Assignment 1

Rens van Schouwenburg: 6626394  
&  
Jan Vrieler: 6569552

Pattern Recognition

A comparison of classification algorithms

Contents

[1. Introduction 2](#_Toc532557388)

[1.1 The dataset 2](#_Toc532557389)

[2. Data exploration 2](#_Toc532557390)

[3. Multinomial Logit 2](#_Toc532557391)

[3.1 Density 3](#_Toc532557392)

[3.2 Top-bottom ratio 4](#_Toc532557393)

[3.3 Density and Top-bottom ratio 5](#_Toc532557394)

[4. Data preprocessing 2](#_Toc532557395)

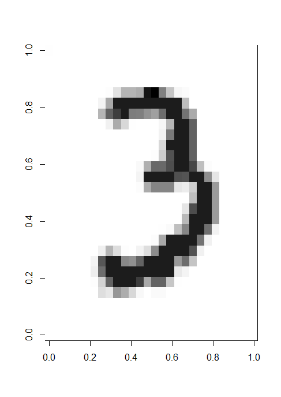
[5. Multinomial logit with LASSO 3](#_Toc532557396)

[6. K-nearest neighbour 3](#_Toc532557397)

[7. Support vector-machine 3](#_Toc532557398)

[8. Conclusion 4](#_Toc532557399)

# 1. Introduction

This report will discuss different classification algorithms and their performance on the Mnist dataset. First the dataset itself will be discussed, then a multinomial logit model will be applied to the dataset using single features. Thereafter three different models will be applied to the dataset and their performances will be compared. The models consist of; a regularised multinomial logit model, a k-nearest neighbour model and a support vector machine. The steps taking in preprocessing the data for the algorithms and the parameters chosen will also be discussed.

## 1.1 The dataset

The Mnist dataset contains 42.000 different handwritten numbers on a 28x28 grid. Each pixel in this grid has a value between 0 and 256. This value will be referred to as ‘pixel value’ throughout the report. The higher the pixel value, the ‘darker’ it is in the picture. A pixel value of 0 refers to an empty or unused pixel, whereas a pixel value of 256 refers to a completely darkened pixel. In total, each handwritten number consists of 784 pixels. Before the data is analysed, the data will be explored to spot potentially superfluous pixels. Figure 1 presents an example of how the handwritten digits are represented in the dataset. As can be seen, some parts of the digit are darker than others, these shades correspond with the ‘pixel value’ described earlier.

# 2. Data exploration

Creating a summary of every feature shows that there are pixels with a maximum value of 0. This suggests that the pixel values are 0 for all 42.000 observations. These pixels cannot contribute to the classification of the observations, because they are the same for every observation. Some pixels have scores very close to 0. A low pixel score doesn’t inherently mean that a pixel is useless for the classification of the different numbers. It might be that a pixel with a lower score is only used by one of the 10 numbers. However, when the score becomes too low, it is more likely that the pixel has been coloured incidentally in a few observations. Therefore, it might be beneficial to also delete pixels with a total score that is close to 0. In this case, a threshold of 1000 is chosen for removing superfluous pixels. A score of 1000 over 42.000 observations is still low, but the threshold is kept low to prevent loss of key features for identifying numbers. There are 151 pixels with a total pixel value under 1000. Almost all of those pixels reside on the corners or outer edges of the image. This seems logical, as the numbers are generally written on the middle of the paper.

Table 1 shows the distribution of the classes in the dataset. It shows that the least abundant number is 5, and that the majority class is 1. The mean of the different numbers is 4200. The majority class provides a baseline accuracy to compare to when evaluating new models. When the majority class is predicted (class 1) 11,15% of the predictions will be correct. Of course, this performance is far from stellar. To be useful, the accuracy of this model should be much higher.

Figure : handwritten number 3

# 3. Multinomial Logit

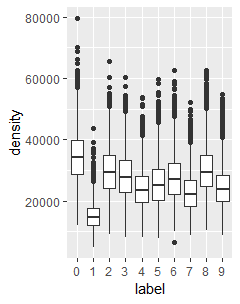
Table 1: count and percentage of classes

|  |  |  |
| --- | --- | --- |
| Class | Count | Percentage |
| 0 | 4132 | 9.83% |
| 1 | 4684 | 11.15% |
| 2 | 4177 | 9.95% |
| 3 | 4351 | 10.36% |
| 4 | 4072 | 9.70% |
| 5 | 3795 | 9.04% |
| 6 | 4137 | 9.85% |
| 7 | 4401 | 10.48% |
| 8 | 4063 | 9.67% |
| 9 | 4188 | 9.97% |

This chapter will discuss three multinomial logit models with different features. First, a feature called ‘density’ is created and discussed. Thereafter, its performance on classifying the handwritten digits will be discussed. The same procedure will be followed with a different feature called the top-bottom ratio. Finally, both features will be applied to the data, and the performance of the model will be discussed.

A multinomial logit model, also known as a multinomial logistic regression, is a classification method that can be applied to cases with multiple classes. It can be used to predict the class of an observation based on its variables. In case of this paper, which handwritten digits does the observation represent, based on the pixel values in it pixels. The model uses a linear combination of the features of an observation to predict its class. In this chapter, the model will be applied to the mnist dataset with either one or two features. The entire Mnist dataset was used for training and testing in this chapter. Note that this deviates from the next chapter.

## 3.1 Density

The first model tested was ran on a feature called ‘density’. The density of an observation can roughly be translated to its ‘ink cost’. To obtain the density of each observation, the sum of all pixel values in one observation was calculated. This resulted in 42.000 different densities. Table 2 shows the average density for each of the classes along with the standard deviation. Figure 2 visualises the distribution of density per class.

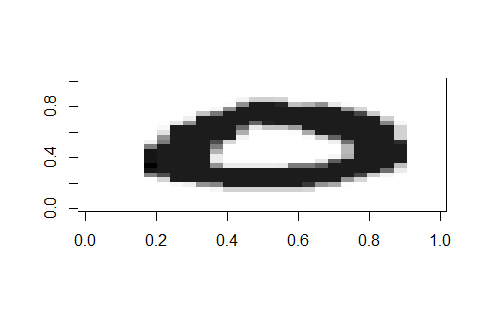


Figure 2: boxplot of density per class

Table 2 shows the mean densities for all 10 classes, and their standard deviations. It shows that some classes have a much different mean density than others. For instance, the digit 0, on average, uses double the ink that is used for the digit 1. In theory, these numbers should be easily distinguishable from each other based on their density. Reversely, the digits 9 and 4 are only an average pixel value if 300 apart from each other. This should make them more difficult to distinguish. In general, the means of the classes are relatively close to each other. Correct classification is made even more difficult by the fact that the standard deviations are relatively high. This shows that many observations deviate from the mean by several thousand points in pixel value.

Figure : digit no. 111

Table 2: mean and standard deviation of density per class

|  |  |  |
| --- | --- | --- |
| Class | Mean | Std. dev. |
| 0 | 34.632,41 | 8462,91 |
| 1 | 15.188,47 | 4409.93 |
| 2 | 29.871,10 | 7653,92 |
| 3 | 28.320,19 | 7574.98 |
| 4 | 24.232,72 | 6375,42 |
| 5 | 25.835,92 | 7527,60 |
| 6 | 27.734,92 | 7531,41 |
| 7 | 22.931.24 | 6169,04 |
| 8 | 30.184,15 | 7778,35 |
| 9 | 24.553,75 | 6466,00 |

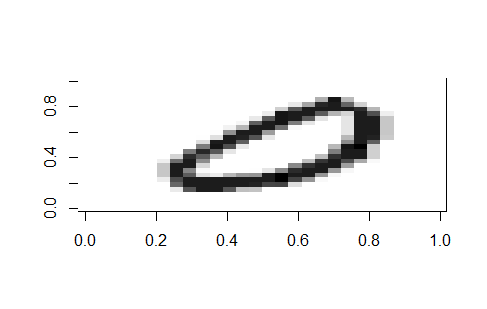
Figure 2 visualises the problem described above. The ‘boxes’ represent 50% of all density values for a specific class. For almost all classes, except digit 1, there is another class at the same level. This will product difficulties in classifying the observations. Additionally, the ‘whiskers’ of the boxes represent the other 50% of the data, barring the outliers. The variability of the observations make it logical that the value of density fluctuates within class. Factors like the style of handwriting, the pen used, ink colour or how much force is applied to the paper while writing can all influence the density of an observation. Figure 3 and 4 show two different digits 0 from the dataset. This further illustrates the data shown in figure 2 and table 2.

Figure : digit no.18

Figure 5 shows a stacked bar chart with the outcome of the classification using only density as feature.

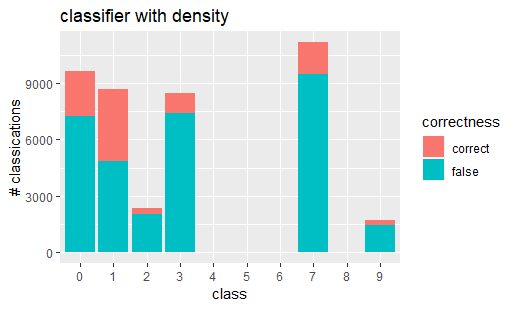


Figure 5: performance of density as a classifier

As can be seen from the figure, density does not perform well as a classifier. Four out of ten classes are not predicted at all, and those classes that are predicted are generally wrong. The predictions made based on table 2 and figure 2 seem to be correct. Digit 1 seems to be classified correctly the most often. The model was not able to discern the different classes with a mean around 24.000.

The accuracy of this model is 22,7%. Compared to the baseline performance of predicting the majority class, the performance has almost doubled. However, 22,7% is still farm from reliable.

## 3.2 Top-bottom ratio

In search of more distinctive feature to classify the data, a new feature call the ‘top-bottom ratio’ was created. The pixels in the pictures of the handwritten digits are ordered from left to right. Which means that the top and bottom half can easily be separated from each other. To create the top-bottom ratio, the sums of the pixel values of the first 387 pixels and second 387 pixels were calculated. This created a dataset with the sum of pixel value in the top half and bottom half for each observation. To obtain the ratio, the sum of the bottom half pixel values was divided by the sum of the top half pixel values. The calculation is shown in equation 1.

This creates a ratio where a value of >1 means more pixel value on the bottoms half, and a value of <1 means more pixel value on the top half. It is assumed that on average the digits are written in the middle of the paper, dividing the digits into two equal halves. Of course this assumption is not always satisfied, but it is assumed that it holds in general.

Table 3 and figure 6 show the mean ratio for each class and its standard deviations.

|  |  |  |
| --- | --- | --- |
| Class | Mean | Std. dev. |
| 0 | 0,925 | 0,085 |
| 1 | 0,904 | 0,075 |
| 2 | 0,703 | 0,138 |
| 3 | 0,973 | 0,146 |
| 4 | 0,798 | 0,172 |
| 5 | 0,983 | 0,174 |
| 6 | 0,665 | 0,120 |
| 7 | 1,218 | 0,220 |
| 8 | 0,969 | 0,108 |
| 9 | 1,006 | 0,196 |

Table 3: mean and standard deviation per class

Table 3 shows the same problem takes place with the top-bottom ratio as with density. Though there is more difference between the classes, some are still very similar. It also seems that the assumption that on average, the numbers are written in the middle of the paper is violated. Symmetrical digits should be very close to a mean of 1. However, this is not the case.

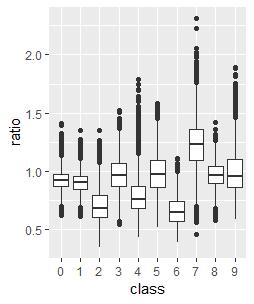


Figure : Boxplot of ratio

Figure 6 confirms that there is more diversity in ratio than there was in density between the numbers. There are still some very similar numbers however. Judging by their similarity in the boxplot, 3 and 5 should be very hard to distinguish for the model. Though the feature does show promise in distinguishing between the digits 2 and 7. This makes sense intuitively, because the biggest part of a 2 is on its bottom, whereas a 7 is more ‘top heavy’. Once again, the variability of the data hinders the ability of the model to correctly classify the data based on this feature. Handwriting plays a large part in the ratio of a digit. As well as where on the paper the digit has been written.

The results of a multinomial logit model with the top-bottom ratio as the only feature are shown in figure 7.

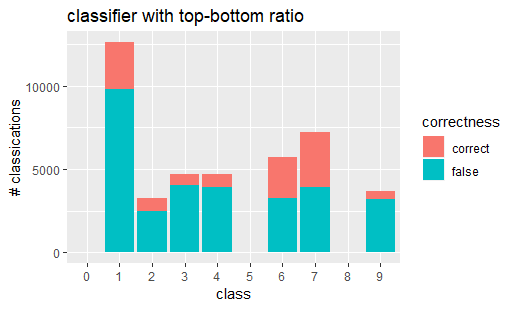


Figure 7: performance of top-bottom ratio as a classifier

Figure 7 shows an improvement in performance over density, but still doesn’t use all classes for classification. There are still 3

classes that are never predicted by the model, and if classes are predicted, then the predictions are mostly incorrect. The accuracy

of this model is 27,0%, which still makes the performance very poor. As was predicted, number 5 could not be distinguished from other classes. Likewise, 0 and 8 were too ‘average’ to stand out between the other predictors.

## 3.3 Density and Top-bottom ratio

This paragraph will discuss a multinomial logit model with both density and top-bottom ratio as features. To assess the relation between top-bottom ratio and density, a scatterplot was created. Figure 9 shows the relation between density and top-bottom ratio for all ten classes. From the scatterplot it can already be seen that there isn’t a strong correlation between the two features. This makes sense intuitively, as the amount of ink used, and the ratio between two halves of a number do not necessarily have anything to do with each other when writing a number. Especially in the lower left corner, the classes do not seem to show a clear distinction from each other. The only exception being digit 7, which seems to use a lot of ink, and has a large top-bottom ratio. Digit 1 shows a centralised clump at the bottom of the figure, but due to it being surrounded by other classes, it is not likely to be predicted often. It is also expected that digit 5 still won’t be predicted, as both features didn’t classify any observation as digit 5 on their own.

Figure 8 shows the result of using both features as a classifier for the Mnist dataset.

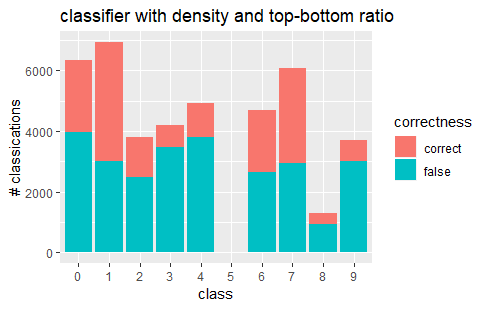


Figure 8: performance of ratio and density

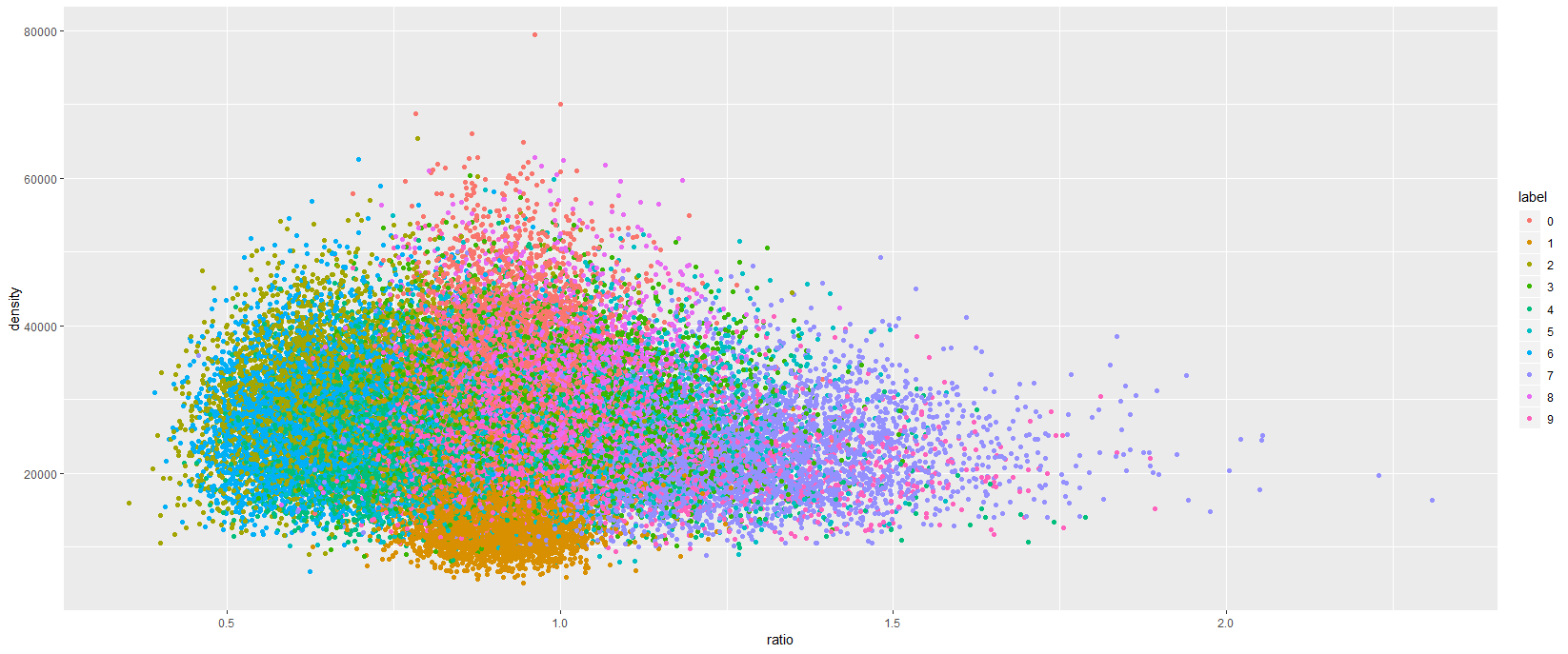
As expected, the model still doesn’t predict the digit 5. The combination of density and top-bottom ratio cannot distinguish 5 from any of the other classes. This also makes sense when looking at figures 2 and 6, where 5 is in the ‘middle of the pack’. Digit 1 is picked often, and classified correctly more than 50% of the time. This shows that the centrality of its positioning makes it more recognisable for the model, which makes it perform better on 1 than on the other numbers. The other example of this is 7. Its position in the scatterplot us more over to the right, making the outliers to the lower right of the chart more recognisable as a 7. Though there is still a lot of room for improvement.

Figure 9: scatterplot of density and ratio

However, the combined features have improved the performance of the model. The performance with the two features combined amounts to 37,3%. The performance still isn’t very good, but compared to the baseline, the accuracy of the model has risen by 26,15%.

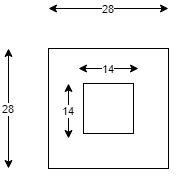
Based on these results, it becomes clear that adding more features to the model would be beneficial to its performance. However, it could also be beneficial to apply features that are better able to individually distinguish between different classes.

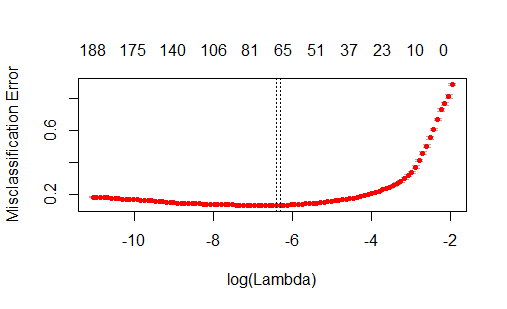
Figure 10: Image cropping

# 4. Data preprocessing

Due to restraints in resources, the Mnist dataset had to be cropped for the following chapters. The analyses ran with single features do not require high computing power. The algorithms in the next chapter, however, do require more power. Therefore, some measures were taken to reduce the time spent calculating results.

It should be noted that the following measures will impair the quality and generalisability of the models that result from running the algorithms.

First, the images were cropped in such a way that only the centre of the image was still used. This eliminates the mostly unused corners and edges of all images. It does however also delete important information for digits that do use these spaces. Cropping the image resulted in a reduction of features from 784 to 196. Figure 10 shows how the image was cropped.

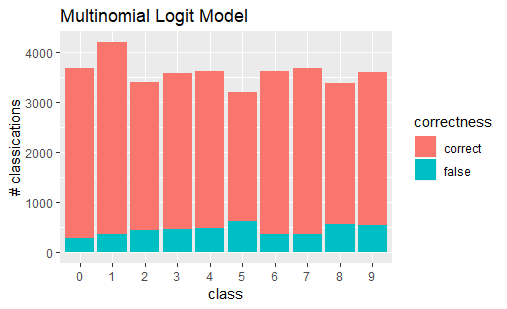
Furthermore, the training set contains 6005 observations out of the total 42.000. This creates a testing set of 35.995 observations. For replication purposes: the seed used for creating a random training set is 123.

Due to constraints in time and computational resources, the hyperparameters in the models have not been selected by use of cross-validation, but have been selected on their theoretical applicability to the data.

# 5. Multinomial logit with LASSO

Figure 11: Feature selection with lambda.1se

After the data was preprocessed, another multinomial logit model was applied to the data. However, this time a regularisation parameter was added. The regression was performed with the R statistics package ‘glmnet’. The regularisation parameter in the model is called lambda. Glmnet provides two standard values for the lambda value: lambda.min and lambda.1se.

The lambda value is a value that ‘punishes’ a model for its complexity. The higher the amount of features in a model, the higher its complexity. The lambda is a penalty for the amount of features in the model. The higher the value of lambda is set, the more the algorithm is incentivised to produce a model with a low level of features.

When lambda.min is selected, the algorithm selects the model which makes the least amount of classification errors. This means that this lambda value produces a model that best fits the training data. However, this might also lead to an overly complicated model which is prone to overfitting. The lamda.1se produces the simplest model, within one standard deviation from the value of lambda.min. This leads to a less intricate model, but also a model that might be more generalisable to testing data.

Figure 12: multinomial Logit Model

In this report, the lambda.1se rule is selected because of the small amount of training data relative to the testing data. A more general model might be more applicable to a large amount of testing data, while overfitting or bias might occur in a small training sample.

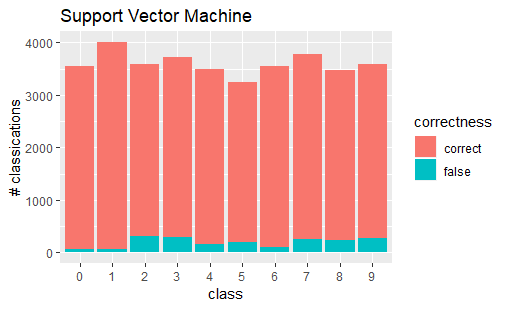
Figure 11 shows the features selected with lambda 1.se. This shows that out if the 194 features used, around 66 features are used to predict the final classes.

Figure 12 shows the outcome of the multinomial logit model. The performance of this model is much better than the performance with one or two features. This makes sense, because this model uses around 66 features to predict the class of the observations. It is interesting to see that the most errors are still being made on the number 5. This was also the case when using density and top-bottom ratio. The accuracy if this model is 87,3%, which is a significant improvement on predicting the majority class.

# 6. K-nearest neighbour



# 7. Support vector-machine

Support Vector machines is an algorithm that can be used for classification problems. It plots all features in an n-dimensional space. N, in this case, is the number of features entered into the model. It attempts to find a line, or vector, that separates the classes from each other. Support vector machines usually are two-class classifiers. Therefore there are additional techniques needed for multiclass problems. Two of these techniques are one-versus-many , and the one-versus-one. In one versus many, the support vector machine attempts to separate one class from all other classes. In one versus one, the support vector machine trains many models that compare two vectors with each other. The one versus many approach requires less models to be estimated and is therefore more efficient. However, it does experience some drawbacks that the one versus one method does not.

The R package used is called e1071. This package contains a function called svm (). This package, as a default, uses the one versus one method for multiclass problems. The results in this chapter are therefore a result of this approach.

One of the hyperparameters for support vector machines is called the kernel. As said before, the SVM algorithm attempts to find a line that separates the classes from each other. Sometimes this isn’t possible in a two dimensional plane. In that case, the kernel function can add additional dimensions to the equation, which does make it possible to draw a line that divides the two classes. The default in the used library is a radial kernel, which is also known as a Gaussian kernel.

Another important hyperparameter in the SVM algorithm is the C value. This value in SVM acts much like the lambda value in the multinomial logit model. When creating a support vector that separates the two classes, the C parameter punishes mistakes. A low value of C allows for more mistakes in the classification of the training set, resulting in a model that is less accurate on the training set, but might be more generalisable. A higher value of C allows the SVM to make less mistakes while creating the support vector, creating a stricter model that is prone to overfitting. The goal is to find a ‘sweet spot’ where the C parameter is neither too strict, or to lax. This creates a model that finds a balance between generalisability to new data, and applicability to the training data. As a default, the C parameter is set to a value of 1.

Figure 13: support vector machine

Figure 13 shows the performance of the Support Vector Machine. The performance of the model improves upon the performance of the regularised multinomial regression. It has an accuracy of 94,5%, which eclipses the logit model by 8,2% and the baseline by 83,35%.

# 8. Conclusion