# Machine Learning Project 1 Writeup

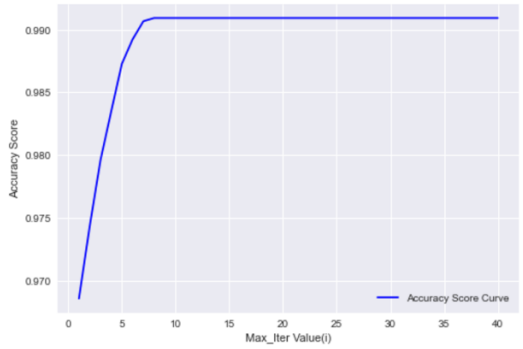
Nick Murphy

4 April 2021

## Part One: Logistic Regression for Digit Classification

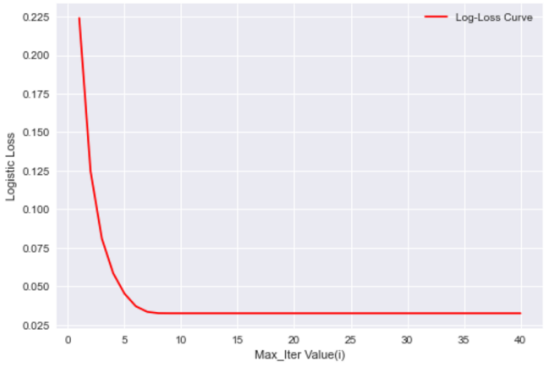
### 1.1: Accuracy and Log-Loss of Logistic Regression with various max iteration values

Accuracy Based on Number of Iterations




```
Max Accuracy Score:  0.9909322033898305
Max_Iter of max accuracy score:  7
```

Log-Loss Based on Number of Iterations

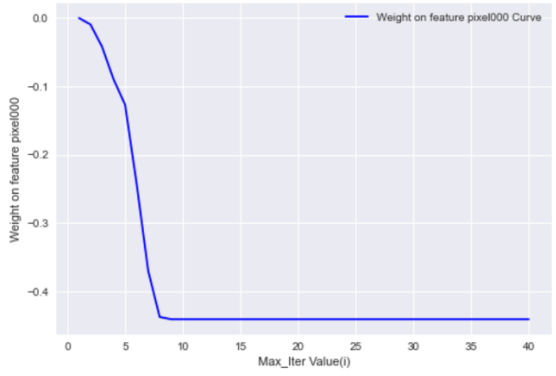



```
Min Logistic Loss:  0.03245505358104707
Max_Iter of min logistic loss:  8
```

**Discussion of Part 1.1:**

These results show that as the maximum number of iterations increases, accuracy increases while log-loss decreases. Accuracy reaches a maximum at 7 iterations, while log-loss reaches a minimum at 8 iterations. Both become flat after these iterations. This tells us that while increasing the maximum number of iterations helps improve accuracy and log-loss, the utility increases at a decreasing rate after 7/8 iterations.

### 1.2: Plotting max iteration values against the weights placed on feature pixel000




```
Min Weight on feature pixel000:  -0.4406741022772648
Max_Iter of Min Weight on feature pixel000:  8
```

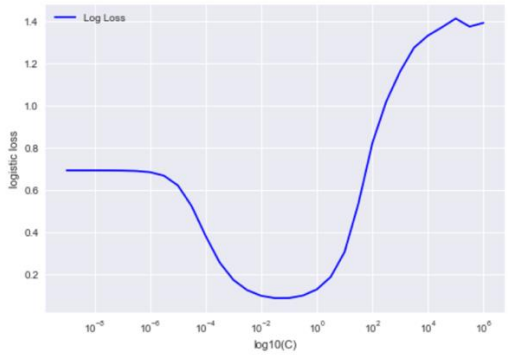
These results show us that the minimum weight on feature pixel000 occurs when the number of iterations is set to 8, with a weight of -0.44067. Weights decrease as the number of iterations increases, reaching a minimum at 8 and flattening out afterwards. Because the curve flattens out, we can tell these features dominate the model.

# Machine Learning Project 1 Writeup

Nick Murphy

4 April 2021

**Part 1.3: Exploring Different Values of Inverse Penalty**



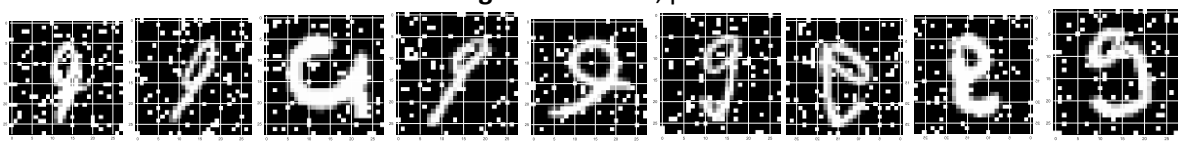

```
Best C-value for LR with 784-feature data: 0.0316228
Validation set log-loss at best C-value: 0.0897
Accuracy of model using best C-value: 0.9672214
```

```
Best C-value: 0.03162277660168379
Predicted     0     1
True
0           942    32
1            33   976
```

The best C value for a Logistic Regression on this data is 0.0316, which provides an accuracy of 0.97 and a log-loss of 0.0897, which are both strong scores. We can see from the confusion matrix that this model correctly classified a vast majority of the data, with 1918 correct classifications out of 1983.

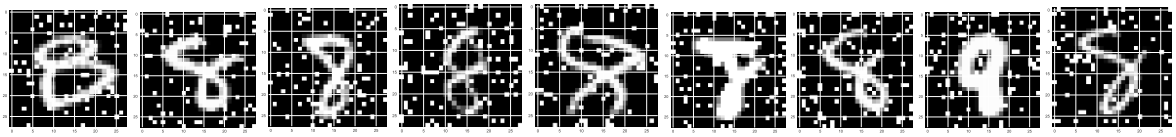
**Part 1.4: Analyzing Mistakes of the Best Model**

**False Negatives**: True 9, predicted 8



From this set of false negatives we can see that this classifier mistook a nine for an eight when that nine had a curved up tail or when it was slanted. This could be because of the extra noise found between the curl and the head of the number that make it look slightly more like an eight.

**False Positives**: True 8, Predicted 9



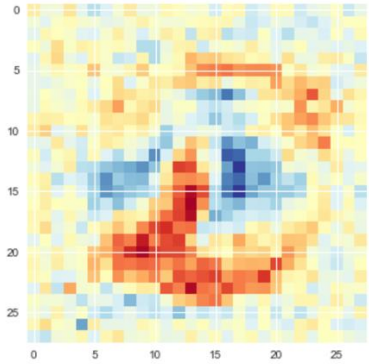
On the other hand, it mistook an eight for a nine when that eight had a broken top right curve and when the bottom tail was skinny. For the numbers with the broken top right, it is possible that the classifier mistook the two disconnected tails as the tail of an upside down nine.

# Machine Learning Project 1 Writeup

Nick Murphy

4 April 2021

**Part 1.5: Analysis of Weight Coefficients**



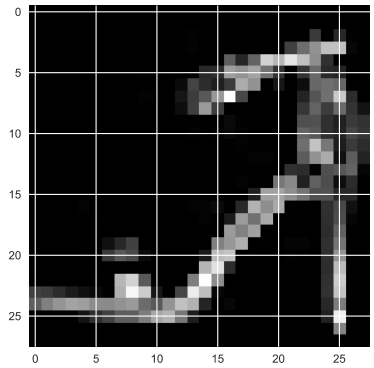
In this plot, the darker colors occur when a heavier weight is placed on a pixel. A darker red occurs on a heavier negative weight, while a darker blue occurs on a heavier positive weight. From these results we can see that this model places heavier weights on pixels that occur near the center of the image, as well as in the bottom curl of an eight or nine. This backs the idea that errors are occurring due to extra noise around the bottom of the number.

## Part Two: Sneakers versus Sandals

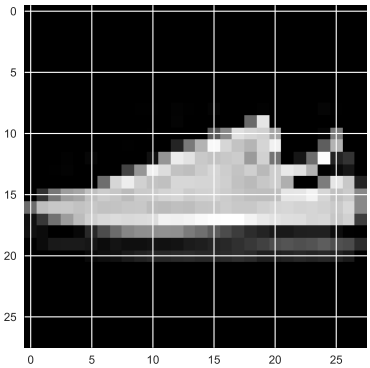
## 2.1 Inspecting the Dataset

The data from the .csv files was read in and the first header line is eliminated. We can transform the image data into a 28 x 28 matrix to allow us to use the imshow() function to display the image. We know that an output value of 0 represents a sneaker and an output value of 1 represents a sandal.

Sandal



Sneaker



# Machine Learning Project 1 Writeup

Nick Murphy

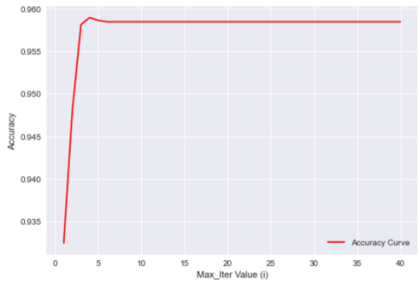
4 April 2021

## 2.2 Prelude: Metrics to Compare Models

To compare models, I wrote a function that prints out the plots below as well as the maximum accuracy and minimum log-loss of the model. This was crucial, as the best way to compare models is with a static metric/system.

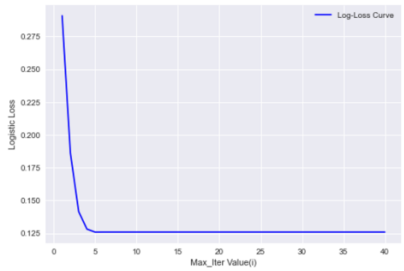
## 2.2: Basic Model and Logistic Regression on Initial Data

Accuracy Based on Number of Iterations




```
Max Accuracy Score:  0.959
Max_Iter of max accuracy score:  3
```

Log-Loss Based on Number of Iterations

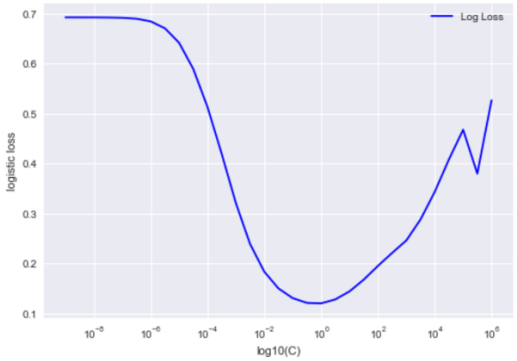



```
Min Logistic Loss:  0.12585320051406795
Max_Iter of min logistic loss:  4
```

The first model I built was a basic logistic regression on the training data with no added features. This was done to create a baseline prediction accuracy for the rest of the process of building of the model. We can see below that the highest accuracy according to number of iterations was 95.9%, with a log-loss of 0.126.

## 2.3: Finding the Best Penalty Strength

By using the same logspace function as done in **1.3**, we can see that the baseline models produces the highest accuracy when C is 1. This is a good value to have a baseline for future research into feature transformations.




```
Best C-value for LR with 784-feature data: 1.0000000
Validation set log-loss at best C-value: 0.1199
Accuracy of model using best C-value: 0.9558333
```

## 2.4: Adding Polynomial Features

The first step I took to improve this model was to try increasing the degree of the features and testing with cross-validation using 5 folds. This drastically hurt the performance and runtime of the model, as only increasing to a degree of 2 made the runtime more than twenty times longer and decreased accuracy by almost 20 percent.

# Machine Learning Project 1 Writeup

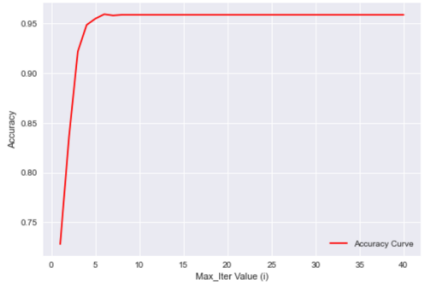
Nick Murphy

4 April 2021

## 2.5: More Feature Transformations

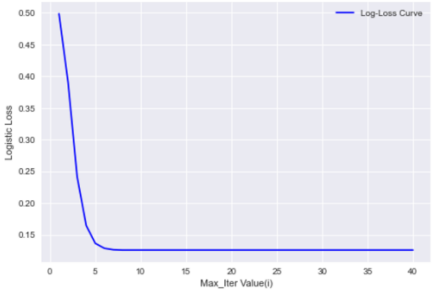
### 2.5.1: Adding Features that Count Overall Numbers of Black Pixels:

Accuracy Based on Number of Iterations




```
Max Accuracy Score:  0.95925
Max_Iter of max accuracy score:  5
```

Log-Loss Based on Number of Iterations



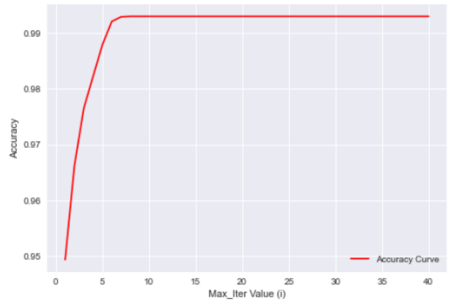

```
Min Logistic Loss:  0.12584049854797252
Max_Iter of min logistic loss:  7
```

As suggested in the project specification, I added a feature that maintained the number of black pixels in the image. To measure the performance of this model, I compared its accuracy and log-loss to the baseline model shown in **2.2**. The performance of this model is displayed below. From these results we can see that classifier performance improved but only slightly, increasing accuracy from 95.9% to 95.925% and decreasing log-loss from 0.12585 to 0.12584.

### 2.5.2: Adding Features that Consider Spatial Data

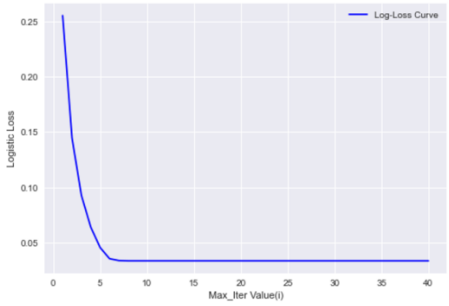
Upon initially inspecting the images, I wondered if patterns in the pixel data could be identified and used to make better predictions. I added features that tracked the relationship between a pixel and its adjacent pixels: which adjacent pixels share the same rounded grayscale value, as well as the sum of the grayscale values of each pixel's adjacent pixels. This drastically improved the performance of my classifier, decreasing error to a rate of 2.4%, increasing accuracy to 99.3% and decreasing log-loss to 0.0336.

Accuracy Based on Number of Iterations




```
Max Accuracy Score:  0.993
Max_Iter of max accuracy score:  7
```

Log-Loss Based on Number of Iterations




```
Min Logistic Loss:  0.03356074936915654
Max_Iter of min logistic loss:  7
```

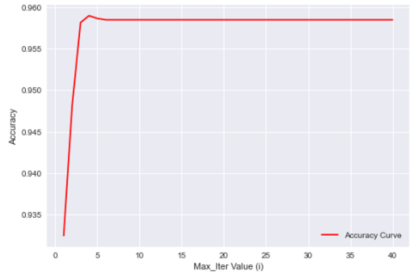
# Machine Learning Project 1 Writeup

Nick Murphy

4 April 2021

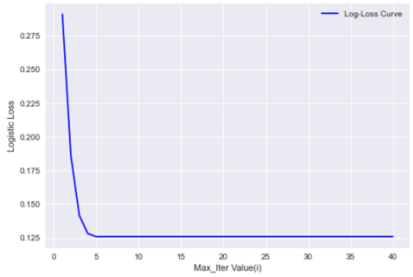
## 2.5.3: Flipping Data Horizontally

Accuracy Based on Number of Iterations




```
Max Accuracy Score:  0.959
Max_Iter of max accuracy score:  3
```

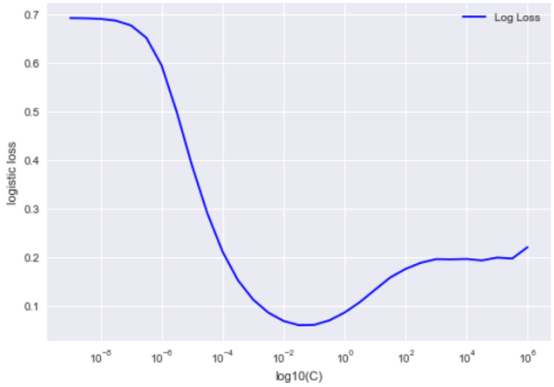
Log-Loss Based on Number of Iterations




```
Min Logistic Loss:  0.12585320053937638
Max_Iter of min logistic loss:  4
```

As suggested in the project specification, I tried to augment the data such that the image was flipped horizontally. According to the accuracy and log-loss metrics, this data augmentation made literally no change to the performance of my classifier.

## 2.6: Combining Methods of Improving Performance




```
Best C-value for LR with 784-feature data: 0.0316228
Validation set log-loss at best C-value: 0.0610
Accuracy of model using best C-value: 0.9779167
```

After individually testing the above feature transformations, I wanted to see what would happen when I added multiple features to the data. I mixed and matched the above methods, but no combination improved performance above the baseline model more than the model that solely considered adjacent pixels. In fact, some even harmed performance. Hence, I tried to improve this 'best' model as much as I could by finding the best C-value for this specific model. That ended up being 0.0316, shown above, and improved performance slightly: error rate dropped to 2% (from 2.4%) and AUROC increased to .998 (from 0.997).

# Machine Learning Project 1 Writeup

Nick Murphy

4 April 2021

## 2.7: Conclusion

| | | | |
|---|---|---|---|
| 2 | The Least Accurate | 0.020499999999999963 | 0.99804 |

Unfortunately, my progress was drastically hindered by the hardware of my computer. I thought that my 2-year-old MacBook Pro would be able to handle this project, but I was proven wrong. If I had a better computer, I would have continued my attempt to implement a SVM to aid this classifier, but alas my computer crashed every time I tried to run my SVM program.

I stand by my choice of metric to compare models, as this is what led me to have this very low error rate and high AUROC. If I had more time, I would try to go through every false classification in order to figure out how my classifier is making mistakes and fix the source of this error. Regardless, I'm proud I was able to reduce my error rate to 2.05%.

I found it very interesting that adding multiple new features didn't improve performance. While it significantly hurt runtime, I thought that adding multiple features would improve accuracy, but in some cases, it made my classifier less accurate. I hoped that combining flipping the data horizontally and another one of my feature transformations would drastically improve performance but including the horizontal flip hurt performance on both occasions.

My model that utilized spatial data and a penalty strength of 0.0316 was by far the most accurate, with an error rate of 2.05% and an area under ROC curve of .998. From performing these feature transformations taught me that augmenting the data in ways that provide context to each feature leads to better classifier performance. This is shown in the results from adding a feature that has the sum of all the black pixels and adding a feature that considers adjacent pixels.