ML135 - PROJECT 01

Submitted By – Priyanshi Somani

Part One: Logistic Regression for Digit Classification

1. Logistic regression model with the *max*_iter set to *i*, where *i* = 1,2,3….40

Figure 1.1

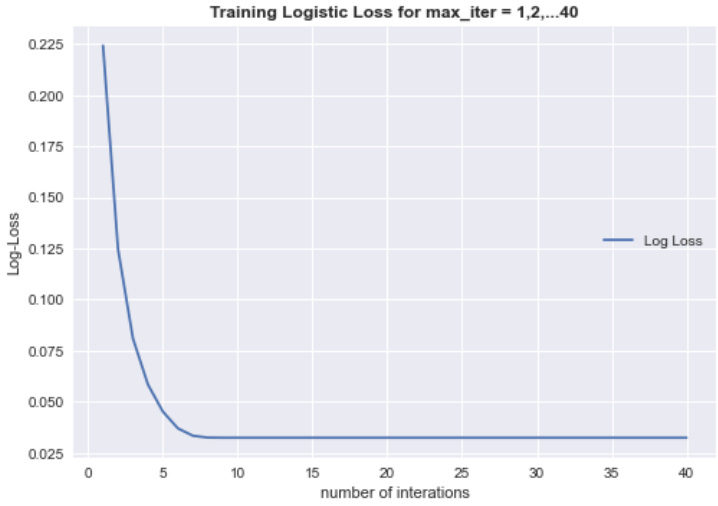


Figure 1.1

Discussion – Max iterations is the number of iterations the solver will take to converge. It also helps us understand the complexity of the model, by giving the features the right weights. Here in Figure 1.1 we can see that when max iterations is 1, the model has the lowest accuracy and highest lost. As we increase the number of iterations for the model to converge, we can see that the accuracy is improving and log loss in decreasing. However, after 7 to 8 iterations the model does not improve at all. The training accuracy and log loss is more or less the same after 7 to 8 iterations. If increase the iterations, we might be at a risk of overfitting the model to the training data.

2. Weight applied to feature *pixel000* with the *max*_iter set to *i*, where *i* = 1,2,3….40

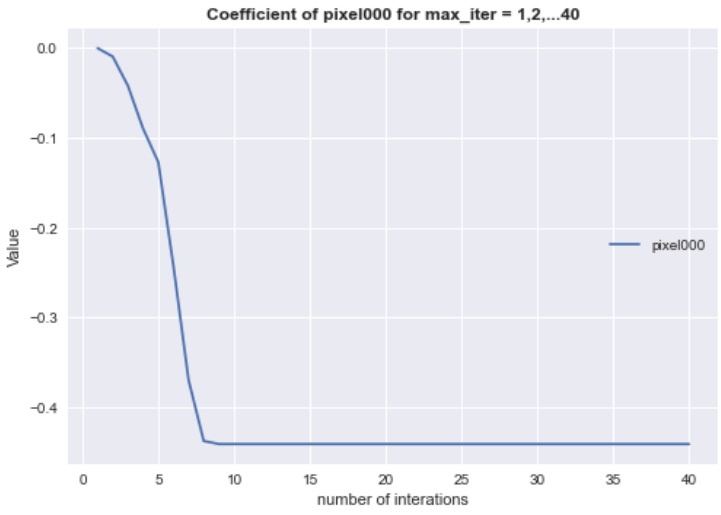


Figure 2

Discussion – We can see the coefficient of pixel000 going down as we increase the number of iterations. This is because by increasing the number of iterations the model gets more iterations to find the best value of coefficients. At max_iter=1, the model predicted a higher importance to pixel000, however after more iterations, the model learns that pixel000 has negative correlation with the result. Hence, we can see that changing the iterations affects the complexity of the model.

3. Effect of Regularization on Logistic Regression Model

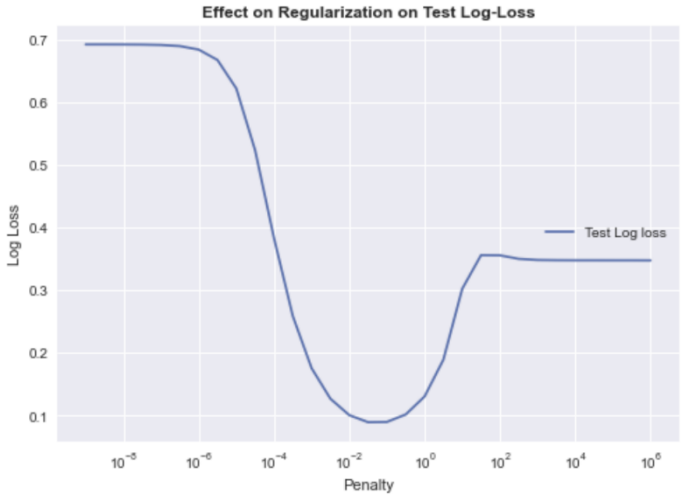


Figure 3.1

```
Minimum Test Log Loss 0.089690
Regularization Value 0.031623
TEST ACCURACY: 0.967221
Confusion Matrix
Predicted    0    1
True
0          942   32
1           33  976
```

Table 1

Discussion – Regularization is the penalty we apply to the model, when the model is too complex. Regularization parameter is inversely related to penalty. Hence, we can see, that when there is high regularization, that means none of the features are given importance the model has a log loss of 0.7. This is the highest loss because no features are used in the model. The model is underfitting here. As we increase the parameter value (decrese regularization strength) the model starts giving features some weight, hence better fits the data and log loss decreases. As we increase parameter value, decrease regularization strength, we can see that the log loss is increasing again. This is because model has no penalty on a highly complex model. Hence after reaching the minimum log loss the model has started overfitting. We can see the best regularization is at 0.03 value, which is lower than 1, which means model is penalised for complexity and this is the best model with log loss 0.089.

4. Misclassifications made by the model –

a) Image of 8, model predicts 9

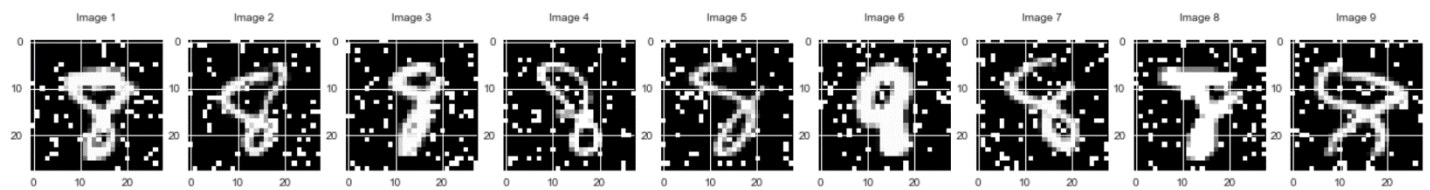


Figure 4.1

b) Image of 9, model predicts 8

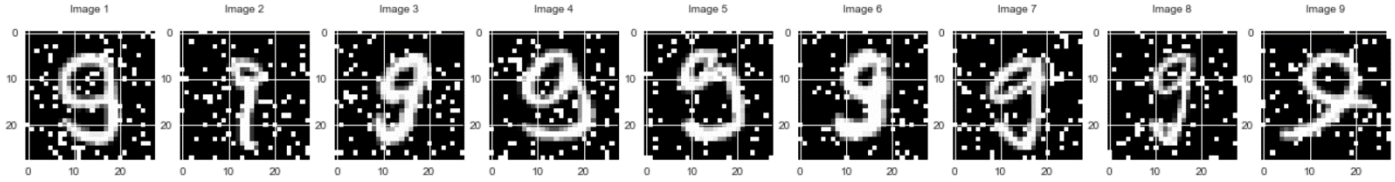


Figure 4.2

Discussion – From Figure 4.1, we can see that the model is labelling a few images of 8 predicted as 9. These are false positives. This could be because, in a few of the images the two circles of 8 are not distinguishable (like Image 3 or Image 6) or because the one of the circles is not complete (like Image 5 or Image 9). We can also see for some images of 8, when the thickness is more the model is again not able to classify well.

From Figure 4.2, we can see the model is labelling a few images of 9 as 8, these are false negatives. From the above images we can see, for images of 9 where the bottom curve is either too big (like in image 4 or image 1, where the bottom curve of 9 is almost touching the top circle.

5. Coefficients of Logistic Regression Model

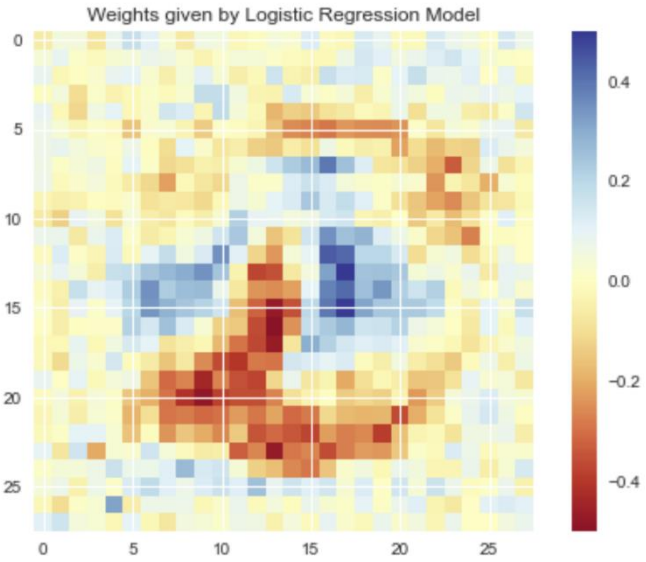


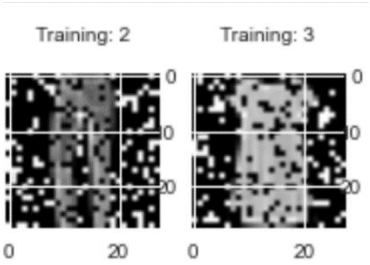
Figure 5

Discussion – The above plot shows the coefficient values assigned by the model, on the raw pixels. The colour bar on the right shows that features that are close to red, are likely to features that contribute to labelling an image as 8, and features that are close to blue are features that contribute to labelling an image as 9. The above representation is quite accurate. The image is of size 28x28. For a image with 8, the bottom pixels i.e. 14:28 would be populated, whereas for 9, they might not be so much. Furthermore, we can see how the blue is mostly on the outside, that is close to boundary whereas red is highly concentrated in the middle, because 8 has 2 wholes in the middle and a boundary around it.

Part Two: Trousers v. Dresses

1. Feature Transformations -

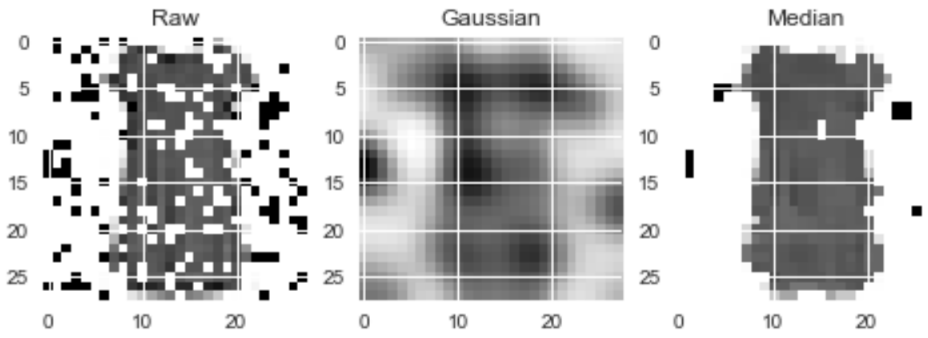
I first set up a Logistic Regression classifier with k-fold cross-validation and run the model using the raw data. The error rate of this model is 0.063, relatively higher than other models. This is because no pre-processing is currently done on the data. The raw images given for training have a lot of noise in it as shown below.



Therefore, in order to remove the noise from the image I applied a blurring filter and then sharpened the image. In this process I tried a few filters supported by python like Gaussian, Laplace, Median, Unifrom filter and more.

Gaussian Filter – In order to remove the noise, I blurred the image using a gaussian filter of strength 3 and then sharpened the image using filter of strength 3. However, gaussian filter did not give a good result, because it does not preserve the edges well. Hence, I used the median filter instead.

Median Filter – I removed the noise in the raw image, using a median filter. This filter performed better than the Gaussian filter as shown below –



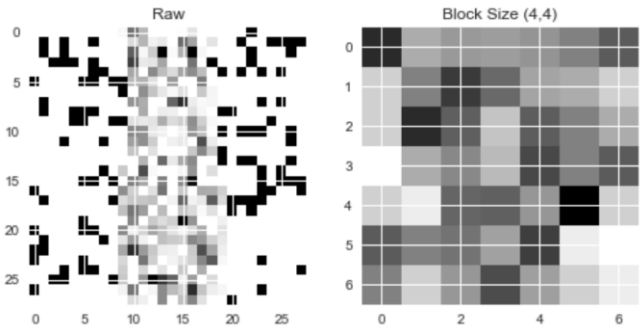
The above are images after applying gaussian filter and median filter for removing noise and sharpening the image. As shown in the summary table, upon applying the Gaussian filter on the raw data, the model performs worse than its performance on raw data. On the other hand, applying median filter has decreased the error rate to 0.038.

I tried image processing techniques to highlight the edges like the sobel filter along the x and y axis. However, these were not were helpful to reduce the error rate. Instead they only preserved the edges because of which the error rate increased. Hence I did not choose this filter.

2. Adding Features –

a) Extracting special information from images – I calculated the mean pixel value of blocks of varying sizes like (4,7) (7,7) (4,4). This would help understand what parts of trouser/dress images are darker

and which are lighter. I added these mean values are additional features. Below is an example of a block mean of a raw image -



b) Features that count overall numbers of white or black pixels – I calculated the number of black, white and grey pixels. I then divided each of them with the total number of pixels (i.e 784) in order to standardise these features. However, looking at the table adding this feature does not improve the overall performance of the model. Hence, I did not use this feature in the final model.

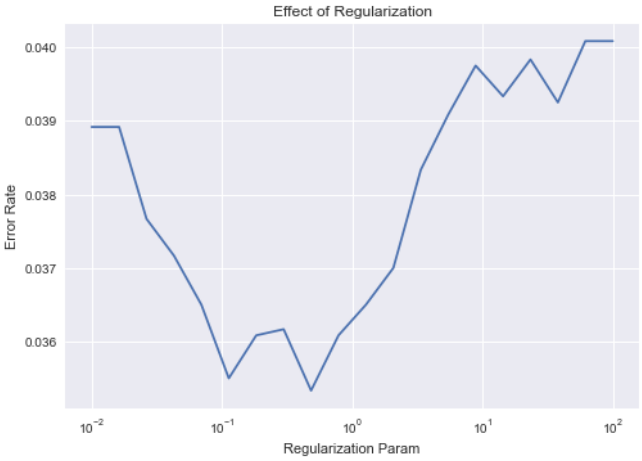
3. Model Tuning –

a) K-fold Cross validation – I used 10-fold cross validation to select model with best accuracy. Since I was trying to reduce the error rate, I chose the metric to compare in validation as accuracy. Then I averaged the scores on all 10 folds. Before using 10-fold cross validation I was getting a very high training accuracy which means model was overfitting on training data. After using 10-fold cross validation, my accuracy expectedly dropped.

```
TRAIN ACCURACY: 0.960917
Confusion Matrix
Predicted     0     1
True
0          5813   187
1           282  5718
```

```
TRAIN ACCURACY mean of 10-fold CV 0.951
```

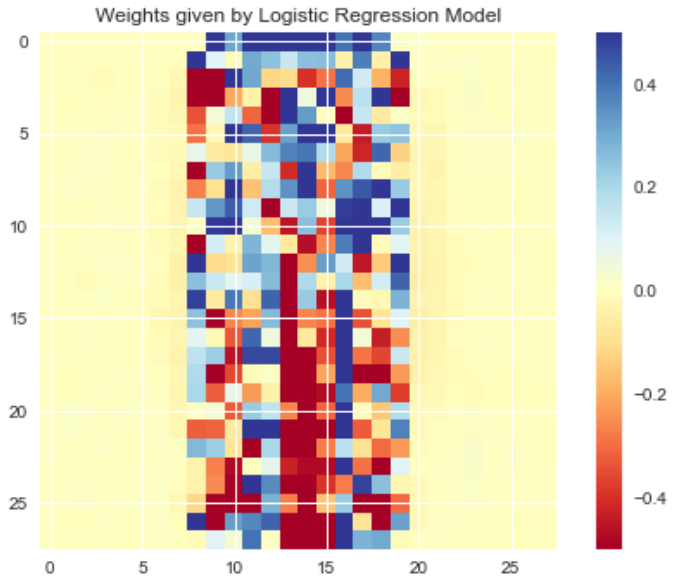
b) Regularization – Since the model has so many features, it is necessary to see if the model is giving some features more importance than the others. Hence I checked the error rate at various values of regularization parameters (*C_grid = np.logspace(-2, 2, 20)).* While checking different values, I noticed I got the lowest error rate for reg_param = 0.483. This means that before regularization, the model was overfitting on training data, and applying a higher penalty gave the best model with lowest error rate. Below is a plot with the results of changing penalty on a model, and we can see that lowest error rate is achieved at 0.483 param.



c) Logistic regression solvers – I tried different logistic regression solvers for this problem. Below is the table of the different solvers I used.

```
newton-cg:0.03524
lbfgs:0.035333333
liblinear:0.03550
sag:0.03524999999
saga:0.0355833333
```

d) Analysis of model coefficients – Below is the colour map of the coefficient value assigned to each pixel. From the below image we can see that the model is able to understand the two legs of a trouser. Furthermore, it is also able to distinguish to some extend the sleeves of a dress. As there is a red concentration around the pixels near the sleeves.



Summary –

		Training		Testing	
	Model Details	**Error Rate**	**AUROC Score**	**Error Rate**	**AUROC Score**
1.	No pre-processing of input data	0.063	0.993	0.067	0.975
2.	Gaussian Filter to remove noise	0.061	0.978	0.072	0.970
3.	Median Filter to remove noise	0.038	0.992	**0.0474**	**0.987**
4.	Median Filter with Spatial Features	0.035	0.993	**0.0384**	**0.987**
5.	Median Filter with Spatial features and pixel counts	0.036	0.992	**0.0400**	**0.988**