Pattern Recognition: Handwritten Digits

Jaspreet Singh, 3754022 Daniël Stekelenburg, 4153286

December 19, 2016

1 Exploratory analysis

From looking at the summary of the data set, we conclude that some predictors always have a value of 0. Taking these predictors into account when classifying an observation would be a waste of time, so we can neglect these variables.

When applying the summary function, we observe that some pixels are never used. The five given values are all 0. These pixels are positioned around the borders of the image. By constructing a surface plot for the means per pixel (Figure 1), we see an offset to the right and bottom relative to the center of the image. The variables that are never used (max = 0) will be considered as superfluous. These will be filtered out of the data. We also observe that for many variables Q_1, Q_2 (median) and Q_3 are equal to 0. For the pixels lie in the elevated area of the surface plot we see that Q_2 and sometimes Q_3 have a value greater than 0. The percentile ranks are not used to filter out variables because they tell very little about the contribution of a variable to the classification problem. The some goes for the mean of the variables. A mean with a value 0.00 might be caused by that a variable for a certain digit has a low value, by taking the mean and by rounding this value might become 0. This is why looking at the maximum tells the most about superfluous variables. This results in that 76 variables are ignored. The location of these variables is at the borders and corners of the image.

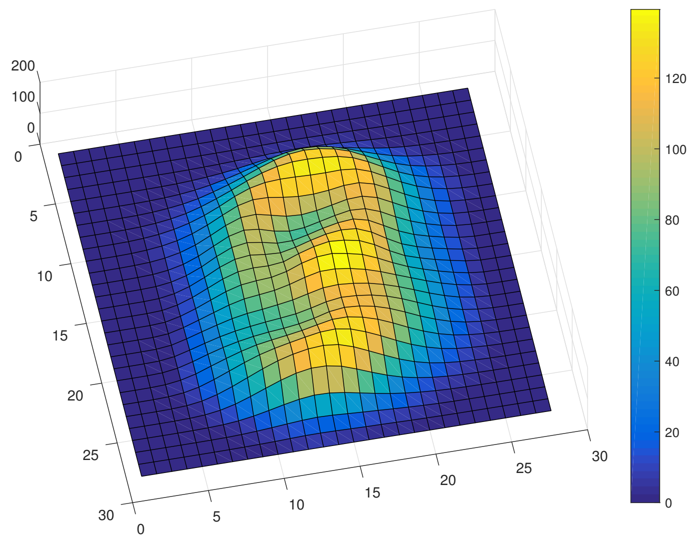
Figure 1: Surface plot for the mean pixel values. Blue indicates low mean pixel values, orange/yellow indicates higher mean pixel values

From Figure 2 we read that the class '1' is the majority class. If we simply predict the majority class, the the percentage of correctly identified digits (accuracy) is 11.2%.

Before we go further and dive into the performance of different classification methods for this classification problem, we first want to explain our way of validating a classifier. The complete data set we obtained, consists of 42000 observations. However, because we lack computational resources to use this complete set to train and test on, we will be using a subset of 1000 observations for training and a subset of the same size to test our classification method on. These samples are drawn randomly and the samples cannot contain the same observation. For every classification method in this study, we will use the training set to train the model and for cross-validation, and the test set for testing.

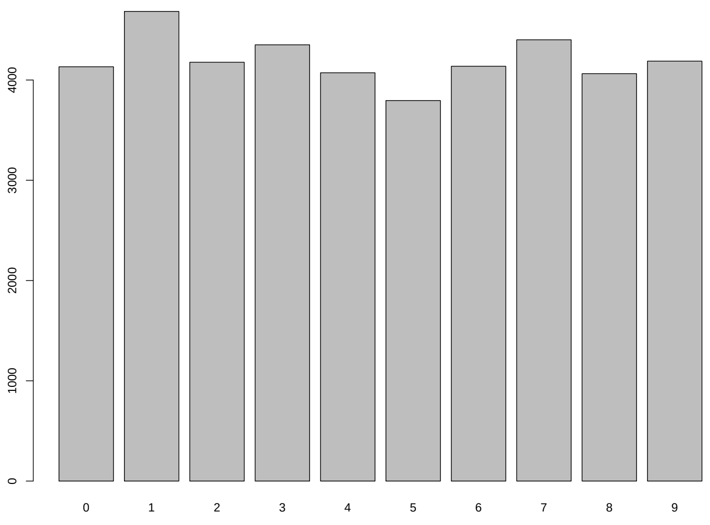

Figure 2: Histogram of number of occurrences per class label.

2 Features

In this section the features derived from the raw pixel data are explained. The method of extracting the features is explained as well as their performance is measured by applying the multinomial logistic regression model, the accuracy, and an explanation is given why these features might help to discriminate between the classes.

2.1 Pixel Density

To calculate the pixel density of each class, we add a feature to the data set, which we simply call 'density'. This feature is the average amount of ink that is used for a pixel and is simply calculated by using the *rowMeans* function in R. We compute this feature value for every observation and get the following results (Figure 3).

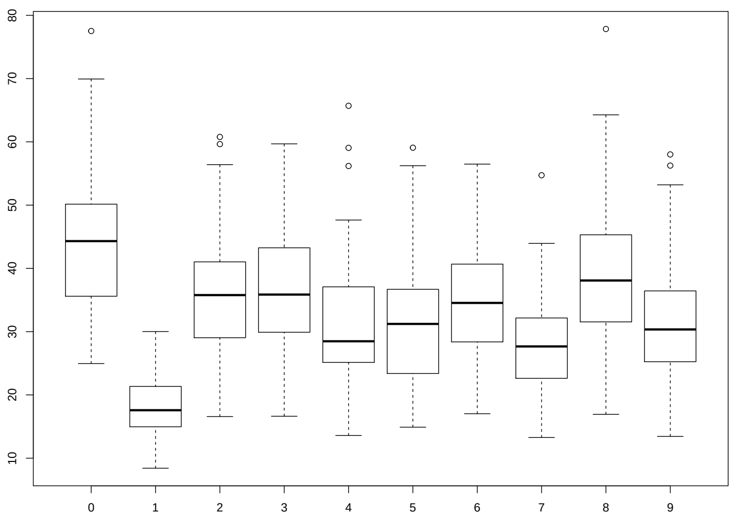

Figure 3: Density boxplot per class

We create a multinomial logistic regression model on the training set and use this model to predict the labels in the test set. This prediction results in the following confusion matrix (Table 1) when we try to predict the label of a digit using this model.

	0	1	2	3	4	5	6	7	8	9
0	48	2	0	26	0	0	0	2	0	12
1	0	83	0	0	0	0	0	19	0	7
2	32	8	0	32	0	0	0	7	0	21
3	27	8	0	38	0	0	0	8	0	27
4	5	22	0	25	0	0	0	18	0	28
5	17	9	0	26	0	0	0	15	0	31
6	20	18	0	34	0	0	0	11	0	27
7	5	19	0	14	0	0	0	18	0	42
8	28	2	0	34	0	0	0	6	0	20
9	7	16	0	23	0	0	0	18	0	35

Table 1: Confusion matrix when using a multinomial logistic regression model on the test set. This model tries to predict the class of a digit, only using the density-feature as a predictor.

When you look at Table 1, you can see that the model does not predict very well. A perfect prediction would only have non-zero values at the diagonal. However, this matrix contains a lot of non-zero values outside the diagonal. Also, some columns only contain zero's, which means that the model did not predict this class at all. In our case, the model didn't predict a single 2, 4, 5, 6 or 8.

We calculate the accuracy of the prediction on the test data by using the following equation:

$$accuracy(T) = sum(diagonal(T))/sum(T) \tag{1}$$

In this case, T is Table 1. The test set accuracy of this classification model is 22.2%.

We can conclude that this model predicts the class of digits poorly but it is better than guessing or assigning to the majority class.

2.2 Number of white area planes

To improve our classification accuracy, we introduce a second feature, which is the number of white space regions in a digit. These regions are also called planes. We have chosen to use this feature, because you can distinguish groups to numbers when you know the number of white area planes. Take a look at Figure 4.

To improve the accuracy, we introduce a second feature which is the number of white space regions in an image. These region are called planes. This feature allows one to discriminate the digits based on the number of planes. Figure 4 gives the distribution of number of planes per digit in the training set. We see that digits that contain circles have a higher number of planes.

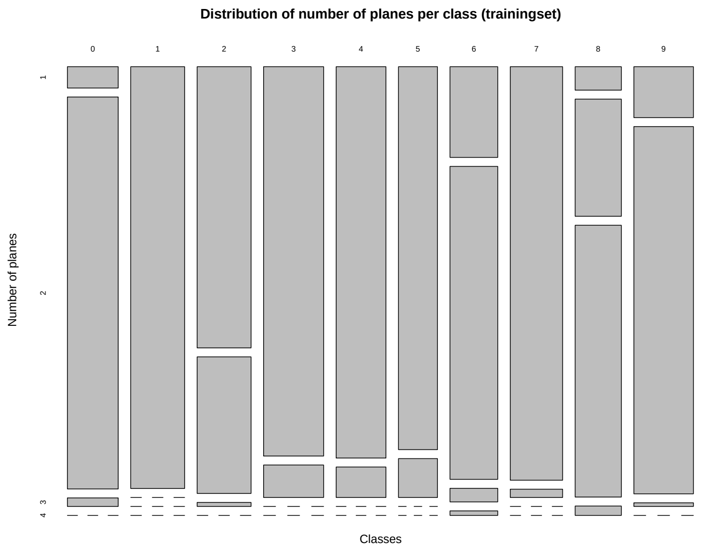

Figure 4: Distribution of number per planes per class label

This feature is implemented by repeatedly using breadth-first search to visit all white pixels. The number of times BFS is used, is the number of planes. From the top left pixel we start until we have all visited all pixels in the plane. Then we search for the next unvisited white pixel. This indicates that the pixel lies in another plane. This is repeated untill all white pixel have been visited.

Let's compute the accuracy of a prediction which uses this feature, like we did in Section 2.1.

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	4	0	0	0	0	2	84
1	0	0	0	109	0	0	0	0	0	0
2	0	0	0	68	0	0	0	0	2	30
3	0	0	0	98	0	0	0	0	1	9
4	0	0	0	94	0	0	0	0	1	3
5	0	0	0	93	0	0	0	0	0	5
6	0	0	0	28	0	0	0	0	5	77
7	0	0	0	97	0	0	0	0	0	1
8	0	0	0	4	0	0	0	0	67	19
9	0	0	0	19	0	0	0	0	3	77

Table 2: Confusion matrix when using a multinomial logistic regression model on the test set. This model tries to predict the class of a digit, only using the planes-feature as a predictor.

Now we know the results of the prediction (Table 2, we again can compute the accuracy using Equation 1. The accuracy of this prediction is 24.2%, which is slightly better than the accuracy of the prediction based on the pixel density. You can see that this model only predicted digits 3, 8 and 9. Since the planes-feature only has a few different occurring values, it is impossible for the model to distinguish digits which have the same number of white regions. In this case, the model roughly predicts every digit with 1 white region as a 3, a digit with 2 white regions as a 9 and a digit with 3 white regions as an 8.

Plane-feature value	Occurrences in test set	# of predicted digit
1	614	614 (digit 3)
2	305	305 (digit 9)
3	79	81 (digit 8)
4	2	-

Table 3: Plane-feature value occurrences in the test set and the predicted values

From Table 3, we see that there are 614 digits where the number of recognized white area planes is 1, 305 digits where this number is 2 and 79 digits where this number is 3. When we compare these numbers to the number of predictions of a 3, 8 or 9 in Table 1, we should see a rough match. As you can see in Table 3, the values of the second and third row are almost exactly the same. Apparently, there are two digits who have 4 white regions. These digits are predicted as an 8, which explains the difference on the third row.

2.3 Using both features are predictors

When we use a multinomial logistic regression model where we base our classification on botch features, the pixel density and the number of white area planes, we retrieve the following result when training this model on the training set and applying the model to the test set.

	0	1	2	3	4	5	6	7	8	9
0	61	0	0	4	0	0	0	0	2	23
1	0	84	0	1	0	0	0	24	0	0
2	13	6	0	51	0	0	0	11	2	17
3	3	8	0	74	0	0	0	16	1	6
4	0	21	0	37	0	0	0	36	1	3
5	3	9	0	53	0	0	0	31	0	2
6	21	2	0	18	0	0	0	8	5	56
7	0	18	0	37	0	0	0	42	0	1
8	8	0	0	3	0	0	0	1	67	11
9	17	3	0	5	0	0	0	11	3	60

Table 4: Confusion matrix when using a multinomial logistic regression model on the test set. This model tries to predict the class of a digit, using both features as predictors.

The accuracy of this prediction, using Equation 1, is 38.8%. We expected that using both features as predictors would improve the accuracy, which it did.

2.4 Bounding box aspect ratio

Before we are going to study other classification methods, we first want to explain another feature we have implemented. For every digit in the training- and test set, we have computed the usage of the the bounding box. Let B denote the set of pixels in the bounding box and v_i the value of the i-th pixel, then we get:

$$\frac{\log(\sum_{v_i \in B} v_i)}{|B|}$$

In order to compute this aspect ratio, you have to know the width and the height of the digit. We simply loop through the 28x28 pixels, and find the first and last row containing a non-white pixel (i.e. with a value greater than 0) and find the first and last column containing a non-white pixel. When you have found these rows and columns, you can compute the width and height of the digit and get the aspect ratio by dividing the width by the height. The result of applying this to the trainingset can be seen in Figure 5

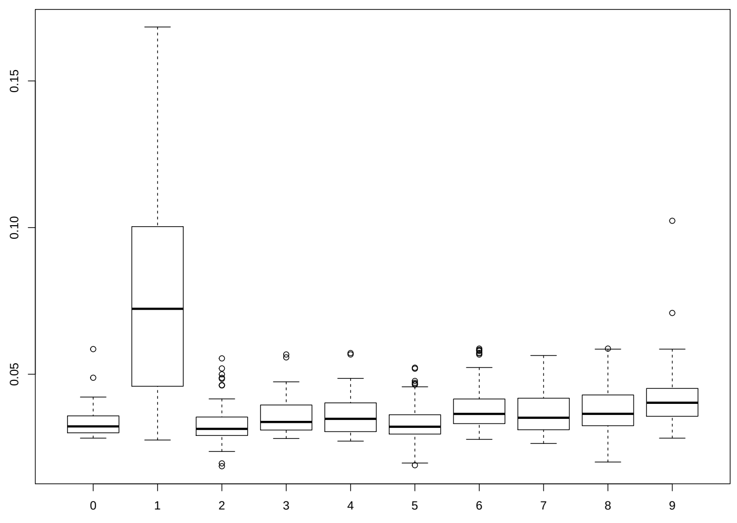

Figure 5: Bounding box ratio boxplot per class

The accuracy of the prediction of a multinom logit model by using this feature as the only predictor, is 22.4%. The accuracy when using a multinom logit model where you use the three features explained as predictors is 40.8%. We can conclude that using more features as predictors for predicting the label of a digit results in higher accuracy.

3 Classification Methods

In this section, we will explain multiple classification methods. Each model will be trained on the same training set and tested on the same test set. The three features explained in Section 2 will be used as predictors. On top of that we will also look at the performance of the three classifying methods by using the raw pixel data. This will give us an indication on how well this data is contained on the used features. Training and testing in this case will be performed in that exact same way as training and testing the features.

3.1 Regularized Multinomial Logit Model (using LASSO penalty)

Plotting the lambda against misclassification error results in Figure 6. The best value for λ seems to be 0.01196303. Following from the given Figure 6, this value for λ is not optimal, but rather a heuristic choice. It produces a less complex model is in terms of estimated expected generalization error within one standard error of the minimum, which is in this case $\lambda = 0.02494707$. The misclassificiation rate does not decrease significantly as we increase complexity of the model. Thus it is better to stick with the less complex model.

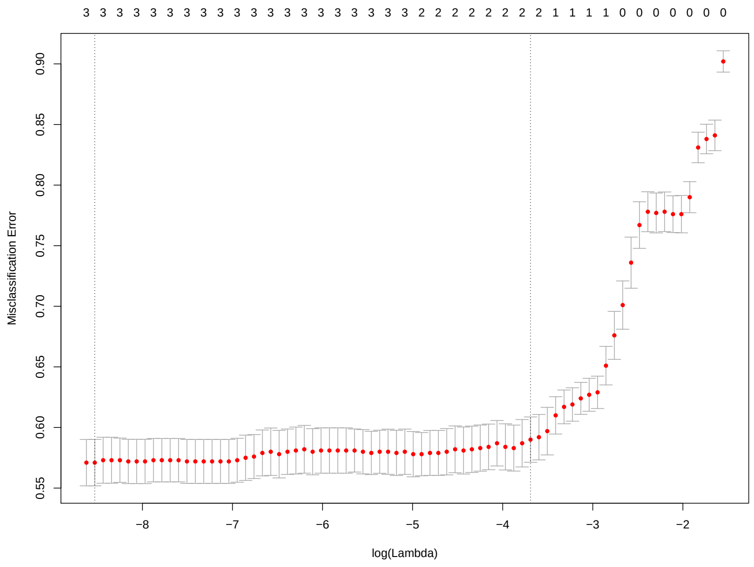

Figure 6: Lambda against misclassification error for the features.

We do the same for the raw pixel data and get a lambda of $\lambda = 0.009931918$ from Figure 7. By comparing Figures 6 and 7 it seems that the model has a lower misclassification rate on the raw pixel data overall already.

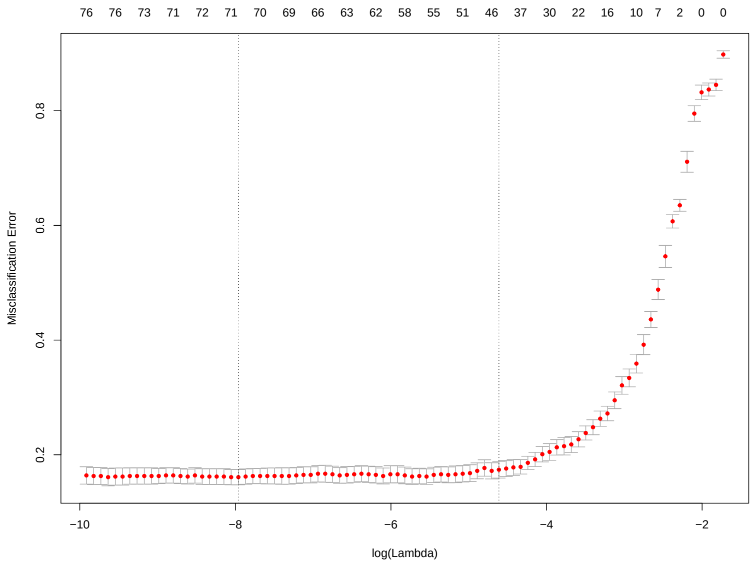

Figure 7: Lambda against misclassification error for the pixels.

After applying the lasso-variant, we get the following confusion matrix (Table 5) for the feature data.

	0	1	2	3	4	5	6	7	8	9
0	54	0	0	4	0	0	0	0	2	30
1	0	85	0	4	0	0	0	20	0	0
2	10	2	0	56	0	0	0	10	2	20
3	4	6	0	82	0	0	0	9	1	6
4	0	11	0	48	0	0	0	35	1	3
5	4	3	0	57	0	0	0	32	0	2
6	18	6	0	22	0	0	0	4	5	55
7	0	8	0	52	0	0	0	37	0	1
8	8	0	0	3	0	0	0	0	67	12
9	11	3	0	9	0	0	0	9	3	64

Table 5: Confusion matrix when using a multinomial logistic regression model (using LASSO) on the test set.

This prediction has an accuracy on the test set of 20.7%. Based on the three features, the multinomial logistic regression model (using LASSO) is not able to distinguish between digits very well. Most wrongly classified digits are classified to the class '3'.

If we now apply the same method to the raw pixel data, we an accuracy of 83.9%. Because the raw pixel data contains more variables than the data with only the calculated feature values, the LASSO method is much better at penalizing in high-dimensional space because it is easier to find variables which increase the misclassificiation rate. This might explain the difference in performance.

3.2 K-Nearest Neighbors

In this section, we will be looking at the performance of k-nearest neighbor classifiers. In Figure 8, we have plotted the relation between k and the accuracy of the classifier on the test set. We have tested kNN-classifiers with k from 1 to 100. Figure 8 shows that a higher value for k by using *leave-one out cross validation* leads to the lower test set accuracy. Thus, we have looked at the results for a classifier with a low value for k and concluded that a classifier with k valued from 1 to 3 resulted in the best test set accuracy, where k=1 is the winner with an accuracy of 96.6% on the test set.

	0	1	2	3	4	5	6	7	8	9
0	88	2	0	0	0	0	0	0	0	0
1	0	108	1	0	0	0	0	0	0	0
2	0	0	99	1	0	0	0	0	0	0
3	1	0	2	102	3	0	0	0	0	0
4	0	0	0	2	96	0	0	0	0	0
5	0	0	0	0	7	88	3	0	0	0
6	0	0	0	0	0	0	107	2	1	0
7	0	0	0	0	0	0	2	96	0	0
8	0	0	0	0	0	0	0	2	85	3
9	0	0	0	0	0	0	0	0	2	97

Table 6: Confusion matrix when using a K-Nearest Neighbors model on the test set.

The confusion matrix shown in Table 6 shows that most misclassification might be because of outliers and noise (irreducible error) in the data. We see a very low number of missclassifications. This means that the three features summarize the 700+ variables from the raw pixel data very well in this case.

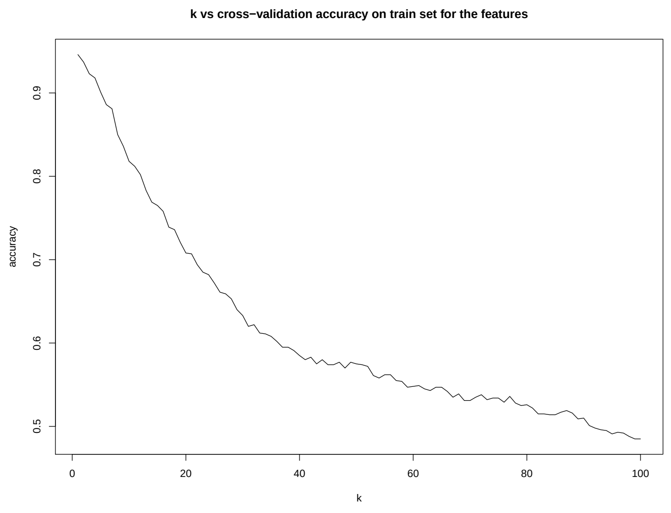

Figure 8: k-Nearest Neighbors accuracy plot for the features.

The same is done for the raw pixel data, the results of the cross-validation against k are plotted in Figure 9. In this case $k = 1$ comes out as the winner with highest accuracy on the cross-validation training set, with an accuracy of 88.6%. The accuracy on the test set is 88.8%.

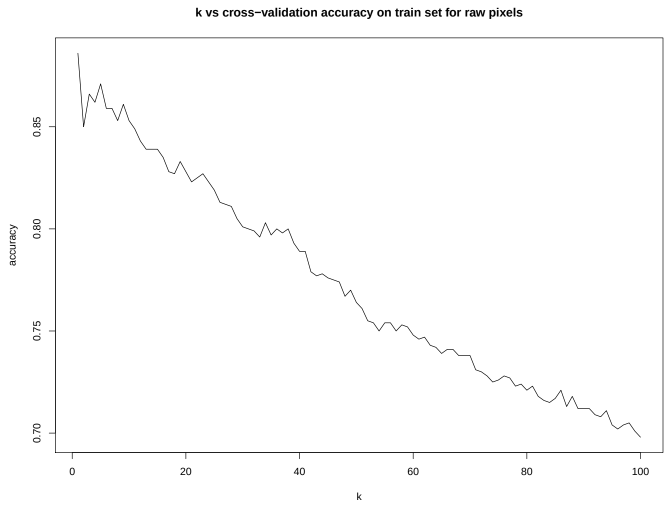

Figure 9: k-Nearest Neighbors accuracy plot for the pixels.

In this case the classification method based on the calculated features performs better than classification based on the pixel data. One of the reasons is that the 'curse of dimensionality'. This means that in higher-dimensional space neighbor tend to be far away from each other. In this case we have 4-dimensional versus a 709-dimensional space. So a decrease in performance is to be expected when using high-dimensional data in kNN.

3.3 Support Vector Machines

In this section the results of applying support vector machines (SVM) will be shown. We will look at two kernel types, linear and radial. The optimized parameters have been obtained by using k-fold cross validation.

When we apply svn with a linear kernel on the computed features, before optimizing the parameters, the accuracy on the test set is 40.8%. The default value for the cost parameter is cost = 1. After using 10-fold cross validation to find the optimal cost, we obtain the optimal value for the cost parameter, cost = 11. After using this value in the SVM, we get an accuracy of 51.1% on the test set. The results are shown in Table 7. We see that it is hard for the SVM to differentiate between digits. This might be caused by the data, which is not entirely separable.

	0	1	2	3	4	5	6	7	8	9
0	62	0	0	4	0	0	6	0	2	16
1	0	82	0	1	0	0	0	26	0	0
2	17	1	0	44	0	0	3	23	2	10
3	3	4	0	71	0	0	1	23	1	5
4	0	3	0	35	0	0	1	56	1	2
5	3	0	0	46	0	0	0	47	0	2
6	22	1	0	18	0	0	5	9	5	50
7	0	7	0	27	0	0	0	63	0	1
8	8	0	0	2	0	0	1	2	67	10
9	15	1	0	5	0	0	5	13	3	57

Table 7: Confusion matrix when using a SVM model with a linear kernel on the test set.

We apply the same method as above to the raw pixel data. After using 10-fold-cross validation, we obtain the optimal value for the cost parameter, cost = 1, which results in an accuracy of 10.8% on the test set. Everything is classified as a '3'.

For the radial kernel before optimizing the parameters, accuracy on the test set is 41.0%. The default value of the cost parameter is cost = 1 and the default value for the γ-parameter is $\gamma = 0.01$. After using 10-fold cross validation to find the optimal cost and γ, we obtain the values cost = 8 and $\gamma = 0.1$. After using these values in the SVM, we get an accuracy of 50.6%. Here the accuracy also decreases in stead of increasing. The results are shown in Table 8. We see that it is hard for the SVM to differentiate between digits. This might because of that the data is not entirely separable.

	0	1	2	3	4	5	6	7	8	9
0	61	0	0	4	0	0	1	0	2	22
1	0	82	0	1	0	0	0	26	0	0
2	16	1	0	45	0	0	1	22	2	13
3	3	4	0	71	0	0	1	23	1	5
4	0	3	0	35	0	0	0	56	1	3
5	3	0	0	50	0	0	0	43	0	2
6	21	1	0	18	0	0	1	9	5	55
7	0	4	0	27	0	0	0	66	0	1
8	9	0	0	2	0	0	0	2	67	10
9	15	1	0	5	0	0	0	13	3	62

Table 8: Confusion matrix when using a SVM model with a radial kernel on the test set.

We apply the same method as above to the raw pixel data. After using 10-fold-cross validation, we obtain the optimal value for the cost and gamma parameters, cost = 1 and $\gamma = 0.00390625$, which results in an accuracy of 10.8% on the test set. Everything is classified as a '3'.

We see that the performance decreases on the test set when using the raw pixel data. An explanation is that the data because less separable when using the raw pixel data. We see for the raw pixel data in both cases that all test cases result in the class '3' being assigned.

4 Discussion

In this section we discuss the findings of applying the various classification methods discussed in this report. The results are summarized in Table 9.

Classification Method	**Parameters**	**Accuracy on test set**
Multinomial log. regr.	Density only	22.2%
Multinomial log. regr.	Planes only	24.2%
Multinomial log. regr.	Bounding box only	22.4%
Multinomial log. regr.	Density + Planes	38.8%
Multinomial log. regr.	Planes + Bounding box	36.7%
Multinomial log. regr.	All features	40.8%
Regularized multnom. log.	LASSO penalty, $\lambda = 0.02494707$	20.7%
K-nearest neighbors	k = 1	96.6%
Support vector machine	linear kernel, cost = 11	40.7%
Support vector machine	radial kernel, cost = 50, $\gamma = 0.01$	41.0%

Table 9: Summarized results for the accuracy per classification method and parameters based on the features.

Classification Method	**Parameters**	**Accuracy on test set**
Majority class	class = '1'	11.2%
Regularized multnom. log.	LASSO penalty, $\lambda = 0.009931918$	83.9%
K-nearest neighbors	k = 1	88.8%
Support vector machine	linear kernel, cost = 1	10.8%
Support vector machine	radial kernel, cost = 1, $\gamma = 0.00390625$	10.8%

Table 10: Summarized results for the accuracy of per classification method and parameters based on the raw pixel data.

As you can see, most classification methods based on the implemented features, result in a higher accuracy on the test data than the methods based on the pixel data. However, regularized multinomial logistic regression and the support vector machines perform better when trained on the raw pixel data than on the implemented features.

In both cases we see that SVM's perform worse than kNN. An explanation for this is that the data is non-separable. This results in that not a good margin and fit for the hyperplane can be found in the case of SVM's, thus resulting in a worse performance. In both cases a low value for k has been found when using kNN. This also indicates that the data is not clustered, which also explains why SVM's do not perform that well. A reason might be the quality of the features. As seen in Section 2, we see that the mean and quantile values lie closely together for the density and bounding box ratio. This makes it harder to distinguish between characters. This is also the reason for the low performance by applying the multinomial logit regression model and the regularized multinomial logit regression model with LASSO penality. With the multinomial logit regression model we see that increasing the number of features increased the performance. The multinomial logit regression model with LASSO penality also performs poorly on the features because it cannot decrease the weight of a low-dimensional space without losing too much information. However, it performs very well when using the raw pixel data because it is able to decrease the weights of the variables that can be considered as noise or do not contain a lot of information.

From the confusion matrices of the multinomial logit models we have tested, we see that these models don't predict a 2, 4, 5. We can conclude that this error is caused by our features, which don't model the difference between certain sets of digits very well. As a result, the classifier often chooses a 3, 7 or a 9. SVM seems to have the same problem. Also, SVM often predicts a 6 as a 9.

From the confusion matrix of kNN, we see that this classifier does not make as many mistakes as the other methods. Comparing multinomial logit and SVM to kNN, we can conclude that kNN is overall the best classification method for predicting a digit from the given data set.