ДОДАТОК А

Лістинг коду алгоритм

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

# Завантаження даних
data = pd.read_json('/Users/pogremuhalike/Downloads/All_Beauty_5.json', lines=True)

# Перетворення часу на UNIX timestamp
data['reviewTime'] = pd.to_datetime(data['reviewTime'])
data['unixReviewTime'] = data['reviewTime'].astype(np.int64) // 10 ** 9

# Обчислення медіанних рейтингів для кожного товару
medians = data.groupby('asin')['overall'].median()

# Додавання медіан до даних
data = data.join(medians, on='asin', rsuffix='_median')

# Обчислення відхилень від медіани
data['delta_R'] = data['overall'] - data['overall_median']

# Параметр затухання
lambda_decay = 0.01

# Поточний час (максимальний час у наборі даних)
current_time = data['unixReviewTime'].max()
```

```
# Обчислення ваг
data['weight'] = np.exp(-lambda_decay * (current_time - data['unixReviewTime']) / (60 * 60 * 24 * 365))

# Обчислення зважених рейтингів з урахуванням відхилення від медіани
data['weighted_delta_R'] = data['weight'] * data['delta_R']

# Функція для обчислення зважених рейтингів
def weighted_rating(group):
total_weight = group['weight'].sum()

if total_weight == 0:
return np.nan

return group['weighted_delta_R'].sum() / total_weight

# Зведення результатів для кожного товару
grouped = data.groupby('asin').agg( total_weight=('weight', 'sum'), weighted_delta_sum=('weighted_delta_R', 'sum') ).reset_index()

# Обчислення зважених рейтингів
grouped['weighted_rating'] = grouped.apply( lambda x: x['weighted_delta_sum'] / x['total_weight'] if x['total_weight'] != 0 else np.nan, axis=1
)

# Порівняння явного та неявного зворотного зв'язку
grouped_explicit = grouped[['asin', 'weighted_rating']].rename(columns={'weighted_rating': 'explicit_rating'})
```

```
grouped_implicit = grouped[['asin', 'total_weight']].rename(columns={'total_weight': 'implicit_feedback'})
# Злиття результатів для подальшого аналізу
comparison = pd.merge(grouped_explicit, grouped_implicit, on='asin')

# Аналіз розбіжностей
comparison['discrepancy'] = np.abs(comparison['explicit_rating'] - comparison['implicit_feedback'])

# Визначення інтервалів для аналізу (наприклад, щомісячно)
data['month'] = data['reviewTime'].dt.to_period('M')
interval_data = data.groupby('month').agg(
sales=('overall', 'count'),
avg_rating=('overall', 'mean')
).reset_index()

# Обчислення розбіжностей для кожного інтервалу
interval_discrepancy = data.groupby('month').apply(
lambda x: pd.Series({'discrepancy':
np.abs(weighted_rating(x) - x['weight'].sum())})
).reset_index()

# Спрощення графіка
plt.figure(figsize=(14, 7))
plt.plot(interval_data['month'].astype(str),
interval_data['sales'], 'b--o', label='Sales', markersize=5)
plt.plot(interval_data['month'].astype(str),
interval_data['avg_rating'], 'r--o', label='Ratings', markersize=5)
plt.plot(interval_discrepancy['month'].astype(str),
```

```
interval_discrepancy['discrepancy'], 'k--o', label='Divergence', markersize=5)

# Визначення можливих атак
threshold = interval_discrepancy['discrepancy'].quantile(0.95)
possible_attack = interval_discrepancy['discrepancy'].apply(lambda x: x if x > threshold else 0)
plt.plot(interval_discrepancy['month'].astype(str),
possible_attack, 'g-o', label='Possible attack', markersize=5)

# Додавання підписів та легенди
plt.xlabel('Interval index')
plt.ylabel('Signs of a shilling attack')
plt.legend() plt.grid(True)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

# Показ графіка
plt.show()
```

Лістинг 1 – Лістинг коду алгоритму виявлення шиллінг-атак

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	70 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Гончаренко Д.О.			Дослідження еволюційних методів для персоналізованого календарного планування на основі аналізу повідомлень про зустрічі в месенджері	Шифр групи	Код напр./спец.
Перевірив		Чалий С.Ф.				СШІм-22-3	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					