Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №3

з дисципліни «Вступ до технології Data Science»

на тему

«МАКЕТ ІНТЕЛЕКТУАЛЬНОЇ ERP СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ»

Виконала:
студентка групи ІС-12
Павлова Софія

Перевірив:
Баран Д. Р.

Київ – 2023

1. Постановка задачі

Мета роботи:

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.

Завдання II рівня:

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

***** Додаткове завдання до рівня II:**

2. Розробити програмний скрипт, що забезпечує розв'язок задачі лінійного програмування для умов, зазначених в Лекції_6 з використанням інструментів бібліотеки Google OR-Tools.

3. Здійснити опис практичної інтерпретації задачі лінійного програмування, що розв'язана Вами для конкретної прикладної галузі (із власного практичного досвіду, або з аналізу інформаційних джерел). Опис подати у протоколі.

2. Виконання

2.1. Програмний скрипт, що реалізує багатокритеріальне оцінювання ефективності позашляховиків різних виробників

Синтез багатокритеріальної моделі

Сформулюємо задачу багатокритеріального оцінювання ефективності позашляховиків:

За *критерії ефективності* приймемо:

1. Вартість автомобіля → **min**
2. Кількість пробігу → **min**
3. Максимальна швидкість → **max**
4. Час розгону від 0 км/год до 100 км/год → **max**
5. Витрата пального в міському режимі на 100 км → **min**
6. Робочий об’єм двигуна → **max**
7. Об’єм паливного баку → **max**
8. Об’єм багажника без відкидання заднього ряду сидінь → **max**
9. Висота → **max**
10. Максимальне навантаження на дах → **max**

Розглянемо наступні *позашляховики різних виробників*:

1. Land Rover Defender
2. Toyota Land Cruiser
3. Jeep Wrangler
4. Ford Bronco
5. Mercedes-Benz G-Class
6. Nissan Patrol
7. Chevrolet Tahoe
8. BMW X5

9. Porsche Cayenne

10. Subaru Outback

Така модель може бути використана для вибору позашляховика відповідно до конкретних потреб користувача, забезпечуючи більше об'єктивний та комплексний підхід до визначення ефективності автомобіля.

Заповнимо сформульовану модель даними. Представимо її у вигляді таблиці, де позашляховики будуть відповідати стовпцям, а критерії ефективності – рядкам.

Використаємо для кожного рядка умовне форматування за *зелено-червоною кольоровою шкалою* так, щоб *найкращі* значення відповідно до зазначеного критерію були виділені *зеленим* кольором, а *найгірші – червоним*.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	Land Rover Defender	Toyota Land Cruiser	Jeep Wrangler	Ford Bronco	Mercedes-Benz G-Class	Nissan Patrol	Chevrolet Tahoe	BMW X5	Porsche Cayenne	Subaru Outback	Критерій	
2	Вартість автомобіля, $	84500	33900	47700	82000	189900	15500	84000	33000	75000	19800	min	
3	Кількість пробігу, км	94000	235000	92000	6000	47000	238000	60000	58000	76000	84000	min	
4	Максимальна швидкість, км/год	175	175	180	160	210	210	180	243	248	207	max	
5	Час розгону, с	9,8	9,9	8,4	10	6,4	7,1	8,5	5,5	6	9,5	min	
6	Витрата пального, л/100км	8,2	9,6	10,6	15,7	9,6	20,5	15,7	10,9	13,8	9,3	min	
7	Об'єм двигуна, л	2,997	2,755	1,995	2,998	2,925	5,552	5,327	2,998	2,995	2,498	max	
8	Об'єм баку, л	89	87	70	121	75	100	98	83	75	63	max	
9	Об'єм багажника, л	231	259	206	917	487	550	479	650	772	537	max	
10	Висота, мм	2160	1895	2459	1975	1969	2740	1955	1745	1698	1670	max	
11	Навантаження на дах, кг	118	50	90	68	200	70	75	70	100	75	max	
12													
13													

Рисунок 1 – Таблиця багатокритеріального аналізу

Багатокритеріальне оцінювання ефективності

Так, як наша задача є багатокритеріальною і має критерії як на максимум, так і на мінімум – це однозначно приведе до конфлікту між критеріями максимізації і мінімізації. Тому для вирішення подібної задачі використаємо *Алгоритм Вороніна*.

Основна ідея алгоритму полягає в тому, щоб нормалізувати кожен критерій, перетворивши їх так, щоб усі критерії були максимізовані або мінімізовані, і потім об'єднати їх для отримання інтегрованої оцінки.

Оптимальним вважається рішення з *найменшою інтегральною оцінкою*, так як ми шукаємо рішення, оптимальне за всіма критеріями.

На вхід алгоритм приймає вхідну матрицю з критеріями ефективності та ваговими коефіцієнтами цих критеріїв. Для нашої задачі приймемо, що всі критерії ефективності рівнозначні.

Імпорт даних

Для програмної реалізації алгоритму спочатку *імпортуємо дані* та представимо їх у вигляді матриці.

Лістинг коду:

```
import pandas as pd
import numpy as np

def file_parsing (data_name, df):
    for name, values in df[[data_name]].items():
        values
    n_df = int(len(values))
    df_real = np.zeros((n_df))
    for i in range(n_df):
        df_real[i] = values[i].replace(',', '.')
    return df_real, n_df

def matrix_generation (file_name):
    df = pd.read_excel(file_name)
    print(df)
    line_df = int(df.shape[0])
    column_df = int(df.shape[1])
    line_column_matrix = np.zeros(((line_df), (column_df - 2)))
    title_df = df.columns
    for i in range(1, (column_df - 1), 1):
        column_matrix, n_df = file_parsing(title_df[i], df)
        for j in range(len(column_matrix)):
```

```
            line_column_matrix[j, (i - 1)] = column_matrix[j]
    return line_column_matrix, title_df, n_df

def multicriteria_optimization(file_name, vag_koefs):
    # Вхідні дані
    line_column_matrix, title_df, n_df = matrix_generation(file_name)
    column_matrix = np.shape(line_column_matrix)
```

Результат:

У результаті виконання програми отримуємо матрицю.

	0	Land Rover Defender	...	Subaru Outback	Критерій
0	Вартість автомобіля, $	84500	...	19800	min
1	Кількість пробігу, км	94000	...	84000	min
2	Максимальна швидкість, км/год	175	...	207	max
3	Час розгону, с	9,8	...	9,5	min
4	Витрата пального, л/100км	8,2	...	9,3	min
5	Об'єм двигуна, л	2,997	...	2,498	max
6	Об'єм баку, л	89	...	63	max
7	Об'єм багажника, л	231	...	537	max
8	Висота, мм	2160	...	1670	max
9	Навантаження на дах, кг	118	...	75	max

[10 rows x 12 columns]

Рисунок 2 – Матриця значень критеріїв оптимальності

Нормалізація критеріїв

Зчитаємо критерії та перейдемо до наступного етапу – *нормалізації вхідних даних*. Оскільки значення критеріїв оптимальності дуже різні, їх необхідно нормалізувати для кращого результату аналізу. Нормалізуємо значення матриці від 0 до 1.

Лістинг коду:

```
def multicriteria_optimization(file_name, vag_koefs):
    [...]
    # Зчитування критеріїв
    criteria_list = [matrix_adapter(line_column_matrix, i) for i in range(n_df)]

    # Нормалізація критеріїв
    sum_criteria = np.zeros(n_df)
    normalized_criteria = np.zeros((n_df, column_matrix[1]))
    integr_score = np.zeros((column_matrix[1]))
    for j in range(n_df):
        for i in range(column_matrix[1]):
            sum_criteria[j] += (1 / criteria_list[j][i]) if (j == 2 or j == 5 or (j >= 6
and j <= 9)) else \
            criteria_list[j][i]
```

```
    for i in range(column_matrix[1]):
        for j in range(n_df):
            normalized_criteria[j, i] = (1 / criteria_list[j][i]) / sum_criteria[j] if (j
== 2 or j == 5 or (6 <= j <= 9)) else \
            criteria_list[j][i] / sum_criteria[j]
```

Результат:

У результаті отримаємо нормалізовані значення критеріїв.

```
Нормалізовані критерії
[[0.12701037 0.05095446 0.07169698 0.12325267 0.28543514 0.02329776
  0.12625883 0.04960168 0.1127311  0.02976101]
 [0.09494949 0.23737374 0.09292929 0.00606061 0.04747475 0.24040404
  0.06060606 0.05858586 0.07676768 0.08484848]
 [0.11140719 0.11140719 0.10831254 0.12185161 0.09283932 0.09283932
  0.10831254 0.08023151 0.07861394 0.09418482]
 [0.12083847 0.12207152 0.10357583 0.12330456 0.07891492 0.08754624
  0.10480888 0.06781751 0.07398274 0.11713933]
 [0.06618241 0.07748184 0.08555287 0.12671509 0.07748184 0.16545601
  0.12671509 0.08797417 0.11138015 0.07506053]
 [0.10083814 0.10969579 0.15148466 0.1008045  0.10332031 0.05443298
  0.0567321  0.1008045  0.10090548 0.12098154]
 [0.09357655 0.09572773 0.11897589 0.06882903 0.11104417 0.08328313
  0.08498278 0.10034112 0.11104417 0.13219544]
 [0.17462607 0.15574757 0.19581855 0.04398977 0.08283085 0.07334295
  0.08421424 0.06205942 0.0522521  0.07511848]
 [0.0916975  0.10452063 0.08054762 0.10028689 0.10059248 0.07228708
  0.10131284 0.11350521 0.116647   0.11860275]
 [0.06805267 0.1606043  0.08922461 0.1180914  0.04015108 0.11471736
  0.10706954 0.11471736 0.08030215 0.10706954]]
```

Рисунок 3 – Нормалізовані значення для кожного з критеріїв

Нормалізація вагових коефіцієнтів

Для даної задачі цей етап необов’язковий, утім, якщо ми захочемо надати перевагу одному з критеріїв оптимальності цей етап буде обов’язковим. Тому реалізуємо дану нормалізацію для більшої універсальності скрипту.

Лістинг коду:

```
def multicriteria_optimization(file_name, vag_koefs):
    [...]
    # Нормалізація вагових коефіцієнтів
    normalization_koef = sum(vag_koefs)
    normalized_vag_koefs = np.array(vag_koefs) / normalization_koef
    print('\nНормалізовані вагові коефіцієнти')
    print(normalized_vag_koefs)
```

Результат:

У результаті отримаємо нормалізовані значення вагових коефіцієнтів.

```
Нормалізовані вагові коефіцієнти
[0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1]
```

Рисунок 4 – Нормалізовані значення для вагомості кожного обмеження

Розрахунок інтегральної оцінки

Інтегральна оцінка розраховується за формулою:

$$Y(y_0) = \sum_{l=1}^{b} \gamma_{0l} (1 - y_{0l})^{-1} \to \min$$

Лістинг коду:

```
def multicriteria_optimization(file_name, vag_koefs):
    [...]
    for i in range(column_matrix[1]):
        [...]
        # Розрахунок інтегральної оцінки
        integr_score[i] = np.sum(vag_koefs * (1 - normalized_criteria[:, i]) ** (-1))
    print('\nНормалізовані критерії')
    print(normalized_criteria)

    # Генерація оптимального рішення
    min_integr_score_index = np.argmin(integr_score)
    optimal_vehicle = title_df[min_integr_score_index + 1]
    print('\nІнтегрована оцінка:')
    print(integr_score)
    print('\nОптимальний позашляховик:', optimal_vehicle)
    return
```

Результат:

У результаті отримаємо наступні значення інтегрованої оцінки.

```
Інтегрована оцінка:
[11.18502771 11.43496441 11.2528955  11.04859089 11.20542592 11.17195468
 11.07029113 10.918288   11.01240268 11.06685603]

Оптимальний позашляховик: BMW X5
```

Рисунок 5 – Отримані значення інтегральної оцінки за алгоритмом Вороніна та оптимальний розв’язок

Перевірка реалістичності розв’язку

Переконаємось, що в таблиці позашляховик *BMW X5* має багато *зелених* та *жовтих* полів, що означає *найкращі значення* критеріїв і мало *червоних*, тобто мало *небажаних значень* критеріїв.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	Land Rover Defender	Toyota Land Cruiser	Jeep Wrangler	Ford Bronco	Mercedes-Benz G-Class	Nissan Patrol	Chevrolet Tahoe	BMW X5	Porsche Cayenne	Subaru Outback	Критерій	
2	Вартість автомобіля, $	84500	33900	47700	82000	189900	15500	84000	33000	75000	19800	min	
3	Кількість пробігу, км	94000	235000	92000	6000	47000	238000	60000	58000	76000	84000	min	
4	Максимальна швидкість, км/год	175	175	180	160	210	210	180	243	248	207	max	
5	Час розгону, с	9,8	9,9	8,4	10	6,4	7,1	8,5	5,5	6	9,5	min	
6	Витрата пального, л/100км	8,2	9,6	10,6	15,7	9,6	20,5	15,7	10,9	13,8	9,3	min	
7	Об'єм двигуна, л	2,997	2,755	1,995	2,998	2,925	5,552	5,327	2,998	2,995	2,498	max	
8	Об'єм баку, л	89	87	70	121	75	100	98	83	75	63	max	
9	Об'єм багажника, л	231	259	206	917	487	550	479	650	772	537	max	
10	Висота, мм	2160	1895	2459	1975	1969	2740	1955	1745	1698	1670	max	
11	Навантаження на дах, кг	118	50	90	68	200	70	75	70	100	75	max	
12													
13													

Рисунок 6 – Нормалізовані значення для вагомості кожного обмеження

Однак така оцінка правильності результату відносна, тому переконаємось, що розроблений скрипт знаходить оптимальний розв’язок у однозначній задачі.

Для цього приймемо тільки перший критерій ефективності. Тобто вагові коефіцієнти всіх обмежень будуть дорівнювати 0, крім першого.

Для цього використаємо дану частину скрипту при виклику функції оптимізації.

Лістинг коду:

```
vag_koefs = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
multicriteria_optimization(file_name, vag_koefs)
```

Результат:

У результаті отримаємо оптимальний розв’язок – позашляховик *Nissan Patrol*.

```
Інтегрована оцінка:
[1.14548898 1.05369021 1.07723446 1.14057946 1.39945309 1.02385349
 1.1445037  1.05219042 1.12705404 1.0306739 ]

Оптимальний позашляховик: Nissan Patrol
```

Рисунок 7 – Результат перевірки скрипту

Переконаємось, що *Nissan Patrol* дійсно має найменшу вартість.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	Land Rover Defender	Toyota Land Cruiser	Jeep Wrangler	Ford Bronco	Mercedes-Benz G-Class	Nissan Patrol	Chevrolet Tahoe	BMW X5	Porsche Cayenne	Subaru Outback	Критерій	
2	Вартість автомобіля, $	84500	33900	47700	82000	189900	15500	84000	33000	75000	19800	min	
3	Кількість пробігу, км	94000	235000	92000	6000	47000	238000	60000	58000	76000	84000	min	
4	Максимальна швидкість, км/год	175	175	180	160	210	210	180	243	248	207	max	
5	Час розгону, с	9,8	9,9	8,4	10	6,4	7,1	8,5	5,5	6	9,5	min	
6	Витрата пального, л/100км	8,2	9,6	10,6	15,7	9,6	20,5	15,7	10,9	13,8	9,3	min	
7	Об'єм двигуна, л	2,997	2,755	1,995	2,998	2,925	5,552	5,327	2,998	2,995	2,498	max	
8	Об'єм баку, л	89	87	70	121	75	100	98	83	75	63	max	
9	Об'єм багажника, л	231	259	206	917	487	550	479	650	772	537	max	
10	Висота, мм	2160	1895	2459	1975	1969	2740	1955	1745	1698	1670	max	
11	Навантаження на дах, кг	118	50	90	68	200	70	75	70	100	75	max	
12													
13													

Рисунок 8 – Підтвердження працездатності розробленої моделі та скрипту

2.2. Скрипт, що забезпечує розв'язок задачі лінійного програмування з Лекції_6

Формулювання задачі в Лекції_6 виконано наступним чином:

Нехай необхідно мінімізувати цільову функцію $Q = -2X_1 - 2X_2$,
за обмежень:

$$1.5X_1 + 2X_2 - X_4 \le 12,$$
$$X_1 + 2X_2 - X_3 \le 8,$$
$$4X_1 - X_5 \le 16,$$
$$4X_2 - X_6 \le 12,$$
$$X_j \ge 0, \quad (j = 1 \dots 6).$$

Рисунок 9 – Постановка задачі з Лекції_6

Дана задача має 6 змінних, а отже стандартно розглядається в просторі R^6, але так, як у цільовій функції наявні лише 2 змінні – задача може бути розглянута в просторі R^2.

Тоді аналогічно лекції, виконаємо відповідні перетворення для спрощення розрахунків.

Отримаємо наступні обмеження:

$$-2X_1 - 2X_2 \rightarrow \textbf{\textit{min}}$$

$$1.5X_1 + 2X_2 \leq 12,$$

$$X_1 + 2X_2 \leq 8,$$

$$4X_1 \leq 16,$$

$$4X_2 \leq 12,$$

$$X_j \geq 0, \quad (j = 1 \ldots 6)$$

Для збільшення швидкості обчислень перейдемо до цілих чисел:

$$-2X_1 - 2X_2 \rightarrow \textbf{\textit{min}}$$

$$3X_1 + 4X_2 \leq 24,$$

$$X_1 + 2X_2 \leq 8,$$

$$4X_1 \leq 16,$$

$$4X_2 \leq 12,$$

$$X_j \geq 0, \quad (j = 1 \ldots 6)$$

Тепер перейдемо до програмної реалізації за допомогою засобів бібліотеки Google OR-Tools.

Оголосимо модель, ініціалізуємо змінні, введемо обмеження та цільову функцію. Викличемо розв’язувач та відобразимо результати.

Лістинг коду:

```
from ortools.sat.python import cp_model

def cp_model_solver():
    # Оптимізаційна математична модель
    model = cp_model.CpModel()
    var_upper_bound = max(24, 8, 16, 12)
    x1 = model.NewIntVar(0, var_upper_bound, 'x1')
    x2 = model.NewIntVar(0, var_upper_bound, 'x2')
    x3 = model.NewIntVar(0, var_upper_bound, 'x3')
    x4 = model.NewIntVar(0, var_upper_bound, 'x4')
    x5 = model.NewIntVar(0, var_upper_bound, 'x5')
    x6 = model.NewIntVar(0, var_upper_bound, 'x6')

    # Обмеження
    model.Add(3*x1 + 4*x2 <= 24)
    model.Add(1*x1 + 2*x2 <= 8)
```

```
    model.Add(4*x1 <= 16)
    model.Add(4*x2 <= 12)

    # Цільова функція ефективності
    efficiency_function = - 2*x1 - 2*x2
    model.Minimize(efficiency_function)

    # Вирішувач
    solver = cp_model.CpSolver()
    status = solver.Solve(model)

    if status == cp_model.OPTIMAL:
        efficiency_function_opt = solver.ObjectiveValue()
        print('Значення цільової функції в т. оптимуму:', efficiency_function_opt)
        x1_opt = solver.Value(x1)
        x2_opt = solver.Value(x2)
        x3_opt = solver.Value(x3)
        x4_opt = solver.Value(x4)
        x5_opt = solver.Value(x5)
        x6_opt = solver.Value(x6)
        print('x1 =', x1_opt)
        print('x2 =', x2_opt)
        print('x3 =', x3_opt)
        print('x4 =', x4_opt)
        print('x5 =', x5_opt)
        print('x6 =', x6_opt)
    else:
        print('Точки оптимуму не знайдено')

cp_model_solver()
```

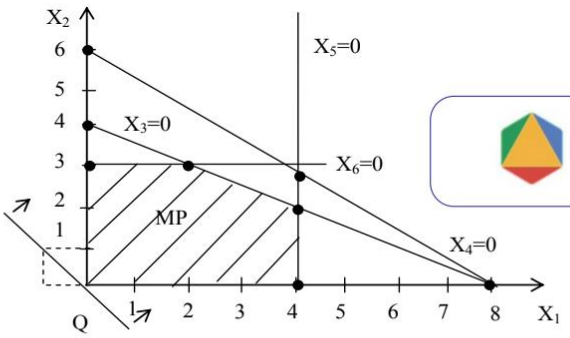
Результат:

У результаті отримуємо оптимальні значення змінних для даної цільової функції та оптимальне значення цільової функції.

```
Значення цільової функції в т. оптимуму: -12.0
x1 = 4
x2 = 2
x3 = 0
x4 = 0
x5 = 0
x6 = 0
```

Рисунок 10 – Точка оптимуму та значення цільової функції в цій точці

Переконаємося, що результати графічного та аналітичного методу збігаються.



Для вибору з множини рішень (МР) необхідно переміщати пряму для *Q* праворуч. Найкраще рішення є в одній з крайніх точок багатокутника. Мінімум цільової функції – розв'язок буде в точці із координатами: *X1=4; X2=2. Решта параметрів – обраховується з нерівності.*

Рисунок 11 – Результати графічного методу розв'язання з Лекції_6

Бачимо, що результати співпадають. Це доводить працездатність розробленого скрипту.

2.3. Опис практичної інтерпретації задачі з Лекції_6

Постановка задачі. Студенту потрібно закрити дисципліну «Вступ до технології Data Science». Дисципліна має 6 лабораторних робіт. Викладач увів штрафні бали за невиконання термінів здачі робіт. Для того, щоб студент з високою кількістю заборгованостей міг закрити його предмет, викладач увів цікаві правила:

1. Якщо студент не здав вчасно 1, 2 та 3 роботи, то за кожен штрафний день 1-ї роботи він отримує «+1 бал», 2-ї «+2 бали», утім за кожен штрафний день 3-ї роботи втратить «-1 бал». Максимально таким чином можна набрати до 8 балів.
2. Якщо студент не здав вчасно 1, 2, 3 та 4 роботи, то за кожен штрафний день 1-ї роботи він отримує «+1.5 бали», 2-ї «+2 бали», утім за кожен штрафний день 4-ї роботи втратить «-1 бал». Максимально таким чином можна набрати до 12 балів.
3. Якщо студент не здав вчасно 1, 2, 3, 4 та 5 роботи, то за кожен штрафний день 1-ї роботи він отримує «+4 бали», утім за кожен штрафний день 5-ї роботи втратить «-1 бал». Максимально таким чином можна набрати до 16 балів.

4. Якщо студент не здав вчасно усі (1-6) роботи, то за кожен штрафний день 2-ї роботи він отримує «+4 бали», утім за кожен штрафний день 6-ї роботи втратить «-1 бал». Максимально таким чином можна набрати до 12 балів.

Усі правила діють одночасно. За штрафний день вважається повна доба.

Задача студента – мінімізувати кількість штрафних балів з 1-ї та 2-ї лабораторних робіт для того, щоб успішно скласти першу атестацію.

Висновок:

Проведено дослідження принципів програмування обмежень (CP-SAT) та багатокритеріального прийняття рішень (Multicriteria decision analysis).

Розроблено програмний скрипт для багатокритеріального оцінювання ефективності позашляховиків різних виробників. Формування показників та критеріїв ефективності було виконано з урахуванням специфіки завдання. Синтез багатокритеріальної оптимізаційної моделі підкреслив важливість узгодження різних критеріїв для прийняття збалансованих рішень.

Розроблено програмний скрипт для розв’язання задачі лінійного програмування з використанням бібліотеки Google OR-Tools. Це дозволило ефективно вирішити задачу оптимізації за визначеними обмеженнями.

Здійснено опис практичної інтерпретації задачі лінійного програмування, яка вирішена за допомогою розробленого скрипту. Приклад реальної ситуації було представлено у контексті даної дисципліни, де необхідно було мінімізувати штрафні бали за 1 та 2 лабораторні роботи для першої атестації.