МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

# Практикум №6

з курсу «Програмування інтелектуальних інформаційних систем»

на тему: «Аналіз часових послідовностей»

Викладач:
Курченко О.А.

Виконав:
Хільчук А.В.
студент 3 курсу
групи ІП-14 ФІОТ

Київ-2023

Практична робота №6

**Тема:** Аналіз часових послідовностей
**Завдання:**
1. Підготувати навчальну вибірку у вигляді таблиць MS Excel і зберегти їх як персональні файли. Для підготовки даних використовувати тематичні сайти Інтернет, результати проходження практик, довідники і каталоги.
2.Відобразити часові ряди даних (наприклад по ціні, мінімальній та максимальній ціні за день).
3. Чи є сезонні коливання? Відобразити графіки, виключивши сезонні коливання, якщо вони є.
4. Спрогнозувати котировку на тиждень вперед (використовуючи регресію та модель ARIMA). Яка якість прогнозу?
    1. Визначити коефіцієнт кореляції кожного набору
    2. Побудувати графіки розсіювання
    3. Побудувати найбільш підходящу функцію регресії par2=f(par1)
5. Зробити висновки.
Предметна бласть – кількість вакансій розробника ПЗ в Україні

**Виконання:**
1. Відповідно до заданого варіанту підготувати необхідні дані у вигляді таблиць MS Excel і зберегти їх як персональні файли. Для підготовки даних використовувати тематичні сайти Інтернет, результати проходження практик, довідники і каталоги.

   Для виконання даної лабораторної роботи було взято такий набір даних: https://github.com/uadata/uadata/blob/main/rozrobka-softu.csv
   Він щоденно фіксує кількість вакансій розробників ПЗ на українських порталах пошуку роботи, а також зміну даного показника.

2. Відобразити часові ряди даних (наприклад по ціні, мінімальній та максимальній ціні за день).
   Код програми наведено в **додатку А.**

   Зчитуємо, видаляємо стовпець зміни, адже він нерелевантний у даному контексті, та перейменовуємо стовпці:

| date | vacancies |
|---|---|
| 2022-05-20 | 15916 |
| 2022-05-21 | 15633 |
| 2022-05-22 | 15218 |
| 2022-05-23 | 15370 |
| 2022-05-24 | 15876 |
| ... | ... |
| 2023-11-03 | 7306 |
| 2023-11-04 | 7078 |
| 2023-11-05 | 6870 |
| 2023-11-06 | 6944 |
| 2023-11-07 | 7027 |

[537 rows x 1 columns]

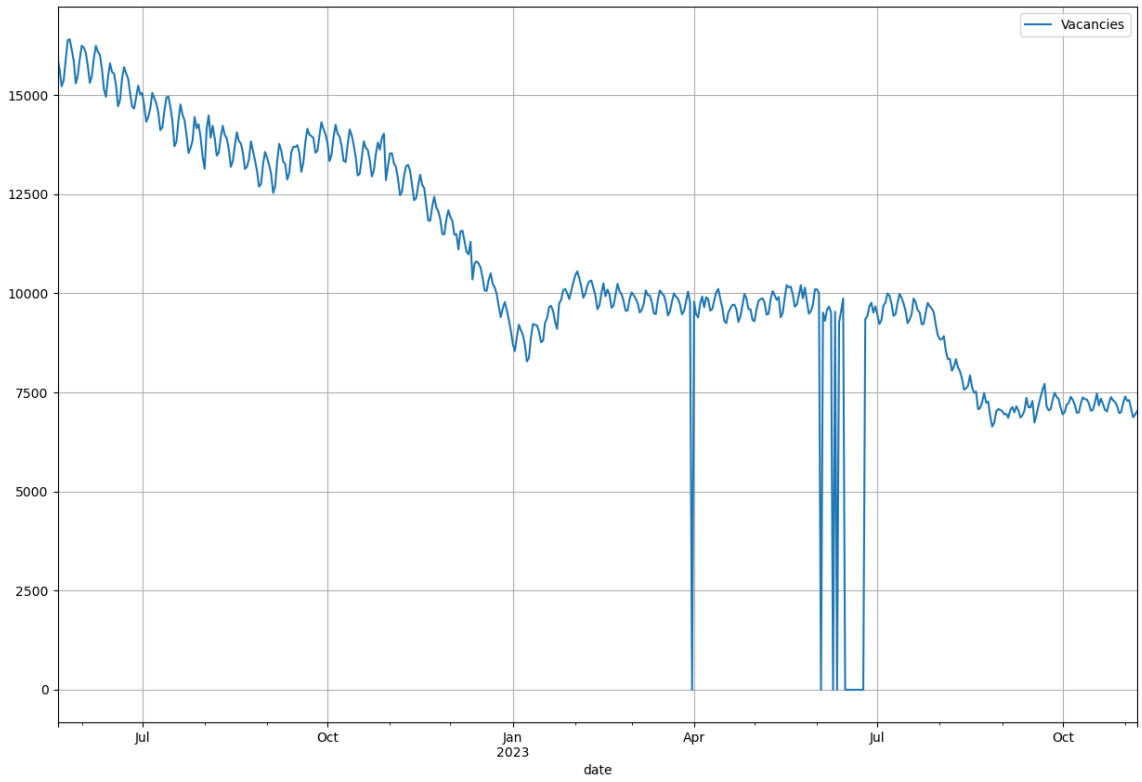
[537 rows x 1 columns]

| | vacancies |
|---|---|
| count | 537.000000 |
| mean | 10598.249534 |
| std | 3146.542104 |
| min | 0.000000 |
| 25% | 9226.000000 |
| 50% | 9943.000000 |
| 75% | 13409.000000 |
| max | 16407.000000 |

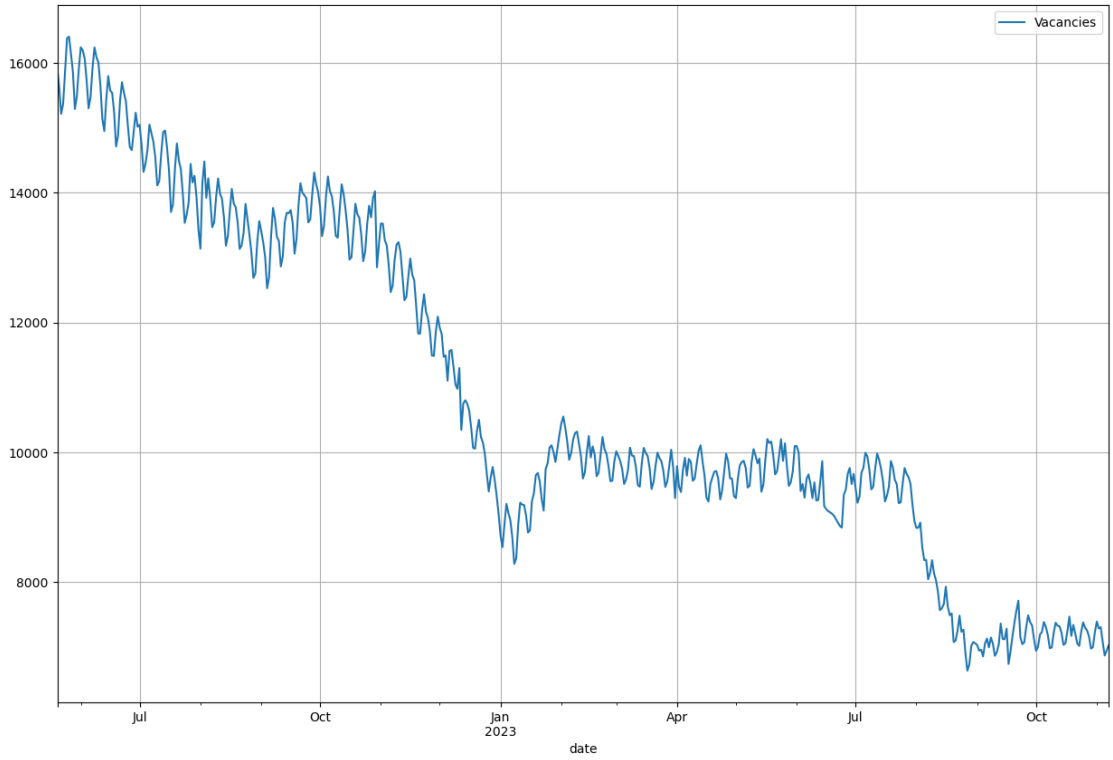
Насторожує присутність нульових значень. Переглядаємо кількість відсутніх:

```
Count of missing values: 0
```

Візуалізуємо ряд:



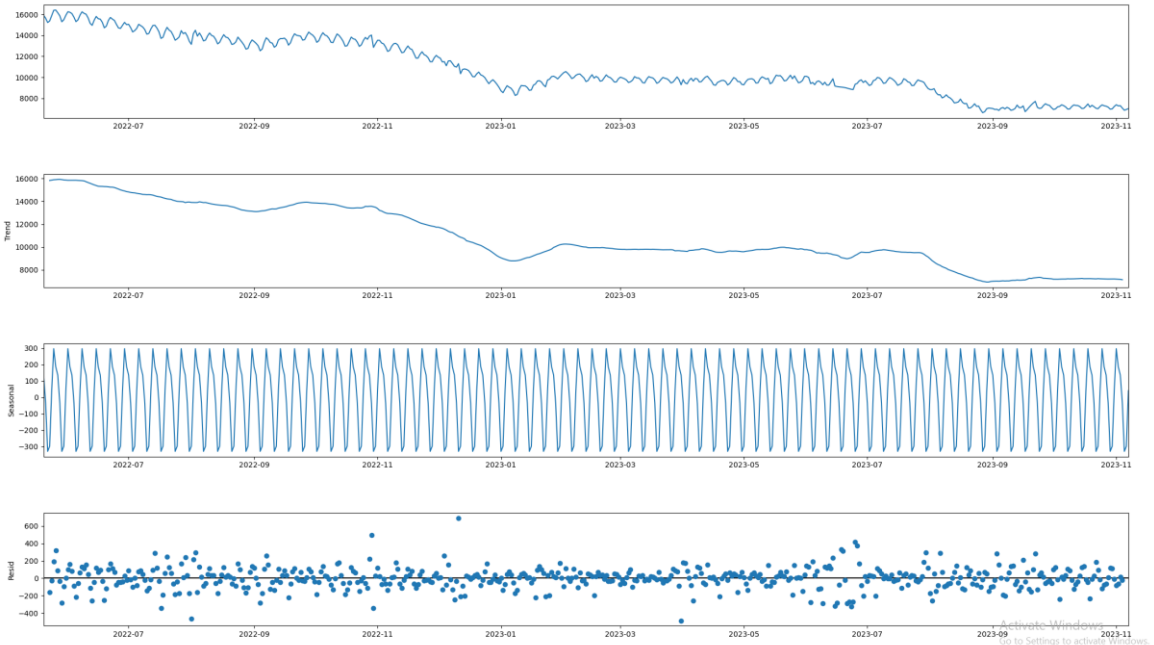

Графік демонструє наявність великої кількості нульових значень, які скоріш за все спричинені технічними неполадками, а не дійсною раптовою відсутністю вакансій. Аби виправити це, замінюємо нульові значення на середні ковзкі з розміром вікна в 20:




3. Чи є сезонні коливання? Відобразити графіки, виключивши сезонні коливання, якщо вони є.

   Попередньо на графіку можна помітити деякі циклічні коливання.

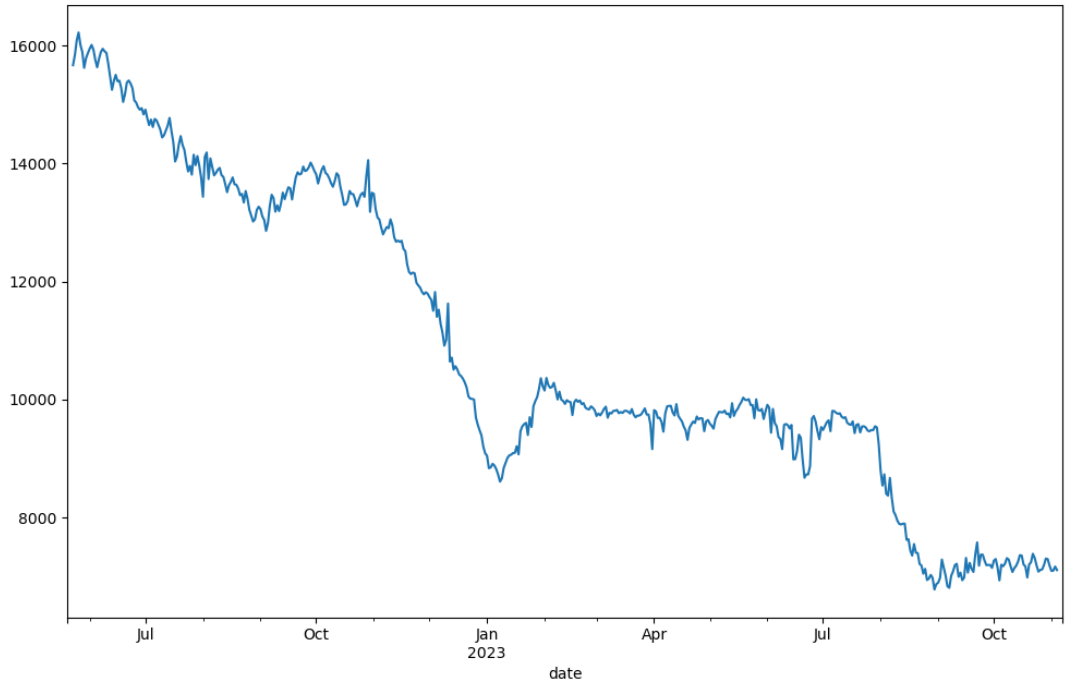
   Проводимо декомпозицію даних на основні шари:




   Бачимо присутність значної циклічної сезонної складової, тому робимо висновок, що сезонність присутня.

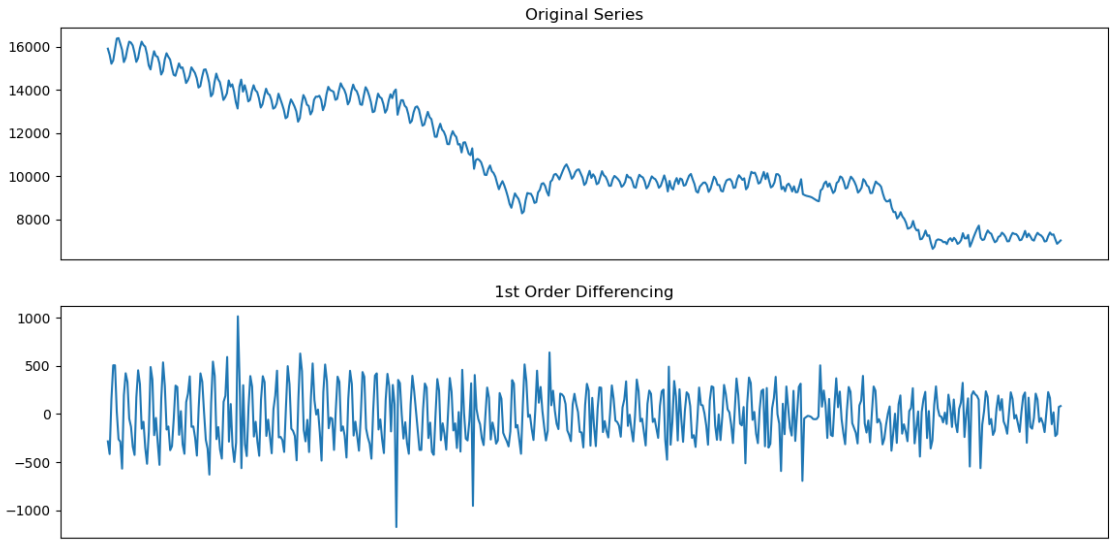
   Сезонна компонента в даному випадку має додавальний характер,

отож, аби виключити сезонун компоненту, потрібно просто додати тренд та залишок:




4. Спрогнозувати котировку на тиждень вперед (використовуючи регресію та модель ARIMA). Яка якість прогнозу?

   Почнімо з ARIMA. Спочатку необхідно підібрати гіперпараметри – p d q.
   Для встановлення значення параметра d проводимо диференціювання послідовності:

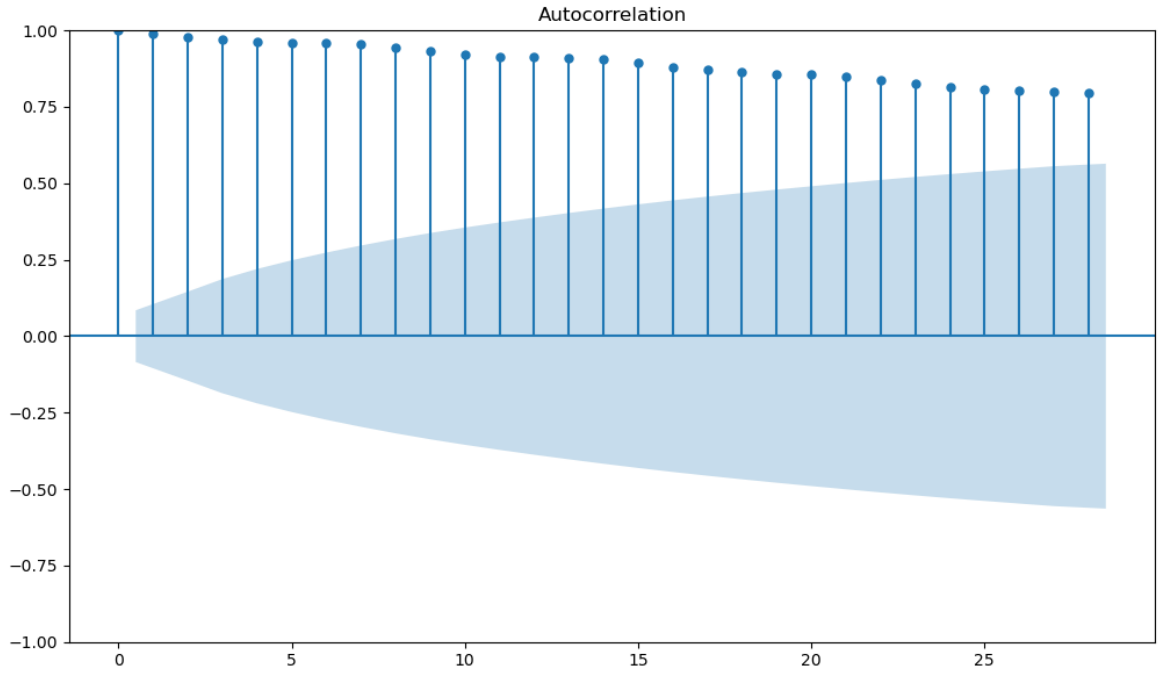


Як бачимо, початкова послідовність демонструє сезонність, однак після першого диференціювання вона, схоже, набуває стаціонарного характеру. Аби переконатися, застосуємо тест дікі фуллера:

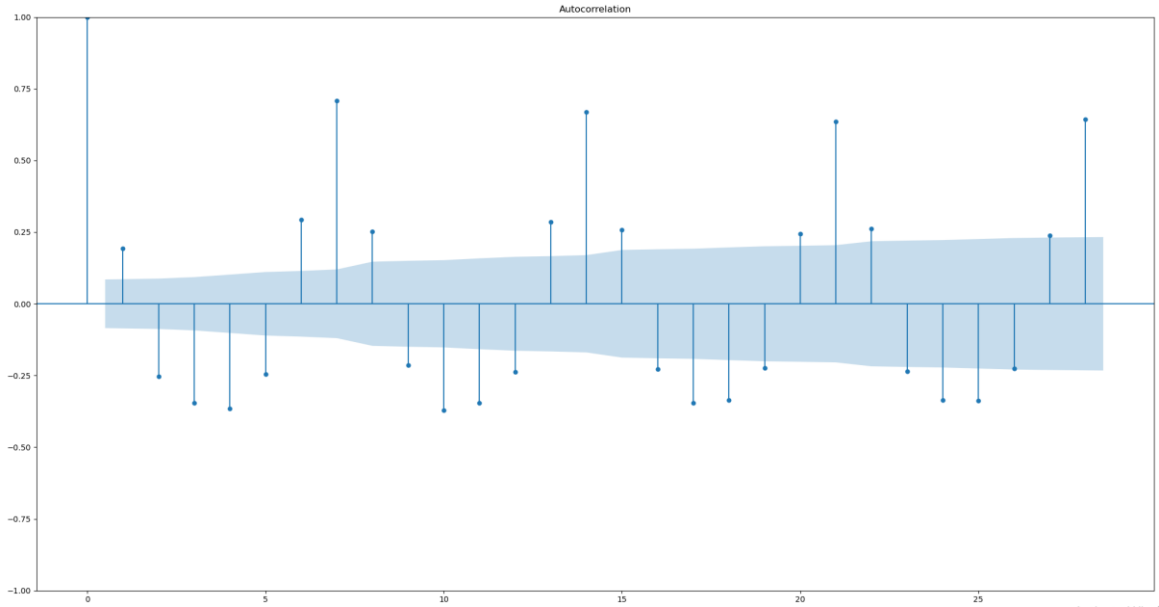
```
ADF Statistic: -4.112811
p-value: 0.000922
Critical Values:
    1%: -3.443
    5%: -2.867
    10%: -2.570
Series is stationary
```

Отож d=1

Тоді для встановлення q візуалізуємо автокореляцію:

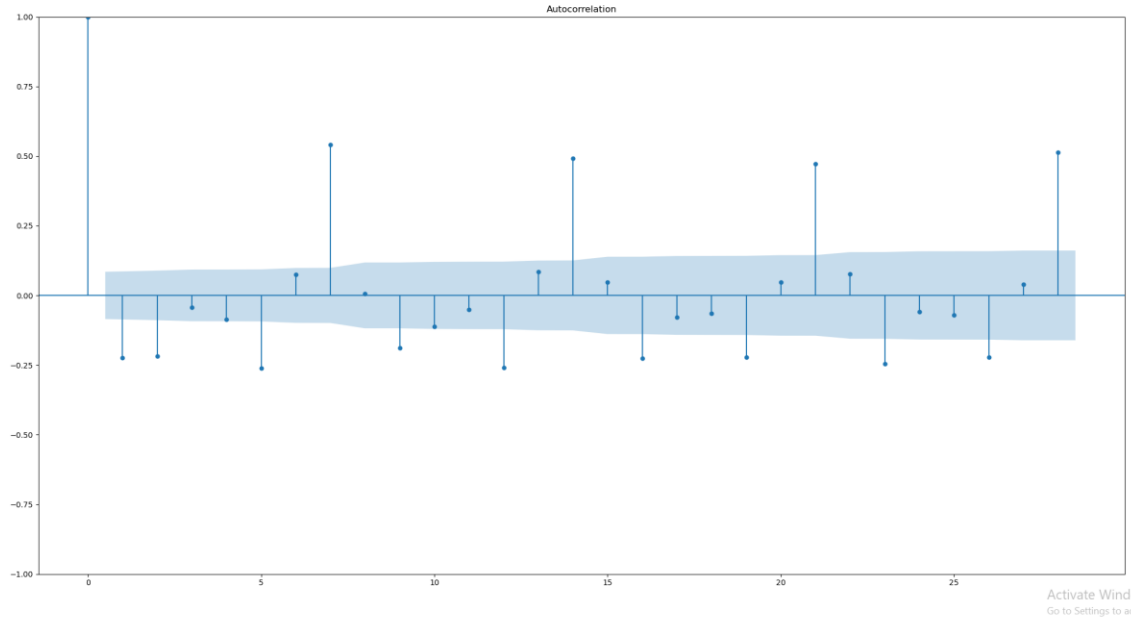



Бачимо, що автокорелція всюди сильна, частково через не-стаціонарність. Переглянемо автокореляцію для продиференційованої послідовності:



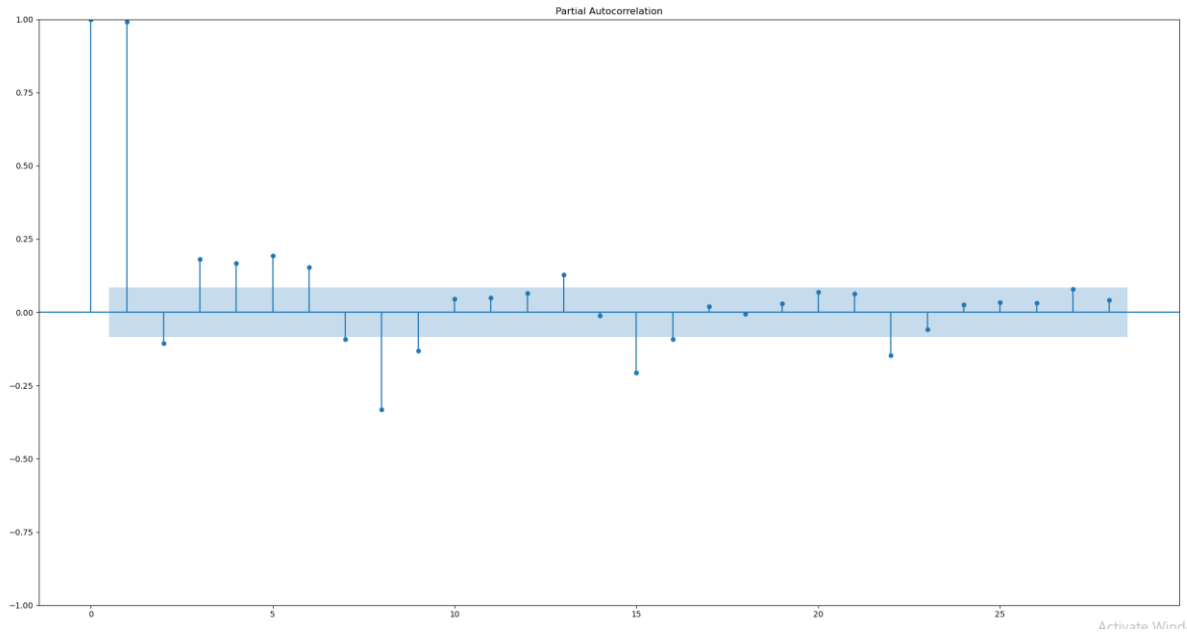


Краще, але все ще присутня велика кількість послідовно автокорельованих періодів. Диференціюємо ще раз:



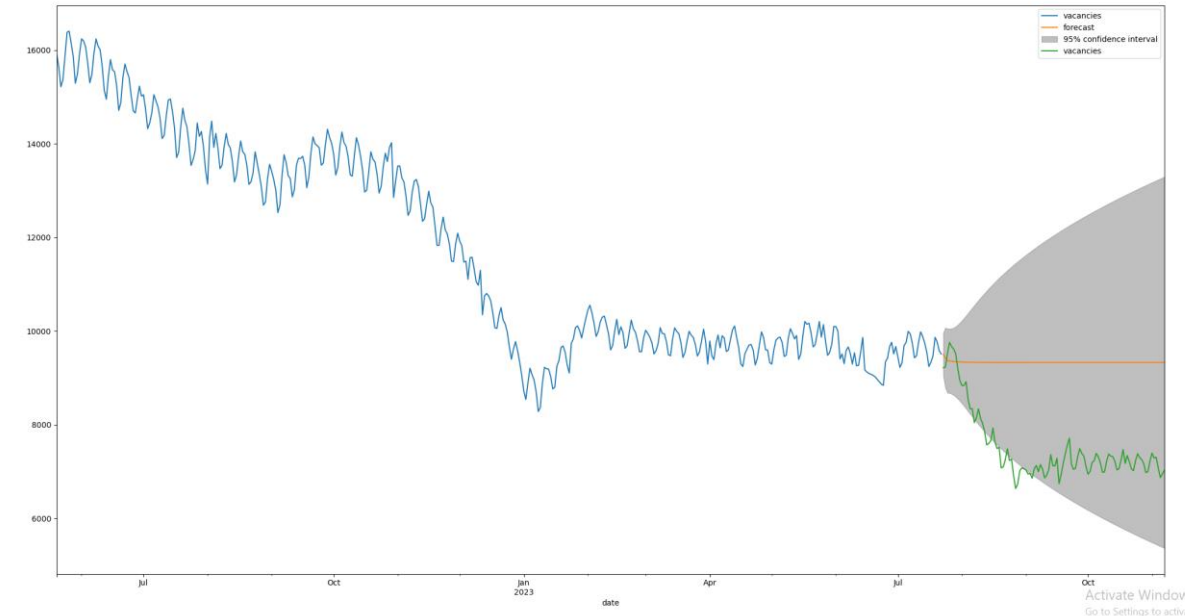


Бачимо, що 4 періоди значною мірою виходять за межі критичної зони, тому q = 4.

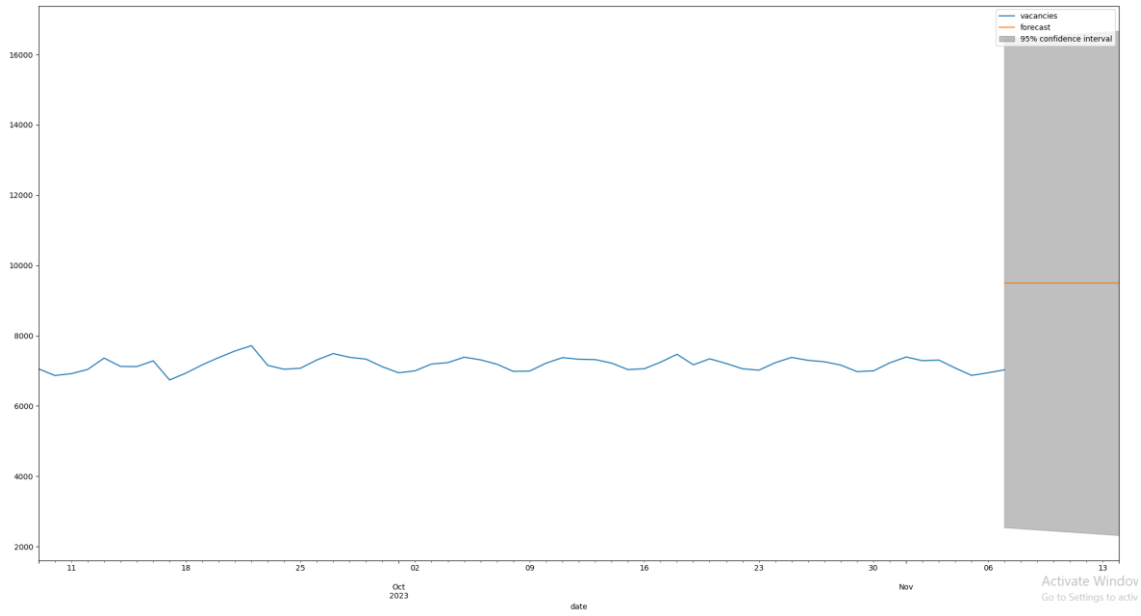



Бачимо, що від початку найближчою до критичної області є третій період, отож p = 2

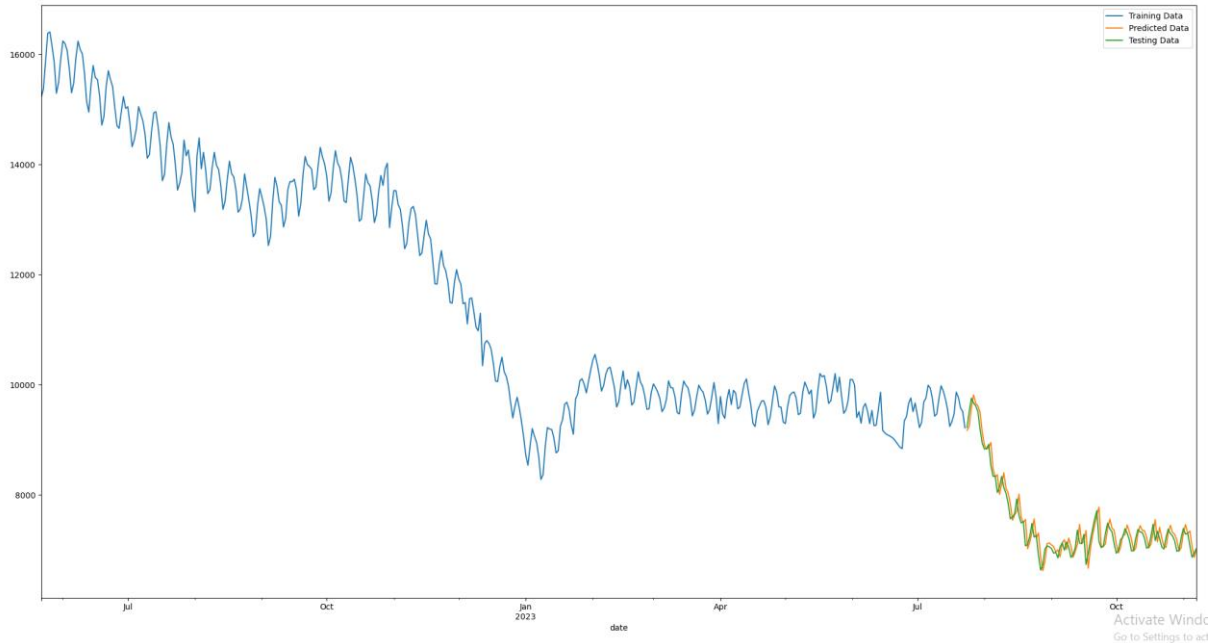
Тепер створюємо та тренуємо модель.  У результаті маємо такі передбачення для тестової вибірки:




Метрики непотрібні, адже, вочевидь, точність моделі надзвичайно далека від дійсності.
Передбачуємо наступний тиждень:




Тепер створюємо регресію. У якості типу моделі було обрано лінійну регресію, а в кості предикторів вона прийматиме останні два попередні значення числа вакансій. Результат:

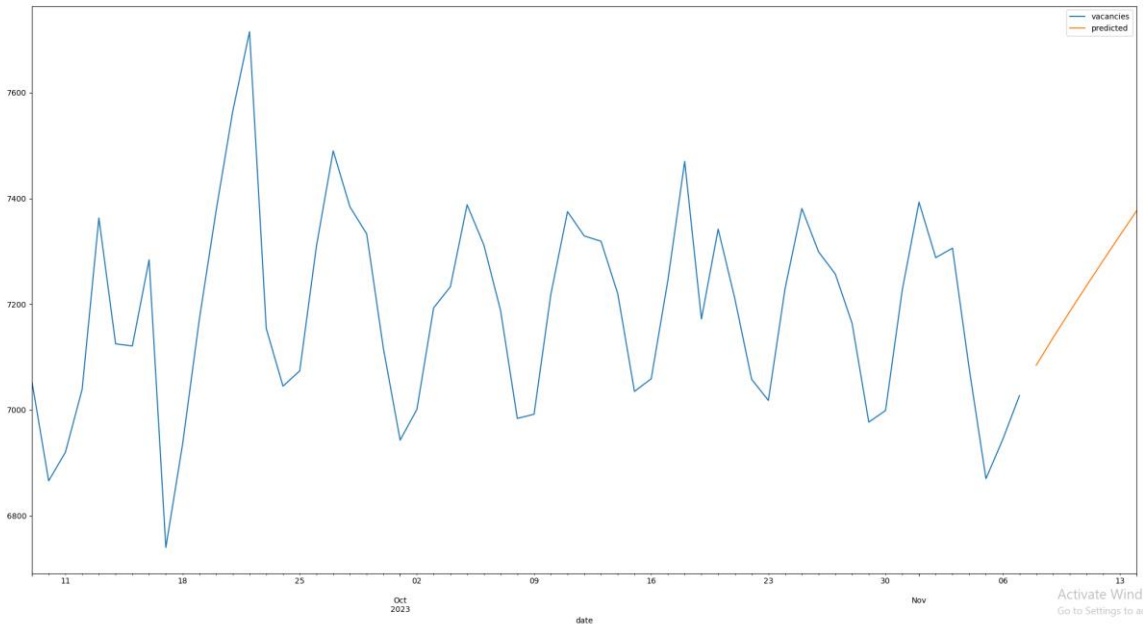



Для чисельної оцінки якості звертаємося до MSE та R^2:

```
Mean Squared Error: 37515.24
R-squared (R2) Score: 0.93
```

Як бачимо, середня квадратична похибка доволі велика, але R^2 доволі близький до 1.

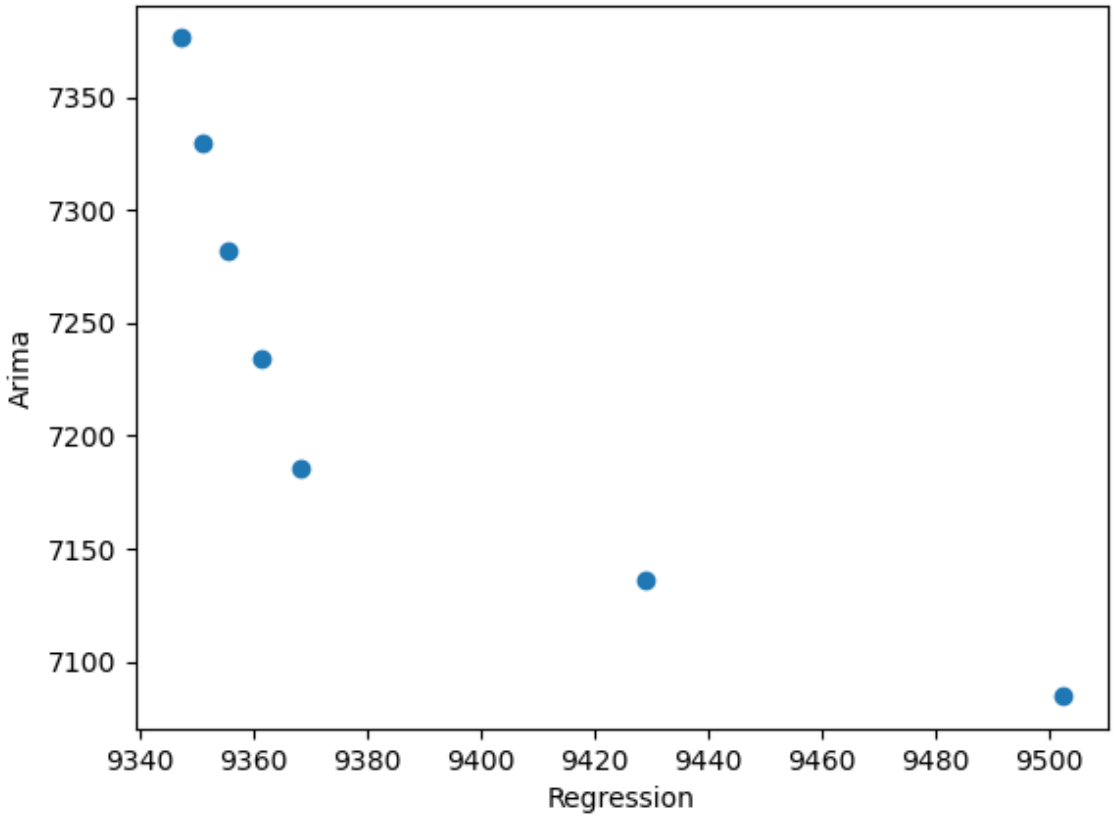
Передбачуємо наступний тиждень:




Тепер проаналізуємо кореляцію між наборами передбачень:

```
Correlation of two prediction sets: -0.8569448462298827
```

Присутня сильна негативна кореляція.
Будуємо діаграму розсіювання:



## Висновок

Отож, у ході виконання лабораторної роботи було взято набір даних про кількість вакансій розробника ПЗ та підготовлено їх для подальшого аналізу: перейменовано назви стовпців на більш лаконічні та замінено нульові значення на середні. Часовий ряд було візуалізовано та проаналізовано на сезонну компоненту – її було виявлено. Відповідно, опісля було побудовано лінійний графік послідовності з виключенням даної складової. Проведено підбір гіперпараметрів для моделі ARIMA за допомогою аналізу стаціонарності, автокореляції та часткової автокреляції, створено та натреновано екземпляр такої моделі й застосовано його для формування передбачень, результати візуалізовано. Під час виконання роботи було створено лінійну регресію, що базується на двох попередніх значеннях кількості вакансій. Дану модель аналогічно було натреновано та використано для передбачення, а результати візуалізовано. Було проаналізовано ефективність даної регресії, базуючись на метриках MSE та R^2 – результати виявилися відносно задовільними. Урешті-решт було перевірено корельованість таких наборів передбачень – кореляція виявилася сильною негативною- та побудовано їхню діаграму розсіювання. Набуто практичних навичок аналізу часових послідовностей та створення моделей передбачення для них мовою Python.

# Додаток А

## Текст скрипту для виконання лабораторної роботи

```
import pandas as pd
import statsmodels.tsa.api as smt
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import statsmodels.graphics.tsaplots as tsaplots
from sklearn.linear_model import LinearRegression
from datetime import timedelta
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from statsmodels.tsa.stattools import adfuller
import numpy as np

if __name__=="__main__":
    pd.set_option('display.max_columns', None)

    data=pd.read_csv("data/rozrobka-softu.csv",index_col=['Дата'],
parse_dates=['Дата'],encoding="utf-8")
    print(data)

    data.drop(columns=["дельта"], inplace=True)
    data.rename_axis("date", inplace=True)
    data.rename(columns={"Популярні мови програмування згідно кількості вакансій в
Україні":"vacancies"}, inplace=True)

    print(data)
    print(data.describe())

    print(f"\nCount of missing values: {data['vacancies'].isna().sum()}")

    fig, ax = plt.subplots(figsize=(15, 10))
    data.plot(ax=ax)
    plt.legend(["Vacancies"])
    ax.grid()
    plt.show() #zeros detected

    for index, row in data.iterrows():
        if row['vacancies'] == 0:
            start_idx = max(data.index.get_loc(index) - 20, 0)
            rolling_avg = data.loc[data.index[start_idx]:index,
'vacancies'].mean()
            data.at[index, 'vacancies'] = rolling_avg

    fig, ax = plt.subplots(figsize=(15, 10))
    data.plot(ax=ax)
    plt.legend(["Vacancies"])
    ax.grid()
    plt.show() #better

    # decompose to basic layers
    decomp = smt.seasonal_decompose(data, model='additive')
    decomp.plot()
    plt.show()

    seasonal_substracted = decomp.trend+decomp.resid
```

```
    seasonal_substracted.plot()
    plt.show()

    #creating arima

    time_difference = data.index - data.index.min()
    threshold=(data.index.max() - data.index.min()).days*0.8

    data_train=data[time_difference.days < threshold]
    data_test = data[time_difference.days >= threshold]

    #picking parameters

    fig, (ax1, ax2) = plt.subplots(2)
    ax1.plot(data["vacancies"])
    ax1.set_title('Original Series')
    ax1.axes.xaxis.set_visible(False)

    ax2.plot(data["vacancies"].diff())
    ax2.set_title('1st Order Differencing')
    ax2.axes.xaxis.set_visible(False)
    plt.show()#becomes stationary - d = 1

    adfuller_test = adfuller(data["vacancies"].diff().dropna())
    print('ADF Statistic: %f' % adfuller_test[0])
    print('p-value: %f' % adfuller_test[1])
    print('Critical Values:')
    for key, value in adfuller_test[4].items():
        print('\t%s: %.3f' % (key, value))
    if adfuller_test[0] > adfuller_test[4]['5%']:
        print('Series is non-stationary')
    else:
        print('Series is stationary')

    plot_acf(data["vacancies"])
    plt.show()  # nonstationary

    plot_acf(data["vacancies"].diff().dropna())
    plt.show()  # better, but too many sequential lags oout of critical area

    plot_acf(data["vacancies"].diff().diff().dropna())
    plt.show() # 4 lags have strong autocorrelation - q = 4
    plot_pacf(data["vacancies"])
    plt.show()# third is the closest to limit - so p = 1 (first lag ignored)"""


    model = ARIMA(data_train["vacancies"], order=(1, 1, 4))
    arima_result = model.fit()



    #evaluating quality
    fig, ax = plt.subplots(figsize=(10, 6))
    data_train.plot(ax=ax)
    tsaplots.plot_predict(arima_result, ax=ax,
                          start=data_test.index.min(),
                          end=data_test.index.max())
    data_test.plot(ax=ax)
    plt.show()
```

```
    future_data_arima = arima_result.forecast(steps=7)

    #predicting data for next week
    fig, ax = plt.subplots(figsize=(10, 6))
    data.tail(60).plot(ax=ax)
    tsaplots.plot_predict(arima_result, ax=ax,
                          start=data.index.max(),
                          end=data.index.max()+timedelta(days=7))
    plt.show()

    #going with regression
    lag_orders = 2

    regr_data=pd.DataFrame(data)

    for i in range(1, lag_orders + 1):
        regr_data[f'vacancies_lag_{i}'] = regr_data['vacancies'].shift(i)

    regr_data = regr_data.dropna()

    X = regr_data[['vacancies_lag_1', 'vacancies_lag_2']]
    y = regr_data['vacancies']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, shuffle=False)

    model = LinearRegression()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f"Mean Squared Error: {mse:.2f}")
    print(f"R-squared (R2) Score: {r2:.2f}")

    y_pred_series = pd.Series(y_pred, index=y_test.index)

    fig, ax = plt.subplots(figsize=(10, 6))
    y_train.plot(ax=ax, label='Training Data')
    y_pred_series.plot(ax=ax, label='Predicted Data')
    y_test.plot(ax=ax, label='Testing Data')
    plt.legend()
    plt.show()

    last_date = data.index.max()
    future_dates = pd.date_range(start=last_date + timedelta(days=1), periods=7)
    future_data_regr = pd.DataFrame({'date': future_dates, 'vacancies': np.nan})
    future_data_regr.set_index('date', inplace=True)

    y_tmp=pd.DataFrame(y, index=y.index)

    for i in range(7):
        last_observation = y_tmp.iloc[-1]['vacancies']
        lag_1 = y_tmp.iloc[-1]['vacancies']
        lag_2 = y_tmp.iloc[-2]['vacancies']

        prediction = model.predict([[lag_1, lag_2]])
        future_data_regr['vacancies'].iloc[i] = prediction[0]
```

```
        y_tmp.loc[last_date + pd.DateOffset(days=i + 1)]=prediction[0]

    fig, ax = plt.subplots(figsize=(10, 6))
    y.tail(60).plot(ax=ax, label="vacancies")
    future_data_regr['vacancies'].plot(ax=ax, label="predicted")
    plt.legend()
    plt.show()

    #correlations
    correlation = np.corrcoef(future_data_arima.values,
future_data_regr['vacancies'].values)[0, 1]
    print(f"Correlation of two prediction sets: {correlation}")

    plt.scatter(future_data_arima, future_data_regr['vacancies'])
    plt.xlabel('Regression')
    plt.ylabel('Arima')
    plt.show()
```