Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут комп'ютерних наук та інформаційних технологій
Кафедра автоматизованих систем управління

Звіт

до лабораторної роботи № 1

з дисципліни

"**Моделювання процесів та смарт систем**"

Виконав: студент групи ОІ-35

Маселко Володимир

Прийняв : асистент каф. АСУ

Мельник Р. В.

Львів – 2024

Лабораторна робота № 1

Концепція моделі “чорний ящик” та встановлення залежності між її вхідними і вихідними параметрами.»

Мета: Зрозуміти концепцію моделі “чорний ящик” та набути навички застосування методу найменших квадратів для визначення залежності між вхідними і вихідними параметрами моделі.

Варіант: 15

Короткі теоретичні відомості

Модель “чорного ящика”. Для більш визначеної та точної характеристики конструкції системи слід розвинути її модель, перетворюючи наявні відомості так, щоб у результаті отримати більш зручну форму моделі, додаючи у модель за необхідності додаткові відомості. Для людини важливу роль відіграють образні, візуальні моделі, тому перейдемо від визначення системи до його візуального еквіваленту. По-перше, визначення системи нічого не говорить про її внутрішню будову. Тому її можна зобразити у вигляді непрозорого “ящика”, виділеного з навколишнього середовища. Слід зазначити, що вже навіть така, максимально проста модель відображає дві наступні важливі властивості системи: цілісність і відокремленість від середовища. По-друге, у визначенні системи побічно йдеться про те, що хоча “ящик” і виокремлений із середовища, але не є повністю від нього ізольованим. Справді, адже досягнутою метою є наперед заплановані зміни у навколишньому середовищі, якісь продукти роботи системи, призначені для споживання поза нею. Інакше кажучи, система пов’язана із середовищем і з допомогою цих зв’язків впливає нього (рис.1). Ці зв’язки називаються виходами системи та зображені на (рис. 1) як стрілки напрямлені із системи у зовнішнє середовище. Крім того, у визначенні є вказівка і на наявність зв’язків іншого типу. Система є засобом, тому мають існувати можливості її використання, тобто зв’язки із середовищем, які спрямовані ззовні у систему. Зобразимо ці зв’язки також у вигляді відповідних стрілок, спрямованих від середовища до системи (див. рис. 1), і назвемо їх входами системи.

В результаті ми збудували модель системи, яка отримала назву “чорного ящика” (рис.1). Ця назва образно підкреслює повну відсутність відомостей про внутрішній зміст “ящика”: у цій моделі задаються, фіксуються, перераховуються тільки вхідні та вихідні зв’язки системи із середовищем (навіть “стінки ящика”, тобто межі між системою та середовищем, у цій моделі зазвичай не описуються, лише маються на увазі, визнаються існуючими). Така модель незважаючи на зовнішню простоту і на відсутність відомостей про нутрощі системи, часто виявляється корисною. У багатьох випадках досить змістовного словесного опису входів та виходів; тоді модель “чорного ящика” є просто їх списком. Наприклад, модель телевізора така: входи – шнур електроживлення, антена, елементи управління та налаштування; виходи – екран та динаміки. В інших випадках потрібно кількісний опис деяких або всіх входів та виходів. Намагаючись максимально формалізувати модель “чорного ящика”, ми приходимо до задання двох множин X і У вхідних та вихідних змінних, але жодних інших відношень між цими множинами фіксувати не можна (інакше це вже буде не “чорний ящик”, а “прозорий ящик”).

Завдання 1. Побудова прикладу моделі “чорний ящик”

1. Вибір системи

Обрана система – ліхтарик. Це пристрій, призначений для освітлення, який працює на батарейках або акумуляторах, використовуючи лампочку або світлодіод.

2. Входи системи

- Електрична енергія

- Натискання кнопки ввімкнення/вимкнення
- Регулювання світлового потоку
- Волога
- Температура
- Механічні пошкодження

3. Виходи системи

- Яскравість променя
- Колір променя
- Фокус променя
- Нагрівання ліхтарика
- Споживання енергії
- Міцність ліхтарика
- Зручність тримання в руці

4. Небажані входи та виходи

- Попадання вологи
- Попадання пилу
- Високі температури
- Механічні удари та падіння
- Розрядження ліхтарика

5. Способи усунення недоліків

- Герметичний корпус для захисту від вологи та пилу
- Ударостійкі матеріали: прогумований корпус, амортизаційні вставки
- Захист від перегріву: використання алюмінієвого радіатор, автоматичне зниження яскравості при нагріванні
- Енергозберігаючі режими: регулювання яскравості, автоматичне вимкнення при бездіяльності
- Індикатор заряду, щоб користувач знав, коли слід замінити батарею або зарядити акумулятор

Завдання 2. Визначення залежності між вхідними і вихідними параметрами моделі

x1i	0	0	0	1	1	2	2	2
x2i	1,5	2,5	3,5	1,5	3,5	1,5	2,5	2,5
yi	2,3	8,5	0,5	2	1	9,1	5	7,2

Код програми

```
import numpy as np
import matplotlib.pyplot as plt
```

Формування системи рівнянь для методом найменших квадратів за формулою

$$\begin{cases} \frac{\partial S}{\partial a_0} = 2\sum_{i=1}^{n}(a_0 + a_1x_{1i} + a_2x_{2i} - y_i)\cdot 1 = 0 \\ \frac{\partial S}{\partial a_1} = 2\sum_{i=1}^{n}(a_0 + a_1x_{1i} + a_2x_{2i} - y_i)\cdot x_{1i} = 0 \\ \frac{\partial S}{\partial a_2} = 2\sum_{i=1}^{n}(a_0 + a_1x_{1i} + a_2x_{2i} - y_i)\cdot x_{2i} = 0 \end{cases}$$

```
def calcResColumn(res, allRows, i, j):
# n variable is allRows column index
# allRows and res are related, because allRows.shape[0] == res.shape[0] + 1 == res.shape[1]
# Therefore i and j indexes can be used for allRows rows
	for n in range(allRows.shape[1]):
		res[i][j] += allRows[j][n] * allRows[i][n]

def calcResRow(res, allRows, i):
	# j variable is res column index
	for j in range(res.shape[1]):
		calcResColumn(res, allRows, i, j)

def createMatrix(rows, y):
	# first row - ones
	# middle rows - x[n]
	# last row - y
	allRows = np.vstack((np.ones(rows.shape[1]), rows, y))

	res = np.zeros((allRows.shape[0] - 1, allRows.shape[0]))

	# i variable is res row index
	for i in range(res.shape[0]):
		calcResRow(res, allRows, i)

	res *= 2

	return res
```

Розв’язування системи рівнянь методом Гауса

```
def simplifyMatrix(matr, n):
	for i in range(n + 1, matr.shape[0]):
		for j in range(matr.shape[1] - 1, n - 1, -1):
			matr[i][j] -= matr[n][j] / matr[n][n] * matr[i][n]

# if matrix have any 0 on main diagonal, then result is error
def solveGuassingMatrix(matr):
	simplifiedMatrix = matr.copy()
	# n represents current step of simplifying
	for n in range(simplifiedMatrix.shape[0] - 1):
		simplifyMatrix(simplifiedMatrix, n)

	res = np.ones(simplifiedMatrix.shape[0])

	# solve every equation
	for i in range(simplifiedMatrix.shape[0] - 1, -1, -1):
		currentCoef = simplifiedMatrix[i][i]
		currentSum = 0
		# calculate new row based on previous x results
		for j in range(res.shape[0] - 1, i, -1):
			currentSum += res[j] * simplifiedMatrix[i][j]
		res[i] = (simplifiedMatrix[i][-1] - currentSum) / currentCoef

	return res
```

Знаходження коефіцієнтів

```
# Дані
rows = np.array([[0, 0, 0, 1, 1, 2, 2, 2], [1.5, 2.5, 3.5, 1.5, 3.5, 1.5, 2.5, 2.5]])
y = np.array([ 2.3, 8.5, 0.5, 2, 1, 9.1, 5, 7.2])

matr = createMatrix(rows, y)
print(matr)

coeffs = solveGuassingMatrix(matr)
print("\nКоефіцієнти:", coeffs)
```

Обчислення значення функції у точці (x1=1.5, x2=3)

```
# Обчислення значення функції у точці (x1=1.5, x2=3)
x1, x2 = 1.5, 3
y_pred = coeffs[0] + coeffs[1] * x1 + coeffs[2] * x2
print(f"Значення функції у точці (1.5, 3): y = {y_pred:.4f}")
```

Знаходження точності отриманого наближення за величиною коефіцієнта детермінації R^2

```
def r2_score(rows, y, coeffs):
	res = 1
	numerator = 0
	for i in range(y.shape[0]):
		y_pred = coeffs[0]
		for j in range(1, coeffs.shape[0]):
			y_pred += coeffs[j] * rows[j - 1][i]
		numerator += (y_pred - y[i]) ** 2
```

```
        denominator = 0
        y_mean = np.mean(y)
        for cur_y in y:
                denominator += (y_mean - cur_y) ** 2

        res -= numerator / denominator
        return res
```

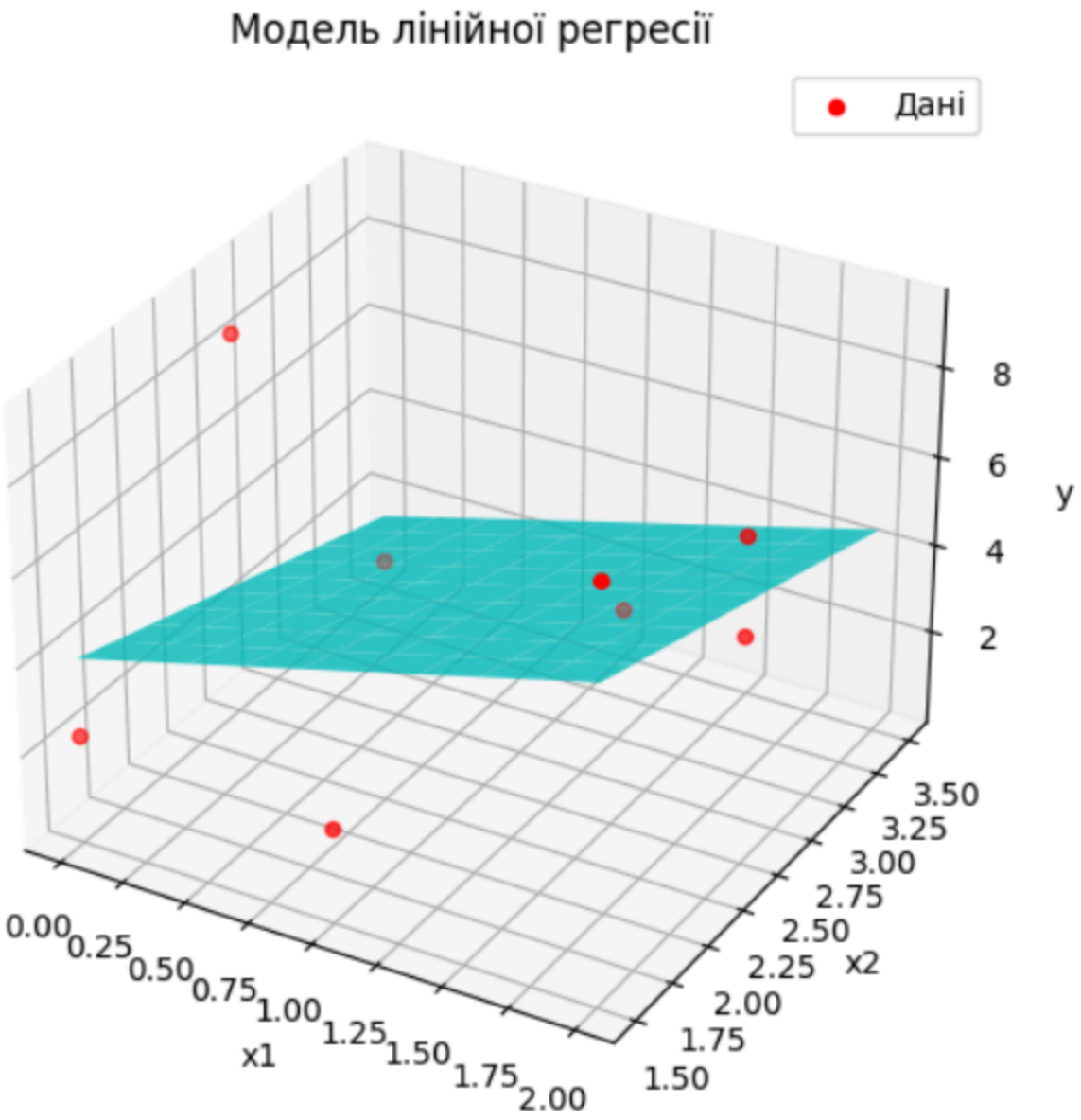
```
# Коефіцієнт детермінації R^2
r2 = r2_score(rows, y, coeffs)
print(f"Коефіцієнт детермінації R^2 = {r2:.4f}")
```

Побудова графіка

```
# Побудова графіка
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(rows[0], rows[1], y, color='red', label='Дані')

# Побудова площини регресії
x1_range = np.linspace(min(rows[0]), max(rows[0]), 10)
x2_range = np.linspace(min(rows[1]), max(rows[1]), 10)
x1_grid, x2_grid = np.meshgrid(x1_range, x2_range)
y_grid = coeffs[0] + coeffs[1] * x1_grid + coeffs[2] * x2_grid
ax.plot_surface(x1_grid, x2_grid, y_grid, alpha=0.8, color='cyan')

ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('y')
ax.set_title('Модель лінійної регресії')
plt.legend()
plt.show()
```



Висновок

У ході виконання даної лабораторної роботи я зрозумів концепцію моделі “чорний ящик” та набув навички застосування методу найменших квадратів для визначення залежності між вхідними і вихідними параметрами моделі.