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

Кафедра інформаційних систем та технологій

Лабораторна робота №1

з дисципліни «Вступ до технології Data Science»

на тему

«ПІДГОТОВКА ТА АНАЛІЗ ДАНИХ ДЛЯ СТАТИСТИЧНОГО НАВЧАННЯ»

Виконала:
студентка групи ІС-12
Павлова Софія

Перевірив:
Баран Д. Р.

Київ – 2023

1. Постановка задачі

Мета роботи:

Виявити дослідити та узагальнити особливості застосування методів статистичного навчання для задач визначення статистичних характеристик вхідного потоку даних з використанням спеціалізованих пакетів мови програмування Python.

Завдання III рівня:

1. Провести парсинг самостійно обраного сайту. Вміст даних, що підлягають парсингу – обрати самостійно.
2. Результати парсингу зберегти у файлі. Тип файлу обрати самостійно.
3. Оцінити динаміку тренду реальних даних.
4. Здійснити визначення статистичних характеристик результатів парсингу.
5. Синтезувати та верифікувати модель даних, аналогічних за трендом і статистичними характеристиками реальним даним, які є результатом парсингу.
6. Провести аналіз отриманих результатів.

2. Виконання

2.1. Провести парсинг самостійно обраного сайту. Вміст даних, що підлягають парсингу – обрати самостійно

Зважаючи на основну мету парсингу – збір та систематизацію даних, та події, які зараз актуально моніторити, для парсингу було обрано сайт https://www.minusrus.com/. Даними, які будемо парсити будуть втрати російського війська з початку повномасштабного вторгнення.

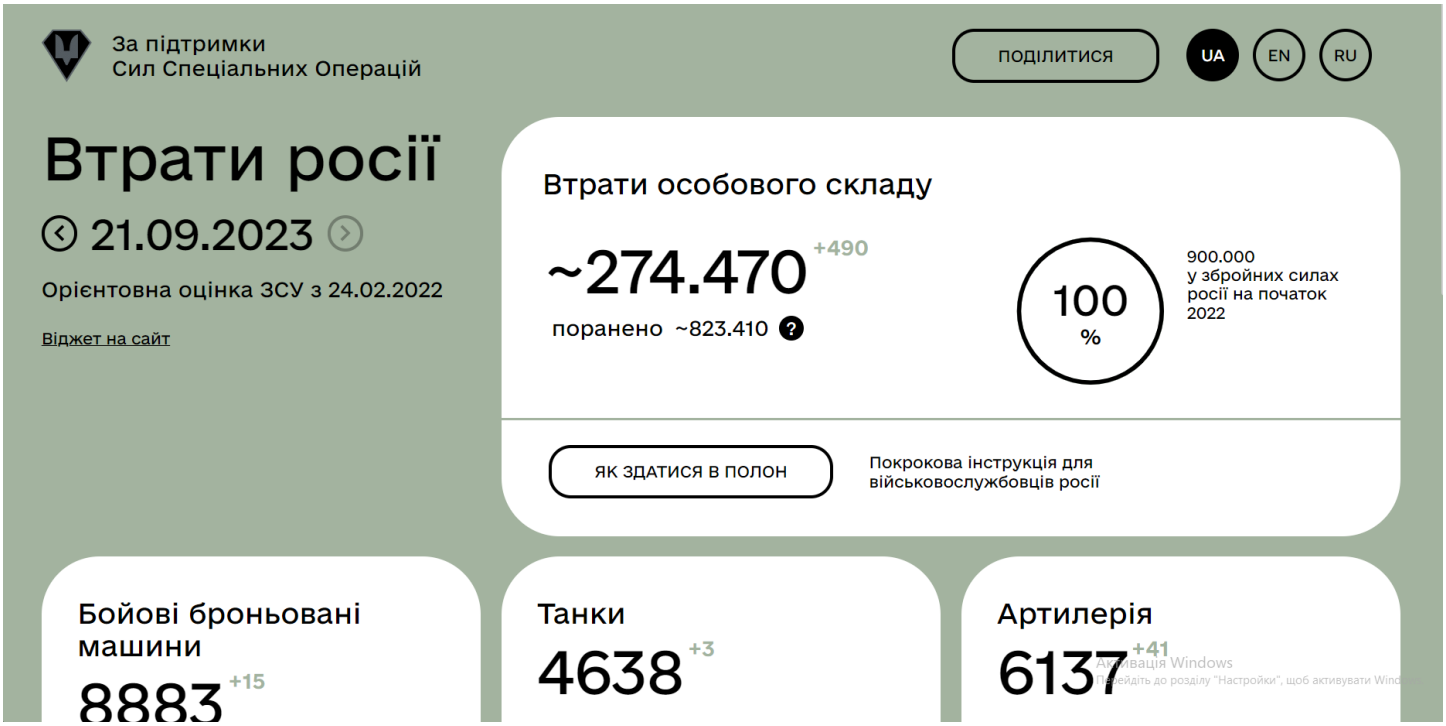

Рисунок 1 – Вигляд сайту для парсингу

Категоріями даних виступатимуть: *Особовий склад*, *Бойові броньовані машини*, *Танки*, *Артилерія*, *Літаки*, *Гелікоптери*, *Кораблі*.

Лістинг коду:

```
import requests
from bs4 import BeautifulSoup as bs
import pandas as pd

# Відповіді HTTP-сервера:

# 200: запит виконано успішно.
# 400: запит не сформовано належним чином.
# 401: несанкціонований запит, клієнт повинен надіслати дані автентифікації.
# 404: вказаний у запиті ресурс не знайдено.
# 500: внутрішня помилка сервера HTTP.
# 501: запит не реалізований сервером HTTP.
# 502: служба не доступна.

# URL веб-сторінки, яку будемо парсити
url = "https://www.minusrus.com/"

# Очищення значень
def clean_number(raw_number):
    cleaned_number = ''.join(filter(lambda char: char.isdigit(), raw_number))
    return int(cleaned_number)

def parsing_site (url):
    # Відправляємо GET-запит на веб-сторінку
    response = requests.get(url)
    # Створюємо словник для збереження результатів
    result_list = {'date': [], 'personnel': [], 'armed_vehicles': [], 'tanks': [],
'artilleries': [], 'aircrafts': [], 'helicopters': [], 'ships': []}
    print('-------------------------------------')
    print('Статус код: ' + str(response.status_code))
    print('-------------------------------------')
    print('HTML сторінка:\n\n' + response.text)
    # Перевіряємо, чи успішно отримали сторінку
    if response.status_code == 200:
        soup = bs(response.text, "html.parser")
        dates = soup.find_all('span', class_="date__label")
        personnel = soup.find_all('span', class_="card__amount-total")[0]
        armed_vehicles = soup.find_all('span', class_="card__amount-total")[1]
        tanks = soup.find_all('span', class_="card__amount-total")[2]
        artilleries = soup.find_all('span', class_="card__amount-total")[3]
        aircrafts = soup.find_all('span', class_="card__amount-total")[4]
        helicopters = soup.find_all('span', class_="card__amount-total")[5]
        ships = soup.find_all('span', class_="card__amount-total")[6]
        # Вносимо дані в список
        for date in dates:
            result_list['date'].append(date.text)
        for person in personnel:
            result_list['personnel'].append(clean_number(person.text))
        for a_v in armed_vehicles:
            result_list['armed_vehicles'].append(int(a_v.text))
        for tank in tanks:
            result_list['tanks'].append(int(tank.text))
        for artillery in artilleries:
            result_list['artilleries'].append(int(artillery.text))
        for aircraft in aircrafts:
            result_list['aircrafts'].append(int(aircraft.text))
        for helicopter in helicopters:
            result_list['helicopters'].append(int(helicopter.text))
        for ship in ships:
            result_list['ships'].append(int(ship.text))
```

```python
# Виводимо результати парсингу
    print('-------------------------------------')
    print('Результат париснгу:')
    print('-------------------------------------')
    print(result_list['date'])
    print(result_list['personnel'])
    print(result_list['armed_vehicles'])
    print(result_list['tanks'])
    print(result_list['artilleries'])
    print(result_list['aircrafts'])
    print(result_list['helicopters'])
    print(result_list['ships'])
    return result_list
```

Результат:

У результаті виконання програми отримуємо скрипт HTML-сторінки сайту, та отримані дані парсингу:

```
-----------------------------------
Статус код: 200
-----------------------------------
HTML сторінка:

<!doctype html>
<html data-n-head-ssr lang="uk_UA" data-n-head="%7B%22lang%22:%7B%22ssr%22:%22uk_UA%22%7D%7D">
  <head >
    <meta data-n-head="ssr" charset="utf-8"><meta data-n-head="ssr" name="viewport" content="width=device-w
          window.dataLayer = window.dataLayer || [];
          function gtag(){dataLayer.push(arguments);}
          gtag(&#x27;js&#x27;, new Date());

          gtag(&#x27;config&#x27;, &#x27;G-XXXXXX&#x27;);
        </script><link rel="preload" href="/_nuxt/c8ace95.js" as="script"><link rel="preload" href="/_nuxt/
.resize-observer[data-v-8859cc6c]{background-color:transparent;border:none;opacity:0}.resize-observer[data-
.nuxt-progress{background-color:#000;height:2px;left:0;opacity:1;position:fixed;right:0;top:0;transition:wi
.main{width:calc(33.33333% - var(--grid-gutter)*2/3)}.main__wrapper{display:flex;justify-content:space-betw
.header{display:flex;justify-content:space-between;padding:32px 0 40px}.logo{align-items:center;display:fle
.share-variety{cursor:pointer;padding:10px}.share-variety:hover{color:gray}
.vue-circular-progress{display:inline-block}.vue-circular-progress .circle{position:relative}.vue-circular-
.support[data-v-7e0c7f8e]{display:flex;justify-content:space-around;padding:88px 58px 32px}.support__block[
.footer{border-top:2px solid var(--black);margin-top:56px;padding:22px 0 72px}.footer p{font-size:18px;line
  </head>
  <body >
    <div data-server-rendered="true" id="__nuxt"><!----><div id="__layout"><div class="container"><header c
          +420
        </span></div> <div class="amount-details"><div class="amount-details__item"><span>поранено</span> <
            ~828.810
            <span class="base-tooltip">?</span></span></div></div></div> <div class="statistics"><div class
  </body>
</html>
```

Рисунок 2 – Скрипт сайту

```
-----------------------------------
Результат парисngу:
-----------------------------------
['25.09.2023']
[276270]
[8927]
[4667]
[6260]
[315]
[316]
[20]
```

Рисунок 3 – Дані парсингу

2.2. Результати парсингу зберегти у файлі. Тип файлу обрати самостійно

Результати парсингу будемо зберігати в файлі *.xlsx*.

Код програми напишемо так, щоб наш парсер дописував нові дані в існуючу таблицю, а не створював кожного разу нову. Таким чином, через декілька днів парсингу можна накопичити необхідний дата сет.

Лістинг коду:

```
# Читаємо існуючу таблицю з файлу, якщо вона існує
try:
    existing_df = pd.read_excel("rosnia_new.xlsx")
except FileNotFoundError:
    existing_df = None

# Перетворюємо результати парсингу в DataFrame
new_df = pd.DataFrame(data=parsing_site(url))

# Перевіряємо, чи є існуюча таблиця
if existing_df is not None:
    # Дописуємо нові дані до існуючої таблиці
    updated_df = pd.concat([existing_df, new_df], ignore_index=True)
else:
    updated_df = new_df

# Зберігаємо оновлену таблицю
updated_df.to_excel("rosnia_new.xlsx", index=False)
```

Результат:

Створилась таблиця «rosnia.xlsx», у кожен стовпець якої додано інформацію по втратах росії за вказаний день:

	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	25.09.2023	276270	8927	4667	6260	315	316	20						
3														

Рисунок 4 – Таблиця з вмістом парсингу

Повторимо парсинг сайту, змінюючи дату, за яку хочемо отримати інформацію. Зберемо дані за 1-21 вересня 2023 року.

	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 5 – Датасет, з яким будемо працювати

2.3. Оцінити динаміку тренду реальних даних

Для початку зчитаємо дані з таблиці та відсортуємо їх у порядку зростання дати. Програму реалізуємо таким чином, щоб користувач міг обирати режим зчитування даних.

Лістинг коду:

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

'''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 зчитування даних
print('Оберіть джерело вхідних даних та подальші дії:')
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])
```

Результат:

При виборі режиму роботи програми = 1, отримуємо вибірку стовпця «personnel» – втрати особового складу.

```
Оберіть джерело вхідних даних та подальші дії:
1 - Особовий склад
2 - Бойові броньовані машини
3 - Танки
4 - Артилерія
5 - Літаки
6 - Гелікоптери
7 - Кораблі
mode:1
--------------------------------------------------
Вибірка - personnel
--------------------------------------------------
0     263490
1     264060
2     264660
3     265120
4     265680
5     266290
6     266900
7     267540
8     268140
9     268630
10    269210
11    269760
12    270350
13    270970
14    271440
15    271790
16    272320
17    272940
18    273460
19    273980
20    274470
Name: personnel, dtype: int64
Джерело даних:  https://www.minusrus.com/
```

Рисунок 6 – Вибірка, з якою будемо працювати

Визначимо характеристики вибірки та лінію тренду.

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

'''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'''

    [...]
if data_mode in range(1, len(names) + 1):
    [...]
    # Визначення тренду
    df_zglad, coef = mnk(df_real_sorted, True)
    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')
```

Результат:

У результаті отримуємо регресійну модель та графік розподілу випадкової величини.

```
------------------------------------------------------
Регресійна модель
------------------------------------------------------
y(t) =  263418.70129870024  +  602.9829118250829  * t  +  -2.5153793574886176  * t^2
```

Рисунок 7 – Динаміка лінії тренду

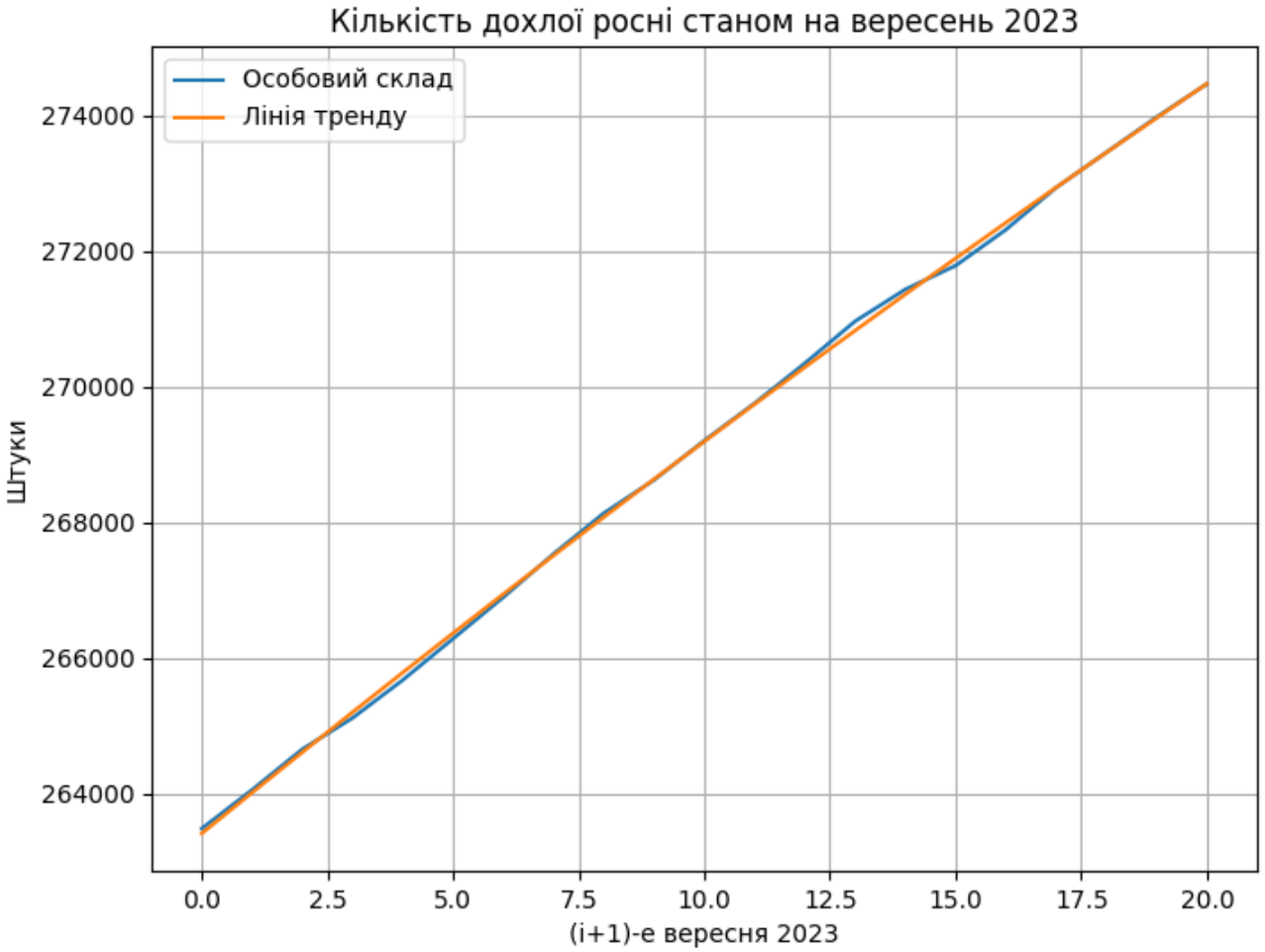

Рисунок 8 – Графік розподілу реальних даних

Отримано наступні коефіцієнти регресійної моделі:

y(t) = *263418.70129870024* + *602.9829118250829* * t + *-2.5153793574886176* * t^2

Бачимо, що графік має лінійну залежність.

2.4. Здійснити визначення статистичних характеристик результатів парсингу

Лістинг коду:

```
[...]

'''Блок визначення характеристик'''

# Для вибірки
def stat_characteristics_in(df):
    # 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)
    print('-----------------------------------------------------')
    print('Статистичні характеристики вибірки')
    print('-----------------------------------------------------')
    print('Кількість елементів вибірки =', iter)
    print('Матиматичне сподівання =', mat_spod)
    print('Дисперсія =', duspers)
    print('Середнє квадратичне відхилення =', ser_kvad_vid)
    return mat_spod, ser_kvad_vid

'''Основний блок'''

    [...]
if data_mode in range(1, len(names) + 1):
    [...]
# Визначення характеристик вибірки
ser, skv = stat_characteristics_in(df_real_sorted)
```

Результат:

```
-----------------------------------------------------
Статистичні характеристики вибірки
-----------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 9.479533348764691e-10
Дисперсія = 4337.055626077996
Середнє квадратичне відхилення = 65.85632563450527
```

Рисунок 9 – Статистичні характеристики вибірки

2.5. Синтезувати та верифікувати модель даних, аналогічних за трендом і статистичними характеристиками реальним даним, які є результатом парсингу

За отриманою регресійною моделлю, синтезуємо штучну вибірку.

Лістинг коду:

```
[...]

'''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

'''Основний блок'''

    [...]
if data_mode in range(1, len(names) + 1):
    [...]
    # Синтезація моделі
    model = model(coef[0, 0], coef[1, 0], coef[2, 0], data_mode)
```

Результат:

Отримано модель, аналогічну за трендом до реальних даних.

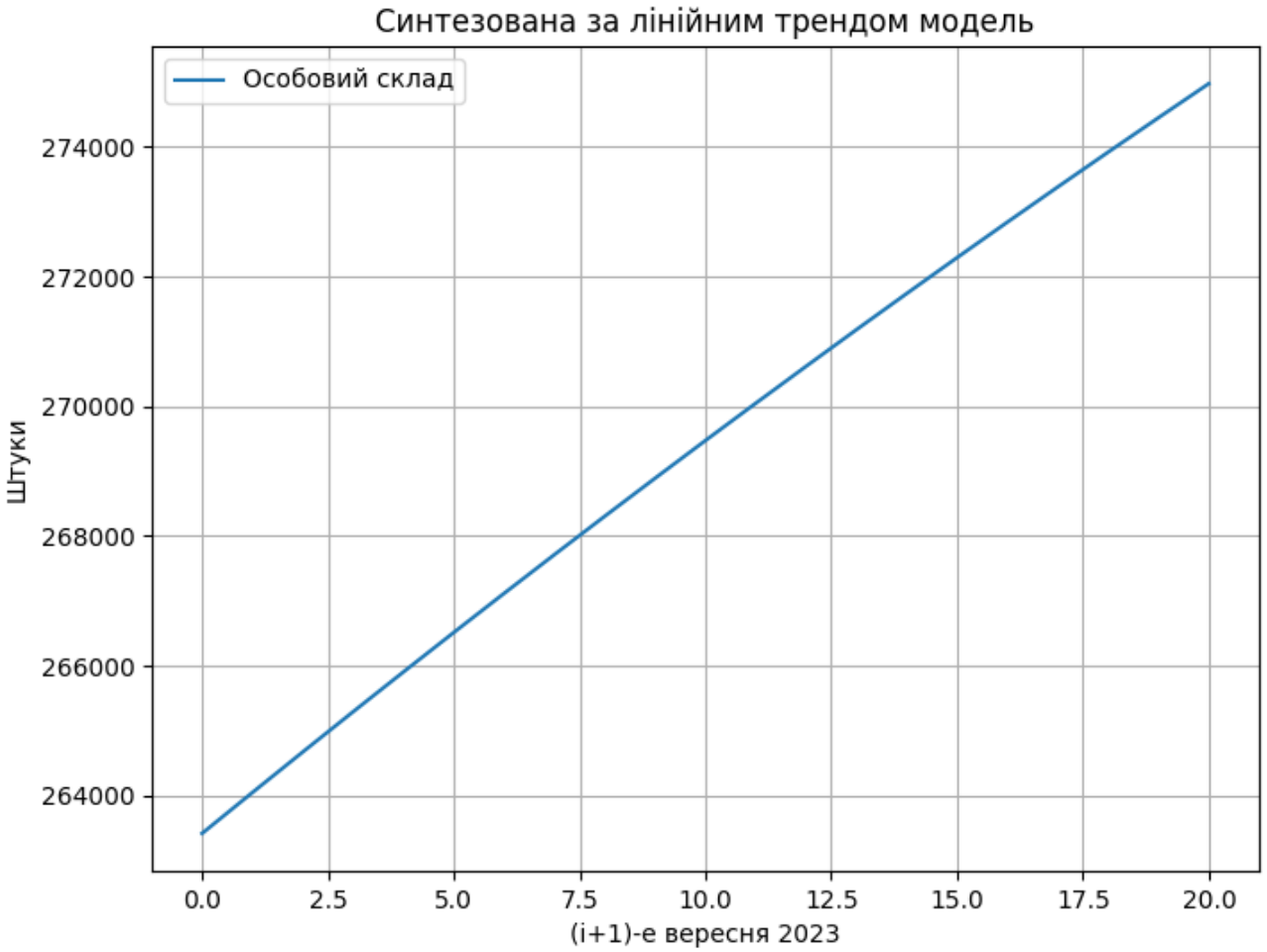

Рисунок 10 – Синтезована за трендом модель

Додамо до моделі стохастичний шум, так, як реальні дані часто містять випадковий шум. Використаємо *нормальний закон* для генерації стохастичного шуму.

Лістинг коду:

```
[...]

'''Блок моделі'''

[...]
# Функція додавання нормального шуму
def norm_shum (model, n, m, skv):
    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
```

```
'''Основний блок'''

    [...]
if data_mode in range(1, len(names) + 1):
    [...]
    # Додавання стохастичного шуму
    model_shum = norm_shum(model, len(model), ser, skv)
    plot(model_shum, model_shum, 'Модель + нормальний шум', names[data_mode-1],
names[data_mode-1], '(i+1)-е вересня 2023', 'Штуки')
```

Результат:

У результаті отримаємо графік моделі з нормальним шумом.

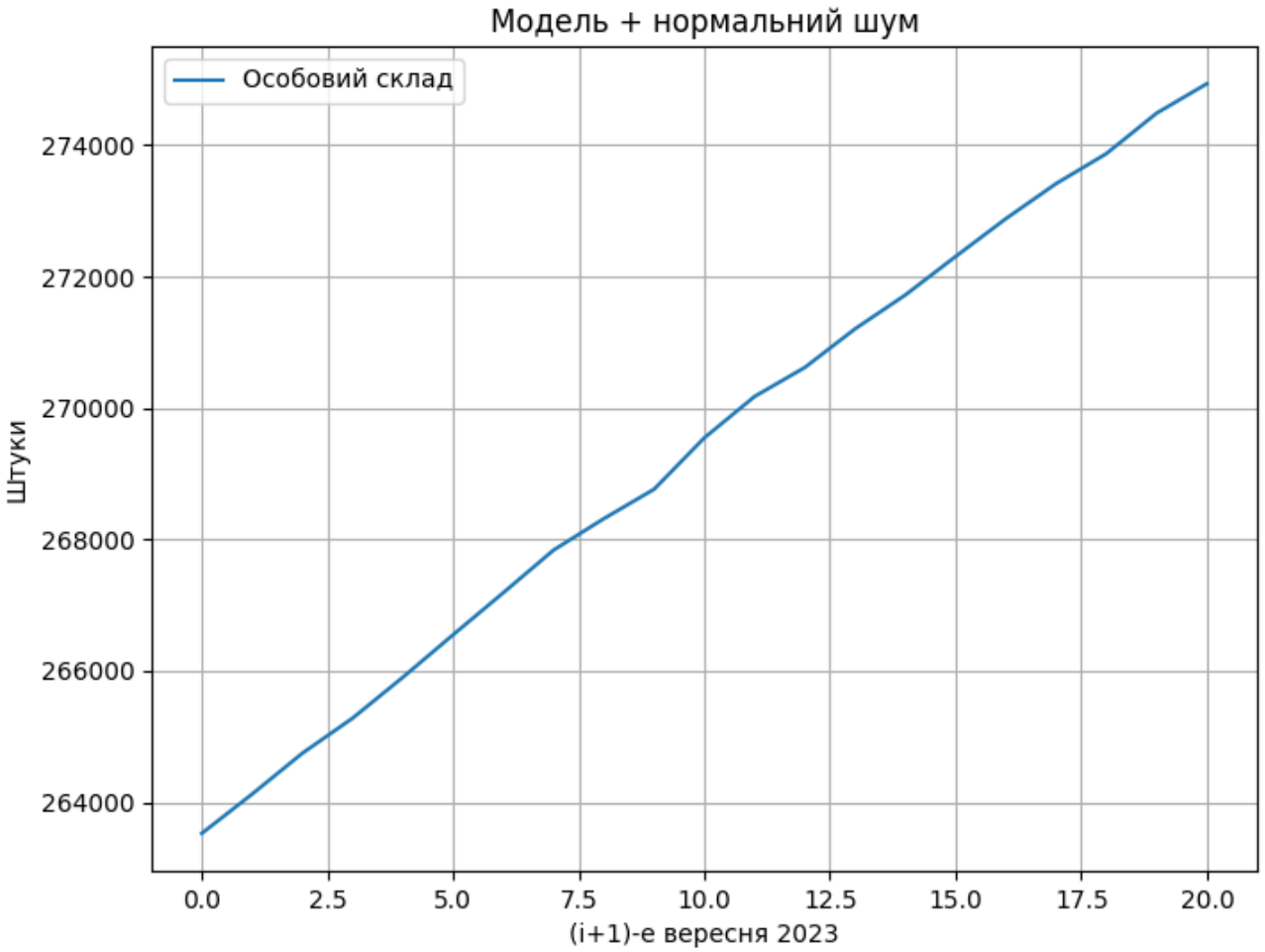

Рисунок 11 – Синтезована модель з нормальним шумом

Визначимо характеристики синтезованої моделі.

Лістинг коду:

```
[...]

'''Блок визначення характеристик'''

[...]

# Для моделі
def stat_characteristics_out (df):
    # 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 моделі')
    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)

[...]

'''Основний блок'''

    [...]
if data_mode in range(1, len(names) + 1):
    [...]
    # Визначення характеристик моделі
    stat_characteristics_out(model_shum)
```

Результат:

Бачимо, що статистичні характеристики моделі наближені до характеристик реальних даних. Про це також свідчить відносно невелика динамічна похибка моделі.

```
--------------------------------------------------
Статистичні характеристики вибірки
--------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 9.479533348764691e-10
Дисперсія = 4337.055626077996
Середнє квадратичне відхилення = 65.85632563450527
--------------------------------------------------
Статистичні характеристики моделі
--------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 7.206662779762632e-10
Дисперсія = 2992.012882312153
Середнє квадратичне відхилення = 54.699295080578075
Динамічна похибка моделі = 40.082313314344816
```

Рисунок 12 – Статистичні характеристики моделі

Оцінимо якість створеної моделі та порівняємо графіки. Для оцінки використаємо коефіцієнт детермінації R^2.

Лістинг коду:

```
[...]

'''Блок оцінювання якості моделі'''

# Функція для обчислення коефіцієнта детермінації R^2
def r2_score(actual, predicted):
    # Обчислення середнього значення реальних даних
    mean_actual = np.mean(actual)
    # Сума квадратів відхилень реальних даних від їх середнього
    sst = np.sum((actual - mean_actual) ** 2)
    # Сума квадратів помилок (відхилень) моделі
    ssr = np.sum((actual - predicted) ** 2)
    # Обчислення коефіцієнта детермінації R^2
    r2 = 1 - (ssr / sst)
    print('-----------------------------------------------------')
    print('Якість моделі')
    print('-----------------------------------------------------')
    print('Кількість елементів вибірки =', len(predicted))
    print('Коефіцієнт детермінації (ймовірність апроксимації) =', r2)
    return r2

'''Основний блок'''

    [...]
if data_mode in range(1, len(names) + 1):
    [...]
    # Оцінювання якості моделі
    r2_score(df_real_sorted, model_shum)
    plot(model_shum, df_real_sorted, 'Відповідність моделі реальним даним', 'Модель',
names[data_mode-1], '(i+1)-е вересня 2023', 'Штуки')
```

Результат:

Коефіцієнт детермінації свідчить про 99% відповідність моделі реальним даним.

```
-----------------------------------------------------
Якість моделі
-----------------------------------------------------
Кількість елементів вибірки = 21
Коефіцієнт детермінації (ймовірність апроксимації) = 0.9900093635200369
```

Рисунок 13 – Верифікація моделі

Порівняємо графіки вибірок і побачимо, що вони близькі за значеннями, але все ж мають певну розбіжність.

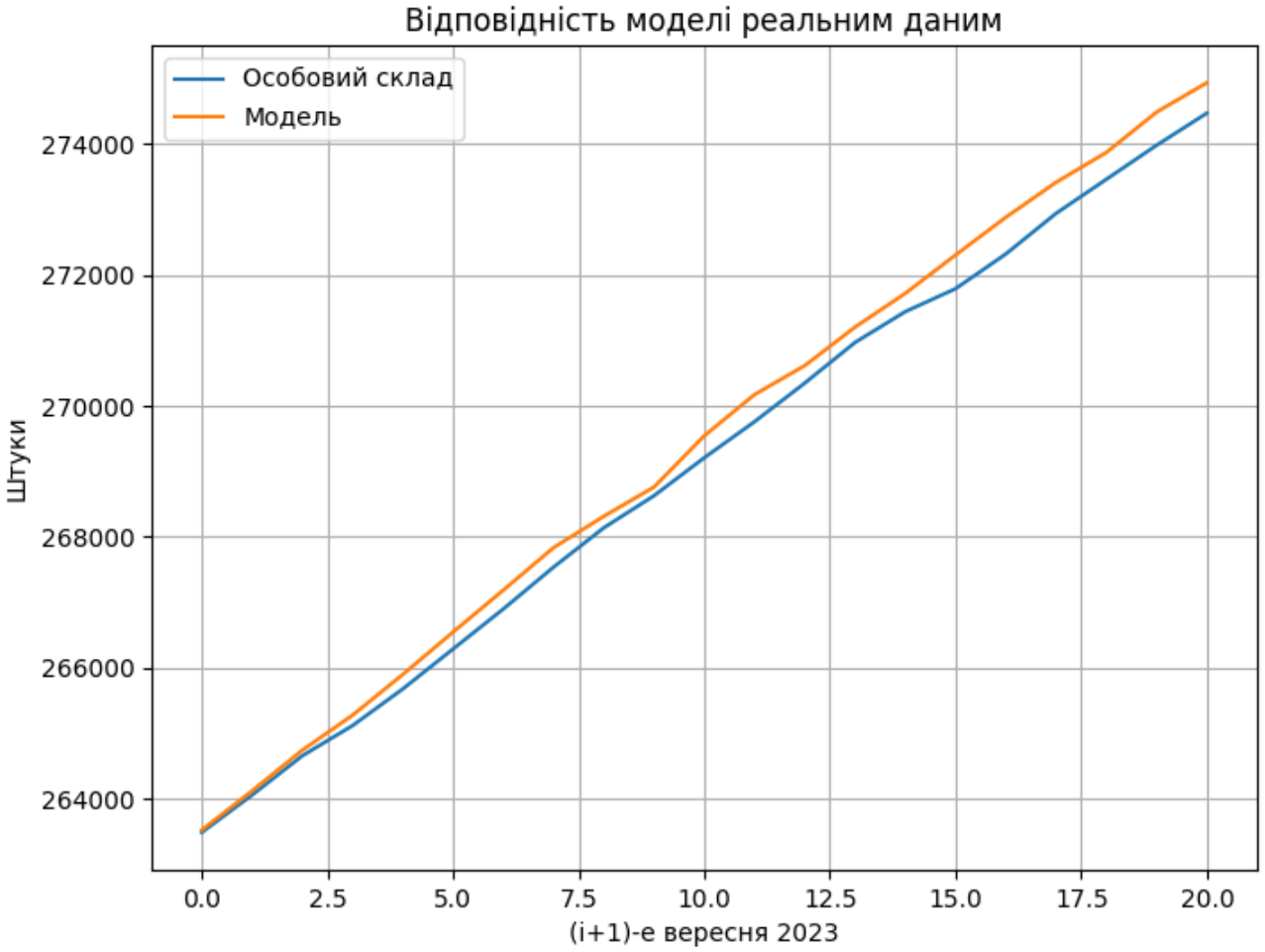

Рисунок 14 – Відповідність моделі реальним даним

2.6. Провести аналіз отриманих результатів

За вхідну вибірку ми взяли *реальні дані*.

Визначені характеристики вхідної вибірки – підтверджено графіком:

Часова надмірність даних з лінійним розподілом.

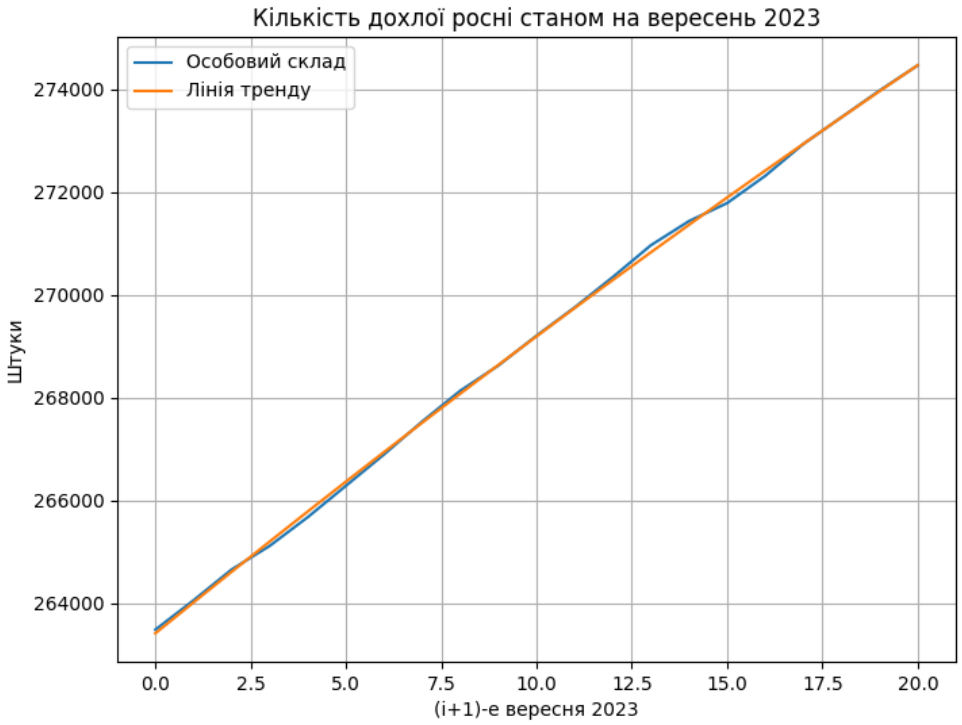

Рисунок 15 – Графік, що підтверджує часову надмірність та лінійність розподілу даних

Статистичні характеристики вхідної вибірки – підтверджено виводом:

Кількість реалізацій випадкової величини (*об'єм вибірки*) = 21.

Середнє значення розподілу = 9.479533348764691e-10.

Середньо квадратичне відхилення розподілу = 65.85632563450527.

```
----------------------------------------------------
Статистичні характеристики вибірки
----------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 9.479533348764691e-10
Дисперсія = 4337.055626077996
Середнє квадратичне відхилення = 65.85632563450527
```

Рисунок 16 – Вивід, що підтверджує статистичні характеристики вибірки

Регресійна модель вхідної вибірки – підтверджено виводом:

y(t) = *263418.70129870024* + *602.9829118250829* * t + *-2.5153793574886176* * t^2

```
-----------------------------------------------------
Регресійна модель
-----------------------------------------------------
y(t) =  263418.70129870024  +  602.9829118250829  * t  +  -2.5153793574886176  * t^2
```

Рисунок 17 – Вивід, що підтверджує параметри регресійної моделі

Синтезована штучна модель, аналогічна до вхідної вибірки за трендом – підтверджено графіком:

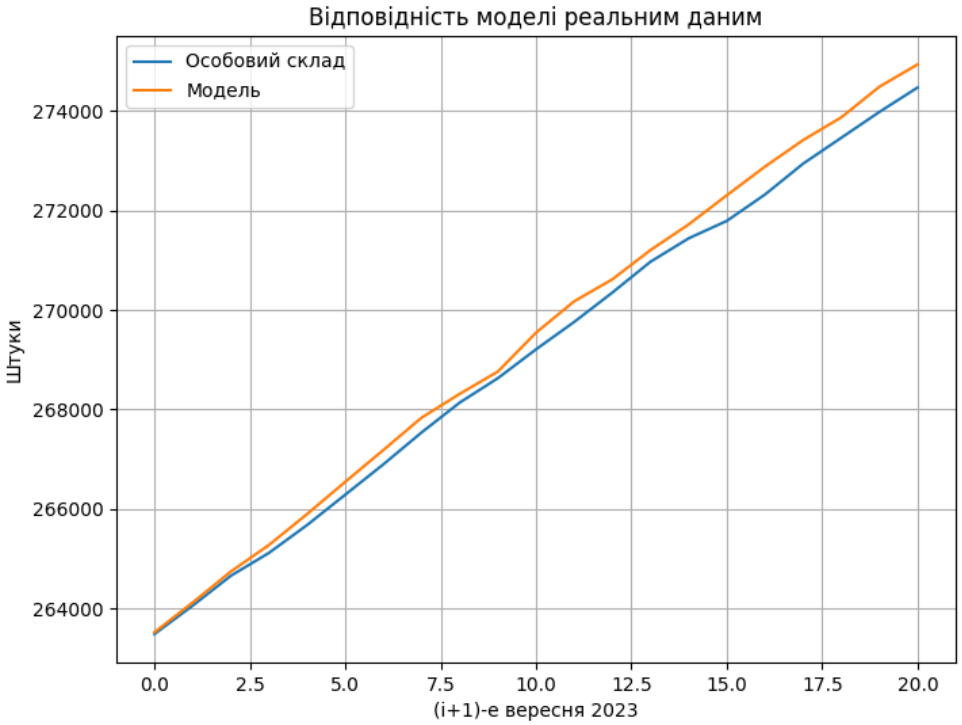

Рисунок 17 – Графік, що підтверджує відповідність тренду моделі до вибірки

Статистичні характеристики синтезованої моделі, наближені до вхідної вибірки – підтверджено виводом:

Кількість реалізацій випадкової величини (*об'єм моделі*) = 21.

Середнє значення моделі = 7.206662779762632e-10.

Середньо квадратичне відхилення моделі = 54.699295080578075.

Динамічна похибка моделі = 40.082313314344816.

```
--------------------------------------------------
Статистичні характеристики моделі
--------------------------------------------------
Кількість елементів вибірки = 21
Матиматичне сподівання = 7.206662779762632e-10
Дисперсія = 2992.012882312153
Середнє квадратичне відхилення = 54.699295080578075
Динамічна похибка моделі = 40.082313314344816
```

Рисунок 16 – Вивід, що підтверджує статистичні характеристики моделі

Синтезована модель верифікована – підтверджено виводом:

Коефіцієнт детермінації (ймовірність апроксимації) = 0.9900093635200369.

```
--------------------------------------------------
Якість моделі
--------------------------------------------------
Кількість елементів вибірки = 21
Коефіцієнт детермінації (ймовірність апроксимації) = 0.9900093635200369
```

Рисунок 17 – Вивід, що підтверджує верифікацію моделі

Висновок:

Методи статистичного навчання, такі як регресія та аналіз часових рядів, можуть бути використані для визначення статистичних характеристик вхідного потоку даних в Python.

Регресійний аналіз може бути використаний для визначення залежності між змінною (наприклад, кількість втрат росії), та однією або декількома іншими змінними (наприклад, час).

Методи аналізу часових рядів можна використовувати для визначення статистичних характеристик, таких як середнє значення та середньо квадратичне відхилення.

Відповідність обрахованих числових характеристик статистичної вибірки та синтезованої моделі доводять адекватність розрахунків та правильність виконання поставленого завдання.

Розроблені програми можна використовувати для парсингу сайтів, визначення статистичних характеристик реальних даних та синтезації моделей, аналогічних за трендом та статистичним характеристикам реальним даним.