Assignment 1 report

108062135 呂佳恩

For this assignment, I chose to use the linear regression with gradient descent approach.

The following is my code:

```
for x in input_datalist:
  tmp_date = x[0]
  tmp = np.append([], np.char.split(x[0], '/'))

  if(len(tmp[0][1])<2):
      tmp[0][1] = '0'+ tmp[0][1]
  if(len(tmp[0][2])<2):
      tmp[0][2] = '0'+ tmp[0][2]
  x[0] = tmp[0][0] + tmp[0][1] + tmp[0][2]
  if(int(x[0])>=20211015):
      date = np.append(date, tmp_date)
      testing_dataX = np.append(testing_dataX, x[1].replace(',',''))
  else:
      training_dataX = np.append(training_dataX,x[1].replace(',',''))
      training_dataY = np.append(training_dataY,x[2].replace(',',''))

testing_dataX = testing_dataX.astype(float)
training_dataX = training_dataX.astype(float)
training_dataY = training_dataY.astype(float)
return()
```

This is the method I used to process the data. I first removed all the '/' from the data. Since it would make it hard for me to compare the dates. Since the date is in a increasing order, another approach would be to use a flag, and when the date 2021/10/15 is reached. Raise the flag. However, if the date is not in a increasing order, this would not work.

I then padded 0s for the months and dates with under two digits, so that January would be 01 and not 1. This is also for comparing the dates.

I then appended the dates. 2021/1/3 would be 20210103. This made it easy for me to compare the dates.

```
def Regression(X,Y, learning_rate, iterations):
  alpha = learning_rate
  m = len(Y)
  cost_list = []
  theta = np.zeros(2)
  for i in range(iterations):
    hx = np.dot(X, theta)
    cost = (1/(2*m))*np.sum(np.square((hx-Y)))
    d_theta = (1/m)*np.dot(X.T, hx-Y)
    theta = theta - alpha*d_theta
    cost_list.append(cost)
  return theta, cost_list
```

This is my model for training. The function receives four variables. The training data X and the training data Y, with the learning rate and iterations we want to train. Then we apply the functions of gradient descent and cost. I store the cost into a cost list so that I could be sure that the training is converging. The learning rate I used was 0.000005 with 1000000 iterations. I found out that a learning rate around 0.0001 would cause it to diverge. Therefore, I went with a learning rate I found that would converge and used more iterations for the training.

I used the data from 9/15 to 10/14 to validate the result of my training, the result were quite similar to the market stock prices of TSMC.