In the lecture04_code_logistic.zip file, we presented two implementations of a logistic regression model: one using Scikit-learn and the other developed with custom functions. Please modify the self-built implementation in two key ways.

First, the self-built model currently uses two features but has only two parameters, theta_1 and theta_2, with theta_0 set to 0. Please introduce the intercept coefficient, theta_0, into the model and adjust the gradient descent algorithm to accommodate this change. In your write-up, compare the performance of the modified model to the original implementation.

The Original model performed a little bit worse than the Scikit model overall, it was constantly a little bit less accurate. Scikits's accuracy was often in the high 80%s to low 90%s while the Original model was often in the mid 80s. However it was A LOT more variable, some accuracies dropped down into the 70s and reached as high as 94. Adding in the theta_0 helped significantly. With theta_0, the accuracy was often on par with the Scikit model. I don't think it was as accurate however, there were several times where it was 1-2% less accurate than the Scikit model but importantly it was much less variable, most of the output stood in the high 80s to low 90s, much like the Scikit model. Overall the model with theta_0 performed noticeably better than the Original model, which I think is to be expected.

Second, the self-built model currently employs a full-batch gradient descent algorithm, using all training samples in each iteration. Modify it to implement mini-batch gradient descent. For each iteration, randomly select a batch size of 20 training samples to calculate the gradient. Additionally, experiment with batch sizes of 30 and 60, and analyze how these changes impact your model's performance.

I ran the code several times as the train_test_split() function randomizes which data points are used for training and testing, so no two runs are ever the same. Overall I found that the minibatch decent model with a batch size of 20 was the most accurate. It had an average accuracy of 90.304% and a stdDev of 3.72% which just barely beat out the minibatch model with a batch size of 30. The model with a batch size of 60 was on par

with the SciKit model, as their average accuracy and standard deviations are nearly identical. All mini decent models performed better than the full batch models in both accuracy and variability.

Here is the table I created with 10 different runs for each model, listing their accuracy in each run, then calculating the average accuracy and the standard deviation:

SciKit	MiniBatch(60)	MiniBatch(30)	MiniBatch(20)	GD_Intercept	GD
84.85	87.88	87.88	90.91	84.85	78.79
87.88	87.88	90.91	84.85	87.88	87.88
81.82	90.91	93.94	90.91	84.85	78.79
93.94	90.91	87.88	93.94	93.94	93.94
87.88	87.88	90.91	93.94	87.88	84.85
87.88	87.88	84.85	87.88	87.88	81.82
87.88	84.85	90.91	87.88	87.88	87.88
90.91	90.91	87.88	96.97	90.91	84.85
87.88	84.85	84.85	87.88	78.79	78.79
84.85	84.85	93.94	87.88	84.85	81.82
Average					
87.577	87.88	89.395	90.304	86.971	83.941
StdDev					
3.334529952	2.47398464	3.272774053	3.724695961	4.052605335	4.958266834

The code was submitted along with this pdf.