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 «Вступ до технології Data Science»

Тема:

СТАТИСТИЧНЕ НАВЧАННЯ З ПОЛІНОМІАЛЬНОЮ РЕГРЕСІЄЮ

Виконав:
Студент 3 курсу кафедри ІПІ ФІОТ,
Навчальної групи ІП-11
Лошак В.І.

Перевірив:
Професор кафедри ОТ ФІОТ
Писарчук О.О.

Київ 2023

I. 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.

II. Завдання:

Реалізація проекту триває та спрямовано на збільшення функціональності програмної компоненти

Лабораторія провідної IT-компанії реалізує масштабний проект розробки універсальної платформи з обробки Big Data масиву статистичних даних поточного спостереження для виявлення закономірностей і прогнозування розвитку контрольованого процесу. Платформа передбачає розташування back-end компоненти на власному хмарному сервері з наданням повноважень користувачам заздалегідь адаптованого front-end функціоналу універсальної платформи.

III. Завдання IV рівня(максимум 15 балів):
Реалізувати групу вимог 1 та (або) 2 з імплементацією однієї з групи вимог 3.
Докладно опитати отримані R&D рішення

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 3:

3.1. Здійснити розробку власного алгоритму виявлення аномальних вимірів та / або «навчання» параметрів відомих алгоритмів «бачити» властивості статистичної вибірки.

IV. Результати виконання лабораторної роботи.

1. **Отримання вхідних даних із властивостями, заданими в Лр_1**

Для отримання даних попередньої лабораторної використано бібліотеку pandas. Загружені дані про тренд та залишки скомбіновано в один data frame. В лабораторній 1 ці дані вже було очищено від аномалій.

```python
trend_df = pd.read_csv('trend_model.csv')
residual_df = pd.read_csv('residuals_model.csv')

combined_df = pd.DataFrame()
combined_df['date'] = trend_df['timeOpen']
combined_df['price'] = trend_df['trend'] + residual_df['residuals']
combined_df.set_index('date', inplace=True)
fig, ax = plt.subplots(figsize = (12, 6))
combined_df.plot( ax=ax)
trend_df.plot(x='timeOpen', y='trend', ax=ax)

combined_df.reset_index( inplace=True)
```

[192] ✓ 0.5s Python

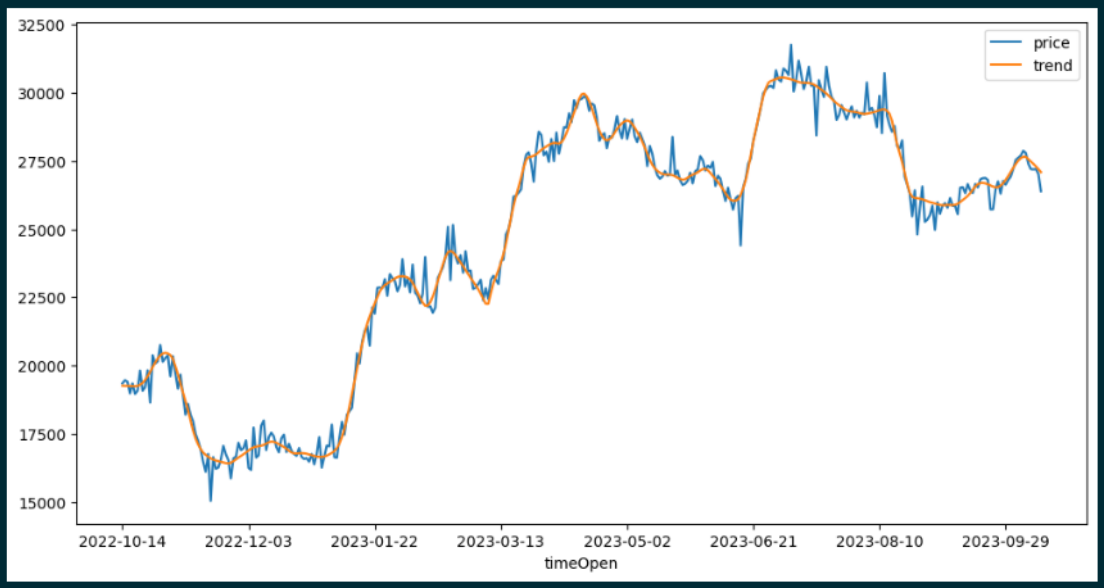


Рис. 1— Візуалізація тренду та комбінованої моделі даних.

2. Модель вхідних даних із аномальними вимірами;

Для того щоб додати аномалії до даних використано метод додавання кратної певному мультиплікатору кількості стандартних відхилень. Згенеровані аномалії розподілені рівноміро.

```python
threshold = 5
anomaly_percentage = 0.1 # 10% of total data
price_std = np.std(residual_df['residuals'])

total_data = len(combined_df)
num_anomalies = int(total_data * anomaly_percentage)
anomaly_indices = np.random.choice(np.arange(total_data), size=num_anomalies, replace=False)

combined_df_with_anomalies = combined_df.copy()
for index in anomaly_indices:
    sign = np.random.choice([-1, 1])
    combined_df_with_anomalies.loc[index, 'price'] += sign * threshold * price_std

combined_df_with_anomalies.plot()
combined_df.plot()
```

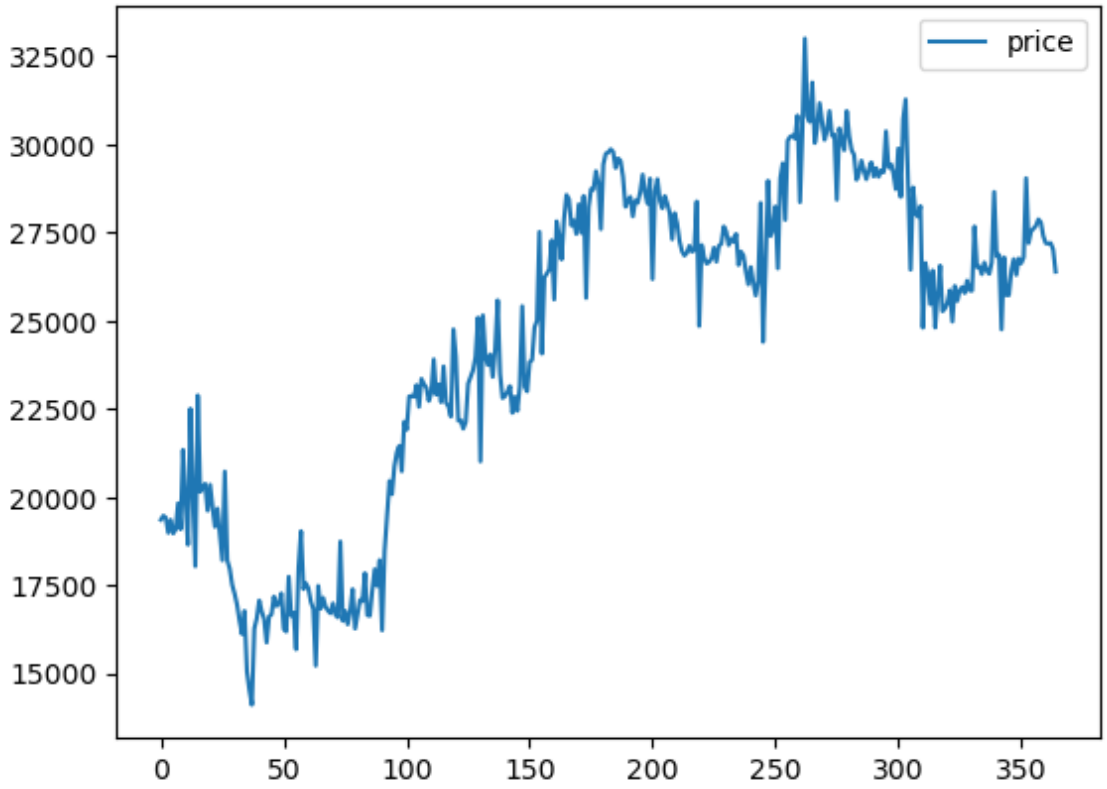


Рис. 2— Візуалізація часового ряду з доданими аномаліями що відхиляються на 5 стандартних відхилень.

3. **Очищення вхідних даних від аномальних вимірів. Спосіб виявлення аномалій та очищення обрати самостійно;**

3.1. Здійснити розробку власного алгоритму виявлення аномальних вимірів та / або «навчання» параметрів відомих алгоритмів «бачити» властивості статистичної вибірки.

Для розробки власного алгоритму для виявлення аномалій було вирішено модифікувати алгоритм Sliding_Window_AV_Detect_medium наведений Писарчук О.О. в файлі L_1_3_Statistical_learnin. В ході модифікації було створено алгоритм для підбору оптимальних параметрів для заданого часового ряду, а також модифіковано спосіб збільшення вікна. Оновлений алгоритм medium має вигляд:

Крок 1: Створити вікно даних з заданою параметром розмірністю

Крок 2: Визначити статистичні характеристики поточного вікна

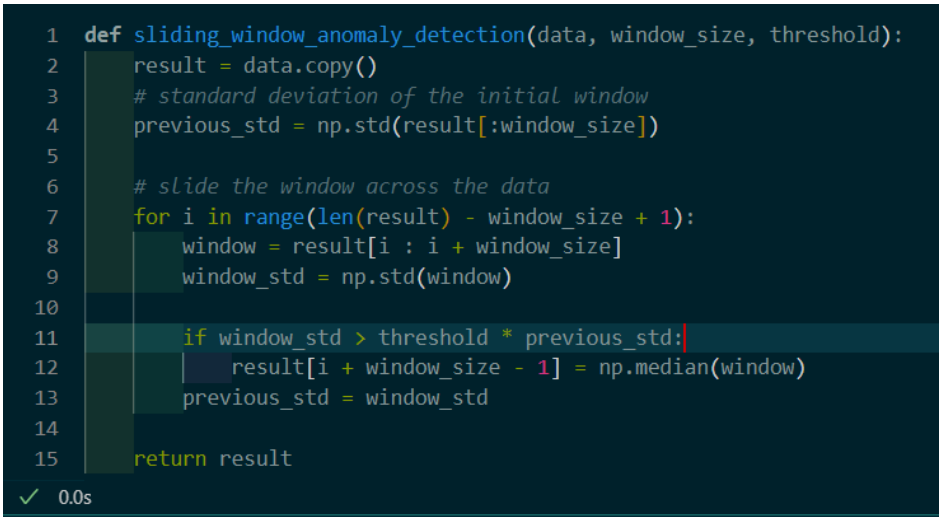
Крок 3: Зсунути вікно вздовж інтервалу на одну поділку

Крок 4: Визначити статистичні характеристики нового вікна

Крок 5: Порівняти стат. Характеристики нового вікна з даними попереднього вікна і замінити включене до вікна значення на медіанне значення по поточному вікну.

Крок 6:встановити характеристики поточного вікна замість минулих характеристик

Крок 7: Якщо не досягнуто кінця даних — повернутися на крок 3.

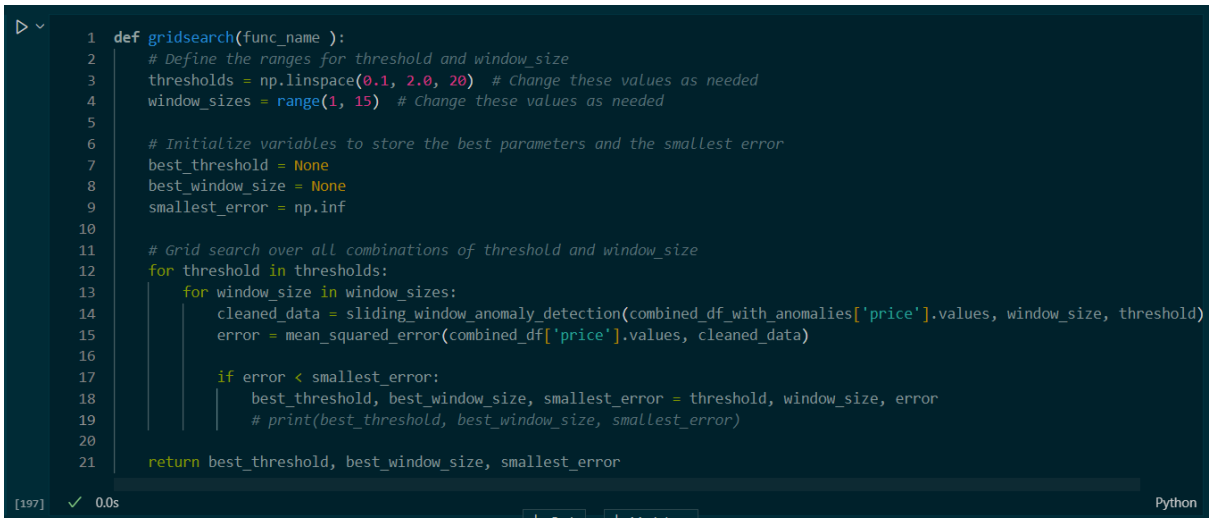


```python
def sliding_window_anomaly_detection(data, window_size, threshold):
    result = data.copy()
    # standard deviation of the initial window
    previous_std = np.std(result[:window_size])

    # slide the window across the data
    for i in range(len(result) - window_size + 1):
        window = result[i : i + window_size]
        window_std = np.std(window)

        if window_std > threshold * previous_std:
            result[i + window_size - 1] = np.median(window)
        previous_std = window_std

    return result
```

0.0s

Рис. 3— Оновлений алгоритм детекції аномалій.

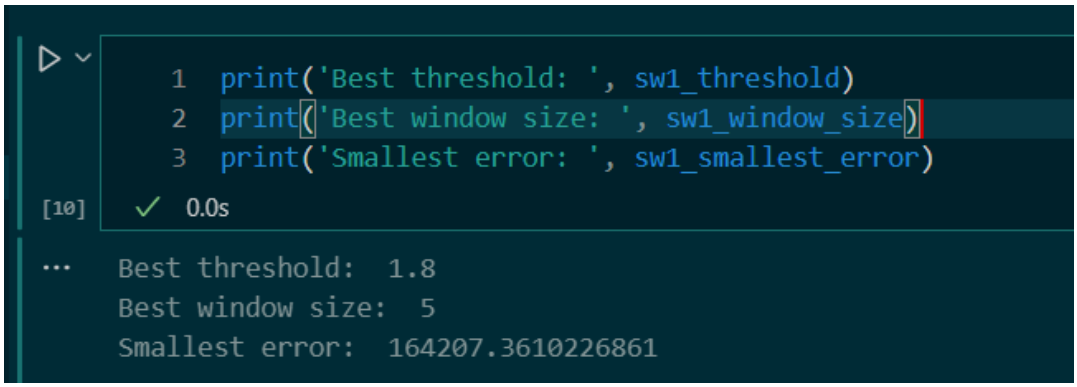
Для підбору параметрів моделі було розроблено алгоритм що проходиться через усі значення розміру вікна та трешхолду в певному діапазоні значень, та для кожної комбінації оцінює якість очистки використовуючи MSE.



```python
def gridsearch(func_name ):
    # Define the ranges for threshold and window_size
    thresholds = np.linspace(0.1, 2.0, 20)  # Change these values as needed
    window_sizes = range(1, 15)  # Change these values as needed

    # Initialize variables to store the best parameters and the smallest error
    best_threshold = None
    best_window_size = None
    smallest_error = np.inf

    # Grid search over all combinations of threshold and window_size
    for threshold in thresholds:
        for window_size in window_sizes:
            cleaned_data = sliding_window_anomaly_detection(combined_df_with_anomalies['price'].values, window_size, threshold)
            error = mean_squared_error(combined_df['price'].values, cleaned_data)

            if error < smallest_error:
                best_threshold, best_window_size, smallest_error = threshold, window_size, error
                # print(best_threshold, best_window_size, smallest_error)

    return best_threshold, best_window_size, smallest_error
```

[197] ✓ 0.0s Python

Рис. 4— Алгоритм підбору параметрів для очищення вибірки від аномалій способом sliding_window_medium.



```python
print('Best threshold: ', sw1_threshold)
print('Best window size: ', sw1_window_size)
print('Smallest error: ', sw1_smallest_error)
```

[10] ✓ 0.0s

```
Best threshold:  1.8
Best window size:  5
Smallest error:  164207.3610226861
```

Рис. 5— Значення підібраних параметрів алгоритму sliding_window_medium

```
sw1_threshold, sw1_window_size, sw1_smallest_error = gridsearch
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 10))
sw1_cleaned_data = sliding_window_anomaly_detection(combined_df_with_anomalies['price']
ax1.plot(sw1_cleaned_data, label='Cleaned data')
ax1.set_xlabel('Day of the year')
ax1.set_ylabel('Price')
ax2.plot( combined_df['price'], label='Original data')
# sw2_threshold, sw2_window_size, sw2_smallest_error = gridsearch(Sliding_Window_AV_Det
# sw2_cleaned_data = Sliding_Window_AV_Detect_medium(combined_df_with_anomalies['price'
# ax2.plot(sw2_cleaned_data)
```

[220] ✓ 1.0s Python

```
[<matplotlib.lines.Line2D at 0x1c8681769d0>]
```

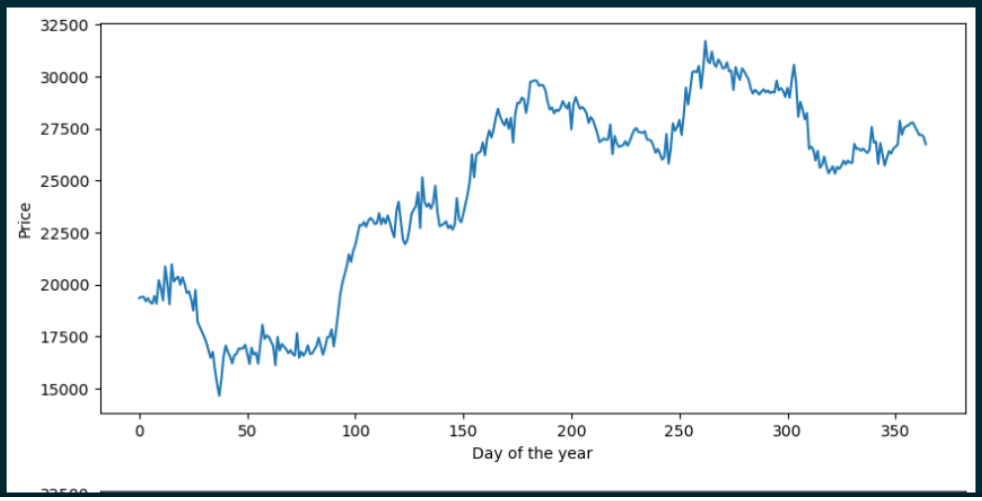


Рис. 6—Ряд очищений від аномалій з використанням оптимальних параметрів очищення.

4. **Визначення показників якості та оптимізація моделі (вибір моделі залежно від значення показника якості). Показник якості та спосіб оптимізації обрати самостійно.**

Для створення прогнозу було обрано побудувати поліноміальну модель. Для оцінки якості моделі вибрано показник R2. Для підбору оптимального значення параметру поліному використано алгоритм пошуку що використовує R2. Пошук параметра здійснено в діапазоні 1—10.

```
(9, 1541856.8568168175)
```

```
from sklearn.metrics import r2_score

# Fit the polynomial
coefficients = np.polyfit(range(len(sw1_cleaned_data)), sw1_cleaned_data, 9)
polynomial = np.poly1d(coefficients)

# Calculate the R2 score
r2 = r2_score(sw1_cleaned_data, polynomial(range(len(sw1_cleaned_data))))
print('R2 score: ', r2)
```

[12] ✓ 0.0s

```
0.926731255402576
```

Рис. 7— Метрика R2 для оптимальної поліноміальної моделі.

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;**

```python
degrees = range(1, 10)  # Change the range of degrees as needed

best_degree = None
smallest_error = np.inf

for degree in degrees:
    # Fit the polynomial
    coefficients = np.polyfit(range(len(sw1_cleaned_data)), sw1_cleaned_data, degree)
    polynomial = np.poly1d(coefficients)

    # Calculate the mean squared error
    error = mean_squared_error(sw1_cleaned_data, polynomial(range(len(sw1_cleaned_data))))

    # Update the best degree and smallest error if necessary
    if error < smallest_error:
        best_degree = degree
        smallest_error = error

# Plot the polynomial fit
plt.plot(sw1_cleaned_data)
plt.plot(polynomial(range(len(sw1_cleaned_data))), label=f'Degree {best_degree} Polynomial Fit')
plt.legend()
plt.show()
```

Рис. 8— Алгоритм підбору параметрів поліному.

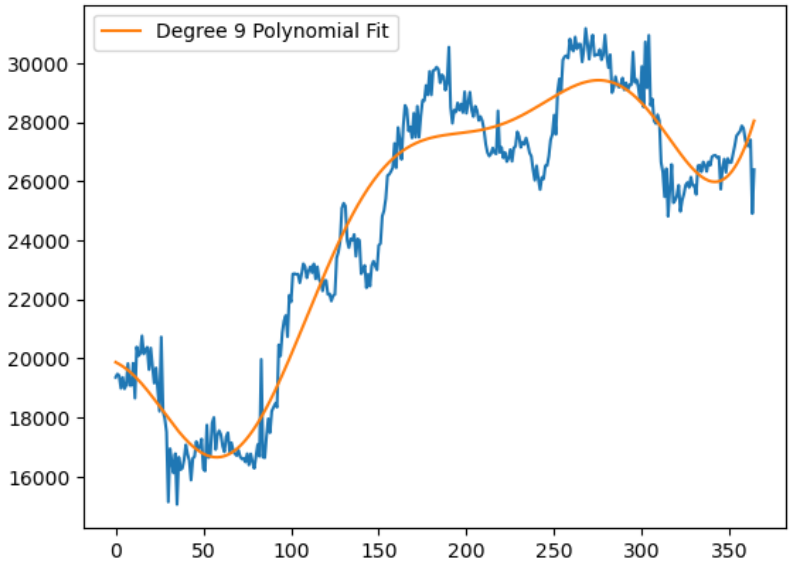


Рис. 9— Поліноміальна модель натренована за МНК після проведення підбору параметрів

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

З вище наведених метрик зроблено висновок що модель достатньо добре описує дані одже ми можемо використати її для прогнозування.

```
predictions_long = np.polyval(coefficients, range(round(len(sw1_cleaned_data)*1.5)))
predictions_short = np.polyval(coefficients, range(round(len(sw1_cleaned_data)+50)))
fig, ax = plt.subplots(1, 2, figsize=(10, 5))
ax[0].plot(sw1_cleaned_data)
ax[0].plot(predictions_long, label='Long-term predictions')
ax[1].plot(predictions_short, label='Short-term predictions')
ax[1].plot(sw1_cleaned_data)
plt.show()

✓ 0.6s
```

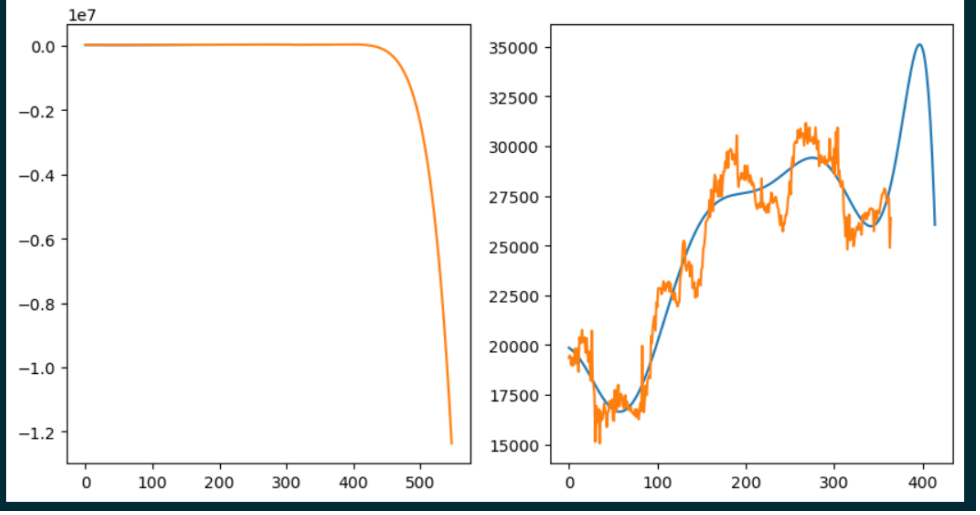


Рис. 10— Прогноз з використанням поліноміальної моделі на 1.5 та +10 відповідно до розміру вибірки.

7. **Провести аналіз отриманих результатів та верифікацію розробленого скрипта.**

З графіків видно що в короткостроковій перспективі модель прогнозує різкий стрибок вгору ціни Bitcoin з подальшим його стрімким падінням, що приводить нас до висновку що така волатильність може бути використана для спекуляцій на ринку. В той же час різке падіння відображає факт того що поліноміальна модель містить в собі певні надоліки в плані прогнозування оскільки не враховує історичні дані про діапазон в якому ціни біткоїн зазвичай перебувають що і приводить до появи від’ємних значень ціни і безпрецедентного прогнозу падіння. Незважаючи на вище сказане, в короткостроковій перспективі модель показує себе задовільно.

IV. Висновки.

У ході цієї роботи, я практикував побудову поліноміальної моделі для прогнозування часових рядів. Мною було створено/модифіковано алгоритм sliding_window_medium очищення даних від аномальних вимірів що адаптується до наданих даних за своїми параметрами і якісно відтворює оригінальні дані.

Виконав: студент ФІОТ Лошак В.І. ІП-11