ECE 2372 - Homework 2 on Logistic Regression

Jiyang Liu 4731134

Problem 1: Gradient Descent

Let's implement Logistic Regression and use gradient descent algorithm to perform the maximum likelihood estimate of our parameters. For the step size I would like you to experiment with different values, this is a common approach in deciding step size. Explicitly start somewhere $\alpha \approx 1$ and work your way down to α values well below that $\alpha \ll 1$ (may by cutting in half or tenth), also play around with your stopping criteria.

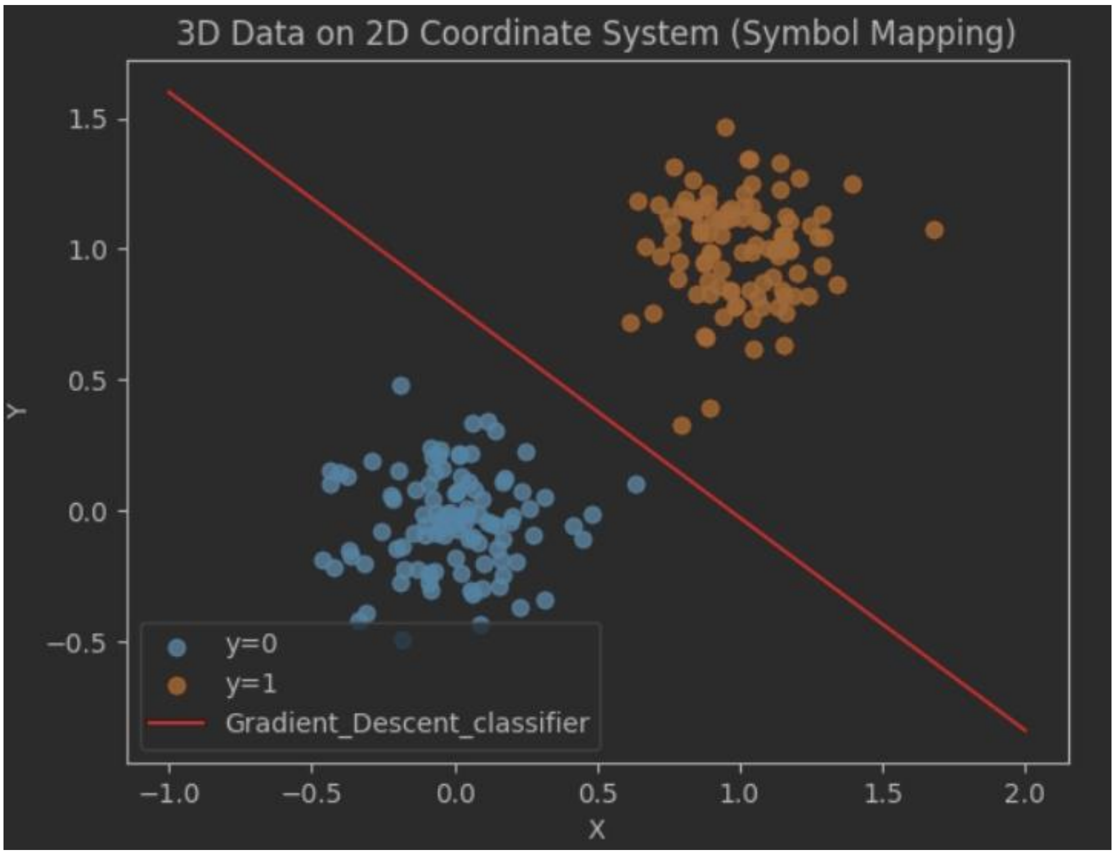
a. Implement gradient descent algorithm for this problem.

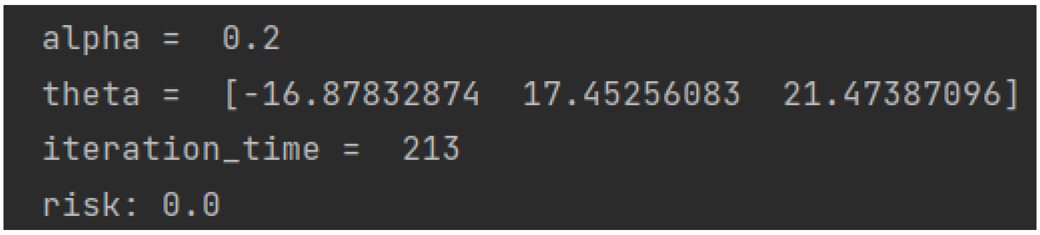
b. Test your code on the four given synthetic data sets. Report the value of step size you use and the iterations required for each dataset. Please plot classifiers your implementation returns for each data set. I would like you to also compare your results with the results you obtained with LDA in the first assignment.

c. Did you notice any speed difference in the convergence among these synthetic data sets? If so, why would you think this might of happened?

The code is in HW2_1.ipynb.

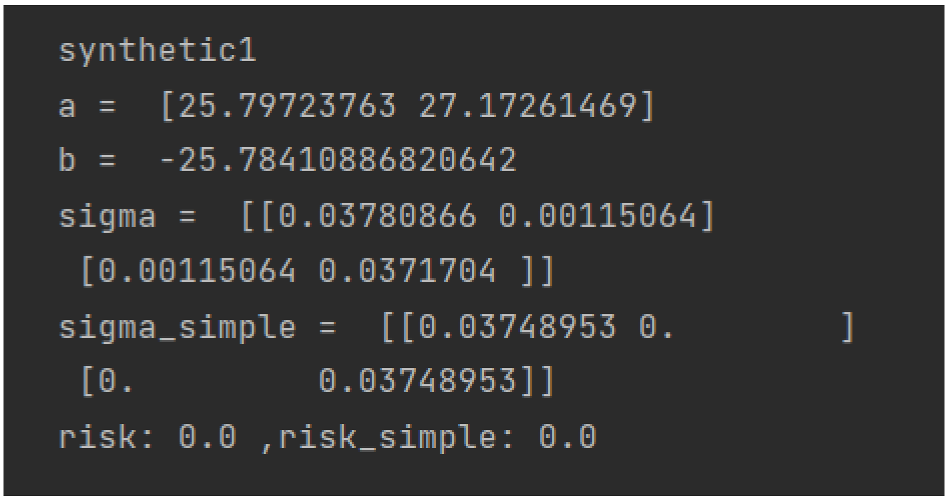
Synthetic 1





```
alpha =  0.2
theta =  [-16.87832874  17.45256083  21.47387096]
iteration_time =  213
risk: 0.0
```

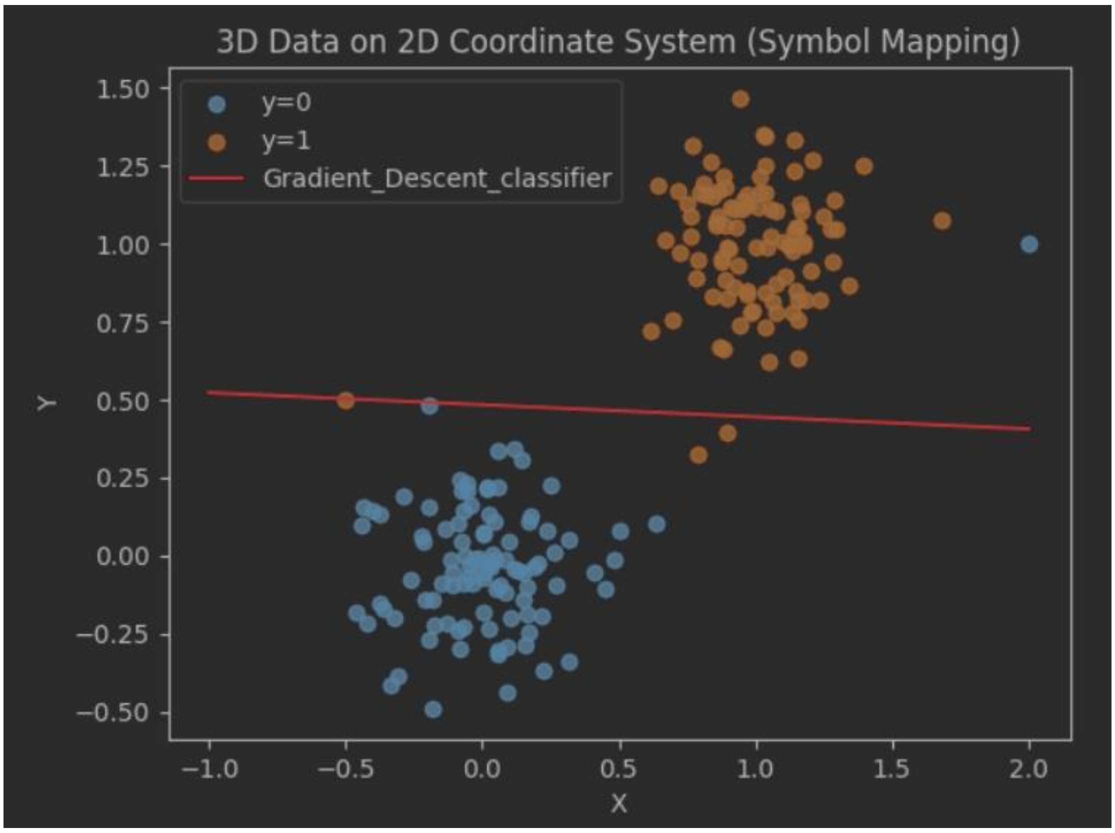
Compared with LDA:

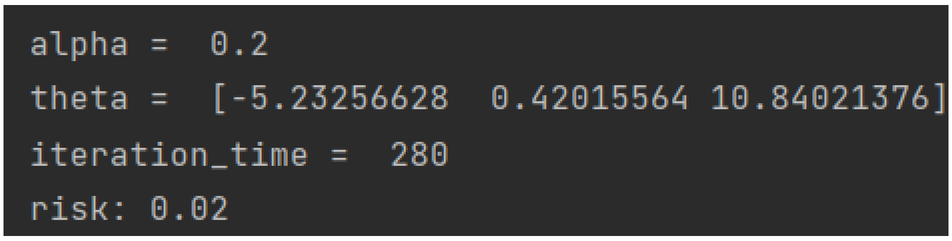


```
synthetic1
a =  [25.79723763 27.17261469]
b =  -25.78410886820642
sigma =  [[0.03780866 0.00115064]
 [0.00115064 0.0371704 ]]
sigma_simple =  [[0.03748953 0.        ]
 [0.         0.03748953]]
risk: 0.0 ,risk_simple: 0.0
```

Risks is all 0, and they have completed the classification very well.

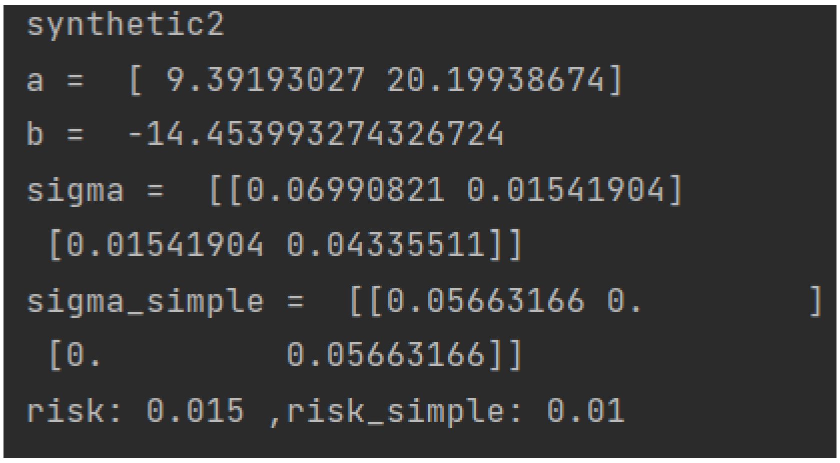
Synthetic 2





```
alpha =  0.2
theta =  [-5.23256628  0.42015564 10.84021376]
iteration_time =  280
risk: 0.02
```

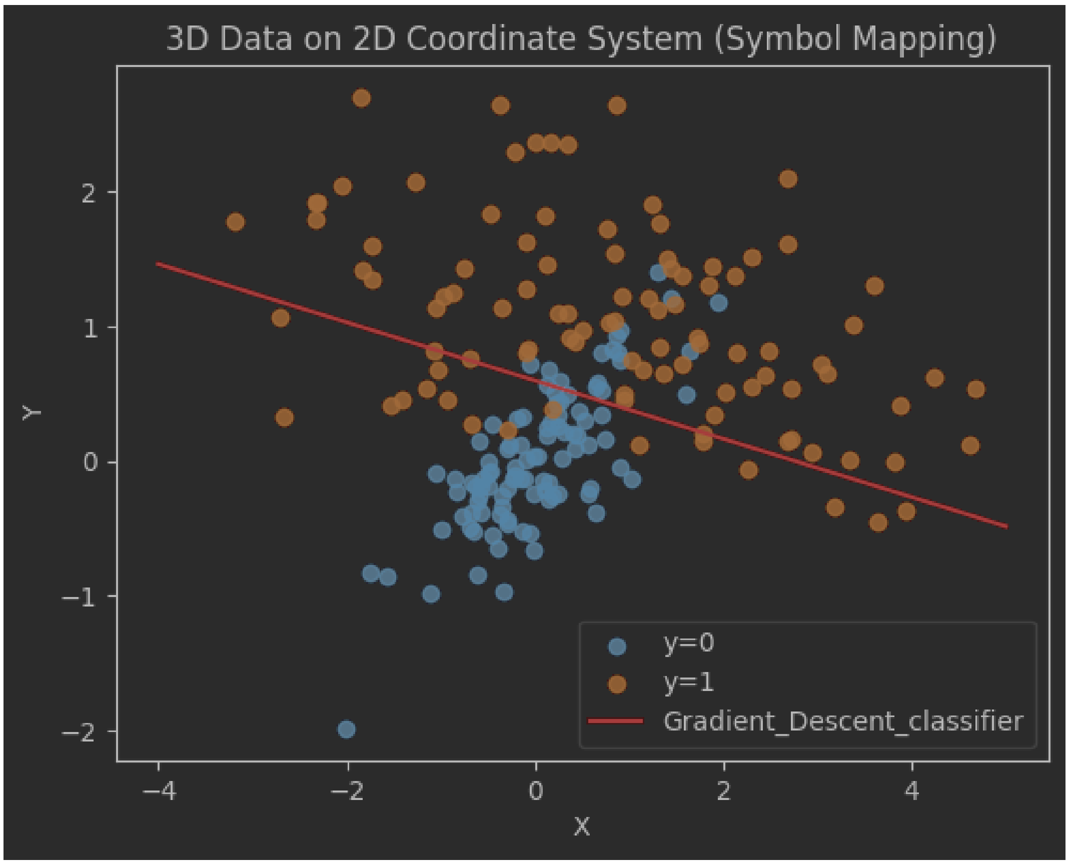
Compared with LDA:

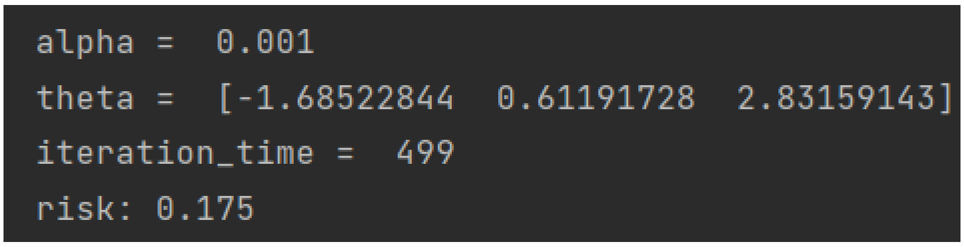


```
synthetic2
a =  [ 9.39193027 20.19938674]
b =  -14.453993274326724
sigma =  [[0.06990821 0.01541904]
 [0.01541904 0.04335511]]
sigma_simple =  [[0.05663166 0.        ]
 [0.         0.05663166]]
risk: 0.015 ,risk_simple: 0.01
```

The Risks are very similar, both are low, and they both complete the classification well.

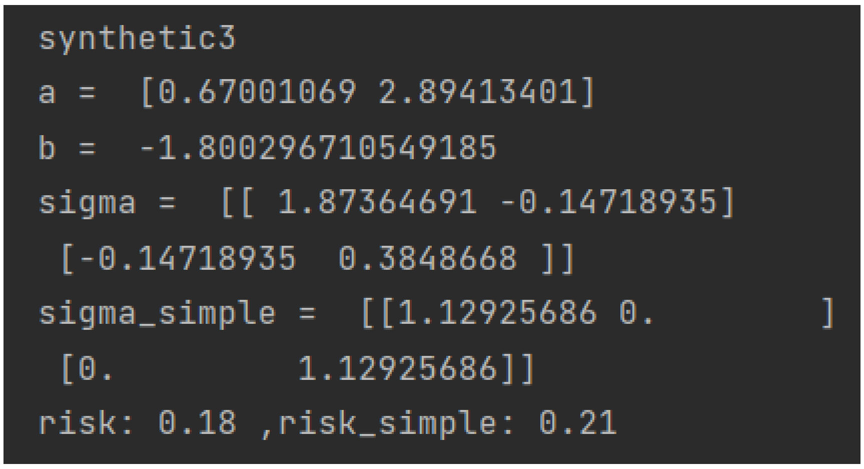
Synthetic 3





```
alpha =  0.001
theta =  [-1.68522844  0.61191728  2.83159143]
iteration_time =  499
risk: 0.175
```

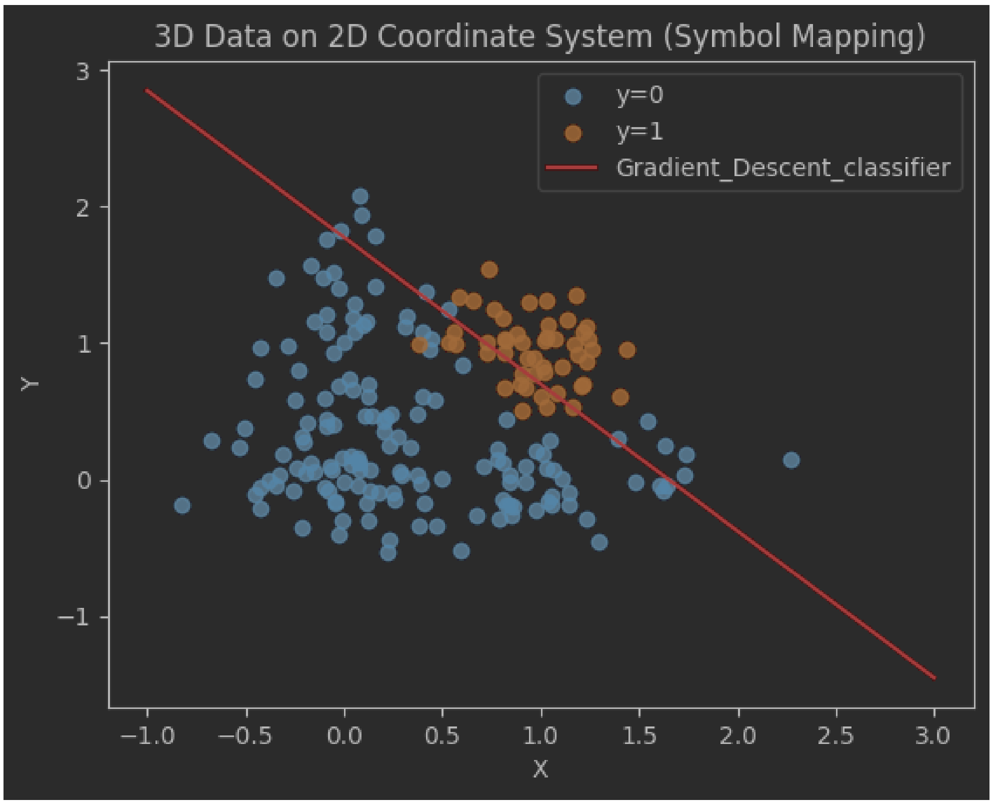
Compared with LDA:

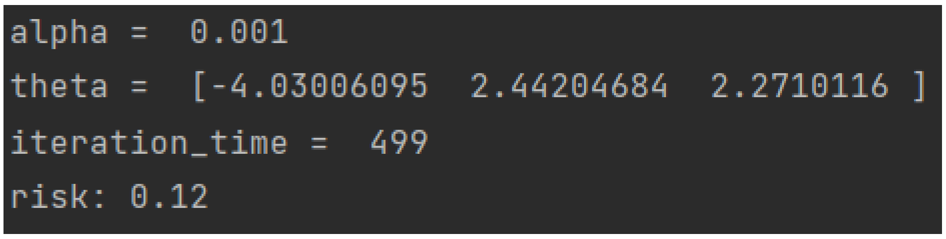


```
synthetic3
a =  [0.67001069 2.89413401]
b =  -1.800296710549185
sigma =  [[ 1.87364691 -0.14718935]
 [-0.14718935  0.3848668 ]]
sigma_simple =  [[1.12925686 0.        ]
 [0.         1.12925686]]
risk: 0.18 ,risk_simple: 0.21
```

If using the original step size in data with a high degree of mixing, the risk will be very large. We reduced the step size and the risks of the final result were very close to LDA.

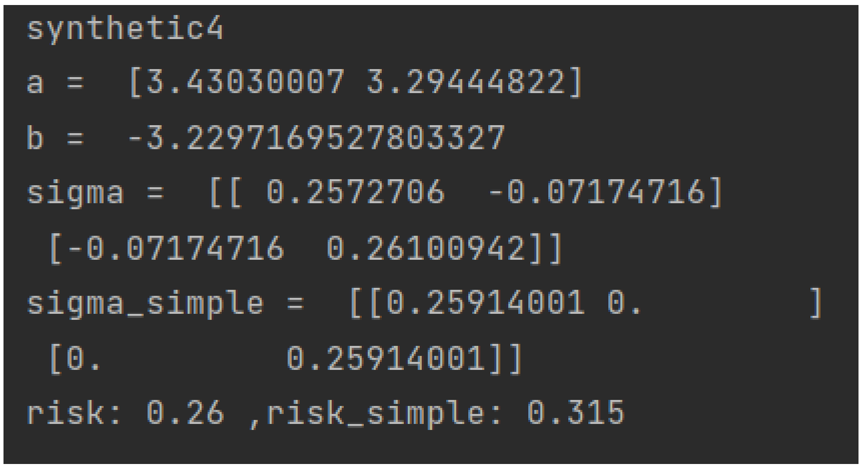
Synthetic 4





```
alpha =  0.001
theta =  [-4.03006095  2.44204684  2.2710116 ]
iteration_time =  499
risk: 0.12
```

Compared with LDA:



```
synthetic4
a =  [3.43030007 3.29444822]
b =  -3.2297169527803327
sigma =  [[ 0.2572706  -0.07174716]
 [-0.07174716  0.26100942]]
sigma_simple =  [[0.25914001 0.        ]
 [0.         0.25914001]]
risk: 0.26 ,risk_simple: 0.315
```

The mixing degree of the data in Synthetic 4 is similar to that in Synthetic 3, but the data in Synthetic 4 is more concentrated. The risk of gradient descent is much smaller than the risk of LDA, indicating that the gradient descent method is more suitable in this situation.

Problem 2: Newton's Method Now let's try out Newton's method as a solver to MLE step in logistic regression.

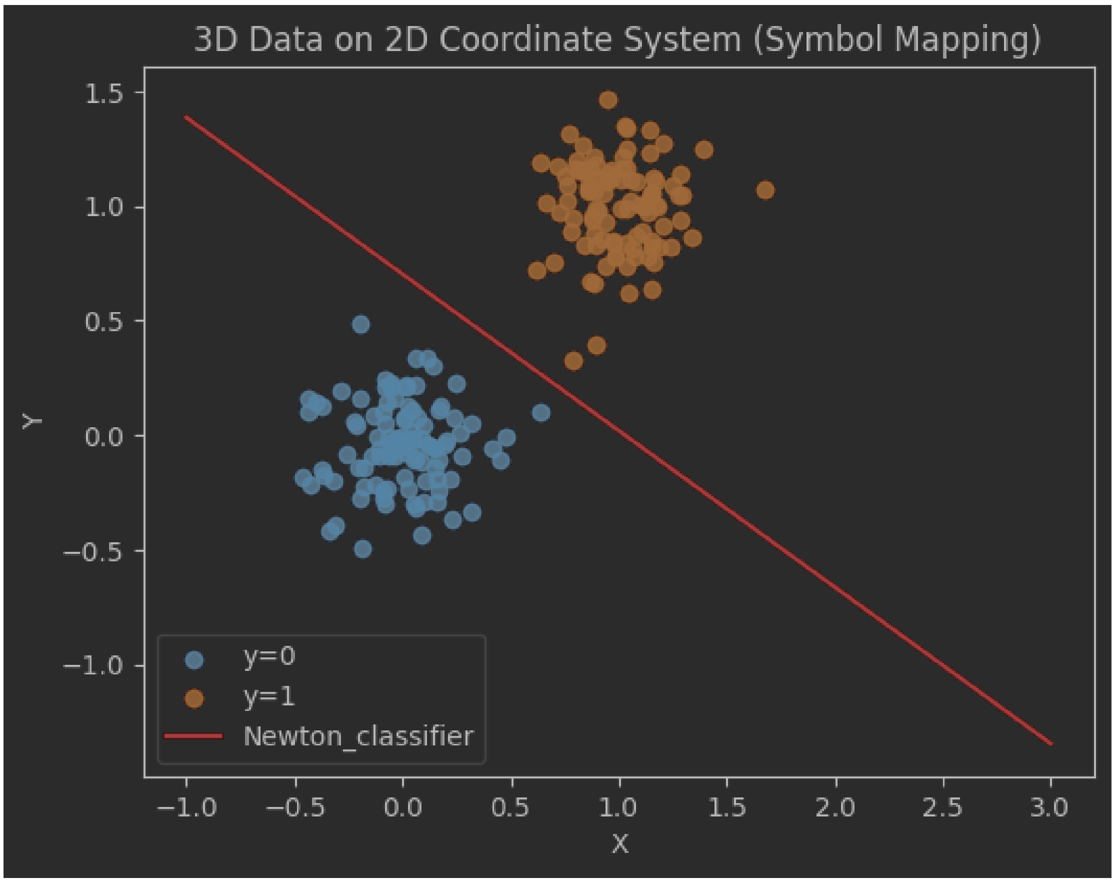
a. Implement Newton's method for this section. Note that I calculated the Hessian below, (you can verify if you want to but not need to):

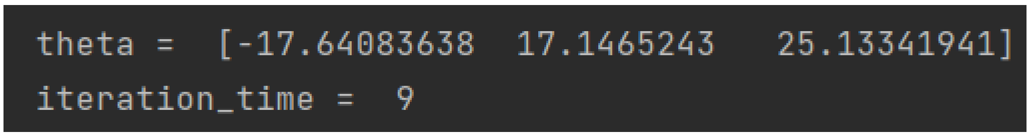
$$\frac{\nabla^2 l(\theta)}{\partial \theta} = -\sum_{i=1}^{n} \tilde{x}_i \tilde{x}_i^T g(\theta^T \tilde{x}_i)(1 - g(\theta^T \tilde{x}_i))$$

b. Test your code on the four given synthetic data sets. Report the number of iterations required for each dataset and compare it with the results you obtained from the first problem. Plot the classifier (should be same with the problem 1).

The code is in HW2_2.ipynb.

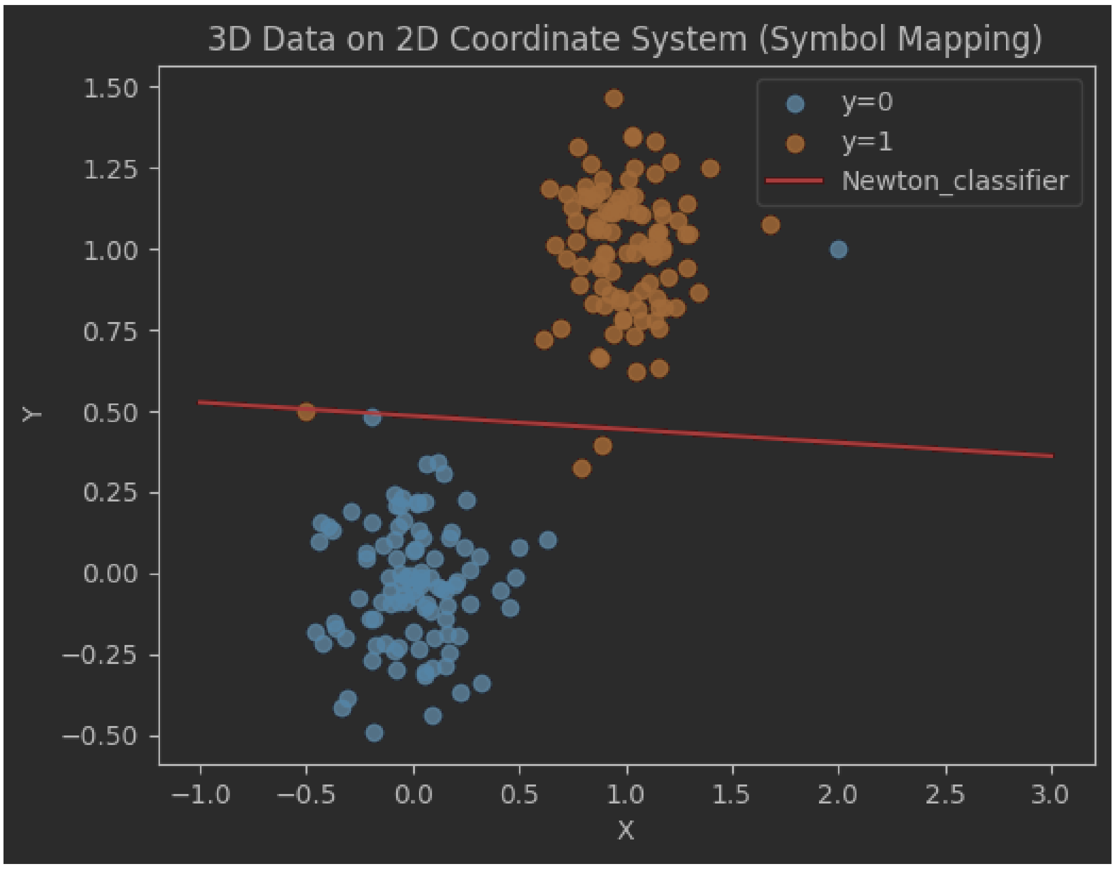
Synthetic 1

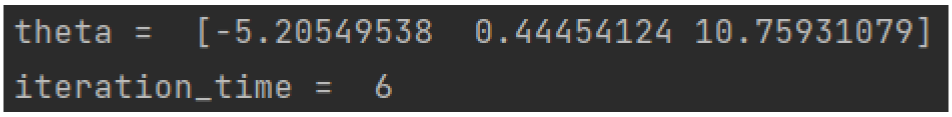




```
theta =  [-17.64083638  17.1465243   25.13341941]
iteration_time =  9
```

Synthetic 2





```
theta =  [-5.20549538  0.44454124 10.75931079]
iteration_time =  6
```

Discussion:

From Synthetic 1 and Synthetic 2, we can see that the number of iterations is much smaller than gradient decent. This shows that the Newton method can make the gradient drop very fast, and does not require multiple experiments to determine the step size, which can save a lot of time.

For Synthetic 3 and Synthetic 4, I didn't get valid result. It shows that my theta didn't converge. I think the Newton method may not be stable for messy data, or there may be something wrong with my code.

Problem 3: Stochastic Gradient Descent

Lastly let's implement stochastic gradient descent to solve the maximum likelihood estimation step in the logistic regression. In regular gradient descent, in order to compute the gradient we need to comput the following sum

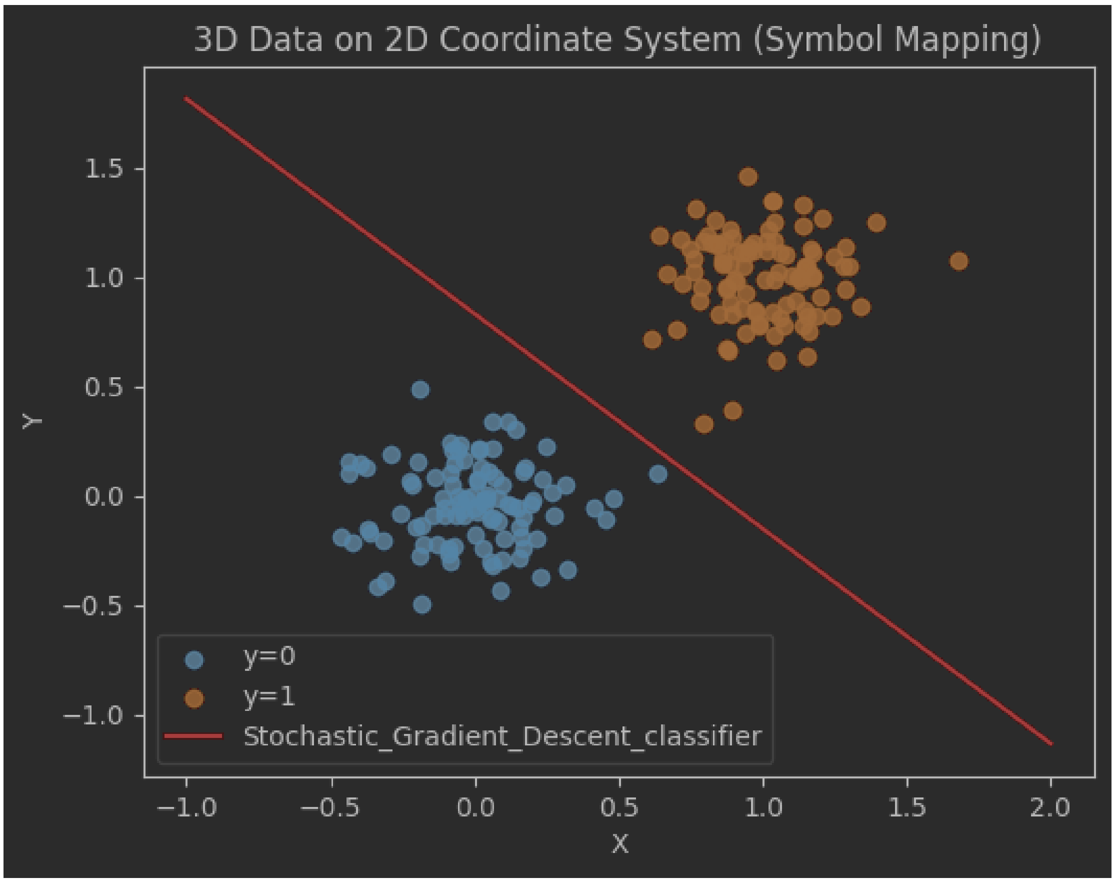
$$\sum_{i=1}^{n} \tilde{x}_i \left(y_i - g(\theta^T \tilde{x}_i)\right)$$

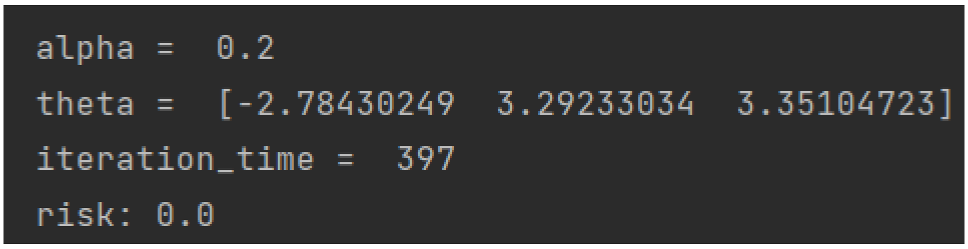
This sum gets challenging as n gets large (and this is not a problem in the data-sets we are using here). To tackle the computational burden with increasing n, we can select one $\tilde{x}_i$ at random and compute $\tilde{x}_i \left(y_i - g(\theta^T \tilde{x}_i)\right)$ as a rough approximation to the sum. This method is known as stochastic gradient descent and in general results in more iterations while enabling to do each iteration cheaper and can help when n is very large.

Implement stochastic gradient version of Logistic regression. Test your implementation with the same data sets. Please write down the number of iterations required for each datasets and compare the results with the two previous versions. Include a copy of figures depicts the resulting classifiers.

The code is in HW2_3.ipynb.

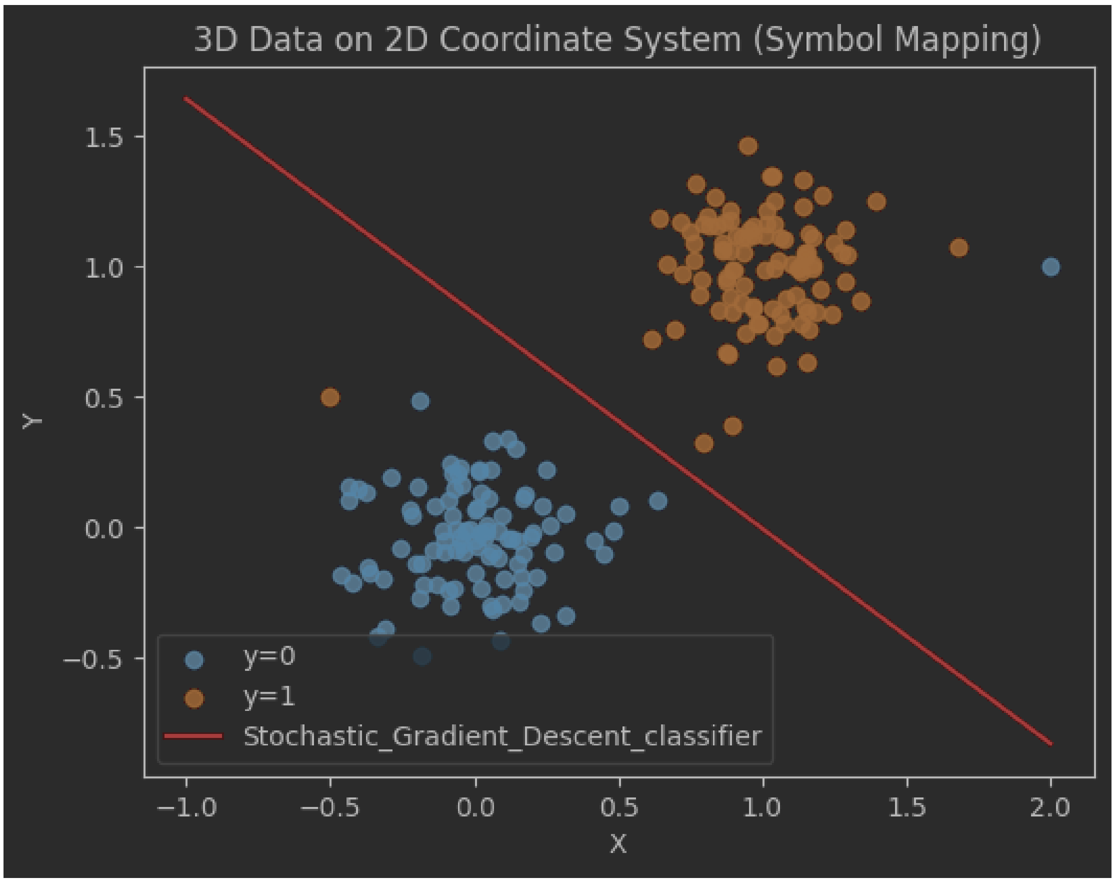
Synthetic 1

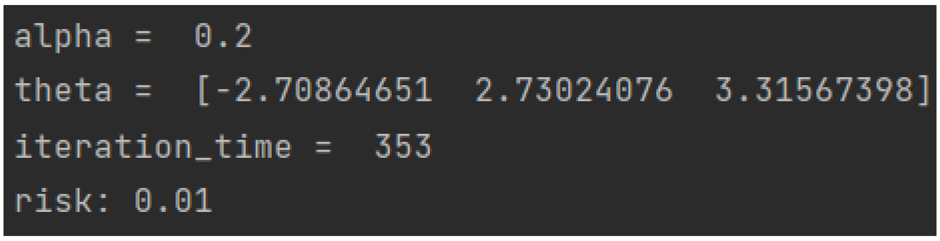




```
alpha =  0.2
theta =  [-2.78430249  3.29233034  3.35104723]
iteration_time =  397
risk: 0.0
```

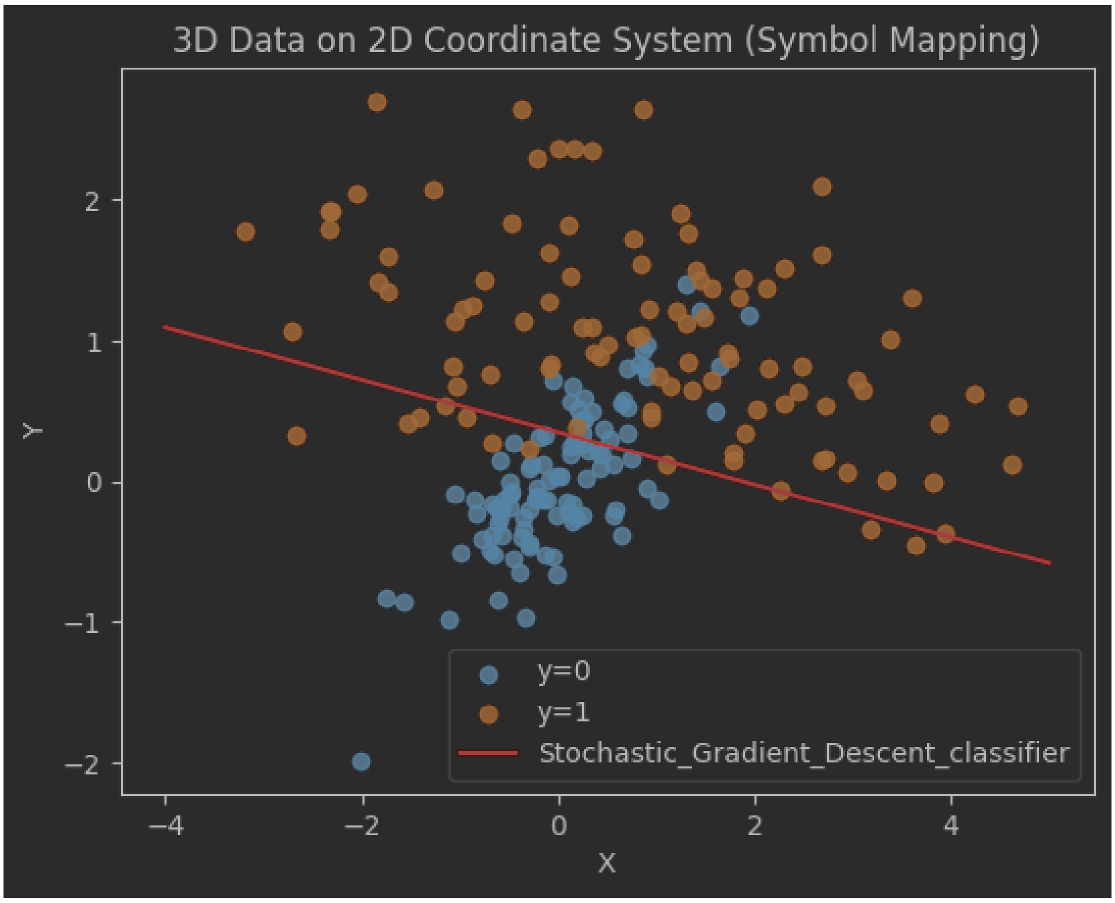
Synthetic 2

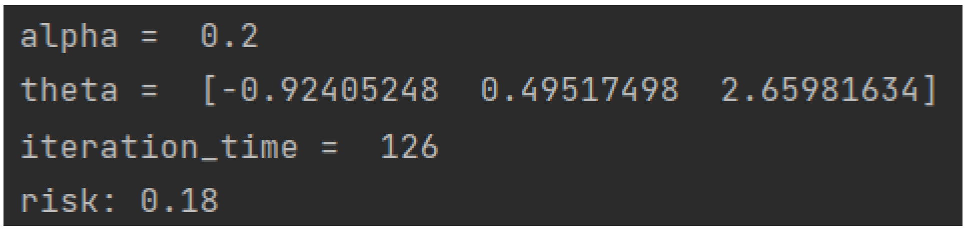




```
alpha =  0.2
theta =  [-2.70864651  2.73024076  3.31567398]
iteration_time =  353
risk: 0.01
```

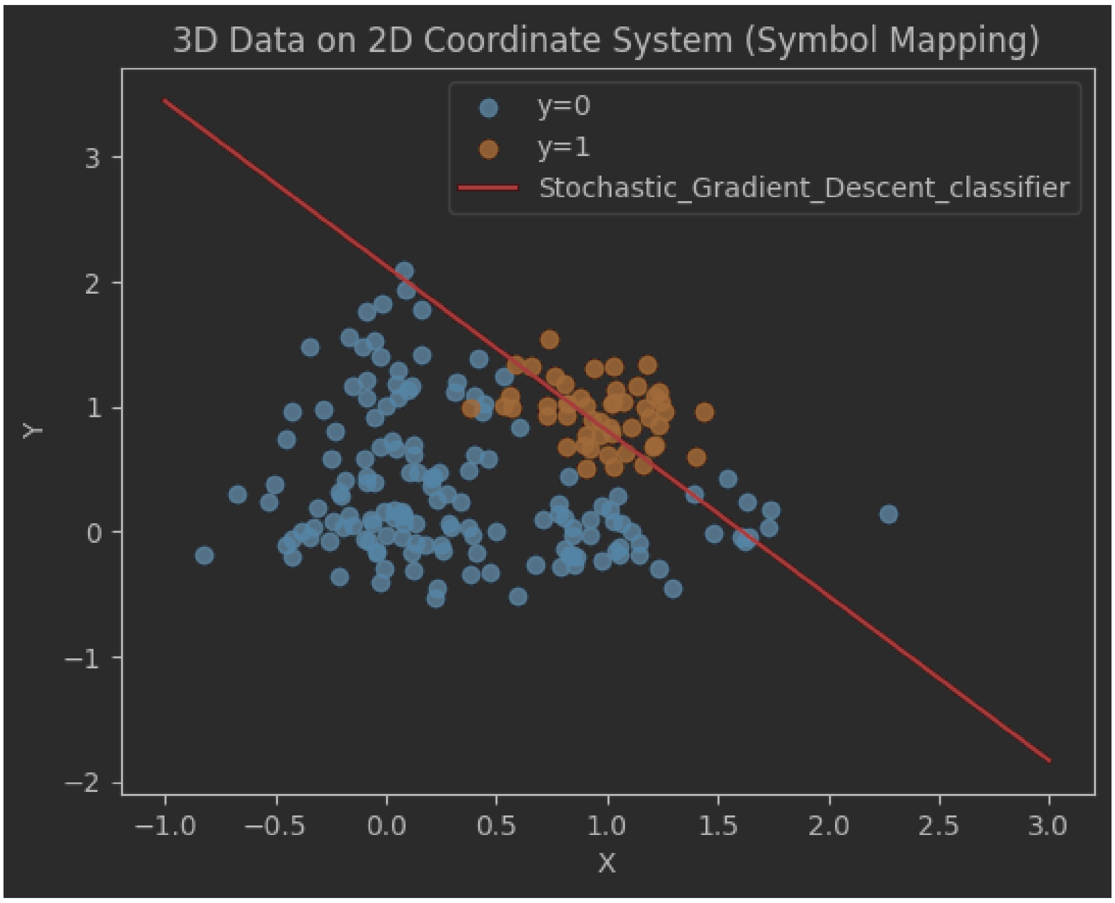
Synthetic 3

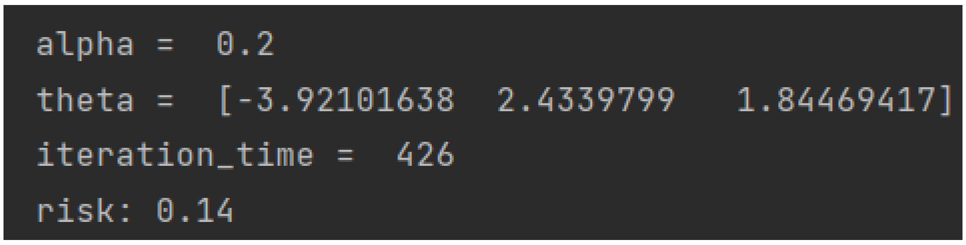




```
alpha =  0.2
theta =  [-0.92405248  0.49517498  2.65981634]
iteration_time =  126
risk: 0.18
```

Synthetic 4





```
alpha =  0.2
theta =  [-3.92101638  2.4339799   1.84469417]
iteration_time =  426
risk: 0.14
```

Discussion:
For Synthetic 1 and Synthetic 2, the stochastic gradient descent method has the largest number of iterations among the three methods. This shows that stochastic gradient descent is not a good method for data that is easy to classify.

For Synthetic 3 and Synthetic 4, the risk and number of iterations of stochastic gradient descent are the least. This shows that this method works well for complex and large amounts of data.