FPT QUY NHON UNIVERSITY
ARTIFICIAL INTELLIGENCE

FPT UNIVERSITY

MAI391

ASSIGNMENT 03

Topic:

LINEAR REGRESSION

Class:	AI18C	
Students:	Nguyen Tan Thang	QE180019
	Le Quoc Chinh	QE170250
	Cao Ngoc Y	QE180047
	Phan Thanh Duy	QE180090
	Nguyen Thi Tu Anh	QE180073
	Huynh Tan Tai	SE173225

Quy Nhon, 7-2024

Contents

1 Problem Statement **2**

2 Data Source and Prepare Data **2**

2.1 Data Source 2

2.2 Prepare Data 2

3 Descriptive Statistics **2**

4 Gradient Descent **3**

4.1 Normalize Data 3

4.2 Compute Cost Function J(θ) 3

4.3 Gradient Descent 3

4.4 Visualizing 4

4.4.1 Visualizing Cost Function J(θ) 4

4.4.2 Plotting the convergence 4

4.5 Training Data with Univariate Linear Regression Fit (Best Fit Line) 5

4.5.1 Linear Regression Fit 5

4.5.2 Model Evaluation 5

5 Conclusion **6**

1. Problem Statement

- Implementing Gradient Descent and Linear Regression using Python with Collected Data.

2. Data Source and Prepare Data

2.1 Data Source

- Data link:

github.com/thangthewinner/MAI391_AS03/blob/main/data/stock21_clean.csv

- The dataset contains information related to stocks currently listed on the Vietnam stock exchange in 2021.

2.2 Prepare Data

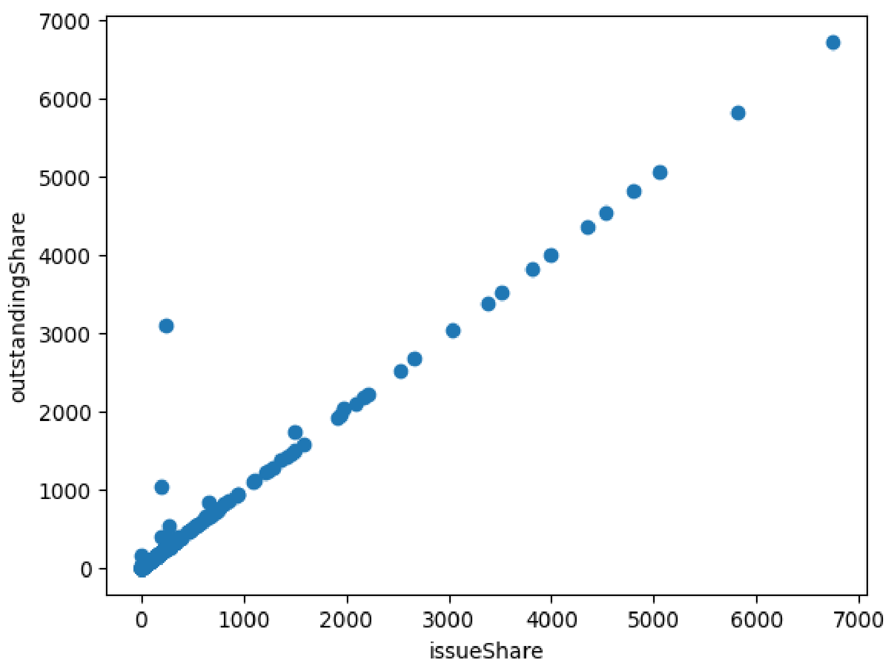
- After observing and performing some statistics on the data, We discovered that Issue Share and Outstanding Share have a linear relationship with a correlation of up to 0.98. Therefore, we decided to use the two attributes for this task.

3. Descriptive Statistics

- Issue Share: The number of shares the company has issued since its establishment until now.

- Outstanding Share: The number of shares currently outstanding in the market.

- Scatter plot between Issue Share and Outstanding Share:



- Looking at the scatter plot, we see that as the issue share increases, the outstanding share also increases. This makes it very suitable for creating a linear regression model.

4. Gradient Descent

4.1 Normalize Data

- From the scatter plot we can see that issueShare and outstandingShare all in range from 0 to 7000. This will make prediction difficult and time-consuming due to excessively large values. Therefore, what we need to do is normalize it to be within the range of -1 to 1.

- Normalization formula:

$X_{norm} = \frac{X-\mu}{\sigma}$ - Through this formula, we have created a function to normalize the data.

4.2 Compute Cost Function J(θ)

- The formula of the cost function for linear regression is given by:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left((\theta_0 + \theta_1 x^{(i)}) - y^{(i)} \right)^2$$

- Through this formula, we have created a function to compute the cost of the data.

4.3 Gradient Descent

- The update formula for parameters θ in the gradient descent algorithm is:

$$\theta := \theta - \alpha \nabla_{\theta} J(\theta)$$

- Through this formula, we have created a function to update formula for parameters θ of the data.

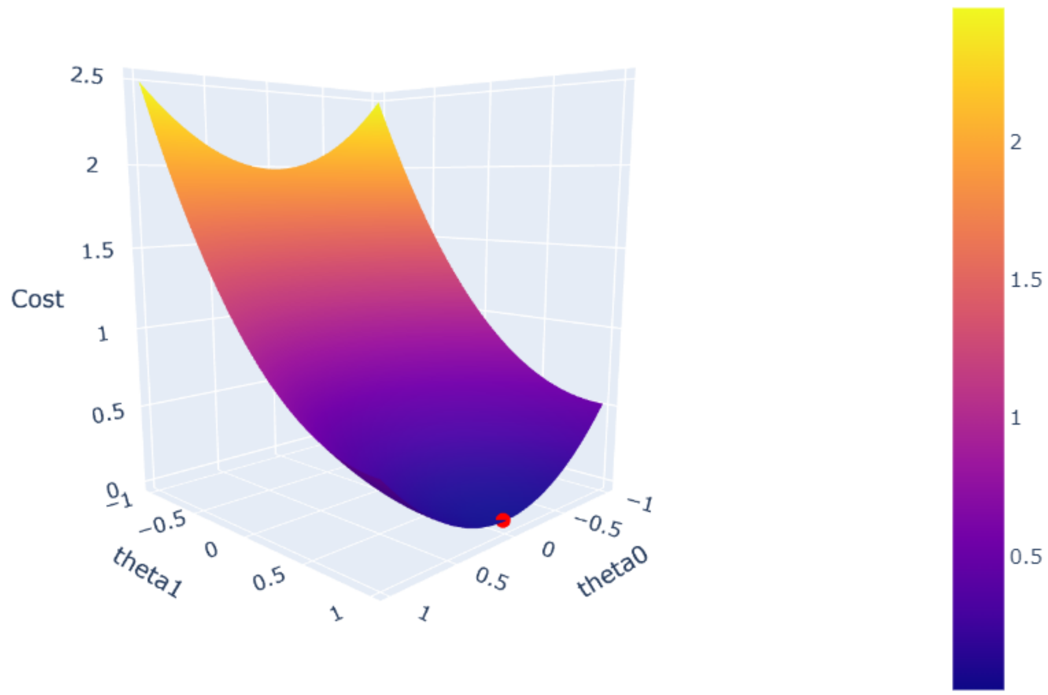
- Test gradient descent for 1500 iterations with a learning rate (alpha) of 0.01:

```
Iteration: 0, Cost: 0.01598548726633169
Iteration: 100, Cost: 0.01598548726633169
Iteration: 200, Cost: 0.01598548726633169
Iteration: 300, Cost: 0.01598548726633169
Iteration: 400, Cost: 0.01598548726633169
Iteration: 500, Cost: 0.01598548726633169
Iteration: 600, Cost: 0.01598548726633169
Iteration: 700, Cost: 0.01598548726633169
Iteration: 800, Cost: 0.01598548726633169
Iteration: 900, Cost: 0.01598548726633169
Iteration: 1000, Cost: 0.01598548726633169
Iteration: 1100, Cost: 0.01598548726633169
Iteration: 1200, Cost: 0.01598548726633169
Iteration: 1300, Cost: 0.01598548726633169
Iteration: 1400, Cost: 0.01598548726633169
theta0:  2.782399344117694e-17
theta1:  0.9838846606525207
```

4.4 Visualizing

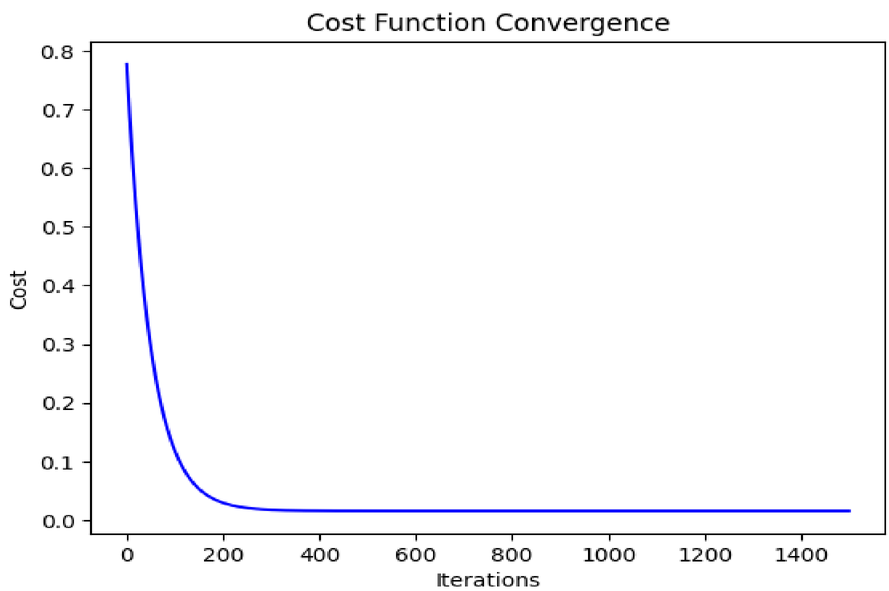
4.4.1 Visualizing Cost Function J(θ)

- This is a 3D representation of the cost function with a 3-dimensional grid of θ_0, θ_1, and $J(\theta)$ values. The red point in the plot represents the point where the cost function achieves its minimum value after applying the gradient descent algorithm.



4.4.2 Plotting the convergence

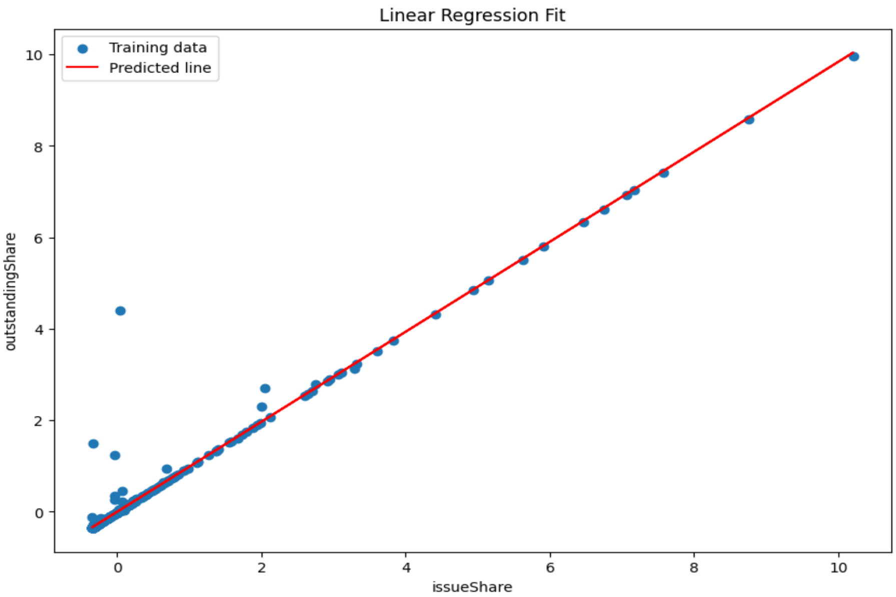
- This is a plot showing the convergence of the cost function.



4.5 Training Data with Univariate Linear Regression Fit (Best Fit Line)

4.5.1 Linear Regression Fit

- This is the optimal regression line through the training data.



4.5.2 Model Evaluation

- Actual Values and Predicted Values:

	Actual	Predicted
627	32.5	27.056821
367	10.0	9.838844
113	200.0	196.776881
396	361.5	359.412972
338	12.4	12.200167
...	...	...
226	13.1	12.888886
124	7.9	7.871075
617	16.0	15.742151
282	4.8	4.722645
382	1885.2	1854.818876

160 rows × 2 columns

- MAPE evaluates the average percentage deviation of predictions from actual values.

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_{true,i} - y_{pred,i}}{y_{true,i}} \right| \times 100$$

- In which:

$y_{true,i}$ is the actual value of sample i.
$y_{pred,i}$ is the predicted value of sample i.
n is the number of samples.

- Applying MAPE to the linear regression model, MAPE = 4.65% means that the model's predictions deviate by approximately 4.65% from the actual values on the test data.

5. Conclusion

- Through this exercise, we have learned how to implement a linear regression model in practice. This helps us gain a deeper understanding of the linear regression model.