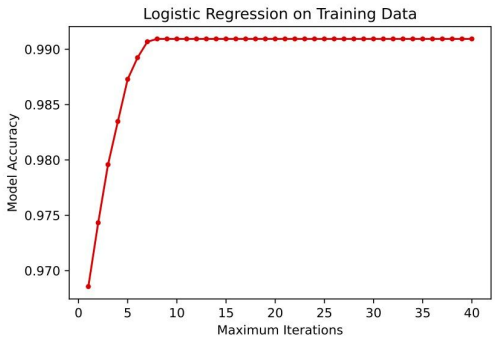
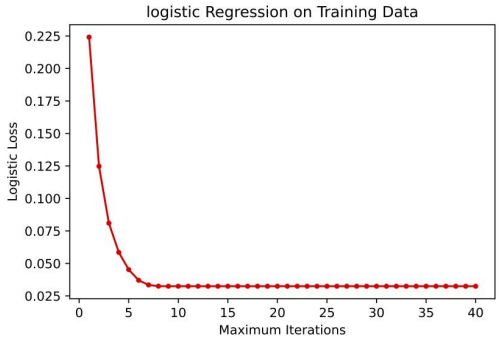
Part One

1.



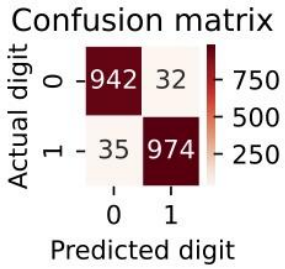


The max_iter is the number of iterations taken for the solver to converge. We can see that the model accuracy increases as the maximum iteration increases. More specifically, when max_iter increases from 1 to 7, it increases greatly, and after 7 it only increases by really small increments. Likewise, the logistic loss decreases greatly when max_iter increases from 1 to 7 and after 7 it only decreases by really small increments.

2.

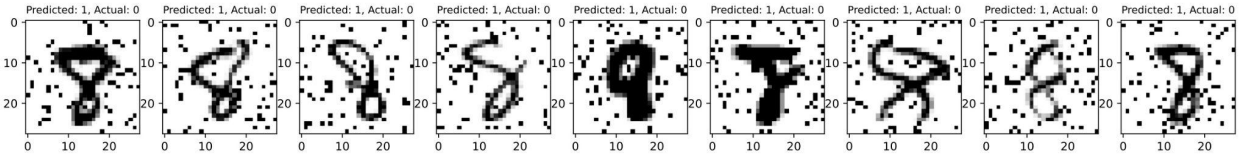
The graph above shows the feature weight of pixel000 generated from the logistic regression model when the given max_iter was used as its parameter. Based on the graph, when max_iter is 1 to 8, the feature weight decreases greatly and after 8 it only decreases by really small increments. Based on the y-axis, the feature weight is negative. This tells us that the pixel000 corresponds to number 8.

3. C = 0.03162277660168379
Logistic loss = 0.08997173640406692
Accuracy score = 0.9662128088754413

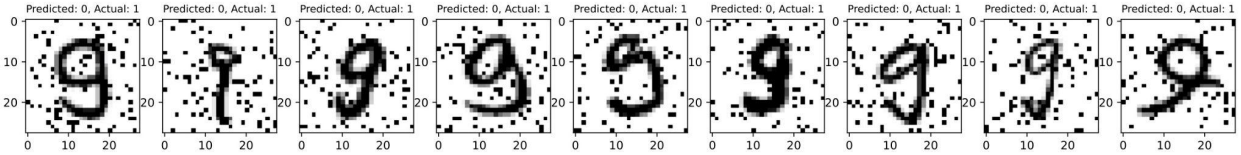


Regularization penalty c=0.03162277660168379 gives the least logistic loss and greatest accuracy score. By calculation, logistic loss is (35+32)/(35+32+942+974) and accuracy score is (942+974)/(35+32+942+974)

4.

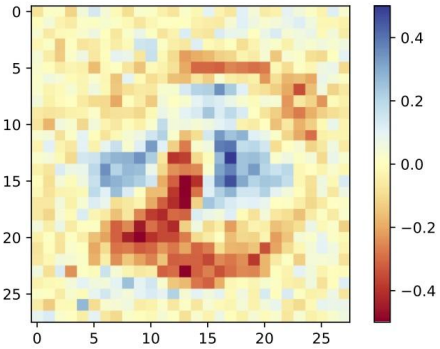


The figure above shows examples of the numbers 8 being falsely identified as 9 by the classifier. A common factor we can see that may have caused these mistakes is the bottom half of the number 8. The bottom circle in these examples appear to be smaller than the ones on top. 9 tends to have more black pixels on their top half than bottom half. Since it is a linear weighted model, the bottom half of the numbers above were underweighed, causing the model to predict it as 9.



The figure above shows examples of the numbers 9 being falsely identified as 8 by the classifier. A common factor we can see is that the bottom part of the numbers 9 above is not vertical "|" but rather slanted/curved like "_/" or even "_/". This may have caused the classifier to predict the numbers as 8 due to its shape. Since it is a linear weighted model, the bottom half of the numbers above were outweighed, causing the model to predict it as 9.

5.

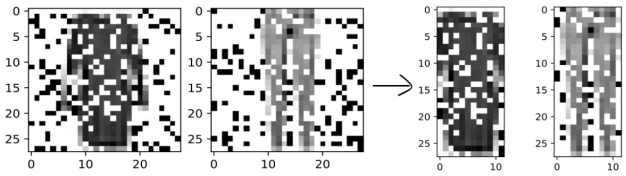


- yellow colored pixels correspond to weight=0. We can see the majority of the outer surrounding to be yellow and this makes sense as they're irrelevant in determining whether the number is 8 or 9 as the number is in the middle of the image not the outer part, thus having 0 weight.
- orange-red = negative weight which corresponds to 8. We can see that the bottom middle part is mostly red and we can say that this showcases the bottom circle of the number 8 "_/".
- light-dark blue = positive weight which corresponds to 9.
- On the top middle area it is a mixture of blue and red so it's hard to tell from that area whether it is a number 8 or 9. Reason behind this is that the numbers 8 and 9 are both circles on their top half "o", thus it is difficult to differentiate.

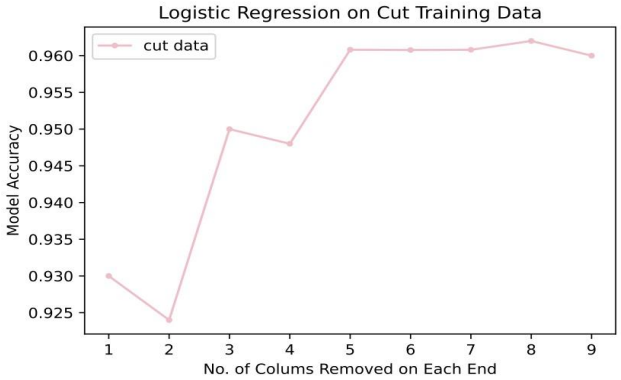
Part Two

Feature Transformations

1. Features Deleted: Side Columns Removed (nickname "cut")



First transformation done was removing columns on each end of the column. Since the actual pixels that compose the dress/trouser image are only in the middle, we can disregard the pixels on the sides as they are "unnecessary" features that affect the prediction model.



The model accuracy graph on the left shows the model accuracy as the number of columns removed on each end increases. We can see that removing 8 columns on each end provides the greatest model accuracy so this "cut 8" feature transformation was used such that the dimensions of each image data goes from (28×28) to (28×12).

At this point, the error rate on the testing data is 0.051, when the logistic regression model was created only with the x_train data changed by the "cut 8" feature transformation.

2. New Features Added: No. of black and white pixels (nickname "bw")

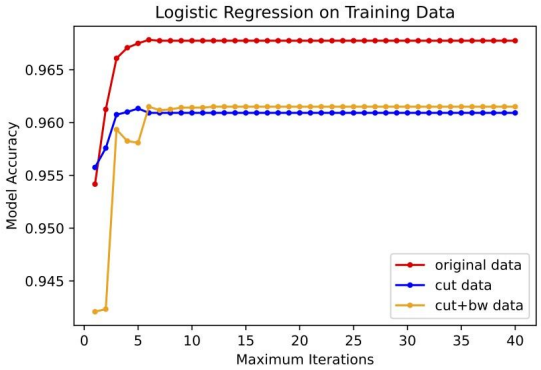
We can see from the (28×12) image above that dresses tend to have more black pixels than trousers, mainly due to the empty middle area of the trousers between the right and left leg section. So, the second transformation done was adding new features that count the number of black and white pixels. Since the pixels are in gray-scale values, a number X was used to determine if the given pixel values(p) are categorized as black or white. If $p \geq X$, count as black pixel. If $p < X$, count as white pixel.

The error rate graph on the right shows the error rate produced corresponding to a range of values $0.4 \leq X \leq 0.6$ used with 0.25 intervals. It was concluded that $X = 0.425$ gives us the smallest error rate, so this X value was used to calculate the number of black and white pixels on each of the image data and 2 new columns were added to the x_train data.

At this point, the error rate on testing data went from 0.051 to 0.049, when the logistic regression model was created with the x_train data that was changed by the "cut 8" and "bw" feature transformations.

Regularizations / Parameters

1. Maximum iteration



When creating the logistic model, max_iter was set as max_iter = 1,2,..,40 and the corresponding max_iter which produced the best model accuracy was used in the end. On the model accuracy graph to the left, focusing on only the cut+bw data, we can see that max_iter = 6 gives the greatest model accuracy.

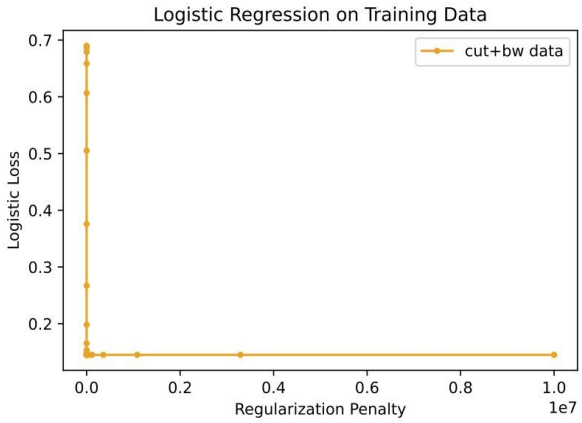
At this point, the error rate on testing data went from 0.049 to 0.045, when the logistic regression model was created with max_iter = 6 instead of the default value 100 on the x_train data (changed by "cut 8" and "bw").

Analysis on the graph above:

Why is the model accuracy on the original data greater than on the cut/cut+bw data for logistic regression on training data?

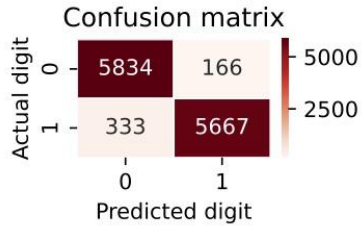
Reason behind this could be that when creating the prediction model, there were patterns on the end columns that became factors in differentiating between dress/trouser (even though these columns do not make up the image of the dresses/trousers). And since the model accuracy was conducted with the x_train data, it makes sense that the loss of these side columns may have caused the accuracy to predict to decrease. However, when predicting other data such as the x_test, we are not sure that their side columns have the same patterns as the x_train data. For example, perhaps the dresses on x_train data generally have more black pixels on the left columns than trousers, while in x_test data the dresses generally may have more black pixels on the right column than trousers. The prediction model is created using the x_train model and if the side columns were used, then when testing on the x_test data there would likely be more false predictions as they would predict a dress is a trouser because of the feature weight that there's less black pixels on the left column. In conclusion, while the model accuracy may seem worse on cut data, it is theoretically better when predicting on other non x_train datas as we are removing the possibility of the side columns being a huge factor on predicting as side columns patterns may differ for different data sets.

2. Regularization Penalty



Regularization penalty is used to prevent overfitting and underfitting when creating the logistic model, such that the accuracy of prediction when predicting other/new data sets improves, by blocking out noise data and prioritizing important data. Here, the regularization penalty was set to a range of values c = np.logspace(-7,7,30) and the corresponding c which produced the least logistic loss was used. It was found that c = 0.573615251044868 gives the least logistic loss of 0.145.

At this point, the error rate on testing data went from 0.045 to 0.044, when the logistic regression model was created with max_iter = 6, c = 0.573615251044868 instead of the default value max_iter = 100, c = 1.0 on the x_train data (changed by "cut 8" and "bw").



To the left is the confusion matrix table of the model on the training data. We can calculate the accuracy of the model to be (5667+5834)/12000 = 95.84%.

3. **L1 lasso/L2 ridge Regularization**

Regularization	Error Rate
L1 regularization	0.049
L2 regularization (default)	0.045

Both regularization's purpose is to prevent overfitting and underfitting. The main difference between the two is that L1 shrinks the coefficients of less important features to zero, which result in the removal of some features. While this usually works for feature selection with a large number of features, L1 results in an increase in the error rate for our data in this case. A reason for this may be due to the "cut" feature transformation that was done to our data initially. Since we've already cut the sides of the columns that were deemed not important, using L1 may result in important pixel data being removed, thus resulting in the accuracy of the predictor model to decrease due to loss of important data. Thus, for our logistic model, the default L2 regularization was used, resulting with no change in the error rate 0.044.

Conclusion/Summary

Step	Feature Transformation on x_train data	Parameters for Building Model	Error Rate on Testing Data
0	no transformation (original)	solver='liblinear'	0.0675
1	"cut 8"	solver='liblinear'	0.051
2	"cut 8" + "bw"	solver='liblinear'	0.049
3	"cut 8" + "bw"	solver='liblinear', max_iter=6	0.045
4	"cut 8" + "bw"	solver='liblinear' max_iter=6, c=0.57...	0.044

- In step 1, side columns were cut to remove pixels that do not compose the dress/trouser pixels. Purpose of this is to remove the possibility of patterns found on these side columns that could affect the prediction model as different data sets may have different patterns on their side columns.
- In step 2, 2 features were added, the number of black and white pixels. This increases the accuracy as we know dresses generally should have more black pixels than trousers.
- In step 3 and 4, to improve accuracy by avoiding overfitting and underfitting, ranges of maximum iteration and regularization penalty values were tested to find the numbers that give the best accuracy and least logistic loss.
- Additionally, L2 regularization was used instead of L1 to prevent important pixel data from being removed as the "cut 8" transformation done in step 1 has already removed the not important side columns data.

We can see that as we go from step 0 to 4, as we add more feature transformations and use non-default parameters, the error rate on testing data decreases.