MINI-PROJECT

"A Self Explanatory and Generalized Program for Regression Analysis – Quadratic Equation."

NOVEMBER 1, 2021

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING, PUNE

Numerical and Statistical Methods

A Self-Explanatory and Generalized Program for Regression Analysis – Quadratic Analysis.

1. Introduction

Regression is the process of adjusting model parameters to fit a prediction y to measured values z. There are independent variables x as inputs to the model to generate the predictions y. For machine learning, the objective is to minimize a loss function by adjusting model parameters. A common loss function is the sum of squared errors between the predicted y and measured z values.

x = Independent Variable, Input, Feature

y = Dependent Variable, Output, Label

z = Output Measurement

The is objective is to minimize a loss function such as a sum of squared errors between the measured and predicted values:

Loss= $\sum_{i=1}^{n}(y_i - z_i)^2$

where n is the number of observations. Regression requires labelled data (output values) for training. Classification, on the other hand, can either be supervised (with z measurements, labels) or unsupervised (no labels, z measurements). Run the following code to load 30 sample data points with input x and measured output z.

1.2 Polynomial Regression with degree 2.

A polynomial model may also be quadratic:

$y = ax^2 + b x + c$

A quadratic model is really just a linear model with two inputs x and $z=x^2$.

$y = a z + b x + c$

This is also called multiple linear regression when there is more than one input y=f (x, z) where f is a function of inputs x and z.

In this project we are building a Self-Explanatory and Generalized Program for Regression Analysis for Quadratic Equations, with the help of this generalized program we can solve any quadratic regression.

2. About Program Language

The program is built in Python language, some modules are also used in the program as mentioned :

```
Matplotlib.pyplot -- to create graphs
numpy -- to perform mathematical operations
```

3. Problem Statement

Regression Analysis is a complicated and time consuming task in order to find precise solution with less time build a python program which is generalized as well as self-explanatory.

4. Solution Design

Steps to build program

- At first we need to import some modules which we are going to use in the program

1. `matplotlib.pyplot` – Matplotlib.pyplot is a collection of functions that male matplotlib work like MATLAB. Each pyplot function makes some change to a figure. Examples - Create a figure, plot some lines in a plotting area, decorate the plot with labels, etc
2. `numpy` – numpy is a python library used for working with arrays. It also has functions for working in the domain of linear algebra, fourier transform and matrices. NumPy aims to provide an array object that is up to 50x faster than the traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.
3. Sys

- After importing the modules then we need input from the user for the required variables. For that we collect the data.
- When data is ready we use some list to store the input from the user.
- By using some loops, python built in functions and built the logic for some calculations.
- Now we have to display all the input values so the user can confirm that the entered values are correct, calculated values and also plotting the calculated values on the graph.

5. Implementation

After having the steps to follow now write an actual program with all the comments so that users can understand the function of each and every operation. So to better understand we have divided the code in the section according to their function.

- Importing the required modules

```
#Importing Required Libraries
import matplotlib.pyplot as plt
import time
import numpy as np
import sys
```

Here in the above code we are importing the required modules for solving the python problem

- Collecting the input from the user and storing them separately

```
# Reading value of n
n = int(input("How many data points? "))
```

In above code mentioned we take the number of inputs from the user.

```
# Storing  the input values of x and y from user in list
lst1 = [ ]
lst2 = [ ]
```

In above code mentioned we will make two separate lists two store data points.

```
# Creating numpy array x & y to store n data points
x = np.zeros(n)
y = np.zeros(n)


# Reading data
print("Enter data:")

#Taking Input and Storing the data for variables x and y
for i in range(n):
   x[i] = float(input("x["+str(i)+"]= "))
   y[i] = float(input("y["+str(i)+"]= "))
```

Here in the above code we are taking the n number of inputs as mentioned by user and storing the values in list in form of x and y.

-

```
# Finding required sum for least square methods
s0, s1, s2, s3, s4, s5, s6 = 0,0,0,0,0,0,0
```

```
for i in range(n):
    s0 = s0 + x[i]
    s3 = s3 + x[i]*x[i]
    s1 = s1 + y[i]
    s2 = s2 + x[i]*y[i]
    s4 = s4 + x[i]*x[i]*y[i]
    s5 = s5 + x[i]*x[i]*x[i]
    s6 = s6 + x[i]*x[i]*x[i]*x[i]
```

Here in the above code in order to solve the equations we are going take the summation x^1, x^2, y^1, x^2y^1, x^3, x^4, x^1y^1 to solve further.

```
#Creating and Storing the data in matrix as given below
A = [[s3, s0, n  ], [s5, s3, s0 ], [s6, s5, s3 ]]
B = [[s1], [s2], [s4]]
```

Here in the above code we are solving the equations using the matrix.

- Displaying the input according the required variables

```
# Displaying  the input values of x and y from user in list
lst1.append(x)
lst2.append(y)
print("The data values for Time from users is/are", lst1)
print("The data values for Tensile Strength from users is/are",
lst2)
```

Here in the above code we are displaying the inputs taken from the user.

- Solving for the Matrices from user input and displaying points on the curve

```
#Solving the Matrices to get the values of a, b and c
p = np.linalg.solve(A,B)
print(A)
print(B)
print(p)
print("p[0][0]=", p[0][0])
print("p[1][0]=", p[1][0])
print("p[2][0]=", p[2][0])
a = p[0][0]
b = p[1][0]
c = p[2][0]
```

Here in the above code we are solving the matrix in order to find the coefficient of the equations.

- Displaying Polynomial Equation of Degree 2/ Quadratic Equation, datapoints on the curve and graph plotted with the datapoints

```
#Displaying the Polynomial Equation of Degree 2/ Quadratic
Equation,fitting the Curve from data points
print("And equation is: y = %0.4f x^2 + %0.4f x + %0.4f" %(a,b,c))
```

```
#Displaying the curve using datapoints from user
p2 = np.polyfit(x,y,2)
print(p2)

#Displaying/ PLotting the curve according to the data points
acquired from the user
plt.plot(x,y,'ro',label='Measured (y)')
plt.plot(x,np.polyval(p2,x),'b-',label='Predicted (y)')
plt.legend(); plt.show()
print("End of program")
```

Here in the above code we are displaying the graph of the output.

6. Program

Input -

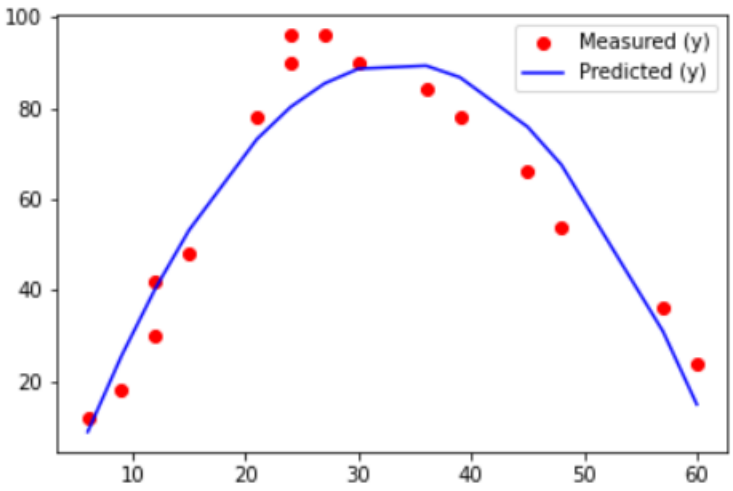
```
# Quadratic regression method in Python
# Fitting y = ax^2 + bx + c to given n data points
#Importing Required Libraries
import matplotlib.pyplot as plt
import time
import numpy as np
import sys

# Reading value of n
n = int(input("How many data points? "))

# Storing  the input values of x and y from user in list
lst1 = [ ]
lst2 = [ ]

# Creating numpy array x & y to store n data points
x = np.zeros(n)
y = np.zeros(n)

# Reading data
print("Enter data:")

#Taking Input and Storing the data for variables x and y
for i in range(n):
    x[i] = float(input("x["+str(i)+"]= "))
    y[i] = float(input("y["+str(i)+"]= "))

# Finding required sum for least square methods
s0, s1, s2, s3, s4, s5, s6 = 0,0,0,0,0,0,0

for i in range(n):
    s0 = s0 + x[i]
```

```
        s3 = s3 + x[i]*x[i]
        s1 = s1 + y[i]
        s2 = s2 + x[i]*y[i]
        s4 = s4 + x[i]*x[i]*y[i]
        s5 = s5 + x[i]*x[i]*x[i]
        s6 = s6 + x[i]*x[i]*x[i]*x[i]

    #Creating and Storing the data in matrix as given below
    A = [[s3, s0, n  ], [s5, s3, s0 ], [s6, s5, s3 ]]
    B = [[s1], [s2], [s4]]

    # Displaying  the input values of x and y from user in list
    lst1.append(x)
    lst2.append(y)
    print("The data values for Time from users is/are", lst1)
    print("The data values for Tensile Strength from users is/are",
    lst2)

    #Solving the Matrices to get the values of a, b and c
    p = np.linalg.solve(A,B)
    print(A)
    print(B)
    print(p)
    print("p[0][0]=", p[0][0])
    print("p[1][0]=", p[1][0])
    print("p[2][0]=", p[2][0])
    a = p[0][0]
    b = p[1][0]
    c = p[2][0]

    #Displaying the Polynomial Equation of Degree 2/ Quadratic
    Equation,fitting the Curve from data points
    print("And equation is: y = %0.4f x^2 + %0.4f x + %0.4f" %(a,b,c))

    #Displaying the curve using datapoints from user
    p2 = np.polyfit(x,y,2)
    print(p2)

    #Displaying/ PLotting the curve according to the data points aquired
    from the user
    plt.plot(x,y,'ro',label='Measured (y)')
    plt.plot(x,np.polyval(p2,x),'b-',label='Predicted (y)')
    plt.legend(); plt.show()
    print("End of program")
```

Output –

```
How many data points? 16
Enter data:
```

```
x[0]= 6
y[0]= 12
x[1]= 9
y[1]= 18
x[2]= 12
y[2]= 30
x[3]= 12
y[3]= 42
x[4]= 15
y[4]= 48
x[5]= 21
y[5]= 78
x[6]= 24
y[6]= 90
x[7]= 24
y[7]= 96
x[8]= 27
y[8]= 96
x[9]= 30
y[9]= 90
x[10]= 36
y[10]= 84
x[11]= 39
y[11]= 78
x[12]= 45
y[12]= 66
x[13]= 48
y[13]= 54
x[14]= 57
y[14]= 36
x[15]= 60
y[15]= 24
The data values for Time from users is/are [array([ 6.,  9., 12., 12., 15.
, 21., 24., 24., 27., 30., 36., 39., 45.,
       48., 57., 60.])]
The data values for Tensile Strength from users is/are [array([12., 18., 3
0., 42., 48., 78., 90., 96., 96., 90., 84., 78., 66.,
       54., 36., 24.])]
[[17847.0, 465.0, 16], [800253.0, 17847.0, 465.0], [39217527.0, 800253.0,
17847.0]]
[[942.0], [28332.0], [1004508.0]]
[[ -0.10698872]
 [  7.17306139]
 [-30.25286764]]
p[0][0]= -0.10698871880667671
p[1][0]= 7.173061390898328
p[2][0]= -30.252867639060177
And equation is: y = -0.1070 x^2 + 7.1731 x + -30.2529
[ -0.10698872   7.17306139 -30.25286764]
```




```
End of program
```