**Міністерство освіти і науки України**

**Національний технічний університет України «КПІ» імені Ігоря Сікорського**

**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**

**з лабораторної роботи №2**

**з навчальної дисципліни «Технології Data Science»**

**Тема:**

**СТАТИСТИЧНЕ НАВЧАННЯ З ПОЛІНОМІАЛЬНОЮ РЕГРЕСІЄЮ**

**Виконав:**

Студент 4 курсу кафедри ІПІ ФІОТ,

Навчальної групи ІП-11

Трикош І. В.

**Перевірив:**

Професор кафедри ОТ ФІОТ

Писарчук О.О.

**Київ 2024**

## **І. Мета:**

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.

## **ІІ. Завдання:**

**Група вимог\_1:**

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 (вибір моделі залежно від значення показника якості). Показник якості та спосіб оптимізації обрати самостійно.

5. Статистичне навчання поліноміальної моделі за методом найменших квадратів (МНК – LSM) – поліноміальна регресія для вхідних даних, отриманих в п.1,2. Спосіб реалізації МНК обрати самостійно;

6. Прогнозування (екстраполяцію) параметрів досліджуваного процесу за «навченою» у п.5 моделлю на 0,5 інтервалу спостереження (об’єму вибірки);

7. Провести аналіз отриманих результатів та верифікацію розробленого скрипта.

**Група Вимог\_2:**

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.

**Завдання ІІ рівня – максимально 9 балів.**

Реалізувати групу вимог 1 та 2.

Дані індексу S&P 500: <https://www.nasdaq.com/market-activity/index/spx/historical?page=1&rows_per_page=1000&timeline=y1>

## **ІІІ. Результати виконання лабораторної роботи.**

* 1. **Синтезована математична модель**

Відповідно до умов задачі синтезовано математичну модель, що очищає вхідні дані від аномальних вимірів, визначає показники якості та оптимізує модель, виконує статистичне навчання поліноміальної моделі за МНК, прогнозує значення на 0.5 інтервалу спостереження, і виконує рекурентне згладжування alpha-beta фільтром.

Виявлення та видалення аномальних вимірів відбувається за алгоритмом ковзного вікна. Цей алгоритм полягає у реалізації згладжування тривіальним ковзним вікном. У цьому вікні виконується розрахунок статистичного середнього. Розмір вікна визначається квазілінійністю тренду. Суть алгоритму:

1. На вході є статистична вибірка .
2. Формується ковзне вікно розміром вимірів.
3. Визначається оцінка статистичного середнього:
4. Формується очищена вибірка шляхом заміни елементів вибірки оцінками статистичного середнього , починаючи з останнього виміру, включеного до ковзного вікна.
5. Зміщення ковзного вікна на наступний вимір вибір праворуч на .
6. Повторення п. 2-5 в межах вхідної статистичної вибірки .

Показником якості моделі було обрано . Це показник достовірності апроксимації. Він змінюється в межах від 0 до 1. Чим ближче до 1, тим краща модель. обчислюється за формулою:

де – об’єм вибірки, , – вимірюване значення, – оцінка виміряного значення.

Оптимізація моделі відбувається за допомогою зміни параметрів розподілу шуму, а саме дисперсії.

Метод найменших квадратів - метод знаходження наближеного розв'язку надлишково-визначеної системи. Мета полягає в налаштуванні параметрів функції моделі, щоб найкраще відповідати набору даних. Для знаходження коефіцієнтів поліноміальної регресії першого порядку метод працює так:

Нехай є пар точок , , де – незалежна змінна, а – залежна змінна. Потрібно знайти модельну функцію . І ця функція повинна якнайкраще відповідати даним. Ця відповідність вимірюється її залишком, який є різницею між спостережуваним значенням та значенням, передбаченим моделлю:

Метод найменших квадратів знаходить оптимальні значення параметрів шляхом мінімізації суми квадратів залишків:

На виході матимемо значення параметрів та .

Рекурентне згладжування відбувається за допомогою alpha-beta фільтра. Це лінійний фільтр, і при обробці нелінійних процесів з’являється розбіжність фільтра. Критерієм оптимальності фільтра є зменшення дисперсії помилки відтворення згладжених параметрів. Основні формули:

* 1. **Результати архітектурного проектування та їх опис**

На рис. 1 зображена блок-схема до моделі до програмного скрипту:

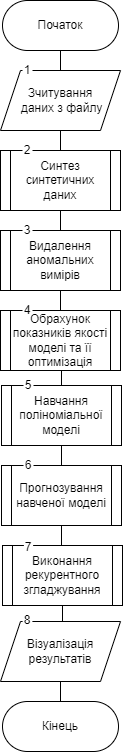


Рисунок 1 – Блок схема алгоритму

Робота алгоритму розпочинається зі зчитування даних з файлу у датафрейм. Ці дії виконуються у блоці 1 блок-схеми алгоритму рис.1.

У блоці 2 виконується синтез синтетичних даних. Сюди входить побудова лінії тренду, додавання до неї шуму та аномалій.

У блоці 3 видаляються аномальні виміри з допомогою алгоритму ковзного вікна.

У блоці 4 виконується обрахунок та зміна параметрів розподілу шуму для збільшення цього показника.

У блоці 5 відбувається навчання поліноміальної моделі за методом найменших квадратів.

У блоці 6 виконується прогнозування ціни індексу на півроку за навченою у пункті 5 моделлю.

У блоці 7 відбувається рекурентне згладжування alpha-beta фільтром.

У блоці 8 відбувається вивід результатів. На цьому виконання алгоритму завершене.

* 1. **Опис структури проекту програми**

Для реалізації розробленого алгоритму мовою програмування Python з використанням можливостей інтегрованого середовища PyCharm сформовано проєкт.

Проєкт базується на лінійній бізнес-логіці функціонального програмування та має таку структуру.

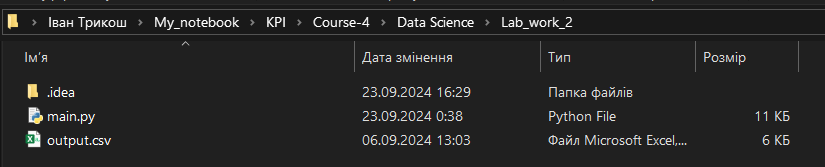


Рисунок 2 – Структура проєкту

Lab\_work\_2 – головний каталог проєкту

main.py – файл програмного коду алгоритму

output.csv – річні дані про індекс S&P 500

* 1. **Результати роботи програми відповідно до завдання (допускається у формі скриншотів)**

Результатом роботи програми є сукупність послідовності графічних вікон, що реалізують умови завдання лабораторної роботи.

1. На рисунках 3-5 зображено реальні та синтетичні дані та їх показники:

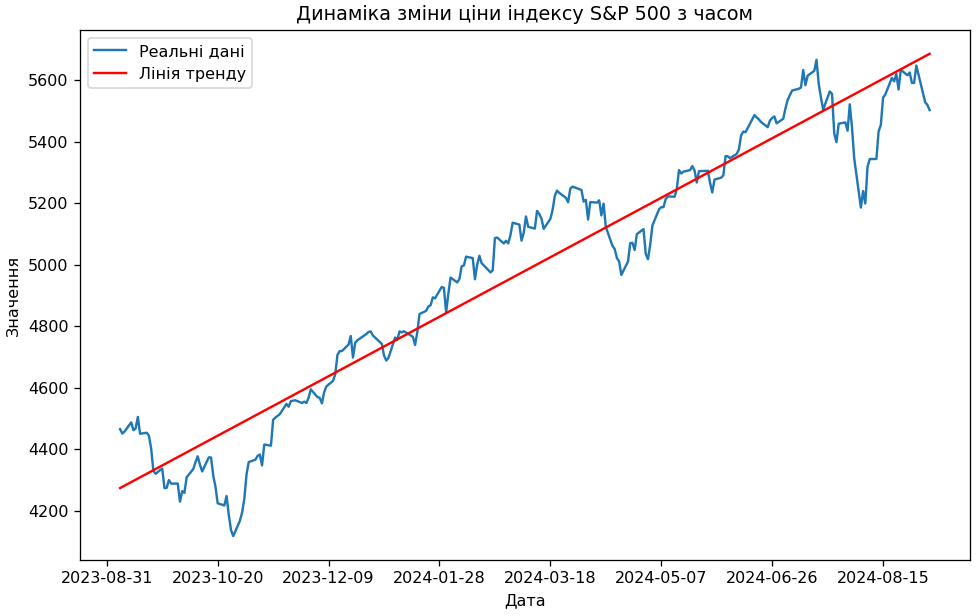


Рисунок 3 – Реальні дані

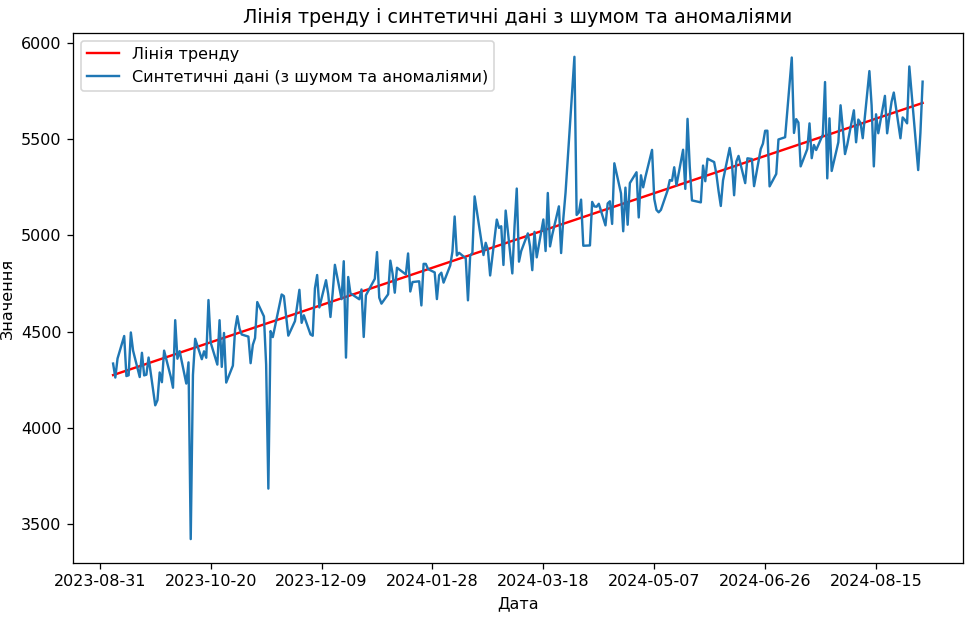


Рисунок 4 – Синтетичні дані

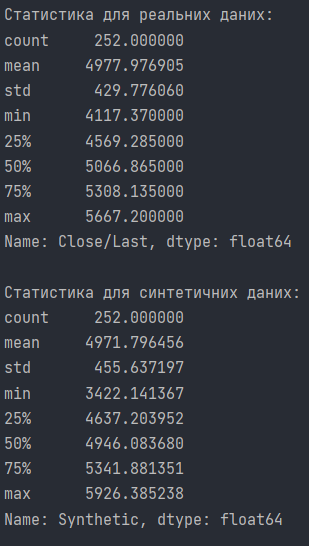


Рисунок 5 – Статистичні показники реальних та синтетичних даних

Бачимо, що статистичні показники доволі схожі, тому модель синтезовано коректно.

1. На рисунку 6 зображено реальний та очищені синтетичні дані:

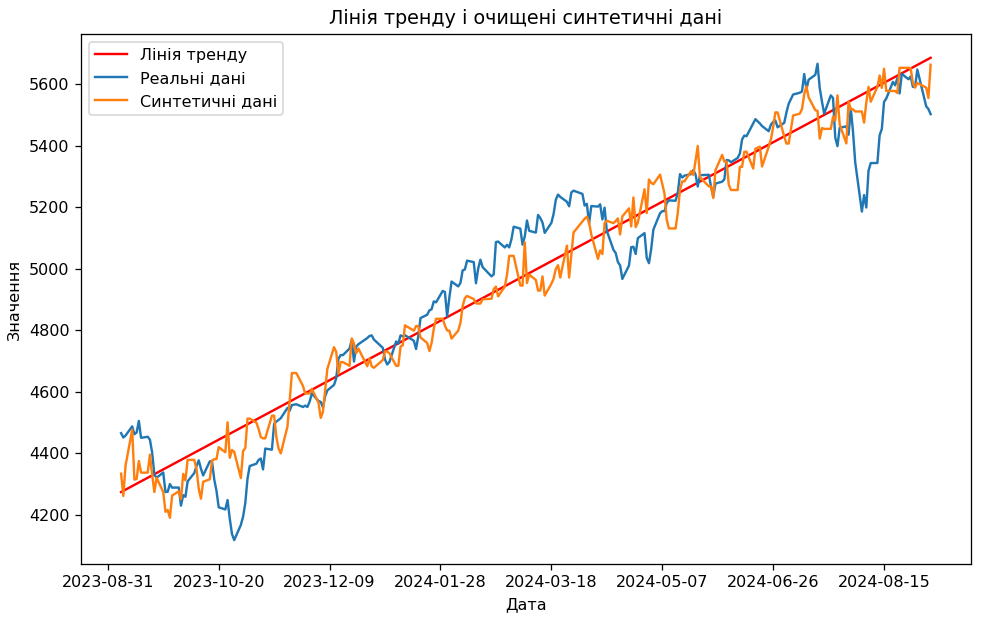


Рисунок 6 – Реальні і очищені синтетичні дані

Бачимо, що тепер аномалій немає.

1. На рисунках 7-8 зображено показники моделі при різних параметрах розподілу шуму:



Рисунок 7 – Показник якості моделі для коефіцієнту дисперсії 1



Рисунок 8 – Показник якості моделі для коефіцієнту дисперсії 0.75

При меншій дисперсії шуму якість моделі дещо зросла.

1. На рисунку 9 зображено навчену поліноміальну регресію третього порядку:

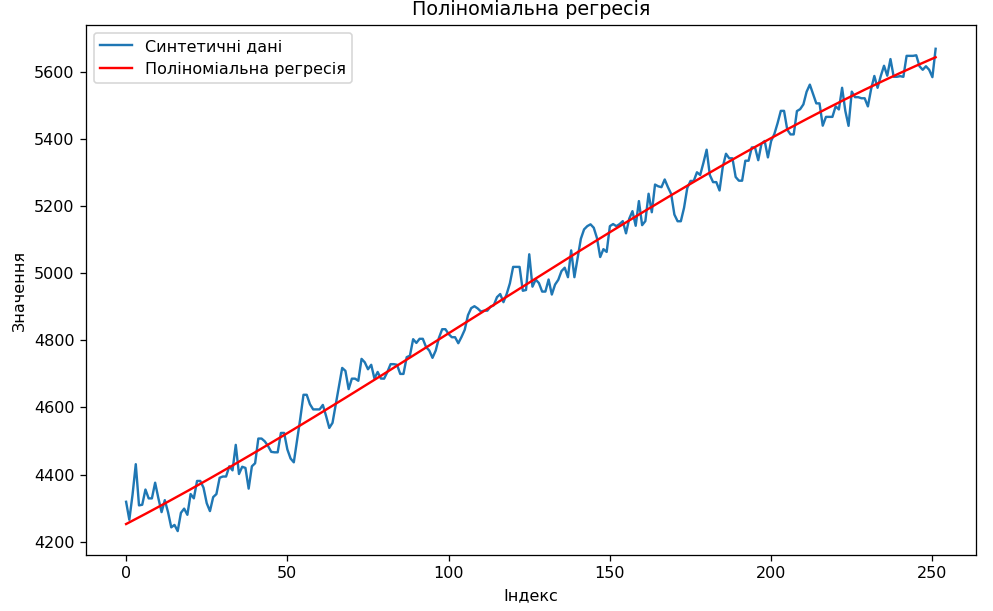


Рисунок 9 – Навчена поліноміальна модель

Навчена модель доволі близька до синтезованої.

1. На рисунку 10 зображено прогноз навченої моделі на півроку:

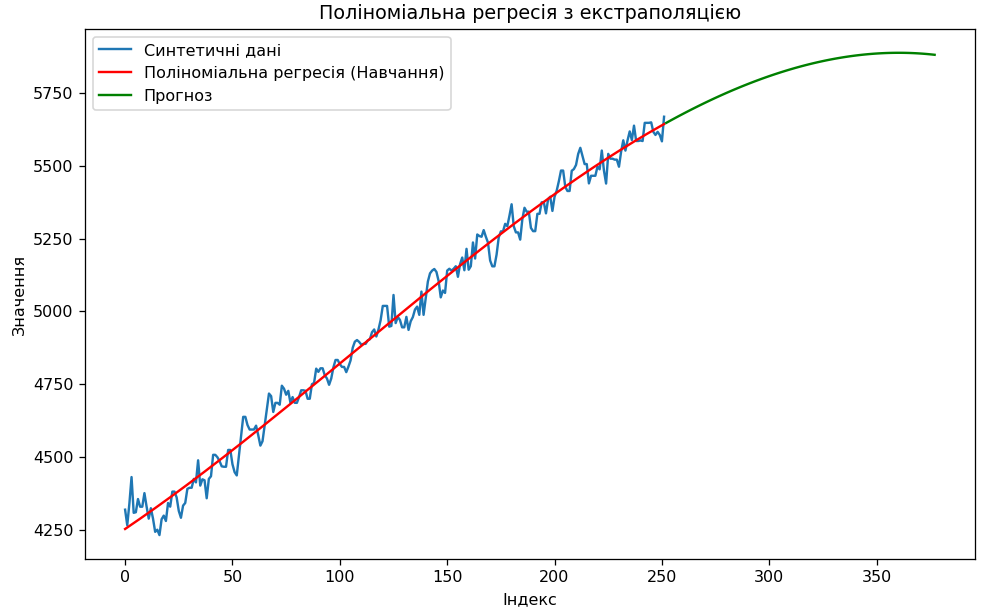


Рисунок 10 – Прогноз навченої моделі

1. На рисунку 11 зображено рекурентне згладжування alpha-beta фільтрами:

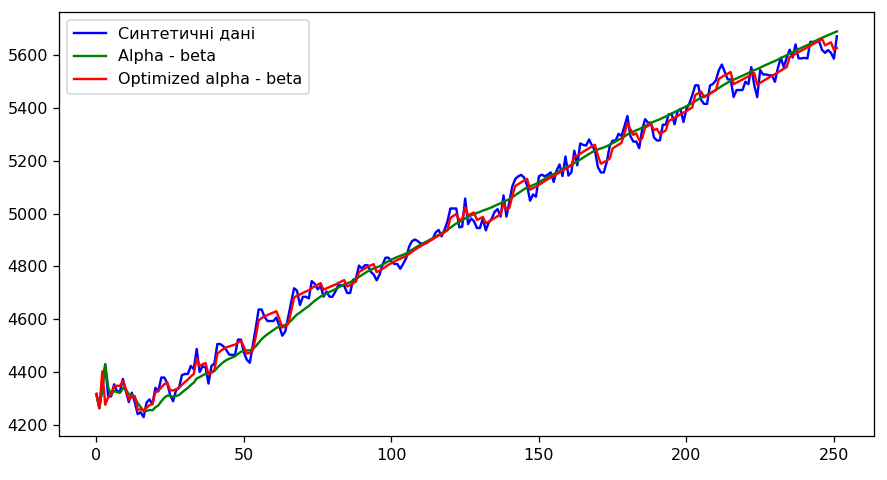


Рисунок 11 – Рекурентне згладжування alpha-beta фільтрами

Бачимо, що фільтр з оптимізацією спрацював краще. Оптимізація – якщо похибка між прогнозованим та реальним значенням перевищує встановлений поріг, то прогнозоване значення коригується із заданим коефіцієнтом корекції.

* 1. **Програмний код, що забезпечує отримання результату (допускається у формі скриншотів)**

Програмний код послідовно реалізує алгоритм на рис.1 та спрямований на отримання результатів, поданих на рис.3-11.

Для спрощення програмного коду і раціоналізації обчислень застосовано функціональні механізми створення підпрограм.

При цьому використано можливості Python бібліотек: numpy, math, sklearn, pandas, seaborn, matplotlib.

Контекстні коментарі пояснюють сутність окремих скриптів наведеного коду програми.

main.py

*"""  
 - Створення синтетичних даних про ціну індексу S&P 500 на біржі NASDAQ за рік на основі  
лінії тренду реальних даних, шуму та аномальних вимірів за нормальним розподілом.  
 - Очищення даних від аномальних вимірів з допомогою методу ковзного вікна.  
 - Визначення якості моделі з допомогою R² та оптимізація моделі за допомогою зміни параметрів розподілу шуму  
 - Статистичне навчання поліноміальної моделі за МНК та прогнозування ціни індексу на наступні пів року  
 - Рекурентне згладжування α - β фільтром та усунення "розбіжності" фільтра з допомогою коригування вихідного значення  
"""  
  
import* matplotlib.pyplot *as* plt  
*import* numpy *as* np  
*import* pandas *as* pd  
*import* seaborn *as* sns  
*import* math  
*from* sklearn.metrics *import* r2\_score  
  
  
*# Зчитування даних у датафрейм та початкове перетворення*df = pd.read\_csv('output.csv')  
df['Date'] = pd.to\_datetime(df['Date'])  
df = df.sort\_values(by='Date', ascending=*True*).reset\_index(drop=*True*)  
  
*# Переведемо дату в кількість днів для лінійної регресії*df['Days'] = (df['Date'] - df['Date'].min()).dt.days  
  
*# Обчислення коефіцієнтів лінійної регресії y = α + β \* x*coefficients = np.polyfit(df['Days'], df['Close/Last'], 1)  
beta = coefficients[0]  
alpha = coefficients[1]  
  
*# Обчислення значень для лінії тренду*df['Trend'] = beta \* df['Days'] + alpha  
  
*# Побудова графіка реальних даних та лінії тренду*plt.figure(figsize=(10, 6))  
sns.lineplot(x='Date', y='Close/Last', data=df, label='Реальні дані')  
sns.lineplot(x='Date', y='Trend', data=df, label='Лінія тренду', color='red')  
plt.gca().xaxis.set\_major\_locator(plt.MaxNLocator(nbins=10))  
plt.title('Динаміка зміни ціни індексу S&P 500 з часом')  
plt.xlabel('Дата')  
plt.ylabel('Значення')  
plt.show()  
  
*# Додаємо шум (нормальний розподіл з середнім 0 і стандартним відхиленням, близьким до реального)*np.random.seed(42)  
std\_dev = np.std(df['Close/Last'] - df['Trend']) \* 0.75  
noise = np.random.normal(loc=0, scale=std\_dev, size=len(df))  
  
*# Створюємо синтетичні дані, додаючи шум до тренду*df['Synthetic'] = df['Trend'] + noise  
  
*# Додаємо аномалії (2% від усіх даних)*num\_anomalies = int(df.shape[0] \* 0.02)  
anomaly\_magnitude = 5 \* std\_dev  
anomaly\_indices = np.random.choice(df.index, num\_anomalies, replace=*False*)  
df.loc[anomaly\_indices, 'Synthetic'] += np.random.normal(loc=0, scale=anomaly\_magnitude, size=num\_anomalies)  
  
*# Побудова графіка синтетичних даних і лінії тренду*plt.figure(figsize=(10, 6))  
sns.lineplot(x='Date', y='Trend', data=df, label='Лінія тренду', color='red')  
sns.lineplot(x='Date', y='Synthetic', data=df, label='Синтетичні дані (з шумом та аномаліями)')  
plt.gca().xaxis.set\_major\_locator(plt.MaxNLocator(nbins=10))  
plt.title('Лінія тренду і синтетичні дані з шумом та аномаліями')  
plt.xlabel('Дата')  
plt.ylabel('Значення')  
plt.legend()  
plt.show()  
  
*# Описова статистика для реальних даних*real\_statistics = df['Close/Last'].describe()  
print("Статистика для реальних даних:")  
print(real\_statistics)  
  
*# Описова статистика для синтетичних даних*synthetic\_statistics = df['Synthetic'].describe()  
print("\nСтатистика для синтетичних даних:")  
print(synthetic\_statistics)  
  
  
*def* remove\_anomalies(data\_with\_anomalies, window\_size):  
 *""" Видалення аномальних вимірів з допомогою алгоритму ковзного вікна """* number\_of\_data = len(data\_with\_anomalies)  
 iterations = math.ceil(number\_of\_data - window\_size) + 1  
 medians = np.zeros(number\_of\_data)  
 *for* i *in* range(iterations):  
 window = np.zeros(window\_size)  
 *for* j *in* range(window\_size):  
 window[j] = data\_with\_anomalies[i + j]  
 medians[i + window\_size - 1] = np.median(window)  
 cleared\_data = np.zeros(number\_of\_data)  
 *for* i *in* range(number\_of\_data):  
 cleared\_data[i] = medians[i]  
 *for* i *in* range(window\_size):  
 cleared\_data[i] = data\_with\_anomalies[i]  
 *return* cleared\_data  
  
  
*# Виявлення та очищення аномалій*df['Synthetic'] = remove\_anomalies(df['Synthetic'], 4)  
  
*# Побудова графіка синтетичних даних і лінії тренду*plt.figure(figsize=(10, 6))  
sns.lineplot(x='Date', y='Trend', data=df, label='Лінія тренду', color='red')  
sns.lineplot(x='Date', y='Close/Last', data=df, label='Реальні дані')  
sns.lineplot(x='Date', y='Synthetic', data=df, label='Синтетичні дані')  
plt.gca().xaxis.set\_major\_locator(plt.MaxNLocator(nbins=10))  
plt.title('Лінія тренду і очищені синтетичні дані')  
plt.xlabel('Дата')  
plt.ylabel('Значення')  
plt.legend()  
plt.show()  
  
*# Обчислення R²*r2 = r2\_score(df['Close/Last'], df['Synthetic'])  
print(f'\nКоефіцієнт детермінації R² для синтетичних даних = {r2}') *# 0.925 при 1 \* std; 0.928 при 0.75\*std;  
  
# --------------------------- Вимога 1.5 ---------------------------  
  
# Визначаємо дані для навчання моделі*X = np.arange(len(df)) *# Для спрощення візьмемо індекси замість дат*y = df['Synthetic']  
  
*# Вибираємо ступінь поліноміальної моделі*degree = 3  
*# Знаходимо коефіцієнти поліноміальної моделі*coeffs = np.polyfit(X, y, degree)  
*# Створюємо поліноміальну функцію на основі коефіцієнтів*poly\_model = np.poly1d(coeffs)  
*# Прогнозування для навчальних даних*y\_pred = poly\_model(X)  
  
*# Оцінка моделі за допомогою R²*r2 = r2\_score(y, y\_pred)  
print(f'\nКоефіцієнт детермінації R² для навченої поліноміальної моделі = {r2}')  
  
*# Побудова графіка результатів*plt.figure(figsize=(10, 6))  
sns.lineplot(x=X, y=df['Synthetic'], label='Синтетичні дані')  
plt.plot(X, y\_pred, label="Поліноміальна регресія", color='red')  
plt.title('Поліноміальна регресія')  
plt.xlabel('Індекс')  
plt.ylabel('Значення')  
plt.legend()  
plt.show()  
  
*# --------------------------- Вимога 1.6 ---------------------------  
  
# Прогноз на 0,5 інтервалу спостереження*n\_new = int(0.5 \* len(X)) *# Кількість нових точок для прогнозу*X\_future = np.arange(len(X), len(X) + n\_new) *# Новий інтервал для прогнозу  
  
# Прогноз на нові дані*y\_future\_pred = poly\_model(X\_future)  
  
*# Побудова графіка для прогнозу*plt.figure(figsize=(10, 6))  
sns.lineplot(x=X, y=df['Synthetic'], label='Синтетичні дані')  
plt.plot(X, y\_pred, label="Поліноміальна регресія (Навчання)", color='red')  
plt.plot(X\_future, y\_future\_pred, label="Прогноз", color='green')  
plt.title("Поліноміальна регресія з екстраполяцією")  
plt.xlabel("Індекс")  
plt.ylabel("Значення")  
plt.legend()  
plt.show()  
  
  
*# --------------------------- Вимога 2.5 ---------------------------  
  
def* alpha\_beta\_filter(data):  
 *""" α - β фільтр """* number\_of\_data = len(data)  
 in\_values = np.array(data, dtype=float).reshape(number\_of\_data, 1)  
 out\_values = np.zeros((number\_of\_data, 1))  
 interval = 1  
  
 speed = (in\_values[1, 0] - in\_values[0, 0]) / interval  
 extra = in\_values[0, 0] + speed  
 alpha = 2 \* (2 \* 1 - 1) / (1 \* (1 + 1))  
 beta = 6 / (1 \* (1 + 1))  
  
 out\_values[0, 0] = in\_values[0, 0]  
 *for* i *in* range(1, number\_of\_data):  
 out\_values[i, 0] = extra + alpha \* (in\_values[i, 0] - extra)  
 speed = speed + (beta / interval) \* (in\_values[i, 0] - extra)  
 extra = out\_values[i, 0] + speed  
  
 alpha = (2 \* (2 \* i - 1)) / (i \* (i + 1))  
 beta = 6 / (i \* (i + 1))  
 *return* out\_values  
  
  
*def* alpha\_beta\_filter\_with\_optimization(data):  
 *"""  
 α - β фільтр з оптимізацією для подолання явища розбіжності фільтра.  
 Оптимізація: якщо похибка між прогнозованим та реальним значенням перевищує встановлений поріг,  
 то прогнозоване значення коригується із заданим коефіцієнтом корекції.  
 """* number\_of\_data = len(data)  
 in\_values = np.array(data, dtype=float).reshape(number\_of\_data, 1)  
 out\_values = np.zeros((number\_of\_data, 1))  
 interval = 1  
 threshold = 50  
 correction\_factor = 0.5  
  
 speed = (in\_values[1, 0] - in\_values[0, 0]) / interval  
 extra = in\_values[0, 0] + speed  
 alpha = 2 \* (2 \* 1 - 1) / (1 \* (1 + 1))  
 beta = 6 / (1 \* (1 + 1))  
  
 out\_values[0, 0] = in\_values[0, 0]  
  
 *for* i *in* range(1, number\_of\_data):  
 error = in\_values[i, 0] - extra  
 *if* abs(error) > threshold:  
 extra += correction\_factor \* error  
  
 out\_values[i, 0] = extra + alpha \* error  
 speed = speed + (beta / interval) \* error  
 extra = out\_values[i, 0] + speed  
  
 alpha = (2 \* (2 \* i - 1)) / (i \* (i + 1))  
 beta = 6 / (i \* (i + 1))  
  
 *return* out\_values  
  
  
*# Застосування фільтра alpha-beta*abf = alpha\_beta\_filter(df['Synthetic'])  
abf\_optimized = alpha\_beta\_filter\_with\_optimization(df['Synthetic'])  
  
*# Візуалізація результатів*plt.plot(df['Synthetic'], label='Синтетичні дані', color='blue')  
plt.plot(abf, label='Alpha - beta', color='green')  
plt.plot(abf\_optimized, label='Optimized alpha - beta', color='red')  
plt.legend()  
plt.show()

## **IV. Висновки.**

У ході виконання лабораторної роботи було проведено дослідження особливостей реалізації процесів статистичного навчання із застосуванням методів обробки Big Data масивів та калмановської рекурентної фільтрації з використанням можливостей мови програмування Python. Було навчено поліноміальну модель та спрогнозовано значення індексу S&P 500 на півроку та виконано фільтрацію синтетичних даних.

Виконав: студент Трикош І. В.