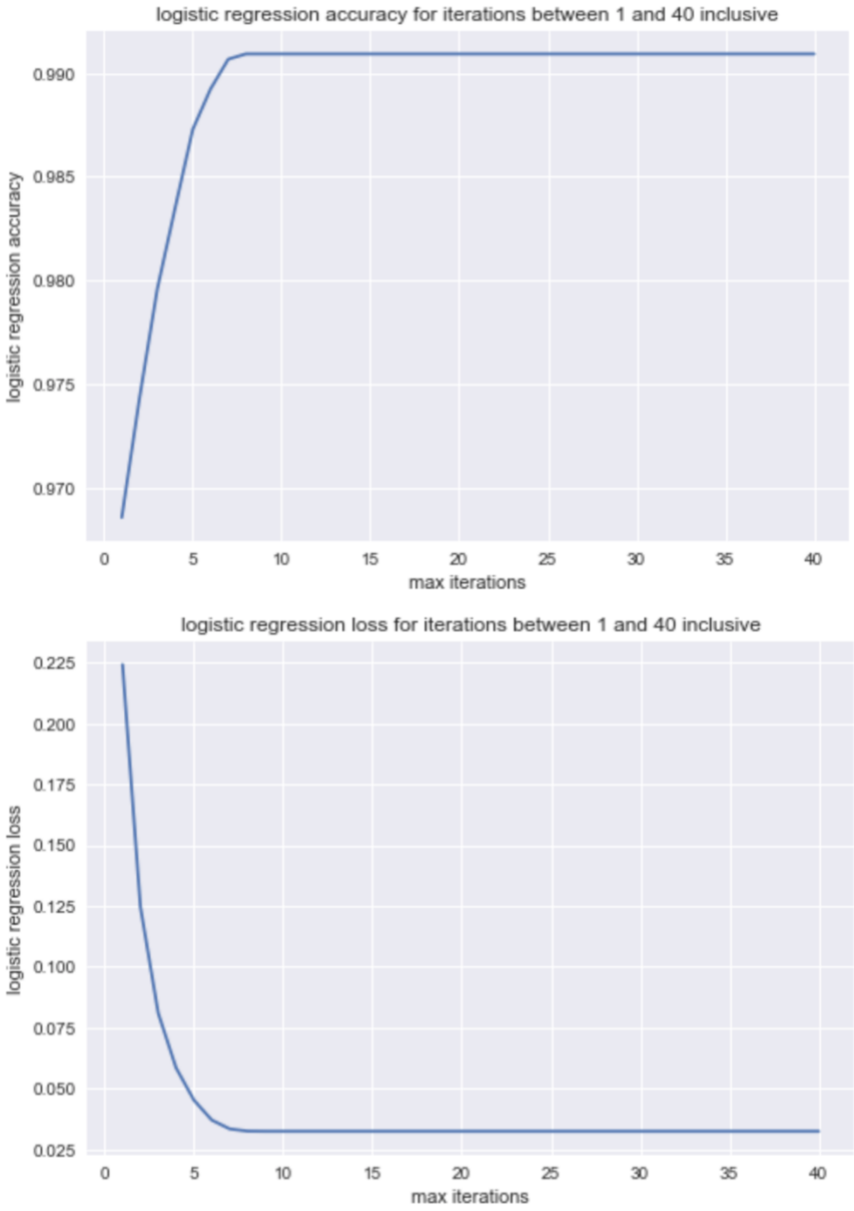
Project 1

Vickie Liu rliu07

2021.4

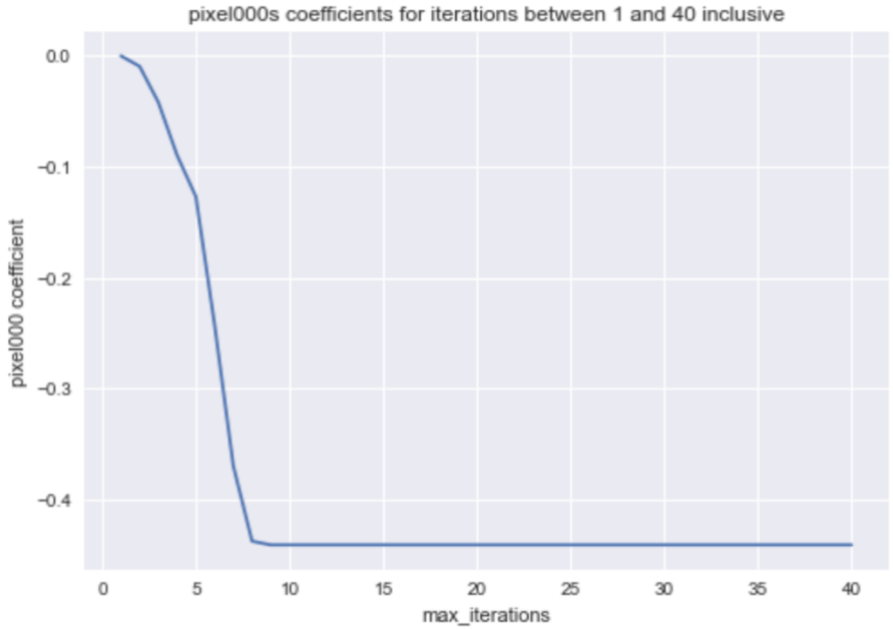
Part One: Regression for Digit Classification

1. Fit Logistic Regression



In the figure on the top, accuracy increases as maximum iterations increases from 1 to 8 and then it levels off at accuracy equals to 0.9909322033898305 for maximum iterations greater than and equal to 8. For the figure on the bottom, log loss decreases as maximum iterations increases from 1 to 9 and levels off at log loss equals to 0.03245505358104707 for maximum iterations greater than and equal to 9. In addition, accuracy and log loss are inverse (as accuracy increases log loss decreases, vice versa). Both figures suggest that for this particular model, by increasing the iterations beyond 9 will not affect the accuracy. This is because gradient descent has converged (reaching minimum) and increasing the iteration will no longer update weights.

2. Coefficients of pixel000



The figure shows that the coefficients of pixel000 changes (decreases in this case for pixel000) for iterations between 1 to 9 and levels off for iterations greater than and equal to 9. This graph shares the same general feature as the two figures discussed in the previous section where value changes for iterations before 9 and stays constant afterwards. It has the shape because gradient descent converges (reach minimum) at the 9^{th} iteration and for iterations beyond 9, coefficients weights are no longer updated therefore it is represented as a flat line in the graph.

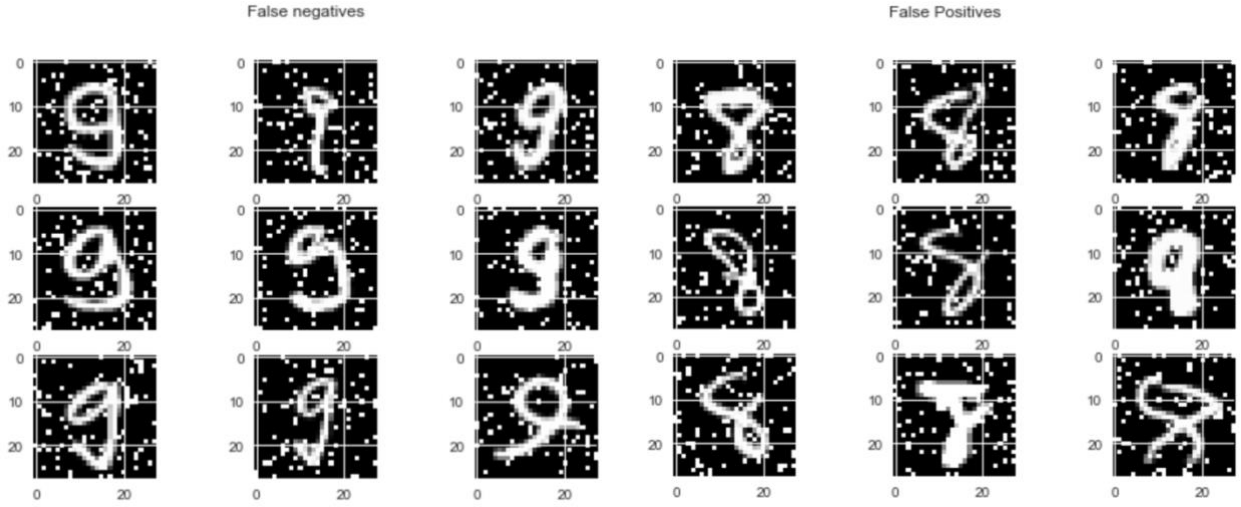
3. Inverse penalty strength C

```
Best C-value [giving the lowest log loss for test data]: 0.0316
Log loss for the best C-value: 0.090
Accuracy for the best C-value: 0.967
```

Using the threshold of 0.5, the following confusion matrix is produced

```
Confusion Matrix:
Predicted    0    1
True
0          942   32
1           33  976
```

4. Mistake Analysis

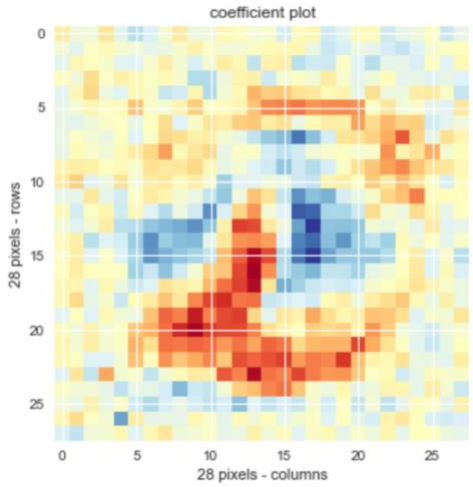


On the left is a 3 by 3 subplot of false negatives and on the right is a 3 by 3 subplot of false positives.

For false negatives (should have been classified as positive, 9, but instead has been classified as negative, 8), it can be seen that the many 9s have the bottom part curved up and for some, the curve up is very close to the circle in the top. This presents challenge for the logistic model to discern whether it is a nine or an eight as they possess the characteristic curvy bottom of 8. In addition, some falsely classified 9s top circle is really small. This feature has also made it difficult for the classifier to correctly classify it as nine.

For false positives (should have been classified as negative, 8, but instead has been classified as positive, 9), it can be seen that 8s does not have very rounded bottom circle and for some, their top part is not a closed circle. These all contribute to wrong classification of the classifier as they do not possess characteristic features of a usual eight. In addition, most of these wrongly classified images do not have a diagonally positioned straight line going from bottom left to top right (the straights still exist but their positions are shifted and are out of the usual identification zone for the classifier). This may result in the classifier recognizing them as 9 instead of 8.

5. Final weights produced by the classifier



In the graph above, the red pixels (negative) are associated to 8 and blue pixels (positive) are associated to 9. The more darkly colored pixels are near the center of the figure above. This means that when deciding which pixels to look at for classification, the pixels in the center makes the greatest influence. The lighter pixeled yellow and blue are where coefficients are near 0, meaning that these pixels do not make as large an influence when deciding the probability of a given image.

Also, from the darker red pixels, it can be seen that the top semi-circle of the top circle in eight and the bottom semi-circle of the bottom circle in 8 as well as a line gong from the bottom left to top right (linking the two circles) are the main features that the classifier looks for when deciding whether the given image is 8 or not.

At the same time, the features of 9 (darker blue pixels) concentrate near the center region that have not been colored as red. I think this is the case because the features for 8 have been extracted (features are more discernable, especially the curvy bottom, thus can be more easily extracted) and anything that is not 8 will be classified as 9 instead.

Part Two: Sneakers versus Sandals

1. Background and Data Inspection

For this part of the project, we are given arrays representing images (size 28×28) of sneakers and sandals with sneakers being labeled as 0 and sandals being labeled as 1. x_train and x_test are provided with data size of 12000×784 and 2000×784 respectively. For y data, y_train is provided while y_test is not. However, $predicted\ y\_test$ can be compared with y_test through leaderboard on Gradescope which will calculate the error rate as well as AUROC value.

The training data contains 6000 (50%) sandals and 6000 (50%) sneakers. To better understand the classification of the data, visualization is made for 18 randomly selected (9 each) sneakers and sandals from x_train:

Figure 1: randomly selected sneakers from x_train

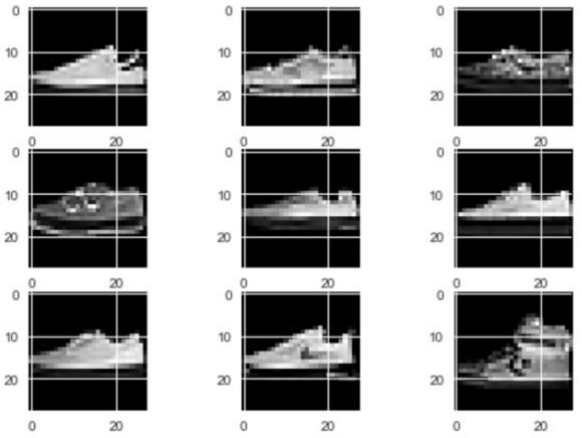
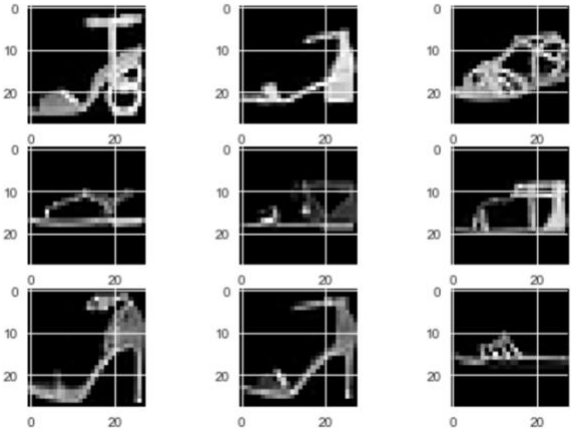


Figure 2: randomly selected sandals from x_train



2. Logistic Regression on Initial Data and Parameter Exploration

The goal of this section is to inspect the initial data and to find the optimum C-parameter by building logistic regression models on the initial data before applying any feature transformation.

2.1 Baseline

Before exploring different C values, a baseline model is constructed by fitting a logistic regression model (solver being 'liblinear') on x_train, y_train using default parameters and without applying any feature transformation and generating predicted y values on x_test, the leaderboard result is:

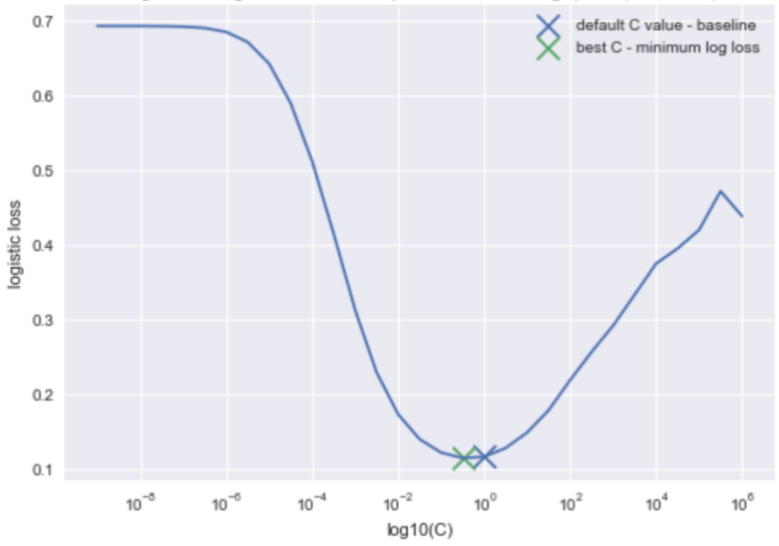
Model Name	**Error Rate**	**AUROC**
Baseline	0.04250	0.993241

Table 1: Baseline result

2.2 C-parameter

C is a float representing the inverse of regularization strength and smaller value means greater regularization strength[1]. In order to find the best C value, the original x_train is fitted to 30 logistic regression models each with a different C-parameter regularly spaced between log-space of -9 and 6. Other parameters including solver ('liblinear') and train-test split (80%-20%) have been kept constant to ensure consistency. The range of log from -9 to 6 is adopted because it has been suggested in previous homework assignments as well as part one of the current project. In addition, by exploring this range, a minimum C value of 0.3162 is identified, shown in figure 3 below. The convexity of the graph produced also reassures that this is a reasonable range as log loss decreases before reaching 0.3162 and increases afterwards, rendering 0.3162 to be the minimum within the range.

Figure 3: logistic loss for C-parameters in logspace(-9, 6, 31)



After finding the best C value on the training data, a model fitted on the entire y_train, x_train is constructed with C equals 0.3162 and the leaderboard result is:

Model Name	**Error Rate**	**AUROC**
Optimum C (C = 0.3162)	0.0385000	0.993573

Table 2: Optimum C-value result

By comparing the results in Table 2 to the baseline result in Table 1, it can be seen that C value of 0.3162 produces better result than the default value of 1 as error rate decreases from 0.0425 to 0.0385. In addition, as shown in figure 3, for training data, C value of 0.3162 produces lower log loss than 1.This suggests that model gives better result with increased regularization strength. This may have occurred because stronger regularization decreases variance and thus helps to reduce overfitting. Given the finding, C-value of 0.3162 will be adopted in the following sections of the project.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

3. Feature Transformations

By having the parameters being determined, in this section, several feature transformations will be performed and discussed, including data augmentation, feature addition of the summation value for each row and column, and data transformation base on certain threshold value.

To ensure consistency, for all models tested in this section the following parameters are adopted:

1. C-value: 0.3162
2. Solver: liblinear
3. Train-test split*: 80% training and 20% testing
   (*: only for 3.3's threshold finding, all leaderboard prediction model are generated with all training data available as greater data size helps to avoid overfitting)

3.1 Data Augmentation – Double the Training Size

Method and Motivation

In this feature transformation, all images will be horizontally flipped to double the training size.

This feature transformation is tried because by having more training data, the model can potentially correctly classify more edge cases. In addition, by examining figure 1 and 2 which display randomly selected images, it can be concluded that most shoes in the training set have the toe of the shoe pointing to the left. This means that if there are image with toe pointing to the right in the testing set, the classifier may not be able to correctly identify it.

Leaderboard Result

Model Name	**Error Rate**	**AUROC**
Data Augmentation (original + horizontally flipped)	0.054	0.984488

Table 3: Data augmentation (original + horizontally flipped) result

Figure 4: Error Rate and AUROC for baseline and data augmentation

Discussion
As seen from figure 4 in the previous page, the performance of the classifier in fact decreases after training the model with the augmented training dataset when compared with both baseline and best C value's leaderboard results. Compare with baseline, for model based on augmented training data, its AUROC decreases from 0.993241 to 0.984488 and error rate increases from 0.04250 to 0.054.

This may be due to the fact that most of the images in the testing set also have the tip of the shoe pointing left, and by doubling the training data with half facing left and half facing right, it can no longer take advantage of the fact that most are facing one way and the coefficients become less efficient.

3.2 Feature Addition – Horizontal and Vertical Summation

Method and Motivation
In order to better illustrate the added feature, an example of *horizontal* summation features of an 3×3 image is shown below:

$$\begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.5 & 0.5 & 0.5 \\ 0.9 & 0.9 & 0.9 \end{bmatrix} \begin{matrix} 0.1 + 0.1 + 0.1 \\ \rightarrow 0.5 + 0.5 + 0.5 \\ 0.9 + 0.9 + 0.9 \end{matrix} \begin{matrix} 0.3 \\ \rightarrow 1.5 \\ 2.7 \end{matrix}$$

For this example, after the original 1×9 row vector, $[0.3 \quad 1.5 \quad 2.7]$ are appended. Thus, the row vector now has the size 1×12 with the newly added features. In terms of the actual dataset x_train, because each image has 28 rows, 28 new values for each array will be appended after horizontal aggregation and the resulting size of x_train becomes 12000×812. In addition, after each row has appended the new value, the newly added columns which does not guarantee a range of 0 to 1 will be normalized so each value lies within the range 0 to 1 inclusive.

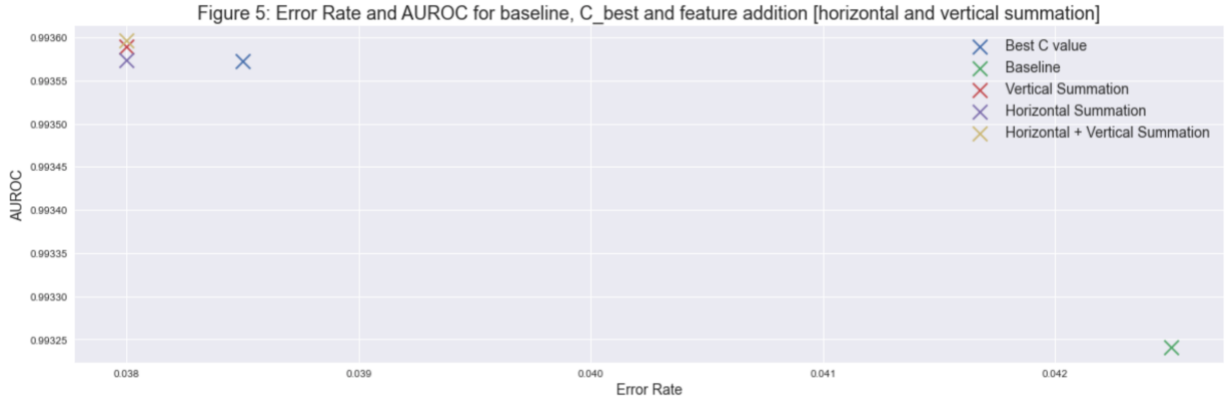
Apart from horizontal summation features, the same process will be completed for vertical summation (except this time, values for each image column is added instead of by row) giving another augmented x_train of size 12000×812. Last but not least, a combination of horizontal and vertical summation features will be added to result in another augmented x_train of size 12000×840.

The motivation for this feature transformation is because by observing images of sandals and sneakers obtained from x_train, sneakers generally have more rows with greater horizontal summation values compare to sandals, due to the fact that sandals generally does not cover as much skin. In addition, some sandals has heels which should render several columns with greater vertical summation values. Thus, horizontal and vertical summation features are explored independent and as a combination.

Leaderboard Result

Model Name	Error Rate	AUROC
Horizontal Summation	0.0380	0.993574
Vertical Summation	0.0380	0.993589
Horizontal+Vertical Summation	0.0380	0.993596

Table 4: Feature addition (horizontal and vertical summation) result



Discussion

As shown in figure 5 above, the error rate for all three feature additions explored are the same while AUROC values vary slightly, with horizontal summation's AUROC (0.993574) value being smaller than vertical summation's (0.993589) being smaller than the combination of horizontal-and-vertical summation's (0.993596).

Because all three feature additions models also adopts the C value of 0.3162 during construction and their error rates are smaller than best C value's displayed ($0.038 < 0.0385$), this suggests that this feature addition is effective in improving the classifier on top of changing the C value. But because the error rate has only been reduced by 0.0005, it is effective only to a small extent.

3.3 Data Transformation – Change values to 1 when above certain threshold

Method and Motivation

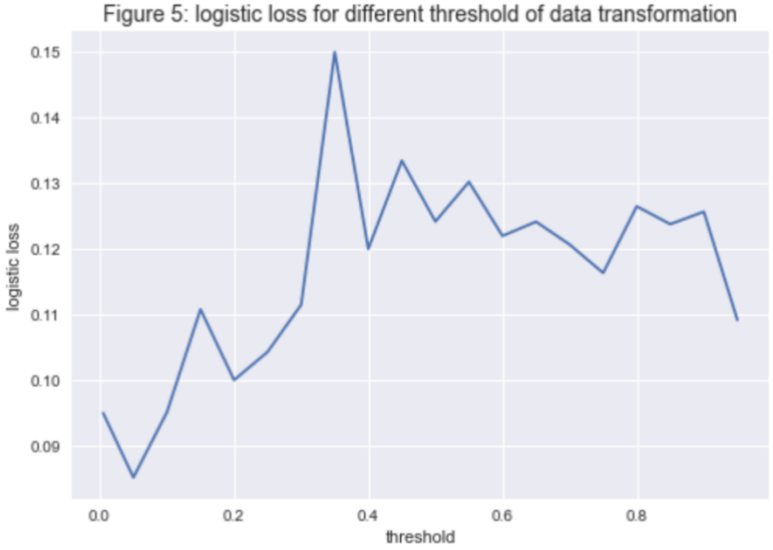
For this feature transformation, a series of threshold is explored to determine the best threshold above which the value will be set to 1, for example with the threshold of *0.5*, the following transformation occurs:

$$\begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.6 & 0.6 & 0.6 \\ 0.9 & 0.9 & 0.9 \end{bmatrix} \rightarrow \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The range threshold explored ranges from 0.05 to 0.95 inclusive with step size of 0.05. In addition, because after exploring these values 0.05 found to produce the lowest log loss, 0.005 is added to ensure that it is not the boundary value that gives the best log loss.

This transformation is applied because after visualizing several x_test and x_train images, I found that for some the contrast between bright and dark pixels is not obvious and in extreme cases, it is even hard for me to discern whether it is sandal or sneaker. Thus, by setting pixels satisfying criteria to one, I would like to explore ways of accentuating the contour by increasing the contrast of pixels in the image and letting relatively light pixels to become even whiter.

Best Threshold Exploration

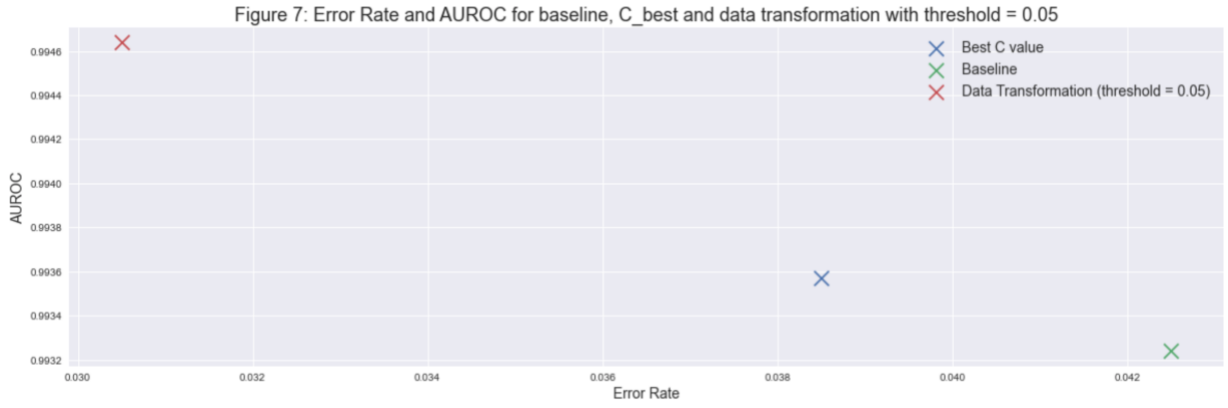


As seen from the figure above, threshold of 0.05 gives the best result (lowest log loss) and thus, this is used to produce the $predicted\ y\_test$.

Leaderboard Result

Model Name	**Error Rate**	**AUROC**
Data Transformation (threshold = 0.05)	0.0305000	0.994638

Table 5: Data transformation (threshold = 0.05) result



Discussion

The error rate presented for the current feature transformation in table 5 is the lowest among all three feature transformations tested in section 3 while the AUROC is also the highest. In addition, as shown in figure 7, its error rate is significantly lower than that of the best C-value, 0.0305 < 0.0385. This suggests that the combination of C value equals to 0.3162 and threshold equals to 0.05 can more effectively classify image, potentially due to the fact that dimmed images are brightened and contours become more discernable which together produces better result than other transformations explored.

4. Conclusion

4.1 Discussion -All leaderboard results

Figure 8: Error Rate and AUROC for all models tried

Figure 8 is a scatter plot comparing all feature transformation results discussed in Section 3 with baseline and best C value results in Section 2. There are only five markers available because for Table 4's feature transformation results, the error rates for horizontal, vertical and "horizontal + vertical" summation are the same and AUROC only varies by a miniscule amount. Therefore, due to overlap in scatterplot, only the dark blue marker is visible.

The model with the lowest error rate and highest AUROC is the best. If represented in the figure, it should be closer to the top left corner. Looking at the seven leaderboard submission records including baseline, best C value, data augmentation, horizontal summation, vertical summation, horizontal and vertical summation, and data transformation with threshold, data transformation with threshold equals 0.05 (section 3.3) together with the inverse penalty strength set to 0.3162 (found in section 2.2) gives the lowest error rate and highest AUROC. Thus, this is the most accurate model attained.

4.2 Evaluation and Further Investigation Topics

Potential areas of improvement include exploring different k values for k-fold cross validation and constructing each model several times and take average to avoid anomaly.

On the other hand, the pros of this project include exploring a range of C parameters and within the three feature transformations explored, two (3.2 feature addition and 3.3 data transformation with threshold) performs better than baseline, suggesting that these are along the right track and are potential aspects of further investigation such as using the idea of convolution to explore spatial patterns.