Cory Wixom
803494504

Sub-task1: See python code.

```
################ START TODO ################
# Hint: theta = (X^T X)^{-1} X^T y
XTranspose = X.transpose()
XTransposeProductWithX = XTranspose.dot( X )
XTransposeProductWithy = XTranspose.dot( y )
theta_method1 = inv(XTransposeProductWithX).dot(XTransposeProductWithy)
################ END TODO ##################
```

Subtask2: See python code.

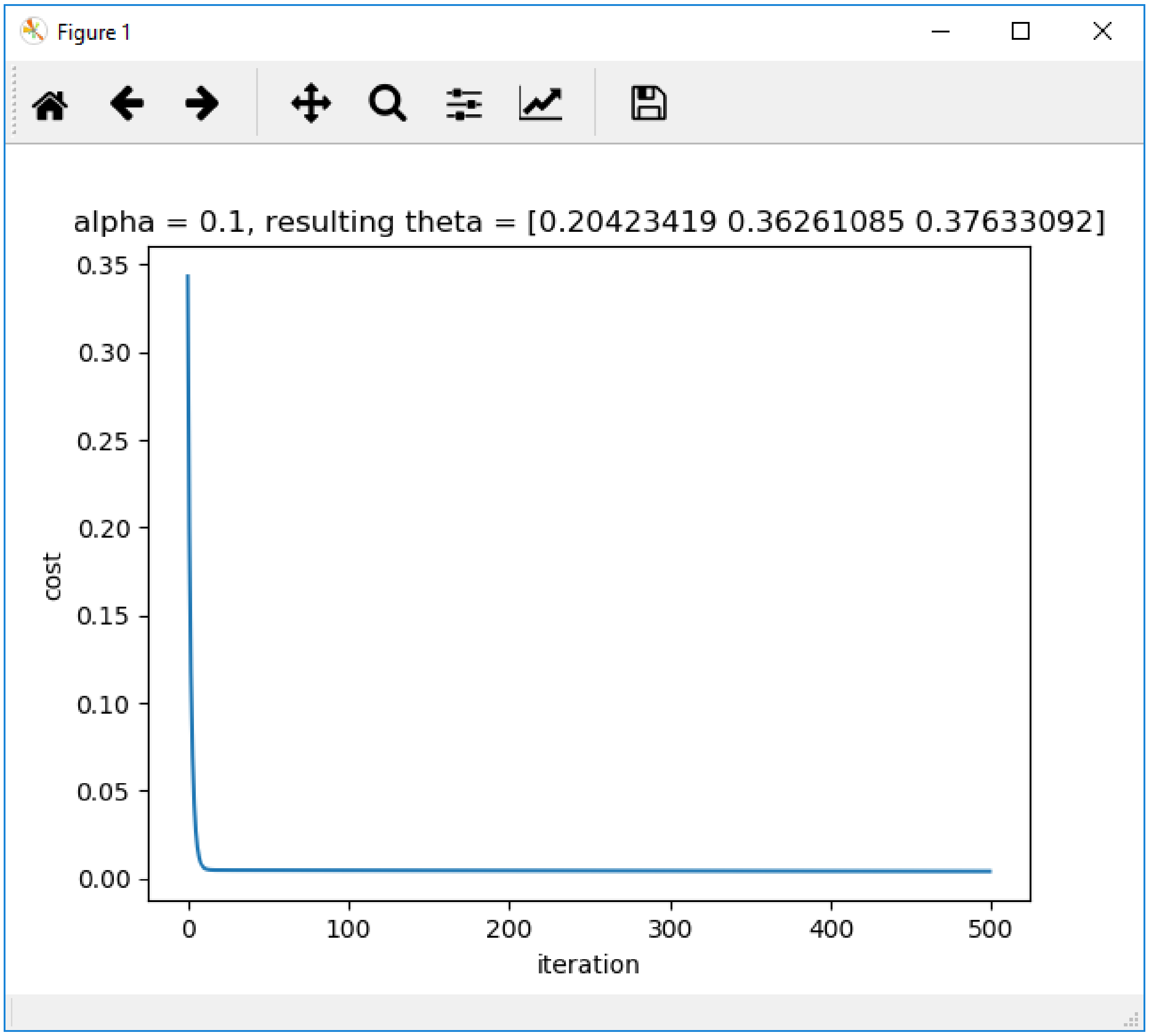
```
       ################ START TODO #################
        # Make predictions
        # Hint: y_hat = theta_0 + (theta_1 * x_1) + (theta_2 * x_2)
        # Shape of y_hat: m by 1
        y_hat = X.dot(theta)

        # Compute the difference between predictions and true values
        # Shape of residuals: m by 1
        residuals = y_hat - y

        # Calculate the current cost
        cost =  np.sum(residuals ** 2)/(2 * m)
        cost_array.append(cost)

        # Compute gradients
        # Shape of gradients: 3 by 1, i.e., same as theta
        gradients = X.T.dot(residuals) / m

        # Update theta
        theta = theta - alpha * gradients
        ################ END TODO ##################
```

Subtask3:

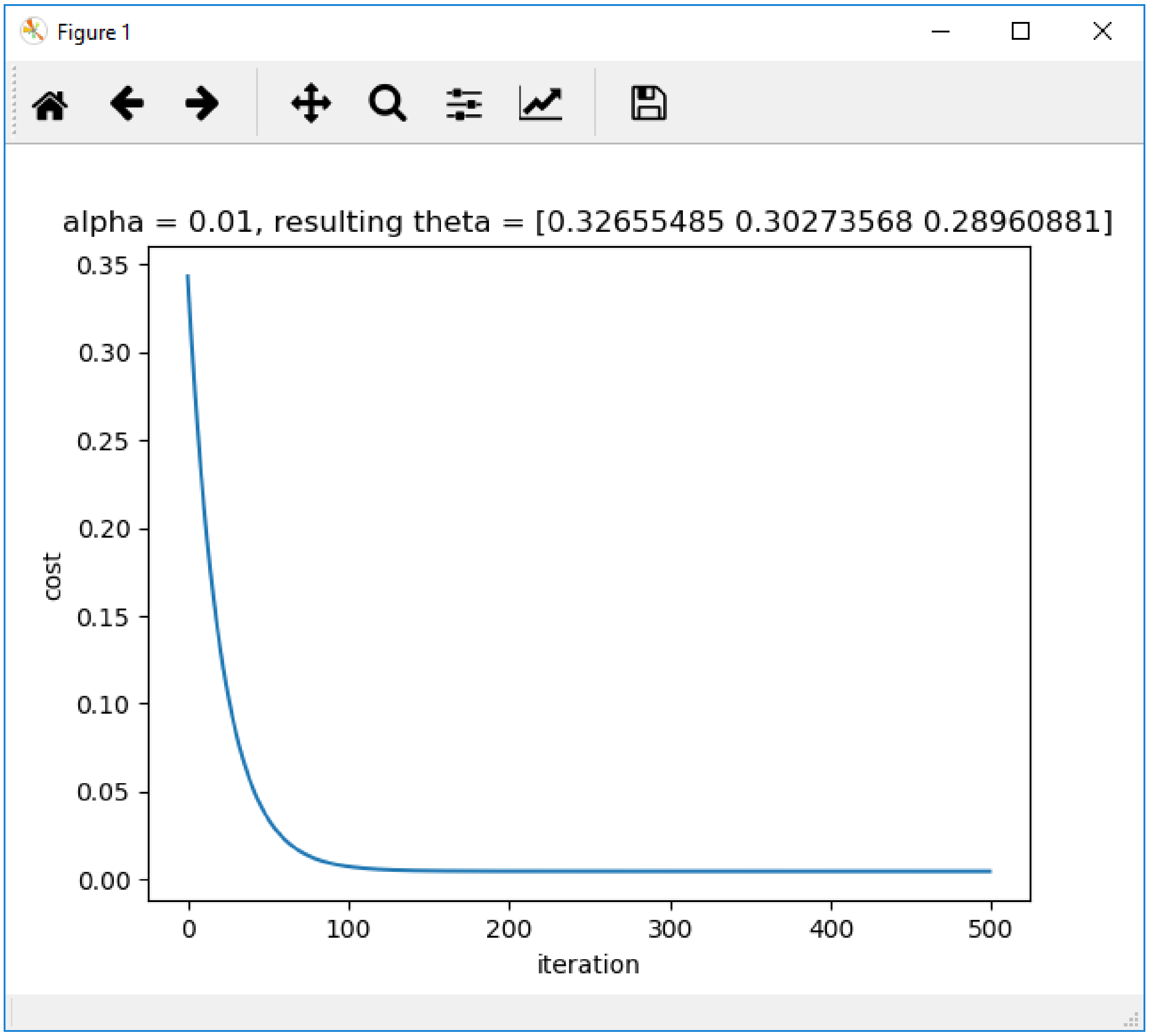
```
################ START TODO ################
# Hint: make predictions using X_test and theta_method1
yhat_test_1 = X_test.dot(theta_method1)
mse_1 = np.mean((yhat_test_1 - y_test)**2)
print('MSE of theta_method1 on testing data: ', mse_1)

# Hint: make predictions using X_test and theta_method2
yhat_test_2 =X_test.dot(theta_method2)
mse_2 = np.mean((yhat_test_2 - y_test)**2)
print('MSE of theta_method2 on testing data: ', mse_2)
################ END TODO ##################
```

Subtask4:

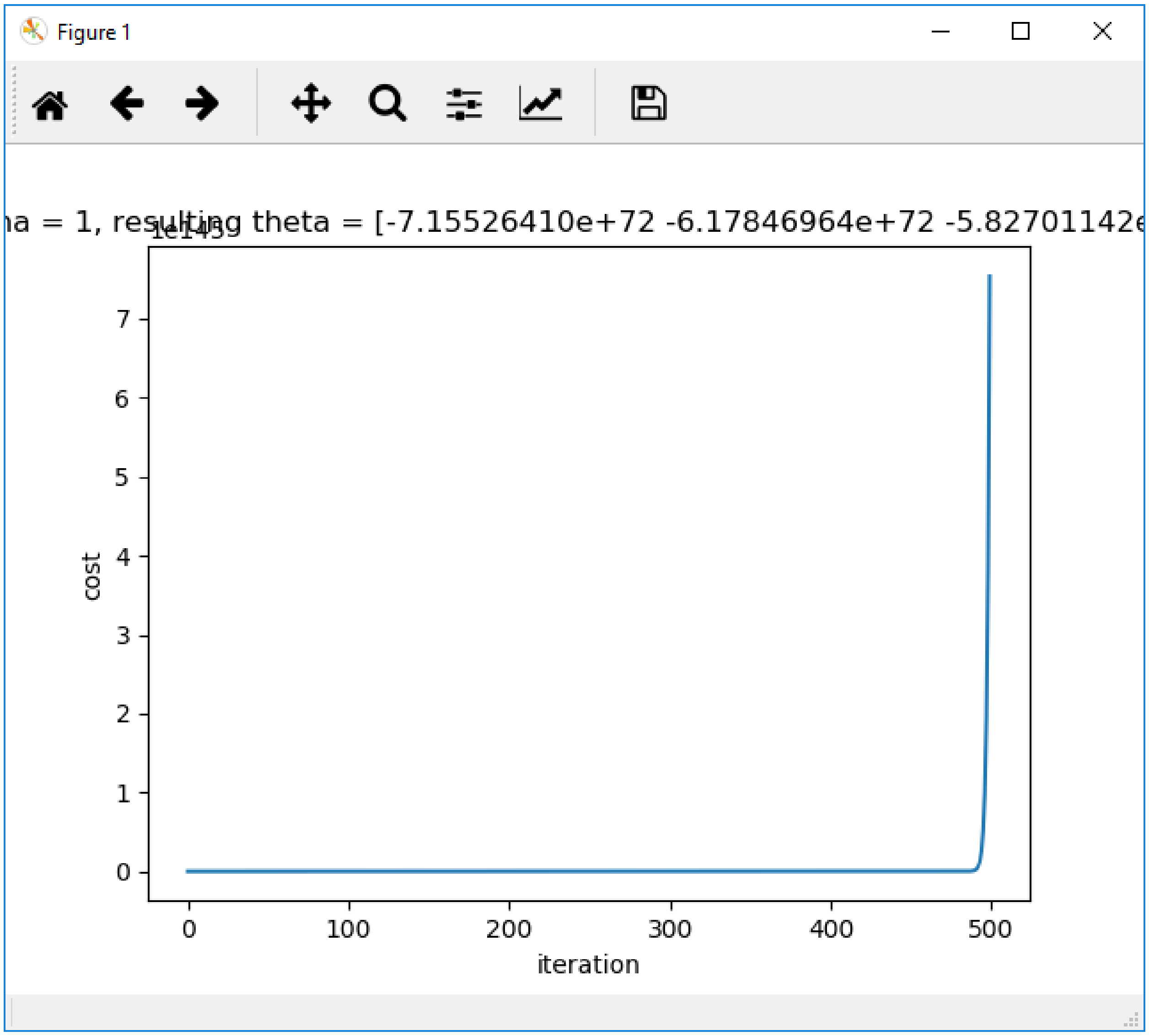
```
ALPHA = 0.1
MAX_ITER = 500
Shape of original data: (105, 3)
theta_method1: [-0.31358288  0.87211124  0.47195201]
Residual sum of squares (RSS) of method 1:  0.3747279427571582
theta_method2: [0.20423419 0.36261085 0.37633092]
Residual sum of squares (RSS) of method 2:  0.4960871040120984
MSE of theta_method1 on testing data:  0.00963677250182623
MSE of theta_method2 on testing data:  0.007237429379813844
```



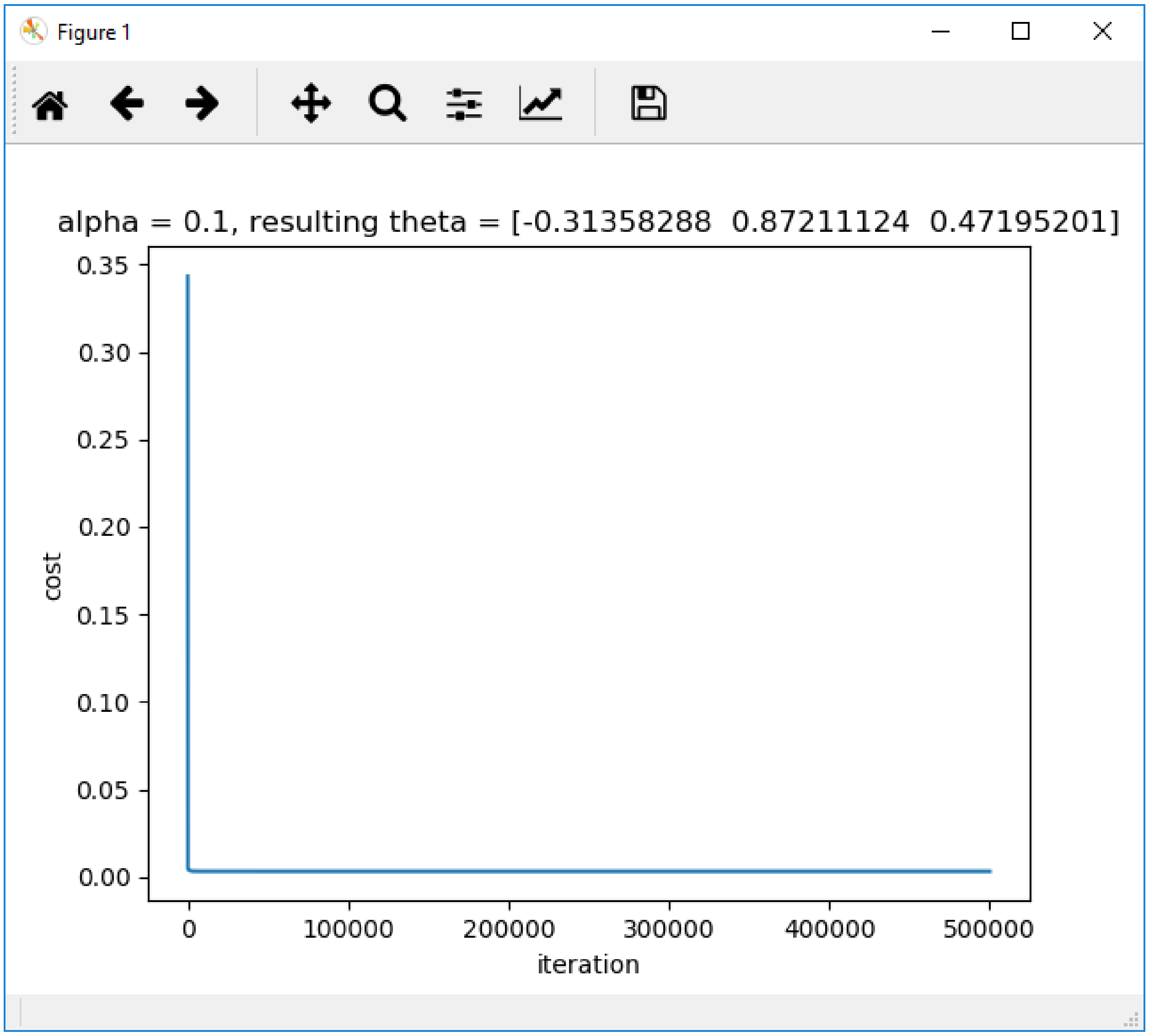
```
ALPHA = 0.01
MAX_ITER = 500
Shape of original data: (105, 3)
theta_method1: [-0.31358288  0.87211124  0.47195201]
Residual sum of squares (RSS) of method 1:  0.3747279427571582
theta_method2: [0.32655485 0.30273568 0.28960881]
Residual sum of squares (RSS) of method 2:  0.5659093417935869
MSE of theta_method1 on testing data:  0.00963677250182623
MSE of theta_method2 on testing data:  0.0072295106619031995
```



```
ALPHA = 1
MAX_ITER = 500
Shape of original data: (105, 3)
theta_method1: [-0.31358288  0.87211124  0.47195201]
Residual sum of squares (RSS) of method 1:  0.3747279427571582
theta_method2: [-7.15526410e+72 -6.17846964e+72 -5.82701142e+72]
Residual sum of squares (RSS) of method 2:  1.777109864567456e+148
MSE of theta_method1 on testing data:  0.00963677250182623
MSE of theta_method2 on testing data:  3.0394779017966665e+146
```



```
ALPHA = 0.1
MAX_ITER = 500000
Shape of original data: (105, 3)
theta_method1: [-0.31358288  0.87211124  0.47195201]
Residual sum of squares (RSS) of method 1:  0.3747279427571582
theta_method2: [-0.31358288  0.87211124  0.47195201]
Residual sum of squares (RSS) of method 2:  0.37472794275715815
MSE of theta_method1 on testing data:  0.00963677250182623
MSE of theta_method2 on testing data:  0.00963677250182379
```



Report:

For this assignment the method1 (calculations) appear to be a better fit for this data when looking at the RSS. The reason for this is identifiable by looking at the data above. The RSS is .37 for method 1, and varies for method 2 depending on the learning rate and the number of iterations. Method 1 is using matrix transposition to calculate the lowest possible values for theta for the cost function, its runs in milliseconds because the number of features is not large. But, when running Gradient Decent (method 2) it is not converging as close as method 1 without running many iterations, which slows down the process. Method 1 is guaranteed to be the best fit, but method 2 is not. Because method 1 runs so fast in this case it is better to use because it is guaranteed to be the best fit. Gradient decent is definitely taking longer to converge than method 1. In order to get it to converge I had to run at 500k iterations, which is an extremely long time.

What is interesting though, is the MSE tells us a different story. The MSE is .007 for gradient decent with the default learning rate and iterations, but .01 for method 1. The reason for this discrepancy is the data. The thetas for method 1 and method 2 are calculated against the training data, then the MSE is run against the test data, which is a different sample. So, what it tells me is that there is a bigger variance in the results than what the training data takes into account. The test data appears to have people that have higher SAT and higher GPAs on average than the training data. So, likely that is what is affecting the results between the 2.

Overall, I don't know that I have an answer as to which is the best one to use. The variance between the 2 methods is not huge. It appears the gradient decent method is less prone to mis-classifications in the actual data, so I would think that may be the best way to go.

1 tests I wanted to run in order to help me in the decision was I wanted to train the model using all 105 records (instead of 60), then run the MSE against it to see how that compares. Below are the results.

```
ALPHA = 0.01
MAX_ITER = 500
Shape of original data: (105, 3)
theta_method1: [-0.06234478  0.62017319  0.43647674]
Residual sum of squares (RSS) of method 1:  0.7590471383029533
theta_method2: [0.33431556 0.30427402 0.28890138]
Residual sum of squares (RSS) of method 2:  0.8865075704034706
MSE of theta_method1 on testing data:  0.007891053381741196
MSE of theta_method2 on testing data:  0.006966728983675318
```

So, it looks like the MSE and RSS follow the same trends I saw with other methods. I believe method 1 may actually be over fitting to the data which is causing the residuals are better but the MSE is worse. The data in itself has enough variance that because the gradient decent method is not fully converging, it's helping the MSE. Overall, I think we may need more data to accurately determine which route to take.