Ex.No: 3 **Linear Regression** **05 May 2023**

Code:

```
"""
    This module provides functionality for printing a line of linear
regression, by taking
    x and y values as different parameters.

    Using linear regression formulas, constants b0 and b1 are found.

    They are substituted into the linear regression line formula to find
the equation

    Original Author: Pranesh Kumar

    Created on: 03 May 2023
"""

# importing csv module for reading a csv file
import csv


def readcsvfile(filepath):
    """
    Reads each line of the csv file
    format:
    program number,estimated proxy size,planned LOC (added+modified),actual
LOC (added+modified),actual
    development hours

    Requirements:
    • 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

    :param filepath: path of csv file
    :return: nested list containing each line of the csv file
    """

    csvreader = csv.reader(open(filepath, "r"))
    lines = []
    for line in csvreader:
        lines.append(line)

    return lines


def statement1(lines):
    """
    Here, the x value is the estimated proxy size and the y value is the
actual LOC

    :param lines: nested list containing each line of the csv file
    :return: None
    """

    n = 10
```

```
    xavg = 0.0
    yavg = 0.0

    b1 = 0.0
    b0 = 0.0

    numeratorsum = 0
    denominatorsum = 0

    for line in lines:
        estimatedproxy = float(line[1])
        actualloc = float(line[3])

        xavg += estimatedproxy
        yavg += actualloc

        numeratorsum += (estimatedproxy * actualloc)
        denominatorsum += (estimatedproxy * estimatedproxy)

    xavg /= 10
    yavg /= 10

    b1 = (numeratorsum - (n * xavg * yavg)) / (denominatorsum - (n * xavg *
xavg))
    b0 = yavg - (b1 * xavg)

    print("• X: estimated proxy size; Y: actual LOC (added+modified)")
    print(f"y = {round(b0, 3)} + {round(b1, 3)}x\n")


def statement2(lines):
    """
    Here, the x value is the estimated proxy size and the y value is the
actual time taken.

    :param lines: nested list containing each line of the csv file
    :return: None
    """

    n = 10

    xavg = 0.0
    yavg = 0.0

    b1 = 0.0
    b0 = 0.0

    numeratorsum = 0
    denominatorsum = 0

    for line in lines:
        estimatedproxy = float(line[1])
        actualtimetaken = float(line[4])

        xavg += estimatedproxy
        yavg += actualtimetaken

        numeratorsum += (estimatedproxy * actualtimetaken)
        denominatorsum += (estimatedproxy * estimatedproxy)
```

```
    xavg /= 10
    yavg /= 10

    b1 = (numeratorsum - (n * xavg * yavg)) / (denominatorsum - (n * xavg *
xavg))
    b0 = yavg - (b1 * xavg)

    print("• X: estimated proxy size; Y: actual time taken")
    print(f"y = {round(b0, 3)} + {round(b1, 3)}x\n")


def statement3(lines):
    """
    Here, the x value is the planned LOC and the y value is the actual LOC

    :param lines: nested list containing each line of the csv file
    :return: None
    """

    n = 10

    xavg = 0.0
    yavg = 0.0

    b1 = 0.0
    b0 = 0.0

    numeratorsum = 0
    denominatorsum = 0

    for line in lines:
        plannedloc = float(line[2])
        actualloc = float(line[3])

        xavg += plannedloc
        yavg += actualloc

        numeratorsum += (plannedloc * actualloc)
        denominatorsum += (plannedloc * plannedloc)

    xavg /= 10
    yavg /= 10

    b1 = (numeratorsum - (n * xavg * yavg)) / (denominatorsum - (n * xavg *
xavg))
    b0 = yavg - (b1 * xavg)

    print("• X: planned LOC (added+modified); Y: actual LOC
(added+modified)")
    print(f"y = {round(b0, 3)} + {round(b1, 3)}x\n")


def statement4(lines):
    """
    Here, the x value is the planned LOC and the y value is the actual time
taken

    :param lines: nested list containing each line of the csv file
    :return: None
    """
```

```
    n = 10

    xavg = 0.0
    yavg = 0.0

    b1 = 0.0
    b0 = 0.0

    numeratorsum = 0
    denominatorsum = 0

    for line in lines:
        plannedloc = float(line[2])
        actualtimetaken = float(line[3])

        xavg += plannedloc
        yavg += actualtimetaken

        numeratorsum += (plannedloc * actualtimetaken)
        denominatorsum += (plannedloc * plannedloc)

    xavg /= 10
    yavg /= 10

    b1 = (numeratorsum - (n * xavg * yavg)) / (denominatorsum - (n * xavg *
xavg))
    b0 = yavg - (b1 * xavg)

    print("• X: planned LOC (added+modified); Y: actual time taken")
    print(f"y = {round(b0, 3)} + {round(b1, 3)}x\n")


# driver code
if __name__ == "__main__":
    data = readcsvfile("loc.csv")
    statement1(data)
    statement2(data)
    statement3(data)
    statement4(data)
```

Output:

- X: estimated proxy size; Y: actual LOC (added+modified)

y = -22.553 + 1.728x

- X: estimated proxy size; Y: actual time taken

y = -4.039 + 0.168x

- X: planned LOC (added+modified); Y: actual LOC (added+modified)

y = -23.924 + 1.431x

- X: planned LOC (added+modified); Y: actual time taken

y = -23.924 + 1.431x