University of
Asia Pacific

University of Asia Pacific

Department of Computer Science & Engineering

Project : 02

Course Title: Artificial Intelligence and Expert Systems LAB

Course Title: CSE-404

Project Name: Implementation of Multivariable Linear Regression with SK-Learn and without SK-Learn

Submitted To

Dr. Nasima Begum

Associate Professor. Dept. of CSE

Submitted By

Md Shamaun nabi

Reg:18201050

Sec: B1

Dept.of CSE

Problem Statement

Implementation of Multivariable Linear Regression with SK-Learn and without SK-Learn

Introduction

Multivariate Regression is a method used to measure the degree at which more than one independent variable (predictors) and more than one dependent variable (responses), are linearly related

Problem Description

Multivariable linear regression is a **supervised machine learning algorithm** involving multiple data variables for analytics. MRL is an extension of multiple regression with one dependent variable and multiple independent variables. We try to predict the output based on the number of independent variables.

Dataset

Dataset name : Motorcycle Details Dataset
Dataset URL :
https://www.kaggle.com/prasadperera/the-boston-housing-dataset/data?select=housing.csv

The dataset has 4 columns:

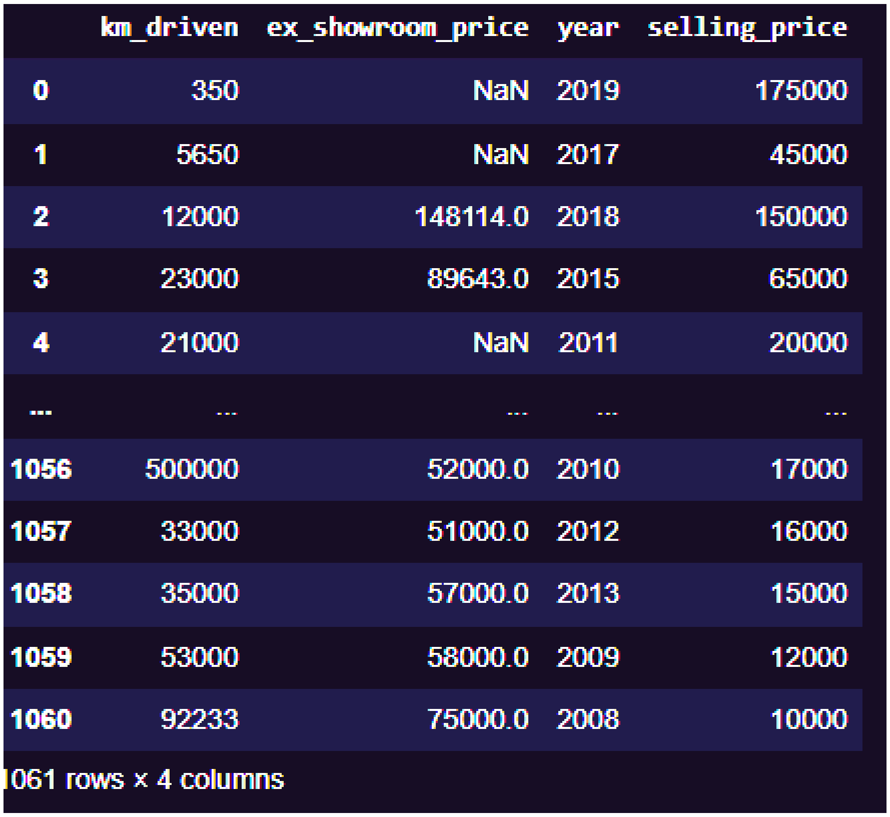
- **selling_price**
- **ex_showroom_price**
- **Year**
- **Km_driven**

Input Feature For This Project

- **Variable (x1)** : km_driven
- **Variable (x2)** : ex_Showroom_price
- **Variable (x3)** : Year

Output

❖ **Output(Y):** selling_price



	km_driven	ex_showroom_price	year	selling_price
0	350	NaN	2019	175000
1	5650	NaN	2017	45000
2	12000	148114.0	2018	150000
3	23000	89643.0	2015	65000
4	21000	NaN	2011	20000
...	...	...	...	...
1056	500000	52000.0	2010	17000
1057	33000	51000.0	2012	16000
1058	35000	57000.0	2013	15000
1059	53000	58000.0	2009	12000
1060	92233	75000.0	2008	10000

1061 rows × 4 columns

Here in the dataset the **Selling Price** is a dependent feature and the rest of the columns are independent features plotting the independent variable vs department variable to see how independent variables affect the independent variable.

VISUALIZATION:

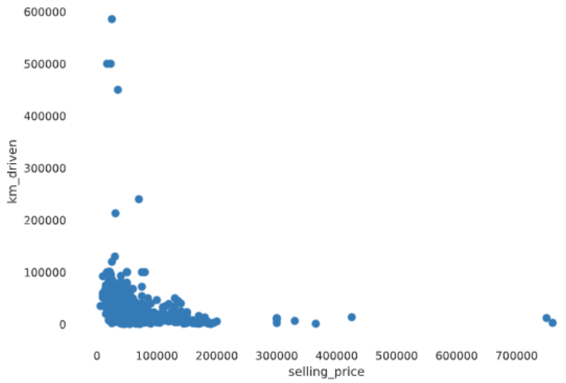


Fig: Selling_price vs Km_driven

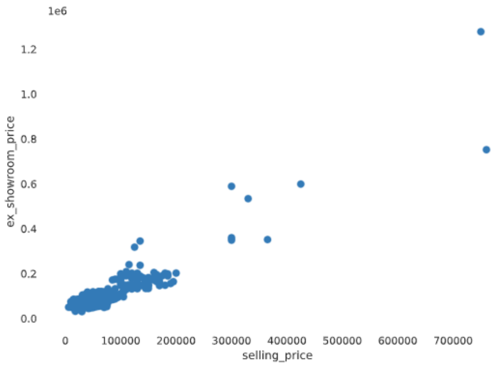


Fig: Selling_price vs Ex_Showroom Price

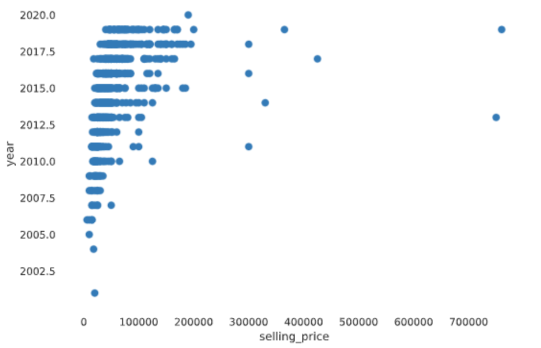


Fig: Selling_price vs Year

Step of Linear Regression

1. Consider the Dataset:

The first step is dataset considering for unknown feature to find the corresponding output **Y**

2. Parameter Initialization:

For Parameter Initialization there is formula :

1+no.F

Where the 1 is Called the **Base Parameter** and (no.F) refers **Number of Feature**

If dataset have 3 Column than the feature input will be 2 , 1 Column is reject because that is the output

In my case there are a total of 4 columns. and if I reject 1 column for output than number of feature will be **n = 3**
and apply the formula **1+nf = 1+3 = 4**

So parameter will be 4

3. Hypothesis Function:

Hypothesis Function is to find out predicted output from the actual output.

Hypothesis Formula :

For Multiple Linear Regression the hypothesis function will be :

$$\mathbf{y_{predict}} = \sum_{i=0}^{n} weight(i) * X(i) + bias$$

N = number of feature

So in my case applying this formula the predicted and actual output are not the same. That's why I can say that there is some mismatch or error.

4. Cost Function:

Now i want to see the differences between the Actual Output which i get from the hypothesis function and predicted output
That's why here is the **Cost function**.

For cost function, we are using Means Squared Error. The formula for the MSE

$$Cost = \frac{1}{m}\sum_{i=0}^{m} (y_{predict}(i) - y_{true}(i))^{2}$$

M = number of samples

5. Gradient Descent:

These are the most Important Steps. To minimizing The Error.To minimizing error means find out the suitable parameter value for model.so here i want to minimize the cost that's why i update the parameter Updating the weights and bias value we are using Gradient Descent(GD)

$$weights_{i} = weights_{i} - \frac{\alpha}{m} * \sum_{i=0}^{m} [(y_{predict}(i) - y_{true}(i)) * X(i)]$$

$$bias = bias - \frac{\alpha}{m} * \sum_{i=0}^{m} [(y_{predict}(i) - y_{true}(i))]$$

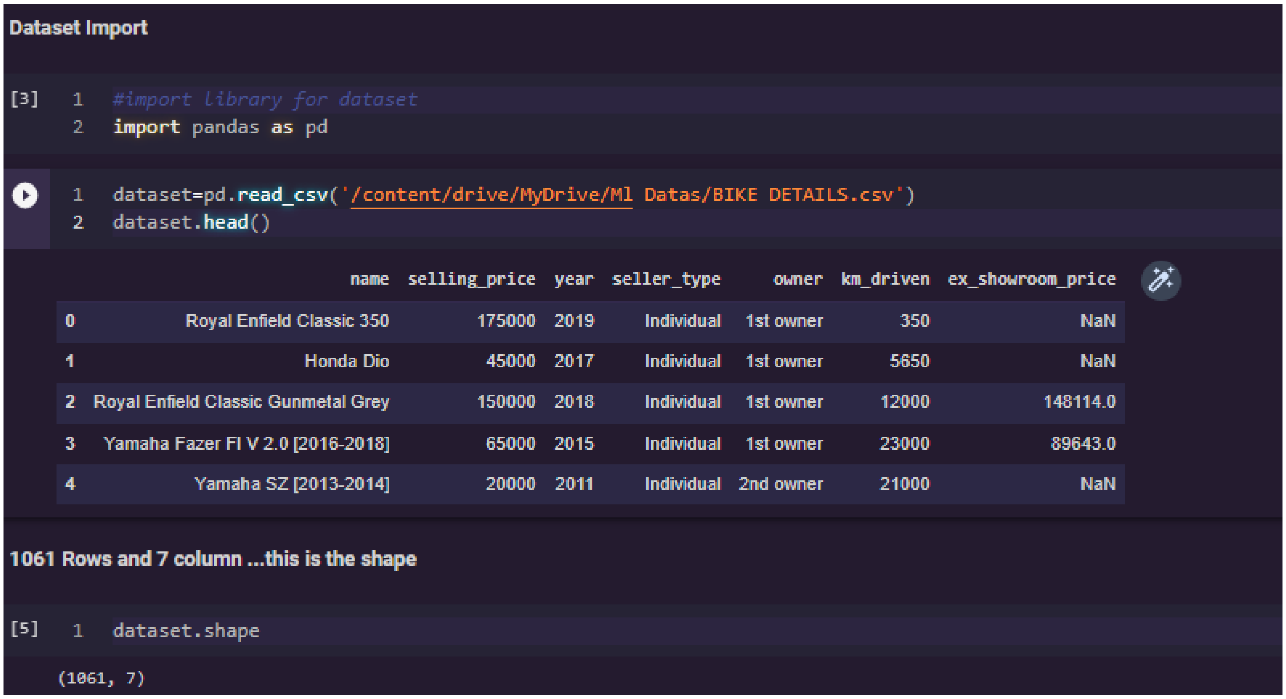
α = Learning Rate

Tools and Language:

- Tools:VSCode, excel, Microsoft word document
- Language: python
- Modules: pandas, sklearn.linear_model, sklearn.metrics , matplotlib.

Implementation

Dataset Upload:



Dataset Import

[3]

```
#import library for dataset
import pandas as pd
```

```
dataset=pd.read_csv('/content/drive/MyDrive/Ml Datas/BIKE DETAILS.csv')
dataset.head()
```

	name	selling_price	year	seller_type	owner	km_driven	ex_showroom_price
0	Royal Enfield Classic 350	175000	2019	Individual	1st owner	350	NaN
1	Honda Dio	45000	2017	Individual	1st owner	5650	NaN
2	Royal Enfield Classic Gunmetal Grey	150000	2018	Individual	1st owner	12000	148114.0
3	Yamaha Fazer FI V 2.0 [2016-2018]	65000	2015	Individual	1st owner	23000	89643.0
4	Yamaha SZ [2013-2014]	20000	2011	Individual	2nd owner	21000	NaN

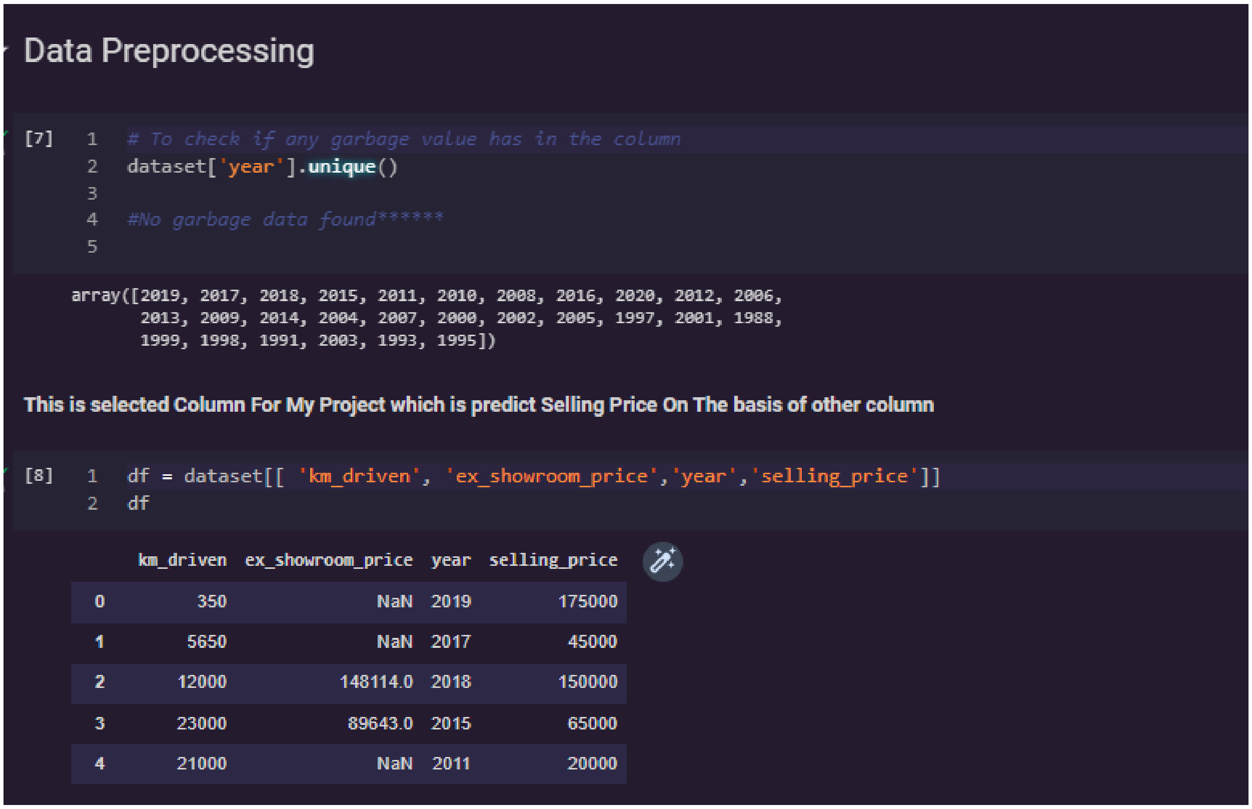
1061 Rows and 7 column ...this is the shape

[5]

```
dataset.shape
```

```
(1061, 7)
```

Data Preprocessing:



Data Preprocessing

```
# To check if any garbage value has in the column
dataset['year'].unique()

#No garbage data found******

```

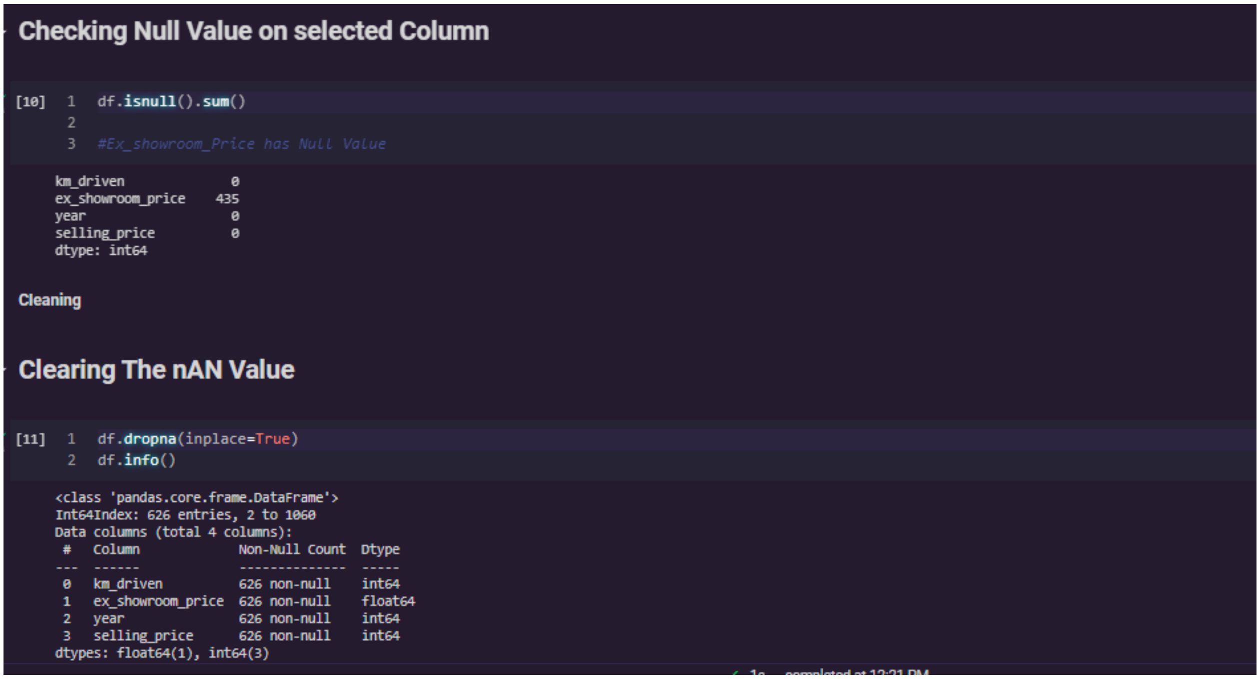
```
array([2019, 2017, 2018, 2015, 2011, 2010, 2008, 2016, 2020, 2012, 2006,
       2013, 2009, 2014, 2004, 2007, 2000, 2002, 2005, 1997, 2001, 1988,
       1999, 1998, 1991, 2003, 1993, 1995])
```

This is selected Column For My Project which is predict Selling Price On The basis of other column

```
df = dataset[[ 'km_driven', 'ex_showroom_price','year','selling_price']]
df
```

	km_driven	ex_showroom_price	year	selling_price
0	350	NaN	2019	175000
1	5650	NaN	2017	45000
2	12000	148114.0	2018	150000
3	23000	89643.0	2015	65000
4	21000	NaN	2011	20000

Data Cleaning:



Checking Null Value on selected Column

```
df.isnull().sum()

#Ex_showroom_Price has Null Value
```

```
km_driven              0
ex_showroom_price    435
year                   0
selling_price          0
dtype: int64
```

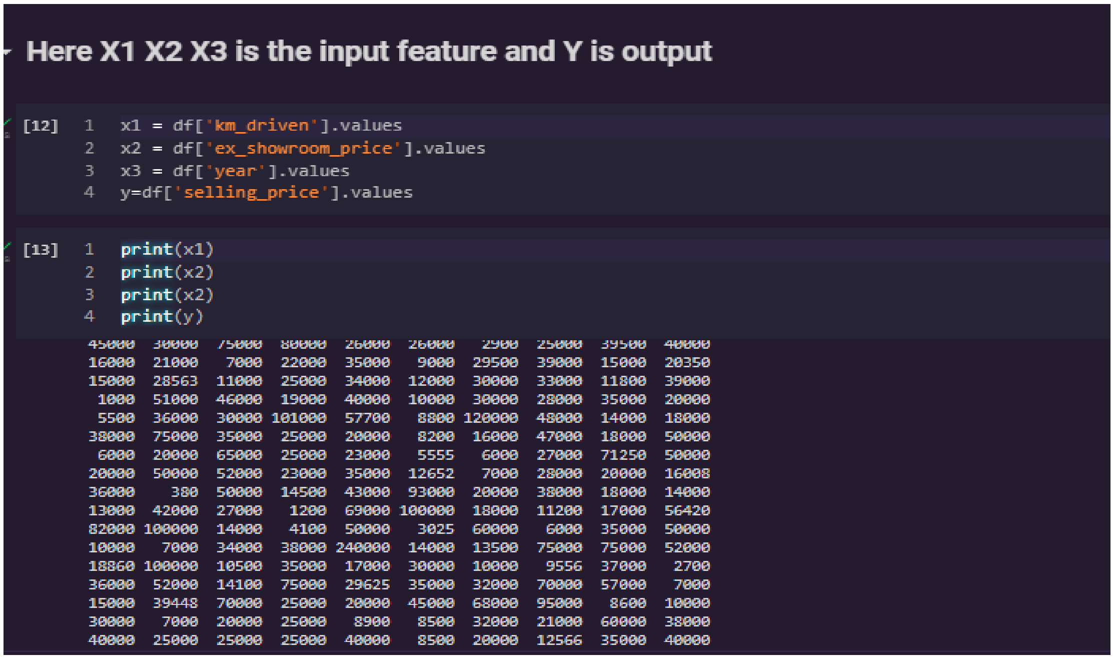
Cleaning

Clearing The nAN Value

```
df.dropna(inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 626 entries, 2 to 1060
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   km_driven          626 non-null    int64  
 1   ex_showroom_price  626 non-null    float64
 2   year               626 non-null    int64  
 3   selling_price      626 non-null    int64  
dtypes: float64(1), int64(3)
```

Feature Input and Output Selection:



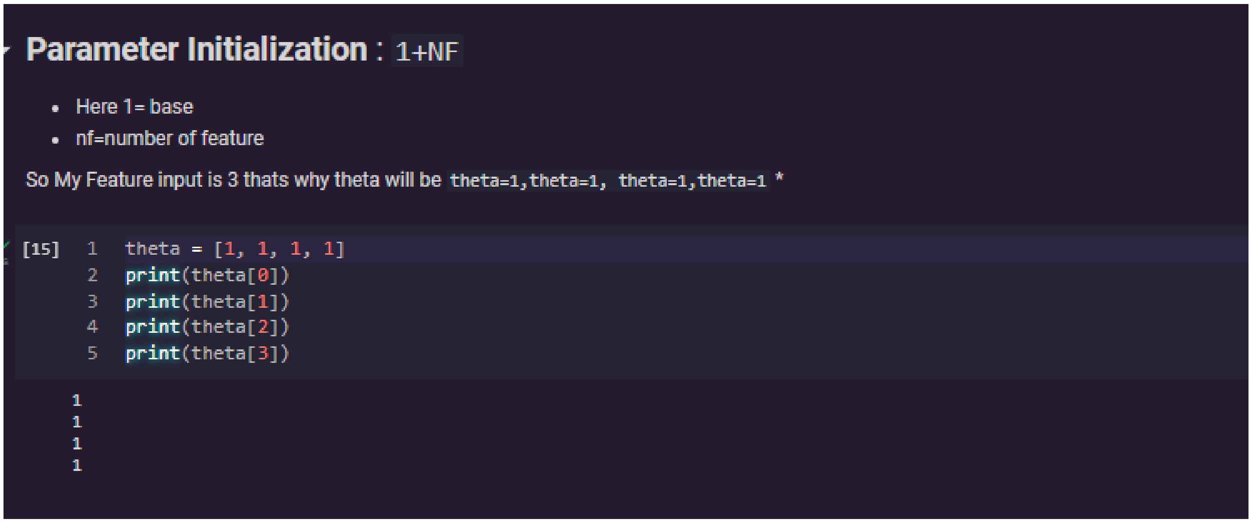
Here X1 X2 X3 is the input feature and Y is output

[12]
```
x1 = df['km_driven'].values
x2 = df['ex_showroom_price'].values
x3 = df['year'].values
y=df['selling_price'].values
```

[13]
```
print(x1)
print(x2)
print(x2)
print(y)
```

```
 45000  30000  75000  80000  26000  26000   2900  25000  39500  40000
 16000  21000   7000  22000  35000   9000  29500  39000  15000  20350
 15000  28563  11000  25000  34000  12000  30000  33000  11800  39000
  1000  51000  46000  19000  40000  10000  30000  28000  35000  20000
  5500  36000  30000 101000  57700   8800 120000  48000  14000  18000
 38000  75000  35000  25000  20000   8200  16000  47000  18000  50000
  6000  20000  65000  25000  23000   5555   6000  27000  71250  50000
 20000  50000  52000  23000  35000  12652   7000  28000  20000  16008
 36000    380  50000  14500  43000  93000  20000  38000  18000  14000
 13000  42000  27000   1200  69000 100000  18000  11200  17000  56420
 82000 100000  14000   4100  50000   3025  60000   6000  35000  50000
 10000   7000  34000  38000 240000  14000  13500  75000  75000  52000
 18860 100000  10500  35000  17000  30000  10000   9556  37000   2700
 36000  52000  14100  75000  29625  35000  32000  70000  57000   7000
 15000  39448  70000  25000  20000  45000  68000  95000   8600  10000
 30000   7000  20000  25000   8900   8500  32000  21000  60000  38000
 40000  25000  25000  25000  40000   8500  20000  12566  35000  40000
```

Parameter Initialization:



Parameter Initialization : `1+NF`

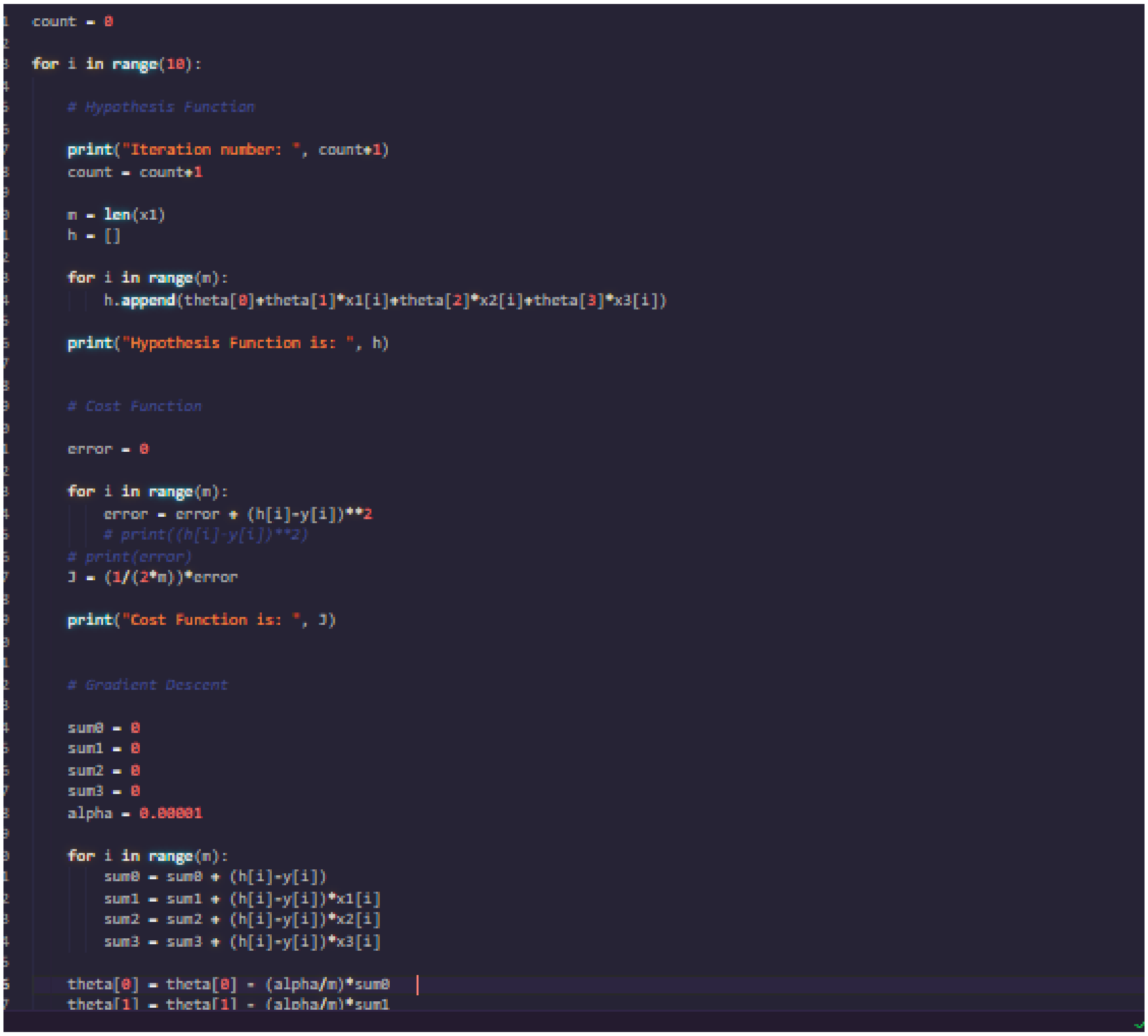
- Here 1= base
- nf=number of feature

So My Feature input is 3 thats why theta will be `theta=1,theta=1,` `theta=1,theta=1` *

[15]
```
theta = [1, 1, 1, 1]
print(theta[0])
print(theta[1])
print(theta[2])
print(theta[3])
```

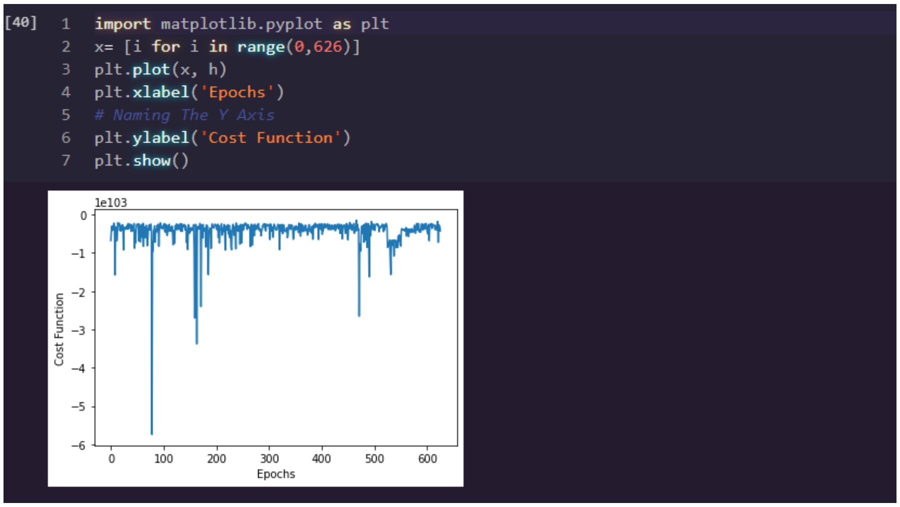
```
1
1
1
1
```

Function Calculation:



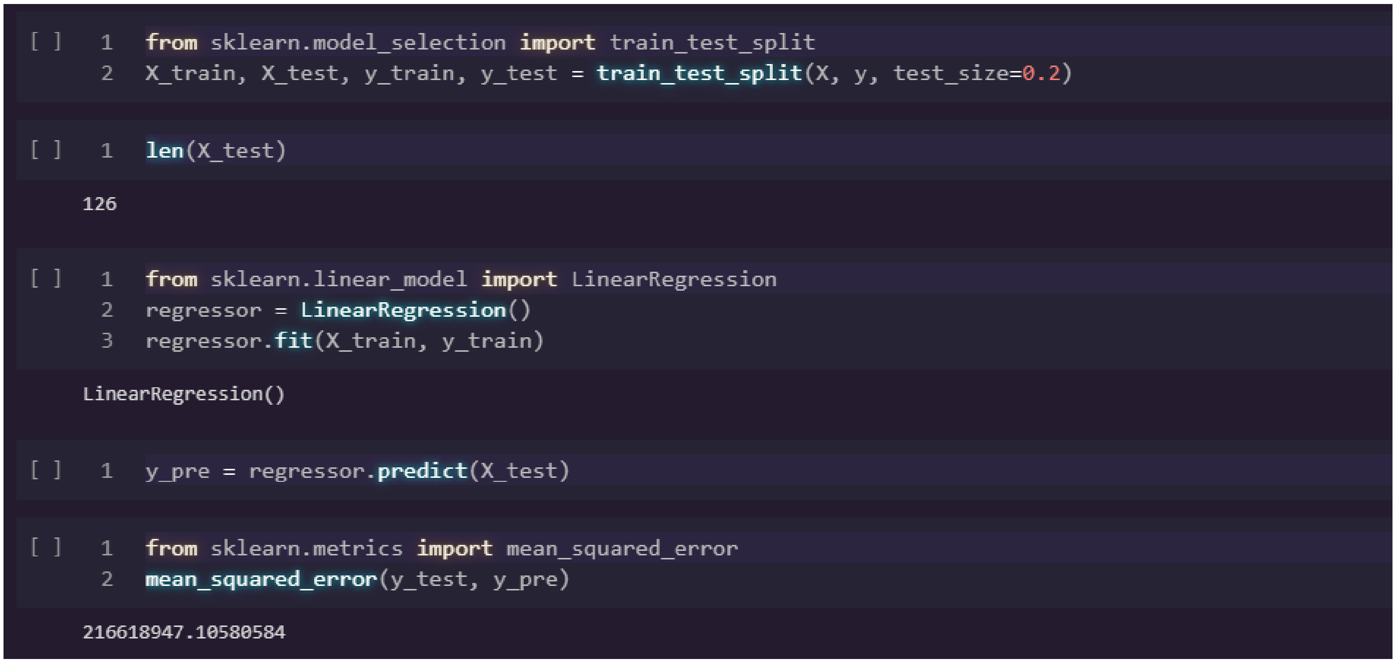
```python
count = 0

for i in range(10):

    # Hypothesis Function

    print("Iteration number: ", count+1)
    count = count+1

    m = len(x1)
    h = []

    for i in range(m):
        h.append(theta[0]+theta[1]*x1[i]+theta[2]*x2[i]+theta[3]*x3[i])

    print("Hypothesis Function is: ", h)


    # Cost Function

    error = 0

    for i in range(m):
        error = error + (h[i]-y[i])**2
        # print((h[i]-y[i])**2)
    # print(error)
    J = (1/(2*m))*error

    print("Cost Function is: ", J)


    # Gradient Descent

    sum0 = 0
    sum1 = 0
    sum2 = 0
    sum3 = 0
    alpha = 0.00001

    for i in range(m):
        sum0 = sum0 + (h[i]-y[i])
        sum1 = sum1 + (h[i]-y[i])*x1[i]
        sum2 = sum2 + (h[i]-y[i])*x2[i]
        sum3 = sum3 + (h[i]-y[i])*x3[i]

    theta[0] = theta[0] - (alpha/m)*sum0
    theta[1] = theta[1] - (alpha/m)*sum1
```

Graph View:



```python
[40]
import matplotlib.pyplot as plt
x= [i for i in range(0,626)]
plt.plot(x, h)
plt.xlabel('Epochs')
# Naming The Y Axis
plt.ylabel('Cost Function')
plt.show()
```

Implement with Sklearn Model:



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
len(X_test)
```

```
126
```

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

```
LinearRegression()
```

```
y_pre = regressor.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pre)
```

```
216618947.10580584
```

Raw code values

Hypothesis Function is: [2.7548715761665257e+56

Cost Function is: 2.4975042716815650e+112

Updated Parameters : [-1.738651281569117e+51]

SKlearn module

Cost value: 216618947.10580584

Conclusion:

This project assisted me in determining the linear relationship between explanatory (independent) and response (dependent) variables. I had a lot of challenges while working on this project. Finding a good dataset was really time consuming for me. It was complex to implement the Multivariable Linear Regression without SK-Learn. I have learnt a lot of new stuff and the uses of different modules.

—The End—