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

Кафедра інформаційних систем та технологій

Лабораторна робота №2

з дисципліни «Вступ до технології Data Science»

на тему

«СТАТИСТИЧНЕ НАВЧАННЯ З ПОЛІНОМІАЛЬНОЮ РЕГРЕСІЄЮ»

Виконала:
студентка групи ІС-12
Павлова Софія

Перевірив:
Баран Д. Р.

Київ – 2023

1. Постановка задачі

Мета роботи:

Виявити дослідити та узагальнити особливості реалізації процесів статистичного навчання із застосуванням методів обробки Big Data масивів та калмановської рекурентної фільтрації з використанням можливостей мови програмування Python.

Завдання IV рівня:

1. Отримання вхідних даних із властивостями, заданими в Лр_1;
2. Модель вхідних даних із аномальними вимірами;
3. Очищення вхідних даних від аномальних вимірів. Спосіб виявлення аномалій та очищення обрати самостійно;
4. Визначення показників якості та оптимізація моделі Показник якості та спосіб оптимізації обрати самостійно.
5. Залежно від результатів п.4 реалізувати рекурентне згладжування alfa-beta, або alfa-beta-gamma фільтром сформованих в п.1, 2 вхідних даних. Прийняти заходи подолання явища «розбіжності» фільта.
6. Провести аналіз отриманих результатів та верифікацію розробленого скрипта.
7. Реалізувати **R&D** процеси для етапів статистичного навчання.

   3.1. Здійснити «навчання» параметрів відомих алгоритмів «бачити» властивості статистичної вибірки.

2. Виконання

2.1. Отримання вхідних даних із властивостями, заданими в Лр_1

Відповідно до лабораторної роботи № 1, отримаємо на вхід нашої програми дані по втратах російського війська з моменту повномасштабного вторгнення станом на 1-21 вересня 2023 року.

Дані подаються на вхід у вигляді .xlsx таблиці. Категоріями даних виступатимуть: *Особовий склад*, *Бойові броньовані машини*, *Танки*, *Артилерія*, *Літаки*, *Гелікоптери*, *Кораблі*.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	date	personnel	armed_vehicles	tanks	artilleries	aircrafts	helicopters	ships						
2	21.09.2023	274470	8883	4638	6137	315	316	20						
3	20.09.2023	273980	8868	4635	6096	315	316	20						
4	19.09.2023	273460	8851	4628	6062	315	316	20						
5	18.09.2023	272940	8834	4623	6027	315	316	20						
6	17.09.2023	272320	8828	4620	6003	315	316	20						
7	16.09.2023	271790	8824	4616	5988	315	316	20						
8	15.09.2023	271440	8814	4612	5972	315	316	20						
9	14.09.2023	270970	8810	4599	5944	315	316	20						
10	13.09.2023	270350	8792	4584	5902	315	316	19						
11	12.09.2023	269760	8778	4568	5872	315	316	19						
12	11.09.2023	269210	8767	4560	5839	315	316	19						
13	10.09.2023	268630	8755	4554	5811	315	316	19						
14	09.09.2023	268140	8739	4544	5789	315	316	19						
15	08.09.2023	267540	8726	4529	5753	315	316	19						
16	07.09.2023	266900	8703	4506	5722	315	316	19						
17	06.09.2023	266290	8682	4497	5685	315	316	19						
18	05.09.2023	265680	8670	4489	5649	315	316	19						
19	04.09.2023	265120	8663	4480	5611	315	316	19						
20	03.09.2023	264660	8649	4476	5582	315	316	18						
21	02.09.2023	264060	8637	4471	5560	315	316	18						
22	01.09.2023	263490	8613	4459	5530	315	316	18						
23														

Рисунок 1 – Таблиця з реальними даними

З огляду на об’єм та розмір вхідних даних, будемо працювати з даними у вигляді структури *DataFrame*. Так, як отримані дані представлені в таблиці в порядку додавання нової інформації на сайт, відсортуємо їх за зростанням дати для зручності подальшої роботи з ними.

Згідно з поставленим завданням, наша програма прийматиме на вхід реальні дані і створюватиме за їх трендом модель, з якою буде продовжувати працювати.

Також залишимо опцію працювати з реальними даними. Це буде доцільно для демонстрації роботи фільтрації.

Реалізуємо функцію вибору категорії вхідних даних.

Лістинг коду:

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

'''Блок отримання вибірки'''

# Функція сортування даних
def sort(df):
    df_sorted = df.sort_values(by='date', ascending=True)
    # Скидаємо індекси та змінюємо їх на нові
    df_sorted = df_sorted.reset_index(drop=True)
    return df_sorted

# Функція парсингу реальних даних
def file_parsing (url, file, column):
    df = pd.read_excel(file)
    # Сортування
    df_sorted = sort(df)
    print('-----------------------------------------------------')
    print('Вибірка -', column)
    print('-----------------------------------------------------')
    for name, values in df_sorted[[column]].items():
        print(values)
    df_1 = np.zeros((len(values)))
    for i in range(len(values)):
        df_1[i] = values[i]
    print('Джерело даних: ', url)
    return df_1

'''Основний блок'''

# URL веб-сторінки, з якої отримано реальні дані
url = "https://www.minusrus.com/"

columns = ['personnel', 'armed_vehicles', 'tanks', 'artilleries', 'aircrafts',
'helicopters', 'ships']
names = ['Особовий склад', 'Бойові броньовані машини', 'Танки', 'Артилерія', 'Літаки',
'Гелікоптери', 'Кораблі']
vtratu = ['дохлої', 'знищених броньованих машин', 'знищених танків', 'знищеної
артилерії', 'збитих літаків', 'збитих гелікоптерів', 'потоплених кораблів']

# Вибір режиму отримання даних
print('Оберіть режим отримання вхідних даних:')
print('1 - Модель')
print('2 - Реальні дані')
data_var = int(input('mode:'))
# Якщо режим існує
if data_var in range(1, 3):
    # Вибір джерела вхідних даних
    print('Оберіть джерело вхідних даних:')
```

```
    for i in range(len(names)):
        print(i + 1, '-', names[i])
    data_mode = int(input('mode:'))
    # Якщо джерело даних існує
    if data_mode in range(1, len(names) + 1):
        # Зчитування та сортування вибірки
        df_real_sorted = file_parsing(url, 'rosnia.xlsx', columns[data_mode - 1])
```

Оберемо за джерело вхідних даних – *модель*, категорію – *кораблі*. Далі будемо працювати саме з цими даними.

Результат:

У результаті виконання програми отримуємо вибірку.

```
Оберіть режим отримання вхідних даних:
1 - Модель
2 - Реальні дані
mode:1
Оберіть джерело вхідних даних:
1 - Особовий склад
2 - Бойові броньовані машини
3 - Танки
4 - Артилерія
5 - Літаки
6 - Гелікоптери
7 - Кораблі
mode:7
--------------------------------------------------
Вибірка - ships
--------------------------------------------------
0     18
1     18
2     18
3     19
4     19
5     19
6     19
7     19
8     19
9     19
10    19
11    19
12    19
13    20
14    20
15    20
16    20
17    20
```

```
18    20
19    20
20    20
Name: ships, dtype: int64
Джерело даних:  https://www.minusrus.com/
```

Рисунок 2 – Вибірка реальних даних

Визначимо тренд за допомогою алгоритму МНК згладжування та характеристики вхідних даних.

Лістинг коду:

```
[...]

'''Блок лінії тренду'''

# Функція МНК згладжування для визначення лінії тренду
def mnk (df, mode):
    iter = len(df)
    var = np.zeros((iter, 1))
    val = np.ones((iter, 3))
    for i in range(iter):  # Формування структури вхідних матриць МНК
        var[i, 0] = float(df[i])  # Формування матриці вхідних даних
        val[i, 1] = float(i)
        val[i, 2] = float(i * i)
    valT = val.T
    valT_T = valT.dot(val)
    valT_TI = np.linalg.inv(valT_T)
    valT_TI_valT = valT_TI.dot(valT)
    C = valT_TI_valT.dot(var)
    df_reg = val.dot(C)
    if mode is True:
        print('-----------------------------------------------------')
        print('Регресійна модель')
        print('-----------------------------------------------------')
        print('y(t) = ', C[0, 0], ' + ', C[1, 0], ' * t', ' + ', C[2, 0], ' * t^2')
    return df_reg, C

'''Блок визначення характеристик'''

# Для вибірки
def stat_characteristics_in(df, mode):
    # Статистичні характеристики вибірки з урахуванням тренду
    df_zglad, coef = mnk(df, False)
    iter = len(df)
    df_1 = np.zeros((iter))
    for i in range(iter):
        df_1[i] = df[i] - df_zglad[i, 0]
    mat_spod = np.mean(df_1)
    duspers = np.var(df_1)
    ser_kvad_vid = np.sqrt(duspers)
    if mode is True:
        print('-----------------------------------------------------')
        print('Статистичні характеристики вибірки')
        print('-----------------------------------------------------')
```

```
        print('Кількість елементів вибірки =', iter)
        print('Математичне сподівання =', mat_spod)
        print('Дисперсія =', duspers)
        print('Середнє квадратичне відхилення =', ser_kvad_vid)
    return mat_spod, ser_kvad_vid

'''Основний блок'''

[...]
# Якщо режим існує
if data_var in range(1, 3):
    [...]
    # Якщо джерело даних існує
    if data_mode in range(1, len(names) + 1):
        [...]
        # Визначення тренду та коефіцієнтів регресії
        df_zglad, coef = mnk(df_real_sorted, False)
        # Визначення характеристик вибірки
        ser, skv = stat_characteristics_in(df_real_sorted, True)
```

Результат:

У результаті виконання програми знаходимо коефіцієнти регресії, необхідні для побудови моделі та отримуємо статистичні характеристики вибірки.

```
------------------------------------------------------
Статистичні характеристики вибірки
------------------------------------------------------
Кількість елементів вибірки = 21
Математичне сподівання = 4.9907168345054655e-14
Дисперсія = 0.07619953327700953
Середнє квадратичне відхилення = 0.2760426294560489
```

Рисунок 3 – Статистичні характеристики реальних даних

2.2. Модель вхідних даних із аномальними вимірами

Будуємо з отриманих коефіцієнтів регресійну модель. Якщо характеристики моделі будуть максимально наближені до характеристик вхідної вибірки – модель побудовано правильно. Так, як реальні дані завжди мають шуми, додамо до моделі стохастичний шум за нормальним законом розподілу.

Лістинг коду:

```
[...]

'''Блок визначення характеристик'''

[...]

# Для моделі
def stat_characteristics_out (df, text):
    # Статистичні характеристики вибірки з урахуванням тренду
    df_zglad, coef = mnk(df, False)
    iter = len(df_zglad)
    df_1 = np.zeros((iter))
    for i in range(iter):
        df_1[i] = df[i] - df_zglad[i, 0]
    mat_spod = np.mean(df_1)
    duspers = np.var(df_1)
    ser_kvad_vid = np.sqrt(duspers)
    # Глобальне лінійне відхилення оцінки - динамічна похибка моделі
    delta = 0
    for i in range(iter):
        delta = delta + abs(df[i] - df_zglad[i, 0])
    delta_average = delta / (iter + 1)
    print('-----------------------------------------------------')
    print('Статистичні характеристики', text)
    print('-----------------------------------------------------')
    print('Кількість елементів вибірки =', iter)
    print('Матиматичне сподівання =', mat_spod)
    print('Дисперсія =', duspers)
    print('Середнє квадратичне відхилення =', ser_kvad_vid)
    print('Динамічна похибка моделі =', delta_average)

'''Блок моделі'''

def model(a, b, c, mode):
    # Генерування часової послідовності (змінної x)
    x = np.linspace(0, 21, 21)  # Від 0 до 21 з 21 рівномірно розподіленим значенням
    # Розрахунок значень моделі
    y = a + b*x + c*x*x
    # plot(y, x, 'Синтезована за лінійним трендом модель', names[mode-1], names[mode-1],
'(i+1)-е вересня 2023', 'Штуки')
    return y

# Функція додавання нормального шуму
def norm_shum(model, n, m, skv):
    # генерація випадкового шумуза нормальним законом
    shum = np.random.normal(m, skv, n)
    df_shum = np.zeros(n)
    for i in range(n):
        df_shum[i] = model[i] + shum[i]
    return df_shum

'''Допоміжний блок'''

# Функція для виведення графіку
def plot(df, df_1, title, label, label_1, xlabel, ylabel):
    plt.figure(figsize=(8, 6))
    if label != label_1:
        plt.plot(df_1, label=label_1)
    plt.plot(df, label=label)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
```

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

'''Основний блок'''

[...]
# Якщо режим існує
if data_var in range(1, 3):
    [...]
    # Якщо джерело даних існує
    if data_mode in range(1, len(names) + 1):
        [...]
        # Якщо модель
        if (data_var == 1):
            specified_text = 'Модель'
            specified_text_1 = 'моделі'
            # Синтезація моделі
            df = model(coef[0, 0], coef[1, 0], coef[2, 0], data_mode)
            # Стохастичний шум
            df_real_sorted = norm_shum(df, len(df), ser, skv)
            # Характеристики
            stat_characteristics_out(df_real_sorted, 'моделі + шум')
            ser, skv = stat_characteristics_in(df_real_sorted, False)
            plot(df_zglad, df_real_sorted, 'Модель + нормальний шум', 'Лінія тренду',
names[data_mode - 1], '(i+1)-е вересня 2023', 'Штуки')
```

Результат:

У результаті отримуємо характеристики моделі. Бачимо, що вони максимально наближені до характеристик реальних даних, а динамічна похибка моделі доволі низька.

```
--------------------------------------------------
Статистичні характеристики вибірки
--------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 4.9907168345054655e-14
Дисперсія = 0.07619953327700953
Середнє квадратичне відхилення = 0.2760426294560489
--------------------------------------------------
Статистичні характеристики моделі + шум
--------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 6.259543148362788e-14
Дисперсія = 0.06604976575968806
Середнє квадратичне відхилення = 0.25700148980052245
Динамічна похибка моделі = 0.2047111772905452
```

Рисунок 4 – Статистичні характеристики моделі

Додатково можемо переконатися, що графік моделі з нормальним шумом відповідає лінії тренду реальних даних.

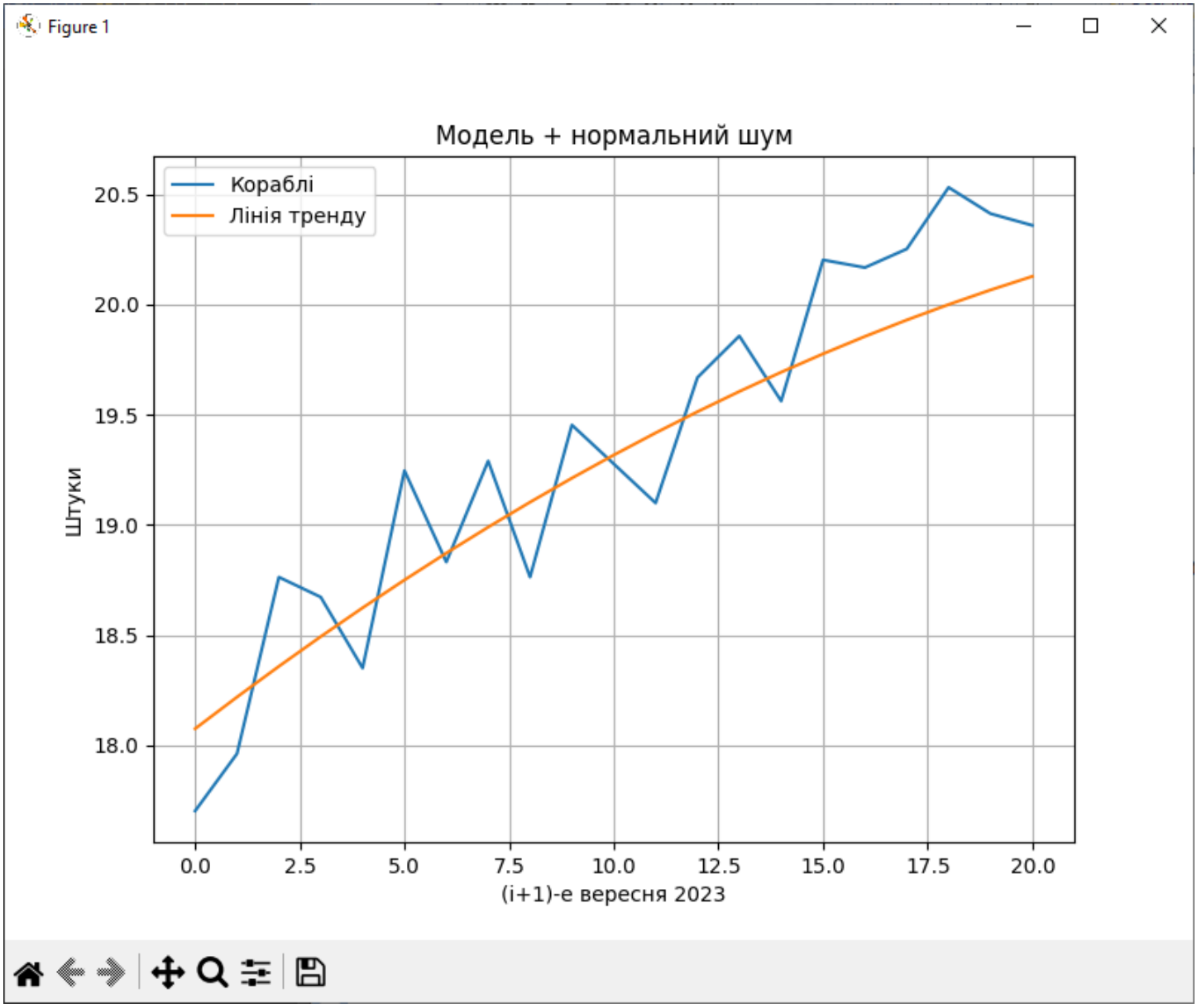

Рисунок 5 – Графік моделі з нормальним шумом

Отже, можемо зробити висновок, що модель створено успішно. Можемо переходити до наступного завдання.

Нам необхідно, щоб модель містила аномальні відхилення. За визначенням аномальними відхиленнями вважаються значення, що суттєво відрізняються від інших точок у вибірці. Тому використаємо функцію генерації випадкових чисел, щоб імітувати

аномальні значення. Таким чином, створимо модель даних з додаванням аномальних випадкових похибок за нормальним законом до вхідних даних.

Лістинг коду:

```
[...]

'''Блок аномалій'''

# Функція додавання аномалій
def av(model, num_anomalies, anomaly_mean, anomaly_std):
    # Генерація випадкових аномалій за нормальним законом
    av = np.random.normal(anomaly_mean, anomaly_std, num_anomalies)
    model_av = np.zeros(num_anomalies)
    for i in range(num_anomalies):
        model_av[i] = model[i] + av[i]
    return model_av

[...]

'''Основний блок'''

[...]
# Якщо режим існує
if data_var in range(1, 3):
    [...]
    # Якщо джерело даних існує
    if data_mode in range(1, len(names) + 1):
        [...]
        # Якщо модель
        if (data_var == 1):
            [...]
            # Аномальні відхилення
            df_av = av(df_real_sorted, len(df_real_sorted), ser, skv)
            # Характеристики
            stat_characteristics_out(df_av, 'моделі + шум + аномалії')
            plot(df_zglad, df_av, 'Модель + нормальний шум + аномалії', 'Лінія тренду',
names[data_mode - 1], '(i+1)-е вересня 2023', 'Штуки')
```

Результат:

У результаті отримуємо модель з аномальними викидами, її характеристики та графік.

```
--------------------------------------------------
Статистичні характеристики моделі + шум
--------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 6.259543148362788e-14
Дисперсія = 0.06604976575968806
Середнє квадратичне відхилення = 0.25700148980052245
Динамічна похибка моделі = 0.2047111772905452
--------------------------------------------------
Статистичні характеристики моделі + шум + аномалії
--------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 6.107283990699909e-14
Дисперсія = 0.113828504414611
Середнє квадратичне відхилення = 0.3373848016947577
Динамічна похибка моделі = 0.26601765590754006
```

Рисунок 6 – Статистичні характеристики моделі з аномальними викидами

По значенню середньоквадратичного відхилення бачимо, що до вибірки додались аномалії. Додатково можемо переконатися в цьому з графіка.

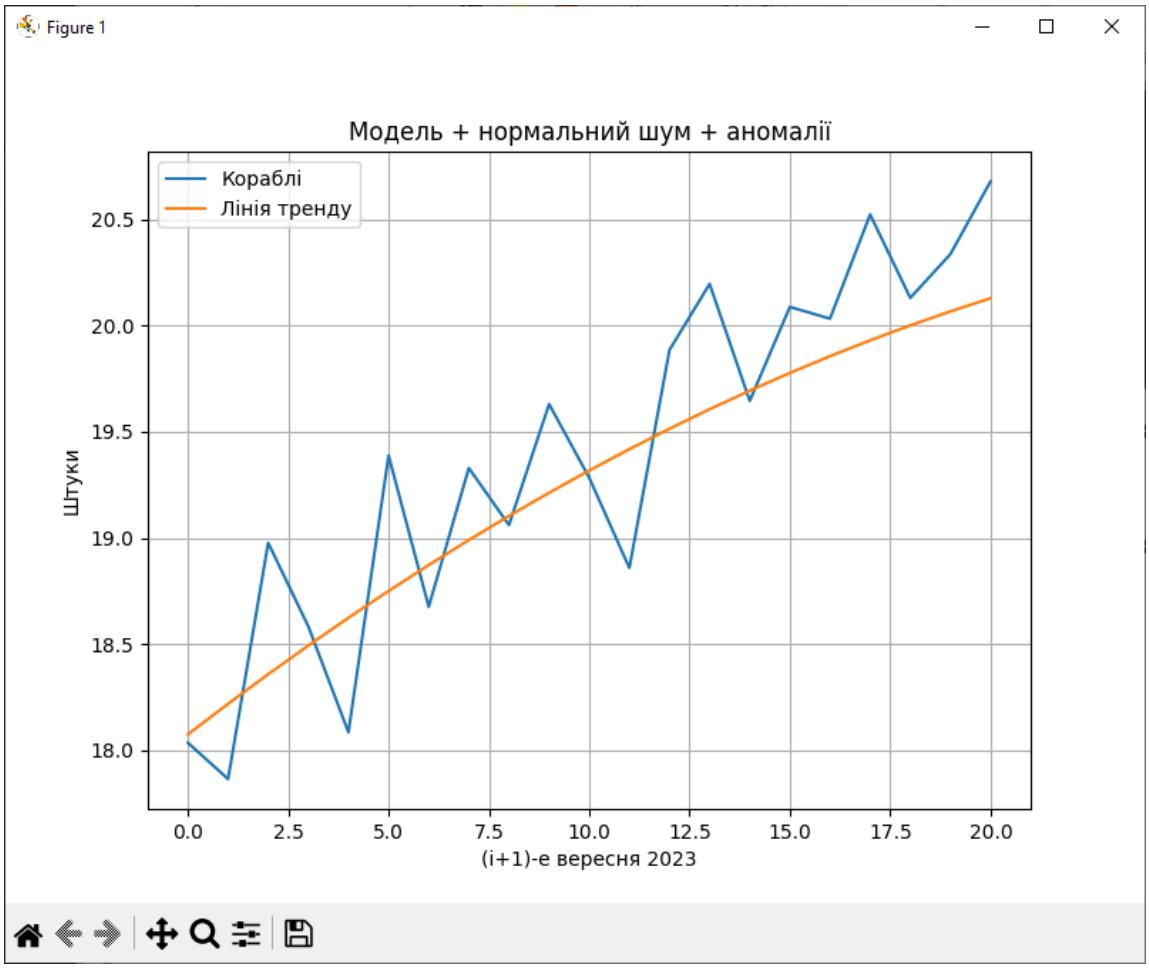

Рисунок 7 – Графік моделі з нормальним шумом та аномальними викидами

Аналогічно реалізуємо функціонал для вибору користувачем режиму зчитування реальних даних.

Лістинг коду:

```
[...]

'''Основний блок'''

[...]
# Якщо режим існує
if data_var in range(1, 3):
    [...]
    # Якщо джерело даних існує
    if data_mode in range(1, len(names) + 1):
        [...]
        # Якщо модель
        if (data_var == 1):
            [...]
        # Якщо реальні дані
        else:
            specified_text = 'Вибірка'
            specified_text_1 = 'вибірки'
            plot(df_zglad, df_real_sorted, 'Кількість ' + str(vtratu[data_mode - 1]) + '
росні станом на вересень 2023', 'Лінія тренду', names[data_mode - 1], '(i+1)-е вересня
2023', 'Штуки')
            # Аномальні відхилення
            df_av = av(df_real_sorted, len(df_real_sorted), ser, skv)
            # Характеристики
            stat_characteristics_out(df_av, 'вибірки + аномалії')
            plot(df_zglad, df_av, 'Вибірка + аномалії', 'Лінія тренду', names[data_mode -
1], '(i+1)-е вересня 2023', 'Штуки')
```

2.3. Очищення вхідних даних від аномальних вимірів

Для очищення даних від аномальних вимірів оберемо 3 алгоритми: медіан, МНК та ковзного вікна. Реалізуємо програму таким чином, щоб користувач міг обирати алгоритм очищення від аномалій, оскільки вони є суттєво різними за особливостями використання.

Розглянемо спершу *алгоритм медіан*.

Алгоритм сортує вибірку в порядку зростання так, що середина вибірки – медіана. По вибірці проходить фільтр із заданим кроком (*вікно*), аномаліями вважаються значення, що відхиляються від медіани на *порогове значення*. Усунення аномалій

відбувається за допомогою визначення середнього значення між найближчими двома точками.

Таким чином, напишемо функцію, яка буде знаходити та усувати аномалії за методом медіан. Функція прийматиме параметри розміру вікна та порогові значення.

Згідно з завданням *4 рівня складності*, необхідно прив'язати значення параметрів алгоритму очищення до характеристик вибірки.

Шляхом повторення експерименту зміни параметрів для алгоритму, було встановлено *залежність між об'ємом вибірки та значенням вікна*: *window_size = int(len(df_real_sorted) / 4)* та *залежність між пороговим значенням та середньоквадратичним відхиленням*: *threshold = skv * 2.*

Наперед визначимо показники якості моделі, щоб розуміти, наскільки вдалим було очищення даних від аномалій. За такі метрики візьмемо *середньо абсолютну помилку* (MAE) та *коефіцієнт детермінації* (R^2).

Лістинг коду:

```
'''Блок аномалій'''

[...]

# Функція виявлення та усунення аномалій за алгоритмом medium
def av_medium(df, window_size, threshold):
    n = len(df)
    # Створимо копію вибірки для очищення
    df_cleaned = np.copy(df)
    for i in range(n):
        if i >= window_size:
            # Отримаємо поточне ковзне вікно
            window = df[i - window_size:i]
            # Обчислимо медіану ковзного вікна
            median = np.median(window)
            # Якщо різниця поточного значення та медіани перевищує поріг, вважаємо це
аномалією
            if abs(df[i] - median) > threshold:
                # Замінюємо аномальне значення на медіану
                df_cleaned[i] = median
    return df_cleaned

'''Блок оцінювання якості моделі'''

# Функція для обчислення коефіцієнта детермінації R^2
def mae_r2_score(actual, predicted):
    # Розрахунок MAE
```

```
    mae = np.mean(np.abs(actual - predicted))
    # Розрахунок R^2
    mean_actual = np.mean(actual)
    sst = np.sum((actual - mean_actual) ** 2)
    ssr = np.sum((actual - predicted) ** 2)
    r2 = 1 - (ssr / sst)
    return mae, r2

'''Допоміжний блок'''

# Функція для виведення часу, витраченого на очищення аномалій
def print_time(start_time):
    # Фіксуємо час, на очищення від аномалій
    total_time = (time.time() - start_time)
    print('Час виконання операцій =', total_time, 'с')

# Функція для виведення якості очищення даних від аномалій
def quality_av_detection(df_cleaned, specified_text, specified_text_1, alg, start_time,
av_mode, threshold, window_size, mae, r2, df_zglad, names, data_mode, df_real_sorted,
text_1, text_2, mode):
    stat_characteristics_out(df_cleaned, str(specified_text_1) + '\nочищеної від аномалій
' + str(text_2) + ' за алгоритмом ' + str(alg))
    print('-----------------------------------------------------')
    print('Якість', text_1)
    print('-----------------------------------------------------')
    if mode is False:
        print_time(start_time)
        if av_mode != 3:
            print('Поріг =', threshold)
        print('Розмір вікна =', window_size)
    print('Середньо-абсолютна помилка =', mae)
    print('Коефіцієнт детермінації =', r2)
    if mode is False:
        plot(df_zglad, df_cleaned, str(specified_text) + ' очищена від аномалій ' +
str(text_2) + ' за алгоритмом ' + str(alg), 'Лінія тренду', names[data_mode - 1], '(i+1)-
е вересня 2023', 'Штуки')
        plot(df_real_sorted, df_cleaned, 'Порівняння вхідних даних з очищеними від
аномалій ' + str(text_2) + ' за алгоритмом ' + str(alg), 'Вхідні дані', 'Алгоритм ' +
str(alg), '(i+1)-е вересня 2023', 'Штуки')
    else:
        plot(df_zglad, df_cleaned, str(specified_text) + ' після фільтрації', 'Лінія
тренду', names[data_mode - 1], '(i+1)-е вересня 2023', 'Штуки')
        plot(df_real_sorted, df_cleaned, 'Порівняння вхідних даних з результатами
фільтрації', 'Вхідні дані', 'Фільтрація', '(i+1)-е вересня 2023', 'Штуки')

[...]

'''Основний блок'''

[...]
# Якщо режим існує
if data_var in range(1, 3):
    [...]
    # Якщо джерело даних існує
    if data_mode in range(1, len(names) + 1):
        [...]
        # Вибір алгоритму детекції та усунення аномалій
        print('-----------------------------------------------------')
        print('Оберіть алгоритм виявлення та усунення аномалій:')
        print('1 - Метод medium')
        print('2 - Метод MNK')
        print('3 - Метод sliding window')
        av_mode = int(input('mode:'))
```

```
        # Якщо алгоритм існує
        if av_mode in range(1, 4):
            # Алгоритм медіан
            if av_mode == 1:
                alg = 'medium'
                # Використання характеристик даних
                window_size = int(len(df_real_sorted) / 4)
                threshold = skv * 2
                # Фіксуємо час початку обчислень
                start_time = time.time()
                df_cleaned = av_medium(df_av, window_size, threshold)
                mae, r2 = mae_r2_score(df_real_sorted, df_cleaned)
                quality_av_detection(df_cleaned, specified_text, specified_text_1, alg,
start_time, av_mode, threshold, window_size, mae, r2, df_zglad, names, data_mode,
df_real_sorted, 'моделі', '', False)
```

Результат:

При виборі режиму роботи програми = 1, отримуємо очищену від аномалій методом медіан вибірку, її характеристики та оцінку якості роботи алгоритму.

```
--------------------------------------------------
Оберіть алгоритм виявлення та усунення аномалій:
1 - Метод medium
2 - Метод МНК
3 - Метод sliding window
mode:1
--------------------------------------------------
Статистичні характеристики моделі
очищеної від аномалій  за алгоритмом medium
--------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 5.938107148852266e-14
Дисперсія = 0.08430304102528766
Середнє квадратичне відхилення = 0.2903498596956569
Динамічна похибка моделі = 0.21299589664478302
--------------------------------------------------
Якість моделі
--------------------------------------------------
Час виконання операцій = 0.0 с
Поріг = 0.5140029796010449
Розмір вікна = 5
Середньо-абсолютна помилка = 0.29228059709914167
Коефіцієнт детермінації = 0.7666272612403147
```

Рисунок 8 – Статистичні характеристики очищеної від аномалій методом медіан моделі

За отриманими результатами бачимо, що до внесення аномалій: *СКВ = 0.257*, після внесення аномалій: *СКВ = 0.337*, а після усунення аномалій: *СКВ = 0.29*. Отже, можемо зробити висновок, що алгоритм відпрацьовує.

У поле виводу якості моделі також винесено підібране за статистичними характеристиками вибірки значення параметрів *розміру вікна* та *порогу*.

Бачимо також, що маємо відносно невелику *середньо-абсолютну помилку = 0.292* та відносно великий *коефіцієнт детермінації = 0.766*.

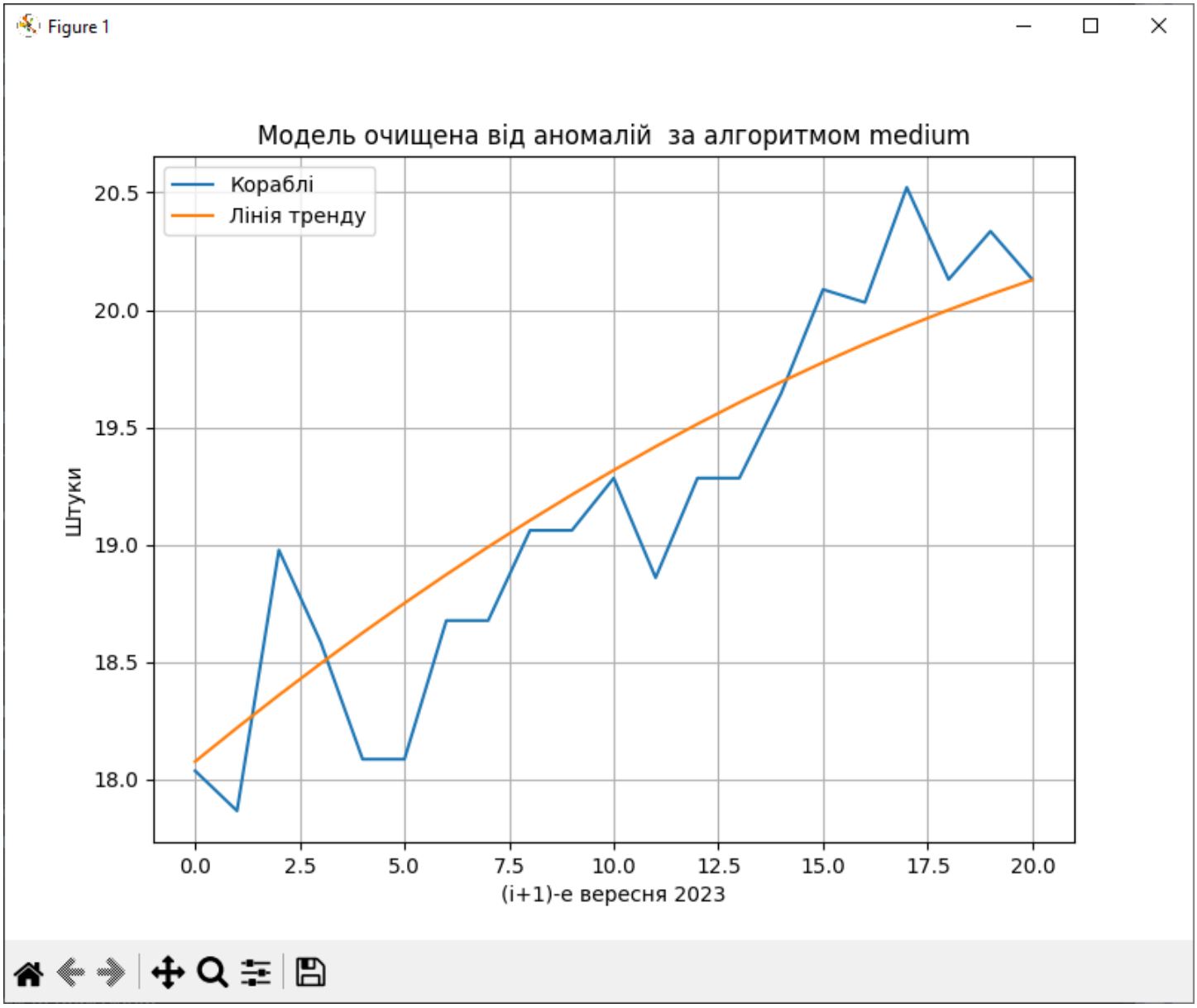

Рисунок 9 – Графік моделі, очищеної від аномалій за алгоритмом медіан

Розглянемо *алгоритм МНК*.

Алгоритм використовує лінійну регресію для апроксимації даних і визначення відхилень між спостережуваними значеннями та їхніми передбаченими значеннями на основі моделі. Аномальними точками вважаються ті, які відхиляються від лінійного тренду моделі на *порогове значення*, і можуть бути виявлені шляхом проходження по вибірці фільтром (*вікном*) і порівняння фактичних та передбачених значень.

Алгоритм приймає такі ж параметри, що й попередній: *розмір вікна* та *поріг*.

Напишемо функцію, яка буде знаходити та усувати аномалії за методом МНК.

Шляхом повторень експерименту, визначимо *залежність між об'ємом вибірки* та *розміром вікна*: *window_size = int(len(df_real_sorted) / 4)* та *залежність між середньоквадратичним відхиленням* і *пороговим значенням*: *threshold = skv * 50.*

Лістинг коду:

```
'''Блок аномалій'''

[...]

# Функція виявлення та усунення аномалій за алгоритмом МНК
def av_mnk(df, window_size, threshold):
    iter = len(df)
    j_wind = int(iter - window_size + 1)
    df_wind = np.zeros(window_size)
    # Розраховуємо коефіцієнти МНК та тренд для еталонного вікна
    speed_standart = mnk_av(df)
    df_zglad, coef = mnk(df, False)
    speed_standart_1 = abs(speed_standart * np.sqrt(iter))
    # Ковзне вікно та обробка даних
    for j in range(j_wind):
        for i in range(window_size):
            l = j + i
            df_wind[i] = df[l]
        # Виявлення аномалій
        duspers = np.var(df_wind)
        skv = np.sqrt(duspers)
        speed_1 = abs(threshold * speed_standart * np.sqrt(window_size) * skv)
        # Усунення аномалій
        if speed_1 > speed_standart_1:
            df[l] = df_zglad[l, 0]
    return df

[...]

'''Основний блок'''

[...]
```

```
# Якщо режим існує
if data_var in range(1, 3):
    [...]
    # Якщо джерело даних існує
    if data_mode in range(1, len(names) + 1):
        [...]
        # Вибір алгоритму детекції та усунення аномалій
        print('-----------------------------------------------------')
        print('Оберіть алгоритм виявлення та усунення аномалій:')
        print('1 - Метод medium')
        print('2 - Метод MNK')
        print('3 - Метод sliding window')
        av_mode = int(input('mode:'))
        # Якщо алгоритм існує
        if av_mode in range(1, 4):
            # Алгоритм медіан
            if av_mode == 1:
                [...]
            # Алгоритм MNK
            elif av_mode == 2:
                alg = 'MNK'
                # Використання характеристик даних
                window_size = int(len(df_real_sorted) / 4)
                threshold = skv * 50
                # Фіксуємо час початку обчислень
                start_time = time.time()
                df_cleaned = av_mnk(df_av, window_size, threshold)
                mae, r2 = mae_r2_score(df_real_sorted, df_cleaned)
                quality_av_detection(df_cleaned, specified_text, specified_text_1, alg,
start_time, av_mode, threshold, window_size, mae, r2, df_zglad, names, data_mode,
df_real_sorted, 'моделі', '', False)
```

Результат:

При виборі режиму роботи програми = 2, отримуємо очищену від аномалій методом МНК вибірку, її характеристики та оцінку якості роботи алгоритму.

```
--------------------------------------------------
Статистичні характеристики моделі
очищеної від аномалій  за алгоритмом MNK
--------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 6.039613253960852e-14
Дисперсія = 0.05317724788376074
Середнє квадратичне відхилення = 0.2306019251518962
Динамічна похибка моделі = 0.16411388103237054
--------------------------------------------------
Якість моделі
--------------------------------------------------
Час виконання операцій = 0.0 с
Поріг = 11.704424745240816
Розмір вікна = 5
Середньо-абсолютна помилка = 0.23378364216194045
Коефіцієнт детермінації = 0.8396430982476054
```

Рисунок 10 – Статистичні характеристики очищеної від аномалій методом МНК моделі

За отриманими результатами бачимо, що до внесення аномалій: *СКВ = 0.234*, після внесення аномалій: *СКВ = 0.354*, а після усунення аномалій: *СКВ = 0.23*. Отже, можемо зробити висновок, що алгоритм відпрацьовує.

Бачимо також, що маємо відносно невелику *середньо-абсолютну помилку = 0.233* та відносно великий *коефіцієнт детермінації = 0.839.*

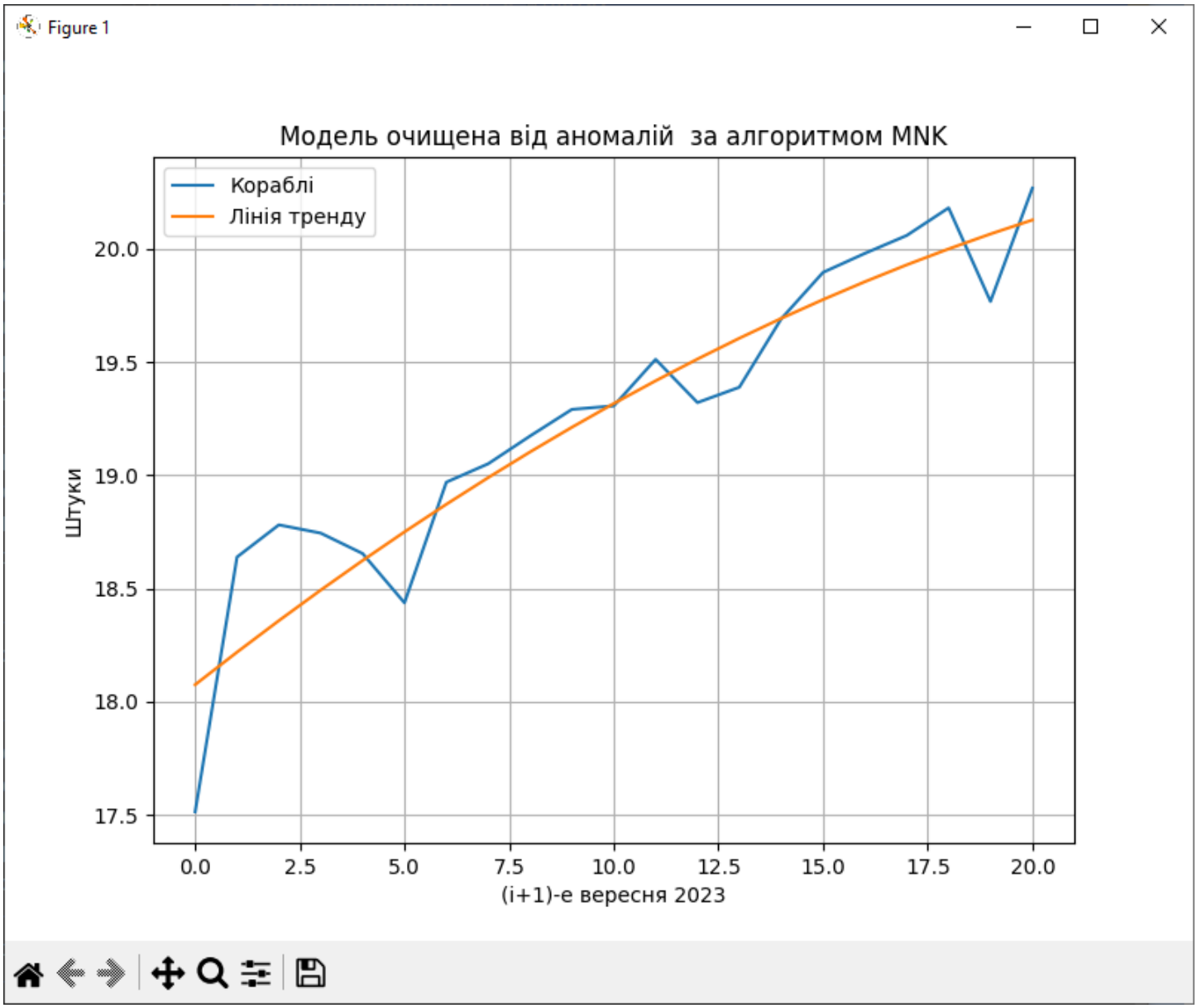

Рисунок 11 – Графік моделі, очищеної від аномалій за алгоритмом МНК

Розглянемо *алгоритм ковзного вікна*.

Суть цього алгоритму полягає в тому, що *вікно*, яке переміщується по вибірці, використовується для обчислення характеристик локального сегмента даних. За

допомогою цих локальних характеристик можна визначити аномальні точки, які відхиляються від середнього або інших параметрів цього локального сегмента.

Отже алгоритм приймає на вхід лише параметр *вікна*.

Напишемо функцію, яка буде знаходити та усувати аномалії за методом ковзного вікна.

Шляхом повторень експерименту, визначимо *залежність між об'ємом вибірки* та *розміром вікна*: *window_size = int(len(df_real_sorted) / 5).*

Лістинг коду:

```
'''Блок аномалій'''

[...]

# Функція виявлення та усунення аномалій за алгоритмом sliding window
def av_sliding_wind(df, window_size):
    iter = len(df)
    j_wind = int(np.ceil(iter - window_size) + 1)
    df_wind = np.zeros(window_size)
    midi = np.zeros(iter)
    # Ковзне вікно та обробка даних
    for j in range(j_wind):
        for i in range(window_size):
            l = (j + i)
            df_wind[i] = df[l]
        # Обчислимо медіану кожного ковзного вікна
        midi[l] = np.median(df_wind)
    # Запишемо медіану назад в df_midi
    df_midi = np.zeros(iter)
    for j in range(iter):
        df_midi[j] = midi[j]
    for j in range(window_size):
        df_midi[j] = df[j]
    return df_midi

[...]

'''Основний блок'''

[...]
# Якщо режим існує
if data_var in range(1, 3):
    [...]
    # Якщо джерело даних існує
    if data_mode in range(1, len(names) + 1):
        [...]
        # Вибір алгоритму детекції та усунення аномалій
        print('-----------------------------------------------------')
        print('Оберіть алгоритм виявлення та усунення аномалій:')
        print('1 - Метод medium')
        print('2 - Метод MNK')
        print('3 - Метод sliding window')
```

```
        av_mode = int(input('mode:'))
        # Якщо алгоритм існує
        if av_mode in range(1, 4):
            # Алгоритм медіан
            if av_mode == 1:
                [...]
            # Алгоритм МNK
            elif av_mode == 2:
                [...]
            # Алгоритм ковзаючого вікна
            else:
                alg = 'sliding_wind'
                # Використання характеристик даних
                window_size = int(len(df_real_sorted) / 5)
                # Фіксуємо час початку обчислень
                start_time = time.time()
                df_cleaned = av_sliding_wind(df_av, window_size)
                mae, r2 = mae_r2_score(df_real_sorted, df_cleaned)
                threshold = 0
                quality_av_detection(df_cleaned, specified_text, specified_text_1, alg,
start_time, av_mode, threshold, window_size, mae, r2, df_zglad, names, data_mode,
df_real_sorted, 'моделі', '', False)
```

Результат:

При виборі режиму роботи програми = 3, отримуємо очищену від аномалій методом ковзного вікна вибірку, її характеристики та оцінку якості роботи алгоритму.

```
-----------------------------------------------------
Статистичні характеристики моделі
очищеної від аномалій  за алгоритмом sliding_wind
-----------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 5.701259570265566e-14
Дисперсія = 0.030947462737884672
Середнє квадратичне відхилення = 0.1759189095517724
Динамічна похибка моделі = 0.1462535845871117
-----------------------------------------------------
Якість моделі
-----------------------------------------------------
Час виконання операцій = 0.0 с
Розмір вікна = 4
Середньо-абсолютна помилка = 0.199976150896179
Коефіцієнт детермінації = 0.8758051688016255
```

Рисунок 12 – Статистичні характеристики очищеної від аномалій методом ковзного вікна моделі

За отриманими результатами бачимо, що до внесення аномалій: *СКВ = 0.210*, після внесення аномалій: *СКВ = 0.310*, а після усунення аномалій: *СКВ = 0.175*. Отже, можемо зробити висновок, що алгоритм відпрацьовує.

Бачимо також, що маємо відносно невелику *середньо-абсолютну помилку = 0.199* та відносно великий *коефіцієнт детермінації = 0.875*.

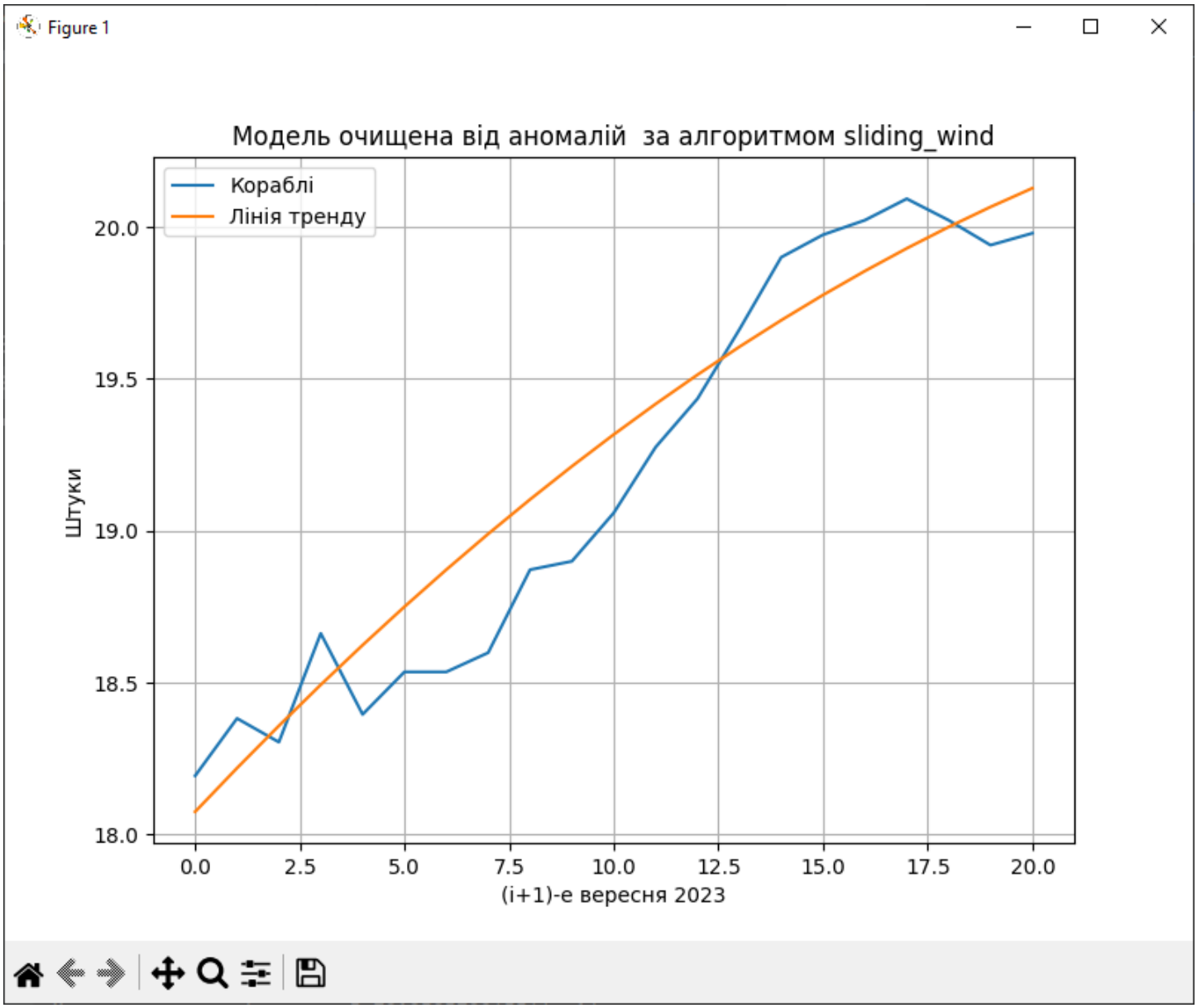

Рисунок 13 – Графік моделі, очищеної від аномалій за алгоритмом ковзного вікна

2.4. Визначення показників якості та оптимізація моделі

Показники якості моделей зазначено в попередньому пункті. Перейдемо до задачі оптимізації.

Оптимізація зводиться до знаходження *балансу* між *динамічною похибкою* моделі та *статистичною*. Візьмемо за динамічну похибку раніше визначену метрику *середньо абсолютної помилки* (MAE), а за статистичну – *коефіцієнт детермінації* (R^2).

Таким чином, наша задача зводиться до мінімізації виразу: *mae + (1 – r2)*. Напишемо для цього функцію, яка буде перебирати значення *розміру вікна* та *порогу* в заданих межах для пошуку *оптимальної комбінації MAE* з *R^2*. На виході отримаємо оптимізовані значення параметрів.

Лістинг коду:

```
[...]

'''Блок оптимізації параметрів усунення аномалій'''

# Функція для оптимізації параметрів window_size та threshold
def optimize(data0, data, min_window_size, max_window_size, min_threshold, max_threshold,
mode):
    best_window_size = None
    best_threshold = None
    best_mae = float('inf')
    best_r2 = -float('inf')
    for window_size in range(min_window_size, max_window_size + 1):
        for threshold in np.linspace(min_threshold, max_threshold, num=100):
            if mode == 1:
                cleaned_data = av_medium(data, window_size, threshold)
            if mode == 2:
                cleaned_data = av_mnk(data, window_size, threshold)
            else:
                cleaned_data = av_sliding_wind(data, window_size)
            mae, r2 = mae_r2_score(data0, cleaned_data)
            if (mae + (1 - r2)) < (best_mae + (1 - best_r2)):
                best_window_size = window_size
                best_threshold = threshold
                best_mae = mae
                best_r2 = r2
    return best_window_size, best_threshold, best_mae, best_r2

[...]

'''Основний блок'''

[...]
# Якщо режим існує
if data_var in range(1, 3):
```

```
    [...]
    # Якщо джерело даних існує
    if data_mode in range(1, len(names) + 1):
        [...]
        # Якщо алгоритм існує
        if av_mode in range(1, 4):
            # Алгоритм медіан
            if av_mode == 1:
                [...]
                # Оптимізація
                min_threshold = skv
                max_threshold = skv * 3
                # Фіксуємо час початку обчислень
                start_time = time.time()
                window_size, threshold, mae, r2 = optimize(df_real_sorted, df_av,
min_window_size, max_window_size, min_threshold, max_threshold, av_mode)
                df_cleaned = av_medium(df_av, window_size, threshold)
                quality_av_detection(df_cleaned, specified_text, specified_text_1, alg,
start_time, av_mode, threshold, window_size, mae, r2, df_zglad, names, data_mode,
df_real_sorted, 'оптимізації', 'та оптимізована', False)
            # Алгоритм MNK
            elif av_mode == 2:
                [...]
                # Оптимізація
                min_threshold = skv * 30
                max_threshold = skv * 70
                # Фіксуємо час початку обчислень
                start_time = time.time()
                window_size, threshold, mae, r2 = optimize(df_real_sorted, df_av,
min_window_size, max_window_size, min_threshold, max_threshold, av_mode)
                df_cleaned = av_mnk(df_av, window_size, threshold)
                quality_av_detection(df_cleaned, specified_text, specified_text_1, alg,
start_time, av_mode, threshold, window_size, mae, r2, df_zglad, names, data_mode,
df_real_sorted, 'оптимізації', 'та оптимізована', False)
            # Алгоритм ковзаючого вікна
            else:
                [...]
                # Оптимізація
                min_threshold = 0
                max_threshold = 0
                # Фіксуємо час початку обчислень
                start_time = time.time()
                window_size, threshold, mae, r2 = optimize(df_real_sorted, df_av,
min_window_size, max_window_size, min_threshold, max_threshold, av_mode)
                df_cleaned = av_sliding_wind(df_av, window_size)
                quality_av_detection(df_cleaned, specified_text, specified_text_1, alg,
start_time, av_mode, threshold, window_size, mae, r2, df_zglad, names, data_mode,
df_real_sorted, 'оптимізації', 'та оптимізована', False)
```

Результат:

У результаті отримаємо оптимізовану очищену від аномалій модель, її характеристики та графік.

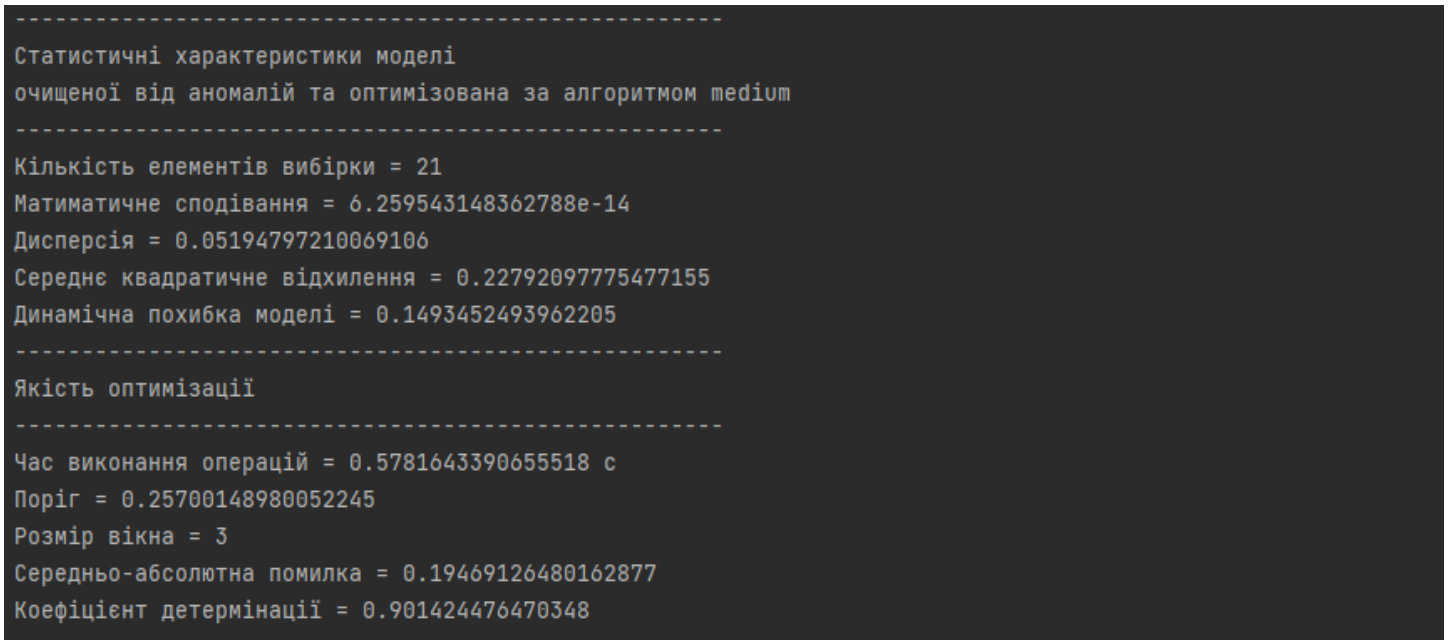

Рисунок 14 – Статистичні характеристики оптимізованої очищеної за алгоритмом медіан моделі

З отриманих даних чудово видно, що *середньо-абсолютна помилка* зменшилась *з 0.292 до 0.195*, а *коефіцієнт детермінації* збільшився *з 0.766 до 0.901*. Що свідчить про успішну оптимізацію моделі.

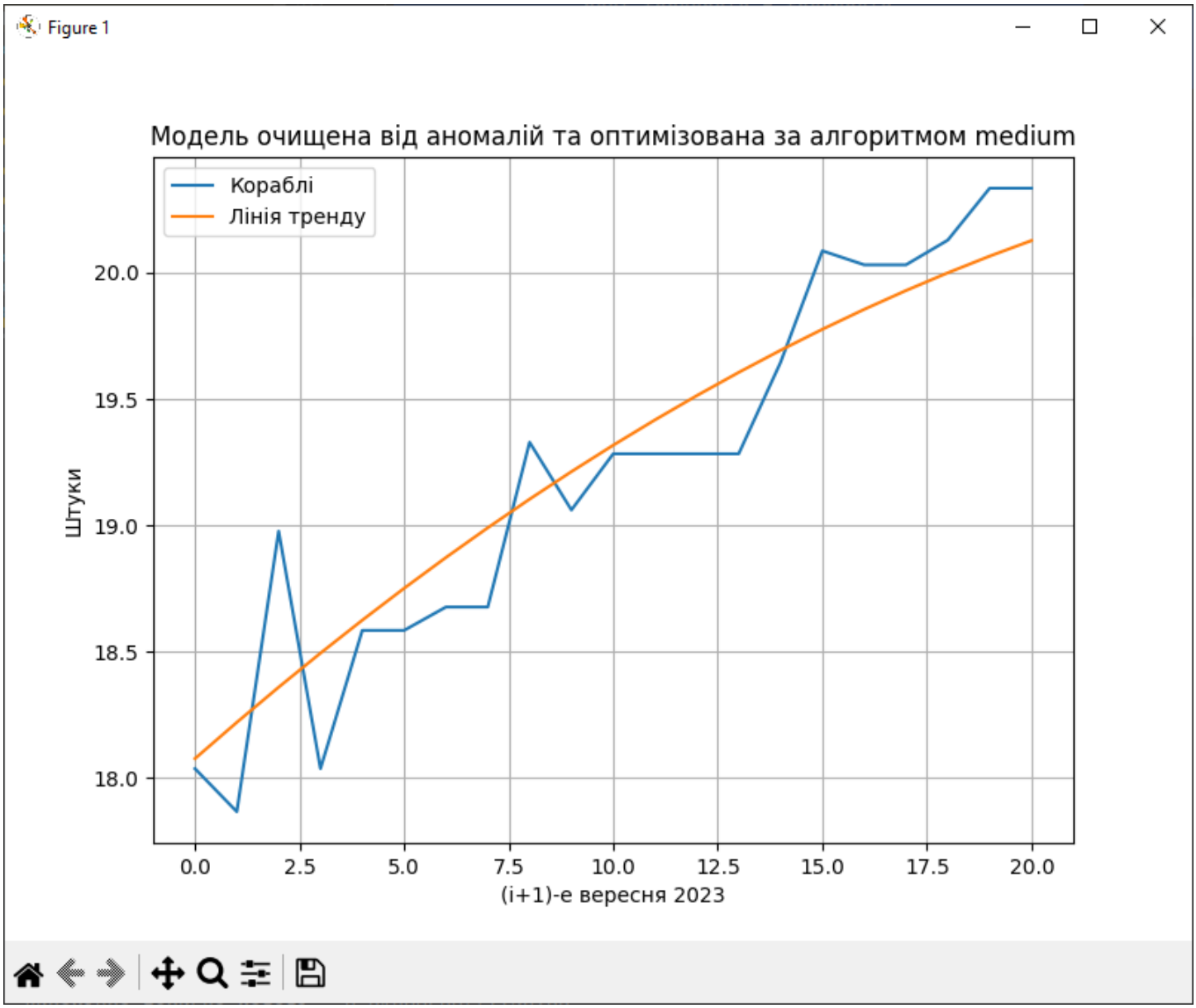

Рисунок 15 – Графік оптимізованої моделі, очищеної від аномалій за алгоритмом медіан

Можемо також переконатися в результатах успішної оптимізації усунення аномалій, порівнявши рисунки 9 та 16.

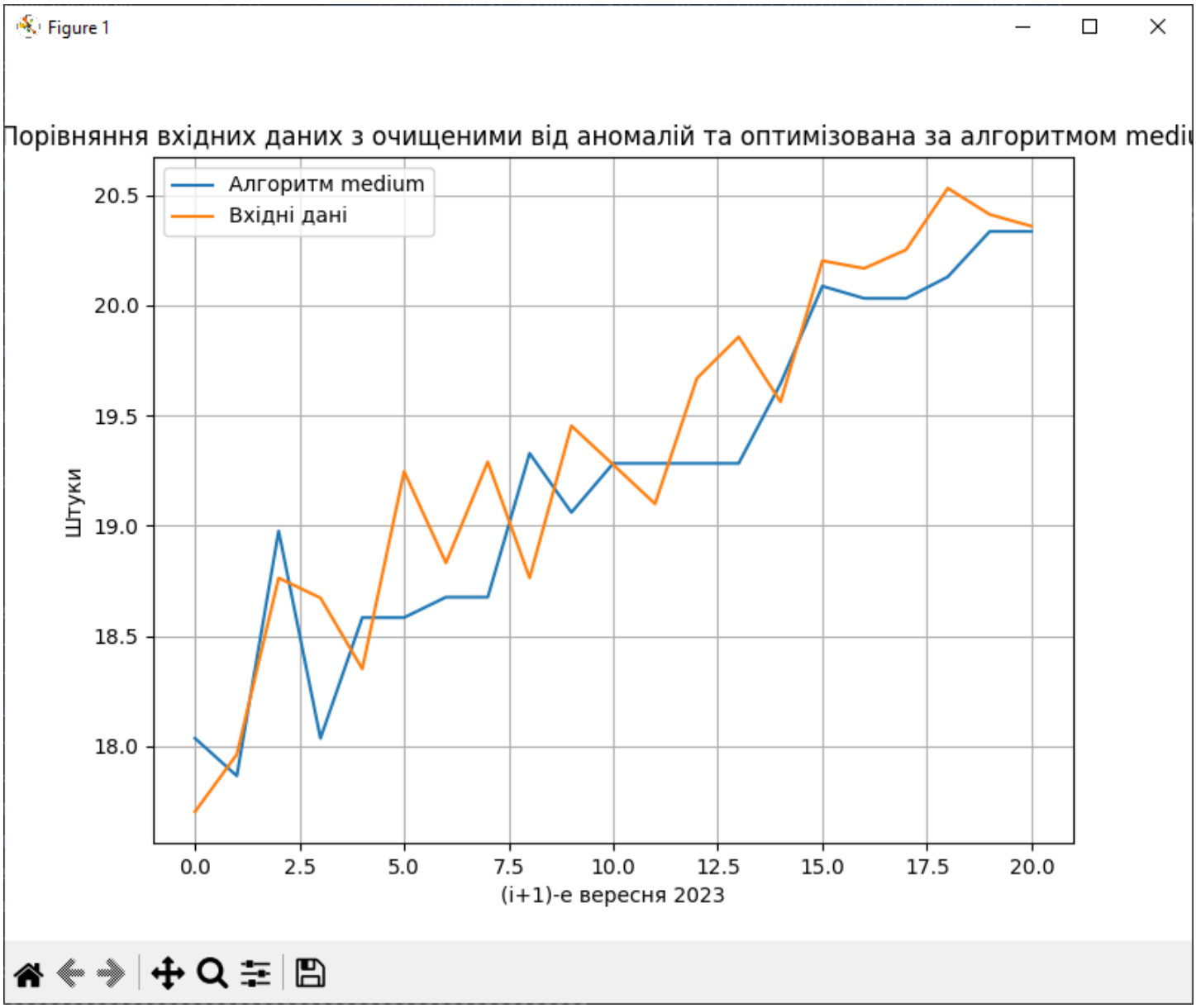

Рисунок 16 – Порівняння графіків вхідних даних з очищеною за алгоритмом медіан й оптимізованою моделлю

Аналогічно оптимізуємо методи МНК та ковзного вікна.

2.5. Реалізувати рекурентне згладжування alfa-beta фільтром сформованих в п.1, 2 вхідних даних.

Так, як усі отримані дані мають *лінійну залежність*, будемо використовувати *фільтр alfa-beta*.

Лістинг коду:

```
[...]

'''Блок фільтрації'''

# Функція алгоритму -a-b фільтру
def a_b_filter(df):
    iter = len(df)
    df_1 = np.zeros((iter, 1))
    df_filtred = np.zeros((iter, 1))
    t = 1
    for i in range(iter):
        df_1[i, 0] = float(df[i])
    # Початкові дані для запуску фільтра
    df_speed_retro = (df_1[1, 0] - df_1[0, 0]) / t
    df_extra = df_1[0, 0] + df_speed_retro
    alfa = 2 * (2 * 1 - 1) / (1 * (1 + 1))
    beta = (6 / 1) * (1 + 1)
    df_filtred[0, 0] = df_1[0, 0] + alfa * (df_1[0, 0])
    # Рекурентний прохід по вимірам
    for i in range(1, iter):
        df_filtred[i, 0] = df_extra + alfa * (df_1[i, 0] - df_extra)
        df_speed = df_speed_retro + (beta / t) * (df_1[i, 0] - df_extra)
        df_speed_retro = df_speed
        df_extra = df_filtred[i, 0] + df_speed_retro
        alfa = (2 * (2 * i - 1)) / (i * (i + 1))
        beta = 6 / (i * (i + 1))
    return df_filtred

[...]

'''Основний блок'''

[...]
# Якщо режим існує
if data_var in range(1, 3):
    [...]
    # Якщо джерело даних існує
    if data_mode in range(1, len(names) + 1):
        [...]
        # Якщо алгоритм існує
        if av_mode in range(1, 4):
            [...]
            # Фільтрація
            df_filtred = a_b_filter(df_cleaned)
            mae, r2 = mae_r2_score(df_real_sorted, df_filtred)
            quality_av_detection(df_filtred, specified_text, specified_text_1, alg, 0,
av_mode, 0, 0, mae, r2, df_zglad, names, data_mode, df_real_sorted, 'фільтрації', '',
True)
```

Результат:

Отримано модель фільтрації та оцінку її якості.

```
--------------------------------------------------
Якість фільтрації
--------------------------------------------------
Середньо-абсолютна помилка = 1.6341440268966214
Коефіцієнт детермінації = -478.1513637441755
```

Рисунок 17 – Оцінка якості фільтрації

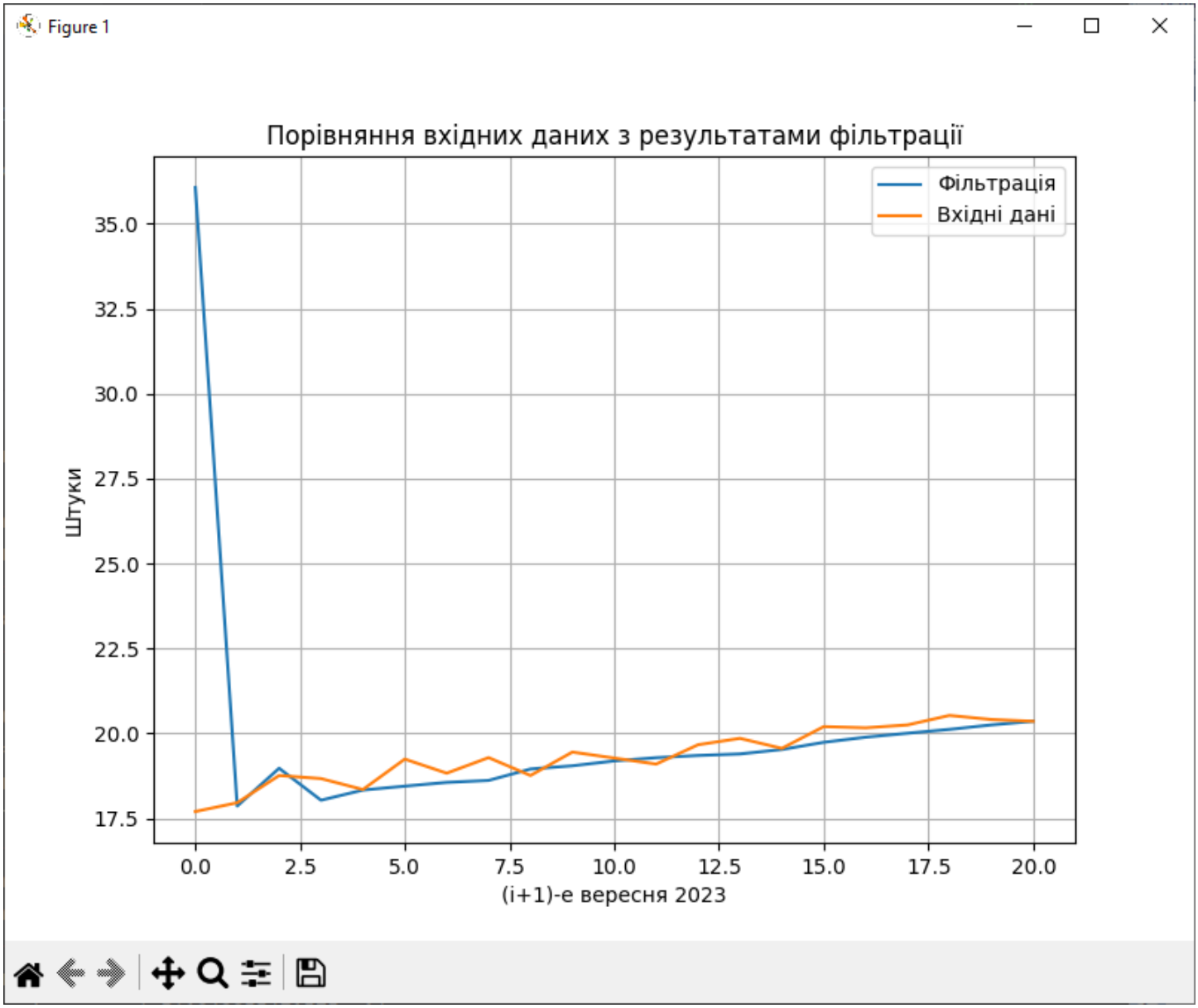

Рисунок 18 – Графік фільтрації моделі

З графіка можемо бачити, що фільтр спочатку навчається, а потім переходить в стадію насичення і показує непогані результати. Бачимо, що фільтр успішно відслідковує дані.

2.6. Провести аналіз отриманих результатів

За вхідну вибірку ми взяли *реальні дані*, отримані в лабораторній роботі № 1. Залишили можливість обирати працювати з реальними даними, чи моделлю, синтезованою за їх трендом.

До вхідних даних додані аномальні виміри – підтверджено зростанням значення середньоквадратичного відхилення.

СКВ до аномалій = *0.257*

СКВ після аномалій = *0.337*

```
-----------------------------------------------------
Статистичні характеристики моделі + шум
-----------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 6.259543148362788e-14
Дисперсія = 0.06604976575968806
Середнє квадратичне відхилення = 0.25700148980052245
Динамічна похибка моделі = 0.2047111772905452
-----------------------------------------------------
Статистичні характеристики моделі + шум + аномалії
-----------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 6.107283990699909e-14
Дисперсія = 0.113828504414611
Середнє квадратичне відхилення = 0.3373848016947577
Динамічна похибка моделі = 0.26601765590754006
```

Рисунок 19 – Вивід, що підтверджує успішне додавання аномалій

Вхідні дані очищені від аномальних вимірів трьома алгоритмами – підтверджено зменшенням значення середньоквадратичного відхилення.

Алгоритм медіан:

СКВ до аномалій = *0.257*

СКВ після аномалій = *0.337*

СКВ після усунення аномалій = *0.29*

Алгоритм МНК:

СКВ до аномалій = *0.234*

СКВ після аномалій = *0.354*

СКВ після усунення аномалій = *0.23*

Алгоритм ковзного вікна:

СКВ до аномалій = *0.21*

СКВ після аномалій = *0.31*

СКВ після усунення аномалій = *0.175*

```
--------------------------------------------------
Статистичні характеристики моделі
очищеної від аномалій  за алгоритмом medium
--------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 5.938107148852266e-14
Дисперсія = 0.08430304102528766
Середнє квадратичне відхилення = 0.2903498596956569
Динамічна похибка моделі = 0.21299589664478302
--------------------------------------------------
Статистичні характеристики моделі
очищеної від аномалій  за алгоритмом MNK
--------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 6.039613253960852e-14
Дисперсія = 0.05317724788376074
Середнє квадратичне відхилення = 0.2306019251518962
Динамічна похибка моделі = 0.16411388103237054
--------------------------------------------------
Статистичні характеристики моделі
очищеної від аномалій  за алгоритмом sliding_wind
--------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 5.701259570265566e-14
Дисперсія = 0.030947462737884672
Середнє квадратичне відхилення = 0.1759189095517724
Динамічна похибка моделі = 0.1462535845871117
```

Рисунок 20 – Вивід, що підтверджує успішне усунення аномалій алгоритмами медіан, МНК, ковзного вікна

Найкраще себе показав алгоритм очищення від аномалій *ковзного вікна*.

Визначені показники якості моделі:

середньо абсолютна помилка (MAE) та *коефіцієнт детермінації* (R^2).

Обраний спосіб оптимізації моделі:

мінімізація виразу: *mae + (1 – r2).*

Модель очищення вибірки від аномалій оптимізовано – підтверджено виводом:

середньо-абсолютна помилка до оптимізації = *0.292*

середньо-абсолютна помилка після оптимізації = *0.19*5

коефіцієнт детермінації до оптимізації = *0.766*

коефіцієнт детермінації після оптимізації = *0.901*

```
--------------------------------------------------
Якість моделі
--------------------------------------------------
Час виконання операцій = 0.0 с
Поріг = 0.5140029796010449
Розмір вікна = 5
Середньо-абсолютна помилка = 0.29228059709914167
Коефіцієнт детермінації = 0.7666272612403147
--------------------------------------------------
Якість оптимізації
--------------------------------------------------
Час виконання операцій = 0.5781643390655518 с
Поріг = 0.25700148980052245
Розмір вікна = 3
Середньо-абсолютна помилка = 0.19469126480162877
Коефіцієнт детермінації = 0.901424476470348
```

Рисунок 21 – Графік, що підтверджує успішну оптимізацію моделі усунення аномалій

Обрано структуру фільтру:

використано *alfa-beta фільтр* через *лінійну залежність даних*.

Накладено фільтр на вхідні дані – підтверджено графіком:

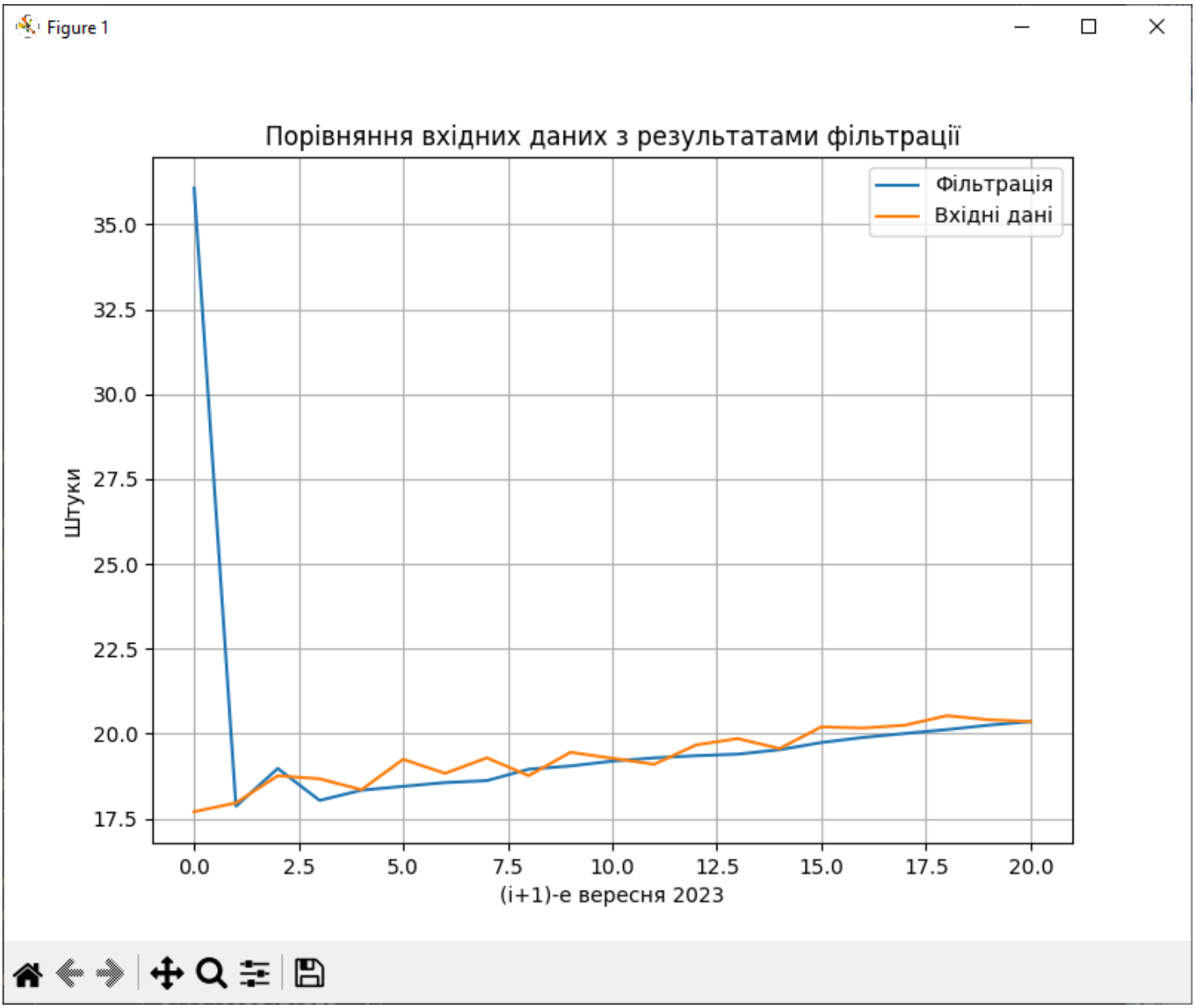

Рисунок 22 – Графік, що підтверджує успішне накладання фільтру на вхідні дані

Знайдено залежність між параметрами алгоритмів усунення аномалій та статистичними характеристиками вибірки – підтверджено отриманим задовільним результатом очищення від аномалій:

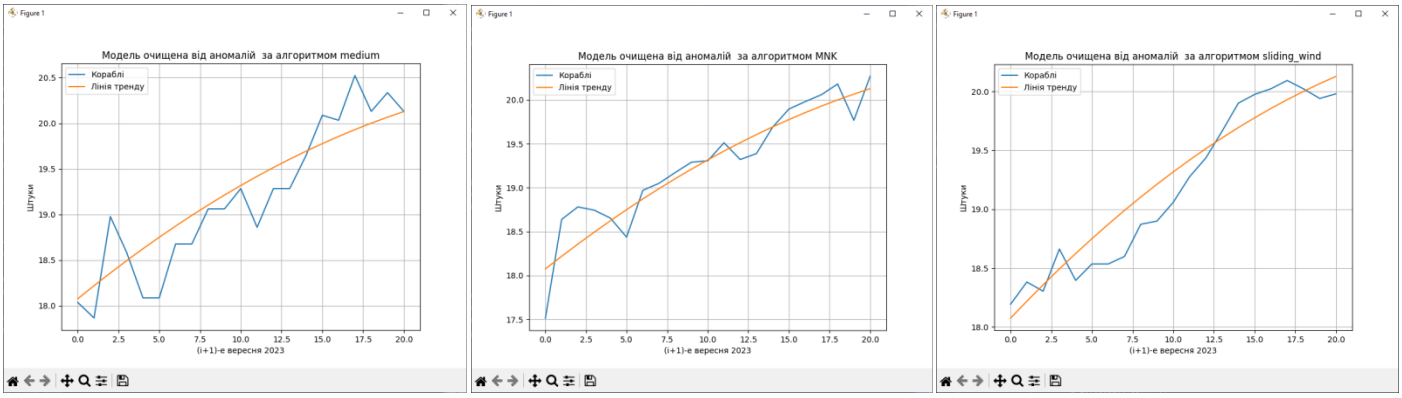

Рисунок 23 – Графіки, що підтверджують задовільний результат усунення аномалій і успішне знаходження залежностей між параметрами алгоритмів та характеристиками вибірки

Висновок:

Ми дослідили та узагальнили особливості реалізації процесів статистичного навчання, використовуючи методи обробки великих масивів даних (Big Data) та Калманівської рекурентної фільтрації.

Експериментально підтверджено, що статистичні характеристики вибірки (об’єм вибірки, математичне сподівання, середньоквадратичне відхилення) грають важливу роль у виявленні аномалій та побудові алгоритмів очищення даних.

На основі спостережень про вплив характеристик вибірки на параметри алгоритмів усунення аномалій, ми реалізували та оптимізували різні алгоритми виявлення та усунення аномалій, такі як метод медіан, метод МНК та алгоритм ковзного вікна.

Розроблену програму можна використовувати для завантаження, обробки та аналізу великих обсягів даних. Вона допомагає виявляти закономірності та залежності в даних, що може бути корисним для прийняття рішень.