НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря Сікорського»
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

ЗВІТ

про виконання практикуму № 1

з дисципліни «Інформаційний аналіз даних»

Виконав:

Студент ІІ курсу

Групи КА-13

Приймак Є.О.

Варіант № 25

Перевірила:

Недашківська Н. І.

Київ – 2023 рік

Завдання:

25. Задача аналізу ринкових кошиків. Задано множину товарів $I = \{i_1, i_2, ..., i_n\}$ та множину транзакцій $D = \{T_1, T_2, ..., T_m\}$, де $T = \{i_k | i_k \in I\} \subseteq I$ - транзакція - це множина товарів, які були куплені разом в одному чеку.

    Підтримкою довільного набору $F \subseteq I$ називається число

$$Supp(F) = \frac{|D_F|}{|D|},$$

де D_F - множина транзакцій, які містять набір F:

$$D_F = \{T_j | F \subseteq T_j\},$$

$|D|$ - кількість елементів у множині D.

Знайти множину частих наборів товарів, використовуючи наступний алгоритм:

(а) Побудувати множину одноелементних частих наборів:

$$L_1 = \{i | i \in I, Supp(i) \geq Supp_{min}\},$$

де $Supp_{min}$ - заданий параметр - поріг мінімальної підтримки.

(б) Для всіх $k = 2, ..., n$:
Побудувати множини k-елементних частих наборів

$$L_k = \{F \cup \{i\} | F \in L_{k-1}, i \in L_1 \setminus F, Supp(F \cup \{i\}) \geq Supp_{min}\}.$$

(в) Якщо $L_k = \emptyset$, то вихід із циклу по k.

(г) $\{L_1 \cup L_2 \cup ... \cup L_k\}$ - результуюча множина частих наборів.

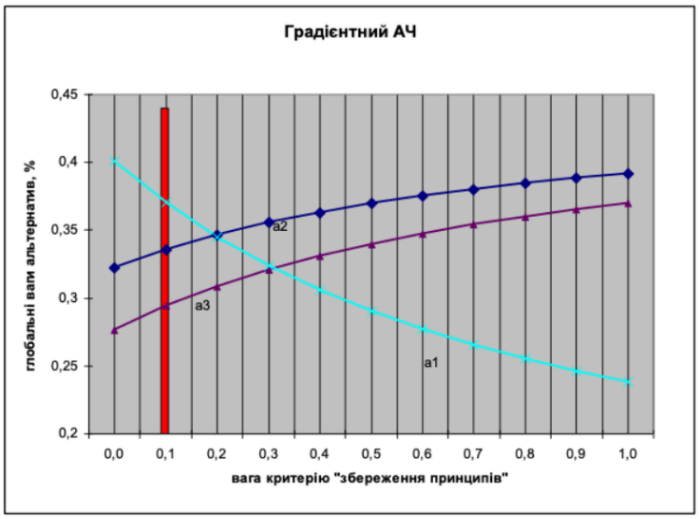


Рис. 3: Приклад градієнтного аналізу чутливості для задачі підтримки прийняття рішень з альтернативами a1, a2, a3

Хід роботи:

1.Спочатку задамо наші товари та відповідні набори транзакцій з ними, та їх кількість (для спрощення кожному виду товару відповідатиме цифра, максимально можливий обсяг транзакції – 5 товарів). Самі товари та їх кількість в чеку обираються випадково:

```
import numpy as np
import matplotlib.pyplot as plt


# Генеруємо тестовий набір даних
#np.random.seed(0)
n = 6  # Кількість товарів
m = 30  # Кількість транзакцій
D = [set(np.random.choice(range(1, n + 1), np.random.randint(1, 6))) for _ in
range(m)]

# Виведемо тестовий набір даних D
print("Тестовий набір транзакцій D:")
for transaction in D:
    print(transaction)
```

Отримуємо:

```
Тестовий набір транзакцій D:
{3, 6}
{1, 3}
{1, 4}
{2, 4, 5, 6}
{2, 4, 6}
{2}
{1, 2, 5, 6}
{1, 2}
{1}
{2, 3, 4, 5}
{4, 6}
{1, 2}
{3, 4, 6}
{5, 6}
{2, 5}
{2, 4}
{2, 3, 4, 5}
{1, 2, 6}
{1, 2, 4, 5}
{6}
{1, 3, 4, 6}
{2}
{1}
{2}
{3, 4, 5, 6}
{3, 6}
{3, 6}
{2, 3, 6}
{1, 5}
```

2. Крок (а): Визначивши поріг мінімальної підтримки suppmin (за замовчанням 10%, можна змінювати), сформуємо множину одноелементних частих наборів

```
# Визначаємо параметр Suppmin (поріг мінімальної підтримки)
Suppmin = 0.1

# Крок (а): Побудувати множину одноелементних частих наборів
L1 = []
for i in range(1, n + 1):
    supp_i = sum(1 for T in D if i in T) / m
    if supp_i >= Suppmin:
        L1.append({i})

# Результуюча множина single_наборів
s_frequent_itemsets = L1.copy()

# Вивести s_frequent_itemsets
print("Одноелементні набори частих товарів (single_frequent_itemsets):")
for itemset in s_frequent_itemsets:
    print(itemset)
```

Отримуємо:

```
Одноелементні набори частих товарів (single_frequent_itemsets):
{1}
{2}
{3}
{4}
{5}
{6}
```

(всі одноелементні набори товарів є частими, оскільки подолали заданий поріг suppmin)

3. Крок (б+в): Формуємо множину до якої входять багатоелементні часті набори

```
# Крок (б): Пошук частих наборів для k = 2, ..., n
for k in range(2, n + 1):
    Lk = []
    for F in L1:
        for i in range(1, n + 1):
            if i not in F:
                F_i = F.union({i})
                supp_F_i = sum(1 for T in D if F_i.issubset(T)) / m
                if supp_F_i >= Suppmin:
                    Lk.append(F_i)
    if not Lk:
        break  #Крок (в), вихід з циклу
    s_frequent_itemsets.extend(Lk)
    L1 = Lk

 #(позбавляємось від дублікатів)
resulting_frequent_itemsets = []
for itemset in s_frequent_itemsets:
    if itemset not in resulting_frequent_itemsets:
        resulting_frequent_itemsets.append(itemset)
```

4. Крок (г): Отримуємо остаточну множину частих наборів

```
# Крок (г), виводимо resulting_frequent_itemsets
print("\n Часті набори товарів після завершення алгоритму
(resulting_frequent_itemsets):")
for itemset in resulting_frequent_itemsets:
    print(itemset)
```

Маємо:

```
 Часті набори товарів після завершення алгоритму (resulting_frequent_itemsets):
{1}
{2}
{3}
{4}
{5}
{6}
{1, 2}
{1, 4}
{1, 5}
{1, 6}
{2, 3}
{2, 4}
{2, 5}
{2, 6}
{3, 4}
{3, 5}
{3, 6}
{4, 5}
{4, 6}
{5, 6}
{2, 4, 5}
{3, 4, 5}
{3, 4, 6}
```

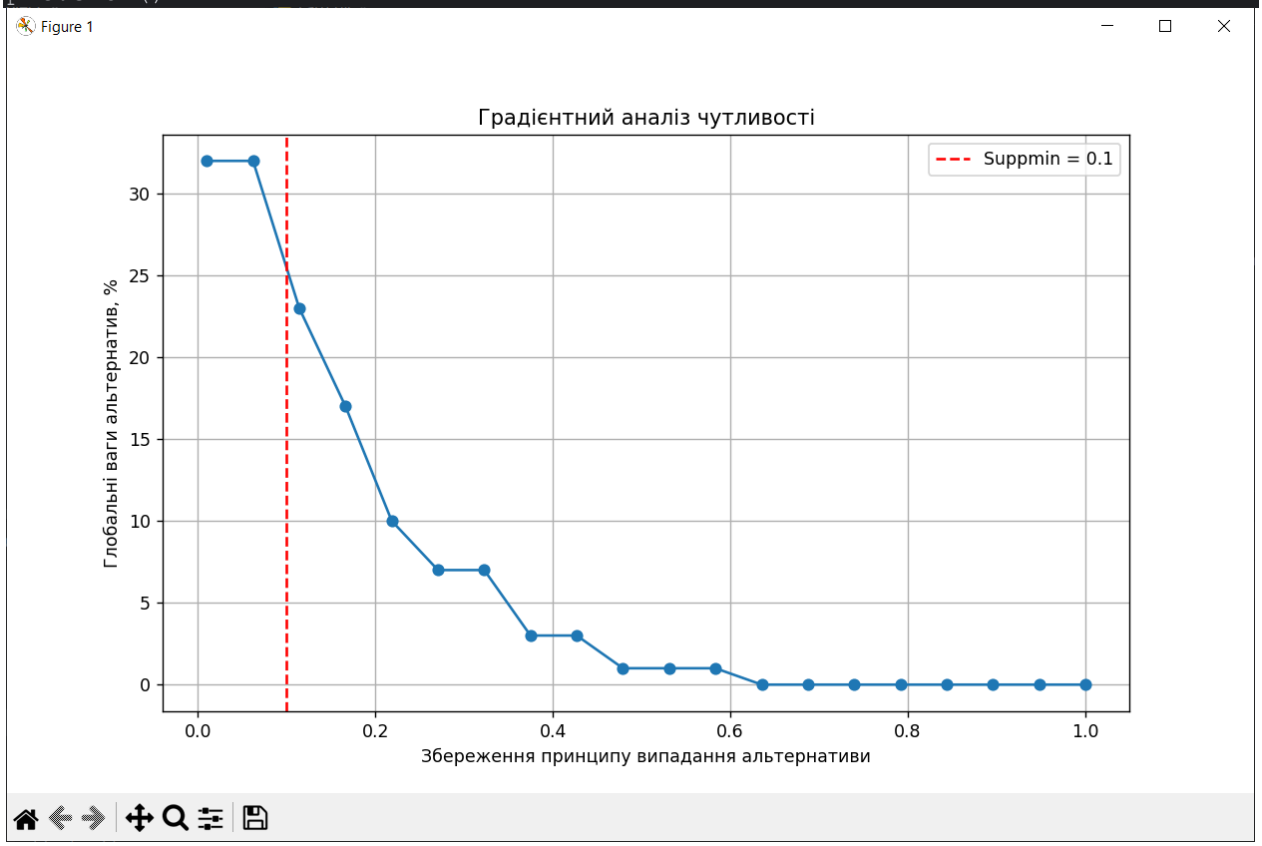
5. Зробимо градієнтний аналіз чутливості для отриманого рішення та візуалізуємо результати:

```
support_values = []  # глобальні ваги альтернатив

for supp_threshold in np.linspace(0.01, 1.0, 20):  # діапазон і крок
    # Рахуємо кількість альтернатив, які відповідають заданому порогу
підтримки
    num_satisfying_alternatives = sum(1 for F in resulting_frequent_itemsets
if
                                      sum(1 for T in D if F.issubset(T)) / m
>= supp_threshold)
    support_values.append(num_satisfying_alternatives)

# Побудова графіку
plt.figure(figsize=(10, 6))
plt.plot(np.linspace(0.01, 1.0, 20), support_values, marker='o', linestyle='-
')
plt.axvline(x=Suppmin, color='red', linestyle='--', label=f'Suppmin =
{Suppmin}')
plt.xlabel('Збереження принципу випадання альтернативи')
plt.ylabel('Глобальні ваги альтернатив, %')
plt.title('Градієнтний аналіз чутливості')
plt.legend()
```

```
plt.grid(True)
plt.show()
```



Очікувано, на графіку малий поріг випадання альтернативи проходять усі значення, а з його збільшенням допустимих частих елементів стає все менше, і з досяганням рівня підтримки 0.6 і вище вони вже не знаходяться.

Аналогічно, можна провести декілька експериментів зі зміненим порогом мінімальної підтримки, або збільшеною вибіркою товарів

Наприклад, для 12 випадкових товарів (suppmin=10%) маємо:

```
Тестовий набір транзакцій D:
{1, 3, 12, 5}
{1, 4, 5}
{8}
{9, 4, 5}
{8, 5}
{9}
{12}
{9, 2, 3, 5}
{3, 12, 6}
{3, 5, 8, 10, 11}
{8, 1}
{4}
{5}
{9}
{1, 10, 11, 5}
{1, 10, 11, 4}
{3, 4, 7}
{9, 12}
{1}
{1, 9, 10, 11, 12}
{6, 7}
{8, 9, 5, 6}
{12, 4, 5, 7}
{2, 3}
{8, 7}
{3, 12, 7}
{1, 2, 10, 9}
{12}
{8, 9, 12}
{4, 5}
```

```
Одноелементні набори частих товарів (single_frequent_itemsets):
{1}
{2}
{3}
{4}
{5}
{6}
{7}
{8}
{9}
{10}
{11}
{12}
 Часті набори товарів після завершення алгоритму (resulting_frequent_itemsets):
{1}
{2}
{3}
{4}
{5}
{6}
{7}
{8}
{9}
{10}
{11}
{12}
{1, 5}
{1, 10}
{1, 11}
{3, 5}
{3, 12}
{4, 5}
{8, 5}
{9, 5}
{9, 12}
{10, 11}
{1, 10, 11}
```