University of Asia Pacific

Artificial Intelligence and Expert System Lab (CSE 404)

Department of CSE

Project: 02

Topic/Question: Multivariable Liner Regression Using Open Source Dataset (data_monthly_rainfall).

Date of Submission: 22march, 2021

Submitted by	Submitted to
Name: Niamul Hasan Id: **17201026** Semester: 4.1 Section: A1	Dr. Nasima Begum Assistant Professor Department of CSE UAP

Tools:

1. Language: Python 3 (for coding)
2. IDE: jupyter notebook (text editor)
3. Google Colab

My dataset title is: Rainfall data of Bangladesh(1970-2016)

The dataset link: https://www.kaggle.com/redikod/historical-rainfall-data-in-bangladesh

My task is to implement the Multivariable Liner Regression Using Open Source Dataset without SK-Learn.

Here the model trained by raw coding in python 3.
I haven't used any library function to train my model.
But only to normalize the dataset, I used a normalized library function from Sk-learn.

Solve demonstration:

At First let's look at the data set: the dataset is about rainfall records of Bangladesh from 1970 to 2016.

Here, I am going to consider 3 features: "Year", "StationIndex", "Month"
label data: "Rainfall"

The marginal difference between values of features is large. So, I have to preprocess the dataset.

So, normalization is must.

Now, Coding: (using Google Colab**)**

the first block of code is to upload the dataset.

```
1  from google.colab import files
2  #uploading dataset
3  uploaded = files.upload()
4
5  for fn in uploaded.keys():
6    print('User uploaded file "{name}" with length {length} bytes'.format(
7        name=fn, length=len(uploaded[fn])))
```

```
Choose Files  data_monthly_rainfall.csv
• data_monthly_rainfall.csv(application/vnd.ms-excel) - 380394 bytes, last modified: 3/17/2021 - 81% done
Saving data_monthly_rainfall.csv to data_monthly_rainfall.csv
```

2nd block of code, Importin some library function.

```
1  import numpy as np # Numpy Header
2  import pandas as pd #pandas Header
3  import random
```

3rd block of code, read the dataset as pandas object.

```
df = pd.read_csv('data_monthly_rainfall.csv') #read dataset
```

4th block of code, checking the data.

```
print(df.tail())
```

```
       Year Station  Month  Rainfall  StationIndex
16750  2016  Teknaf      8       920            33
16751  2016  Teknaf      9       512            33
16752  2016  Teknaf     10       208            33
16753  2016  Teknaf     11        53            33
16754  2016  Teknaf     12         0            33
```

5th block of code. Here 'm' is the length of dataset.

Here we are assigning 'm' as 1000. But the perfect way to do: take full length of the dataset.
But the dataset length is above 16 thousand which is big. To train this big dataset we need much faster computer. So to make it time efficient we are using first 10 thousand data.

```
#m=len(df.Year)#length of dataset
m=10000
print(m)
```

→ perfect way

```
10000
```

Now 6th block. Initializing the features Xs and label data Y.

```
x1=df["Year"]
x2=df["StationIndex"]
x3=df["Month"]
y= df["Rainfall"]
print(x1.tail())
print(x2.head())
print(x3.head())
print(y.head())
len(x1)
```

8th block. Preprocessing of dataset/ Normalization of dataset.

Normalizing the dataset between -1 and 1

Code Block 9.

Setting the value of learning rate (alpha)

Here, as we know that the dataset is huge. To make it time efficient we are using the value of alpha 0.2.

But the perfect way to do this: take the lowest value of all features and divide it by 100 then take the value. But by this the optimization process will take more than day in my computer.
So, I had to make it 0.2.

```
#learning rate
#alpha = min(min(x1),min(x2),min(x3))/100
alpha=0.2
print(alpha)
```

Now the Code block 10 is the optimization process.
where we are initializing the values of coefficients (theta)

Then following the step:

1. Hypothesis function (to calculate hypothetical values)
2. Calculating cost
3. Gradient decent (changing the value of coefficients)

```
J = 100
n = 0 # iteration number
theta = [1,1.5, 8,9]
```

```
for i in range(1000):

  print("Iteration number: ",n+1)
  n = n+1
  # hypothesis function
  h = []
  print("Hypothesis function value is: h0(x)=theta_0+theta_1 * x")
  for i2 in range(m):
    temp = theta[0] + theta[1]*x1[i2] + theta[2]*x2[i2]+theta[3]*x3[i2]
    h.append(temp)

  # cost function
  error_sum = 0
  print("Cost function is: j(theta)=1/(2*m) * i=1_samtionSign_m (h_theta_(x)-y)**2")

  for i3 in range(m):
    error_sum = error_sum + (h[i3] - y[i3])**2

  J = (1/(2*m))*error_sum
  if J == float("inf") :
    theta = [random.randint(-100,100),random.randint(-100,100), random.randint(-100,100),random.randint(-100,100)]
    continue
  print("\nCost function is:",J)
```

```
  # gradient descent
print("\ngradient decent:")
temp0 = 0
for i4 in range(m):
  temp0 = temp0 + (h[i4] - y[i4])

theta[0] = theta[0] - (alpha/m)*temp0

temp1 = 0
for i5 in range(m):
  temp1 = temp1 + (h[i5] - y[i5])*x1[i5]

theta[1] = theta[1] - (alpha/m)*temp1

temp1 = 0
for i5 in range(m):
  temp1 = temp1 + (h[i5] - y[i5])*x2[i5]

theta[2] = theta[2] - (alpha/m)*temp1

temp1 = 0
for i5 in range(m):
  temp1 = temp1 + (h[i5] - y[i5])*x3[i5]

theta[3] = theta[3] - (alpha/m)*temp1

print("New parameter value is: ",theta)
```

```
print("result coefficient is",thetaResult)
```

Now after optimization we can predict the result of given inputs:

```
y=int (input("year :"))
m=int (input("month :"))
d=int (input("district :"))

#prediction
rainfall=theta[0]+theta[1]*y +theta[2]* d +theta[3] *m
print(rainfall)
```

The model works fine.