**Міністерство освіти і науки України**

**Національний технічний університет України «КПІ» імені Ігоря Сікорського**

**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**

**з лабораторної роботи №4**

**з навчальної дисципліни «Вступ до технології Data Science»**

**Тема:**

**ДОСЛІДЖЕННЯ БАГАТОКРИТЕРІАЛЬНИХ МЕТОДІВ ПРИЙНЯТТЯ РІШЕНЬ**

**Виконав:**

Студент X курсу кафедри ОТ ФІОТ,

Навчальної групи ІА-73

Петренко П.П.

**Перевірив:**

Професор кафедри ОТ ФІОТ

Писарчук О.О.

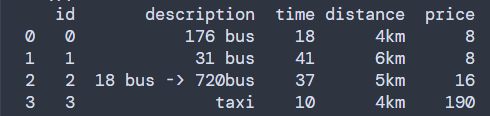
**Київ 2023**

**І. Мета:**

виявити дослідити та узагальнити особливості багатокритеріальних методів прийняття рішень з використанням спеціалізованих пакетів мови програмування Python.

**ІІ. Завдання:**

| **Варіант** (порядковий номер в списку групи) | **Технічні умови завдання** |
| --- | --- |
| 3,18 | Розробити програмний скрипт, що реалізує багатокритеріальне оцінювання ефективності ***маршрутів ВАШОГО переміщення із дому до КПІ***. Формування показників та критеріїв ефективності, синтез багатокритеріальної оптимізаційної моделі здійснити самостійно. |

****

**Рисунок 1. Датасет**

**time distance price total\_score total\_rank importance**

**0 18 4km 8 2.741935 10.8 2**

**1 41 6km 8 1.000000 20.6 1**

**2 37 5km 16 1.585076 21.1 0**

**3 10 4km 190 2.000000 62.2 0**

**Таблиця 1. Результат обробки**

**Метод нормалізації**

Нормалізація - це процес приведення даних до стандартного вигляду або масштабу для спрощення порівнянь. У багатокритеріальному оцінюванні, нормалізація дозволяє привести значення різних критеріїв до одного і того ж діапазону, щоб вони могли бути коректно порівняні.

Значення часу нормалізовано до діапазону [0, 1], де 0 - це найбільше значення, а 1 - найменше.

Відстань також нормалізована до діапазону [0, 1], враховуючи той факт, що менша відстань вважається кращою.

Ціна нормалізована до діапазону [0, 1], де 0 - це найбільше значення, а 1 - найменше.

Нормалізація стандартизує значення, забезпечуючи їхню подальшу обробку на одній шкалі.

total\_score - результат нормалізації

**Метод ранжування**

Метод ранжування використовується для впорядкування альтернатив за допомогою порівнянь за кількома критеріями. У багатокритеріальному оцінюванні, альтернативи отримують рейтинги відповідно до їхньої ефективності за кожним критерієм.

Кожна альтернатива отримує загальний рейтинг на основі попередньо визначених ваг для кожного критерію.

Ранжування дозволяє визначити порядок альтернатив від найбільшої до найменш ефективної, але не дає точного значення, наскільки одна альтернатива краща за іншу.

total\_rank - методом ранжування (див. таблицю 1)

**Метод порогових значень:**

Метод порогових значень визначає певні критерії, які повинні бути виконані для того, щоб альтернатива вважалася прийнятною. Якщо альтернатива задовольняє усі критерії, вона вважається ефективною.

Результати:

Значення "1" в стовпчику importance позначає, що альтернатива задовольняє усі критерії.

Відмінності в результаті:

Цей метод вказує, чи альтернатива відповідає визначеним пороговим значенням для кожного критерію.

rand\_importance - методом відмінності

**total\_score: Сума нормалізованих значень критеріїв, що використовуються для порівнянь.**

**total\_rank: Загальний рейтинг альтернативи на основі ваг критеріїв.**

**importance: Показує, скільки критеріїв задовольняє кожна альтернатива.**

**ІІІ. Довести адекватність сформованих моделей та працездатність розробленого скріпта.**

**Розроблений код повинен бути раціональним та відповідати вимогам до чистого коду.**

**III. Висновок:**

виявив дослідив та узагальнив особливості багатокритеріальних методів прийняття рішень з використанням спеціалізованих пакетів мови програмування Python.

**import pandas as pd**

**def normalization(data):**

**data['normalized\_time'] = 1 - (data['time'] - data['time'].min()) / (data['time'].max() - data['time'].min())**

**data['normalized\_distance'] = 1 - (data['distance'].str.extract('(\d+)').astype(int) - data['distance'].str.extract('(\d+)').astype(int).min()) / (data['distance'].str.extract('(\d+)').astype(int).max() - data['distance'].str.extract('(\d+)').astype(int).min())**

**data['normalized\_price'] = 1 - (data['price'] - data['price'].min()) / (data['price'].max() - data['price'].min())**

**data['total\_score'] = data['normalized\_time'] + data['normalized\_distance'] + data['normalized\_price']**

**return data**

**def ranking(data):**

**weights = {'time': 0.4, 'distance': 0.3, 'price': 0.3}**

**data['total\_rank'] = (data['time'] \* weights['time'] +**

**data['distance'].str.extract('(\d+)', expand=False).astype(int) \* weights['distance'] +**

**data['price'] \* weights['price'])**

**return data**

**def electre(data):**

**thresholds = {'time': 30, 'distance': 5, 'price': 150}**

**data['distance\_to\_threshold\_time'] = abs(data['time'] - thresholds['time'])**

**data['distance\_to\_threshold\_distance'] = abs(data['distance'].str.extract('(\d+)').astype(int) - thresholds['distance'])**

**data['distance\_to\_threshold\_price'] = abs(data['price'] - thresholds['price'])**

**relation\_matrix = pd.DataFrame(index=data.index, columns=data.index)**

**for i in range(len(data)):**

**for j in range(len(data)):**

**if i != j:**

**relation\_matrix.iloc[i, j] = (**

**(data['distance\_to\_threshold\_time'].iloc[i] >= data['distance\_to\_threshold\_time'].iloc[j]) and**

**(data['distance\_to\_threshold\_distance'].iloc[i] >= data['distance\_to\_threshold\_distance'].iloc[j]) and**

**(data['distance\_to\_threshold\_price'].iloc[i] >= data['distance\_to\_threshold\_price'].iloc[j])**

**)**

**data['importance'] = relation\_matrix.sum(axis=1)**

**return data**

**data = pd.read\_csv('varaint\_3.csv')**

**print(normalization(data))**

**print(ranking(data))**

**print(electre(data))**

**print(data[['time', 'distance', 'price', 'total\_score', 'total\_rank', 'importance']])**

Виконав: студент Петренко П.П.