**SSN COLLEGE OF ENGINEERING**

UIT2201 – PROGRAMMING AND DATA STRUCTURES

PSP Exercise-03

U. Pranaav 3122 22 5002 093 IT-B

1.

I. AIM:

Write a Python program that reads a CSV file and performs four different linear regression

tasks (that is, find the regression parameters β0 and β1).

• x: estimated proxy size; y: actual LOC (added+modified)

• x: estimated proxy size; y: actual time taken

• x: planned LOC (added+modified); y: actual LOC (added+modified)

• x: planned LOC (added+modified); y: actual time taken

II. CODE:

# -\*- coding: utf-8 -\*-

'''

This module contains python functions that performs a linear

regression task on the given data present inside a CSV file.

There will be 4 different linear regression tasks performed.

In this source code I have executed my own logic. The code

follows good coding practices.

Your comments and suggestions are welcome.

Created on Wed May 01 2023

Revised on Wed May 01 2023

Original Author: U. Pranaav <pranaav2210205@ssn.edu.in>

'''

import csv

*def* csv\_read(*filepath*):

    '''

    This function reads the contents of the CSV file

    in the given path and returns a list of its

    contents with each element in float form.

    args:

        filepath: the path of the CSV file

    Returns:

        List of lists with each element containing

        float values.

    '''

    values = []

    with open(*filepath*,'r') as csv\_file:

        reader = csv.reader(csv\_file)

        for row in reader:

            tem\_list = []

            for val in row:

                tem\_list.append(float(val))

            values.append(tem\_list)

    return values

*def* csv\_write(*filepath*,*contents*):

    '''

    This function writes the given contents into a

    CSV file in the given path.

    args:

        filepath: the path of the CSV file.

        contents: the contents to be written into

        the CSV file.

    Returns:

        None

    '''

    with open(*filepath*,*mode*='w',*newline*='') as csv\_file:

        to\_write = csv.writer(csv\_file)

        to\_write.writerows(*contents*)

*def* est\_prox\_actual\_LOC(*filepath*):

    '''

    This function performs a linear regression task

    between the estimated proxy size as x and actual

    LOC as y and returns the value of slope [B1] and

    intercept [B0].

    args:

        filepath: the path of the CSV file

    Returns:

        A tuple containing the values of B1 and B0.

    '''

    vals = csv\_read(*filepath*)

    x = [val[1] for val in vals]

    y = [val[3] for val in vals]

    x\_avg = sum(x) / len(x)

    y\_avg = sum(y) / len(y)

    sum\_xy = 0

    sum\_x2 = 0

    n = len(x)

    for i in range(n):

        sum\_xy += x[i]\*y[i]

        sum\_x2 += x[i]\*\*2

    #the slope

    B1 = (sum\_xy - (n \* x\_avg \* y\_avg)) / (sum\_x2 - (n \* (x\_avg \*\* 2)))

    #the intercept

    B0 = y\_avg - B1 \* x\_avg

    val = rsquared(x,y,B1,B0)

    return (B1,B0,val)

*def* est\_prox\_actual\_time(*filepath*):

    '''

    This function performs a linear regression task

    between the estimated proxy size as x and actual

    time taken as y and returns the value of slope [B1]

    and intercept [B0].

    args:

        filepath: the path of the CSV file

    Returns:

        A tuple containing the values of B1 and B0.

    '''

    vals = csv\_read(*filepath*)

    x = [val[1] for val in vals]

    y = [val[4] for val in vals]

    x\_avg = sum(x) / len(x)

    y\_avg = sum(y) / len(y)

    sum\_xy = 0

    sum\_x2 = 0

    n = len(x)

    for i in range(n):

        sum\_xy += x[i]\*y[i]

        sum\_x2 += x[i]\*\*2

    B1 = (sum\_xy - (n \* x\_avg \* y\_avg)) / (sum\_x2 - (n \* (x\_avg \*\* 2)))

    B0 = y\_avg - B1 \* x\_avg

    val = rsquared(x,y,B1,B0)

    return (B1,B0,val)

*def* planned\_LOC\_actual\_LOC(*filepath*):

    '''

    This function performs a linear regression task

    between the planned LOC as x and actual LOC as

    y and returns the value of slope [B1] and

    intercept [B0].

    args:

        filepath: the path of the CSV file

    Returns:

        A tuple containing the values of B1 and B0.

    '''

    vals = csv\_read(*filepath*)

    x = [val[2] for val in vals]

    y = [val[3] for val in vals]

    x\_avg = sum(x) / len(x)

    y\_avg = sum(y) / len(y)

    sum\_xy = 0

    sum\_x2 = 0

    n = len(x)

    for i in range(n):

        sum\_xy += x[i]\*y[i]

        sum\_x2 += x[i]\*\*2

    B1 = (sum\_xy - (n \* x\_avg \* y\_avg)) / (sum\_x2 - (n \* (x\_avg \*\* 2)))

    B0 = y\_avg - B1 \* x\_avg

    val = rsquared(x,y,B1,B0)

    return (B1,B0,val)

*def* planned\_LOC\_actual\_time(*filepath*):

    '''

    This function performs a linear regression task

    between the planned LOC as x and actual time as

    y and returns the value of slope [B1] and

    intercept [B0].

    args:

        filepath: the path of the CSV file

    Returns:

        A tuple containing the values of B1 and B0.

    '''

    vals = csv\_read(*filepath*)

    x = [val[2] for val in vals]

    y = [val[4] for val in vals]

    x\_avg = sum(x) / len(x)

    y\_avg = sum(y) / len(y)

    sum\_xy = 0

    sum\_x2 = 0

    n = len(x)

    for i in range(n):

        sum\_xy += x[i]\*y[i]

        sum\_x2 += x[i]\*\*2

    B1 = (sum\_xy - (n \* x\_avg \* y\_avg)) / (sum\_x2 - (n \* (x\_avg \*\* 2)))

    B0 = y\_avg - B1 \* x\_avg

    val = rsquared(x,y,B1,B0)

    return (B1,B0,val)

*def* gen\_points(*m*,*c*):

    '''

    The given function generates a list of random

    points when given the slope and intercept.

    The input is not modified in any way and there

    are no side effects.

    args:

        m: the slope of line

        c: the intercept of line

    Returns:

        A tuple of lists containing x and y values.

    '''

    x = [val for val in range(0, 10)]

    y = [*m* \* xi + *c* for xi in x]

    return (x,y)

*def* rsquared(*x*, *y*, *slope*, *intercept*):

    '''

    This function calculates the Coefficient of Determination, R-squared, given the values of x, y, slope and intercept

    args:

        x: a list of x values

        y: a list of y values

        slope: the slope obtained from linear regression

        intercept: the intercept obtained from linear regression

    Returns:

        R-squared value

    '''

    yhat = [*slope*\*xi + *intercept* for xi in *x*]

    ybar = sum(*y*)/len(*y*)

    ssreg = sum([(yihat-ybar)\*\*2 for yihat in yhat])

    sstot = sum([(yi-ybar)\*\*2 for yi in *y*])

    return ssreg/sstot

#driver code

if \_\_name\_\_ == '\_\_main\_\_':

    #this part of the code will only be run when the function is called directly

    #it will not be executed when it is imported as a module

    #the file path where the data is stored

    pathway = *r*"D:\college files\DSA\Pranaav-UIT2201-psp-ex-03\data.csv"

    #data with program number,estimated proxy size,planned LOC (added+modified),actual LOC (added+modified),actual development hours

    content = [[1,130,163,186,15.0],

                [2,650,765,699,69.9],

                [3,99,141,132,6.5],

                [4,150,166,272,22.4],

                [5,128,137,291,28.4],

                [6,302,355,331,65.9],

                [7,95,136,199,19.4],

                [8,945,1206,1890,198.7],

                [9,368,433,788,38.8],

                [10,961,1130,1601,138.2]]

    csv\_write(pathway,content)

    slope, intercept, correlation\_coeff = est\_prox\_actual\_LOC(pathway)

    print(*f*"y = {slope*:.2f*}x + {intercept*:.2f*}")

    predicted\_val = slope \* 210 + intercept

    print(*f*"Predicted val is: {predicted\_val*:.2f*}")

    print(*f*"The correlation coefficient R^2 is: {correlation\_coeff}")

    print()

    slope, intercept, correlation\_coeff = est\_prox\_actual\_time(pathway)

    print(*f*"y = {slope*:.2f*}x + {intercept*:.2f*}")

    x, y = gen\_points(slope,intercept)

    predicted\_val = slope \* 210 + intercept

    print(*f*"Predicted val is: {predicted\_val*:.2f*}")

    print(*f*"The correlation coefficient R^2 is: {correlation\_coeff}")

    print()

    slope, intercept, correlation\_coeff = planned\_LOC\_actual\_LOC(pathway)

    print(*f*"y = {slope*:.2f*}x + {intercept*:.2f*}")

    x, y = gen\_points(slope,intercept)

    predicted\_val = slope \* 210 + intercept

    print(*f*"Predicted val is: {predicted\_val*:.2f*}")

    print(*f*"The correlation coefficient R^2 is: {correlation\_coeff}")

    print()

    slope, intercept, correlation\_coeff = planned\_LOC\_actual\_time(pathway)

    print(*f*"y = {slope*:.2f*}x + {intercept*:.2f*}")

    x, y = gen\_points(slope,intercept)

    predicted\_val = slope \* 210 + intercept

    print(*f*"Predicted val is: {predicted\_val*:.2f*}")

    print(*f*"The correlation coefficient R^2 is: {correlation\_coeff}")

    print()

III. OUTPUT:

y = 1.73x + -22.55

Predicted val is: 340.31

The correlation coefficient R^2 is: 0.9110637099775759

y = 0.17x + -4.04

Predicted val is: 31.27

The correlation coefficient R^2 is: 0.8710617661167361

y = 1.43x + -23.92

Predicted val is: 276.58

The correlation coefficient R^2 is: 0.927588756422322

y = 0.14x + -4.60

Predicted val is: 24.83

The correlation coefficient R^2 is: 0.8987665452555462