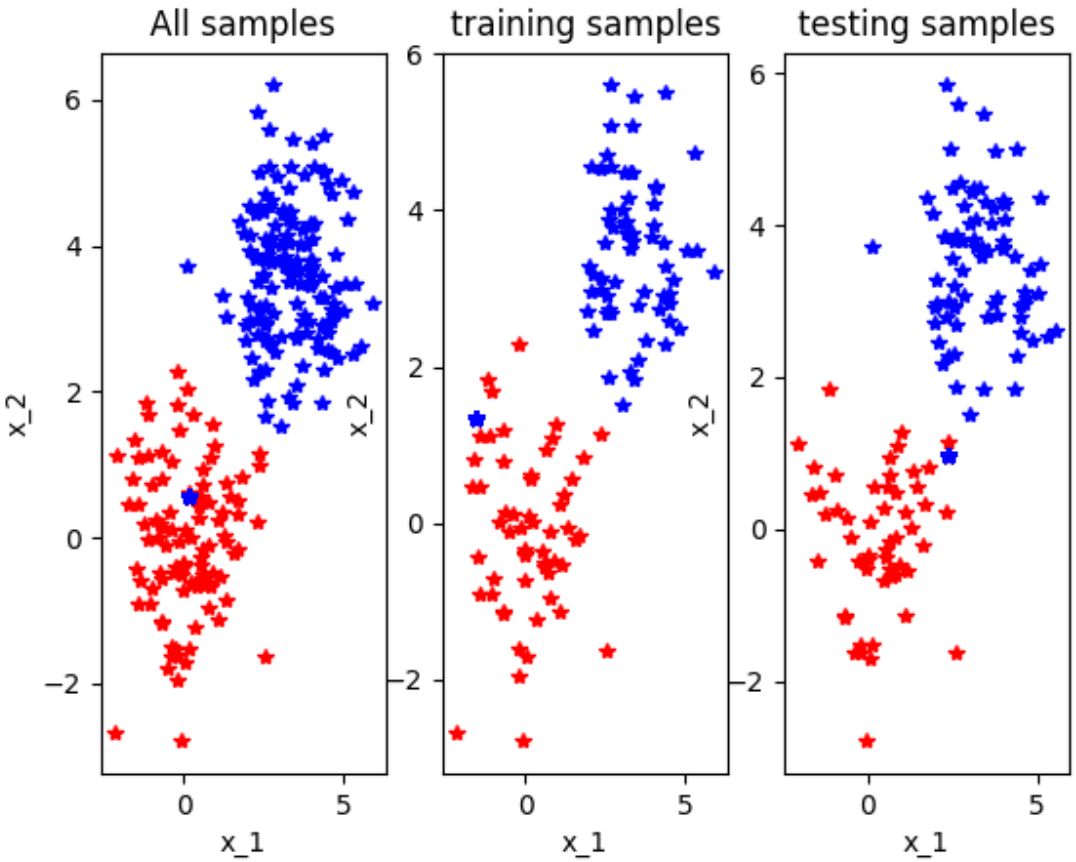
HW# 3 Logistic Regression Report

Overview:
I am creating a binary classifier using a logistic regression machine learning model. The model was built in 4 steps: Splitting Data, Training models, Comparing models, Making predictions, and Evaluating. I will compare the results of the classifier that I built to sklearn's built-in logistic regression function

Step 1: Splitting into Test and Training sets
I initially took a random subset of my data for training and then used the rest for testing. After splitting the data I plotted three images representing the process of splitting the data: All data, Training Data, Testing data.



Step 2: Training both models

After splitting the data, I started to build my sklearn logistic regression model first. This was simple because all I had to do was import logistic regression from sklearn library and call it into my main script. I then trained my sklearn model, using my training data and captured its score. The score represents how many times the model is making the correct predictions over the total amount of samples in the test data set. After training the sklearn model, I built and trained my own binary classifier using logistic regression and optimizing the parameters using gradient descent method. I then evaluated both models by comparing their scores (reported below).

My Score: 0.785
Sklearn Score: 0.985

Based on the scores reported from both models, the Sklearn logistic regression model is performing better than the model that we built manually.

Step 3: Making Predictions

After training my logistic regression model, I used it to get class labels from my test data. To compare I also printed out the predicted labels from sklearn.

My predictions: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
1]

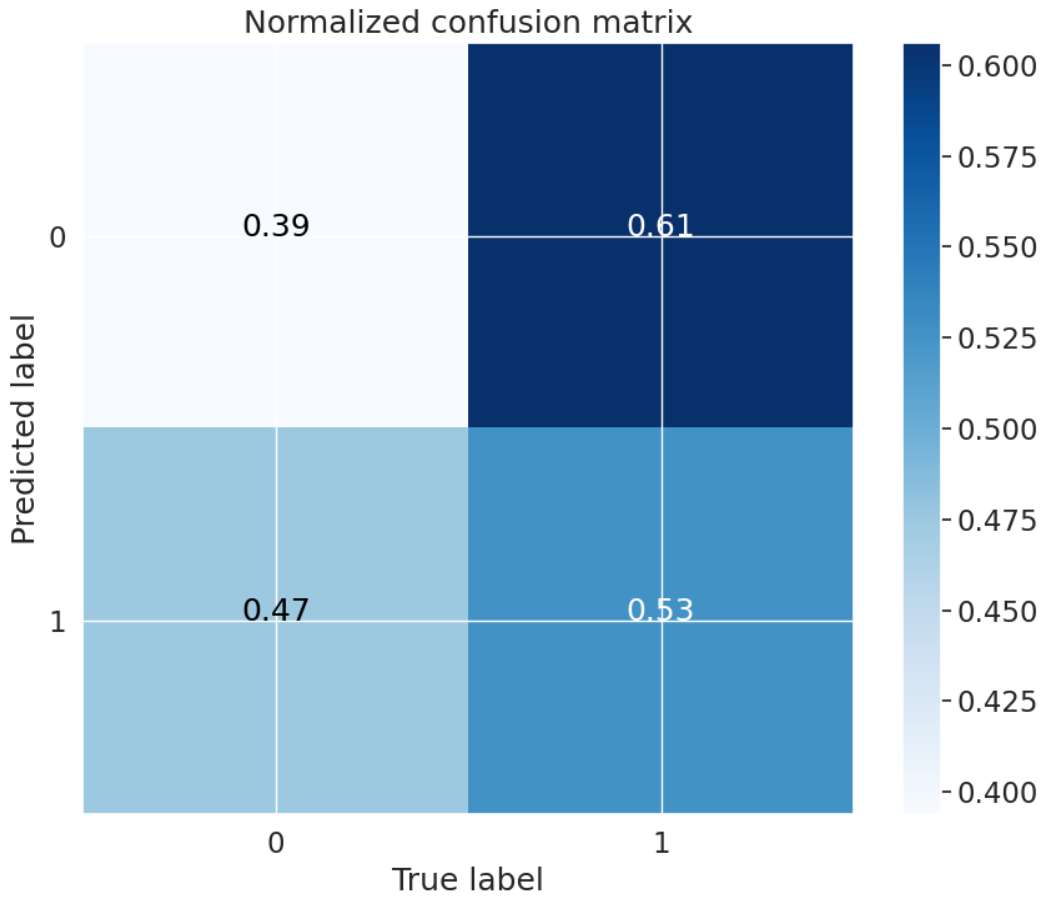
Sklearn predictions: [1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0. 0. 0. 0. 1. 1. 1. 0. 1. 0. 1. 1. 0.
1. 1. 0. 1. 1. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 1. 1. 0. 0.
1. 1. 0. 0. 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 1. 1. 0. 0. 1. 1. 0. 1. 1.
1. 1. 1. 1. 0. 1. 0. 1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 0. 1. 1. 1. 1. 1. 1.
1. 0. 1. 1. 1. 0. 1. 1. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 0. 1. 1. 1. 1. 1.
1. 0. 1. 0. 0. 1. 1. 1. 0. 1.]

At first glance, it seems that my predicted labels have a bias towards class 1 versus sklearns which seems to be calling class 0 more frequently. Considering that sklearn has a higher score from the previous step, it is probably closer to the actual labels from the test dataset.

Step 4: Evaluation

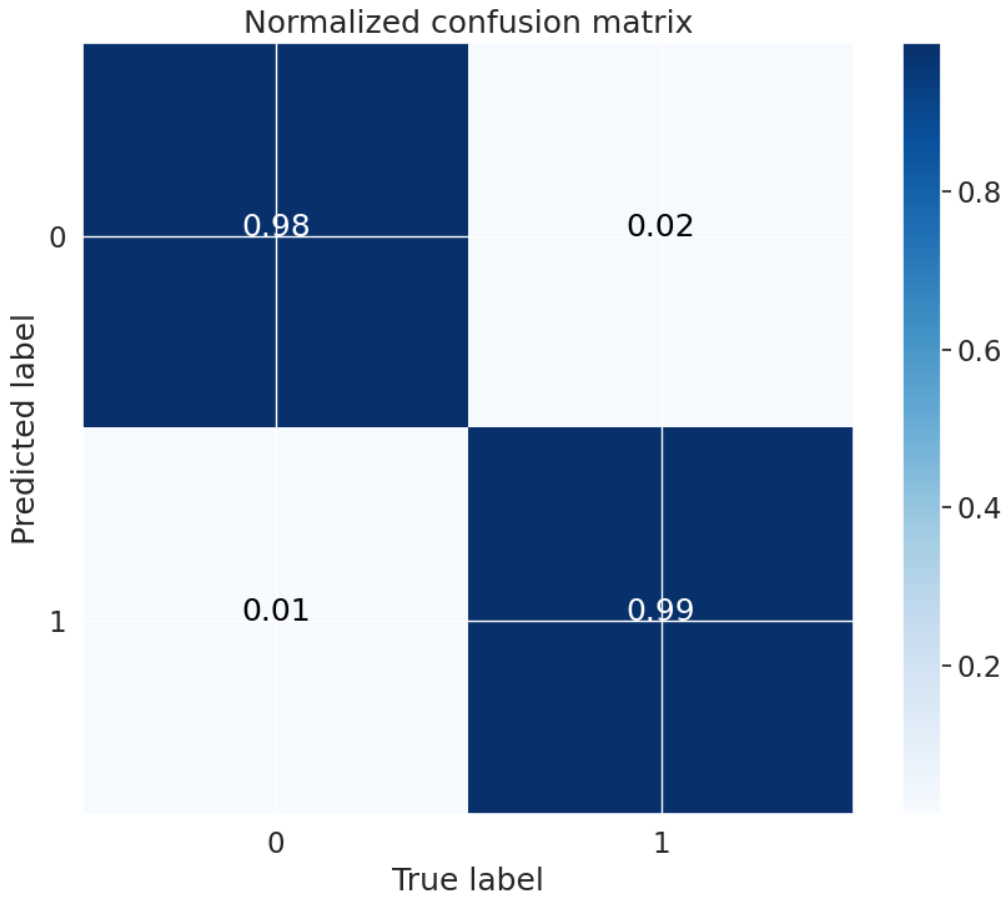
Using both predicted labels from the manually built model and sklearn model, I created a confusion matrix visualizing the performance of both models. After making the confusion matrix, I calculated the recall and precision for each class in both models.

Manual:



precision for classes [0,1]: [0.22, 0.71]
recall for classes [0,1]: [0.39, 0.53]
Accuracy: 0.49

Sklearn:



precision for classes [0,1]: [0.98, 0.99]
recall for classes [0,1]: [0.98, 0.99]
Accuracy: 0.99

Summary:
Comparing both confusion matrices we can see that sklearn is more accurately classifying both classes. The model that I made manually seems to be misclassifying both classes (0,1) at the same rate. Since I am taking a random subset of the data each time I run my code the results may differ from time to time. In general it seems that the manual model classifies the 1 class more often than the 0 class. This could be due to the conditions that were put on the prediction function (if a > 0.5: a==1). Another possible reason could be that our sigmoid function is not good for predicting values and can be updated with another method, such as Relu. In conclusion the logistic regression model built in sklearn seems to perform better than the model that I built manually.