COMP135 Project 1 Writeup

March 2019

Darren Ting

1 Report for Question 1

1.1 Part A

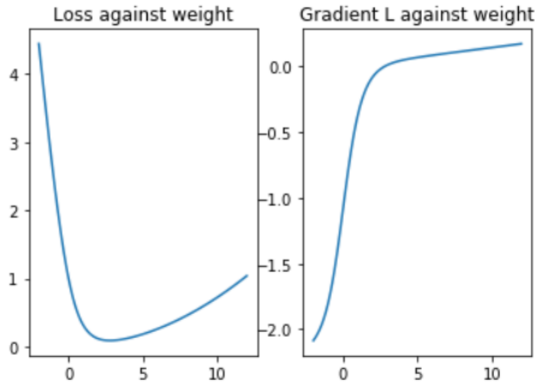


Figure 1: The loss trend makes sense because it appears to be convex, so we can use gradient descent on it to find the minimum. In the graph, the minimum is around 3, which ended up to be my weight in my regression. The Gradient looks just as expected because walking through the gradient means to find the point where the gradient=0, so where the minimum exists. Calculating the gradient seems to increase in that the change is less and less negative until it gets to 0. By inspecting this plot, w_1^* is approximately 2.78.

1.2 Part B

Learned Weights vs Iterations(x)

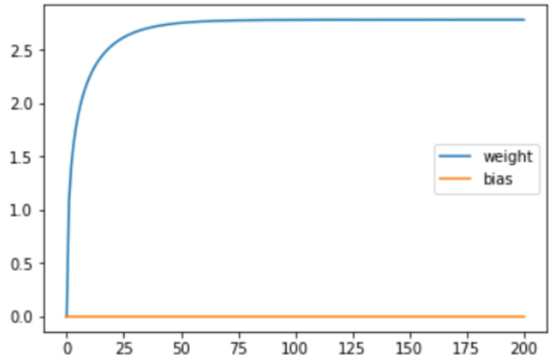
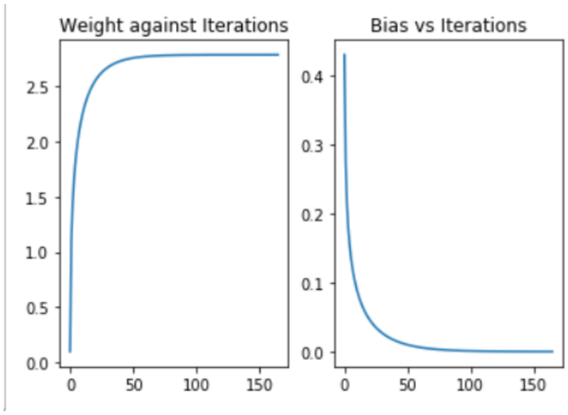
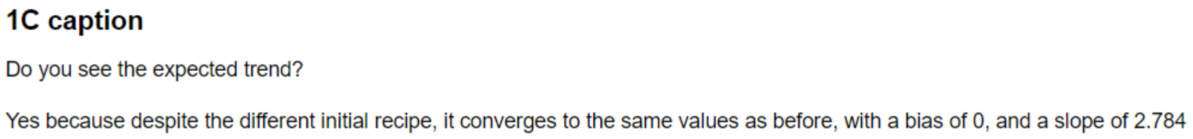


Figure 2: I do see my expected trend because my weight increased to fit the data well. The bias stays the same because it is not needed to increase the bias to fit the data given. Therefore, an increase in weight is necessary. This matches 1a because the minimum of loss was around 2.7 or 2.8, and the weight seemed to have converged to that point.

1.3 Part C





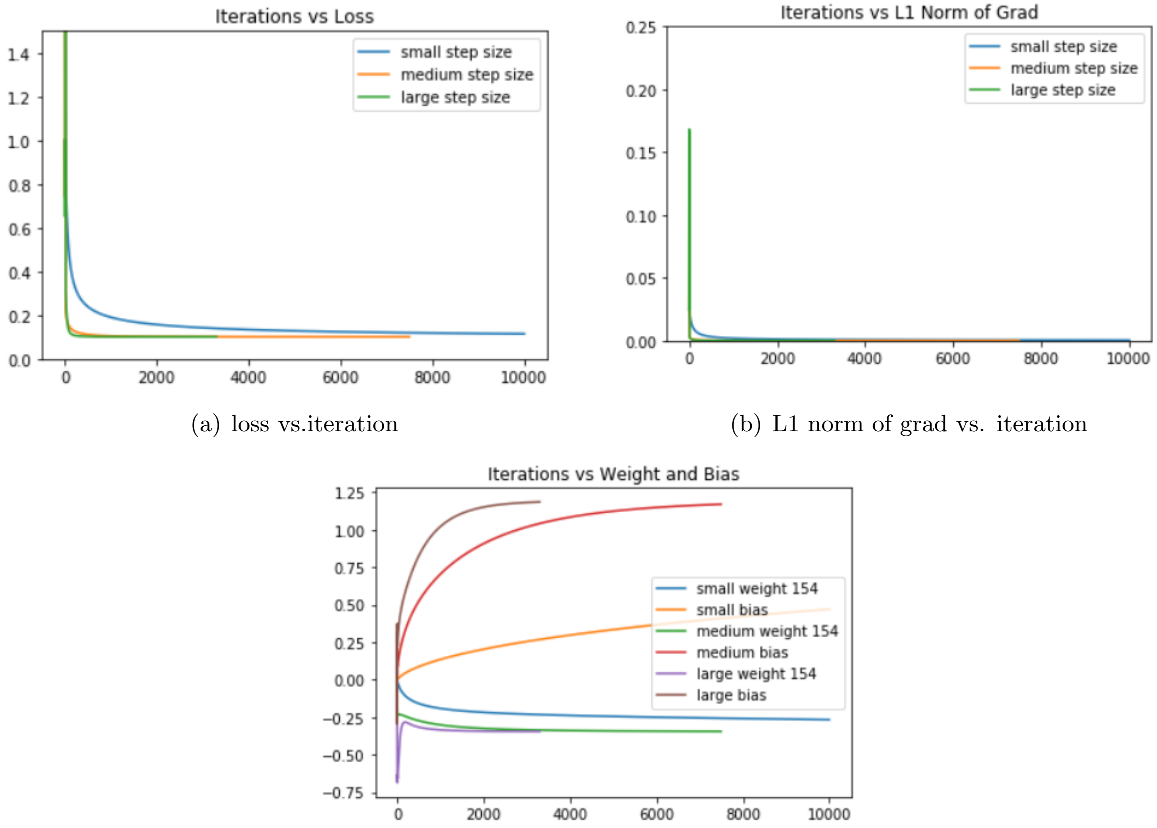
1C caption

Do you see the expected trend?

Yes because despite the different initial recipe, it converges to the same values as before, with a bias of 0, and a slope of 2.784

2 Report for Question 2

2.1 Part A



(a) loss vs.iteration

(b) L1 norm of grad vs. iteration

(c) specific weight vector 154 values (y-axis) vs. iteration

Figure 3: In terms of the Iteration and Loss plot, the loss decreases as there are more iterations. The medium and large step size lines end early because they converge in less than 10,000 steps. The small step size, which doesn't converge approaches the same value that the other two step sizes do. For small step size, I used step_size = 0.01. For medium step_size, I used 0.25, and for large step size, I used 0.78. I found that any number past 0.8 for step_size diverged for the alpha of 10. For the alpha of 10, I would recommend a step size of 0.5 because it would converge faster, but would not run as much of a risk of diverging as 0.78. To generalize, I would recommend a medium step size for any alpha, in that it would not converge too early and have a high error rate or too late and potentially not converge at all.

2.2 Part B

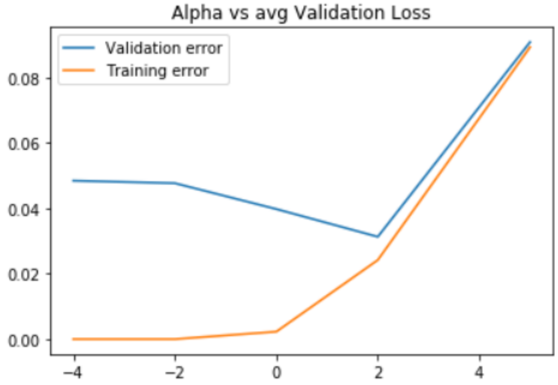
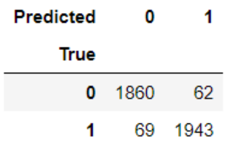


Figure 4: There appears to be overfitting when alpha is low, which makes sense because the penalty value alpha is low. My step size selection strategy was to start at a small step size for larger alphas, and then keep lowering it until it converged, or keep increasing it until it diverges, and then step back. The following are the alpha values with their corresponding step sizes:
$\alpha = 0.0001$, step_size = 1.06
$\alpha = .01$, step_size = 1.0625.
$\alpha = 1$, step_size = 0.5.
$\alpha = 100$, step_size = 0.25.
$\alpha = 10000$, step_size = 0.005.
I think the general trend I saw was that as alpha increases, the step size decreases, so they are inversely proportional. Based on the data regarding the validation and training loss, an $\alpha = 100$ is recommended because anything less than that appears to overfit (validation error is very high and training error is very low), and anything past it seems to underfit (both errors are large).

2.3 Part C

Confusion Matrix for Logistic Regression



Predicted	0	1
True		
0	1860	62
1	69	1943

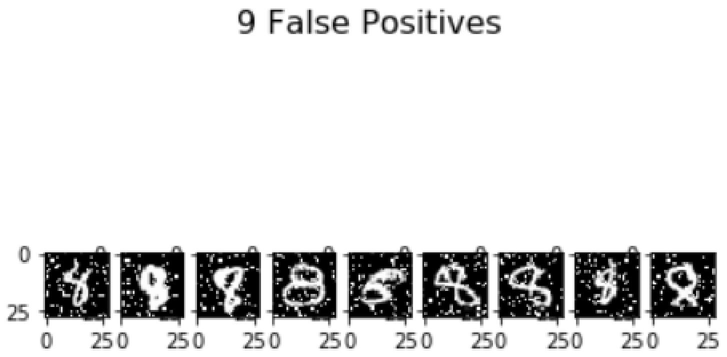


Figure 5: I think these plots were labeled as 9s because it seems to focus on the upper loop as well as the line descending on the right. I think that if there is a line right below the loop, it looks somewhat like a 9, and therefore wil classify it as a 9 even though a bottom loop exists. Therefore, I think it is placing emphasis on certain areas of the picture, while other areas may need higher weights.

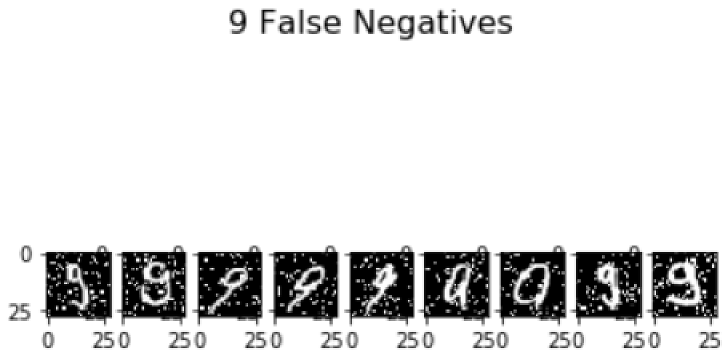


Figure 6: The mistakes my regression seems to make in terms of false negatives is the presence of some sort of either some sort of loop or the presence of a diagonal line on the bottom of the digit. In a normal 9, the bottom does not descend in a diagonal fashion, so my classifier seems to emphasize this in classifying whether or not a digit is a 8 or 9. Potentially, it may classify pictures as 8 based on curves present on the bottom half of the picture.

2.4 Part D

Color Map for Weights

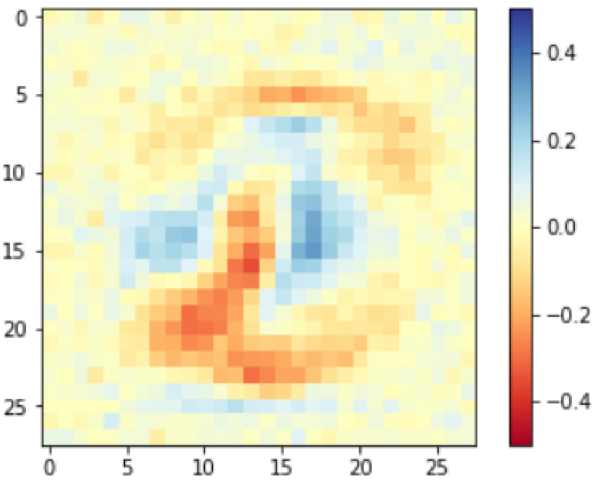
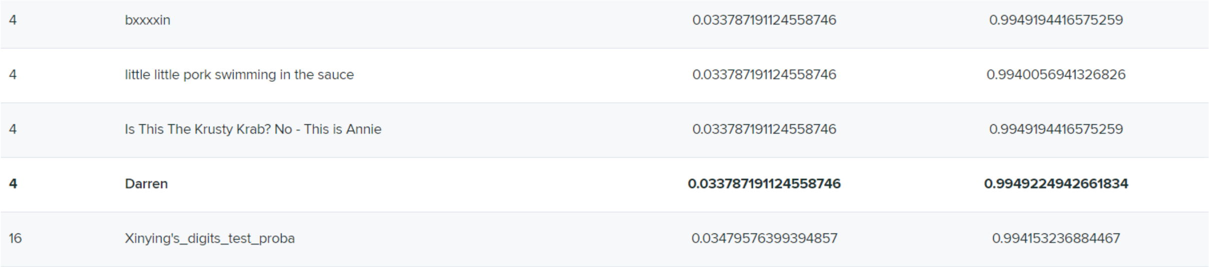


Figure 7: I noticed that the center is heavily concentrated in deciding whether or not the digit is an 8 or 9. According to the color map, it appears that curves present on the bottom of the picture will most likely classify a picture as an 8. What seems to have the largest weight in the picture seems to be a vertical line (denoted by a dark blue) present to the right of the center of the photo, which makes some sense because a 9 versus an 8 seems to have a vertical line descending straight down right below the top loop, while the 8 does not. TO clarify, blue means 9, red means 8.

2.5 Part E

Digits 8 vs 9 Heldout Test Evaluation with Leaderboard



4	bxxxxin	0.033787191124558746	0.9949194416575259
4	little little pork swimming in the sauce	0.033787191124558746	0.9940056941326826
4	Is This The Krusty Krab? No - This is Annie	0.033787191124558746	0.9949194416575259
4	**Darren**	**0.033787191124558746**	**0.9949224942661834**
16	Xinying's_digits_test_proba	0.03479576399394857	0.994153236884467

Figure 8: I decided to use $\alpha = 100$ with a step size of 0.25. It converged after 1223 iterations. My training set for k folds error rate is 0.0333, while my error rate in the test set is 0.0337 which is slightly higher. My test set results were:
error rate: 0.033787191124558746
AUROC: 0.9949224942661834

3 Report for Problem 3

For this problem, I used sklearn's Logistic Regression and LRGradientDescentWithFeatureTransform.py file provided to me.

3.1 Alpha Selection

When choosing an alpha for my logistic regression, I decided to use an equation that Professor Hughes used in class in that $1/2C = \alpha$. With C = 0.1, $\alpha = 5$, which means that overfitting is punished through means of the L2 penalty.

3.2 Feature Transformations

In addition to the features already provided in the starter code (squared_x and avg_x), I implemented multiple features; however, he features that I implemented do not reflect the ones I eventually used. I divided these features into the two following sections. My process for figuring out what features to use was to look at several examples of sandals and sneakers in the MNIST set and find similarities.

What Worked:
The feature that had a positive influence on the error rate was one that rounds up all nonblack colors to 1. As a result, all black areas in the picture stay black, but all colors become white. As a result, the holes in the shoe make it much more obvious to spot for the regression. This transformation is seen below:

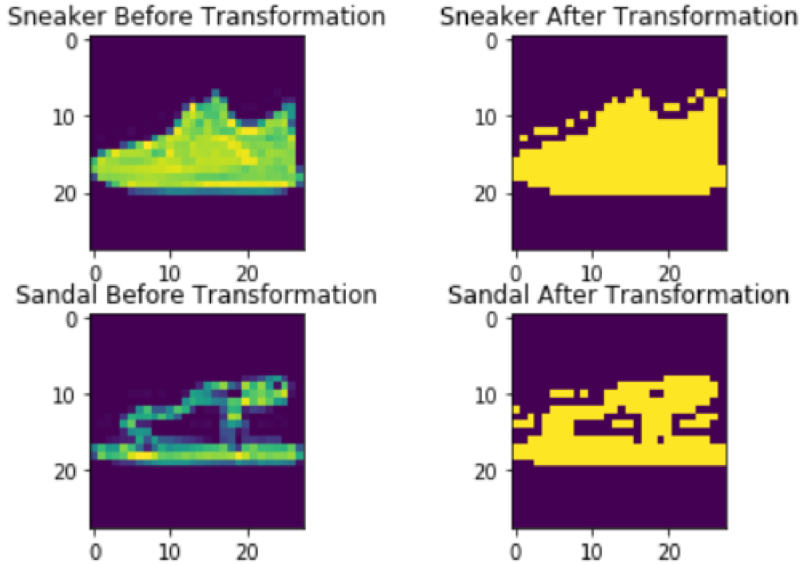
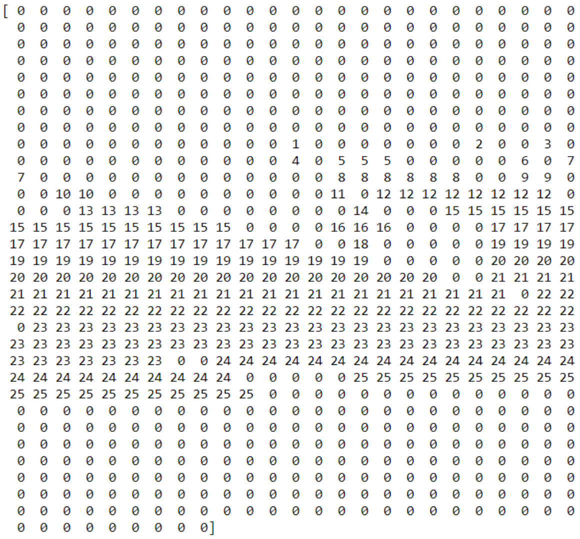


Figure 9: All the colored pixels are rounded up to 1

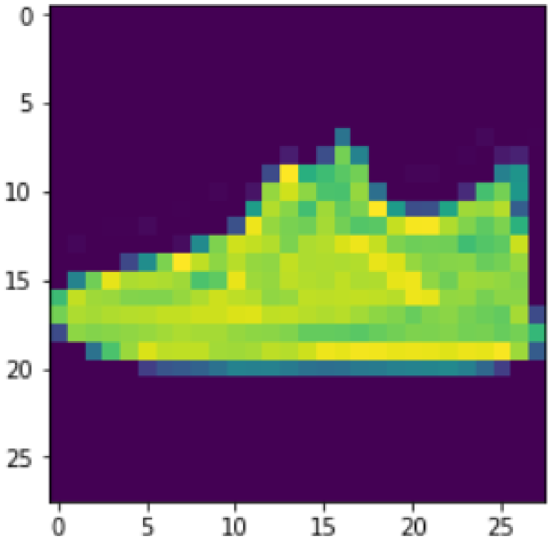
What Didn't Work
I tried to use three additional features: One that counted the number of clusters, one that counted

the number of light pixels, and one that transformed the pixels into label.measure.
To count the number of light pixels, I simply called np.count_nonzero and appended that number to the end of each feature array. My thinking behind this was that there may be more white pixels in pictures of sneakers because there are less holes in there than there would be in a sandal.
For the feature that counted the number of clusters, and one that transformed it into label.measure, I used scipy.ndimage's label.measurements, which returns an array that IDs each cluster of nonzero numbers and a number of clusters. An input picture and its corresponding array is shown below:



```
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  2  0  0  3  0
  0  0  0  0  0  0  0  0  0  0  0  0  4  0  5  5  5  0  0  0  0  0  6  0  7
  7  0  0  0  0  0  0  0  0  0  0  0  0  0  8  8  8  8  8  8  0  0  9  9  0
  0  0 10 10  0  0  0  0  0  0  0  0  0  0 11  0 12 12 12 12 12 12 12 12  0
  0  0  0 13 13 13 13  0  0  0  0  0  0  0  0 14  0  0  0 15 15 15 15 15 15
 15 15 15 15 15 15 15 15 15 15  0  0  0  0 16 16 16  0  0  0  0 17 17 17 17
 17 17 17 17 17 17 17 17 17 17 17 17 17  0  0 18  0  0  0  0  0 19 19 19 19
 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19  0  0  0  0  0 20 20 20 20
 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  0  0 21 21 21 21
 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21  0 22 22
 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22
  0 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
 23 23 23 23 23 23 23  0  0 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
 24 24 24 24 24 24 24 24 24 24  0  0  0  0  0 25 25 25 25 25 25 25 25 25 25
 25 25 25 25 25 25 25 25 25 25 25  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0]
```

(a) label.measurements(x_NF[500])



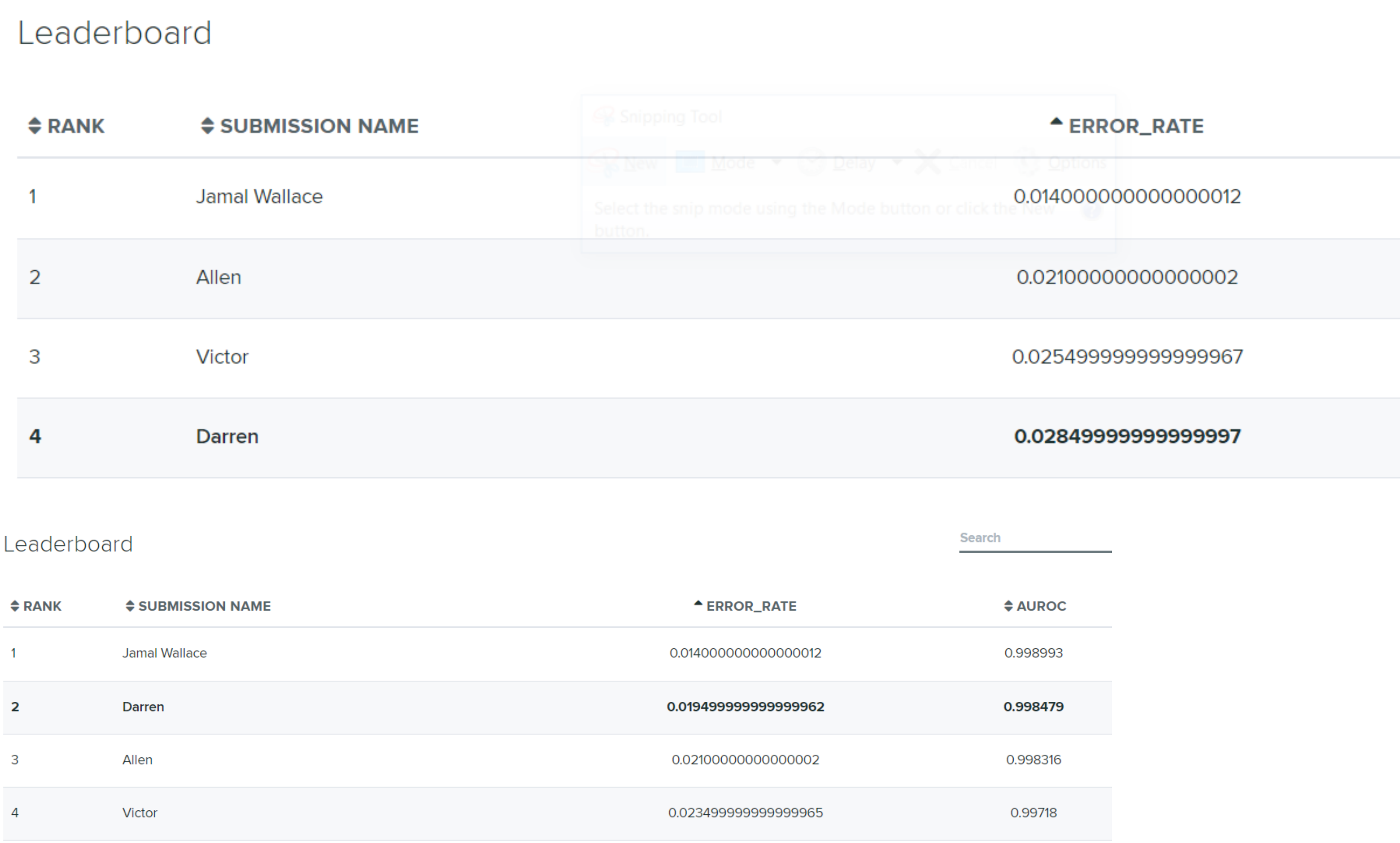
(b) x_NF[500]

My thought process behind this was that by showing the different clusters with different numbers, the regression could more easily classify because of the different nonzero numbers. I also appended the total number of clusters at the end of each feature array, which is the 2nd return value of label.measurements. My thought process behind this was that maybe the larger number of clusters, the more likely it is a sandal because of the different gaps/holes in the shoes. These three features, combined with the one that worked yielded this result on the leaderboard:

I think that these transformations made it worse because it would train it based on the larger weights from the cluster numbers, which do not correlate at all with differentiating between sandal and sneaker.
Therefore, the results only use the feature transform that rounds up all nonzero numbers to 1.

3.3 Results

ROC curve contrasting the normal Logistic Regression with the one with feature transformations



Select the snip mode using the Mode button or click the New button.

Leaderboard

⬍ RANK	⬍ SUBMISSION NAME	▲ ERROR_RATE
1	Jamal Wallace	0.014000000000000012
2	Allen	0.021000000000000002
3	Victor	0.025499999999999967
4	**Darren**	**0.028499999999999997**

Leaderboard

Search

⬍ RANK	⬍ SUBMISSION NAME	▲ ERROR_RATE	⬍ AUROC
1	Jamal Wallace	0.014000000000000012	0.998993
2	**Darren**	**0.019499999999999962**	**0.998479**
3	Allen	0.021000000000000002	0.998316
4	Victor	0.023499999999999965	0.99718

Figure 10: Sneakers vs Sandals Heldout Test Evaluation

Confusion Matrices

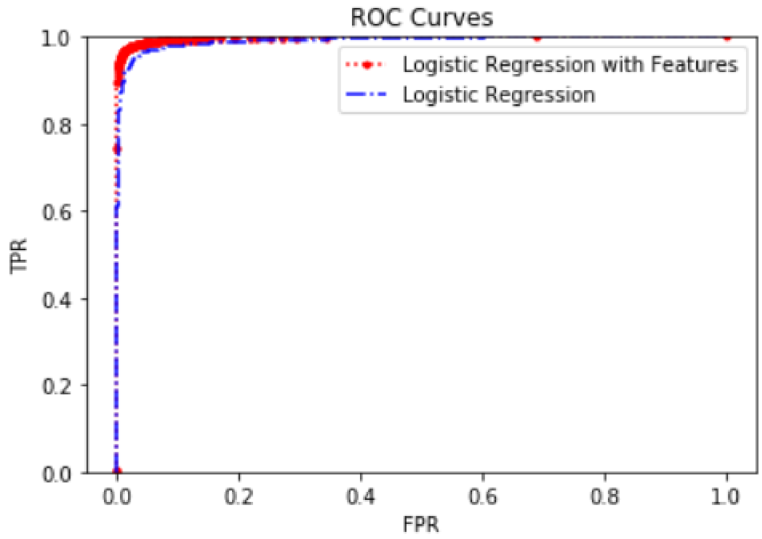
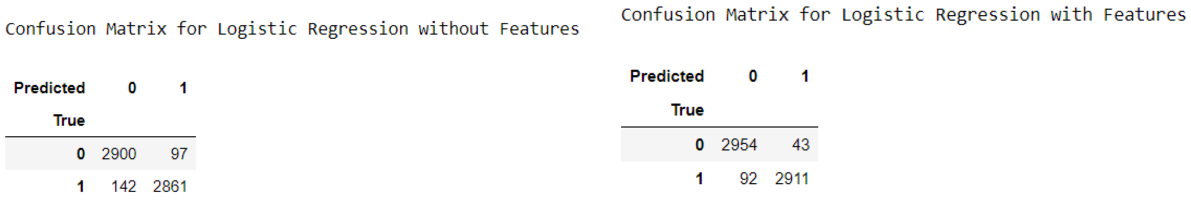


Figure 11: The improvement is relatively minimal in the scope of this graph, but the feature of rounding up nonzero pixels improved performance



```
Confusion Matrix for Logistic Regression without Features
```

Predicted	0	1
True		
0	2900	97
1	142	2861

(a) Without Feature Transformations

```
Confusion Matrix for Logistic Regression with Features
```

Predicted	0	1
True		
0	2954	43
1	92	2911

(b) With Feature Transformations

Figure 12: Based on these matrices, the feature transformation decreases both false positives (by 54 examples) and false negatives (by 50 examples). The regressions were trained with half of the training set, and then validated/tested with the other half.

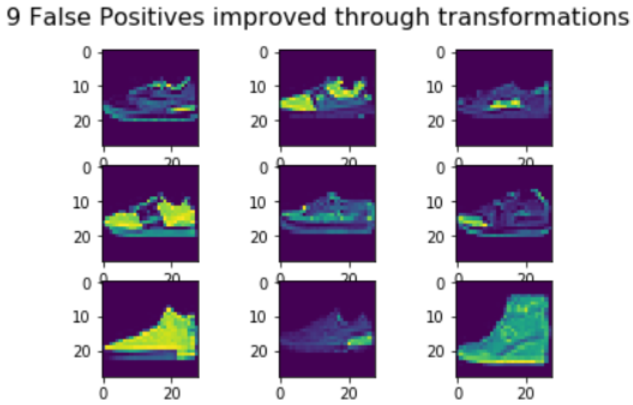


Figure 13: These pictures were classified as sandals by the normal Logistic Regression, but are corrected by the feature transformation. I think it is because the dark areas in the shoes become bright, so the lack of holes are apparent.

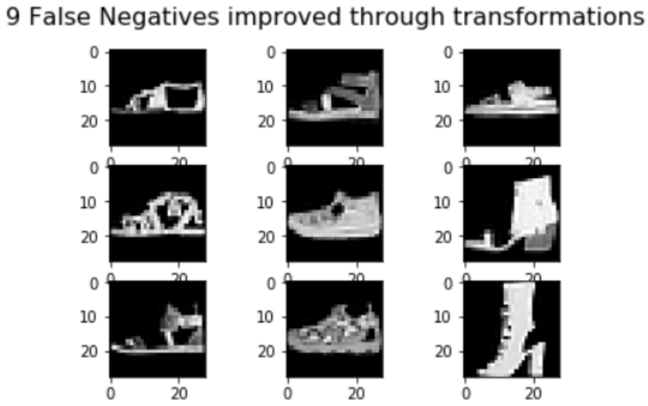
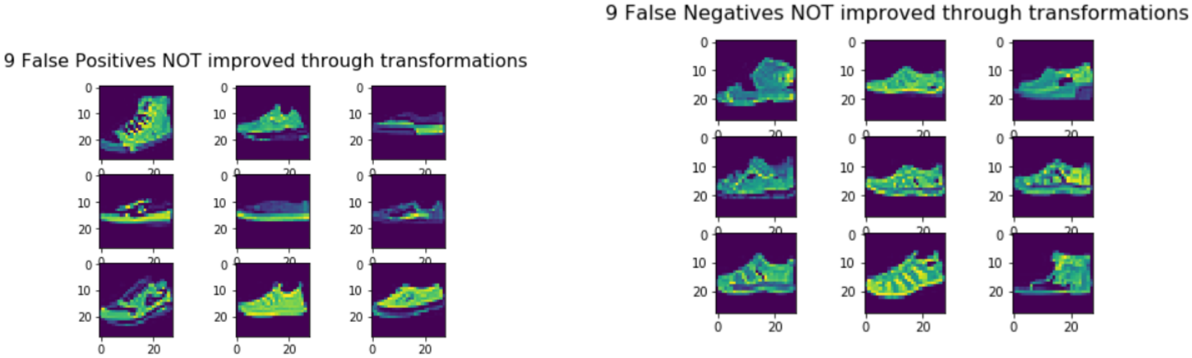


Figure 14: These pictures were classified as sneakers by the normal Logistic Regression, but are corrected by the feature transformation. I think it is because the holes become more obvious, so they are now classified as sandals.



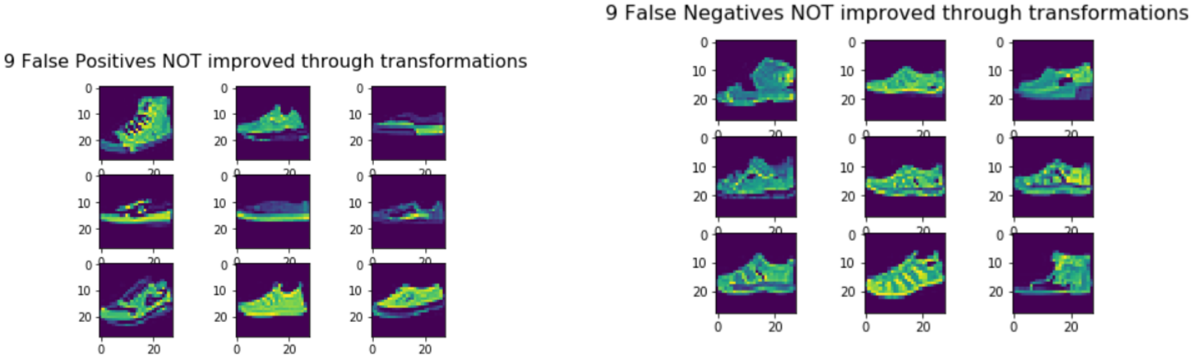
9 False Positives NOT improved through transformations

9 False Negatives NOT improved through transformations

(a) Examples of pictures that remained as False Positives

(b) Examples of pictures that remained as False Negatives

Figure 15: Some of the ones that remained false positives could potentially be due to other factors that the Logistic Regression looks into that are not the holes in the shoe, or that there are black areas on the shoe, which would appear to be holes to the transformation. Some of the false negatives can be explained by the fact that the holes in the shoe are covered by the other side of the shoe, so it would not look like a hole to the regression.



9 False Positives NOT improved through transformations

9 False Negatives NOT improved through transformations

(a) Examples of pictures that remained as False Positives

(b) Examples of pictures that remained as False Negatives

Figure 16: Some of the ones that remained false positives could potentially be due to other factors that the Logistic Regression looks into that are not the holes in the shoe, or that there are black areas on the shoe, which would appear to be holes to the transformation. Some of the false negatives can be explained by the fact that the holes in the shoe are covered by the other side of the shoe, so it would not look like a hole to the regression.