University of Asia Pacific

Department of Computer Science and Engineering

Artificial Intelligence and Expert Systems (CSE-404)

Project - 02

(Multivariable Linear Regression Using Open Source Dataset)

Date of Submission : 22.03.2021

Submitted By:

Name : Shahin Alam
ID : 17201023
Section : A

Submitted To:

Dr. Nasima Begum
Assistant Professor
Dept. of CSE
University of Asia Pacific

Problem Statement

Predicting Movie budget using Multivariable Linear Regression with Open Source Dataset
Dataset link : https://www.kaggle.com/leonardopena/marvel-vs-dc

Algorithm

KEY TAKEAWAYS

- Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable.
- Multiple regression is an extension of linear (OLS) regression that uses just one explanatory variable.
- MLR is used extensively in econometrics and financial inference.

:

.

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon$$

where, for i=n observations:

y_i=dependent variable
x_i=explanatory variables
β_0=y-intercept (constant term)
β_p=slope coefficients for each explanatory variable
ϵ=the model's error term (also known as the residuals)

Goal

I'm going to build a Machine Learning program which will take 9 independent parameters to predict the budget for the movie. I know it's not useful for general people but It will be very useful for Movie directors.

Programing Language

Here I am using **Python** To solve this problem.

Tools

To code in Python I am comfortable with jupyter Notebook
So that, as a tools I am going to use here jupyter Notebook

Code Implementation

Taking reading CSV file

In [161]:
```
import pandas as pd
import os
import numpy as np
dataset= pd.read_csv('budget.csv',encoding= 'unicode_escape')
dataset.head()
```

Convert Object into INT

In [163]:
```
dataset["Minutes"]=dataset["Minutes"].astype(int)
```

In [164]:
```
dataset.info()
```

Encode Object

In [165]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
dataset.Title = encoder.fit_transform(dataset.Title)
dataset.Company = encoder.fit_transform(dataset.Company)
dataset.Budget = encoder.fit_transform(dataset.Budget)
```

Value Initialization

In [168]:
```
#value initialization
theta0,theta1,theta2,theta3,theta4,theta5,theta6,theta7,theta8,theta9 = 2,0.589744,7.202564,63.6666670,131.846154,2013,115109770
x1,x2,x3,x4,x5,x6,x7,x8,x9 = dataset.Title, dataset.Company, dataset.Rate, dataset.Metascore, dataset.Minutes, dataset.Release,
y = dataset.Budget
```

In [169]:
```
dataset['dist']=np.NaN
dataset['pred']=np.NaN
```

Cost Calculation

In [172]:
```
#calculation
from colorama import Fore, Back, Style
cost_for_ploting = []
for step in range(10):

    #hypothesis function
    dataset['pred']= theta0+theta1*x1+theta2*x2+theta3*x3+theta4*x4+theta5*x5+theta6*x6+theta7*x7+theta8*x8+theta9*x9

    #cost function
    dataset['dist']=(dataset.pred-dataset.Budget)**2

    m = dataset.shape[0]

    cost = (dataset['dist'].sum())/(2*m)

    #for ploting
    cost_for_ploting.append(cost)

    print('\n\n')
    print(Fore.BLUE +'For step : ', step)
    print(Fore.RED +'Cost is : {}'.format(cost))

    # alpha/m
    step_size = 0.001


    theta0 = theta0 - (step_size/m)*(np.sqrt(dataset['dist']).sum())
    theta1 = theta1 - (step_size/m)*((np.sqrt(dataset['dist'])*x1).sum())
    theta2 = theta2 - (step_size/m)*((np.sqrt(dataset['dist'])*x2).sum())
    theta3 = theta3 - (step_size/m)*((np.sqrt(dataset['dist'])*x3).sum())
    theta4 = theta4 - (step_size/m)*((np.sqrt(dataset['dist'])*x4).sum())
    theta5 = theta5 - (step_size/m)*((np.sqrt(dataset['dist'])*x5).sum())
    theta6 = theta6 - (step_size/m)*((np.sqrt(dataset['dist'])*x6).sum())
    theta7 = theta7 - (step_size/m)*((np.sqrt(dataset['dist'])*x7).sum())
    theta8 = theta8 - (step_size/m)*((np.sqrt(dataset['dist'])*x8).sum())
    theta9 = theta9 - (step_size/m)*((np.sqrt(dataset['dist'])*x9).sum())


    print(Fore.GREEN+'Gradient Decent optimized thetas : ', theta0,theta1,theta2,theta3,theta4,theta5,theta6,theta7,theta8,theta
    print(Style.RESET_ALL)
```

Output

```
For step :  0
Cost is : 6.059510731580344e+34
Gradient Decent optimized thetas :  -304737374963416.5 -5416207715143045.0 -207088829981792.2 -2302477700940012.5 -2.0691128442
83366e+16 -4.212747257741237e+16 -6.13999506729235e+17 -4.592668930046312e+22 -1.2850061435616022e+23 -3.359662336911013e+23



For step :  1
Cost is : 7.306029183511289e+64
Gradient Decent optimized thetas :  -3.248629554601226e+29 -5.533145298937151e+30 -2.26376191125987e+29 -2.4624478788823137e+30
-2.1970534004163025e+31 -4.507543038991427e+31 -6.546702709511975e+32 -5.024373498080735e+37 -1.3926607887640707e+38 -3.7479146
9922671e+38



For step :  2
Cost is : 9.023338905703879e+94
Gradient Decent optimized thetas :  -3.6097070207062394e+44 -6.147060074118479e+45 -2.515630164579932e+44 -2.7361761453515478e+
45 -2.441206646073113e+46 -5.008621952366281e+46 -7.274358531987038e+47 -5.583417619619519e+52 -1.5475522689148528e+53 -4.16523
1053933239e+53



For step :  3
Cost is : 1.1144312835943529e+125
Gradient Decent optimized thetas :  -4.0115729144931046e+59 -6.831402856888573e+60 -2.7956944361730583e+59 -3.0407926631097175e
+60 -2.712984103753514e+61 -5.566228294342739e+61 -8.084207270310038e+62 -6.205017774653822e+67 -1.7198405004859106e+68 -4.6289
46221168691e+68



For step :  4
Cost is : 1.3763830648935733e+155
Gradient Decent optimized thetas :  -4.458181055702606e+74 -7.591942454701109e+75 -3.1069389134254417e+74 -3.3793239052106385e+
75 -3.0150204396755056e+76 -6.18591611524992e+76 -8.984221518960672e+77 -6.895821994931192e+82 -1.9113102289359729e+83 -5.14428
6504710203e+83
```

Conclusion

This is not the end. I will try to make my program more optimized and low cost.
So that I can use this program in my real life development project.