# Pattern recognition Practical Assignment


| Angel Temelko | Idan Grady | Marc de Fluiter |
|---|---|---|
| 8221113 | 7304447 | 5928087 |


## 1 Introduction

In the era of rapid technological development, the ability to convert and classify objects correctly is becoming more important than ever. Among other challenges, the recognition of handwritten digits is an important problem in optical character recognition, and it can also be used as a test case for theories of pattern recognition and machine learning algorithms(Zhu, 2015). Several standard databases have emerged in order to facilitate machine learning and pattern recognition research. The MNIST dataset is known as one of the commonly used datasets for machine learning and computer vision research for the handwritten digits classification task. In this report we follow the instructions [1], and explore the dataset in different directions. Firstly, based on the relation of the pixels , we created several new features, namely an ink feature, an area feature and a border feature. We trained and classified these new features separately and paired together with (regularized) Multinomial Logit models. Secondly, we perform a classification task on a reduced dimension of the original data set using principal component analysis (PCA[2]) with four different algorithms, namely (regularized) Multinomial Logit,Support Vector Machine,Neural Networks. Moreover, for each model, a hyper parameter tuning was performed with the sklearn library[3], and then using the best model for testing and evaluation. Rather than reinvent the wheel, our focus is to improve our understanding of a variety of classification tasks and the relationship between different features in the dataset. But before going into the answer of the question of how well each algorithm can differentiate between digits, we will first look at the data we worked with.

## 2 Data Exploration

### 2.1 General Information

In the data file mnist.csv, grayscale images of hand-drawn digits are included, for all digits 0 to 9. There are 28 pixels in height and 28 pixels in width for each

[1] https://www.cs.uu.nl/docs/vakken/mpr/computer−labPython.php
[2] https://en.wikipedia.org/wiki/Principal/component/analysis
[3] https://scikit-learn.org/stable/modules/generated/sklearn.modelselection.GridSearchCV.html

image, for a total of 784 pixels in total, where the first pixel indicates the class label. The used dataset contains 42000 digits. The pixel values range from 0 to 255 inclusively, with higher numbers denoting a darker color and lower numbers a lighter one.

### 2.2

| Digit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Amount | 4132 | 4684 | 4177 | 4351 | 4072 | 3795 | 4137 | 4401 | 4063 | 4188 |

| Mean Value | Median | Standard deviation | Variance |
|---|---|---|---|
| 4200 | 4157 | 224.92 | 50590 |

Table 1: Amount Of digits

In Table 1 we can see the amount of digits we have from each class. This is important as we could better understand the distribution of the data sets. Based on these simple statistics, we could understand how the data looks in a very rough manner. One thing we could clearly see, that digit 1, which appears 4684 times in the dataset, would be selected if the algorithms would simply predict the majority class.

Another visible aspects could be drawn out when comparing the median and the mean value. In this case, the mean is greater than the median which indicates the the distribution is positively skewed. This can be also seen in this in Figure 1.

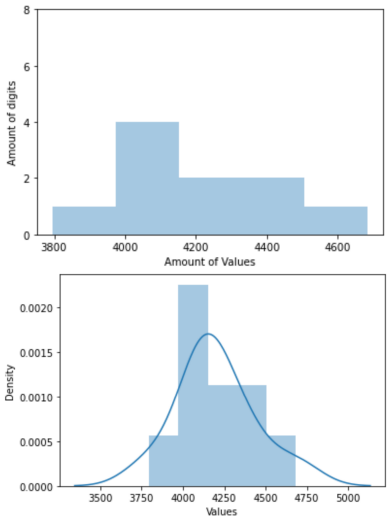



Figure 1: Data Distribution compared to Gaussian Distribution

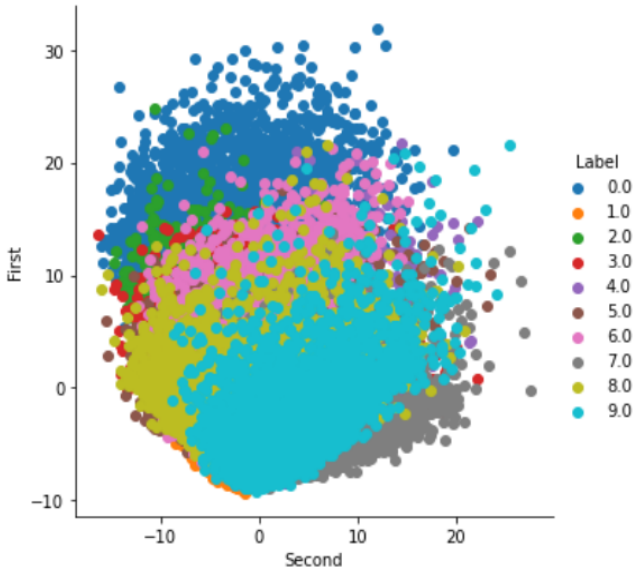
### 2.3 Features Experimentation

The first step that was taken in this process was to understand whether or not all the features, or in our current cases, pixels, contribute to our analysis. Sparse data is a common problem in machine learning these days, and it could alter the performace of the algorithm and its ability to generate an accurate prediction (Wärnling and Bissmark, 2017). When a pixel contains zero value in all the digits, it doesn't contain any useful information for us. To begin with, we summed the number of pixels with zero value and those with non-zero value in the total datasets. With 26621312 white pixels and 6306688 non-white pixels, we get 0.808,

or approximately 80% percent of the pixels
in the datasets contain the value zero. The next step was to see if there are pixels that contain zero values across all digits, and therefore increase the sparsity of the dataset. Over the entire dataset, there are 76 features that contain only zero values (List).

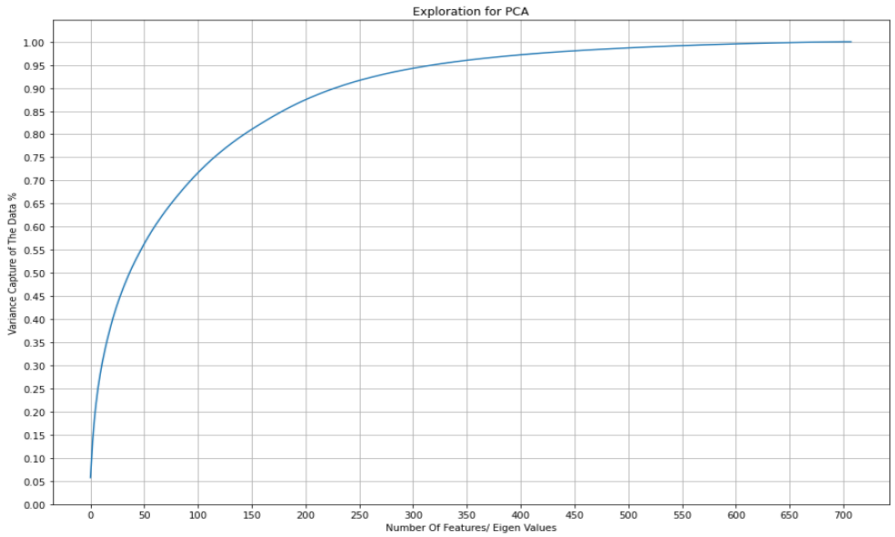
To take that to the maximum, we took it another step further. With the help of PCA, we found the two features which contain the greatest varience across the dataset, and plot it as a graph. We can see an interesting observation. The data seems to cluster in a few places. On intuition, it appears that 0 and 9 are separable. Yet the data is clustered together.

Figure 2: 2D Transformation



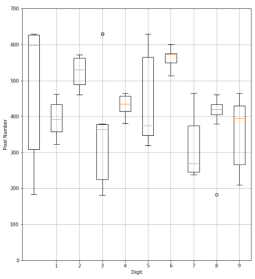

It was only natural to ask, how many features are required in order to capture the data. To explore this question, we used once again PCA techniques. Considering that the total variance is the sum of the variance of the individual components, we measure the ratio between the a principle components to the total variance. By doing that, we understand how much variance is preserved when some features are drooped (Cheplyaka, 2017). We ordered the features based on their variance. This means that an important feature would have a greater variance.

Figure 3: PCA Experiment

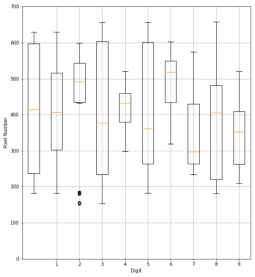



It is clearly visible from Table 3 that we can still capture a good percentage of the variance distribution with drastically fewer features. In order to capture 50% of the total variance in the dataset, we could use less than 50 features, which is $\frac{50}{(28*28)} = 0.063$ percent of the original feature size. The amount of preserved variance required depends on the task. As noted in the book "Multivariate Analysis" by Hair et al (2012)[4], the minimum required is 60%; however it is fairly common for 90% to be maintained(Brems, 2019). In the final stage of our data exploration, we analyzed which pixels contained the greatest variance per digit across the data. The results were plotted below.

Figure 4: The x "Greatest" pixels Per Digit



(a) The 10 pixels with the highest variance



(b) The 50 pixels with the highest variance

[4]https://fdocuments.in/document/hair-multivariate-data-analysis-7th-ed-us.html

# 3 Global Features

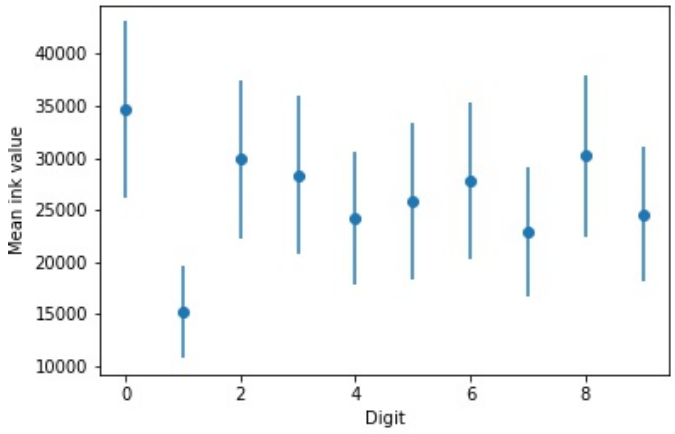
## 3.1 Ink Feature

The first global feature we consider is the *ink* feature. This feature quantifies the amount of 'ink' that is used for an image. It is a single value for every image, which is obtained by summing up all grayscale values of the image. This value represents the amount of ink that would be used if the digit in the image would be written with a real life ink pencil according to the pixel intensities of the given image.

We are interested whether using the 'ink' feature gives a good indication of the class labels, which indicate the digits that are written in the images. For every digit we calculated the mean value and the standard deviation of this feature over all images containing the given digit. This distribution can be found in Figure 5. The specific values are reported in Table D-1. Here we can see that the digit '1' can be distinguished the most easily from the other digits, because the mean value is way lower than for the other digits. If we look at the ranges of one standard deviation about the mean, we also see that there is only a slight overlap for these ranges of the digit '1' with the digits '4', '5', '7' and '9'. Apart from the digit '1' the digit '0' is the most distinguishable, because its mean is fairly higher than the other digits and way higher than the mean for the digit '1'. We do notice that there is a large overlap though with all digits except for the digit '1' in terms of the range one standard deviation about the mean. In general all digits except the digit '1' show a relatively big overlap in these ranges with each other. Therefore, when looking at the *ink* feature, we should be able to distinguish the digit '1' relatively well from the digits '0', '2', '3', '6' and '8', but distinguishing between other pairs of digits will still prove to be difficult.

In general two digits can be distinguished better the further the mean values are away from each other and, partly consequently, the smaller the overlap between the ranges of one standard deviation about the mean is. For the *ink* feature these ranges all overlap quite a bit, except for the digit '1'.

Figure 5: Distribution of the ink feature for each digit.




To really test how well the *ink* feature can distinguish between digits, we fit a logistic regression model to the values of the *ink* feature. Before we fit this model to the data, we firstly scale the *ink* values such that they have a mean of zero and unit variance. Because we are using a rather simple model and just want an indication of how well this feature performs we will train this model on the full set of 'ink' values and then test the model on the full set of 'ink' values as well. Besides that, because it is such a simple model, we will also not apply a penalty for the model complexity to this model. In our implementation we used the '$LogisticRegression$' class from the '$sklearn.linear\_model$' python module with its standard solver '$lbfgs$' to perform the multinomial logistic regression.

The logistic regression model fitted using the 'ink' feature produces the confusion matrix presented in Table 2. We see that the model never predicts the digits '4', '5', '6' and '8', so it was evidently too difficult for this model to distinguish between these and the other digits based only on the ink feature. The accuracy of the model can be calculated from the counts in the confusion matrix by dividing the number of correct predictions by the total number of predictions, this gives an accuracy of approximately 22.7%. If we simply predict the majority class, which is the class corresponding to the digit '1' as we have noted earlier, we only achieve an accuracy of 11.2%. So this already gives a relatively good improvement on predicting the class label for the digit that is written in the image.

| | | Predicted Digit | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **True Digit** | **0** | 2420 | 83 | 322 | 805 | 0 | 0 | 0 | 384 | 0 | 118 |
| | **1** | 10 | 3823 | 5 | 101 | 0 | 0 | 0 | 722 | 0 | 23 |
| | **2** | 1495 | 280 | 327 | 1038 | 0 | 0 | 0 | 874 | 0 | 163 |
| | **3** | 1246 | 408 | 335 | 1037 | 0 | 0 | 0 | 1141 | 0 | 184 |
| | **4** | 440 | 829 | 196 | 886 | 0 | 0 | 0 | 1496 | 0 | 225 |
| | **5** | 727 | 671 | 198 | 846 | 0 | 0 | 0 | 1190 | 0 | 163 |
| | **6** | 1057 | 450 | 296 | 982 | 0 | 0 | 0 | 1145 | 0 | 207 |
| | **7** | 325 | 1190 | 149 | 818 | 0 | 0 | 0 | 1700 | 0 | 219 |
| | **8** | 1430 | 192 | 343 | 1047 | 0 | 0 | 0 | 879 | 0 | 172 |
| | **9** | 484 | 763 | 197 | 869 | 0 | 0 | 0 | 1651 | 0 | 224 |

Table 2: Confusion matrix for the logistic regression model fitted on the *ink* feature. The table shows the counts corresponding to the amount of times where the model predicts the digit labelled 'Predicted Digit' and where the actual class label is the digit labelled 'True Digit'.

From the counts in the confusion matrix we can also determine how well the model can distinguish between a pair of digits. To calculate the accuracy with which the model distinguishes between two digits we look at the confusion matrix restricted to these two digits, so we only consider the four counts where the true digit is either one of the two considered digits and the predicted digit is also one of the two considered digits. For example for distinguishing between the digit '0' and the digit '1' we only consider the 4 counts in the top-left $2 \times 2$ area of the complete confusion matrix. From this we can see that this model can distinguish between the digits '0' and '1' with an accuracy of $\frac{2420+3823}{2420+3823+83+10} = 98.5\%$. For every pair of digits the accuracies of distinguishing between them is given in Table B-1. Here we can see that the digit '1' can indeed be distinguished the best among all digits, followed by the digit '0'. And for example the digit '7' is the hardest to distinguish from the digit '1' based on the *ink* feature.

## 3.2 Extra Features: Area and Border

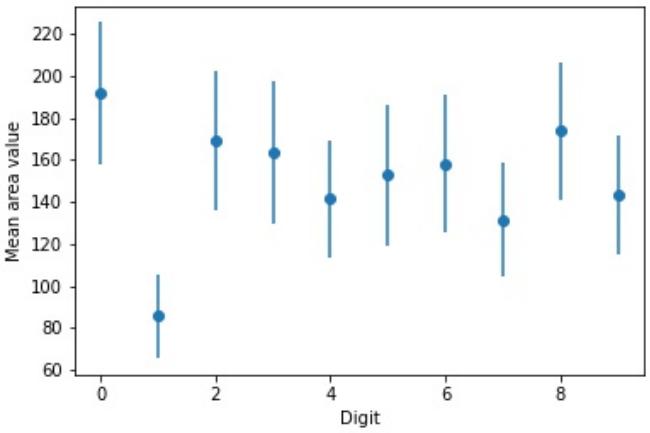
Besides the *ink* feature we also consider the *area* feature and the *border* feature. The *area* feature is quite similar to the *ink* feature. It calculates the number of pixels that have a non-zero grayscale value and so the amount of pixels that are used for writing the digit, which is basically the area of the digit in the image. So you could say that the area captures the 2D-volume, while the ink captures the 3D-volume, which we expect to be strongly corrected. Therefore we expect it to perform somewhat equal to the *ink* feature.

On the other hand the *border* feature captures somewhat different information. This feature calculates the number of transitions between pixels with an intensity of zero to neighbouring pixels with a non-zero intensity. This is done by counting the transitions from zero to non-zero values and from non-zero to zero values by traversing the pixels from left to right. This count is performed

once more by traversing the pixels from top to bottom. These two counts are then added together into the value for the *border* feature. This feature is therefore proportional to the border of the digit in the image. The *border* is of course correlated to the *area*, but we expect the correlation to be less than the ink feature.

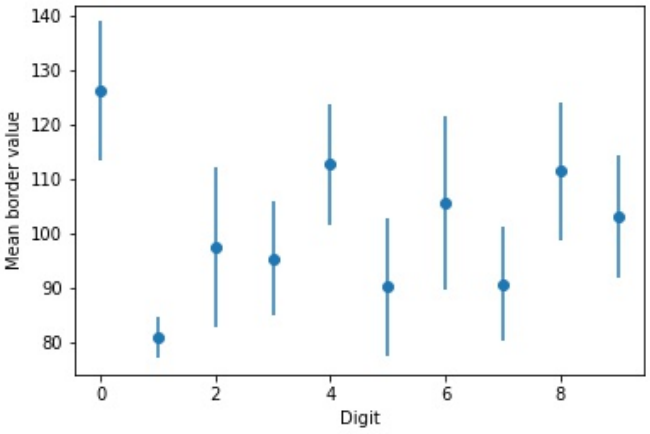
The mean values of the *area* feature are different for every digit like the *ink* feature and therefore give a faint indication of which digit is written. But since the handwritten digits can have very different shapes, sizes and thicknesses, the area of a certain digit could vary quite a bit image by image. This high variance can be seen in Figure 6, which paints a very similar picture to the distribution for the *ink* feature, only on a different scale of size. The values corresponding to this distribution are reported in Table A-2. Therefore fitting with the *area* feature will most likely produce similar results to the *ink* feature.

Figure 6: Distribution of the area feature for each digit.




Since every digit has a very different shape we expect the *border* feature to at least distinguish well between a lot of the digits. Of course having different shapes does not have to imply that the length of the borders are also different. Nevertheless the distribution for the *border* feature, as shown in Figure 7, shows that the mean values for each digit are in general more spread out than for the other features, except for a few values which are very close together, like for example the values for the digits '5' and '7'. The values for the mean and standard deviation of this feature for each digit can be found in Table A-3. The values of the standard deviations show a lot more variation between digits than for the other features. Consequently the *border* feature will probably distinguish some pairs of digits better than the other features, but also some pairs of digits worse.

Figure 7: Distribution of the border feature for each digit.




To compare the distinguishability of the *area* and *border* feature with the *ink* feature we again fitted a logistic regression model to both these features. This model is fitted completely analogous to how this was done for the *ink* feature. That is we used standardized data, used the full range of value for both training and testing and we used the same logistic regression model class with no penalty to fit the data. The confusion matrix of the model fitted on the *area* feature can be found in Table 3 and the confusion matrix of the model fitted on the *border* feature can be found in Table 4. The model fitted on the *area* feature gives an accuracy of 25.5%. Here we see that the *area* feature does predict the digit '8', where the *ink* feature did not, but still does not predict the digits '4', '5' and '6'. So based on the *area* we can distinguish the digit '8' relatively well and we see that this feature is overall slightly better in distinguishing between digits than the *ink* feature. The model fitted with the *border* feature only never predicts the digit '5' and almost never the digit '8', namely only once. So based on this feature the model can distinguish the digits '4' and '6' from the other digits where the other models could not. Still this model predicts these digits often wrongly, but at least it gives a stronger indication of the digit as we can also see in the accuracy of the model, where the accuracy of the model fitted on the *border* feature is 29.2%.

| | | **Predicted Digit** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **True Digit** | **0** | 2511 | 7 | 130 | 850 | 0 | 0 | 0 | 230 | 190 | 214 |
| | **1** | 3 | 4044 | 1 | 24 | 0 | 0 | 0 | 575 | 0 | 37 |
| | **2** | 1459 | 95 | 142 | 1092 | 0 | 0 | 0 | 769 | 195 | 425 |
| | **3** | 1298 | 180 | 131 | 1122 | 0 | 0 | 0 | 916 | 179 | 525 |
| | **4** | 332 | 387 | 75 | 952 | 0 | 0 | 0 | 1705 | 87 | 534 |
| | **5** | 727 | 297 | 92 | 915 | 0 | 0 | 0 | 1149 | 139 | 476 |
| | **6** | 970 | 202 | 123 | 1070 | 0 | 0 | 0 | 1082 | 144 | 546 |
| | **7** | 186 | 796 | 62 | 695 | 0 | 0 | 0 | 2070 | 42 | 550 |
| | **8** | 1519 | 25 | 156 | 1170 | 0 | 0 | 0 | 594 | 190 | 409 |
| | **9** | 414 | 333 | 75 | 899 | 0 | 0 | 0 | 1762 | 69 | 636 |

Table 3: Confusion matrix for the logistic regression model fitted on the *area* feature. The table shows the counts corresponding to the amount of times where the model predicts the digit labelled 'Predicted Digit' and where the actual class label is the digit labelled 'True Digit'.

| | | **Predicted Digit** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **True Digit** | **0** | 3236 | 28 | 57 | 52 | 479 | 0 | 142 | 56 | 0 | 82 |
| | **1** | 8 | 4449 | 7 | 17 | 13 | 0 | 6 | 172 | 0 | 12 |
| | **2** | 360 | 841 | 444 | 504 | 418 | 0 | 316 | 932 | 1 | 361 |
| | **3** | 116 | 869 | 519 | 642 | 266 | 0 | 282 | 1247 | 0 | 410 |
| | **4** | 1185 | 55 | 244 | 135 | 1311 | 0 | 567 | 146 | 0 | 429 |
| | **5** | 56 | 1409 | 356 | 423 | 174 | 0 | 160 | 993 | 0 | 224 |
| | **6** | 807 | 324 | 454 | 480 | 545 | 0 | 365 | 682 | 0 | 480 |
| | **7** | 60 | 1794 | 322 | 381 | 169 | 0 | 177 | 1229 | 0 | 269 |
| | **8** | 1242 | 130 | 306 | 225 | 1060 | 0 | 463 | 224 | 0 | 413 |
| | **9** | 382 | 207 | 635 | 564 | 641 | 0 | 508 | 651 | 0 | 600 |

Table 4: Confusion matrix for the logistic regression model fitted on the *border* feature. The table shows the counts corresponding to the amount of times where the model predicts the digit labelled 'Predicted Digit' and where the actual class label is the digit labelled 'True Digit'.

From the confusion matrices for the *area* and *border* features we can calculate the accuracies of distinguishing between pairs of digits again. These accuracies for the *area* and *border* features are reported in respectively Table B-2 and Table B-3. Overall the accuracies for the *area* feature seem to be a little bit better than the accuracies for the *ink* feature, where the accuracy is better in many cases but in many other cases also worse. Overall there seems to be a slight improvement as also indicated by the accuracy of the complete model. For the *border* feature the accuracies seem to improve again compared to the *area* feature. Once again in a lot of cases there is a drop in accuracy as well, but in

most cases there is an improvement in the accuracy. And these improvements seem to outweigh the drops in accuracies, what we can also conclude by the overall accuracy of the complete model.

### 3.3 Combining Features

By combining the independent features we hope to get more accurate results. On all different combinations of the features *ink*, *area* and *border* we fitted the logistic regression model. The accuracies of these models can be found in Table 5. Those accuracies tell us that using more features gives more accurate results in all cases. Furthermore we can derive that adding the *border* feature always gives a higher increase in the accuracy than the *area* feature and in turn adding the *area* feature always gives a higher increase in the accuracy than the *border* feature. This is also expected, since this corresponds to the case where we used the single features. Furthermore we note that the *border* feature really captures more information than the other features, since the *border* feature on itself gives a higher accuracy than the *ink* feature and the *area* feature combined.

| **Features** | | | **Accuracy** |
|---|---|---|---|
| **Ink** | **Area** | **Border** | |
| ✓ | - | - | 22.69% |
| - | ✓ | - | 25.51% |
| - | - | ✓ | 29.23% |
| ✓ | ✓ | - | 28.63% |
| ✓ | - | ✓ | 35.38% |
| - | ✓ | ✓ | 36.87% |
| ✓ | ✓ | ✓ | 41.65% |

Table 5: Accuracies of the logistic regression model using different combinations of the features *ink*, *area* and *border*.

In Table 4 we can see the confusion matrix for the model fitted on all three combined features *ink*, *area* and *border*. We see that by combining these features, this model does have the ability to distinguish between all digits, that is that no digit is never predicted. These predictions are still wrong most of the time, but compared to the independent features there are far less wrong predictions. This is also signified by the overall accuracy of this model, where its accuracy is 41.65%. This improves a lot on the models fitted on the individual features.

The model for the combined features can also distinguish between pairs of digits far better than the models for the individual features. The accuracies for distinguishing between the pairs of digits are reported in Table B-4. In two cases the accuracy drops very slightly. The accuracy for distinguishing between the digits '0' and '4' drops from 88.3% to 87.3% when compared to the *area* feature and for distinguishing between the digits '4' and '7' the accuracy drops from 89.0% to 88.9% when compared to the *border* feature. Apart fro. these slight decreases in accuracy, the accuracy for the model with the combined features is

higher in all other cases compared to the the models of the individual features. In most cases the accuracy is even improved significantly.

| | | **Predicted Digit** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **True Digit** | **0** | 3193 | 2 | 108 | 56 | 320 | 29 | 36 | 8 | 373 | 7 |
| | **1** | 0 | 4257 | 3 | 5 | 29 | 8 | 0 | 372 | 1 | 9 |
| | **2** | 240 | 80 | 711 | 783 | 167 | 690 | 185 | 505 | 658 | 158 |
| | **3** | 60 | 78 | 641 | 1193 | 105 | 883 | 121 | 677 | 374 | 219 |
| | **4** | 446 | 30 | 63 | 90 | 2051 | 48 | 266 | 227 | 403 | 448 |
| | **5** | 26 | 163 | 309 | 785 | 76 | 1401 | 81 | 567 | 238 | 149 |
| | **6** | 534 | 53 | 473 | 385 | 554 | 132 | 308 | 725 | 512 | 461 |
| | **7** | 25 | 489 | 185 | 262 | 292 | 490 | 145 | 2091 | 53 | 369 |
| | **8** | 526 | 5 | 437 | 266 | 614 | 62 | 296 | 123 | 1491 | 243 |
| | **9** | 137 | 58 | 287 | 372 | 858 | 118 | 304 | 905 | 352 | 797 |

Table 6: Confusion matrix for the logistic regression model fitted on the *ink*, *area* and *border* features. The table shows the counts corresponding to the amount of times where the model predicts the digit labelled 'Predicted Digit' and where the actual class label is the digit labelled 'True Digit'.

# 4 Classification

## 4.1 Prepossessing

Our data exploration stage brought to light two significant points. To begin with, there are pixels that do not contain any information of value. Second, the given Minst dataset is relatively sparse; in fact, approximately 80 percent of the pixels when summing them all across all digits are zero. These two aspects lead us to check the amount of variation we lose once the dimention of the data set is reduced. According to the assignment instructions, we can reduce the dimension of the data set to 14 x 14 , resulting in 196 features. The 196 features with the greatest variance were obtained by using the previously described approach, and the analysis was performed on them. By doing so, we potentially lost approximately 15% of the variance in our dataset ( see Figure 3). It is important to note,that there is no way to ensure that the missing features will not provide valuable information for specific samples. Yet when taking a broader view, we consider essential information if it exists across multiple samples.

## 4.2 Result

| Model Name | Accuracy Test | AccuracyTrain | Precision | Recall | Fscore | Fold | params |
|---|---|---|---|---|---|---|---|
| Logit | 0.898 | 0.953 | 0.893 | 0.901 | 0.898 | 5 | 'C': 1.1, 'penalty': 'l1' |
| SVC | 0.942 | 0.977 | 0.941 | 0.943 | 0.942 | 5 | 'gamma': 0.1, 'C': 0.1 |
| MLPClassifier | 0.945 | 1.0 | 0.944 | 0.940 | 0.946 | 5 | 'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': (20,),'solver':'adam' |

Table 7: Test Result

## 4.3 Analysis

To begin, let's compare the result between Logit and SVC. As shown in Table 7, both models perform quite well, yet there is still a difference between them. SVC appears to perform better than logic across all the observed parameters. During training, both models showed higher accuracy. However, where the Logic dropped from 95 to 89.8%, the SVC dropped by a smaller amount, namely from 97 to 94.5%. The accuracy difference seems to be maintained across most parameter. In pattern recognition, the Precision and Recall values([5]) are important components in the analysis, where higher precision indicates that the model returns more relevant result than irrelevant onces, and higher recall means that the model returns most of the relevant result. Based on this, we observe that the SVC is not only able identify more relevant result, as shown with its greater precision. It also has a higher recall, which indicate .... Interestingly, recall and precision for SVC are almost identical in comparison to Logit. In Logit, the false negative is smaller than the false positive, but in SVC, the false positive and false negative are nearly equal.

To continue, we would compare between the Logit and the MLPclassifier models. We can draw a similar pattern as observed in the previous comparison. The MLPclassifier, which generally performed the best in this analysis, outperformed the Logic in all observed parameters. An interesting observation is when comparing the drop on accuracy between the training procedure to the test classification. The drop is relatively similar. Where by the Logit : $\frac{\mathbf{0.898}}{\mathbf{0.953}} = \mathbf{0.942}$ where by MLPclassifier: $\frac{\mathbf{0.945}}{\mathbf{1}} = \mathbf{0.945}$. Another difference can be seen when comparing the precision and recall parameters. Although MLPclassifier outperformed Logit across all parameters. The correlation between false positives and false negatives appeared to be incongruent. In comparison with false positive, Logit generates a smaller fraction of false negatives, whereas MLPclassifier generates the opposite result.

Our last two models for comparison are MLPClassifier and SVC. They performed similarly across the observed parameters. One of the main differences is seen when comparing the accuracy differences between training and testing. The MLPClassifier was able to classify the data correctly during training, but during testing, the accuracy score declined by 5.5 to 94.5 percent. It appears that the observed drop in SVC is less drastic, namely by 3.5 percent to 94.2%. Various factors could explain this difference. MLPClassifier could have assigned weights to noises to achieve a perfect classification, where SVC was less. Fur-

[5] https://en.wikipedia.org/wiki/Precision_and_recall

thermore, it is important to note that MLPClassifier correctly classified the data even with a single hidden layer. The high number of hidden units by the MLPClassifier could have contributed to the potential overfitting during the training process. There was also a minor difference between the percentage of false positives versus false negatives between the two models: SVC generates a smaller fraction of false negatives, while MLPclassifier generates fewer false positives. Yet the differences are quite small.

## 4.4 Significant

To test between models, we use a statistical test called McNemar's[6] test. The test is based on and makes use of a contingency table, which is depicted in table 3. This table marks how many items were correctly classified by each algorithm. The more commonly used and by Edwards corrected (continuity correction) formula to calculate the statistical result is as follows:

$$x^2 = \frac{(|B - C| - 1)^2}{(B + C)}$$

| **Model** | **SVC** | **Logit** |
|---|---|---|
| **Model** 1 **Correct** | A | B |
| **Model** 1 **Wrong** | C | D |

Table 8: McNemar Visual Explanation with model 1 = LR and model 2 = SVC

As you can see the test calculates a significance over the differences of the models, as it is only interested in items that the models performed different on. The statistic is reporting on the different correct or incorrect predictions between the two models.

### 4.4.1 Classification Test

Table 9: Statistical Testing Tables

| **Model** | **Logit** | **SVC** | **MLPC** |
|---|---|---|---|
| **Logit** | | 679.851 | 713.903 |
| **SVC** | 679.851 | | 0.165 |
| **MLPC** | 713.903 | 0.165 | |

Table 10: $Chai^2$

| **Model** | **Logit** | **SVC** | **MLPC** |
|---|---|---|---|
| **Logit** | | $\mathbf{7.197e^{-150}}$ | $\mathbf{2.833e^{-157}}$ |
| **SVC** | $\mathbf{7.197e^{-150}}$ | | 0.683 |
| **MLPC** | $\mathbf{2.833e^{-157}}$ | 0.683 | |

Table 11: $PValue$

The null hypothesis for each of the combination is that the models would perform similarly. We would reject this null hypothesis if the p-values of the table are

[6] https://wikistatistiek.amc.nl/index.php/$McNemar\_toets$/

smaller than 0.05. The result shows that if the null hypothesis holds for the comparison of MLPC and SVC algorithms However it must be rejected for all other comparisons, including the comparison of Logit and SVC algorithms and that of Logit and MLPC algorithms. This supports the previous findings about the differences between Logit and the two other models in terms of accuracy, recall, and precision. While the significant test indicates that Logit should not be considered as the best algorithm for this task, we did not find any significant differences between SVC and MLPC. This is also in line with our earlier findings across the other parameters.

### 4.4.2 Features Test

To see if the models which were fitted on the individual *ink*, *area* and *border* features performed similarly, we again used McNemar's statistical significance test. Here the null hypothesis is that for two combination of features the separate models would perform similarly and we reject this hypotheses when the p-value is smaller than 0.05. The McNemar's test produced the p-values that are presented in Table 14. These p-values are all extremely small, which clearly tells us that these features are very similar in predicting the digit class and thus capture much of the same information about the written digit in each image.

Table 12: Statistical Tests for the Feature Models

| **Feature** | **Ink** | **Area** | **Border** |
|---|---|---|---|
| **Ink** | | 405.649 | 789.839 |
| **Area** | 405.649 | | 249.626 |
| **Border** | 789.839 | 249.626 | |

Table 13: $Chai^2$

| **Feature** | **Ink** | **Area** | **Border** |
|---|---|---|---|
| **Ink** | | $\mathbf{3.245e^{-90}}$ | $\mathbf{8.735e^{-174}}$ |
| **Area** | $\mathbf{3.245e^{-90}}$ | | $\mathbf{3.134e^{-56}}$ |
| **Border** | $\mathbf{8.735e^{-174}}$ | $\mathbf{3.134e^{-56}}$ | |

Table 14: $PValue$

# 5 Conclusion

A unique feature of every human being is their iris, fingerprint, DNA, etc. Handwriting is one of those features, which is proven scientifically to differ from person to person (Vinjit et al., 2020). Since there are so many different handwriting styles, it is crucial to be able to classify digits correctly.

For the first step, a prepossessing phase was conducted in order to better understand the data. Two conclusions emerged from the experimentation. The data is relatively sparse and contains even large amounts of pixels without any contribution at all. Furthermore, we can capture a substantial amount of the total data variance by using considerably fewer features.

We saw that simple models which are fitted on features that describe the digits globally can improve classification significantly compared to predicting the majority class. Still the accuracy of these models is still quite low, where the right prediction was made only around 25% of the time. These models did show that distinguishing between a given pair of digits could be done relatively well

by these models, where this accuracy is generally higher than 50% and most of the times a lot more accurate, in some cases even just shy of 100%. This is also what we expect from multinomial models. Furthermore we saw that by combining several of these features we gained an accuracy of about 42%, which is a lot higher than for the individual features. So by choosing smart features that globally describe the data and combining these features we can train models that distinguish between classes with an acceptable accuracy. Still the much more complex models we used later on heavily outperform the simple models using a combination of these global features. The statistical significance tests told us that the three features perform very similar to each other. Therefore we would need a lot of features to eventually gain a very high accuracy in our model. This only confirms that much more complexity is needed in this case to gain a highly accurate model.

Additionally, we trained and tested three different models on the 198 features that contained the most variance across the data. The null hypothesis was that all models performed equally well. Comparing the Logit algorithm to MLP and SVC, our null hypothesis is rejected statistically. However, we were unable to reject the null hypothesis when comparing SVC to MLP algorithms, since no significant differences were found between them. It is interesting to observe the difference in training and testing accuracy between both best performing algorithms. Despite MLP's perfect classification during the training phase, it shows a relatively big drop during testing. SVC accuracy decreases as well, but it's less aggressive. Slight differences exist between false positives and false negatives among the SVC and MLP, although the differences aren't significant. This could be factored into deciding between them two for a future task.

In conclusion, SCV and MLP both do very well on digit recognition tasks. Despite the minor differences, we prefer to work with MLP due to its high versatility. As the complexity of the task increases, this versatility could be helpful. Nevertheless, both are equally effective for a simple digit recognition task.

# References


Brems, M. (2019, June 10). *A one-stop shop for principal component analysis* [Medium]. Retrieved December 4, 2021, from https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

Cheplyaka, R. (2017). *Explained variance in PCA*. Retrieved December 4, 2021, from https://ro-che.info/articles/2017-12-11-pca-explained-variance

Vinjit, B. M., Bhojak, M. K., Kumar, S., & Chalak, G. (2020). A review on handwritten character recognition methods and techniques. *2020 International Conference on Communication and Signal Processing (ICCSP)*, 1224–1228. https://doi.org/10.1109/ICCSP48568.2020.9182129

Wärnling, O., & Bissmark, J. (2017). The sparse data problem within classification algorithms, 33.

Zhu, W. (2015). Classification of MNIST handwritten digit database using neural network, 7.

# Appendices

## A Feature Values

| Ink Feature | | |
|---|---|---|
| **Digit** | **Mean Value** | **Standard Deviation** |
| 0 | 34632.41 | 8461.89 |
| 1 | 15188.47 | 4409.46 |
| 2 | 29871.10 | 7653.01 |
| 3 | 28320.19 | 7574.10 |
| 4 | 24232.72 | 6374.63 |
| 5 | 25835.92 | 7526.60 |
| 6 | 27734.92 | 7530.50 |
| 7 | 22931.24 | 6168.34 |
| 8 | 30184.15 | 7777.40 |
| 9 | 24553.75 | 6465.23 |

Table A-1: Mean values and standard deviations of the ink feature for each digit.

| Area Feature | | |
|---|---|---|
| **Digit** | **Mean Value** | **Standard Deviation** |
| 0 | 191.74 | 33.70 |
| 1 | 85.73 | 20.01 |
| 2 | 168.97 | 33.06 |
| 3 | 163.52 | 33.63 |
| 4 | 141.50 | 27.92 |
| 5 | 152.76 | 33.77 |
| 6 | 158.21 | 32.65 |
| 7 | 131.42 | 27.03 |
| 8 | 173.97 | 32.83 |
| 9 | 143.25 | 28.54 |

Table A-2: Mean values and standard deviations of the area feature for each digit.

| Border Feature | | |
|---|---|---|
| **Digit** | **Mean Value** | **Standard Deviation** |
| 0 | 126.12 | 12.83 |
| 1 | 80.80 | 3.78 |
| 2 | 97.37 | 14.80 |
| 3 | 95.29 | 10.47 |
| 4 | 112.64 | 11.04 |
| 5 | 90.06 | 12.84 |
| 6 | 105.53 | 15.92 |
| 7 | 90.68 | 10.41 |
| 8 | 111.38 | 12.62 |
| 9 | 103.14 | 11.36 |

Table A-3: Mean values and standard deviations of the border feature for each digit.

# B Accuracies of Distinguishing Between Digits

| **1** | **2** | **3** | **4*** | **5*** | **6*** | **7** | **8*** | **9** | |
|---|---|---|---|---|---|---|---|---|---|
| 98.5% | 60.2% | 62.8% | 84.6% | 76.9% | 69.6% | 85.3% | 62.9% | 81.5% | **0** |
| | 93.6% | 90.5% | 82.2% | 85.1% | 89.5% | 74.3% | 95.2% | 83.7% | **1** |
| | | 49.8% | 62.5% | 62.3% | 52.5% | 66.5% | 48.8% | 60.5% | **2** |
| | | | 53.9% | 55.1% | 51.4% | 58.3% | 49.8% | 54.5% | **3** |
| | | | | N/A | N/A | 53.2% | N/A | 49.9% | **4*** |
| | | | | | N/A | 58.8% | N/A | 57.9% | **5*** |
| | | | | | | 59.8% | N/A | 52.0% | **6*** |
| | | | | | | | 65.9% | 50.7% | **7** |
| | | | | | | | | 56.6% | **8*** |

Table B-1: Accuracies of distinguishing between digits using the model fitted on the *ink* feature.
*: *Digit not predicted by the model.*

| **1** | **2** | **3** | **4*** | **5*** | **6*** | **7** | **8** | **9** | |
|---|---|---|---|---|---|---|---|---|---|
| 99.8% | 62.5% | 62.8% | 88.3% | 77.5% | 72.1% | 91.7% | 61.2% | 83.4% | **0** |
| | 97.8% | 96.2% | 91.3% | 93.2% | 95.2% | 81.7% | 99.4% | 92.7% | **1** |
| | | 50.8% | 65.4% | 60.7% | 53.6% | 72.7% | 48.6% | 60.9% | **2** |
| | | | 54.1% | 55.1% | 51.2% | 66.5% | 49.3% | 55.2% | **3** |
| | | | | N/A | N/A | 54.8% | 68.6% | 54.4% | **4*** |
| | | | | | N/A | 64.3% | 57.8% | 57.2% | **5*** |
| | | | | | | 65.7% | 56.9% | 53.8% | **6*** |
| | | | | | | | 78.0% | 53.9% | **7** |
| | | | | | | | | 63.3% | **8** |

Table B-2: Accuracies of distinguishing between digits using the model fitted on the *area* feature.
*: *Digit not predicted by the model.*

| **1** | **2** | **3** | **4** | **5*** | **6** | **7** | **8**** | **9** | |
|---|---|---|---|---|---|---|---|---|---|
| 99.5% | 89.8% | 95.8% | 73.2% | 98.3% | 79.1% | 97.5% | 72.3% | 89.2% | **0** |
| | 85.2% | 85.2% | 98.8% | 75.9% | 93.6% | 74.3% | 97.2% | 95.8% | **1** |
| | | 51.5% | 72.6% | 55.5% | 51.2% | 57.2% | 59.1% | 51.2% | **2** |
| | | | 83.0% | 60.3% | 56.9% | 53.5% | 74.0% | 56.0% | **3** |
| | | | | 88.3% | 60.1% | 89.0% | 55.3% | 64.1% | **4** |
| | | | | | 69.5% | 55.3% | N/A | 72.8% | **5*** |
| | | | | | | 65.0% | 44.1% | 49.4% | **6** |
| | | | | | | | 84.6% | 66.5% | **7** |
| | | | | | | | | 59.2% | **8**** |

Table B-3: Accuracies of distinguishing between digits using the model fitted on the *border* feature.
*: *Digit not predicted by the model.*
**: *Digit predicted only once by the model.*

| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | |
|---|---|---|---|---|---|---|---|---|---|
| 99.97% | 91.8% | 97.4% | 87.3% | 98.8% | 86.0% | 99.4% | 83.9% | 96.5% | **0** |
| | 98.4% | 98.5% | 99.1% | 97.1% | 98.9% | 88.1% | 99.9% | 98.7% | **1** |
| | | 57.2% | 92.3% | 67.9% | 60.8% | 80.2% | 66.8% | 77.2% | **2** |
| | | | 94.3% | 60.9% | 74.8% | 77.8% | 80.7% | 77.1% | **3** |
| | | | | 96.5% | 74.2% | 88.9% | 77.7% | 68.6% | **4** |
| | | | | | 88.9% | 76.8% | 90.6% | 89.2% | **5** |
| | | | | | | 73.4% | 69.0% | 59.1% | **6** |
| | | | | | | | 95.3% | 69.4% | **7** |
| | | | | | | | | 79.4% | **8** |

Table B-4: Accuracies of distinguishing between digits using the model fitted on the *ink*, *area* and *border* features.

# C Classification

Table C-1: Classification Confusion Matrices

| | | **Predicted Digit** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **True Digit** | **0** | 3167 | 0 | 22 | 7 | 6 | 30 | 22 | 8 | 23 | 24 |
| | **1** | 1 | 3655 | 48 | 20 | 19 | 21 | 11 | 28 | 64 | 12 |
| | **2** | 14 | 15 | 2927 | 98 | 43 | 14 | 35 | 46 | 44 | 19 |
| | **3** | 7 | 21 | 85 | 3039 | 4 | 101 | 2 | 31 | 115 | 52 |
| | **4** | 11 | 2 | 49 | 8 | 2947 | 31 | 40 | 40 | 14 | 129 |
| | **5** | 58 | 18 | 30 | 145 | 5 | 2619 | 32 | 7 | 103 | 30 |
| | **6** | 26 | 0 | 45 | 15 | 27 | 81 | 3125 | 4 | 19 | 1 |
| | **7** | 5 | 9 | 52 | 29 | 29 | 17 | 1 | 3224 | 20 | 129 |
| | **8** | 17 | 39 | 74 | 59 | 27 | 89 | 19 | 9 | 2835 | 31 |
| | **9** | 5 | 5 | 11 | 29 | 115 | 27 | 4 | 111 | 50 | 2968 |

Table C-2: Logit Confusion Matrix

| | | **Predicted Digit** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **True Digit** | **0** | 3225 | 0 | 16 | 2 | 1 | 12 | 20 | 6 | 14 | 8 |
| | **1** | 0 | 3695 | 10 | 10 | 11 | 11 | 9 | 24 | 40 | 8 |
| | **2** | 28 | 25 | 3191 | 106 | 55 | 43 | 67 | 78 | 51 | 55 |
| | **3** | 3 | 11 | 33 | 3189 | 1 | 57 | 0 | 11 | 52 | 56 |
| | **4** | 6 | 4 | 22 | 7 | 3046 | 7 | 18 | 24 | 16 | 89 |
| | **5** | 4 | 5 | 2 | 46 | 7 | 2817 | 22 | 1 | 50 | 10 |
| | **6** | 24 | 6 | 17 | 6 | 18 | 51 | 3146 | 0 | 11 | 0 |
| | **7** | 1 | 6 | 26 | 28 | 15 | 3 | 0 | 3298 | 11 | 101 |
| | **8** | 10 | 6 | 22 | 40 | 12 | 17 | 9 | 4 | 3015 | 27 |
| | **9** | 0 | 6 | 4 | 15 | 56 | 12 | 0 | 62 | 27 | 3041 |

Table C-3: SVC Confusion Matrix

| | | Predicted Digit | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **True Digit** | **0** | 3214 | 0 | 16 | 5 | 6 | 16 | 18 | 5 | 18 | 20 |
| | **1** | 0 | 3693 | 22 | 11 | 15 | 14 | 14 | 22 | 26 | 5 |
| | **2** | 8 | 18 | 3091 | 50 | 25 | 8 | 13 | 30 | 31 | 16 |
| | **3** | 7 | 11 | 49 | 3234 | 2 | 57 | 6 | 26 | 48 | 38 |
| | **4** | 10 | 4 | 29 | 5 | 3060 | 8 | 31 | 21 | 15 | 85 |
| | **5** | 28 | 8 | 15 | 52 | 1 | 2810 | 22 | 10 | 41 | 15 |
| | **6** | 20 | 1 | 32 | 9 | 19 | 39 | 3166 | 4 | 16 | 1 |
| | **7** | 9 | 12 | 42 | 22 | 17 | 12 | 3 | 3340 | 13 | 74 |
| | **8** | 9 | 11 | 36 | 42 | 21 | 49 | 16 | 11 | 3046 | 34 |
| | **9** | 6 | 6 | 11 | 19 | 56 | 17 | 2 | 39 | 33 | 3107 |

Table C-4: MLPClassifer Confusion Matrix

Table C-5: MCNmar Confusion Matrices

| **Model** | **SVC** | **Logit** |
|---|---|---|
| **Model 1 Correct** | 1399 | 470 |
| **Model 1 Wrong** | 1680 | 30051 |

Table C-6: SVC vs Logit

| **Model** | **MLPC** | **Logit** |
|---|---|---|
| **Model 1 Correct** | 1501 | 352 |
| **Model 1 Wrong** | 1928 | 29819 |

Table C-7: MLPClassifier vs Logit

| **Model** | **SVC** | **MLPC** |
|---|---|---|
| **Model 1 Correct** | 1183 | 670 |
| **Model 1 Wrong** | 686 | 31061 |

Table C-8: SVC vs MLPClassifier

# D Libraries

| Libraries | Used Libraries Per Stage | | |
|---|---|---|---|
| | **Pre** | **Classification** | **Features** |
| numpy | ✓ | ✓ | ✓ |
| pandas | ✓ | ✓ | ✓ |
| matplotlib | ✓ | ✓ | ✓ |
| collections/ Counter | ✓ | ✓ | |
| sklearn.preprocessing/StandardScaler | ✓ | ✓ | |
| sklearn/ decomposition/PCA | ✓ | ✓ | |
| scipy.linalg/eigh | ✓ | ✓ | |
| sklearn/ metrics | ✓ | ✓ | ✓ |
| sklearn/LogisticRegression | | ✓ | ✓ |
| sklearn/ SVC | | ✓ | |
| sklearn/MLPClassifier | | ✓ | |
| sklearn/ GridSearchCV | | ✓ | |
| sklearn/ train_test_split | | ✓ | |
| sklearn/precision_recall_fscore_support | | ✓ | |
| sklearn//mean_absolute_error | | ✓ | |
| statsmodels/ mcnemar | | ✓ | |
| mlxtend/ mcnemar | | ✓ | |
| sklearn/ preprocessing/ scale | ✓ | ✓ | ✓ |

Table D-1: Used Library Across Sections.