Project 1

Part 1

1.1 max_iter

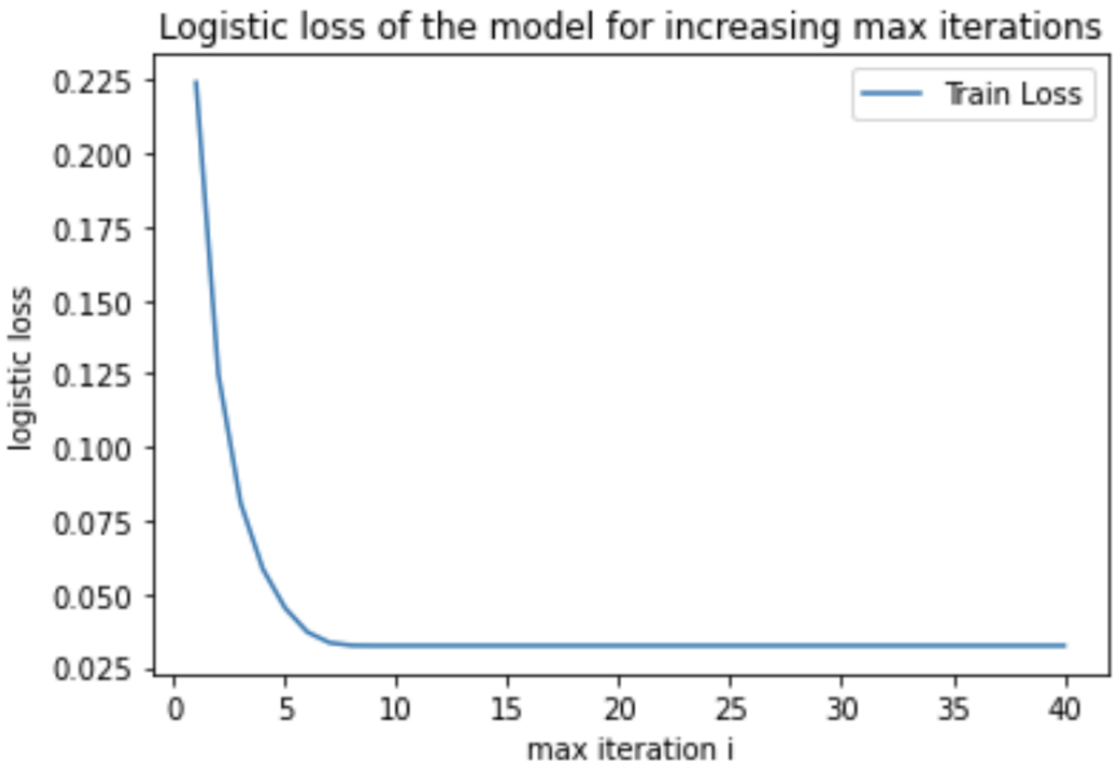


Figure 1: Logistic loss of the model for increasing max iterations

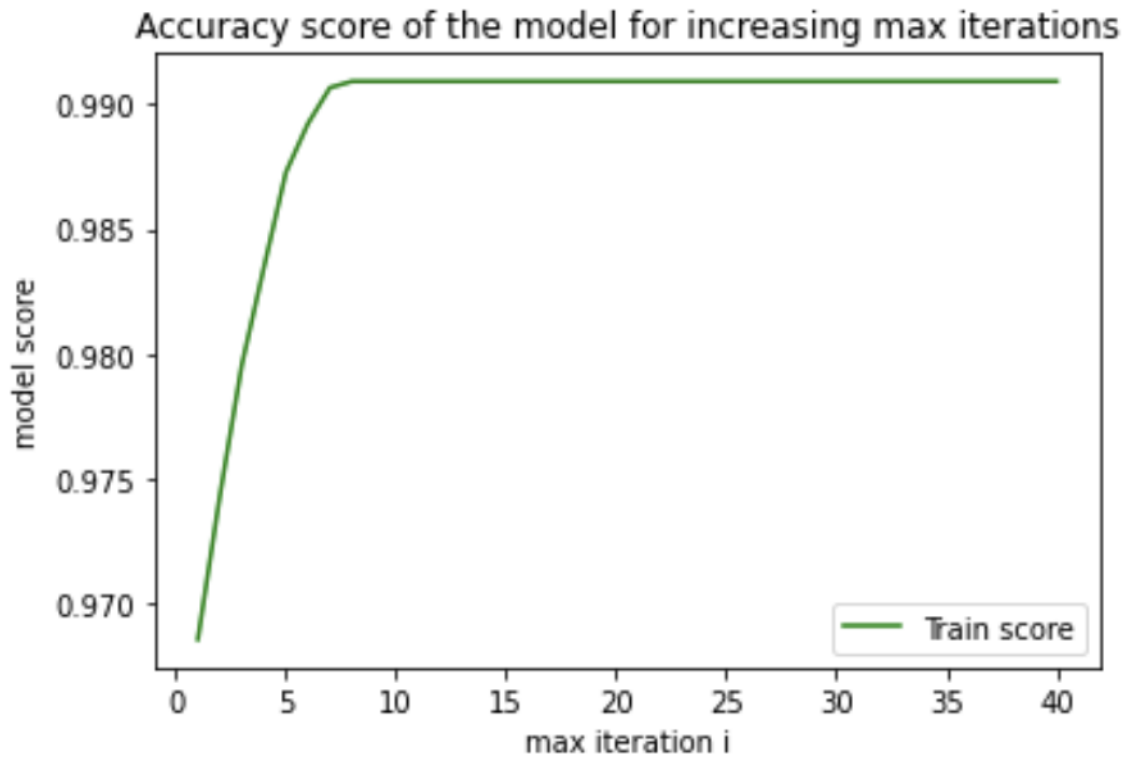


Figure 2: Accuracy score of the model for increasing max iterations

Discussion:

As the max_iter increases, the accuracy rate increases and the log loss decreases. We see the log loss drastically shooting down and the accuracy rate shooting up during the first 9 iterations. The accuracy rate and log loss stay the same with nine or more iterations.

As suggested by the 9 ConvergenceWarnings, with only 9 iterations, the model fails to converge, which results in high error rate and low accuracy rate. As we increase the number of iterations, the model converges and begins to generate constant error result and accuracy rate.

1.2 coef_ for pixel000

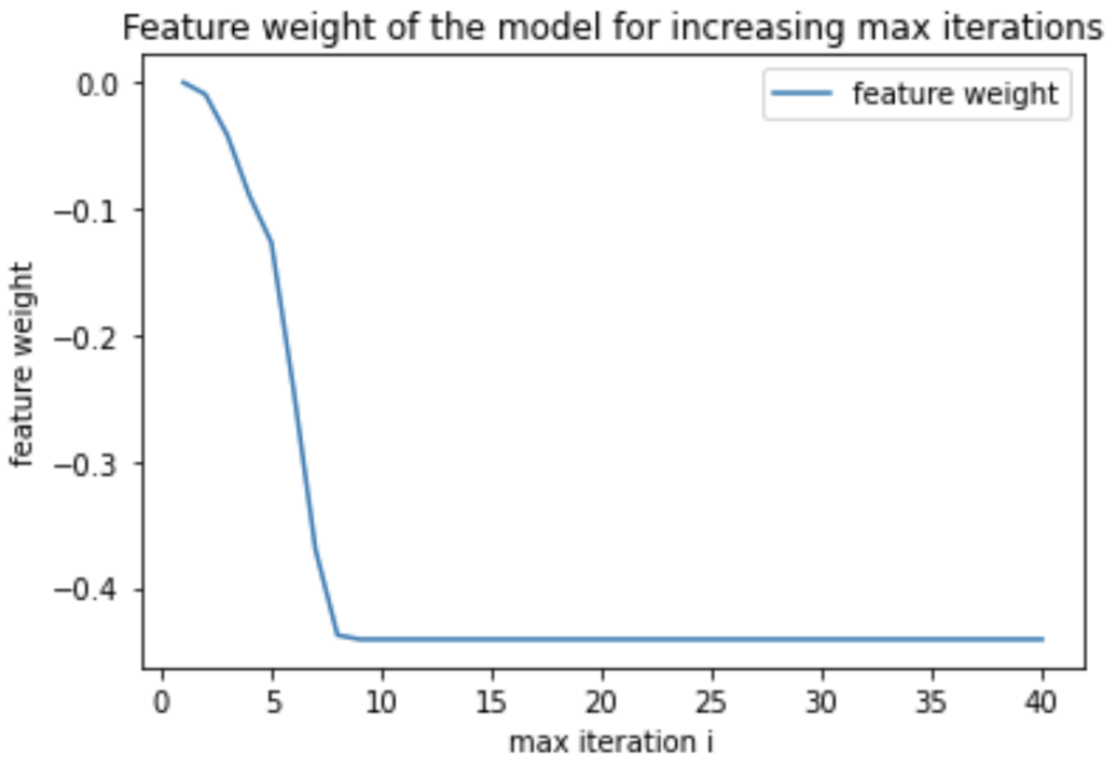


Figure 3: Feature weight of the model for increasing max iterations

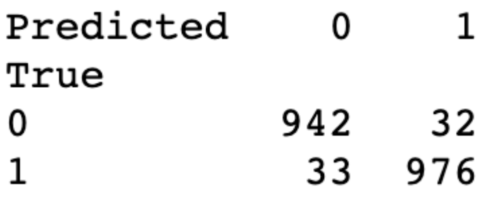
Discussion:

As the max_iter increases, the feature weight decreases. We see the feature weight drastically shooting down during the first 9 iterations, and stays the same with nine or more iterations.

As suggested by the 9 ConvergenceWarnings and corresponds with previous results, with only 9 iterations the model fails to converge, which results in high feature weight. As we increase the number of iterations, the model converges and begins to generate constant feature weight.

1.3 Best C

After comparing the models for different c values in logspace(-9, 6, 31), we get the best C value is at 0.03162277660168379, with the log loss of 0.08968955614249495 and an accuracy rate of 0.9672213817448311.

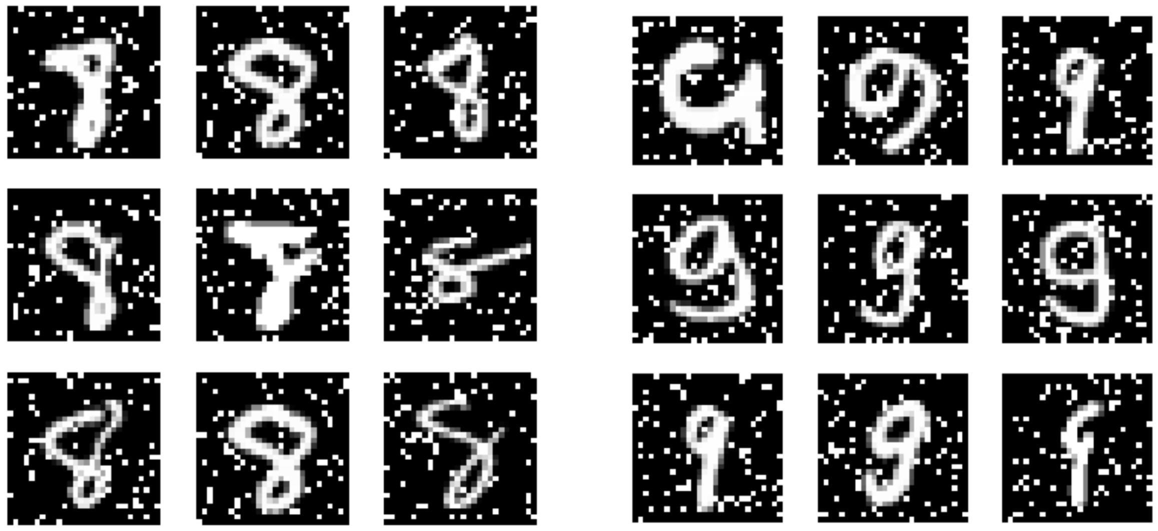


```
Predicted     0     1
True
0           942    32
1            33   976
```

Figure 4: confusion matrix for the best C value on the test data

1.4 failed prediction

Randomly selected 9 samples from failed prediction examples are shown below.



False positives False negatives

Figure 5: samples for failed prediction

Discussion:

For all false negative examples, the circle on the bottom was either overlapped so that it was no longer perceived as a loop. Therefore, although they are in class 0 (digit 8), they are predicted as 9.

Similarly, for all false positive examples, the tail is bent on the bottom, just like how we would normally write a 8. Therefore, although they are in class 1 (digit 9), they are predicted as 8.

1.5 colormap for the coefficients

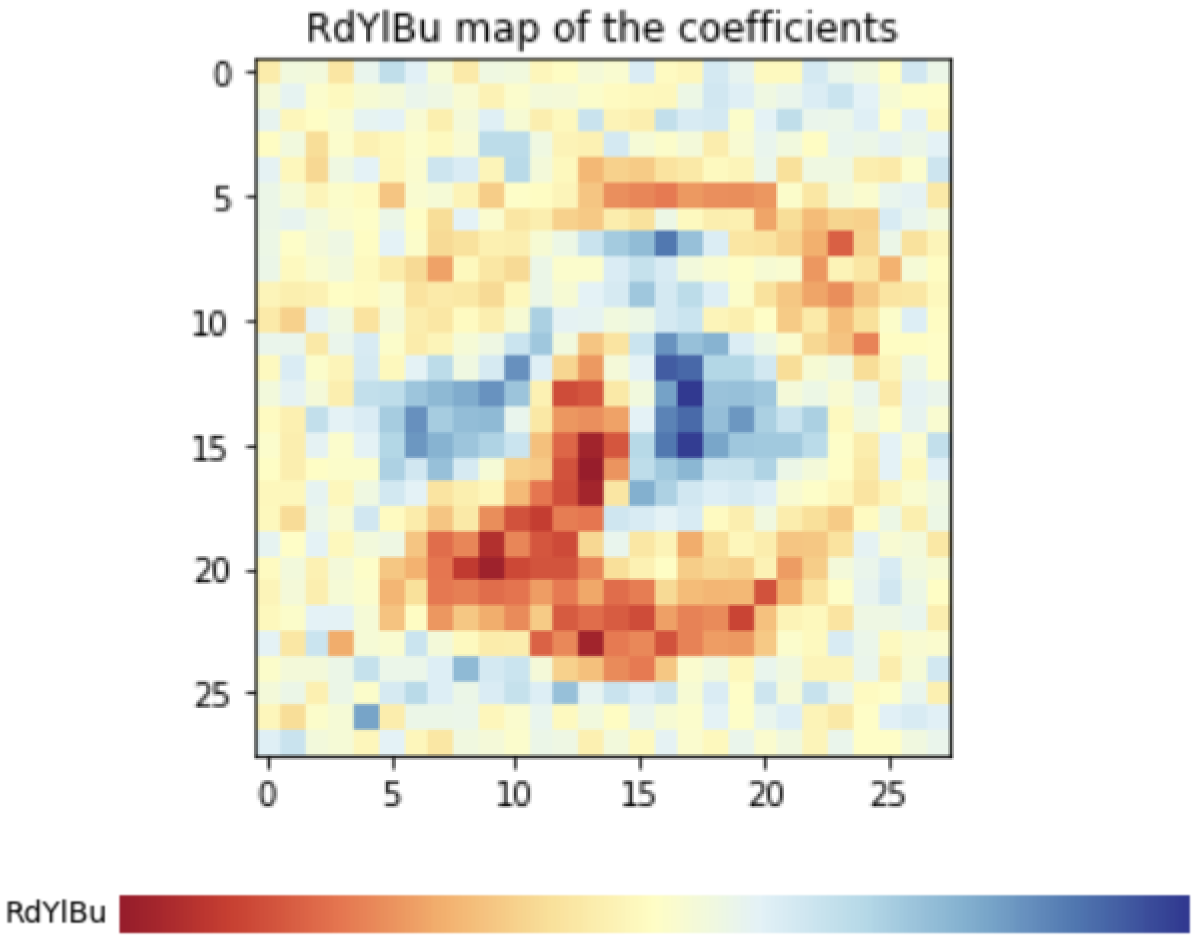


Figure 6: samples for failed prediction

Discussion:

For RdYIBu color maps, color tends to red as the weights move closer to 0(digit 8), and color tends to blue as the weights move closer to 1(digit 9).

In the graph, the lower left half is intensely red, which means these weights are critical in deciding if it’s 8. This conforms with how people normally write 8 and 9, 8 has the left lower half of the curly circle while 9 does not. Similarly we see the middle part of the data is intensely blue, which means they have positive weights and correspond to a 9. This also makes sense since it conforms with how people normally write 8 and 9 ---- people usually write the upper circle of 8 and 9 differently in orientation and size.

Part 2

About the Data

It's important to know our data before transforming features. The training dataset contains 12000 data, which are evenly split among the two classes. Each data point has 784 weights of range [0,1]. By reshaping the weights into 28*28 images, we are able to observe what's classified as sneakers or sandals.

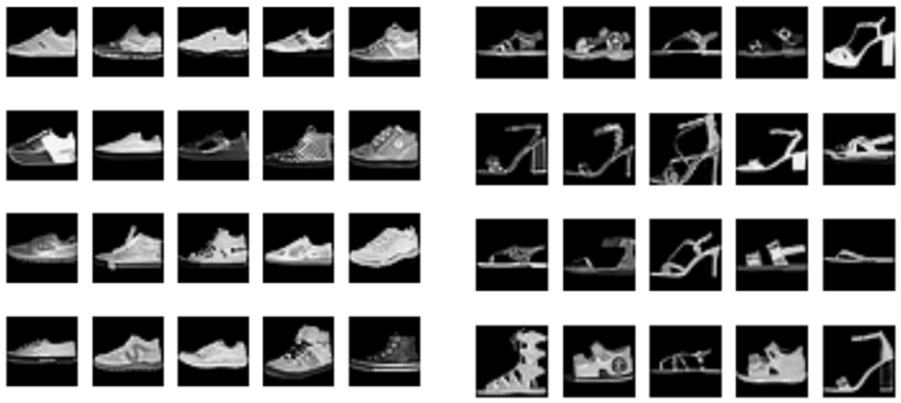


Figure 7: samples of sneakers and sandals

Printing out the data suggests that the shoes are left oriented. They differ in the shape, size, and intensity of the pixels, which gives us some hints on the ways we can choose to transform features.

Pre-test before doing features transformation

1. Basic Logistic Regression

   Using C=1, the logistic regression then generates an error rate at 0.042499999999999998 and an AUROC of 0.993241, which serves as *our baseline error rate.*
2. Ways that improve the result: Standardized and Cross Validation
   a. **Best C**: In order to simulate test runs, I split my original training data into training and validation with a ratio of 0.4. Picking the best C value at 0.03162277660168379 gives us an error rate of **0.04200000000000004** and an AUROC of **0.99222.**

b. **Normalization**: Although the data is already in the same [0,1] range, I still used minmax scaling, which further minimizes the test error rate to **0.039000000000000035** and the AUROC to **0.993629.**

Feature Transformation

1. Histogram

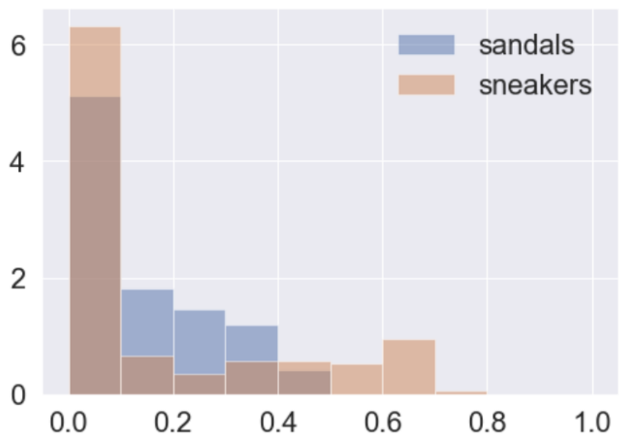


Figure 8: histogram of mean weights in sneakers and sandals samples

After averaging over the dataset and looking at the histogram of the data, we see patterns of sandals generally having more dark pixels, while sneakers have a wider range of intensity. Thus, following that, I add features that count the black and white as well as dark and light pixels in the graph using 0.5 as a threshold.

After trying a different combination, I choose to use the black & light features, which gives an error rate of **0.036000000000000003** and an AUROC of **0.994822.** Using Normalization, we yield a better result of **0.034499999999999975** in error rate and a **0.9953590000000001 of** AUROC score.

2. Feature Amplification

Inspecting the images, we observe that the pixel intensity doesn't contribute much to the classification, it's the fact that some pixels are colored and some are not that helps us ensemble the shape. Thus, I set a threshold to amplify weights to make the data less noisy: weights that are larger than the threshold were set to 1, while weights that are smaller than the threshold are set to 0.

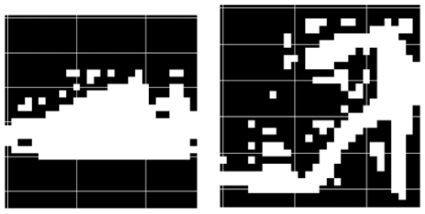


Figure 9: Image of sneakers and sandals after feature amplification

After doing gradient descent on different small weights, I choose to use a threshold of 0.03, which gives an error rate of **0.029499999999999997** and an AUROC of **0.9968330000000001.**

3. Data Augmentation - flipping & rotation

   I doubled the training set size by flipping each image horizontally and rotating the image. Though with the optimization methods (including normalization, cross validation, and penalty rate) added, the error rate remains super high. The lowest I got is **0.054499999999999999** and an AUROC of **0.975999.**

   I inspected the testing data and realized the shoes in testing are oriented towards left, which means flipping or rotating all the data and training the model with shoes oriented for both sides would increase the error based on our testing set, though it might help in real life.

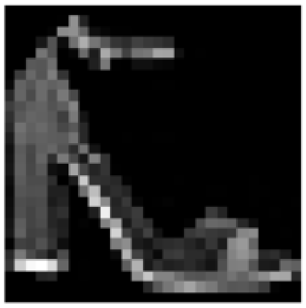


Figure 10: Image with wrong orientation

I followed up with a closer inspection on both training and testing set, and by comparing the nonzero weights in the left half and right half of the image, I filtered out the data with wrong orientation and only flipped those(x_train[4905]). Pairing with the threshold method, I got an error rate of **0.030499999999999997** and an AUROC of **0.994207.** The result improved by a bit, but not much.

4. Spatial Features

   Last but not the least, none of the feature transformation methods above leverage connectivity. But we all know that we perceive shoes as shoes based on the ways pixels are organized. Thus, it might help to add features that view adjacent matrices as a whole and account for spatial features.

I divided the 28*28 grid into size n * n blocks. The best result found at 4*4 grid after looping through 2 to 28 for n. I then tried to add features on average, maximum, and minimum. The best result is obtained using the maximum, which, after adding an inverse penalty of C = 0.31, yields an error rate of **0.03400000000000003** and an AUROC of **0.99501.**

	Mean	**Max**	**Min**
Error rate	**0.04200000000000004**	**0.03400000000000003**	**0.040000000000000036**
AUROC	**0.9928710000000001**	**0.99501**	**0.993757**

Table 1: Error rate and AUROC on testing data with added spatial features of local mean, max and min

Comparison among different models

	Error rate	**AUROC**
Basic Logistic Regression	0.0425	0.9932
Standardized Logistic Regression	0.0420	0.9922
MinMax Scaling	0.0390	0.9936
Histogram	0.0345	0.9954
Feature Amplification w/ threshold	0.0295	0.9968
Doubling data	0.0545	0.9759
Flipping data with incorrect orientation	0.0305	0.9954
Spatial Feature- mean	0.0420	0.9929
Spatial Feature- max	0.0340	0.9950
Spatial Feature- min	0.0400	0.9938

Table 2: Summary of Error rate and AUROC on testing data with different methods

Conclusion:

By comparing all the results above for each feature transformation, we see that the highest model accuracy is attained through setting a threshold and amplifying features. This makes sense because as analyzed above, classification of either a sneaker or a sandal depends greatly on the all or none general pattern rather than the variance in individual pixel intensity. The second best error rate is obtained from removing extraneous data points, which suggests the importance of cleaning up the data. The third best model is obtained with local maximum, which suggests the importances of looking at relatability among features.

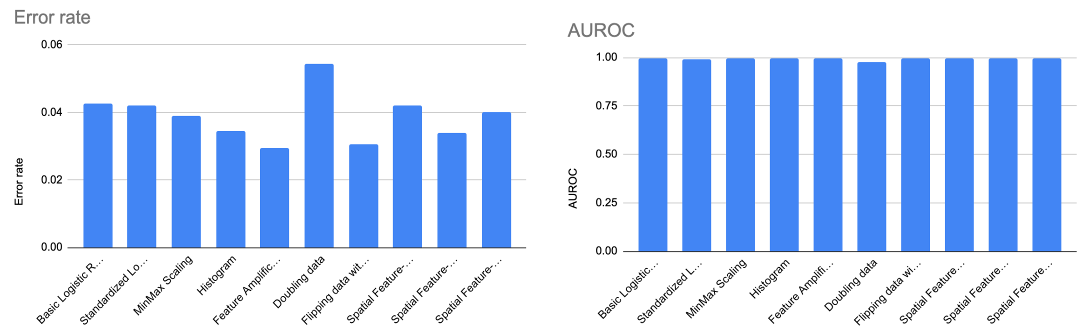


Table 3: Summary of Error rate and AUROC on testing data with different methods

However, the best model I got here doesn't apply to all datasets. For example, a dataset with shoes oriented both ways may obtain a better result through data augmentation. We have to inspect the data and choose the appropriate model based on the model's unique properties.

Further Work:

I did not have enough computing power to run models on smaller weights and finer division. Thus, with more time and computing power, we can approximate for a better result.