Bryan Pritsker

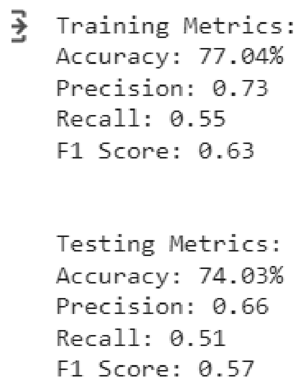
2/23/2024

Machine Learning Homework 1 Report

Part 1: (Accuracy Scores)

Accuracy of Logistic Regression Model:

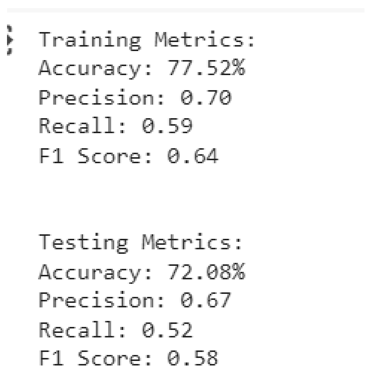
This is the accuracy of the training set and testing set for the Logistic Regression Model. The gradient descent method within the Logistic Regression Model was performed over 70,000 iterations (epoch = 70000) and learning rate = 0.005. The training and testing data were split with an 80/20 ratio.



```
Training Metrics:
Accuracy: 77.04%
Precision: 0.73
Recall: 0.55
F1 Score: 0.63


Testing Metrics:
Accuracy: 74.03%
Precision: 0.66
Recall: 0.51
F1 Score: 0.57
```

Accuracy of Neural Network:



```
Training Metrics:
Accuracy: 77.52%
Precision: 0.70
Recall: 0.59
F1 Score: 0.64


Testing Metrics:
Accuracy: 72.08%
Precision: 0.67
Recall: 0.52
F1 Score: 0.58
```

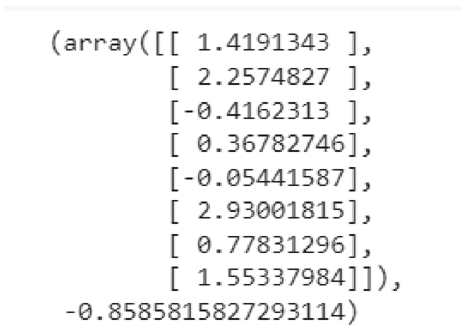
This is the accuracy of the training set and testing set for the Neural Network Model. Forward Propagation and Backward propagation in the Neural Network was performed over 70,000 iterations (epoch = 70000) and learning rate = 0.005. The training and testing data were split with an 80/20 ratio.

Part 2: (Architecture and Weights)

In both the Logistic Regression Model and the Neural Network, the input data was normalized using "normal distribution scaling".

The **Logistic Regression** model was initialized with 8 weights (since the number of features in the dataset was 8) and one bias (we set the initial bias = 0). The weights and bias were iteratively updated using gradient descent and the logistic regression cost function: log-loss cost function. Gradient descent iteratively adjusted the weights and bias to minimize the log-loss cost function.

Logistic Regression final weights and bias:



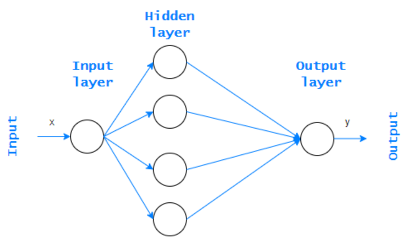
```
(array([[ 1.4191343 ],
        [ 2.2574827 ],
        [-0.4162313 ],
        [ 0.36782746],
        [-0.05441587],
        [ 2.93001815],
        [ 0.77831296],
        [ 1.55337984]]),
 -0.8585815827293114)
```

"The weights are the 2d array, the bias is the number on the bottom."

The **Neural Network** model (very similar to the picture shown below) contained an input layer, one hidden layer, and an output layer. Both the input and output layer contained one neuron respectively, and the single hidden layer contained 5 neurons (as opposed to 4 like in the picture below). The activation function that was used throughout the neural network was the sigmoid activation function, and the cost function was the Log-Loss Cost function. The sigmoid activation function was used twice; first to transform the linear combination from the input layer being sent to the hidden layer, and secondly to transform the linear combination from the hidden layer being sent to the output layer. Each neuron in the hidden layer had its own unique set of weights and biases. The weights and bias were initialized with random values between (-1, 1), and they got iteratively updated through back propagation.

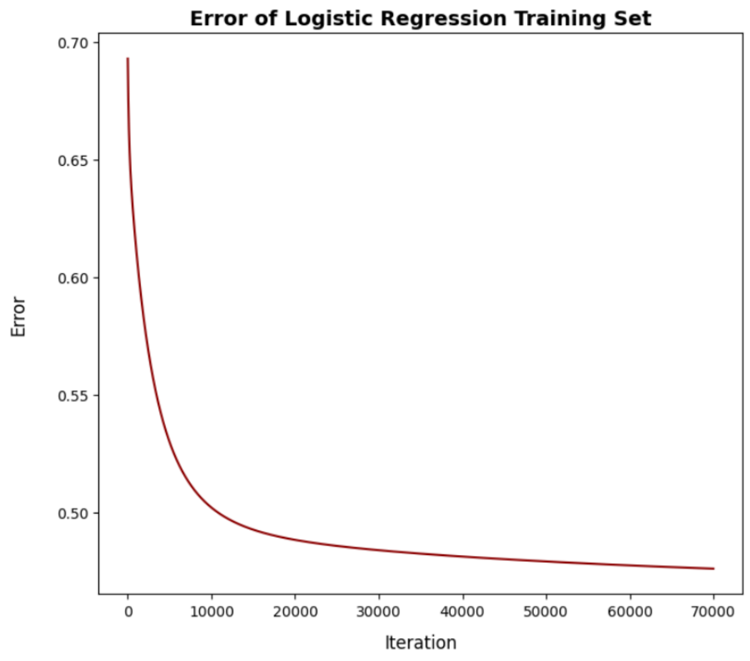
The weights being applied in the input layer had a shape of (8, 5) and the bias was an array of length 5. Therefore, there were 5 unique sets of weights and 5 unique biases in this step.

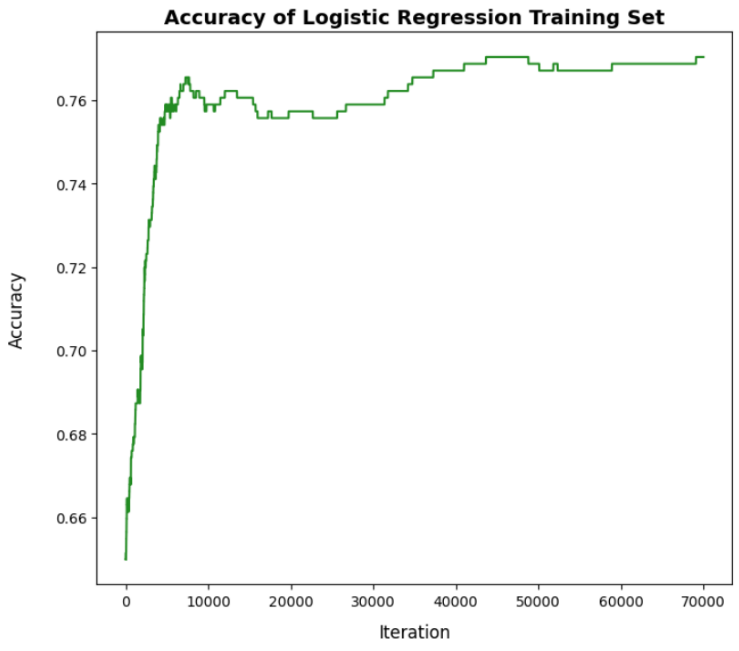
The weights being applied in the hidden layer had a shape of (5, 1) and the bias was an array of length 1. Therefore, there was 1 unique set of weights and 1 unique bias in this step.



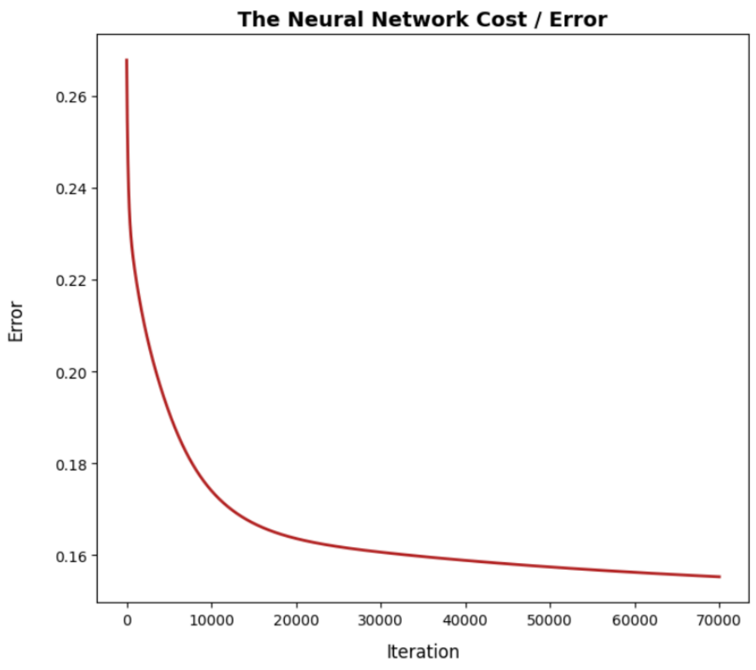
Part 3: (Graphs of Training Error and Accuracy)

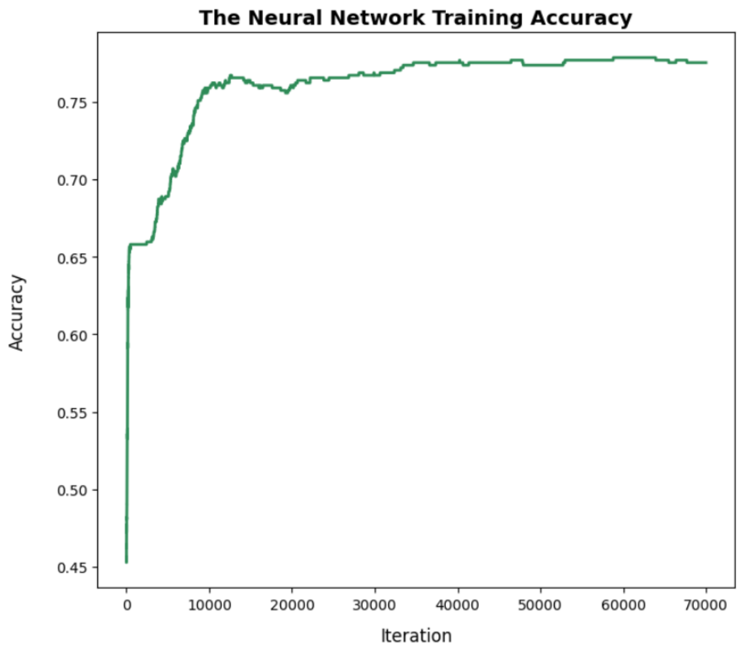
Logistic Regression Plot of Error and Accuracy:





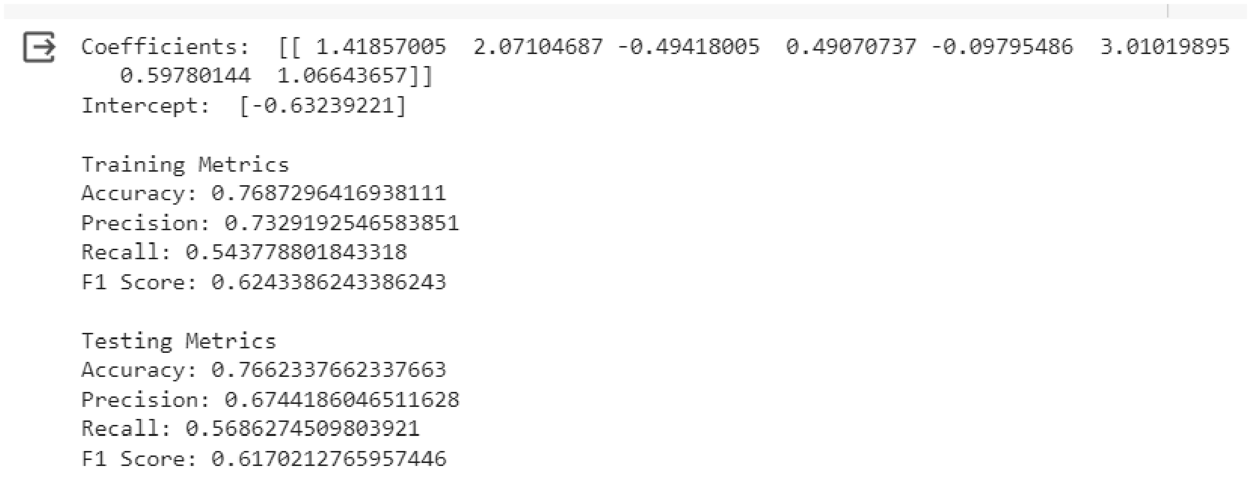
Neural Network Plot of Error and Accuracy:





Part 4: Scikit Learn

Scikit Learn Logistic Regression Results:



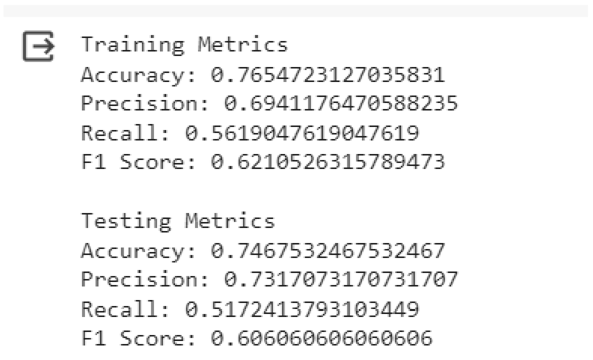
```
Coefficients:  [[ 1.41857005  2.07104687 -0.49418005  0.49070737 -0.09795486  3.01019895
   0.59780144  1.06643657]]
Intercept:  [-0.63239221]

Training Metrics
Accuracy: 0.7687296416938111
Precision: 0.7329192546583851
Recall: 0.543778801843318
F1 Score: 0.6243386243386243

Testing Metrics
Accuracy: 0.7662337662337663
Precision: 0.6744186046511628
Recall: 0.5686274509803921
F1 Score: 0.6170212765957446
```

My Logistic Regression model performed slightly better in training accuracy than the Scikit Learn's Logistic Regression model, but their model performed better in testing accuracy by 2% than my Logistic Regression model. The Scikit Learn's model had a higher recall and F1 score during testing, indicating that they made fewer mistakes on identifying "Positive-Class instances".

Scikit Learn Neural Network Results: (I also only used one hidden layer with 5 neurons here, and the sigmoid activation function)



```
Training Metrics
Accuracy: 0.7654723127035831
Precision: 0.6941176470588235
Recall: 0.5619047619047619
F1 Score: 0.6210526315789473

Testing Metrics
Accuracy: 0.7467532467532467
Precision: 0.7317073170731707
Recall: 0.5172413793103449
F1 Score: 0.606060606060606
```

The Scikit Learn's Neural Network had a slightly worse training accuracy than my Neural Network from scratch, but it had a little bit of a higher testing accuracy. The Scikit Learn's testing results were higher for precision and f1 score, but my model had a higher recall score. Thus, my model performed better at identifying "Positive class instances", but their model had a more balanced overall accuracy with identifying "False-Positives" and "False-Negatives".

Part 5: (Creating a very big Neural Network and effect on performance)

Theoretically speaking, adding more hidden layers to the Neural Network can help the network better fit to the shape of the output curve and learn complex correlations between features in the data. Each hidden layer would allow the network to more accurately create hierarchical representations of the features in the dataset. Thus, the network would be able to better identify patterns in the data, especially if each hidden layer had an increased number of neurons.

Despite all this, adding more layers or more neurons to the network could also lead to overfitting. Similarly to how a decision tree overfits to training data when its maximum depth is not restricted, a neural network also needs to have the right balance between how biased it is acting to the training data.