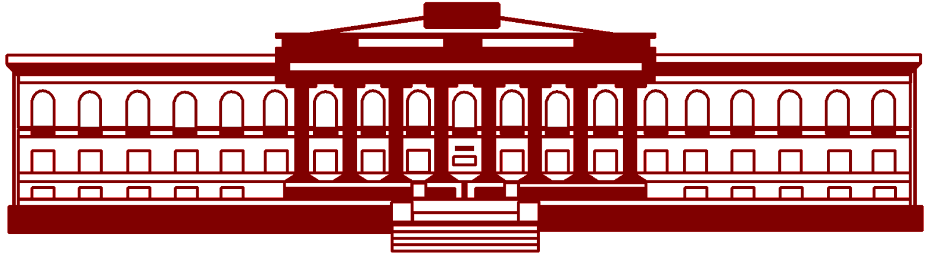
**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**імені ТАРАСА ШЕВЧЕНКА**

****

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра прикладних інформаційних систем**

**Звіт до лабораторної роботи №1**

**з курсу**

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

*студента 3 курсу*

*групи ПП-32*

*спеціальності 122 «Комп'ютерні науки»*

*ОП «Прикладне програмування»*

Сенька Кирила Валерійовича

*Викладач:*

Плескач В. Л., Білий Р.О.

**Київ – 2024**

**Тема:** Прийняття рішення в умовах повної визначеності.

**Мета роботи:** Дослідити методи прийняття рішення в умовах повної визначеності.

**Порядок виконання роботи:**

1. Вирішити задачі засобами MS Excel.
2. Запрограмувати розв’язок задачі.
3. Скласти звіт з лабораторної роботи.
4. Захистити роботу.

Хід роботи:

Скопіював першу таблицю у ексель файл

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерій Альтернатива | **К1** | **К2** | **К3** | **К4** |
| **А1** | 3 | 7 | 2 | 9 |
| **А2** | 8 | 3 | 6 | 7 |
| **А3** | 4 | 8 | 3 | 5 |
| **А4** | 9 | 6 | 5 | 4 |
| **Вес** | 8 | 9 | 6 | 7 |

Далі розрахував оцінки за допомогою формул

|  |  |  |  |
| --- | --- | --- | --- |
| 24 | 63 | 12 | 63 |
| 64 | 27 | 36 | 49 |
| 32 | 72 | 18 | 35 |
| 72 | 54 | 30 | 28 |

Далі підрахував сумарні оцінки та знайшов максимальну

|  |  |  |
| --- | --- | --- |
| 162 | Макс | 184 |
| 176 |  |  |
| 157 |  |  |
| 184 |  |  |

Скопіював другу таблицю у ексель файл. Також для себе відмітив, які критерії необхідно максимізувати, а які мінімізувати. Мінімізувати необхідно тільки другий критерій.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерий Альтернатива | **К1** | **К2** | **К3** | **К4** | **К5** |
| **А1** | 85 | 30 | 22 | 0,65 | 6 |
| **А2** | 60 | 20 | 10 | 0,6 | 7 |
| **А3** | 30 | 12 | 5 | 0,45 | 5 |
| **А4** | 75 | 24 | 13 | 0,7 | 8 |
| **А5** | 40 | 15 | 7 | 0,55 | 7 |
| **Вес** | 7 | 5 | 6 | 8 | 6 |

Нормалізував оцінки за прикладом

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0,8 | 0,333333 |
| 0,545455 | 0,555556 | 0,294118 | 0,6 | 0,666667 |
| 0 | 1 | 0 | 0 | 0 |
| 0,818182 | 0,333333 | 0,470588 | 1 | 1 |
| 0,181818 | 0,833333 | 0,117647 | 0,4 | 0,666667 |

Домножив оцінки на вагу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | 0 | 6 | 6,4 | 2 |
| 3,818182 | 2,777778 | 1,764706 | 4,8 | 4 |
| 0 | 5 | 0 | 0 | 0 |
| 5,727273 | 1,666667 | 2,823529 | 8 | 6 |
| 1,272727 | 4,166667 | 0,705882 | 3,2 | 4 |

Далі підрахував сумарні оцінки та знайшов максимальну

|  |  |  |
| --- | --- | --- |
| 21,4 | Макс | 24,21747 |
| 17,16067 |  |  |
| 5 |  |  |
| 24,21747 |  |  |
| 13,34528 |  |  |

Далі вирішив перше завдання за допомогою Python.

from typing import List, Tuple

# Функція для перевірки валідності даних

def validate\_data(candidates: List[List[int]], weights: List[int]) -> None:

    assert all(len(candidate) == len(weights) for candidate in candidates), "Кількість критеріїв у кандидатів повинна збігатися з кількістю вагових коефіцієнтів"

    assert len(candidates) > 0, "Список кандидатів не може бути порожнім"

    assert len(weights) > 0, "Список вагових коефіцієнтів не може бути порожнім"

# Функція для обчислення загальної оцінки для кожного кандидата

def calculate\_total\_scores(candidates: List[List[int]], weights: List[int]) -> List[float]:

    return [sum(score \* weight for score, weight in zip(candidate, weights)) for candidate in candidates]

# Функція для вибору найкращого кандидата

def select\_best\_candidate(candidates\_scores: List[float], candidates\_names: List[str]) -> Tuple[str, float]:

    best\_index = candidates\_scores.index(max(candidates\_scores))

    return candidates\_names[best\_index], candidates\_scores[best\_index]

# Основна функція

def main() -> None:

    # Дані кандидатів і ваги критеріїв

    candidates = [

        [3, 7, 2, 9],

        [8, 3, 6, 7],

        [4, 8, 3, 5],

        [9, 6, 5, 4]

    ]

    weights = [8, 9, 6, 7]

    candidates\_names = ['A1', 'A2', 'A3', 'A4']

    # Валідація даних

    validate\_data(candidates, weights)

    # Обчислення загальних оцінок

    total\_scores = calculate\_total\_scores(candidates, weights)

    # Вибір найкращого кандидата

    best\_candidate, best\_score = select\_best\_candidate(total\_scores, candidates\_names)

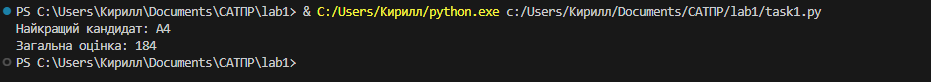
    # Виведення результату

    print(f"Найкращий кандидат: {best\_candidate}")

    print(f"Загальна оцінка: {best\_score}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()



Реалізував друге завдання.

from typing import List, Tuple

# Функція для валідації даних

def validate\_data(lawyers: List[List[float]], weights: List[float]) -> None:

    assert all(len(lawyer) == len(weights) for lawyer in lawyers), "Кількість критеріїв у адвокатів повинна збігатися з кількістю вагових коефіцієнтів"

    assert len(lawyers) > 0, "Список адвокатів не може бути порожнім"

    assert len(weights) > 0, "Список вагових коефіцієнтів не може бути порожнім"

    assert all(max(col) != min(col) for col in zip(\*lawyers)), "Не можна нормалізувати критерії з однаковими мінімальними та максимальними значеннями"

# Функція для знаходження мінімальних і максимальних значень по критеріях

def find\_min\_max(lawyers: List[List[float]]) -> Tuple[List[float], List[float]]:

    mins = [min(col) for col in zip(\*lawyers)]

    maxs = [max(col) for col in zip(\*lawyers)]

    return mins, maxs

# Функція для нормалізації оцінок

def normalize\_lawyers(lawyers: List[List[float]], mins: List[float], maxs: List[float]) -> List[List[float]]:

    normalized\_lawyers = []

    for lawyer in lawyers:

        normalized\_lawyer = [(x - min\_val) / (max\_val - min\_val) if max\_val != min\_val else 0 for x, min\_val, max\_val in zip(lawyer, mins, maxs)]

        normalized\_lawyers.append(normalized\_lawyer)

    return normalized\_lawyers

# Спеціальна нормалізація для другого критерію (стовпець 1)

def special\_normalization(normalized\_lawyers: List[List[float]], lawyers: List[List[float]], maxs: List[float], mins: List[float]) -> None:

    for i in range(len(lawyers)):

        normalized\_lawyers[i][1] = (maxs[1] - lawyers[i][1]) / (maxs[1] - mins[1])

# Функція для обчислення загальної оцінки для кожного адвоката

def calculate\_total\_scores(normalized\_lawyers: List[List[float]], weights: List[float]) -> List[float]:

    return [sum(value \* weight for value, weight in zip(lawyer, weights)) for lawyer in normalized\_lawyers]

# Функція для вибору найкращого адвоката

def select\_best\_lawyer(total\_scores: List[float], candidates\_names: List[str]) -> Tuple[str, float]:

    best\_index = total\_scores.index(max(total\_scores))

    return candidates\_names[best\_index], total\_scores[best\_index]

# Основна функція

def main() -> None:

    # Дані адвокатів і ваги критеріїв

    lawyers = [

        [85, 30, 22, 0.65, 6],

        [60, 20, 10, 0.6, 7],

        [30, 12, 5, 0.45, 5],

        [75, 24, 13, 0.7, 8],

        [40, 15, 7, 0.55, 7]

    ]

    weights = [7, 5, 6, 8, 6]

    candidates\_names = ['A1', 'A2', 'A3', 'A4', 'A5']

    # Валідація даних

    validate\_data(lawyers, weights)

    # Знаходження мінімальних і максимальних значень

    mins, maxs = find\_min\_max(lawyers)

    # Нормалізація оцінок

    normalized\_lawyers = normalize\_lawyers(lawyers, mins, maxs)

    # Спеціальна нормалізація для другого критерію

    special\_normalization(normalized\_lawyers, lawyers, maxs, mins)

    # Обчислення загальної оцінки

    total\_scores = calculate\_total\_scores(normalized\_lawyers, weights)

    # Визначення найкращого адвоката

    best\_lawyer, best\_score = select\_best\_lawyer(total\_scores, candidates\_names)

    # Виведення результату

    print(f"Найкращий адвокат: {best\_lawyer}")

    print(f"Загальна оцінка: {best\_score}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()



**Висновки**

У цій лабораторній я розглянув прийняття рішення в умовах повної визначеності з використанням Excel та реалізував прийняття рішення програмно.