Міністерство освіти і науки України
Національний технічний університет України «КПІ» імені Ігоря Сікорського
Кафедра обчислювальної техніки ФІОТ

ЗВІТ
з лабораторної роботи №3
з навчальної дисципліни «Data Science Technology»

Тема:

МАКЕТ ІНТЕЛЕКТУАЛЬНОЇ ERP СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

Виконав:
Студент 3 курсу кафедри ОТ ФІОТ,
Навчальної групи ІМ-13
Тавлуй Д. О.

Перевірив:
Професор кафедри ОТ ФІОТ
Писарчук О.О.

Київ 2023

I. Мета:
Виявити дослідити та узагальнити принципи формалізації задач, синтезу математичних моделей для автоматизації процесів підтримки прийняття рішень в інтелектуальних ERP системах: програмування обмежень – CP-SAT; багатокритеріальні задачі – Multicriteria decision analysis.
II. Завдання:
Для визначення можливості автоматизації бізнес процесів, що реалізовані в компанії замовника Вам пропонується розробити макет програмної реалізації мовою Python обчислювального алгоритму ERP системи підтримки прийняття рішень за умов:
II рівень складності 8 балів
Я обрав варіант 10

Розробити програмний скрипт, що реалізує багатокритеріальне оцінювання ефективності ***позашляховиків різних виробників***. Формування показників та критеріїв ефективності, синтез багатокритеріальної оптимізаційної моделі здійснити самостійно.

III. Результати виконання лабораторної роботи.

Перш за все, я створив таблицю у якій вказав автомобілі які будуть порівнюватись, а також критерії оцінювання. Всього позашляховиків буде 11. Обрав основні характеристики, які зазначені у самій таблиці. Серед них є як максимізовані (максимальна швидкість, об'єм двигуна, потужність, об'єм багажника, об'єм бака, кліренс), так і мінімізовані (ціна в доларах, розхід палива, розгін 0-100, маса).

Ознака	MB G63	Lexus LX600	Volvo XC90	Toyota LC J300	Chevrolet Tahoe	Jeep Grand Cherokee	Audi SQ8	Kia Sportage	Ford Explorer	Subaru Forester	Bentley Bentayga	Критерій
Ціна в $	269545	139167	57793	79724	81384	67386	92235	37598	38359	55340	282133	мін
Розхід палива	13,1	12,1	8,3	12,1	12,6	13	8,3	7,6	12,4	7,4	14,3	мін
Макс шв	220	210	230	210	180	225	250	197	183	207	306	мах
Розгін 0-100	4,5	6,9	6,5	6,8	8	7,3	4,8	9,4	8,3	9,5	3,9	мін
Об'єм двигуна	4	3,4	2	3,4	5,3	5,7	4	2,5	3,5	2,5	6	мах
Потужність	585	415	310	415	343	347	422	190	249	185	635	мах
Багажник	667	2052	721	1131	722	782	605	540	595	505	484	мах
Маса	2560	2525	2140	2485	2715	2266	2430	1567	2265	1676	2508	мін
Бак	100	110	71	110	98	94	85	54	70	63	85	мах
Кліренс	266	200	238	230	216	218	254	181	211	220	245	мах

Як можна побачити, зеленим позначено найкращі показники, світло-зеленим непогані, а червоним найгірші.
Для мене найголовнішим при виборі позашляховика є його потужність та розхід палива.
З цього випливає, що найбільш оптимальним є варіант Audi SQ8, а за нею Toyota Lad Cruiser J300. А ось Kia Sportage та Subaru Forester виявились найгіршими з запропонованих.

Результати архітектурного проектування та їх опис

Я обрав монолітну архітектуру, адже програма не дуже обширна. Увесь код знаходиться у файлі main.py.

Опис структури проекту програми

Усі функції прописані у main.py файлі, у якому ж і виконується функція з вирахуванням оптимального варіанту.

Результати роботи програми відповідно до завдання

Таблиця виведена у консоль:

```
          Ознака    MB G63  ...  Bentley Bentayga  Критерій
0       Ціна в $  269545.0  ...          282133.0       мін
1   Розхід палива      13.1  ...              14.3       мін
2        Макс шв     220.0  ...             306.0       мах
3    Розгін 0-100       4.5  ...               3.9       мін
4   Об'єм двигуна       4.0  ...               6.0       мах
5      Потужність     585.0  ...             635.0       мах
6       Багажник     667.0  ...             484.0       мах
7           Маса    2560.0  ...            2508.0       мін
8            Бак     100.0  ...              85.0       мах
9        Кліренс     266.0  ...             245.0       мах
```

(так як уся таблиця не влазить у консоль PyCharm, вона показана скороченою)

Обраний позашляховик програмою:

```
Оптимальний позашляховик: Audi SQ8
```

Рейтинг позашляховиків за інтегрованою оцінкою

```
Рейтинг позашляховиків:
1. Audi SQ8: 1.084938798664334
2. Toyota LC J300: 1.0902649303657161
3. Jeep Grand Cherokee: 1.0930355226216175
4. Lexus LX600: 1.0946701593909947
5. Volvo XC90: 1.0978202063478077
6. Chevrolet Tahoe: 1.0994728740432036
7. MB G63: 1.1050378388714754
8. Bentley Bentayga: 1.107631600884117
9. Ford Explorer: 1.1089296643729782
10. Subaru Forester: 1.115812457522824
11. Kia Sportage: 1.1170903187059231
```

Як можна побачити, мої власні очікування повністю підтвердилися: Audi SQ8 виявилась найкращим варіантом, а Kia Sportage – найгіршим.

Програмний код, що забезпечує отримання результату

main.py

```
import pandas as pd
import numpy as np


def file_parsing(data_name, sample_data):
    values = sample_data[data_name].astype(str).str.replace(',',
'.').astype(float)
    return values.to_numpy()


def matrix_generation(file_name):
    sample_data = pd.read_excel(file_name)
    print(sample_data)
    line_sample_data, column_sample_data = sample_data.shape
    line_column_matrix = np.zeros((line_sample_data, column_sample_data -
2))

    for i in range(1, column_sample_data - 1):
        column_matrix = file_parsing(sample_data.columns[i], sample_data)
        line_column_matrix[:, i - 1] = column_matrix

    return line_column_matrix


def matrix_adapter(line_column_matrix, line):
    return line_column_matrix[line, :]


def calc_optimal(file, weight_1, weight_2, weight_3, weight_4, weight_5,
weight_6, weight_7, weight_8, weight_9, weight_10):
    line_column_matrix = matrix_generation(file)
    column_matrix = np.shape(line_column_matrix)
    Integro = np.zeros((column_matrix[1]))

    price = matrix_adapter(line_column_matrix, 0)
    fuel_consumption = matrix_adapter(line_column_matrix, 1)
    max_speed = matrix_adapter(line_column_matrix, 2)
    racing = matrix_adapter(line_column_matrix, 3)
    engine_capacity = matrix_adapter(line_column_matrix, 4)
    power = matrix_adapter(line_column_matrix, 5)
    trunk = matrix_adapter(line_column_matrix, 6)
    mass = matrix_adapter(line_column_matrix, 7)
    fuel_tank = matrix_adapter(line_column_matrix, 8)
    clearance = matrix_adapter(line_column_matrix, 9)

    price_normalized = np.zeros((column_matrix[1]))
    fuel_consumption_normalized = np.zeros((column_matrix[1]))
    max_speed_normalized = np.zeros((column_matrix[1]))
    racing_normalized = np.zeros((column_matrix[1]))
    engine_capacity_normalized = np.zeros((column_matrix[1]))
    power_normalized = np.zeros((column_matrix[1]))
    trunk_normalized = np.zeros((column_matrix[1]))
    mass_normalized = np.zeros((column_matrix[1]))
    fuel_tank_normalized = np.zeros((column_matrix[1]))
```

```
    clearance_normalized = np.zeros((column_matrix[1]))

    weights_normalization_sum = weight_1 + weight_2 + weight_3 + weight_4 +
weight_5 + weight_6 + weight_7 + weight_8 + weight_9 + weight_10

    weight_1_normalized = weight_1 / weights_normalization_sum
    weight_2_normalized = weight_2 / weights_normalization_sum
    weight_3_normalized = weight_3 / weights_normalization_sum
    weight_4_normalized = weight_4 / weights_normalization_sum
    weight_5_normalized = weight_5 / weights_normalization_sum
    weight_6_normalized = weight_6 / weights_normalization_sum
    weight_7_normalized = weight_7 / weights_normalization_sum
    weight_8_normalized = weight_8 / weights_normalization_sum
    weight_9_normalized = weight_9 / weights_normalization_sum
    weight_10_normalized = weight_10 / weights_normalization_sum

    sum_price = sum_fuel_consumption = sum_max_speed = sum_racing =
sum_engine_capacity = sum_power = sum_trunk = sum_mass = sum_fuel_tank =
sum_clearance = 0

    for i in range(column_matrix[1]):
        sum_price += price[i]
        sum_fuel_consumption += fuel_consumption[i]
        sum_max_speed += (1 / max_speed[i])
        sum_racing += racing[i]
        sum_engine_capacity += (1 / engine_capacity[i])
        sum_power += (1 / power[i])
        sum_trunk += (1 / trunk[i])
        sum_mass += mass[i]
        sum_fuel_tank += (1 / fuel_tank[i])
        sum_clearance += (1 / clearance[i])

    for i in range(column_matrix[1]):
        price_normalized[i] = price[i] / sum_price
        fuel_consumption_normalized[i] = fuel_consumption[i] /
sum_fuel_consumption
        max_speed_normalized[i] = (1 / max_speed[i]) / sum_max_speed
        racing_normalized[i] = racing[i] / sum_racing
        engine_capacity_normalized[i] = (1 / engine_capacity[i]) /
sum_engine_capacity
        power_normalized[i] = (1 / power[i]) / sum_power
        trunk_normalized[i] = (1 / trunk[i]) / sum_trunk
        mass_normalized[i] = mass[i] / sum_mass
        fuel_tank_normalized[i] = (1 / fuel_tank[i]) / sum_fuel_tank
        clearance_normalized[i] = (1 / clearance[i]) / sum_clearance

        Integro[i] = ((weight_1_normalized * (1 - price_normalized[i]) **
(-1)) +
                      (weight_2_normalized * (1 -
fuel_consumption_normalized[i]) ** (-1)) +
                      (weight_3_normalized * (1 - max_speed_normalized[i])
** (-1)) +
                      (weight_4_normalized * (1 - racing_normalized[i]) **
(-1)) +
                      (weight_5_normalized * (1 -
engine_capacity_normalized[i]) ** (-1)) +
                      (weight_6_normalized * (1 - power_normalized[i]) **
(-1)) +
                      (weight_7_normalized * (1 - trunk_normalized[i]) **
(-1)) +
                      (weight_8_normalized * (1 - mass_normalized[i]) ** (-
```

```
1)) +
                        (weight_9_normalized * (1 - fuel_tank_normalized[i])
** (-1)) +
                        (weight_10_normalized * (1 - clearance_normalized[i])
** (-1)))

    min = 10000

    optimal_index = 0

    for i in range(column_matrix[1]):
        if min > Integro[i]:
            min = Integro[i]
            optimal_index = i
    sample_data = pd.read_excel(file)
    integro_sorted = sorted(Integro)

    print('\nОптимальний позашляховик:', sample_data.columns[optimal_index
+ 1])
    print('\nРейтинг позашляховиків:')
    for i in range(len(integro_sorted)):
        element_index = list(Integro).index(integro_sorted[i])
        print(f'{i + 1}. {sample_data.columns[element_index + 1]}:
{integro_sorted[i]}')

    return


if __name__ == '__main__':
    file = 'lab3.xlsx'

    weight_1 = weight_4 = weight_5 = weight_7 = weight_10 = weight_8 =
weight_3 = weight_9 = 1
    weight_2 = weight_6 = 2

    calc_optimal(file, weight_1, weight_2, weight_3, weight_4, weight_5,
weight_6, weight_7, weight_8, weight_9, weight_10)
```

Висновки

Під час виконання лабораторної роботи я розробив програму для виявлення найоптимальнішого позашляховика в інтелектуальних ERP системах підтримки прийняття рішень. Програма успішно виконує обчислення та нормалізацію критеріїв, а також обчислює інтегральні показники для варіантів автомобілів. Виведені результати у вигляді інтегральних показників дозволяють визначити найкращий варіант для вибору позашляховика з урахуванням вказаних критеріїв. Також я самостійно визначив критерії які найбільш важливі з використанням вищого вагового коефіцієнту.