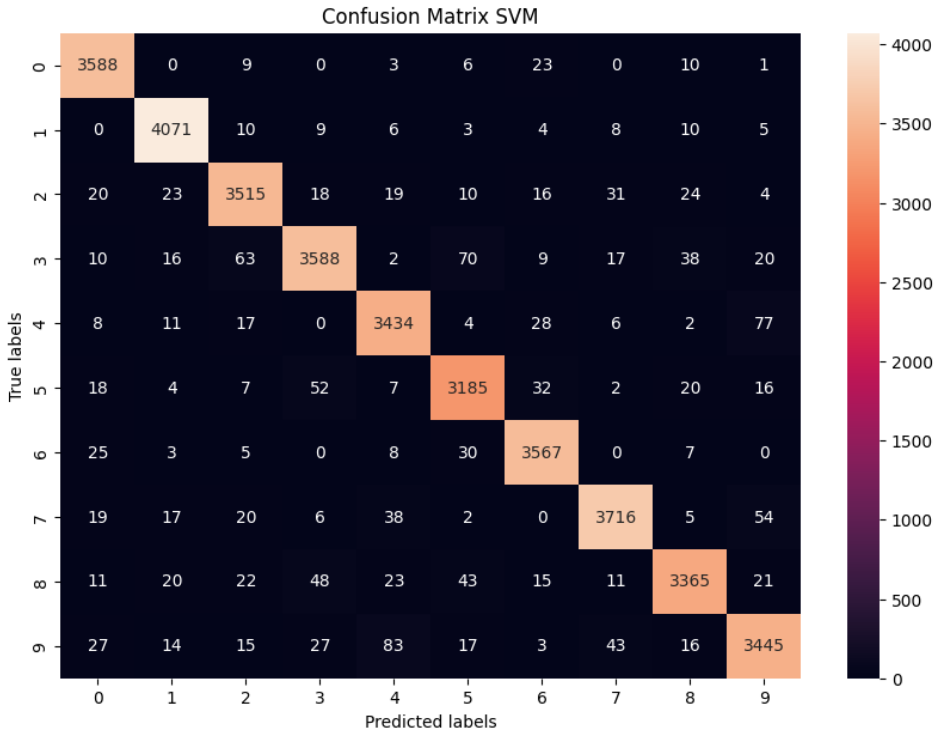

Figure 6: Confusion Matrix SVM (RBF)

In summary, the Support Vector Machine with an RBF kernel proved to be highly effective in the classification of the MNIST dataset, as evidenced by its high accuracy rate of 95.87%. The selection of the best kernel (RBF), along with the optimal C and gamma parameters, played a crucial role in achieving this level of performance.

Comparison of Classification Methods

Model Comparison

In the Model Comparison subsection of the analysis, focus is given on contrasting the performance of the multinomial logit model and the support vector machine (SVM) on the MNIST dataset.

A key aspect of the comparison is the accuracy of each model. The multinomial logit model achieved an accuracy of 89.98%. However, the SVM model outperformed it with an impressive accuracy

of 95.87%. This higher accuracy of the SVM suggests a better performance in classifying the digits classes of the MNIST dataset. To evaluate the models' performance, the 2x2 contingency table was constructed, showing the instances where each model's predictions diverged:

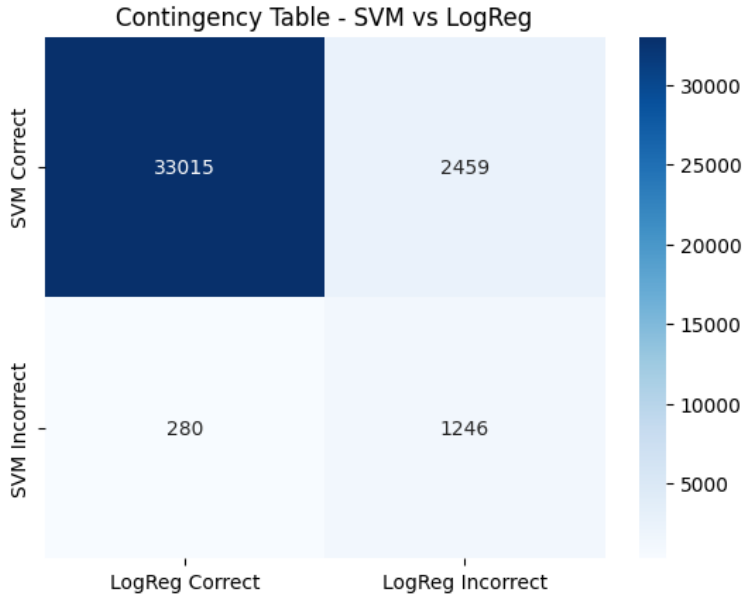

Figure 7: Contingency Table – SVM vs Logistic Regression

In Figure 7, 2459 represents the number of instances where the SVM correctly identified digits that the multinomial logit model misclassified. This number suggests that the SVM has a distinct advantage in correctly classifying certain types of digits that the multinomial logit model struggles with.

Conversely, the 280 instances represent scenarios where the multinomial logit model was successful in classification, but the SVM failed. While notably fewer than the instances favoring the SVM, this still provides important insights. It suggests that there are specific scenarios or digit representations where the multinomial logit model has an edge.

The 1246 instances where both classifiers failed to correctly identify the digits highlight the inherent challenges in the MNIST dataset. These could be particularly complex digits to classify, posing a challenge to both models.

Overall, this breakdown of the contingency table in Figure 7 demonstrates the difference in performance of each model. The SVM shows a clear overall advantage in accuracy and in handling a broader range of digit classifications even though there are a few instances where the multinomial logit model managed to classify correctly and the SVM did not.

Statistical Test

In the comparative analysis of the multinomial logit model and the support vector machine (SVM) on the MNIST dataset, the McNemar was employed test to assess the differences in their performance [8]. This test is particularly suitable for the specific purpose as it is designed to evaluate two correlated, binary outcomes – in this case, the classification results of the models' accuracies are on the same data. The McNemar test is instrumental in determining whether the two models differ significantly in their error patterns.

The test is based on a 2x2 contingency table as seen in the previous subsection that records dissimilar pairs – instances where one model correctly classifies a digit, and the other does not. This table forms the basis for computing the chi-square statistic, which in the analysis yielded a value of 1731.90. Such a high value suggests a substantial disparity in the models' performances.

For the test, the null hypothesis (H_0) was set to state that there is no significant difference in the performance of the multinomial logit model and the SVM. The alternative hypothesis (H_1) proposes that a significant difference exists. Given the extremely low p-value, almost nearing zero, the null hypothesis can be confidently rejected. This rejection implies a statistically significant difference in the performance of the two models.

In choosing our significance level (α), it was opted for a more conservative threshold of 0.01, given the critical nature of accurate digit classification in the MNIST dataset. This lower α reduces the

likelihood of a Type I error – incorrectly rejecting a true null hypothesis – thereby ensuring a more rigorous standard in the findings.

In conclusion, the pronouncedly high McNemar test statistic, coupled with the extremely low p-value, strongly indicates a significant difference in the error patterns between the logistic regression model and the SVM. However, it's important to contextualize this result within the framework of the MNIST dataset. A significant statistical difference does not necessarily imply overall superiority of one model over the other but rather highlights their distinct classification strengths and weaknesses. With the multinomial logit model achieving an accuracy of 89.98% and the SVM outperforming at 95.87%, this difference in accuracy further emphasizes the SVM's efficiency in this specific task.

Conclusion and Discussion

In this study, we performed an exploratory analysis on the MNIST dataset while also looking into evaluating the performance of using different features such as ‘ink’ and ‘quadrant densities’. Furthermore, we applied and evaluated two machine learning models: the regularized multinomial logit model with LASSO penalty and the Support Vector Machine (SVM) after we identified their best hyperparameters. Our comprehensive analysis led to the following key findings:

In our feature analysis, the logistic regression model using the 'ink' feature demonstrated notable limitations with an accuracy of only 22%. While it accurately classified simpler digits like '1', it faced difficulties with more complex digits such as '8', revealing the limitations of 'ink' as a solitary feature. The incorporation of quadrant density features, representing the spatial distribution of ink, offered improvements in classification, especially for digits with distinct spatial patterns achieving an accuracy of 46%. However, the model continued to struggle with more intricate digit forms, suggesting a need for further feature enhancement. A combination of the two features did not result in any

improvement in the classification of the MNIST dataset since the ‘ink’ feature was overshadowed by the quadrant density feature.

The performance of the regularized multinomial logit model with LASSO was noteworthy, achieving an accuracy of 89.98% on the test dataset. This success can be attributed to the optimal selection of hyperparameters, particularly the regularization strength (C value = 0.00625).

In contrast, the SVM, especially with the RBF kernel, achieved an accuracy of 95.87%. This performance was the result of precise hyperparameter tuning, with an optimal balance of complexity and fit to the training data, achieved with a C value of 24.42 and gamma of 1e-06.

Lastly, the McNemar test, conducted to compare the performances of the two models, yielded a test statistic of 1731.90, indicating a statistically significant difference in their classification accuracies. This high test statistic led us to confidently reject the null hypothesis, affirming that the two models indeed have a significant difference in performance. This finding further validated the SVM's higher performance in this context, highlighting its efficiency and consistency in digit classification on the MNIST dataset. The statistical test underscores the distinct advantages of the SVM, particularly in handling the diverse and complex patterns inherent in the dataset.

References

[1] scikit-learn developers. scikit-learn: Machine Learning in Python. Retrieved [insert date you accessed the site], from https://scikit-learn.org/stable/

[2] scikit-learn developers. sklearn.model_selection.train_test_split. In scikit-learn: Machine Learning in Python. Retrieved [insert date you accessed the site], from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[3] scikit-learn developers. Cross-validation: evaluating estimator performance. In scikit-learn: Machine Learning in Python. Retrieved from https://scikit-learn.org/stable/modules/cross_validation.html

[4] OpenCV developers .In OpenCV 4.x documentation. Retrieved from https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d

[5] scikit-learn developers. sklearn.linear_model.LogisticRegression. In scikit-learn: Machine Learning in Python. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[6] scikit-learn developers sklearn.svm.SVC. In scikit-learn: Machine Learning in Python. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[7] scikit-learn developers sklearn.model_selection.GridSearchCV. In scikit-learn: Machine Learning in Python. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[8] statsmodels developersstatsmodels.stats.contingency_tables.mcnemar. In statsmodels: Statistics in Python. Retrieved from https://www.statsmodels.org/dev/generated/statsmodels.stats.contingency_tables.mcnemar.html