Project01 ML

Tim Panilaitis

July 2025

Part 1: Logistic Regression for Digit Classification

0.1 Maximum Iteration Effects on Accuracy and Loss in Logistic Regression Model

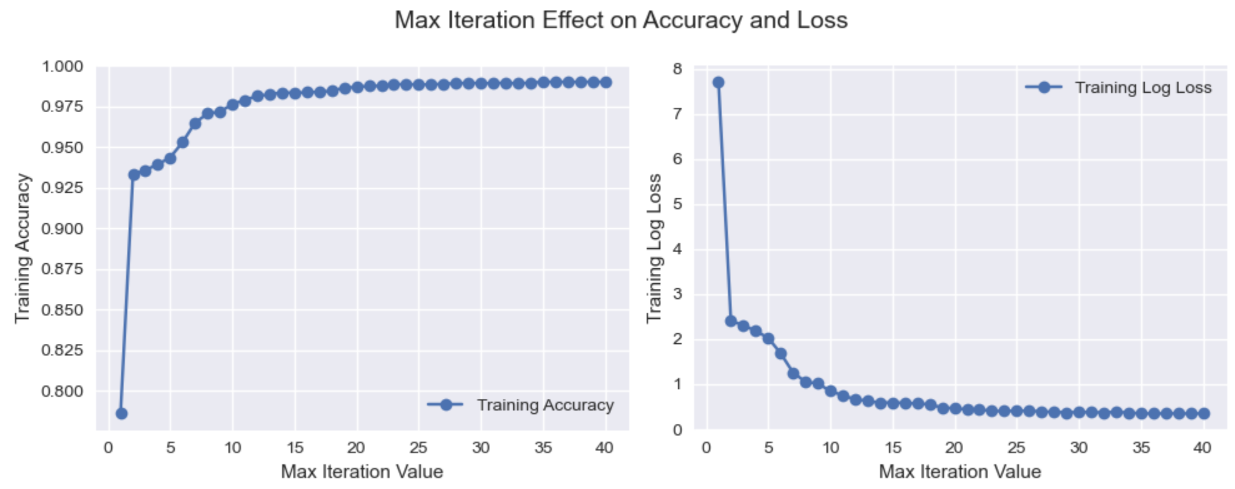


Figure 1: Figure examining the effects of a max iteration value of logarithmic loss and accuracy metrics on training data

Figure 1 showcases the most obvious effect of a higher maximum iteration value. As more iterations are allowed, the model is able to better fit the data. This lowers logarithmic error and raises the accuracy of the model on the training data. However, despite the error lowering on the training data it is likely that the testing error (which shows the general practical use of a model) might not be lowering as well. It is possible that without any constrictions on complexity the model is overfitting the data and thus becoming less useful. A second test on this same model confirms this.

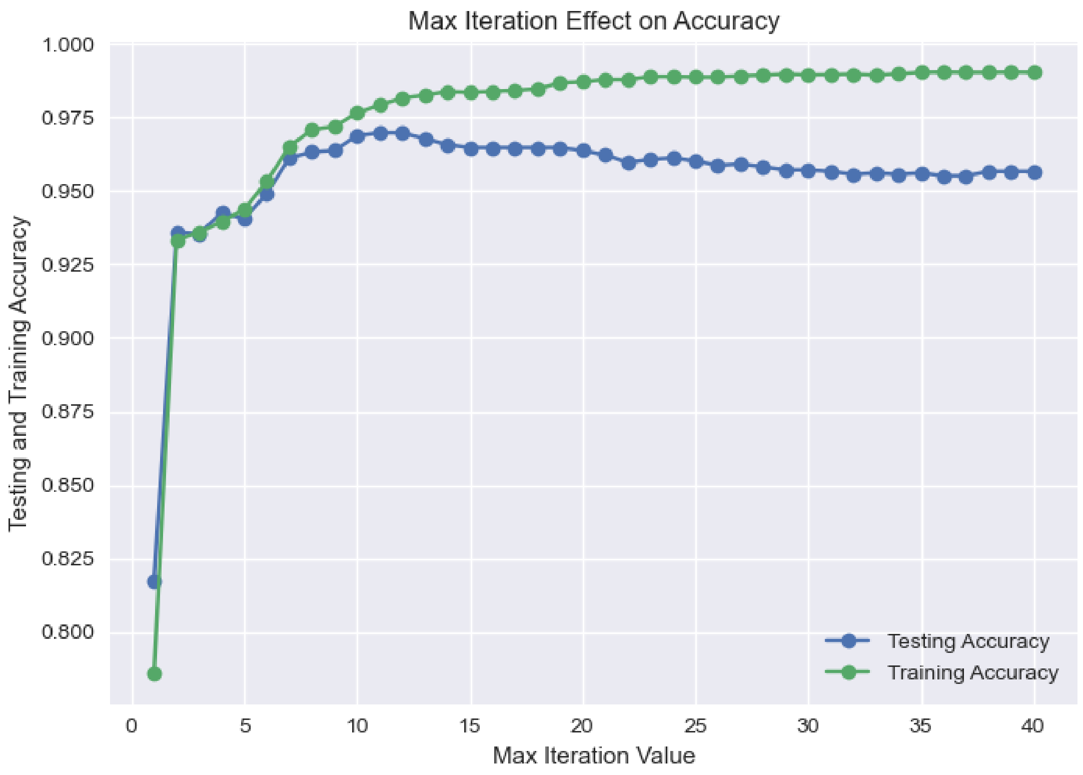


Figure 2: Figure compares the testing and training accuracy of a model as the amount of iterations is raised.

As shown in Figure 2, the training error is increasingly diminished as the model begins to overfit to the training data. The testing error shows a decline over about 12 iterations, meaning that iterations past this point reduce generalization of the model.

0.2 Maximum Iteration Effects on First Weight in Logistic Regression Model

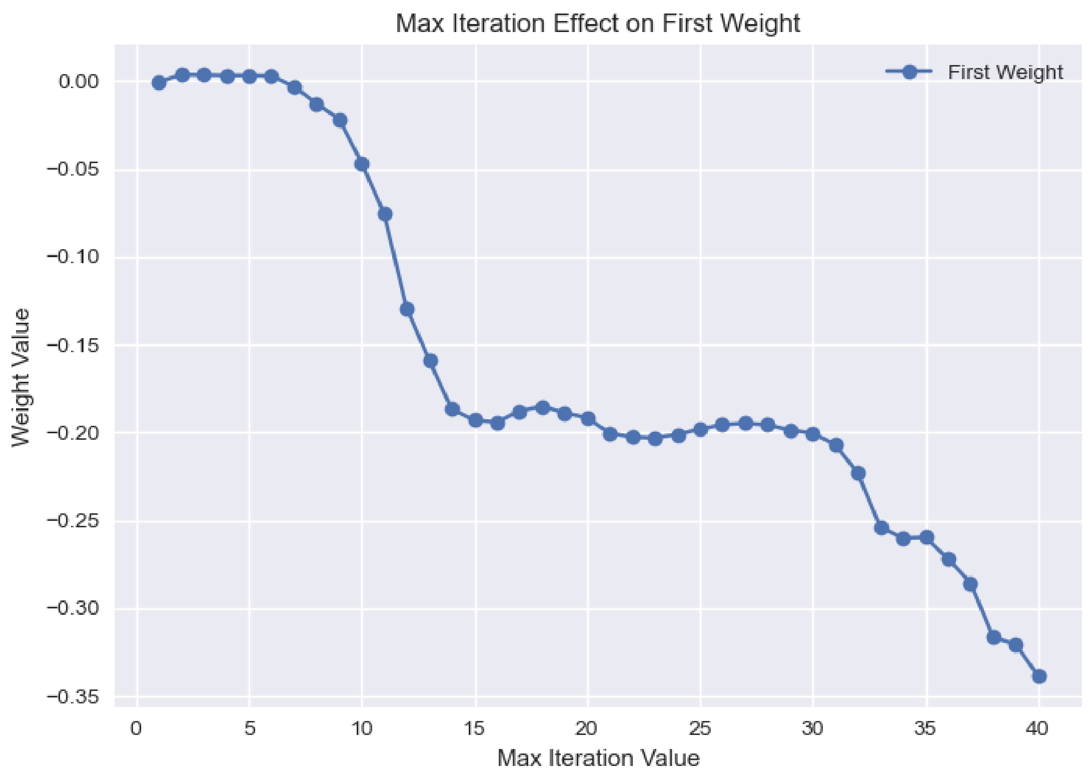


Figure 3: Figure shows the weight coefficient attached to the first pixel.

Figure 3 shows that as the maximum allowed iterations increases the weight of the first feature becomes more negative. Since a negative weight means that a black pixel in the first spot contributes to a prediction closer to 0 and consequently a classification of the handwritten number being an 8. As previously shown, overfitting occurred past roughly 12 iterations, so we can conclude that the first pixel being a black pixel is more likely to produce a prediction of an 8. However, the impact on that conclusion should be quite low (around -0.05). It is possible that as iterations grew the weight began to reflect the noise present in the training data rather than meaningful learning. This may reflect an imbalance in the training data, where true 8s were more likely to have a pixel in the first position, but that generally this feature is not a strong indicator of either classification. It is worth noting that under further inspection of the image files themselves, the written numbers were in the center of the canvas. This further supports the fact that the noise of the image is what drove this weight, and its real impact is insignifigant.

0.3 Finding best C value for Regularized Logistic Model

The best C value is 0.1
Its logarithmic loss is 1.1814611549634972
Its accuracy is 0.9672213817448311

		Predicted	
		0	**1**
True	**0**	943	31
	1	34	975

Table 1: Confusion Matrix (True vs. Predicted) of the Best Model

0.4 Analyzing Mistakes of Best Model

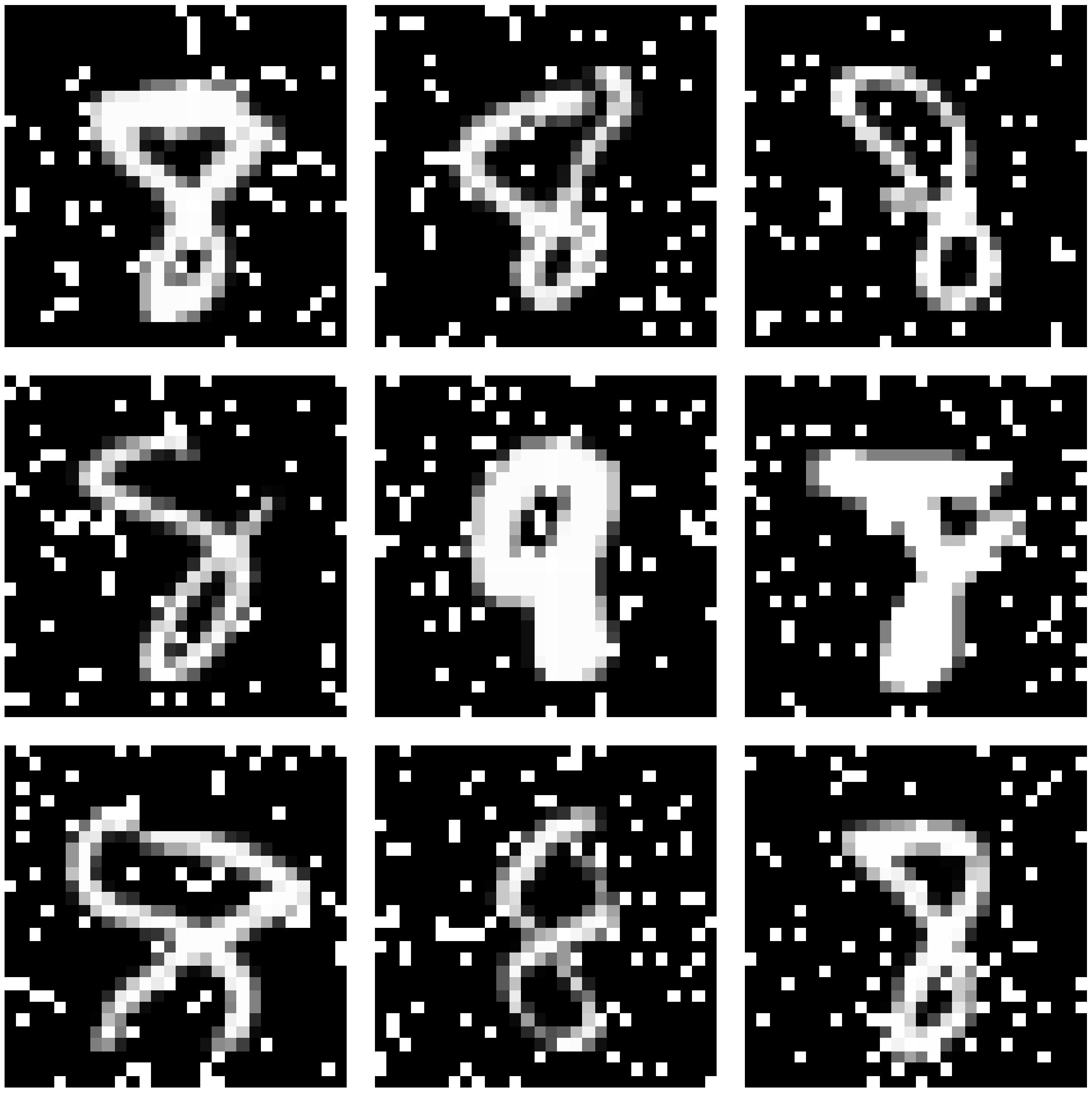


Figure 4: 9 Misclassified Testing Features (False Positives)

Overall, when looking at these mistaken classifications I am quite impressed with the models performance. At least 2/9 I would classify myself as a 9, and around 5/9 I would struggle to read in a normal setting. The things that the model seems to struggle with is if an 8 has a broken line, or has a disproportionately large top loop in comparison to the bottom loop. It also seems to struggle in cases where there is a general bend between the two loops.

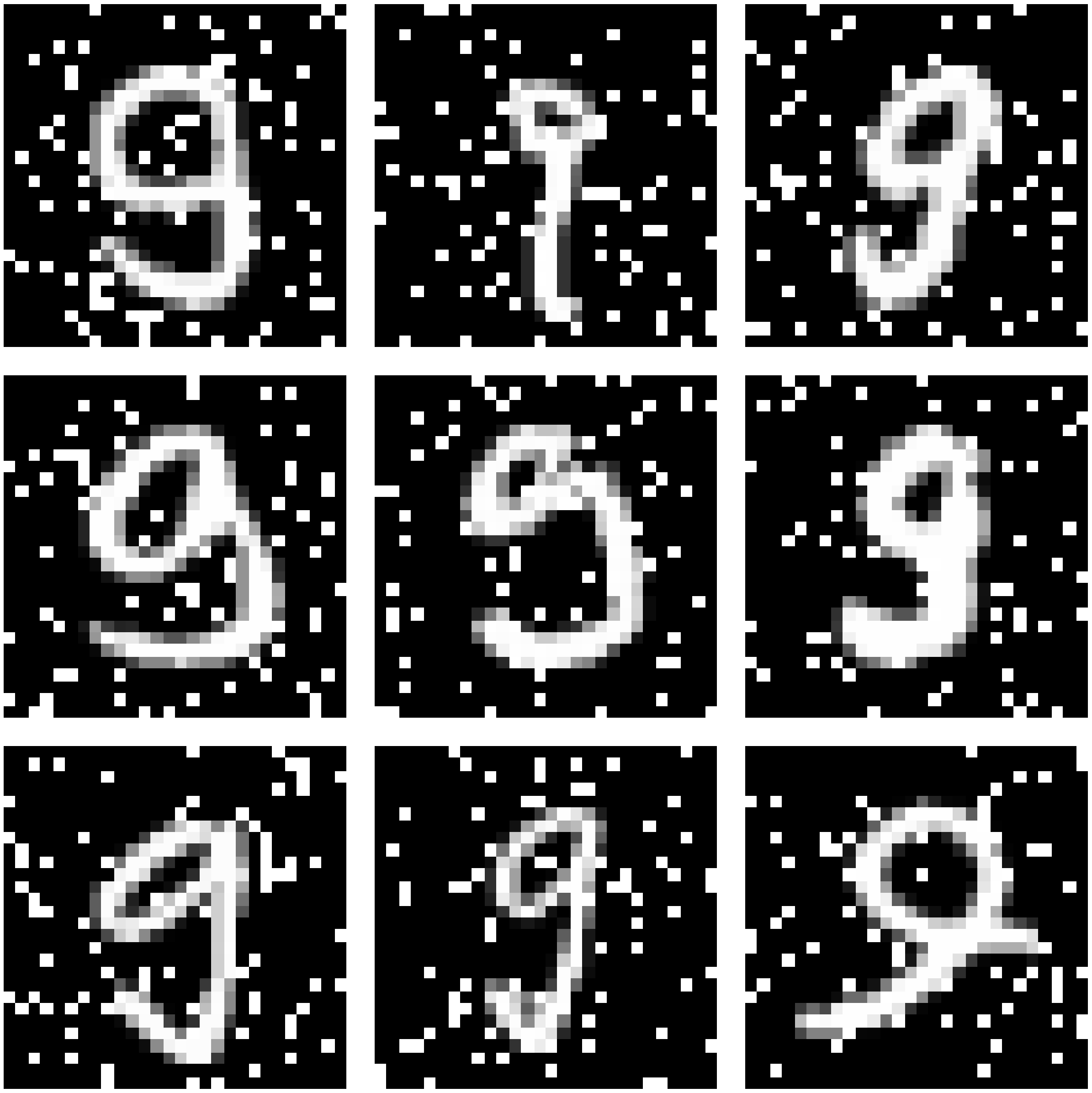


Figure 5: 9 Misclassified Testing Features (False Negatives)

Once again I am impressed by the models performance. 2/9 of the 9s I see as unintelligible and are only able to be understood within the context of this problem, and another 6/9 I would see as difficult to read. The model seems to struggle with 9s that have the bottom loop nearly closed, despite them not being completely closed. It also seems to be confused when the bottom loop forms a snowman-esque shape, where despite not being closed it is still larger than the top.

0.5 Weight Analysis Through Visualization

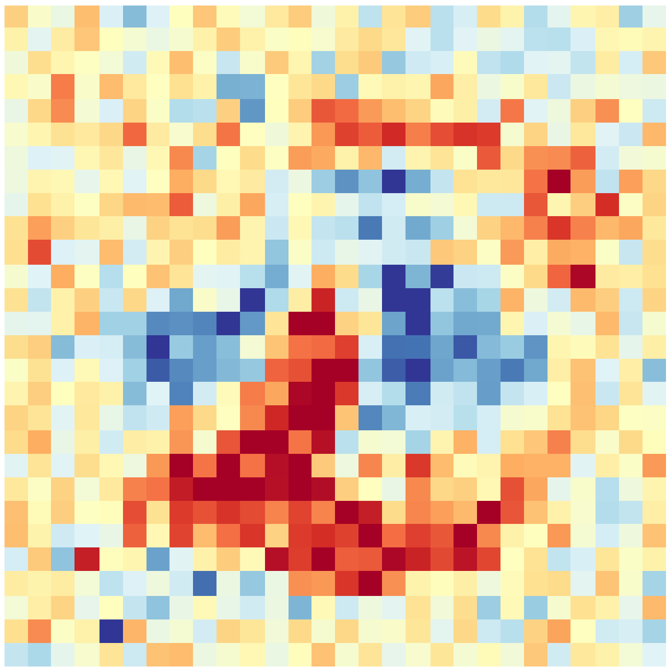


Figure 6: Weights Visualized Where Warmer Colors Represent a Classification of 9 and Cooler Colors a Classification of 8

This image from Figure 6 is interesting as it gives a visual depiction of the areas where the placement of pixels would effect things. We can deduce that warmer colors show pixels that contribute positively to predicting a 9 and cooler colors are pixels that contribute negatively to the prediction (predict an 8). This figure showcases a large red swoop along the bottom, which is precisely where the bottom swoop of a 9 would be. The cooler colors are concentrated in the center, which is where a large amount of 8s would be clumped in.

Part 2: Trousers v. Dresses

0.1 Creating a Baseline

My first order of business was to create a baseline for the test. I used k-fold validation to evaluate the error of each Logistic Regression Model. I started with a Logistic Regression Model with no parameterization and evaluated its average MSE values over 10 folds. The starting error was around 6% on testing data and around 3% on training data.

0.2 Data Analysis

As recommended in the project specification, I decided to look into new features or processing strategies that could yield a more accurate model. To invent more features and manipulate the dataset I decided to do a comparison of 9 trousers and 9 dresses.

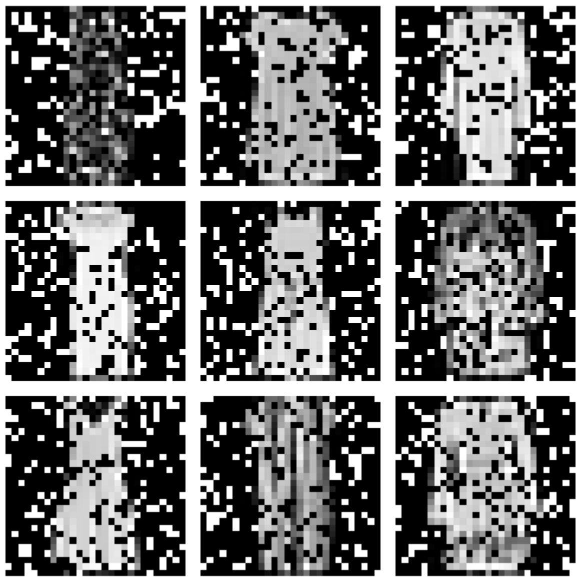


Figure 7: 9 pictures of dresses visualized in black and white

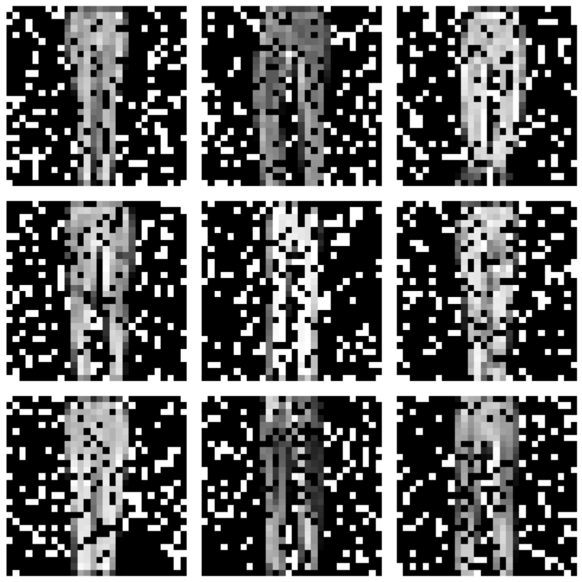


Figure 8: 9 pictures of trousers visualized in black and white

In the comparison of Figures 7 and 8 several key differences were revealed:

- The number of white/gray pixels is generally larger in dresses
- The dresses are generally wider than the trousers
- The trousers seem to have a more rectangular shape, whereas the dresses can be more oval shaped or wider at the top
- The trousers usually have a line down the middle (but this can appear on dresses as well)

To better understand the intensity and density of all color ranges of pixels in the dataset, I decided to visualize using a histogram.

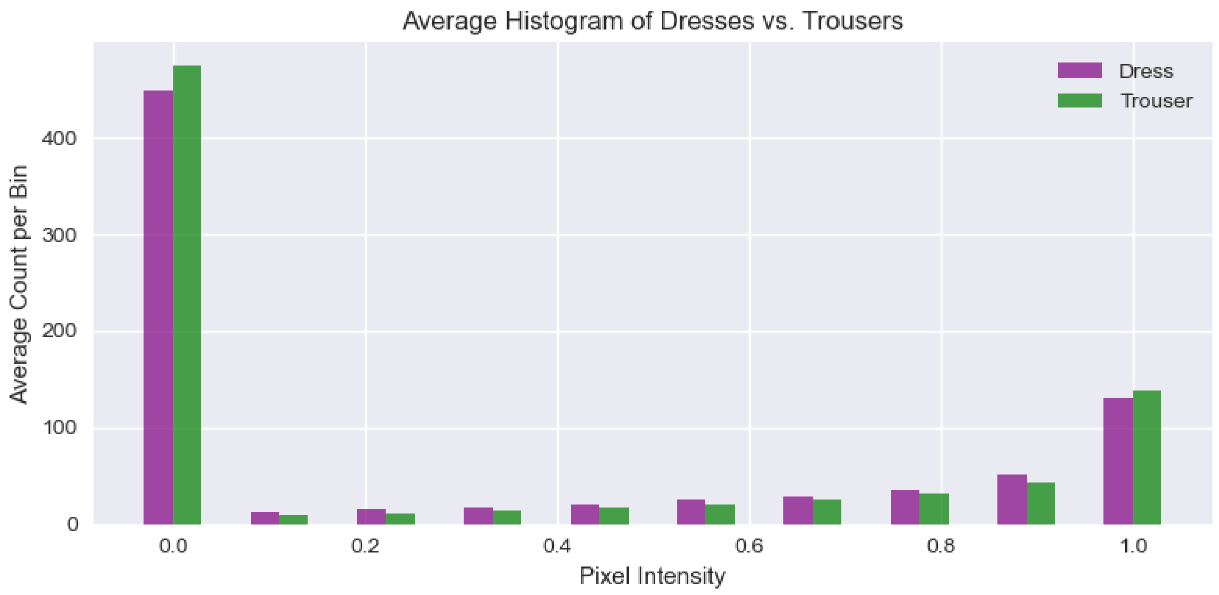
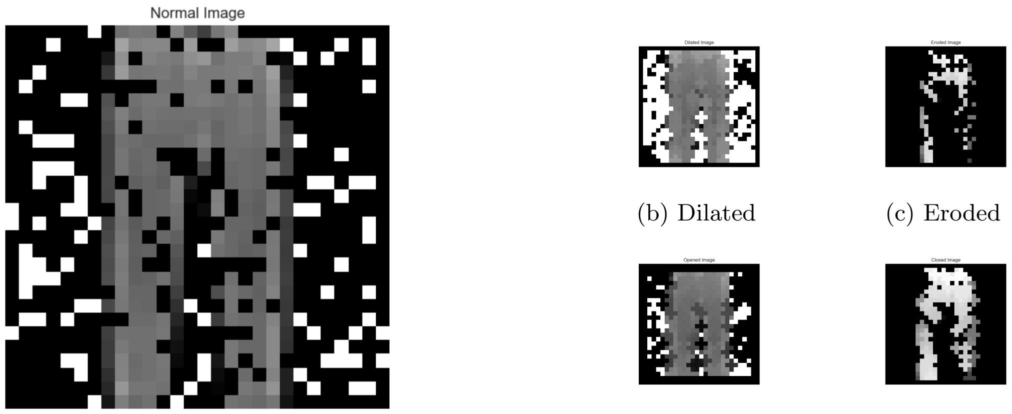


Figure 9: Measure of Pixel Color Density over all Trousers and Dresses

Figure 9 shows that trousers generally have a larger amount of black pixels, which makes logical sense due to the less amount of space they take up within the image. Dresses have a larger amount of each pixel concentration besides white pixels, as more colored pixels occur within dresses.

In general, the clear difference is that of the shape of the dresses and trousers. However, when looking at this data it is difficult to pick out the shape itself as the amount of noise in the image is overwhelming. After doing some research, I researched a strategy that outlines the shape of an image well. This strategy is called opening/closing an image, and when used it can create a stronger and easier to recognize outline[1]. Thus some preprocessing would be useful to have a cleaner image.

0.3 Feature Transformation Through Preproccessing



(b) Dilated (c) Eroded

(d) Opened (e) Closed

(a) Original Image

Figure 10: Left: original image. Right: morphological transformations (dilation, erosion, closing, and opening).

These methods of transformation do several things. Erosion removes a lot of the noise, but creates holes inside the object itself. Dilation can fill out the object but will magnify noise at the beginning. When used individually they do not help for differentiation much, but in combination they are quite powerful. If dilation is followed by erosion it is called closing. If erosion is followed by dilation it is called opening [1].

Using this method I transformed the starting data into a closed set and an opened set in addition to the base dataset. When comparing the use of all three datasets in training a simple model with 0 parameter manipulation I found the following results:

	Test Error (%)	Training Error (%)
Normal Dataset	6.4666	3.0426
Closed Dataset	10.0250	7.6787
Opened Dataset	6.1167	4.3685

Table 2: An evaluation of datasets using k-fold cross validation

This approach shows a lot of promise, as with little difference I have improved the error percent in the opening dataset. However, when looking at Figure 10 the closed version looks extremely promising for the human eye, but failed to converge under the same amount of iterations as before. While a human can pick out the difference it might be more difficult for a machine due to the large amount of noise introduced when dilation occurs before. In order to give a closed dataset a fair chance, I decided to research methods of noise reduction that could occur before the closing/opening of the dataset.

I decided upon several preproccesing techniques; Guassian Blur, Median Blur, and Bilateral Filter. All of which are useful for clearing up noise and making the object stand out better. In particular, the Guassian Filter works to smooth out the image and thereby reduces noise and makes future edge detection easier. The Median Blur Filter focuses mainly on reducing ”salt and pepper” noise by replacing each pixel with the median value of its neighbor. The Bilateral Filter helps smooth general image but still keep the edges sharp [2, 3]. Below is a comparison of its effects on a simple image.

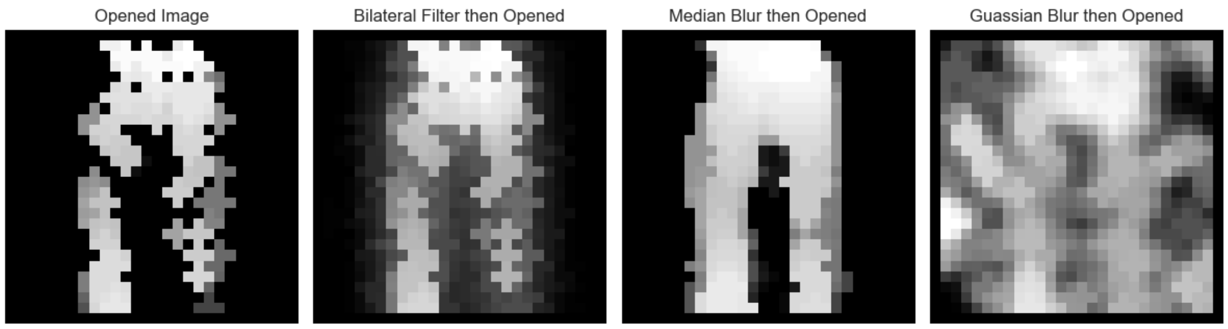


Figure 11: A comparison of the different noise reduction filters before opening

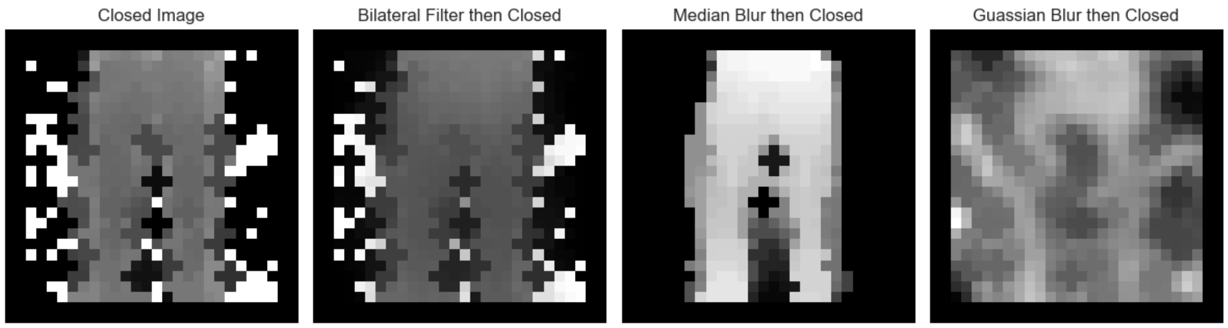


Figure 12: A comparison of the different noise reduction filters before closing

Using these new datasets I ran k-fold cross validation with a non-augmented simple logistic regression model then analyed the testing and training mean-squared-error of each.

	Blur Type	Testing Error (%)	Training Error (%)
Closed	No Blur	10.02	7.68
	Gaussian Blur	5.69	4.95
	Median Blur	7.94	6.65
	Bilateral Filter	10.54	8.19
Opened	No Blur	6.12	4.37
	Gaussian Blur	5.02	4.34
	Median Blur	6.54	5.29
	Bilateral Filter	6.19	4.45

Table 3: Comparison of filtering and preprocessing methods on test and training error.

This experiment was confusing for me to say the least. When analyzing figures 11 and 12 I found that the use of median blur then opened was great for getting a clear outlined shape. The bilateral opened filter was also great in the amount of detail still present, when looking from a human perspective. Therefore, it is surprising to see that both performed with very little error difference to the original unchanged dataset. In fact, the median blur performed comparatively worse.

In all different preprocessing tests the opened dataset performed better by a considerable margin, so all future tests will disregard the closed datasets. While Gaussian Blur performed best in testing metrics, I will be keeping the Median Blur and Bilateral Filters for future tests. Both datasets show promise, and might perform better with a more curated and parameterized Logistic Regression Model. The best performing dataset is the Gaussian Blurred then opened dataset. This seems like it is because it preserves more information throughout the image, allowing a more educated choice.

As of now, the best model is a simple non-parameterized Logistic Regression Model trained on a dataset that was smoothed with a Gaussian Blur then Opened using erosion then dilation. Further experiments into layered smoothing/blurring was inconsequential.

0.4 Creating New Features

Now that my image processing has improved accuracy signifigantly, I would like to expand the total amount of features my model uses. Currently the model relies completely on the transformed image in its totality, but I would like to add global features that sum different color information. In addition, I would like to add features that emphasize shape and size.

The main examples are:

- The mean pixel value
- The histogram representing all color intensities
- The standard deviation of each pixel
- The entropy: a measure of texture
- The edge density: Captures the length of edges in the image. This would be nice to show the detailed nature of the outline (trousers are likely more simple).
- The bounding box: captures the overall shape of the item.

The combination of these features is quite diverse. It manages to capture a large amount of detail of color (mean pixel, standard deviation, entropy, histogram) and shape (edge density, bounding box). This is the culmination of all features added to the Opened Gaussian Blurred Dataset.

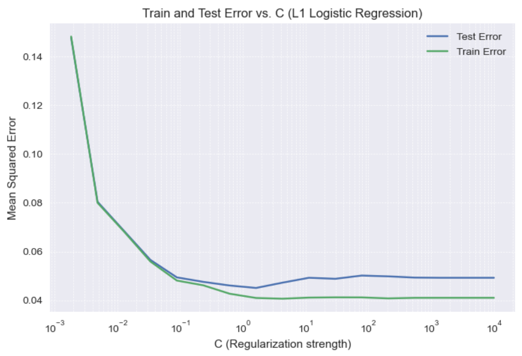
	Test Error (%)	Training Error (%)
Baseline Dataset	6.4666	3.0426
Opened Gaussian Dataset	5.02	4.34
Opened Gaussian Dataset with Extra Features	4.925	4.103

Table 4: An evaluation of best datasets using k-fold cross validation

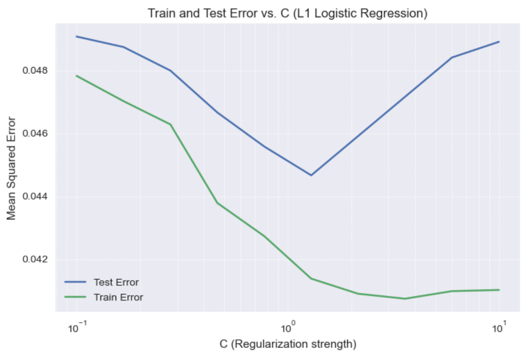
Table 4 shows a definite decrease in testing error over each iteration. While the culmination of all these new added features is a better model, there may be a better subset of these features present.

0.5 Feature Selection

Due to the immense amount of features I decided to train the model using a Lasso penalty. This will drive less impactful features to 0, and allow me to find features that can be removed from the overall set. My strategy for reducing error was to train the lasso model with different regulation strength values, and for each strength value I would collect all unimpactful features and train a new simple Logistic Regression Model with only those features. In each iteration the testing and training mean squared error would be collected.



(a) Lasso Penalty Test



(b) Lasso Penalty Train

Figure 13: Feature Selection Through Lasso Penalty

As shown in Figure 13a, the lowest test error is achieved when the C value is within the bounds of 10^{-1} to 10^{1}. Figure 13b shows the same test run once more, but within the best bounds specified in Figure 13a. I saved the best feature subset to a new data set.

	Test Error (%)	Training Error (%)
Baseline Dataset	6.4666	3.0426
Opened Gaussian Dataset	5.02	4.34
Opened Gaussian Dataset with All Features	4.925	4.103
Opened Gaussian Dataset with Reduced Features	4.692	4.069

Table 5: An evaluation of all best datasets using k-fold cross validation

I am very pleased with the current state of my classifier! I have shaved around 2% testing error without even touching the model parameters.

0.6 Parameter Selection

To find the best parameter selection I used the grid search function from sklearn [4]. This would further enhance the success of the datasets. The resulting parameters are:

- solver = saga
- penalty = l2
- C = 1
- class weight = None

This was then used as the final parameter to reach the results shown in 6.

	Test Error (%)	Training Error (%)
Baseline Dataset	6.4666	3.0426
Opened Gaussian Dataset	5.02	4.34
Opened Gaussian Dataset with All Features	4.925	4.103
Opened Gaussian Dataset with Reduced Features	4.692	4.069
Opened Gaussian Dataset with Reduced Features and Parameters	4.924	4.025

Table 6: An evaluation of all best datasets using k-fold cross validation

1 Summary

I started by applying a Gaussian blur filter to my image dataset to reduce noise and smooth the images. Then, I performed morphological opening using grayscale erosion followed by dilation to further clean and highlight important structures in the images. After preprocessing, I selected the best features from the dataset by loading a precomputed feature selection file to focus the model on the most informative attributes. Finally, using the best combination of preprocessing, feature selection, and model parameters, I trained the final logistic regression model, which achieved improved performance in k-fold validation.

This process taught me the importance of combining careful data preprocessing with hyperparameter tuning to build effective machine learning models. It was also quite fun!

References

[1] A. Dattani, "Seeing Like a Machine: A Beginner's Guide to Image Analysis in Machine Learning," *DataCamp*. [Online]. Available: https://www.datacamp.com/tutorial/seeing-like-a-machine-a-beginners-guide-to-image-analysis-in-machine-learning. [Accessed: Jul. 17, 2025].

[2] M. Patel, "The Complete Guide to Image Preprocessing Techniques in Python," *Medium*. [Online]. Available: https://medium.com/@maahip1304/the-complete-guide-to-image-preprocessing-techniques-in-python-dca30804550c. [Accessed: Jul. 17, 2025].

[3] OpenCV Documentation, "Image Filtering," *OpenCV*. [Online]. Available: https://docs.opencv.org/4.x/d4/d13/tutorial$_{p}y_{f}iltering.html$. [Accessed: Jul. 17, 2025].

[4] F. Pedregosa *et al.*, "GridSearchCV," *Scikit-learn*, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model$_{s}election.GridSearchCV.html$. [Accessed: Jul. 18, 2025].