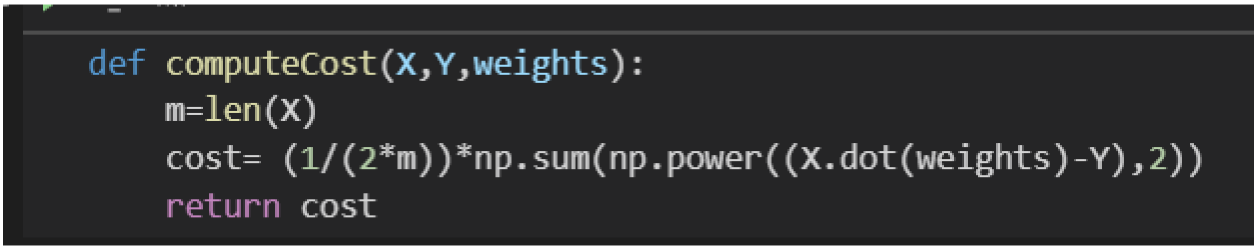
Build Model

- **Used This Cost Function Equation**

$$J(w) = \frac{1}{2m}\sum_{i}^{m}(h(x^i) - y^i)^2$$



```
def computeCost(X,Y,weights):
    m=len(X)
    cost= (1/(2*m))*np.sum(np.power((X.dot(weights)-Y),2))
    return cost
```

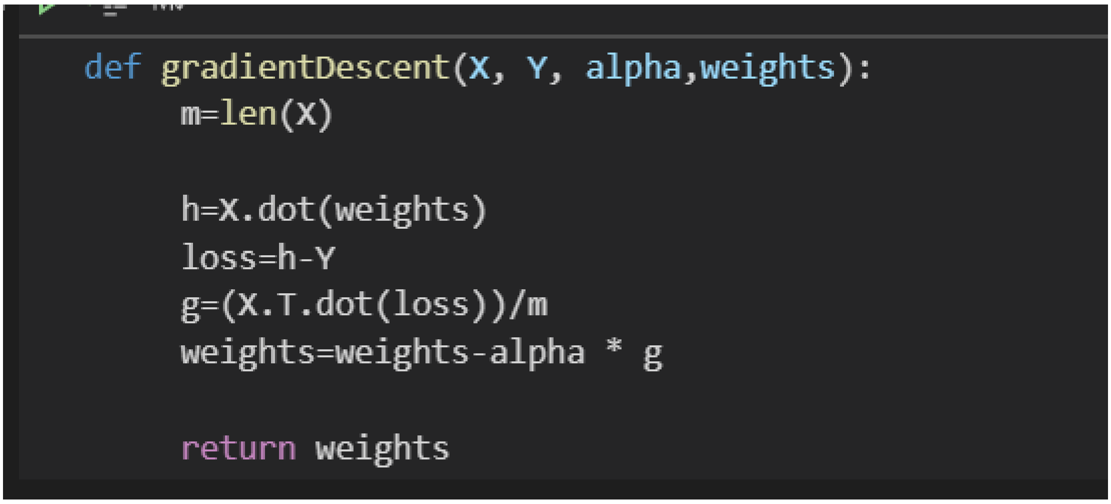
- **Gradient Decent**

$$h(x^i) = \beta^T X$$

$$Loss = h(x^i) - y^i$$

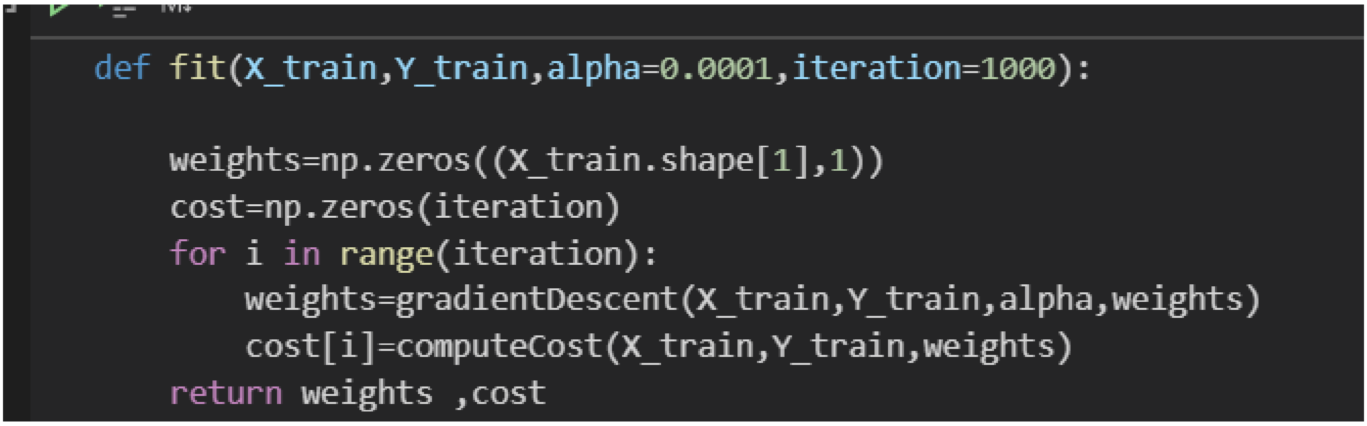
$$g = (h(x^i) - y^i)x^i$$

$$\theta = \theta - \alpha \times g$$



```
def gradientDescent(X, Y, alpha,weights):
    m=len(X)

    h=X.dot(weights)
    loss=h-Y
    g=(X.T.dot(loss))/m
    weights=weights-alpha * g

    return weights
```

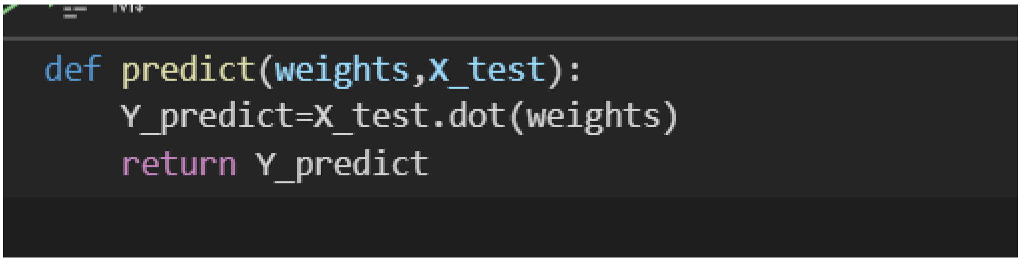
- **Fit Function**



```python
def fit(X_train,Y_train,alpha=0.0001,iteration=1000):

    weights=np.zeros((X_train.shape[1],1))
    cost=np.zeros(iteration)
    for i in range(iteration):
        weights=gradientDescent(X_train,Y_train,alpha,weights)
        cost[i]=computeCost(X_train,Y_train,weights)
    return weights ,cost
```

- **Predict Function**

$$h(x^i) = \beta^T X$$



```python
def predict(weights,X_test):
    Y_predict=X_test.dot(weights)
    return Y_predict
```

- **Evaluate Performance**

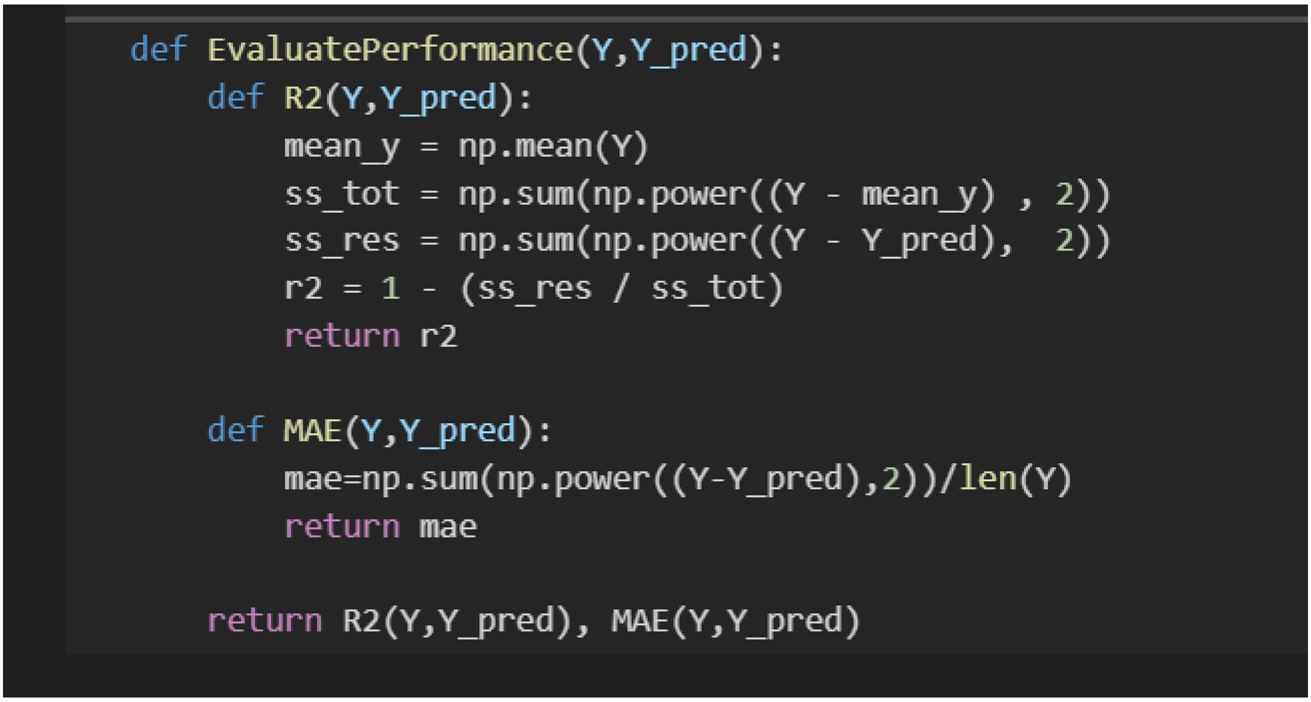
> `I Found that these metrics are the best to evaluate my model and tried to use the two metrics just to practice more`

1. by Mean absolute error (MAE)

$$MAE = \frac{\sum_i^n (Y_{actual} - Y_{pred})^2}{n}$$

2. by Coefficient of Determination or R^2

$$R_2 = \frac{SSR}{SST} = \frac{\sum_i^n (Y_{actual} - Y_{mean})^2}{\sum_i^n (Y_{pred} - Y_{mean})^2}$$

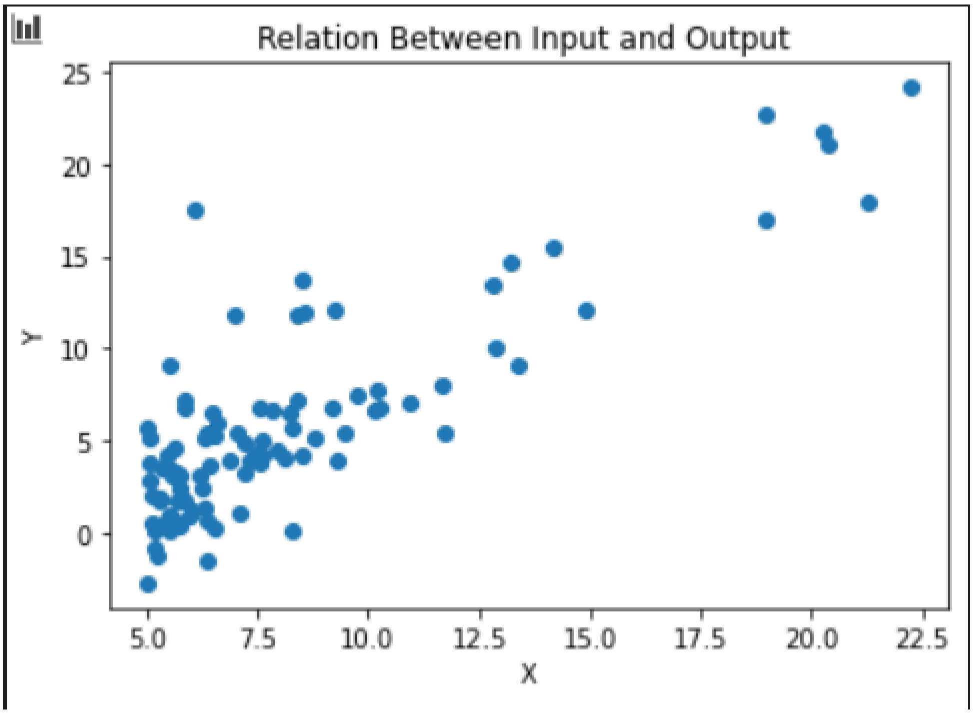


```
def EvaluatePerformance(Y,Y_pred):
    def R2(Y,Y_pred):
        mean_y = np.mean(Y)
        ss_tot = np.sum(np.power((Y - mean_y) , 2))
        ss_res = np.sum(np.power((Y - Y_pred),  2))
        r2 = 1 - (ss_res / ss_tot)
        return r2

    def MAE(Y,Y_pred):
        mae=np.sum(np.power((Y-Y_pred),2))/len(Y)
        return mae

    return R2(Y,Y_pred), MAE(Y,Y_pred)
```

Explore Data

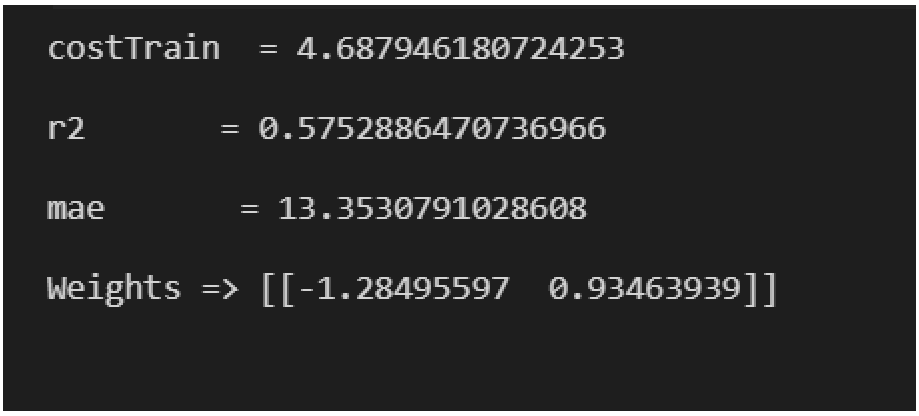
- UniVariate DataSet

- **Plot Data X and Y**



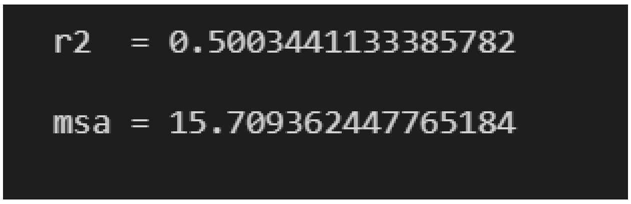
- **Result of My model**

> - r2 = > Coefficient of Determination
> - mae = > Mean absolute error



```
costTrain  = 4.687946180724253

r2       = 0.5752886470736966

mae       = 13.3530791028608

Weights => [[-1.28495597  0.93463939]]
```

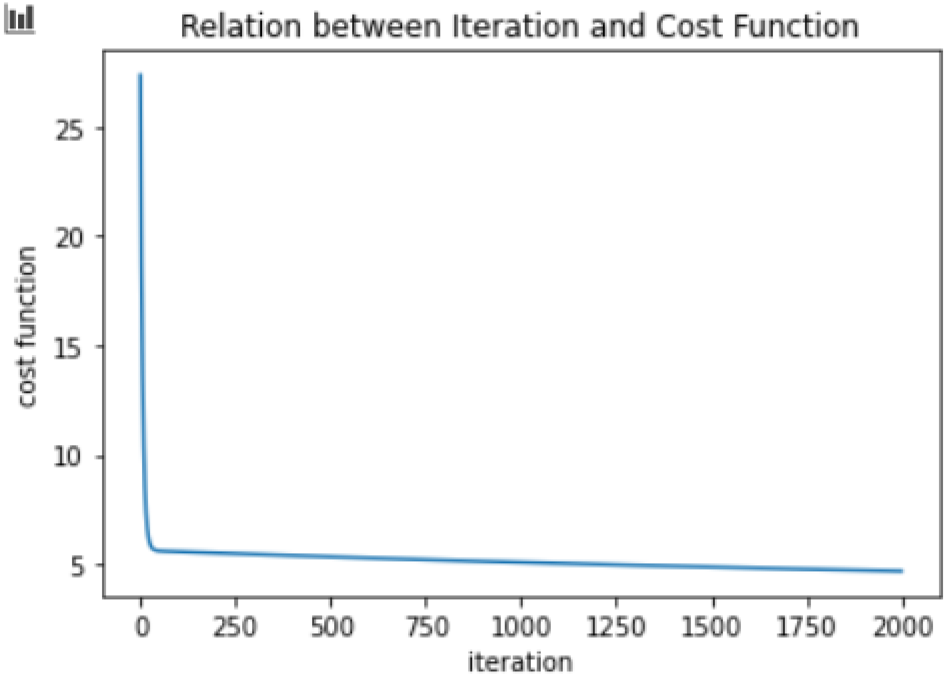
- **Result When used Sicit_learn Linear Regression Model**



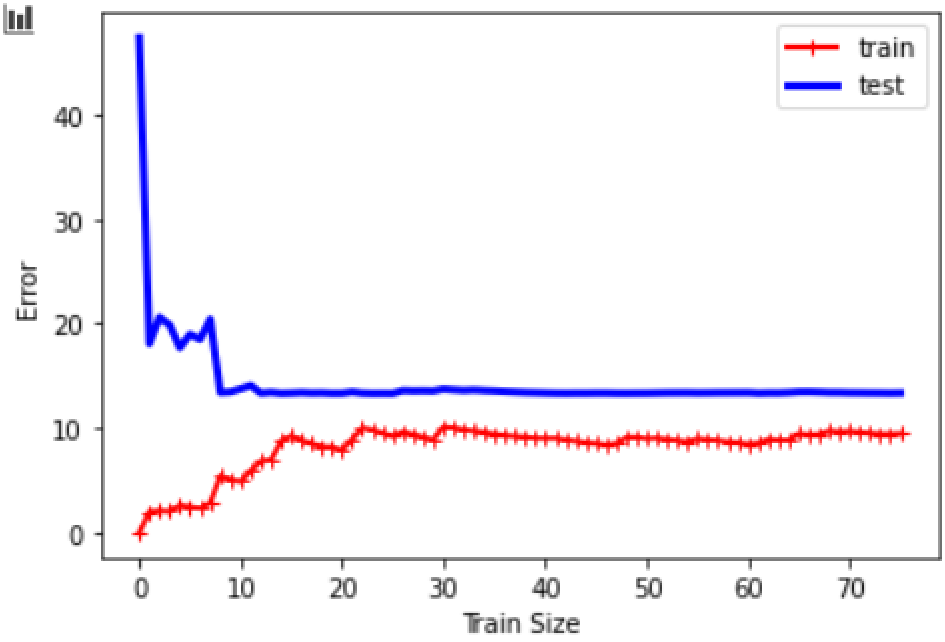
```
r2  = 0.5003441133385782

msa = 15.709362447765184
```

- **I Tried to see at any iteration must stop because the cost function doesn't decrease more**

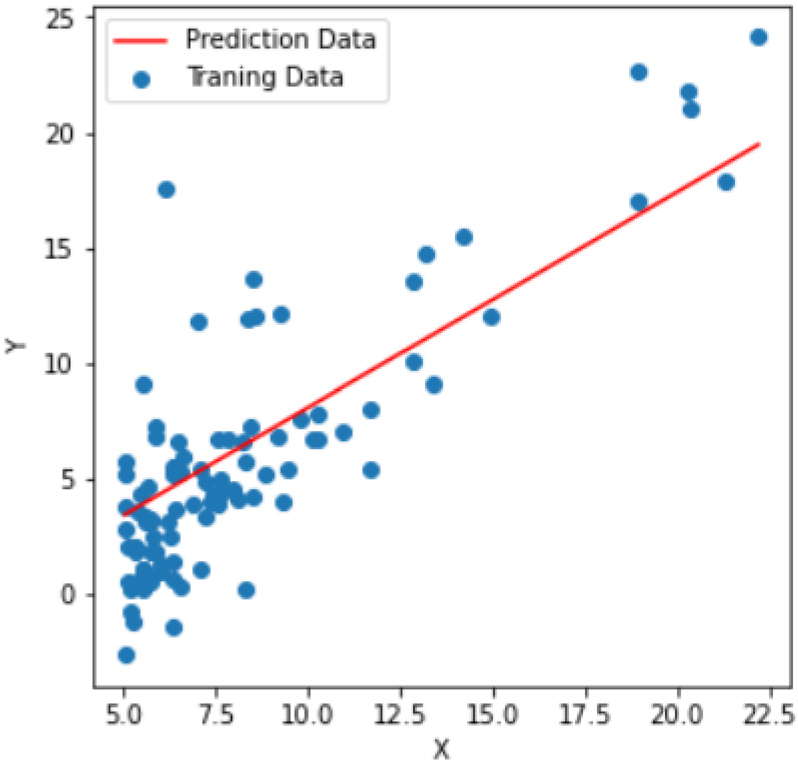
> 2000 iteration sounds good



- **I Tried to Visualize Error of Train and Test data to Compare between them and see the Performance of my model**



- **Plot best fit line on data**



- MuliVariate DataSet

> `at the first, I didn't do scaling but it gives` **nan** `Values in the prediction of output and subsequently in Metrics to evaluate the performance`

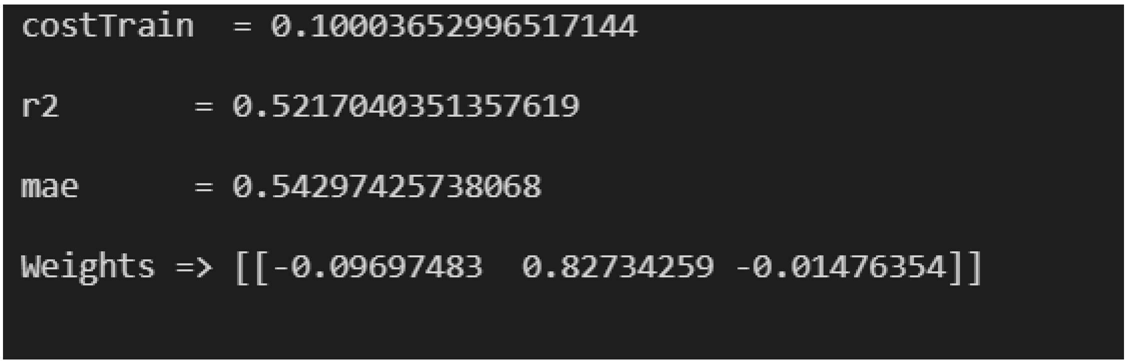
- **Feature Scaling**

> `Due to range of each Features are differ from each other we must do Standardization`

$$X = \frac{X - X_{mean}}{\sigma}$$

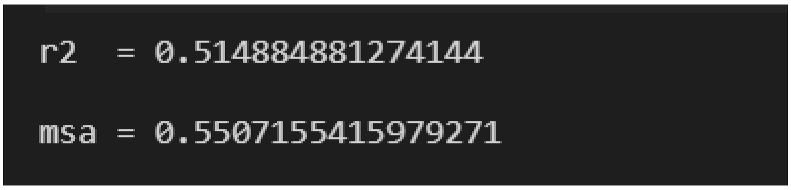
- **Result of My model**

> - r2 = > Coefficient of Determination
> - mae = > Mean absolute error



```
costTrain  = 0.10003652996517144

r2        = 0.5217040351357619

mae       = 0.54297425738068

Weights => [[-0.09697483  0.82734259 -0.01476354]]
```

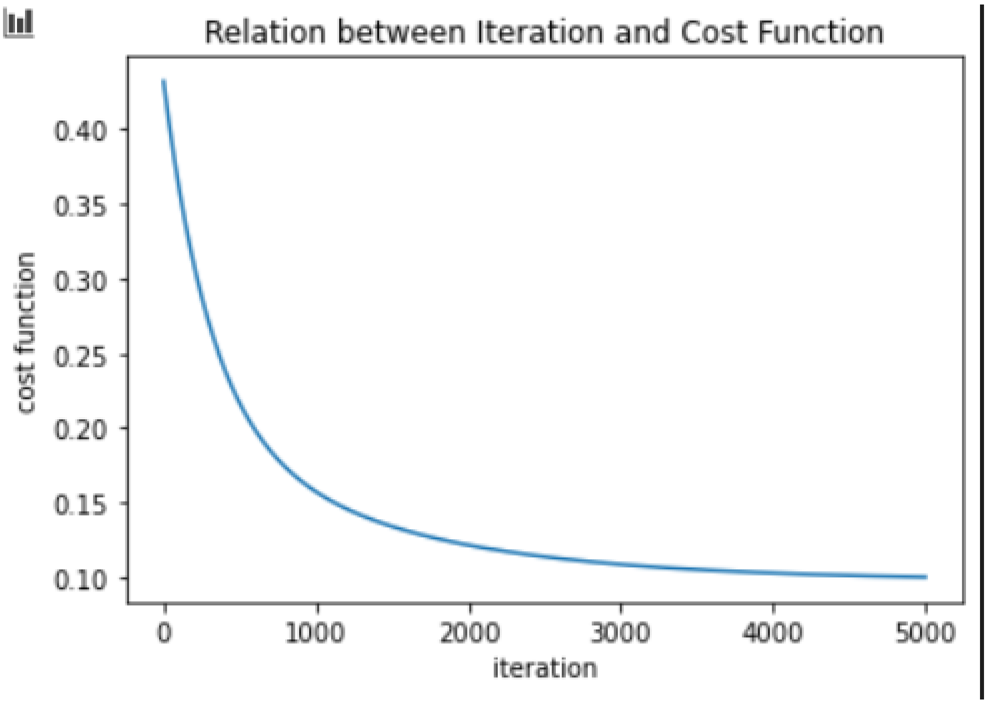
- **Result When used Sicit_learn Linear Regression Model**



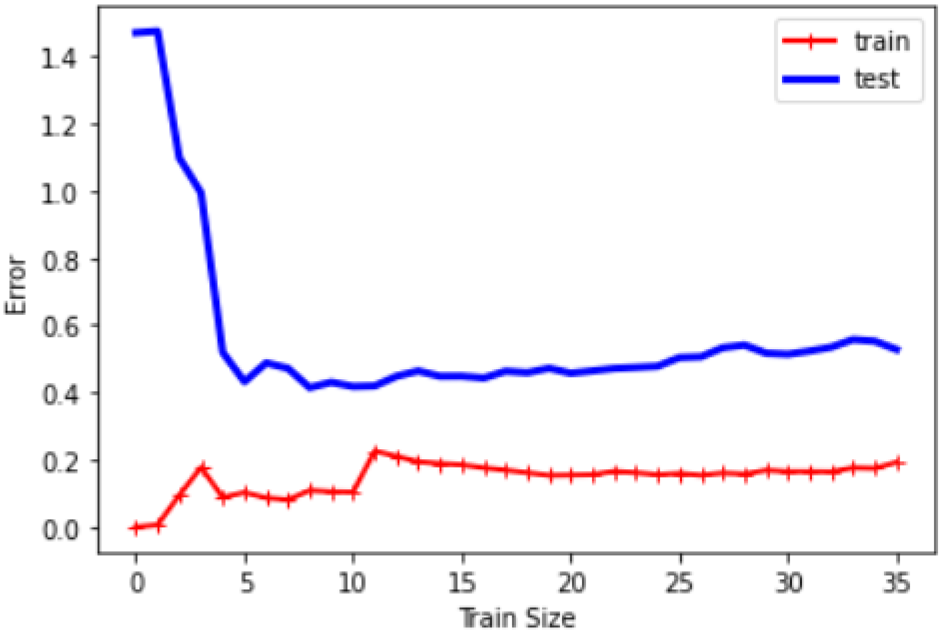
```
r2  = 0.514884881274144

msa = 0.5507155415979271
```

- **I Tried to see at any iteration must stop because the cost function doesn't decrease more**

> 5000 iteration sounds good



- **I Tried to Visualize Error of Train and Test data to Compare between them and see the Performance of my model**


train
test
Error
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35
Train Size