COMP 135: Project 1

Darcy Corson

April 28, 2025

1 Logistic Regression for Digit Classification

In this study, logistic regression models were trained on a dataset using the LogisticRegression implementation from the scikit-learn library, employing the liblinear solver. To investigate the effects of constraining the number of iterations available for the solver's convergence, the max_iter parameter was incremented sequentially from 1 to 40. For each value of max_iter, the model was fitted to the training data, and its performance was assessed. Two performance metrics were employed to evaluate the performance of the logistic regression models: accuracy and logistic loss. Both of these metrics were calculated using the same set of data on which the model was trained.

The accuracy metric measured how well the model predicted the correct outcome by determining the ratio of correct predictions to total predictions. The score() function was used to compute this metric. A model with an accuracy of 1.0 makes only correct predictions; a model with an accuracy of 0 makes only incorrect predictions.

The logistic loss (log loss) metric measured the model's prediction accuracy from a probabilistic perspective. It considered how confident the model is in its predictions and penalized incorrect confidence levels. Thus, lower log loss values indicate better model performance, and higher values suggest poor performance.

The goal of these analyses was to identify specific trends, patterns, or inflection points in the model's performance. Specifically, the intention was to see how varying the number of allowed iterations for the solver impacts the model's accuracy and log loss. The data gathered from these analyses offers insights into how many iterations are optimal for the model to perform well without unnecessary computational overhead.

1.1 Question 1

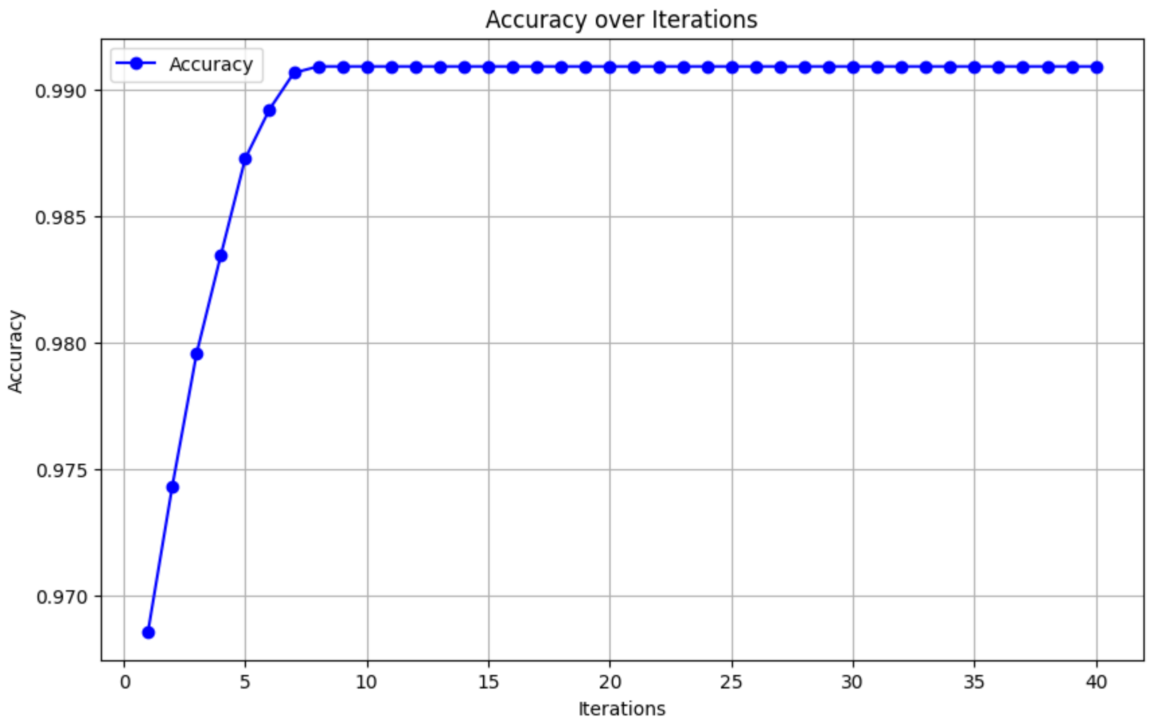


Figure 1: The logistic regression model's accuracy increases from just below 0.970 to approximately 0.990 within the first 10 iterations.

"Accuracy over Iterations" depicts the accuracy of the logistic regression model as a function of the number of iterations allowed for the solver to converge. Here, "iterations" refers to the number of times the algorithm goes through the process of adjusting the model's weights to minimize the error based on the training data. Thus, the graph illustrates the progression of the model's accuracy over a series of training iterations. The x-axis enumerates the number of iterations from 0 to 40, while the y-axis conveys the model's accuracy, which varies between approximately 0.970 and 0.990.

Initially, rapid improvement in accuracy is observed, moving from just below 0.970 to approximately 0.990 within the first 10 iterations. After 10 iterations, however, the accuracy plateaus and remains consistent at about 0.990, showing no significant change up to 40 iterations.

Analysis of the graph reveals that the accuracy starts at near 0.970 and, in initial iterations, there's a dramatic increase in accuracy. This initial surge indicates that the model rapidly integrated the dominant patterns in the training set during these early iterations. After the rapid initial improvement in accuracy, the accuracy growth becomes less significant (after about the 10th iteration). By the 15th iteration, the model's accuracy plateaus, hovering just below 0.990. Thus, after an initial phase of rapid learning, the accuracy remains consistent and stable for iterations 15 to 40. This suggests that the model has reached its optimal performance on the given dataset, at least with respect to the training phase.

The initial sharp rise in accuracy signifies that the model is adept at quickly identifying and adapting to the major trends in the training data. This rapid initial learning phase may indicate the model's capability to adjust its weights when starting from a non-optimal initialization. The graph suggests that for this particular dataset and model configuration, extending training beyond the 15th iteration might not yield substantial gains in accuracy. Therefore, it might be more computationally efficient to halt training after this point.

Though the graph shows a high training accuracy, it's important to ensure that this doesn't result in over-fitting, where the model performs well on the training data but struggles with new, unseen data. Evaluating the model on a separate validation set is important.

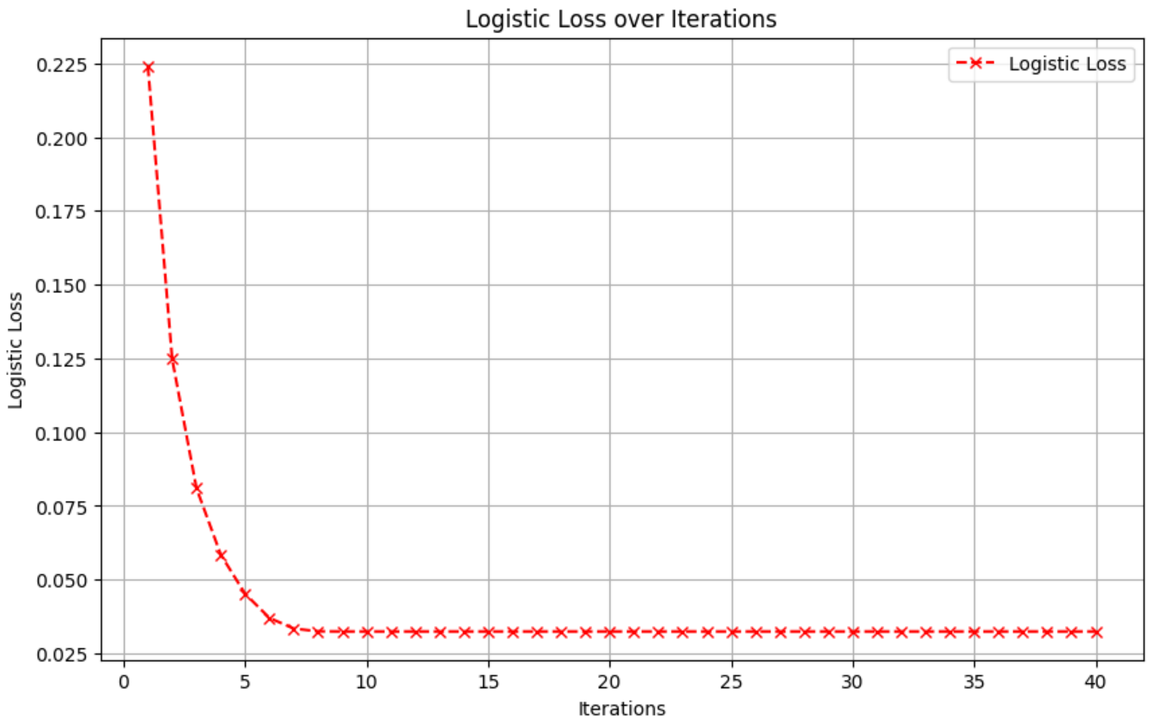


Figure 2: The logistic regression model's log loss decreases from about 0.225 to approximately 0.025 within the first 10 iterations.

”Logistic Loss over Iterations” illustrates the logistic loss of the logistic regression model in relation to the number of iterations undertaken for the solver to converge. In this context, "iterations" signifies the number of cycles the algorithm undertakes to adjust the model's weights for the purpose of minimizing the logistic loss based on the training data. Thus, the graph demonstrates the trajectory of the model's logistic loss over a sequence of training iterations. The x-axis represents the number of iterations, ranging from 0 to 40, and the y-axis details the model's logistic loss, which starts around 0.225 and decreases, stabilizing close to 0.025.

Analysis of the graph reveals a steep decline in log loss, starting near 0.225 and dramatically decreasing in early iterations. This pronounced initial decline suggests that the model quickly detemines the predominant patterns in the data, causing it to adjust its weights in a way that significantly reduces error. As iterations increase, especially beyond the 10th iteration, the rate of log loss reduction starts to decrease. This implies that, even though the model continues to refine its weights, the subsequent reduction in error become increasingly less significant. By the 15th iteration, the log loss plateaus, maintaining a near constant value. This is indicative of the model achieving and maintaining a consistently low log loss for subsequent iterations.

The rapid decrease in log loss during early iterations signifies that the model is highly responsive, especially when it is initialized with weights that may not be optimal. This responsiveness speaks to the model's ability to quickly adjust based on the training data provided. Further, the effective stabilization of the log loss after the 15th iteration suggests that continued training may not result in any notable decline in log loss.

While a lower logistic loss is desirable, it's important to assess the model's performance in real-world scenarios. Evaluating the model on a separate validation set is important.

1.2 Question 2

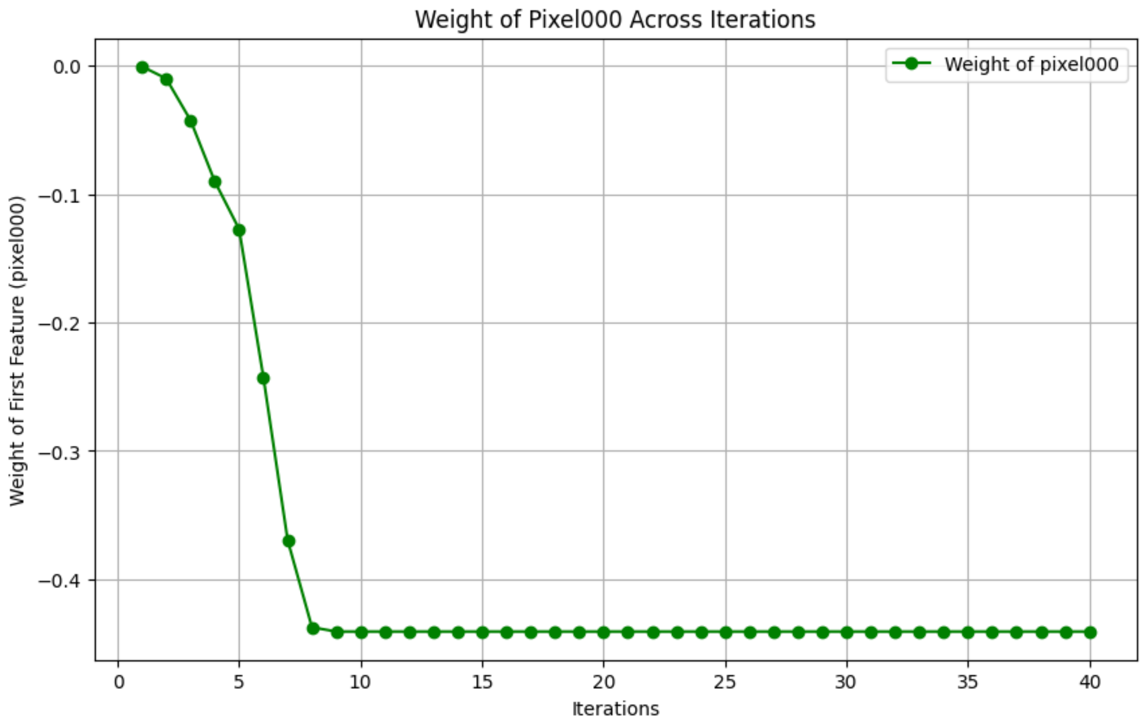
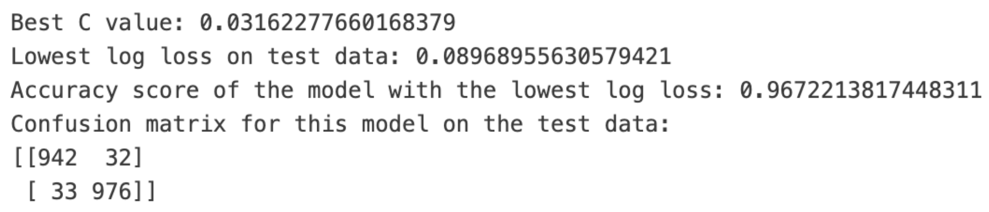


Figure 3: The weight of "pixel000" starts close to 0 and then sharply decreases during the first few iterations.

Analysis of "Weight of Pixel000 Across Iterations" illustrates the dynamics of the weight assigned to the feature "pixel000" as the logistic model undergoes iterative refinement. The weight of "pixel000" starts close to 0 and sharply decreases during the first few iterations. This rapid adjustment suggests that, in the early stages of training, the model undergoes significant recalibration in order to align with the training data. After the drop in weight that occurs up to about the 10th iteration, there's a plateau. The weight seems to stabilize around the -0.4 mark and remains relatively unchanged for subsequent iterations. This indicates that the model has likely reached an optimal weight assignment for the "pixel000" feature, at least within the context of the training data. From around the 15th iteration onward, the weight remains consistent, further reinforcing this idea.

The rapid initial weight adjustment may indicate the model's ability to rapidly discern the significance (or lack thereof) of the "pixel000" feature in the context of the given training dataset. This rapid adjustment could indicate that the feature initially held a sub-optimal weight. The eventual stabilization suggests that further iterations, beyond a certain point (about 10), don't provide substantial refinements for this specific feature's weight. Thus, these unnecessary additional iterations could be eliminated without compromising the weight optimization for "pixel000".

1.3 Question 3



```
Best C value: 0.03162277660168379
Lowest log loss on test data: 0.08968955630579421
Accuracy score of the model with the lowest log loss: 0.9672213817448311
Confusion matrix for this model on the test data:
[[942  32]
 [ 33 976]]
```

Figure 4: The ideal regularization penalty for the logistic model is about 0.3162, yielding an accuracy score of 0.967. The confusion matrix for this model on the test data is shown above.

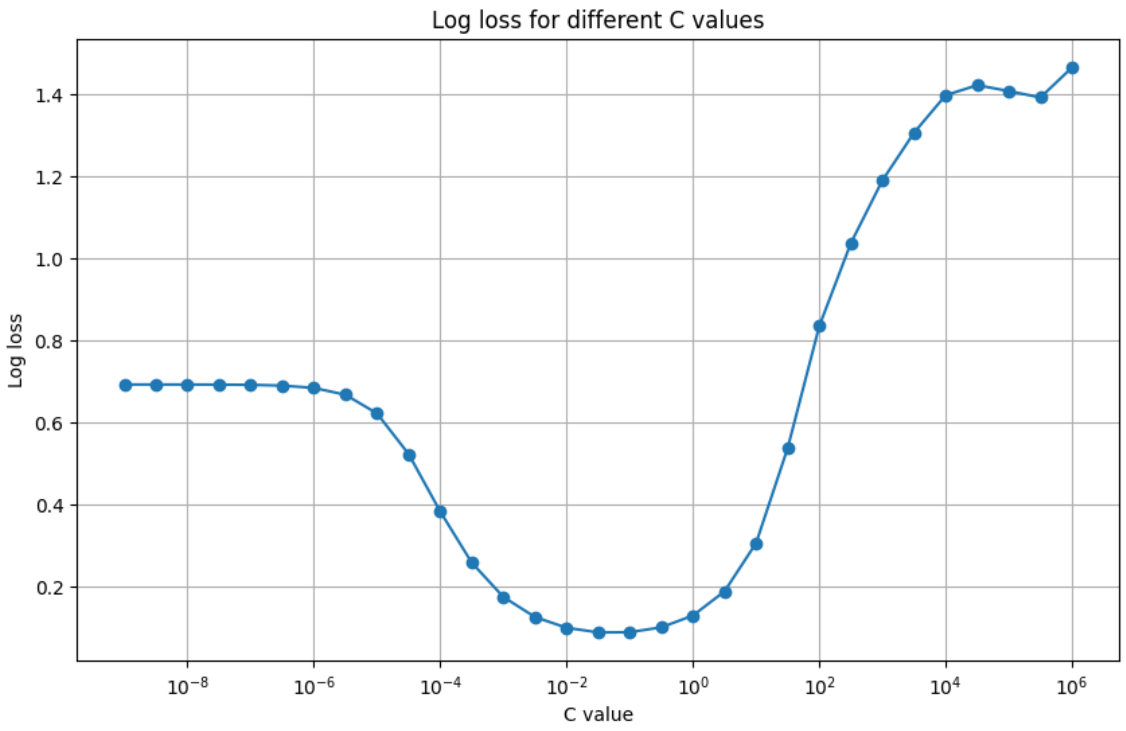


Figure 5: Log loss is minimized at a 'C' value of 10^{-2}, indicating optimal model performance.

In the context of logistic regression, 'C' is the inverse of regularization strength; smaller 'C' values indicate stronger regularization and vice versa. The graph portrays how different levels of regularization impact the model's log loss.

The graph depicts the relationship between the log loss and varying values 'C' for the logistic regression model. The x-axis is plotted on a logarithmic scale and represents the 'C' values, ranging from very small values like 10^{-8} to larger ones up to 10^{6}. The y-axis denotes the log loss values. The curve starts with relatively stable log loss values for smaller 'C' values, reaching a minimum point around the C value of 10^{-2}, indicating optimal model performance. As 'C' increases past 10^{0}, the log loss starts to rise sharply, suggesting a decline in the model's performance with higher 'C' values.

1.4 Question 4

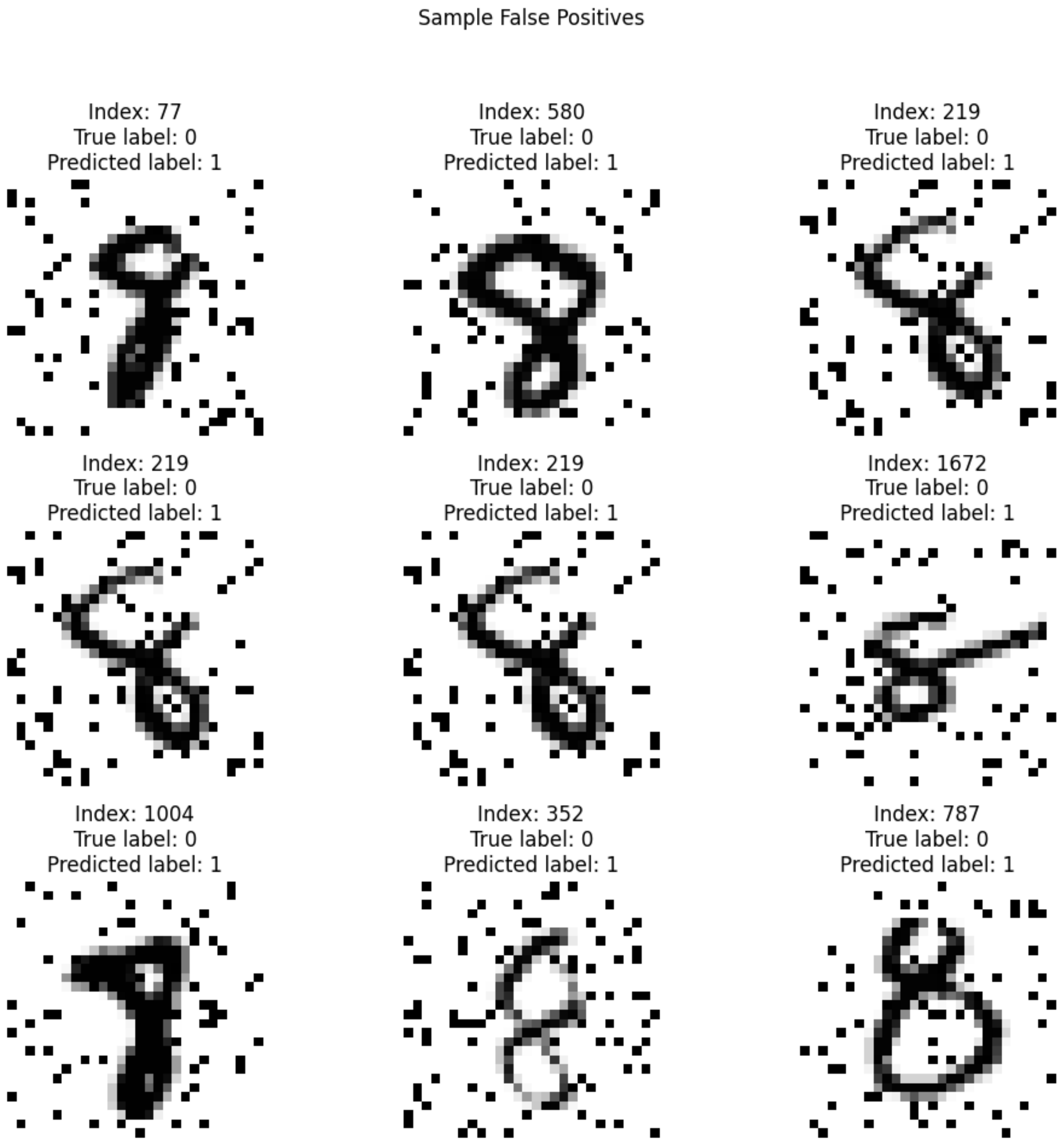


Figure 6: A selection of '8's that were misclassified by the model.

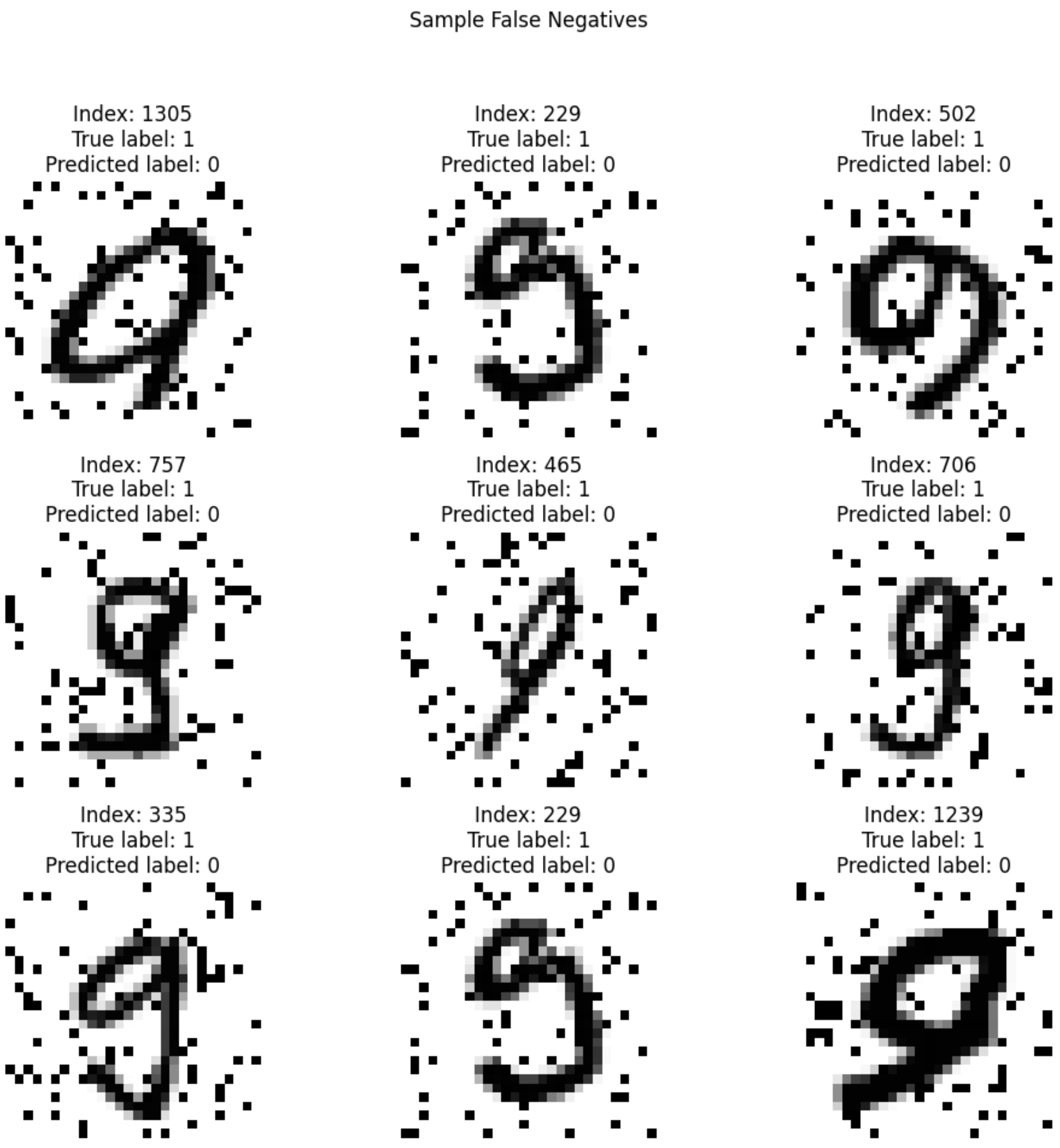


Figure 7: A selection of '9's that were misclassified by the model.

False positives are instances where the classifier incorrectly predicts a positive label (in this case, "1") when the true label is negative (in this case, "0"). Figure

6 showcases several such instances. After analyzing these figures, it is clear that some of these examples have shapes or patterns that could be reminiscent of the positive class. For instance, the shapes in the false positives might have portions or segments that resemble curves or features typically seen in the positive class. This could mislead the classifier. Further, there's also the presence of noise or broken lines in some of these images, which might have contributed to the classifier's confusion. It's conceivable that the model has learned to recognize specific strokes or shapes without considering the entire context or holistic view of the images.

False negatives are instances where the classifier predicts a negative label (in this case, "0") when the true label is positive (in this case, "1"). Examination of Figure 7 makes clear that some of the images do possess the characteristics of the positive class but might be slightly distorted, rotated, or incomplete. This suggests that the model might be sensitive to perfect representations and might struggle with slight deviations or uncommon representations of the positive class. The model's sensitivity could be due to over-fitting or insufficient training data covering these variations.

Overall, it's evident that while the model is generally effective, it's not foolproof. It seems to get misled by certain patterns or shapes, indicating potential areas of improvement. The classifier might benefit from more diverse training data, ensuring it learns from a broader range of representations for each class. Additionally, data augmentation techniques, such as rotation, scaling, or noise addition, could be employed to make the model more robust to variations. Regularization techniques might also help in preventing over-fitting. Finally, a deeper analysis of feature importance might also improve the model and reduce such misclassifications.

1.5 Question 5

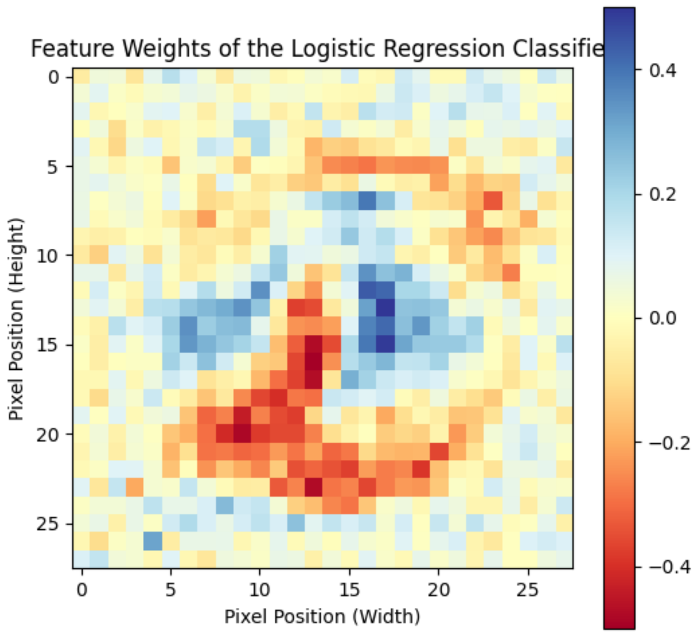


Figure 8: Visualization of the feature weights from a logistic regression classifier. Each cell in the heatmap corresponds to the weight of a specific pixel in the input image. Positive weights (in shades of blue) indicate pixels that contribute to a prediction in favor of the positive class, while negative weights (in shades of red) suggest pixels that contribute to a prediction in favor of the negative class. Neutral weights are represented in yellow.

When the weights from a logistic regression classifier trained to differentiate between images of handwritten '8's and '9's are visualized, the resulting image is a heatmap that indicates which pixels are the most influential in determining whether a given image is classified as one digit or the other. Thus, this representation shows areas of importance that the model recognizes as distinguishing features. Pixels with negative weights are those that contribute to an image being classified as an '8'. These areas likely correspond to parts of the image where '8's would typically differ from '9's. For example, this might be the closed loop in the bottom part of '8', which '9' doesn't have. Pixels with positive weights, conversely, contribute to an image being classified as a '9'. These would correspond to regions of the image where '9's characteristically have features that '8's do not, like the presence of a distinct upper loop and a lack of closure or a straighter line at the bottom.

The reason certain pixels are more influential than others is due to the different shapes of '8's and '9's. When people write these digits, '8's usually have two closed loops with a narrow point in the middle. '9's, on the other hand, usually have an upper loop and a straight or dangling line below. Therefore, the classifier learns from the training examples that certain shapes and pixel intensities are more likely to be associated with one class than the other, and it assigns weights accordingly. These weights are learned features based on the training data provided to the model, reflecting human writing's similarities and differences.

2 Trousers v. Dresses

2.1 Comparison of Results

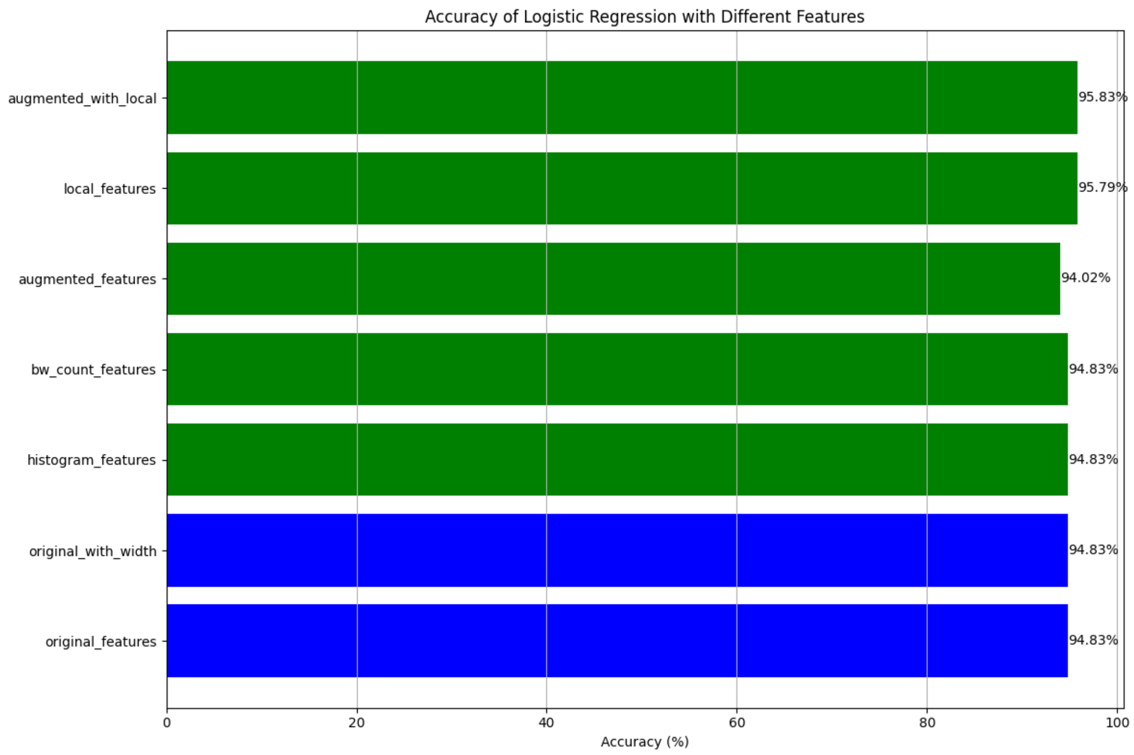


Figure 9: Bar chart displaying the accuracy percentages of logistic regression models using different feature sets, with 'augmented with local' achieving the highest accuracy of 95.83 percent.

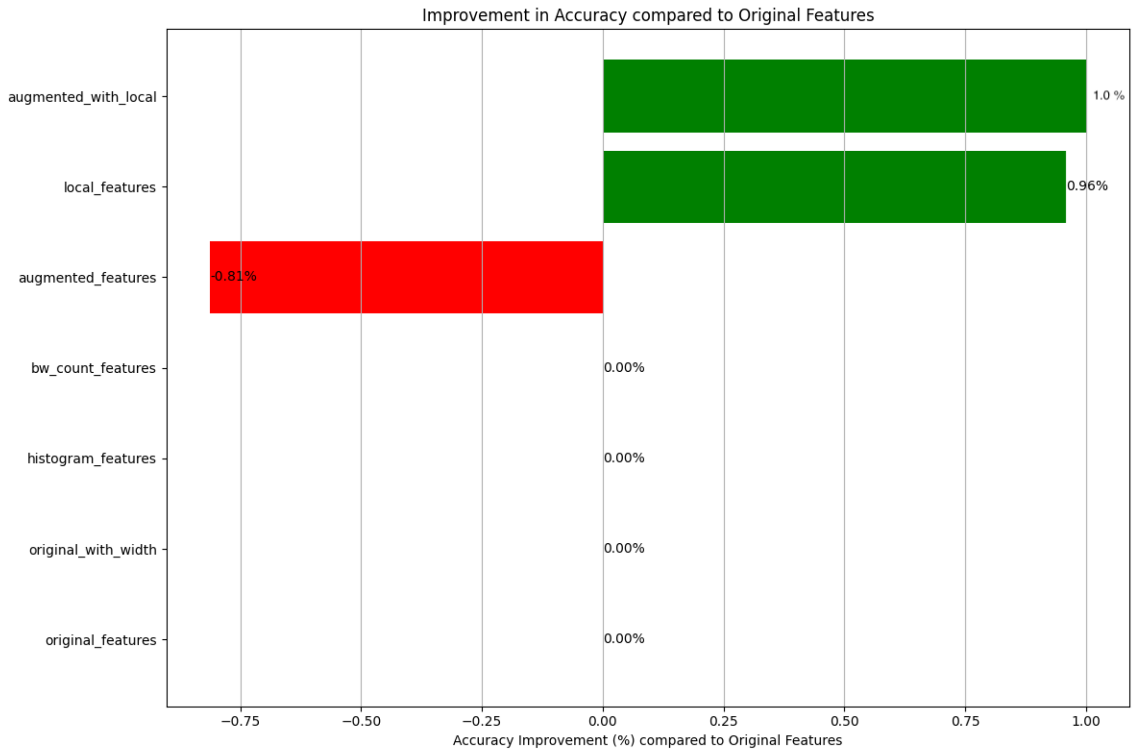


Figure 10: Bar chart illustrating the improvement in accuracy of various feature sets compared to the original features. While 'augmented with local' and 'local features' show positive gains, 'augmented features' displays a decrease in accuracy.

2.2 Object Width

One of the first features explored was the width of the object present in an image. By transforming the image into binary through a predefined thresholding mechanism, the widest section of the object was identified. This binary representation simplified the image into a two-tone version, effectively differentiating the object from its background, making the width estimation straightforward. The hypothesis behind this feature transformation was rooted in the belief that the width of an object within an image might carry distinctive, class-specific attributes. Specifically, the guess about the data for this problem was that dresses (needing to be shoulder-width) would be wider than pants (needing to be waist-width). By normalizing this width relative to the overall image width, the feature remained consistent across varying image resolutions and sizes.

Incorporating some width-based metric with the original image data did not offer any improvement over the baseline. Thus, width-based metrics, such as silhouette or aspect ratios, are not particularly useful for classifying trousers and dresses. Trousers and dresses can sometimes have overlapping or similar silhouettes, especially when viewed from certain angles. For instance, wide-leg trousers might have a silhouette that resembles a straight-cut or A-line dress. Further, not all dresses are wider than trousers, and not all trousers are narrower than dresses. A body-hugging dress might have a similar width-based profile to a pair of trousers, whereas a flare dress might have a wider profile. In addition, a width-based metric, in isolation, doesn't provide any information about the height or length of the garment. This omission means that important distinguishing factors, such as the overall shape and proportion of the garment, might be overlooked. That is, trousers might typically be longer than they are wide, whereas some dresses might have a more square aspect ratio. Finally, if the original features already capture spatial and shape-based information efficiently, then adding width-based metrics might only introduce redundant data without adding significant discriminative power.

2.3 Local Feature Extraction

Local feature extraction proved to be a helpful way to improve the model's classification accuracy for images of trousers and dresses. After dividing each image into smaller segments, a histogram, mean, and variance for each segment was produced. These attributes were used to identify the unique intensity characteristics within these sections, emphasizing the intricate patterns and textures inherent to each class.

Every image, especially those of trousers and dresses that are likely to be constructed of specific and unique material and are of unique, distinguishable and consistent shape, contains localized patterns, textures and structures that could be important in discerning one class from the other. Segmenting the image into various parts and extracting segment-specific allows for these localized patterns to be identified. The assumption was that certain localized patterns are predominant in trousers but scarce in dresses (or vice versa), then such patterns could serve as distinguishing hallmarks. For example, one might expect that some segments for trousers would always contain separation between legs, where no segments for dresses would contain such separation. Or, bold patterns and thus a wide range of pixel intensities may be common for dresses, but not for trousers. The histogram portrays the distribution of pixel intensities within a segment, while the mean offers a general sense of the prevalent intensity, and the variance uncovers the intensity variations - all these combined could spotlight the unique characteristics intrinsic to trousers or dresses.

Experimentation with the number of segments rendered interesting findings. A segmentation into 4 parts did bring about an improvement in the model's accuracy, underscoring the value of localized feature extraction. However, subdividing the images further, into 8 segments, improved the model's accuracy even more, indicating a richer capture of localized patterns which aided in better distinguishing trousers from dresses. Intriguingly, when the segments were increased to 10 or more, there was a decrease in efficacy. This might suggest that beyond a certain granularity, the segments become too fine, potentially losing overarching patterns and introducing noise. Ultimately, segmenting the images into 8 parts and leveraging the resulting local features fortified the model's prediction ability, underscoring the significance of localized patterns in effectively differentiating between trousers and dresses.

2.4 Histograms and Black and White Pixel Count

A holistic view of image attributes was also evaluated in an attempt to improve the precision of classifying images of trousers and dresses. To that end, global image characteristics were also evaluated. Histograms were created to capture the broader intensity distribution of the entire image. In addition, pixels were counted at the edges of the intensity spectrum (black, white). These counts were derived by establishing definitive intensity thresholds.

Classifying images of trousers and dresses necessitates an understanding of both their detailed structures and their overarching appearance. The global intensity histogram is an embodiment of this larger picture. It serves as a panoramic lens, detailing how intensities are scattered across the image. It was hypothesized that such a broad perspective could be instrumental when the overarching color palette or intensity spread becomes a discerning factor between trousers and dresses. Further, trousers and dresses may have distinct design elements, embroideries, or patterns that could manifest as pronounced black or white areas in an image. Counting these extreme-intensity pixels could provide insight into these standout features. For example, a dress with a distinctive white lace pattern might have a higher count of white pixels, whereas trousers with deep pockets or seams might register a spike in black pixel counts.

Interestingly, using black and white pixel count features or histogram features offered no improvement over the baseline. While these features are certainly capable of capturing certain characteristics of the images, they did not prove to be better than the original features for differentiating trousers from dresses. There are several reasons that this may have been the case. First, trousers and dresses can both exhibit a wide range of colors and patterns. If the dataset has both garments in similar colors, a black and white pixel count might not differentiate them effectively. Also, since histograms and black and white pixel counts provide aggregate information about an image, they might miss finer details. Dresses and trousers might have distinct textures, patterns, or localized features that are not captured by these aggregate metrics. In fact, these finer details may have been better captured by local feature extraction. Also, If the original features already capture information about the color distribution and light-dark contrast in the images, then adding histograms or black and white pixel counts might introduce redundant information, offering no new discriminative power.

2.5 Augmented Dataset

For the task of classifying trousers and dresses, "augmented" refers to the dataset after it has been enriched using transformations (e.g. horizontal flipping of images). The initial rationale behind augmenting the dataset was in recognition of the fact that trousers and dresses can appear in various orientations in real-world scenarios. Thus, augmentation like horizontal flips might ensure the model is robust to such variations. However, in looking at the images being evaluated, the orientations of the trousers and dresses do not seem to vary. Another, more relevant benefit of introducing such variations was that it might reduce the risk of the model memorizing the limited training set. Thus, over-fitting could be prevented.

Augmenting the dataset did not improve the performance of the model. In fact, doing so made the model perform less well, decreasing its accuracy by 0.81 percent over the baseline. Since all images in the original dataset were oriented in a consistent manner, horizontal flipping might have introduced a variation that the model perceives as less common. For example, if every trouser in the original dataset had a particular pocket design on the right, flipping would move it to the left, which might not be representative of most trousers in real-world scenarios. Some trousers or dresses might have designs that are inherently asymmetrical. For instance, a dress might have a unique design or cut on one side. Flipping such images could confuse the model, as it would then be seeing variations of the dress that don't exist in the real world.

2.6 Augmented Dataset With Local Feature Extraction

"Augmented with local" signified the combination of the augmented dataset with locally extracted features. Interestingly, doing this improved the accuracy of the model 1 percent over the baseline. The combined use of these two things, as indicated by the "augmented with local" performance, synergistically enhanced the model's ability to classify trousers and dresses more accurately.

This phenomenon can perhaps be explained in a few ways. As discussed before, local features tend to capture intricate patterns, textures, and unique attributes in an image. By augmenting the data and then extracting local features, the model may have been exposed to a richer set of these attributes. This could have helped the model better distinguish between subtle differences in trousers and dresses. Also as discussed before, augmentation introduces variations in the dataset. When combined with local feature extraction, the variability in the dataset may have become even more pronounced, allowing the model to learn from a broader set of patterns and thereby reducing over-fitting. Further, local features might sometimes be too specific, leading the model to over-fit to training data. However, perhaps when used in conjunction with augmented data, this risk may be reduced because the augmented data introduces noise and variability, preventing the model from fixating too much on extremely specific local features.

The "Augmented with local" version of the model was the best performing, achieving an error rate of about 3.75 percent on the final testing input set. This means that about 3.75 percent of the predictions made by the best model on the testing data were incorrect. The other versions of the models would have achieved error rates that were higher than 3.75 percent, had they been submitted for testing using the final testing input set. We would expect the error rates to increase with decreasing accuracy improvement over the baseline.

2.7 Hyperparameter Tuning

Hyperparameter tuning involves adjusting the preset parameters of an algorithm to enhance its performance. These parameters, unlike the ones learned during the training process, need to be set before training starts. For instance, in logistic regression, key hyperparameters include the penalty type and regularization strength. Choosing the right hyperparameters can significantly improve the model's accuracy.

The GridSearchCV method was employed for hyperparameter tuning. This method conducted a search over a specified range of hyperparameter values, training the logistic regression model for each combination. Specifically, penalty values 'l1' and 'l2', corresponding to L1 (Lasso) and L2 (Ridge) regularization, respectively, were considered Also, 'C' values of 0.001, 0.01, 0.1, 1, 10, 100 were explored. GridSearchCV evaluated the logistic regression model using every combination of penalty and 'C' values, resulting in a total of 12 different combinations. Once all the combinations were evaluated, GridSearchCV identified and selected the combination that produced the logistic regression model with the highest accuracy on the validation set.