# **ИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені**

# **ТАРАСА ШЕВЧЕНКА**



## **ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра прикладних інформаційних систем**

**Звіт до лабораторної роботи №4**

# **з курсу**

**«Системний аналіз та теорія прийняття рішень»**

*Студента 3 курсу*

*групи ПП-31 спеціальності 122 «Комп'ютерні науки» ОП «Прикладне програмування»*

Момотюка Михайла Тарасовича

*Викладач:*

Плескач В.Л.

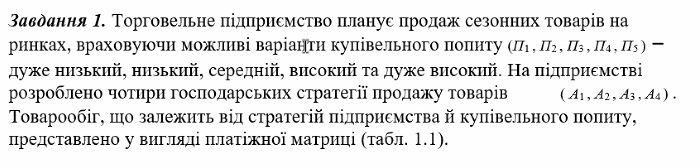
Білий Р.О

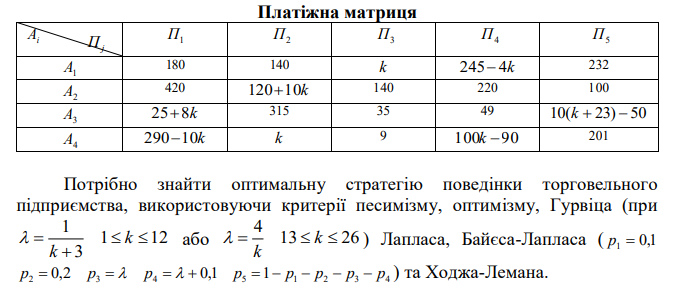
## **Київ – 2023**

**Тема роботи:** Прийняття рішень в умовах невизначеності. Підтримка прийняття рішень з використанням електронних таблиць.

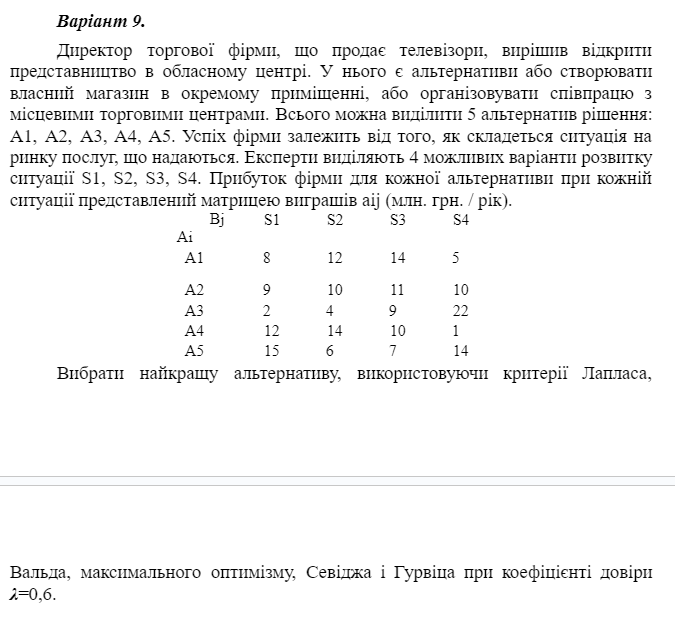
**Мета роботи:** Набуття практичних навичок знаходження оптимального управлінського рішення в умовах невизначеності з використанням критеріїв Лапласа, Вальда, максимального оптимізму, Севіджа, Гурвіца.

**Завдання:**

****

****

Із використанням засобів редактора електронних таблиць Excel та критеріїв песимізму, оптимізму, Гурвіца, Лапласа, Байєса-Лапласа, Ходжа-Лемана знайти оптимальний розв’язок задачі.



**Хід роботи**

**Завдання 1:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bj  Аi | S1 | S2 | S3 | S4 | S5 |
| А1 | 180 | 140 | 11 | 201 | 232 |
| А2 | 420 | 164 | 140 | 220 | 100 |
| А3 | 113 | 315 | 35 | 49 | 290 |
| А4 | 180 | 11 | 9 | 1010 | 201 |

λ=0.07, k=11, p1=0.1, p2= 0.2, p3=0.07, p4=0.17, p5=0.46

**Розрахунок за критерієм Гурвіца:**

F(A1) = 232 \* 0.07 + 11 \* (1 - 0.07) = 26.47

F(A2) = 420 \* 0.07 + 100 \* (1 - 0.07) = 122.4

F(A3) = 315 \* 0.07 + 35 \* (1 - 0.07) = 54.6

F(A4) = 1010 \* 0.07 + 9 \* (1 - 0.07) = 79.07

Обираємо альтернативу A2.

**Розрахунок за критерієм Лапласа:**

F(A1) = (180 + 140 + 11 + 201 + 232 )/5 = 152.8

F(A2) = (420 + 164 + 140 + 220 + 100)/5 = 208.8

F(A3) = (113 + 315 + 35 + 49 + 290)/5 = 160.4

F(A4) = (180 + 11 + 9 + 1010 + 201)/5 = 282.2

Обираємо альтернативу A4.

**Розрахунок за критерієм Байєса-Лапласа:**

Шукаємо значення альтернатив та обираємо найбільше

Формула для обрахунків:



F(A1) = 0.07 \* (0.1 \* 180 + 0.2 \* 140 + 0.07 \* 11 + 0.17 \* 201 + 0.46 \* 232) = 13.1362

F(A2) = 0.07 \* (0.1 \* 420 + 0.2 \* 164 + 0.07 \* 140 + 0.17 \* 220 + 0.46 \* 100) = 11.76

F(A3) = 0.07 \* (0.1 \* 113 + 0.2 \* 315 + 0.07 \* 35 + 0.17 \* 49 + 0.46 \* 290) = 15.2936

F(A4) = 0.07 \* (0.1 \* 180 + 0.2 \* 11 + 0.07 \* 9 + 0.17 \*1010 + 0.46 \* 201) = 19.9493

Обираємо альтернативу A4.

**Розрахунок за критерієм Ходжа- Лемана:**

Знаходимо найбільшу суму критеріїв Байєса-Лапласа та Вальда

F(A1) = 0.07 \* 11 + 0.07 \* (0.1 \* 180 + 0.2 \* 140 + 0.07 \* 11 + 0.17 \* 201 + 0.46 \* 232) = 13.9062

F(A2) = 0.07 \* 100 + 0.07 \* (0.1 \* 420 + 0.2 \* 164 + 0.07 \* 140 + 0.17 \* 220 + 0.46 \* 100) = 18.76

F(A3) = 0.07 \* 35 + 0.07 \* (0.1 \* 113 + 0.2 \* 315 + 0.07 \* 35 + 0.17 \* 49 + 0.46 \* 290) = 17.7436

F(A4) = 0.07 \* 9 + 0.07 \* (0.1 \* 180 + 0.2 \* 11 + 0.07 \* 9 + 0.17 \*1010 + 0.46 \* 201) = 20.5793

Обираємо альтернативу A4.

**Розрахунок за критерієм Вальда:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bj  Аi | S1 | S2 | S3 | S4 | S5 |
| А1 | 180 | 140 | 11 | 201 | 232 |
| А2 | 420 | 164 | 140 | 220 | 100 |
| А3 | 113 | 315 | 35 | 49 | 290 |
| А4 | 180 | 11 | 9 | 1010 | 201 |

Шукаємо найкращий варіант з найгірших кожної альтернативи.

Обираємо альтернативу A2.

**Розрахунок за критерієм Севіджа:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bj  Аi | S1 | S2 | S3 | S4 | S5 |
| А1 | 180 | 140 | 11 | 201 | 232 |
| А2 | 420 | 164 | 140 | 220 | 100 |
| А3 | 113 | 315 | 35 | 49 | 290 |
| А4 | 180 | 11 | 9 | 1010 | 201 |

Будуємо матрицю ризиків

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bj  Аi | S1 | S2 | S3 | S4 | S5 |
| А1 | 240 | 175 | 11 | 809 | 232 |
| А2 | 0 | 151 | 140 | 790 | 132 |
| А3 | 307 | 0 | 35 | 961 | -58 |
| А4 | 240 | 304 | 9 | 0 | 31 |

Для кожної альтернативи обираємо найбільший ризик, та серед них обираємо найменше значення.

Обираємо альтернативу A4.

**Завдання 2:**

Матриця:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bj  Аi | S1 | S2 | S3 | S4 |
| А1 | 8 | 12 | 14 | 5 |
| А2 | 9 | 10 | 11 | 10 |
| А3 | 2 | 4 | 9 | 22 |
| А4 | 12 | 14 | 10 | 1 |
| А5 | 15 | 6 | 7 | 14 |

**Розрахунок за критерієм максимального оптимізму:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bj  Аi | S1 | S2 | S3 | S4 |
| А1 | 8 | 12 | 14 | 5 |
| А2 | 9 | 10 | 11 | 10 |
| А3 | 2 | 4 | 9 | **22** |
| А4 | 12 | 14 | 10 | 1 |
| А5 | 15 | 6 | 7 | 14 |

Шукаємо найбільший виграш. Отже, обираємо альтернативу A3.

**Розрахунок за критерієм Гурвіца:**

F(A1) = 14 \* 0.6 + 5 \* (1 - 0.6) = 10.4

F(A2) = 11 \* 0.6 + 9 \* (1 - 0.6) = 10.2

F(A3) = 22 \* 0.6 + 2 \* (1 - 0.6) = 14

F(A4) = 14 \* 0.6 + 1 \* (1 - 0.6) = 8.8

F(A5) = 15 \* 0.6 + 6 \* (1 - 0.6) = 11.4

Обираємо альтернативу A3.

**Розрахунок за критерієм Лапласа:**

F(A1) = (8 + 12 + 14 + 5)/4 = 9.75

F(A2) = (9 + 10 + 11 + 10)/4 = 10

F(A3) = (2 + 4 + 9 + 22)/4 = 9.25

F(A4) = (12 + 14 + 10 + 1)/4 = 9.25

F(A5) = (15 + 6 + 7 + 14)/4 = 10.5

Обираємо альтернативу A5.

**Розрахунок за критерієм Байєса-Лапласа:**

Шукаємо значення альтернатив та обираємо найбільше

Формула:



F(A1) = 0.6 \* (0.1 \* 8 + 0.2 \* 12 + 0.6 \* 14 + 0.1 \* 5) = 7.26

F(A2) = 0.6 \* (0.1 \* 9 + 0.2 \* 10 + 0.6 \* 11 + 0.1 \* 10) = 6.3

F(A3) = 0.6 \* (0.1 \* 2 + 0.2 \* 4 + 0.6 \* 9 + 0.1 \* 22) = 5.16

F(A4) = 0.6 \* (0.1 \* 12 + 0.2 \* 14 + 0.6 \* 10 + 0.1 \* 1) = 6.06

F(A5) = 0.6 \* (0.1 \* 15 + 0.2 \* 6 + 0.6 \* 7 + 0.1 \* 14) = 4.98

Обираємо альтернативу A2.

**Розрахунок за критерієм Ходжа- Лемана:**

Знаходимо найбільшу суму критеріїв Байєса-Лапласа та Вальда

F(A1) = 0.6\* 5 + 0.6 \* (0.1 \* 8 + 0.2 \* 12 + 0.6 \* 4 + 0.1 \* 5) = 6.66

F(A2) = 0.6\* 9 + 0.6 \* (0.1 \* 9 + 0.2 \* 10 + 0.6 \* 11 + 0.1 \* 10) = 11.7

F(A3) = 0.6\* 2 + 0.6 \* (0.1 \* 2 + 0.2 \* 4 + 0.6 \* 9 + 0.1 \* 22) = 6.36

F(A4) = 0.6\* 1 + 0.6 \* (0.1 \* 12 + 0.2 \* 14 + 0.6 \* 10 + 0.1 \* 1) = 6.66

F(A5) = 0.6\* 6 + 0.6 \* (0.1 \* 15 + 0.2 \* 6 + 0.6 \* 7 + 0.1 \* 14) = 8.58

Обираємо альтернативу A2.

**Розрахунок за критерієм Вальда:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bj  Аi | S1 | S2 | S3 | S4 |
| А1 | 8 | 12 | 14 | 5 |
| А2 | 9 | 10 | 11 | 10 |
| А3 | 2 | 4 | 9 | 22 |
| А4 | 12 | 14 | 10 | 1 |
| А5 | 15 | 6 | 7 | 14 |

Шукаємо найкращий варіант з найгірших кожної альтернативи. Отже, обираємо альтернативу A2.

**Розрахунок за критерієм Севіджа:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bj  Аi | S1 | S2 | S3 | S4 |
| А1 | 8 | 12 | 14 | 5 |
| А2 | 9 | 10 | 11 | 10 |
| А3 | 2 | 4 | 9 | 22 |
| А4 | 12 | 14 | 10 | 1 |
| А5 | 15 | 6 | 7 | 14 |

Будуємо матрицю ризиків

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bj  Аi | S1 | S2 | S3 | S4 |
| А1 | 7 | 2 | 0 | 17 |
| А2 | 6 | 4 | 3 | 12 |
| А3 | 13 | 10 | 5 | 0 |
| А4 | 3 | 0 | 4 | 21 |
| А5 | 0 | 8 | 7 | 8 |

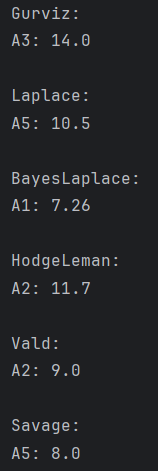
Для кожної альтернативи обираємо найбільший ризик, та серед них обираємо найменше значення.

Обираємо альтернативу A5.

**Лістинг програми:**

import openpyxl  
import numpy as np  
  
  
def get\_matrix\_out\_of\_sheet(sheet):  
 matrix = []  
  
 for row in sheet.iter\_rows(values\_only=True):  
 matrix.append(row)  
  
 return np.array(matrix)  
  
  
def gurviz(workbook):  
 pay\_matrix = get\_matrix\_out\_of\_sheet(workbook.worksheets[0])  
 l = get\_matrix\_out\_of\_sheet(workbook.worksheets[2])[0][0]  
  
 number\_of\_rows = pay\_matrix.shape[0]  
 number\_of\_columns = pay\_matrix.shape[1]  
  
 result = 0  
 index = 0  
  
 for row in range(number\_of\_rows):  
 min\_value = 100000  
 max\_value = -100000  
  
 for column in range(number\_of\_columns):  
 if pay\_matrix[row, column] < min\_value:  
 min\_value = pay\_matrix[row, column]  
  
 if pay\_matrix[row, column] > max\_value:  
 max\_value = pay\_matrix[row, column]  
  
 total\_sum = max\_value \* l + min\_value \* (1 - l)  
  
 if total\_sum > result:  
 result = total\_sum  
 index = row  
  
 print(f'A{index + 1}: {result}')  
  
  
def laplace(workbook):  
 pay\_matrix = get\_matrix\_out\_of\_sheet(workbook.worksheets[0])  
  
 number\_of\_rows = pay\_matrix.shape[0]  
 number\_of\_columns = pay\_matrix.shape[1]  
  
 result = 0  
 index = 0  
  
 for row in range(number\_of\_rows):  
 total\_sum = 0  
  
 for column in range(number\_of\_columns):  
 total\_sum += pay\_matrix[row, column]  
  
 total\_sum /= number\_of\_columns  
  
 if total\_sum > result:  
 result = total\_sum  
 index = row  
  
 print(f'A{index + 1}: {result}')  
  
  
def bayes\_laplace(workbook):  
 pay\_matrix = get\_matrix\_out\_of\_sheet(workbook.worksheets[0])  
 ps = get\_matrix\_out\_of\_sheet(workbook.worksheets[1])  
 l = get\_matrix\_out\_of\_sheet(workbook.worksheets[2])[0][0]  
  
 number\_of\_rows = pay\_matrix.shape[0]  
 number\_of\_columns = pay\_matrix.shape[1]  
  
 result = 0  
 index = 0  
  
 for row in range(number\_of\_rows):  
 total\_sum = 0  
  
 for column in range(number\_of\_columns):  
 total\_sum += pay\_matrix[row, column] \* ps[0, column] \* l  
  
 if total\_sum > result:  
 result = total\_sum  
 index = row  
  
 print(f'A{index + 1}: {result}')  
  
  
def hodge\_leman(workbook):  
 pay\_matrix = get\_matrix\_out\_of\_sheet(workbook.worksheets[0])  
 ps = get\_matrix\_out\_of\_sheet(workbook.worksheets[1])  
 l = get\_matrix\_out\_of\_sheet(workbook.worksheets[2])[0][0]  
  
 number\_of\_rows = pay\_matrix.shape[0]  
 number\_of\_columns = pay\_matrix.shape[1]  
  
 result = 0  
 index = 0  
  
 for row in range(number\_of\_rows):  
 total\_sum = 0  
 min\_value = 100000  
  
 for column in range(number\_of\_columns):  
 total\_sum += pay\_matrix[row, column] \* ps[0, column] \* l  
  
 if pay\_matrix[row, column] < min\_value:  
 min\_value = pay\_matrix[row, column]  
  
 total\_sum += min\_value \* l  
  
 if total\_sum > result:  
 result = total\_sum  
 index = row  
  
 print(f'A{index + 1}: {result}')  
  
  
def vald(workbook):  
 pay\_matrix = get\_matrix\_out\_of\_sheet(workbook.worksheets[0])  
  
 number\_of\_rows = pay\_matrix.shape[0]  
 number\_of\_columns = pay\_matrix.shape[1]  
  
 result = 0  
 index = 0  
  
 for row in range(number\_of\_rows):  
 min\_value = 100000  
  
 for column in range(number\_of\_columns):  
 if pay\_matrix[row, column] < min\_value:  
 min\_value = pay\_matrix[row, column]  
  
 if min\_value > result:  
 result = min\_value  
 index = row  
  
 print(f'A{index + 1}: {result}')  
  
  
def savage(workbook):  
 pay\_matrix = get\_matrix\_out\_of\_sheet(workbook.worksheets[0])  
  
 number\_of\_rows = pay\_matrix.shape[0]  
 number\_of\_columns = pay\_matrix.shape[1]  
  
 matrix = np.zeros((number\_of\_rows, number\_of\_columns))  
 values = np.zeros(number\_of\_rows)  
 indexes = np.zeros(number\_of\_rows)  
  
 result = 0  
  
 for row in range(number\_of\_rows):  
 max\_value = -100000  
  
 for column in range(number\_of\_columns):  
 if max\_value < pay\_matrix[row, column]:  
 max\_value = pay\_matrix[row, column]  
  
 values[row] = max\_value  
 indexes[row] = np.argmax(pay\_matrix[row])  
  
 for column in range(number\_of\_columns):  
 for row in range(number\_of\_rows):  
 index = np.where(indexes == column)[0]  
  
 if index.size != 0:  
 index = index[0]  
 matrix[row, column] = values[index] - pay\_matrix[row, column]  
 else:  
 matrix[row, column] = pay\_matrix[row, column]  
  
 values = np.zeros(number\_of\_rows)  
  
 for row in range(number\_of\_rows):  
 max\_value = -100000  
  
 for column in range(number\_of\_columns):  
 if max\_value < matrix[row, column]:  
 max\_value = matrix[row, column]  
  
 values[row] = max\_value  
  
 index = np.argmin(values)  
  
 result = values[index]  
  
 print(f'A{index + 1}: {result}')  
  
  
file\_path = 'lab4\_2.xlsx'  
  
workbook = openpyxl.load\_workbook(file\_path)  
  
print('Gurviz:')  
gurviz(workbook)  
  
print('\nLaplace:')  
laplace(workbook)  
  
print('\nBayesLaplace:')  
bayes\_laplace(workbook)  
  
print('\nHodgeLeman:')  
hodge\_leman(workbook)  
  
print('\nVald:')  
vald(workbook)  
  
print('\nSavage:')  
savage(workbook)  
  
workbook.close()

Результат:



**Висновок:** У ході лабораторної роботи я набув практичних навичок знаходження оптимального управлінського рішення в умовах невизначеності з використанням критеріїв Лапласа, Вальда, максимального оптимізму, Севіджа, Гурвіца та реалізував ці підходи програмно на мові Python.