MMDT No: mmdt_091
Mentor: Nu Wai Thet

In the code, I think the gradient descent mainly functions to find the best answers.

The process of gradient descent is initiated by the parameters and then I found it challenging in deciding when to stop the gradient descent process because in the instruction we have to repeat steps 2 and 3.

The function of run_gradient_descent() first makes the predicted value of Y and then get_cost(). The more the code loops, the smaller the cost in the learning of the model. I think it can minimize the cost of the function.

For me, it is somehow difficult to understand first because I cannot comprehend how small change should be or what values of tolerance would be appropriate. I test to change the num_iterations from 400 to 1000, because I want to know what happens if the loop can function longer. At 400 iterations, gradient descent might stop functioning early if the cost becomes stable. But in 1000 iterations it takes more time.

I have also made some differences in the learning rate. I want to know what if I changed the learning rate, could there be any difference like that? This can also be related to how many steps we need to take in every iteration. I have tested with (0.0001, 0.01, and 0.1) and then the table shows the difference in changing learning rate.

When 10000 iterations connect to changing the learning rate of 0.0001, this results in the slow improvement of the model.

Learning Rate	Gradient Descent	How it functions
0.0001	It takes long between every steps	So many iterations
0.01	It can leads to lower cost	
0.1	Faster in rates	

We can conclude that a small learning rate is not good for making good progress and large learning rate can cause the model to keep the past optimal point. However, if we keep the model in the middle, the rate of (0.01) may deliver a constant rate. I realized that it is also important to choose the right learning rate.