Handwritten digit recognition using Logistic Regression and Support Vector Machines

Michail Panagiotis Bofos (4325575)
Tiago dos Santos Silva Peixoto Carriço (2311143)
Panagiotis Ioannis Dallas (97874337)

22 November 2023

Abstract

This report describes the task of recognizing handwritten digits provided by the MNIST Dataset. The task that is performed is analyzing a data set of handwritten digits and determining what digit from 0-9 is written. In the first part of our analysis, we extract some features from the raw data and determine the optimal feature. To do so, a multinomial Logistic Regression model is fitted with each feature. In the second part of the analysis, the raw pixel values are separated into 784 features. Two models are implemented and fine tuned; a multinomial logistic model with LASSO penalty (L1) and a Support Vector Machine model.

1 Introduction

Handwritten digit recognition is a common machine learning problem. There are many applications that a digit recognition module can be employed, like mail sorting for postal services, digitisation of documents, and recognition of handwritten digits on bank checks. There are many different approaches that try to solve this problem, and most of them focus on image level features extracted by a Convolutional Neural Network (CNN) [1, 5]. The aim of our reasearch is to explore various features, and test their performance on simple machine learning algorithms. To achieve this we utilized the MNIST Dataset [2], a dataset that contains handwritten digits from 0 to 9. In order to properly describe the task of handwritten digit recognition we decided firstly to investigate what features can be proven useful. We came up with four features; the first one is the amount of ink per digit, the second one is the regional spread per digit, the third is the combination of previous two, the fourth is the raw pixel values as a feature. In Section 2.3 we demonstrate the motivation behind those features and how these features were constructed. To find the best feature, we first extracted the features from the raw data and then we then fitted a multinomial Logistic Regression model to test their performance. Upon completion of the feature selection process, we then focused on finding the best model for that feature. We considered two models, a Logistic Regression model and an SVM model, which we fine tuned by training with many different configurations. Then, we performed a statistical analysis to determine the significance of the results of the two tuned models. Finally, we extended our analysis of the raw pixel values feature, by removing the unused pixel values during the data pre-processing.

2 Experimental Setup

2.1 Description of the data

The MNIST dataset consists of 42000 gray-scale hand-drawn images of digit from 0 through 9. The images used in the MNIST dataset have height and width equal to 28 pixels. Thus, each image can be represented as a vector of length equal to 784, with every value containing the information of one pixel. Each entry within the dataset consists of 785 columns. The initial column identifies the depicted digit, while the subsequent 784 columns represent the individual pixels comprising the image. In Figure 1 we see how we can map our features in order to recreate the original a 28 by 28 image. In Figure 2 we can see one sample of each digit.

```
000 001 002 003 ... 026 027
028 029 030 031 ... 054 055
056 057 058 059 ... 082 083
 |   |   |   | ...  |   |
728 729 730 731 ... 754 755
756 757 758 759 ... 782 783
```

Figure 1: Mapping of pixels in MNIST images

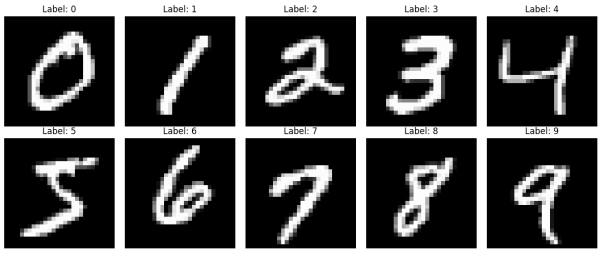

Figure 2: Sample of all labels

2.2 Exploratory analysis

In our efforts to understand the MNIST dataset we stumbled upon some very interesting observations.

2.2.1 Class Distribution

First, we decided to investigate the class distribution in our dataset. Ideally we would assume that each class is equally distributed inside our dataset subsequently occupying approximately 10% of the overall data. In reality all of our classes are somewhat equally distributed with class '1' being the most common with 11.2% and class '5' being the least common class with a 9% presence in our dataset, as depicted in Figure 3. The fact that no specific class is dominant on the dataset indicates that a naive prediction algorithm that always predicts the majority class would not have an accuracy higher than around 10%. This observation is expected as we cannot under no specific circumstances support the hypothesis that a digit is more common that other digits.

2.2.2 Image Distribution

From Figure 2 we can easily observe that some areas of each image are not utilized. This prompted us to investigate if there are any pixels that could be omitted from the dataset. We decided to find the pixels that had standard deviation and mean value equal to zero throughout the dataset. Following these steps we came into the rather intuitive conclusion that the corner pixels of each image were rather unutilized. The results of of our investigation can be found on Figure 4.

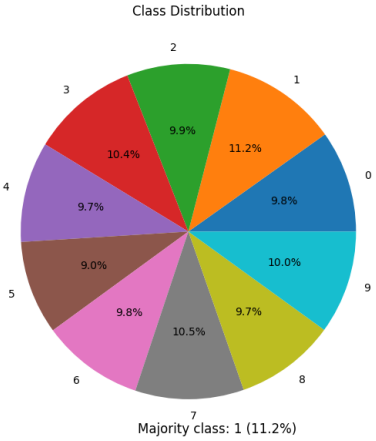

Figure 3: Class distribution percentages

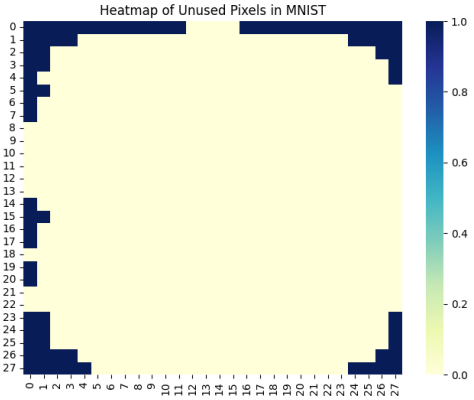

Figure 4: Unused pixels throughout dataset

2.2.3 Digit Variance

To be able to recognize handwritten digits effectively we should be sure that our dataset is diverse. To achieve this, we decided to investigate in what degree each digit from each class could be different from other same-class digits. In Figure 5 we see how all instances of each class put together would look like.

We can observe that our digits can still be easily recognized by a human despite the fact that there are some blurry areas that indicate a plethora of differently shaped digits.

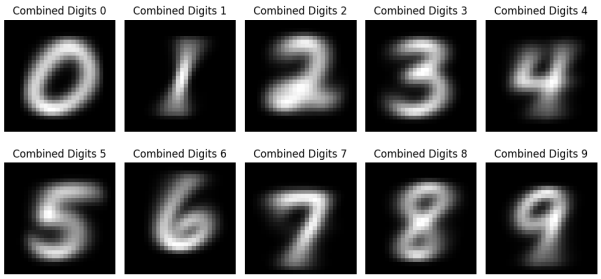

Figure 5: All instances of each class combined

2.3 Feature engineering

In any analytical endeavor, the selection and understanding of features play a pivotal role in interpreting the outcome and the results of the study. Features, also known as variables or attributes, encapsulate the essential characteristics of the data. In this section, we delve into a comprehensive exploration of the diverse features harnessed for our analysis. These features are; Ink feature, Region feature, and Raw pixel values as a feature. We aim to illuminate the diverse nature of our dataset and underscore the significance of feature engineering in extracting meaningful information.

2.3.1 Ink feature

The initial feature extracted from the data is the Ink feature, which characterizes the amount of ink utilized in creating a handwritten digit image. The reasoning behind this feature selection is that the ink that is required to represent distinct digits varies (e.g. Digit-0 requires more ink than Digit-1).

In Figure 6, the distribution of the ink means and the ink standard deviation per digit is shown. It is evident that the mean of 1 is lower than the other means. In addition, it can be seen that the standard deviation of digit 1 is lower than the standard deviations of the other digits. Low standard deviation means data are clustered around the mean, while high standard deviation indicates data are more spread out.

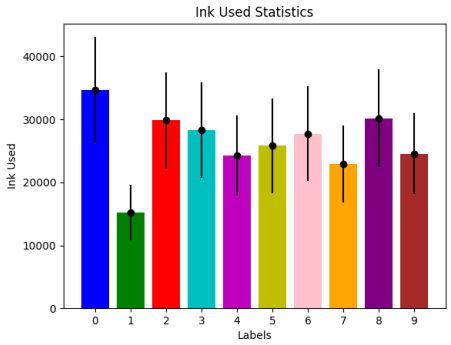

Figure 6: Mean vs. Standard Deviation per digit

This difference in mean and the low standard deviation of digit 1 and the fact that they do not overlap with other digit's ink metrics, can potentially play a significant role when a classifier tries to predict digits of class 1. Due to these statistics and the fact that digit 1 is easily distinguishable from the rest of the digits, digit 1 is expected to be classified well. However, other digits are expected to be harder to classify, since their mean values and standard deviation are similar to one another.

A multinomial logistic regression model is fitted with this single feature, using the sklearn library [4]. The trained model is used to predict handwritten digits. The ink feature is scaled to have zero mean and has unit standard deviation. The performance of the model will be extensively discussed in Section 3.1.1.

2.3.2 Region feature

To improve the classification performance of the ink feature, we extracted the regional feature. This feature is shown in Figure 7. It separates the image into four regions: North West (NW), North East (NE), South West (SW) and South East (SE). The regions are separated in a grid of 14x14 pixels.

For all four regions of every image, the average ink is calculated. This feature was chosen as each digit spreads differently across the entire image and we believe that by having such a feature, the models are able to distinguish each digit better than with the ink feature. This comparison is further discussed in Section 3.1.2, as the region feature is also fitted in a Logistic regression model, and it is compared with the ink feature on several evaluation metrics.

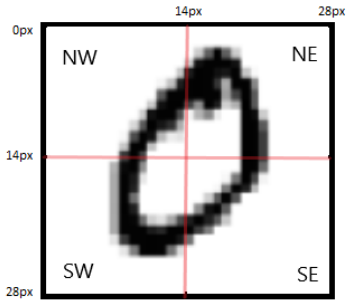

Figure 7: Image regions

In Figure 8, the mean per digit per region is presented. The digit six, based on the figure, is expected to have more ink in the SW and SE regions, while a zero digit might have more ink in the NE and SW regions.

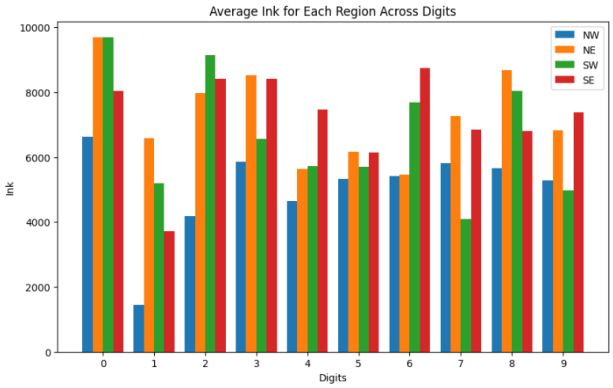

Figure 8: Average ink per region per digit

2.3.3 Raw pixel values as a feature

One of the most straightforward features is to use the pixel values of the image as features. Each pixel's intensity represents the darkness or lightness of that pixel. In the first part of the analysis, the raw pixel values is separated into 784 features and are fitted in a Logistic Regression model, to compare its performance against the ink and the regional feature.

In the second part of the analysis, we use the same 784 features but as the goal of this part is to examine the performance of the models, different sets are used for training, validation and testing purposes. To be more specific, the MNIST dataset is loaded into a panda's [6] dataframe, then with the use of sklearn's train_test_split method, the dataset is split into a training dataset that consists of 5000-digit images and a testing dataset that consists of 37000-digit images. In order to ensure the reproducibility of this analysis and fairness between the analyzed models, a specific seed of 1 is set for the random state parameter. Two models are implemented: a multinomial logistic model with LASSO penalty (L1) and a Support Vector Machine model. A short description of the libraries used in this part of the analysis along with its parameters (for reproducibility) is shown in Appendix table 4.

3 Experimental results

3.1 Feature performance

In this section, we compare and discuss the performance results of each feature discussed in Section 2.3. The evaluation is conducted by employing a Logistic Regression model, wherein the dataset is fitted under uniform configurations to ensure that the comparison is fair. To be more specific, in this part of the analysis we use the complete data set both for training and evaluation.

3.1.1 Ink feature performance

The Logistic Regression model fitted with a single feature, the ink in this case, achieves an accuracy of 22.68%. In Figure 9, the confusion matrix illustrates the model's prediction capability, as it presents the model's strengths and weaknesses per class. By looking at the confusion matrix, it is evident that the model predicts the digit 1 better than any other digit, with an accuracy score of 82%. This result validates the assumption that we made in Section 2.3.1, that the mean and the standard deviation of digit's 1 ink is significantly lower that the rest of the digits, and therefore the model is able to distinguish that digit better. Similarly, the model predicts relatively well the digit 0 with an accuracy of 59%, which has the larger ink mean. In addition, the model does not perform well when it comes to the predictions of digits 2-9, as it makes a lot of prediction errors. Surprisingly, the digits 4,5,6 and 8 are not predicted at all, but instead the model confuses them mostly with digits 3 and 7. We believe that this is also related to the similar mean ink value of those digits and the standard deviation, as illustrated in Figure 6.

The results seem to verify all of our initial assumptions about the issues that the ink feature has, and that is what lead us to investigate other features, like

the region feature, as we want to make a model that is able to predict well all of the digits, and not only 0 and 1.

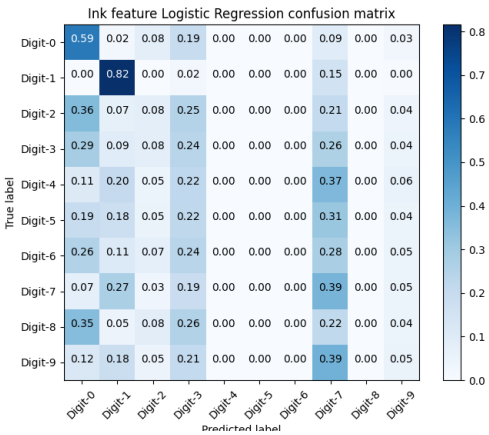

Figure 9: Ink feature confusion matrix

3.1.2 Region feature performance

The Logistic Regression model fitted only with the regional feature achieves an accuracy score of 46%. Compared to the Logistic Regression model that was trained only with the ink feature, the accuracy score increases by a factor of 2. Although the overall accuracy has increased, it is important to compare digit-to-digit performance of each model, as the overall accuracy score may be significantly higher for only 1-2 digits. In Figure 10, the confusion matrix illustrates the model's prediction capability, as it presents the model's strengths and weaknesses per digit. By looking at the confusion matrix, it is noticeable that the model makes better predictions with the regional feature, especially for the digits 4,5,6 and 8 that were not predicted at all by the model with only the ink feature. The only digit that its performance dropped by the regional feature, is the digit 0 which is probably caused by the mean ink in the regions NW and SE, which is similar to other digits, as shown in Figure 8. Additionally, the model does not perform well for digits 8 and 9, as it confuses 25% of the times digit 8 with digit 0, and digit 9 with digit 7. We believe that this is also caused by the spread of those pairs in each region.

This result suggests that our feature selection in this case was successful, as the model is able to predict better in almost all of the classes, and therefore by fitting a more detailed feature, like the regional ink in each region the model is able to distinguish better each digit.

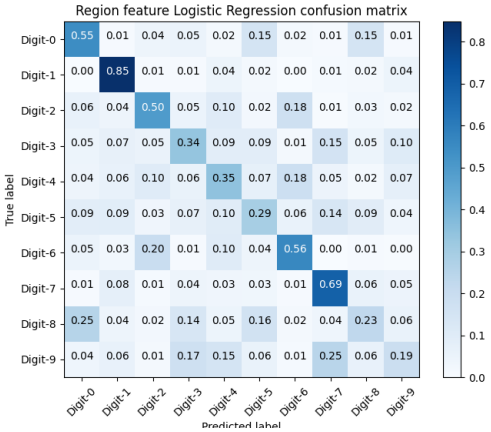

Figure 10: Region feature confusion matrix

3.1.3 Combined feature performance

The regional feature fitted in a Logistic Regression model proved to be a step towards the right direction, as described in the previous section. In an effort to find the best feature given the MNIST dataset, we investigated the performance of a Logistic Regression model fitted with the combination of the ink and the regional features. The Logistic Regression model trained on both features, achieved an accuracy score of 46% and the confusion matrix is shown in Figure 11.

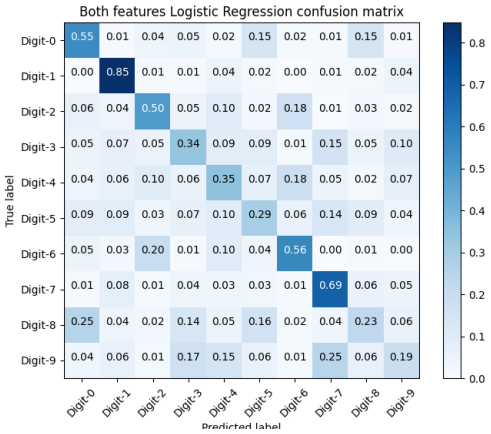

Figure 11: Combined feature confusion matrix

It is evident that the model performs identically to

the single-feature regional Logistic Regression model, as the confusion matrices are the same. There are minor differences in some of the predictions, and these can be found in the classification reports for each model in the Appendix. We believe that the model performs similarly for two reasons. Firstly, the ink feature and the regional feature are related, as we described in Sections 2.3.1 and 2.3.2. Secondly, although we normalised the data upon concatenation, the region feature has 4 data points for each digit (1 for each region) while the ink feature has only 1. One possible solution to this would be to use different weights for the features, but we decided not to do so as the performance would probably match the performance of the ink feature.

3.1.4 Raw pixel values as a feature performance

The Logistic Regression model fitted with the raw pixel values as a feature, achieves an accuracy score of 93%. Compared to the Logistic Regression model that was trained with the regional feature, the accuracy score increases by a factor of 2. The confusion matrix of the model is shown in Figure 12. The model performs significantly better than any of the previous models, as it is able to predict correctly 90% of the time or more each digit. Although these results may seem surprisingly well, it is important to highlight that the model has probably overfitted, as we use all of the raw pixel values for both training and testing at this point of our research.

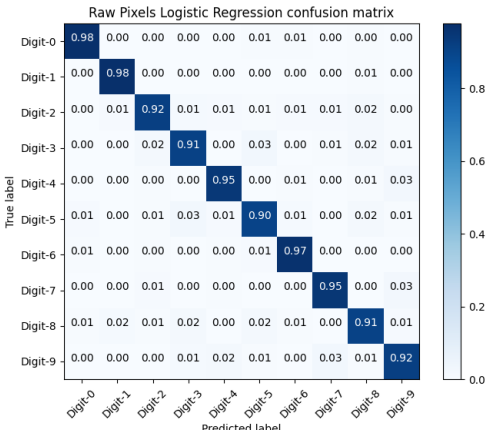

Figure 12: Raw pixel values as a feature confusion matrix

A more complex Logistic Regression model that is fitted with the raw pixel values as a feature is thoroughly reviewed in the following sections, but with different configuration settings and data preprocessing techniques applied.

3.2 Further investigation on raw pixel values

In the previous section, we examined the performance of different features on a simple Logistic Regression model. Our research indicates that the best feature is the raw pixel values, as it performs better accross all labels. Therefore, in this section, we shift our focus from feature selection, and investigate different models and data pre-processing teqnuiques for the raw pixel values feature.

3.2.1 Model definition

The models that are analyzed in this part of the analysis are a Logistic Regression model (with Lasso penalty) and a Support Vector Machine model. To define both of these models and instantiate them, LogisticRegression() and SVC() methods are used from sklearn's library. Each of these model implementations provides a sufficient number of hyper-parameters that can be tuned and therefore provide agility. A short description of these parameters is shown in tables 1 and 2.

LogisticRegression() parameters		
Parameter	Value	Description
C	0.01, 0.1, 1, 10, 100	Inverse of regularization strength
solver	liblinear	Algorithm to use in the optimization problem
max_iter	300	Maximum number of iterations taken for the solvers to converge
penalty	l1	Lasso penalty

Table 1: LogisticRegression parameters used.

As shown in table 1, for the Logistic Regression

model multiple values of C and max_iter parameters are used. C values are normally between 1 and 10, increasing the value improves the classification accuracy (or reduces the regression error) for the training data, but this can also lead to overfitting. By defining multiple models with those values, not only the impact of the inverse regularization strength is shown but also how the solvers can converge by increasing the maximum iterations. The penalty is set to l1 for all models to apply LASSO penalty and the solver is liblinear as it is the one that supports the l1 penalty.

SVC() parameters		
Parameter	Value	Description
C	0.01, 0.1, 1, 5, 10	Inverse of regularization strength
gamma	0.001	Kernel coefficient for 'rfb' and 'poly'
kernel	rfb	Specifies the kernel type to be used in the algorithm

Table 2: SVC parameters used.

Following the same logic as in the Logistic Regression models, multiple C values are used for the Support Vector Machine models, as shown in table 2. Different gamma values are also used in order to check the fitting of the models. By having low gamma values, overfitting is prevented. RBF is the default kernel used within the sklearn's SVM classification algorithm, therefore this is used.

3.2.2 Model selection and parameter tuning

In order to find the best parameter settings for the selected models, sklearn's function GridSearchCV() is used before fitting each model with the training data. By using this function, the parameters of the estimators (Logistic regression and Support Vector Machine) used to apply these methods, are optimized by cross-validated grid-search over a parameter grid. Moreover, GridSearchCV() implements common estimator methods and as a result, training results can be extracted. As the default behavior (refit = True) of GridSearchCV(), after fitting an estimator's model for the given parameter set, the function then finds the best-parameterized model and refits that model across all training data. In Table 3 and 4, the mean validation error of all folds for each model is presented as extracted from GridSearchCV results.

3.2.3 Best performing models

As discussed in Section 2.3.3, to train, validate and test the models we used the same random seeding to ensure that the splits are identical. Therefore, a fair comparison between the best performing models can be made. This can be validated by counting the total occurrences of each digit in Figures 13 & 14. In those figures, the performance of each tuned model is analysed per digit. It is noticeable that the tuned SVM model outperforms the tuned Logistic Regression model, as for every digit the SVM model is able to make better predictions to the new, unseen data. This result suggests that the tuned SVM model is able to generalise better than the tuned Logistic Regrssion model. The overall accuracy score of the tuned Logistic Regression model is 89%, while the tuned SVM model scores 93%.

In Figures 15 & 16, the normalised confusion matrices show the accuracy of both tuned models for every digit separately and also each model's weaknesses are identified. Both models do not perform as well for digits 5 and 8, as they both struggle to distinguish 8 from 5 (4% of the times for the LR model, 3% for the SVM model). Similarly, both of the models do not perform well in predicting digit 5, as it is confused with digit 3 (5% of the times for the LR model, 4% for the SVM model).

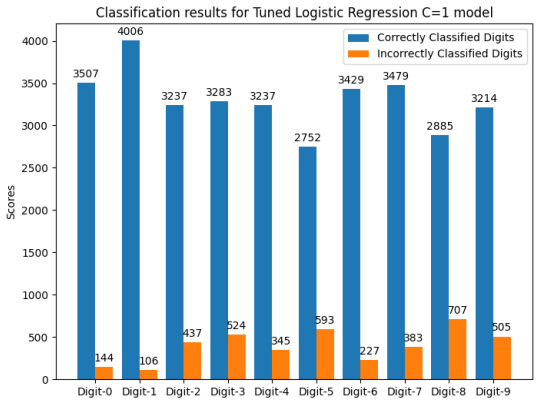

Figure 13: Classification report of the tuned LR model

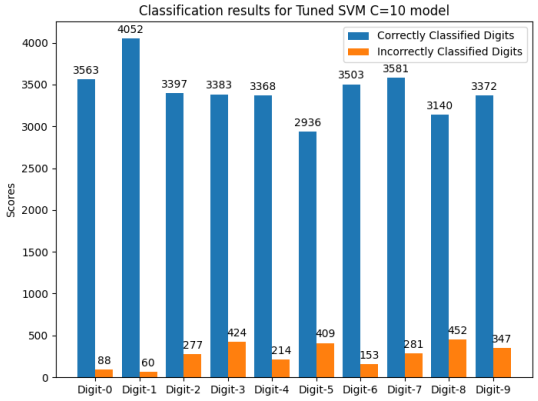

Figure 14: Classification report of the tuned SVM

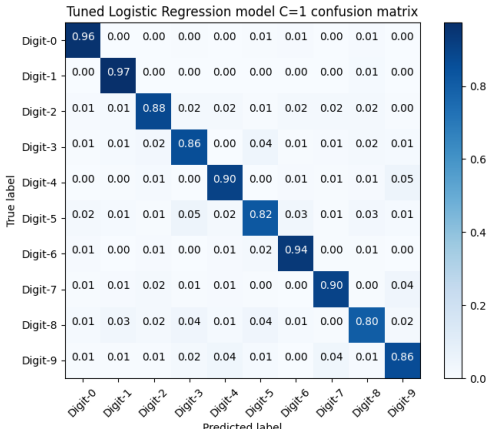

Figure 15: Confusion matrix of the tuned LR model

3.2.4 Statistical Analysis

Following the parameter tuning of each model we retrained our models using 80% of the original dataset for training and the remaining 20% for testing. At first glance the difference between the accuracy of SVM (95%) and Logistic Regression (87%) does not seem very significant. However, in order to confidently determine if there is a significant difference between the accuracies we have to perform a statistical analysis. There are various different tests that could be used. According to Dietterich [3] the choice of the best fitting test is relative to the computational cost of executing the learning algorithm. Dietterich suggests that the two most useful tests are the McNemar's test and the 5 by 2 cross-validations test. McNemar's test is more fitted for training algorithms that can be executed only once and 5 by 2 cross-validation test is preferred for training algorithms that can be executed multiple times.

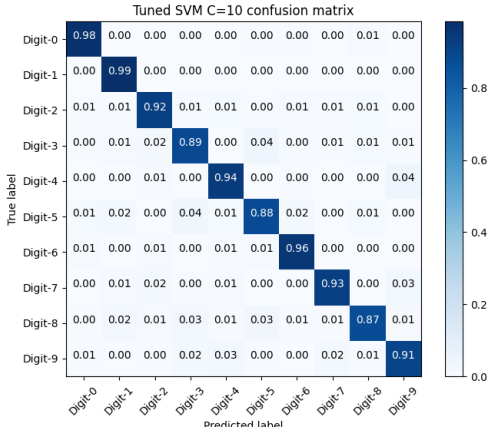

Figure 16: Confusion matrix of the tuned SVM model

Generally, the 5 by 2 cross-validation test is more powerful as it directly measures variation due to the choice of training set. In our case we decided to use the 5 by 2 cross-validation test, however due to the high computational cost we decided to eventually use McNemar's test.

3.2.5 McNemar's Test

In McNemar's test, we established a 2 by 2 contingency table to assess the correct and incorrect in predictions made by the two models. In order to perform this test we first normalized our data and split them to two parts (training set 80% and testing set 20% randomly using seed equal to 1) and trained both our models using our training set. Following that, we used our testing set to make predictions with both models. Finally, we created the McNemar table by comparing the predictions of the models and we calculated the p-value of our test by using the following equation, with n being the sum of predictions that were correctly predicted by one model and incorrectly by the other.

$$p = 2\sum_{i=b}^{n}\binom{n}{i}0.5^i(1-0.5)^{n-i}$$

In Figure 17 we observe that both SVM & Logistic Regression models make 7255 correct predic-

C value	0.01	0.1	1	5	10
Mean Validation Accuracy	0.6434	0.863	0.8882	0.855	0.8392
STD Validation Accuracy	0.0069455	0.01300769	0.01041921	0.01164474	0.01307517
Split	1	2	3	4	5

Table 3: Mean training error of 5 fold cross-validation, Logistic Regression models

C value	0.01	0.1	1	5	10
Mean Validation Accuracy	0.1144	0.682	0.8928	0.9152	0.9218
STD Validation Accuracy	0.0004899	0.00807465	0.0154971	0.01207311	0.00986712
Split	1	2	3	4	5

Table 4: Mean training error of 5 fold cross-validation, SVM models

tions and 376 incorrect predictions. Additionally, we can see that the SVM model correctly predicts 722 samples that the Logistic Regression model does not. Respectively, the Logistic Regression model can correctly predict 47 samples that the SVM model predict incorrectly. The extracted p-value of this matrix is equal to 1.73e-130, which is significantly lower than the usual significance threshold of 0.05, marking the difference between the accuracy of these two models significant.

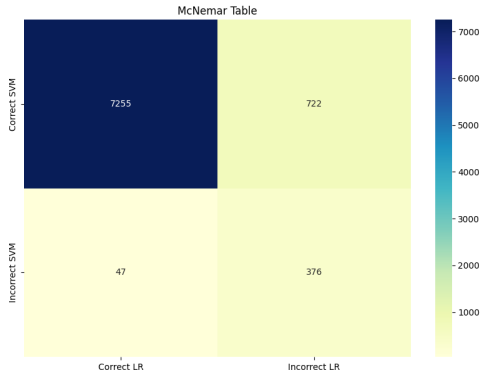

Figure 17: McNemar's table comparing the predictions of the best SVM model and the best Logistic Regression model

3.3 Further Exploration of Raw Pixel values feature

As discussed in Section 2.2.2 some of the pixels of the images were always unused. Based on that knowledge we decided to investigate the effect of removing these unused pixels from our data. To measure the effect of this removal, we decided to use the best performing model configurations for both SVM and Logistic Regression and again perform some McNemar's test to calculate the significance of our preprocessing. Firstly, we performed a McNemar's test between the SVM and Logistic Regression models that both utilized the edited dataset. The accuracy of both models was identical to the accuracy of the non-edited dataset, in Figure 18 we can see the McNemar's table for this test. The calculated p-value is equal to 2.28e-130, again classifying the difference between SVM and Logistic Regression significant for these dataset.

In continuation we decided to measure the significance between these two datasets for both models. First, we tested the SVM model that was trained with the complete dataset against the SVM model that was trained with the edited dataset. In Figure 19 we can see the McNemar's table for this test, as we can observe the two models performed identical predictions making the non-existant difference between their accuracy non-significant.

Second, we performed the same test for Logistic Regression. The results were again very similar but there were 4 cases that there was a mismatch between the two models as we see in Figure 20. The calculated

p-value for this test is equal to 0.62, marking the significance of the difference between the accuracy of these two models non-significant.

In conclusion, the removal of the unused image pixels does not create a significant effect on the performance of SVM or Logistic Regression models for the task at hand. One aspect that could be further investigated would be the execution time difference, however, this aspect is beyond the scope of our analysis.

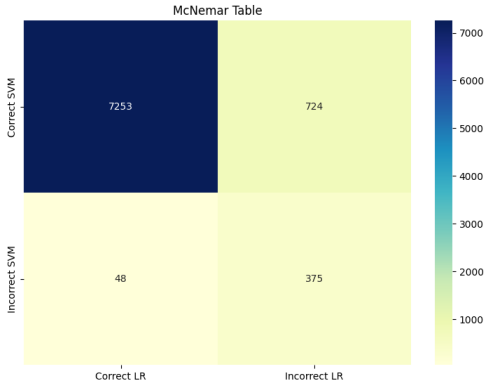

Figure 18: McNemar's table comparing the predictions of the best SVM model and the best Logistic Regression model trained with unused pixels removed

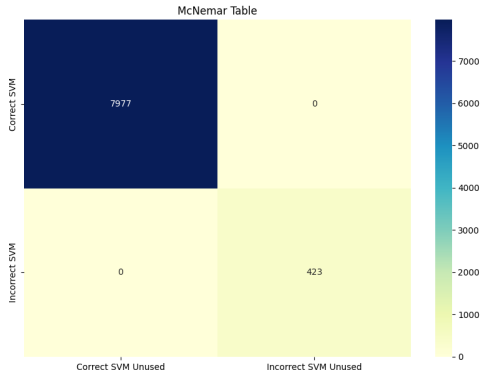

Figure 19: McNemar's table comparing the predictions of the best SVM model trained with the complete dataset and the reduced dataset

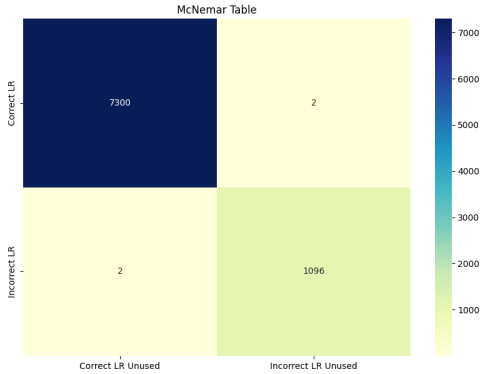

Figure 20: McNemar's table comparing the predictions of the best Logistic Regression model trained with the complete dataset and the reduced dataset

4 Conclusion & Discussion

Our research focused on the digit recognition task; specifically with the usage of the MNIST dataset. In our efforts to complete and optimize the task at hand we tried to analyze and understand the given data. The features we extracted and explored were: the ink used per image, the regional distribution of information and the raw pixel values of each image. The first two features were unable to provide a comprehensive and effective prediction model with the usage of multinomial logistic regression. However, the model trained on raw pixel values seemed to have high potential after some hyper-parameter tuning to avoid any overfitting. In order to further investigate the potential of this feature we decided to also use the SVM model to measure the significance of the prediction accuracy. After some cross-validation analysis the best configurations of each model were found, achieving 89% accuracy for the multinomial logistic regression and 93% accuracy for SVM. The difference between these models were significant enough for the task of digit recognition as shown by the use of the McNemar's test. The next step of our analysis was to investigate the effect of the removal of unused pixels which did not lead to any significant difference. Moving forward, refining our models without compromising accuracy will be a key focus. Exploring alternative feature combinations and tweaking preprocessing techniques could potentially lead to more robust and generalized models for digit recognition. Additionally, delving deeper into other statistical methods to

validate model differences could enhance the credibility of our findings. In conclusion, our research underscores the significance of feature engineering and model selection in the task of digit recognition.

References

[1] Mariam Omar Alshrief, Wafa El-Tarhouni, and Kenz Amhmed Bozed. "Ensemble machine learning model for classification of handwritten digit recognition". In: (2022), pp. 580–585. DOI: 10.1109/MI-STA54861.2022.9837750.

[2] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29 (2012), pp. 141–142.

[3] Thomas G. Dietterich. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms". In: *Neural Computation* 10.7 (Oct. 1998), pp. 1895–1923. ISSN: 0899-7667. DOI: 10.1162/089976698300017197. eprint: https://direct.mit.edu/neco/article-pdf/10/7/1895/814002/089976698300017197.pdf. URL: https://doi.org/10.1162/089976698300017197.

[4] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[5] E. Poisson, C. Viard Gaudin, and P.-M. Lallican. "Multi-modular architecture based on convolutional neural networks for online handwritten character recognition". In: (2002), 2444–2448 vol.5. DOI: 10.1109/ICONIP.2002.1201933.

[6] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

Appendix

Library	Function	Parameters	Scope
pandas	read_csv()	filepath_or_buffer = 'mnist.csv'	Data preparation
sklearn.model_selection	train_test_split()	arrays = mnist_data, train_size = 5000, random_state = 1	Data preparation
sklearn.preprocessing	scale()	axis = 0, with_mean = True, with_std = True, copy = True	Data preparation
sklearn.linear_model	LogisticRegression()	Table 2	Model Definition
sklearn.svm	SVC()	Table 3	Model Definition
sklearn.model_selection	GridSearchCV()	estimator = LogisticRegression() or SVC(), param_grid = From Table 2 or 3	Parameter Tuning
numpy	mean()	NA	Statistical Analysis
numpy	std()	NA	Statistical Analysis
numpy	bincount()	NA	Statistical Analysis
sklearn.metrics	confusion_matrix()	NA	Statistical Analysis
sklearn.metrics	accuracy_score()	normalize=True	Statistical Analysis
mlxtend.evaluate	mcnemar_table()	y_target=real_values, y_model1=predicted_values_m1, y_model2=predicted_values_m2	Statistical Analysis
mlxtend.evaluate	mcnemar()	ary=mcnemar_table, corrected=True	Statistical Analysis

Table 5: Library list

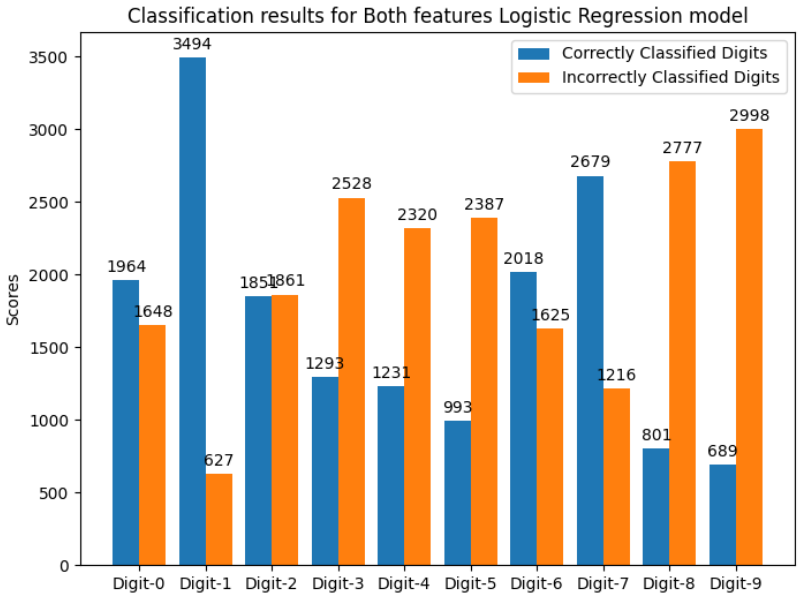

Figure 21: Combined feature classification report

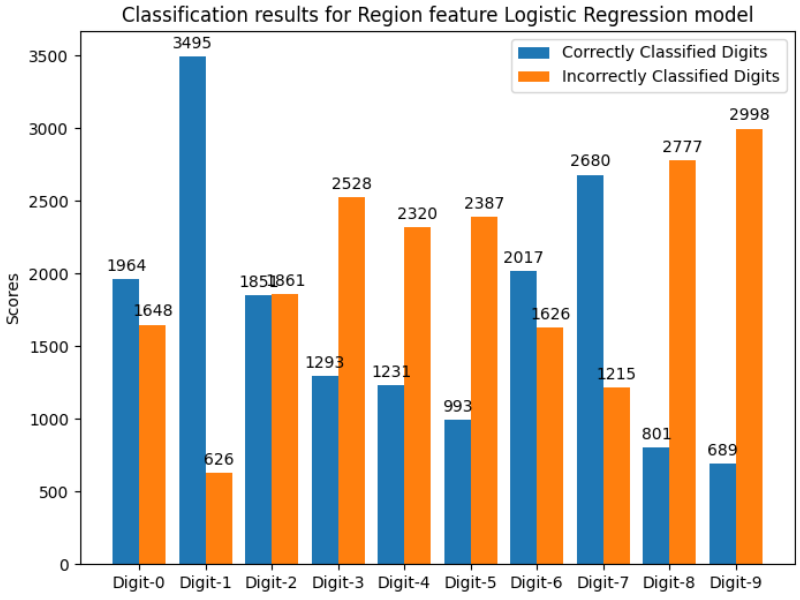

Figure 22: Region feature classification report