Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

Бакалаврська дипломна робота на тему:

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ПОШИРЕННЯ ПИЛКУ САХАРИ У ПОВІТРІ УКРАЇНИ ЗА ДАНИМИ ГРОМАДСЬКОГО МОНІТОРИНГУ»

Виконав: студент 4 курсу, групи 2ІСТ-20б спеціальності 126 – «Інформаційні системи та технології»
_____ Тарас СКРИННИК
Керівник: к.т.н., доц. каф. САІТ
_____ Сергій ЖУКОВ
«31» 05 2024 р.
Рецензент: к.т.н., доц. каф. КН
_____ Ігор АРСЕНЮК
«10» 06 2024 р.

Допущено до захисту

Завідувач кафедри САІТ

_____ д.т.н., проф. Віталій МОКІН

«03» 06 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
д.т.н., проф. Віталій МОКІН
"05" 05 2024 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Скриннику Тарасу Васильовичу

1. Тема роботи: «Інформаційна технологія аналізу та прогнозування поширення пилку Сахари у повітрі України за даними громадського моніторингу»
керівник роботи: Жуков С. О., к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «11» 03 2024 року №80
2. Термін подання студентом роботи 31.05
3. Вихідні дані до роботи:
1) Дані сервісу моніторингу якості повітря EcoCity.
4. Зміст текстової частини:
1) Загальна характеристика об'єкту дослідження та аналіз інформаційних технологій;
2) Підготовка та розвідувальний аналіз даних;
3) Розробка інформаційної технології та прогнозування.
5. Перелік ілюстративного матеріалу:
1) Вигляд датасету;
2) Графіки аномалій;
3) Блок-схема алгоритму інформаційної технології аналізу та прогнозування поширення пилу Сахари Україною;
4) Результати прогнозування якості повітря та поширення пилу Сахари.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 11.03

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Загальна характеристика об'єкту дослідження	11.03	31.03	
2	Підготовка та розвідувальний аналіз даних	01.04	15.04	
3	Створення інформаційної технології прогнозування	16.04	30.04	
4	Прогнозування поширення пилку	01.05	15.05	
5	Оформлення матеріалів до захисту БДР	16.05	31.05	

Студент _____ Тарас СКРИННИК

Керівник роботи _____ Сергій ЖУКОВ

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 84 сторінок формату А4, на яких є 58 рисунків, список використаних джерел містить 31 найменування.

Метою роботи є розробка інформаційної технології для прогнозування поширення пилу Сахари у повітрі України за даними громадського моніторингу.

У роботі представлено розробку алгоритмів для збору, обробки та аналізу даних, а також створення моделей для прогнозування рівня забруднення повітря. Було розроблено програмне забезпечення для обробки даних, яке включає в себе алгоритми аналізу та прогнозування, а також проведено тестування на реальних даних. Розроблена інформаційна технологія забезпечує точність і оперативність прогнозування забруднення повітря пилом Сахари, що дозволяє покращити якість моніторингу та своєчасне реагування на екологічні загрози.

Ключові слова: інформаційна технологія, прогнозування, якість повітря, пилок Сахари, громадський моніторинг.

ABSTRACT

The Bachelor's thesis consists of 84 pages of A4 format, which contain 58 figures. The list of used sources contains 31 titles.

The aim of the work is to develop an information technology for forecasting the spread of Sahara dust in the air of Ukraine based on public monitoring data.

The work presents the development of algorithms for data collection, processing and analysis, as well as the creation of models for forecasting the level of air pollution. Software for data processing was developed, which includes analysis and forecasting algorithms, and testing on real data was carried out. The developed information technology ensures the accuracy and timeliness of forecasting air pollution with Sahara dust, which allows to improve the quality of monitoring and timely response to environmental threats.

Keywords: information technology, forecasting, air quality, Sahara dust, public monitoring.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі якість повітря є однією з найважливіших екологічних проблем, що впливає на здоров'я та добробут населення. Забруднення повітря може призводити до серйозних захворювань, таких як астма, хвороби серця та рак легень. Вивчення якості повітря і розробка ефективних методів моніторингу та прогнозування забруднення є критично важливими для забезпечення здорового середовища проживання. За останній рік на якість повітря в Україні вплинув пил із Сахари, який викликав багато дискомфорту серед населення.

Мета і задачі дослідження. Метою даної роботи є підвищення точності прогнозування поширення пилу Сахари в атмосферному повітрі України за даними громадського моніторингу шляхом створення інформаційної технології.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

– провести аналіз проблеми прогнозування якості повітря та проаналізувати існуючі сервіси моніторингу якості повітря;

– здійснити вибір оптимальних інформаційних технологій та моделей машинного навчання для прогнозування рівня забруднення повітря пилом Сахари для збору та обробки даних;

– здійснити розвідувальний аналіз даних;

– розробити інформаційну технологію аналізу та прогнозування якості повітря та поширення пилу Сахари Україною

Об’єктом дослідження є процес розроблення інформаційної технології аналізу та прогнозування якості повітря та поширення пилу Сахари Україною.

Предметом дослідження є методи машинного навчання та інформаційна технологія аналізу та прогнозування якості повітря та поширення пилу Сахари Україною.

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ОБ'ЄКТУ ДОСЛІДЖЕННЯ ТА АНАЛІЗ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1 Аналіз предметної області

Якість повітря – міра чистоти повітря, яке нас оточує та яким ми дихаємо. Ця міра визначається концентрацією забруднювачів у повітрі, таких як пил, хімічні речовини, гази чи дим.

Якість повітря вимірюється за допомогою індексів, які дають загальну оцінку забруднення. У багатьох країнах ці шкали можуть бути різними, наприклад у Канаді це AQHI або Air Quality Health Index, що у перекладі означає - Індекс здоров'я за якістю повітря, на рисунку 1.1 зображена шкала вимірювання.

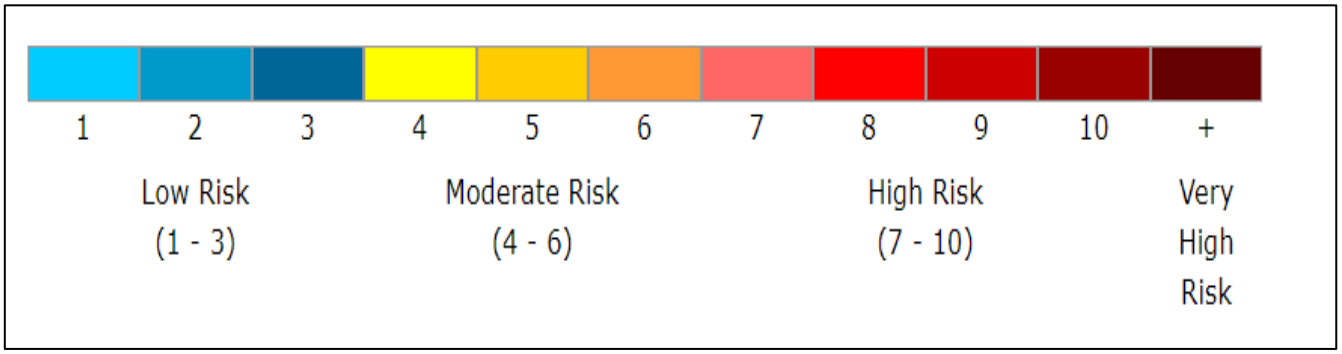


Рисунок 1.1 – Шкала вимірювання AQHI в Канаді

На рисунку 1.2 зображена шкала, яка використовується у Великій Британії, де зелений колір означає, що якість повітря вважається задовільною, а забруднення повітря не становить небезпеки або не становить нічого. Жовтий та помаранчевий - якість повітря прийнятна; однак деякі забруднювачі можуть викликати помірне занепокоєння для здоров'я дуже невеликої кількості людей, які надзвичайно чутливі до забруднення повітря. Відтінки червоного - кожен може почати відчувати вплив

на здоров'я; члени чутливих груп можуть мати більш серйозні наслідки для здоров'я. Чорний - кожен може мати більш серйозні наслідки для здоров’я [1].

Band	Index	Ozone Running 8-hour mean (μg m⁻³)	Nitrogen dioxide 1-hour mean (μg m⁻³)	Sulphur dioxide 15-minute mean (μg m⁻³)	$PM_{2.5}$ particles 24-hour mean (μg m⁻³)	PM_{10} particles 24-hour mean (μg m⁻³)
Low	1	0–26	0–66	0–88	0–11	0–16
	2	27–53	67–133	89–176	12–23	17–33
	3	54–80	134–200	177–265	24–35	34–50
Moderate	4	81–107	201–267	266–354	36–41	51–58
	5	108–134	268–334	355–442	42–46	59–66
	6	135–160	335–400	443–531	47–53	67–75
High	7	161–187	401–467	532–708	54–58	76–83
	8	188–213	468–534	709–886	59–64	84–91
	9	214–240	535–600	887–1063	65–70	92–100
Very High	10	241 or more	601 or more	1064 or more	71 or more	101 or more

Рисунок 1.2 – Шкала вимірювання якості повітря у Великій Британії

В Україні ми використовуємо AQI або ж «Індекс якості повітря» для оцінки щоденного рівня забруднення в 6 категоріях, а саме:

– Добра якість повітря;

– задовільна якість повітря;

– якість повітря несприятлива для чутливих до забруднення повітря груп населення;

– погана якість повітря;

– дуже погана якість повітря;

– надзвичайно погана якість повітря.

Індекс якості повітря ґрунтується на концентраціях п'яти основних забруднювачів [2]:

– Дрібні тверді частинки (PM_1, $PM_{2.5}$ та PM_{10});

– Озон (O_3);

– Діоксид азоту (NO_2);

– Оксид вуглецю (CO);

– Діоксид сірки (SO_2).

Дані про концентрації цих забруднювачів збираються станціями моніторингу повітря по всій країні, а потім використовуються для розрахунку індексу якості повітря.

1.2 Аналіз сервісів моніторингу якості повітря в Україні та аналіз проблеми пилку Сахари

Якість повітря, яке нас оточує та яким ми дихаємо прямо впливає на наш добробут та здоров'я. Забруднене повітря спричиняє близько 7-9 млн смертей щорічно [3].

Погана якість повітря може викликати захворювання дихальної системи (рисунок 1.3 [4]) , як астма, бронхіт, пневмонія, інфаркт, інсульт, рак легень або ж спричиняти інші симптоми, а саме: головний біль, запаморочення, втому, подразнення очей, важке дихання та інші.

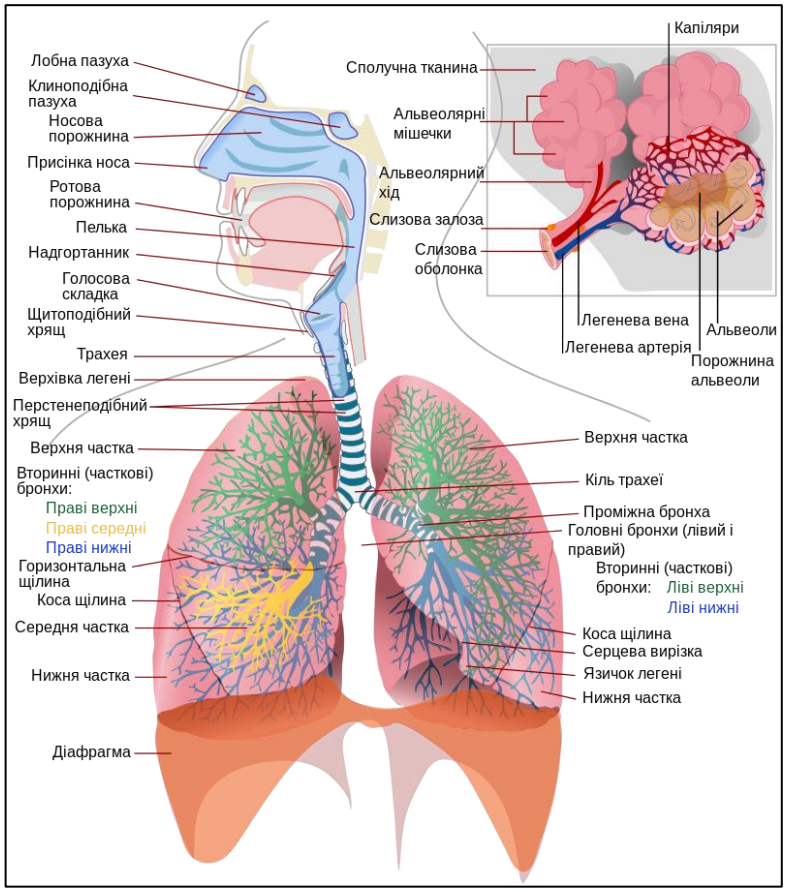


Рисунок 1.3 – Дихальна система людини

Таким чином, моніторинг якості повітря є надзвичайно важливим для забезпечення здоров'я та добробуту населення. Регулярне відстеження рівня забруднення повітря дозволяє не лише своєчасно виявляти небезпечні концентрації шкідливих речовин, але й приймати ефективні заходи для їх зниження. Сьогодні існує багато сервісів моніторингу якості повітря, які надають точну та актуальну інформацію про стан повітря в реальному часі. Розглянемо деякі з них.

SaveEcoBot - це комплексна екологічна система, що охоплює моніторинг довкілля, інструменти для боротьби з забрудненням та платформу для обміну даними [5].

Уявіть собі інтерактивну карту України, де ви можете бачити поточний стан якості повітря у вашому місті, селі чи навіть на вулиці. На рисунку 1.4 зображений приклад відображення інформації на сайті https://www.saveecobot.com/.

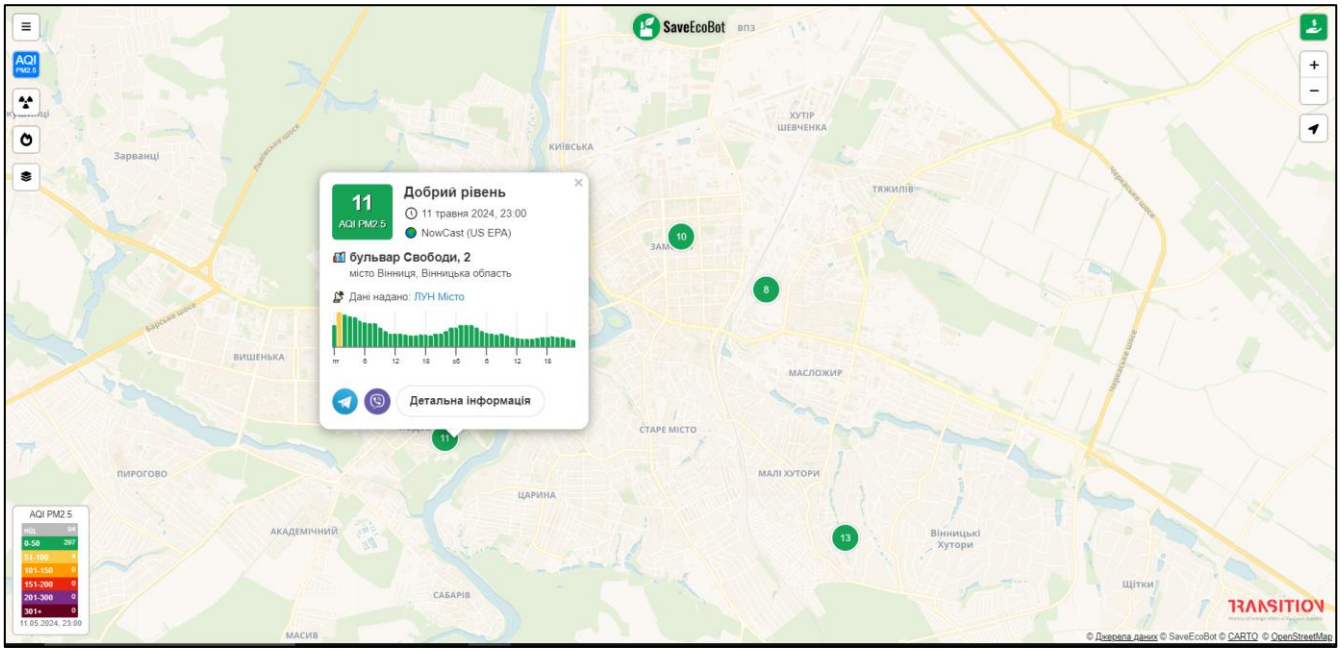


Рисунок 1.4 – Сайт SaveEcoBot

SaveEcoBot збирає дані з різних джерел, включаючи державні системи, місцеві органи влади, громадські станції та комерційні прилади, щоб надати максимально точну картину.

Крім того, SaveEcoBot показує рівень радіаційного фону по всій країні, адже це важливий показник екологічної безпеки. А якщо треба проаналізувати динаміку змін, можна переглянути історичні дані про якість повітря за певний період часу.

SaveEcoBot дає доступ до актуальної інформації про екологічні податки та збори, що стягуються в Україні. Платформа пропонує аналітичні інструменти, які допоможуть вам виявити тенденції та прийняти обґрунтовані рішення.

SaveEcoBot також веде реєстр оцінок впливу на довкілля (ОВД), які є обов'язковими для певних видів діяльності, що можуть мати негативний вплив на довкілля.

SaveEcoBot пропонує віджет, який можна встановити на свій веб-сайт або блог, щоб показувати поточний стан якості повітря у вашому регіоні.

Eco-City.org.ua - це веб-сайт, що належить проекту громадського моніторингу якості повітря в Україні. Це некомерційна ініціатива, яка прагне надати українським

громадянам точну та доступну інформацію про стан повітря. Проект керують волонтери та ентузіасти, які дбають про захист довкілля [6].

Особливості Eco-City.org.ua:

Мапа якості повітря: На сайті є інтерактивна карта, яка показує поточні показники якості повітря для різних місць по всій Україні. Мапа має кольорове кодування, щоб вказати рівень забруднення повітря, де зелений колір позначає хорошу якість повітря, а червоний – погану [6], детальніше про шкалу можна глянути на рисунках 1.5 та 1.6.

Категорія	Значення	Для уразливих та чутливих груп
Категорія якості повітря I	**Добра якість повітря**	Вміст у повітрі забруднюючих речовин у межах норми та не становить небезпеки. Плануйте діяльність, відпочинок або інші активності на відкритому повітрі без додаткових обмежень, застережень або рекомендацій.
Категорія якості повітря II	**Задовільна якість повітря**	Вміст у повітрі забруднюючих речовин у межах норми та не становить небезпеки. Плануйте діяльність, відпочинок або інші активності на відкритому повітрі без додаткових обмежень, застережень або рекомендацій. У рідких випадках дуже невеликої кількості населення, які надзвичайно чутливі до забруднення повітря, можуть виникнути дискомфорт та рефлекторні реакції – кашель, задишка тощо. Надзвичайно чутливим до забруднення повітря людям рекомендуємо зменшити тривалі або важкі фізичні навантаження на відкритому повітрі.
Категорія якості повітря III	**Якість повітря несприятлива**	Вміст у повітрі забруднюючих речовин у межах норми, проте уразливі та чутливі до забруднення категорії населення можуть мати наслідки для стану здоров'я при довготривалому впливі. Враховуйте спеціальні обмеження, застереження та рекомендації під час планування діяльності, відпочинку або інших активностей на відкритому повітрі. Рекомендуємо уразливим та чутливим до забруднення категоріям населення зменшити тривалі або важкі фізичні навантаження на відкритому повітрі. Якщо ви відчуваєте дискомфорт під час дихання та прояви інших рефлекторних реакцій – рекомендуємо дотримуватися звичайних порад та плану лікування від вашого лікаря. Люди хворі на астму, ХОЗЛ та респіратері захворювання можуть відчувати посилення звичних симптомів та рефлекторних реакцій. Рекомендуємо сумлінно дотримуватися свого плану терапії хронічних захворювань та прийому ліків, які призначив ваш лікар. Люди із хронічними захворюваннями серцево-судинної системи можуть відчувати додаткові та посилені рефлекторні реакції – підвищене серцебиття, задишка або незвичайна втома. У випадку довготривалого прояву цих реакцій рекомендуємо звернутися за консультацією до вашого лікаря та сумлінно виконувати його настанови.

Рисунок 1.5 – Шкала якості повітря на веб-сайті EcoCity

Категорія якості повітря IV	**Погана якість повітря**	Вміст у повітрі забруднюючих речовин становить небезпеку при довготривалому впливі. Усі категорії населення можуть відчувати загострення рефлекторних реакцій та мати наслідки для здоров'я при довготривалому впливі. Діють спеціальні обмеження, застереження та рекомендації для тривалості діяльності, відпочинку або інших активностей на відкритому повітрі. Усім уразливим та чутливим до забруднення категоріям населення рекомендуємо виключити та перенести на інших час тривалі або важкі фізичні навантаження на відкритому повітрі. Чутливі до забруднення категорії населення, які мають хронічні захворювання, можуть відчувати значне посилення звичних симптомів та рефлекторних реакцій. Рекомендуємо сумлінно дотримуватися свого плану терапії хронічних захворювань та прийому ліків, які призначив ваш лікар. У випадку довготривалого прояву рефлекторних реакцій рекомендуємо звернутися за консультацією до вашого лікаря та сумлінно виконувати його настанови.
Категорія якості повітря V	**Дуже погана якість повітря**	Для уразливих та чутливих до забруднення категорій населення вміст у повітрі забруднюючих речовин становить небезпеку навіть при короткостроковому впливі. Усі категорії населення можуть відчувати сильне загострення рефлекторних реакцій та мати наслідки для здоров'я небезпеку навіть при короткостроковому впливі. Для уразливих та чутливих до забруднення категорій населення діють спеціальні обмеження, застереження та рекомендації для перебування на відкритому повітря для будь-якої діяльності. Рекомендуємо виключити або перенести на інший час будь-яку діяльність на відкритому повітрі. Якщо у вас виникає рефлекторна реакція на забруднене повітря – кашель, задишка, подразнення слизових оболонок носоглотки, біль в очах тощо – рекомендуємо використовувати засоби індивідуального захисту органів дихання. У випадку довготривалого прояву рефлекторних реакцій або інших відчутних наслідків для вашого здоров'я рекомендуємо звернутися за консультацією до вашого лікаря та сумлінно виконувати його настанови.
Категорія якості повітря VI	**Надзвичайно погана якість повітря**	Вміст у повітрі забруднюючих речовин становить небезпеку навіть при короткостроковому впливі. Усі категорії населення можуть відчувати сильне загострення рефлекторних реакцій та мати наслідки для здоров'я небезпеку навіть при короткостроковому впливі. Для всіх категорій населення діють спеціальні обмеження, застереження та рекомендації для перебування на відкритому повітря для будь-якої діяльності. Рекомендуємо виключити або перенести на інший час будь-яку діяльність на відкритому повітрі. Якщо у вас виникає рефлекторна реакція на забруднене повітря – кашель, задишка, подразнення слизових оболонок носоглотки, біль в очах тощо – рекомендуємо використовувати засоби індивідуального захисту органів дихання. У випадку довготривалого прояву рефлекторних реакцій або інших відчутних наслідків для вашого здоров'я рекомендуємо звернутися за консультацією до вашого лікаря та сумлінно виконувати його настанови.

Рисунок 1.6 – Шкала якості повітря на веб-сайті EcoCity

Дані про якість повітря: На веб-сайті представлені детальні дані про якість повітря для кожної станції моніторингу, а також дані концентрацію різних забруднюючих речовин, таких як $PM_{2.5}$, PM_{10}, NO_2, SO_2 та O_3. На рисунку 1.7 зображений вигляд цих даних на сайті.

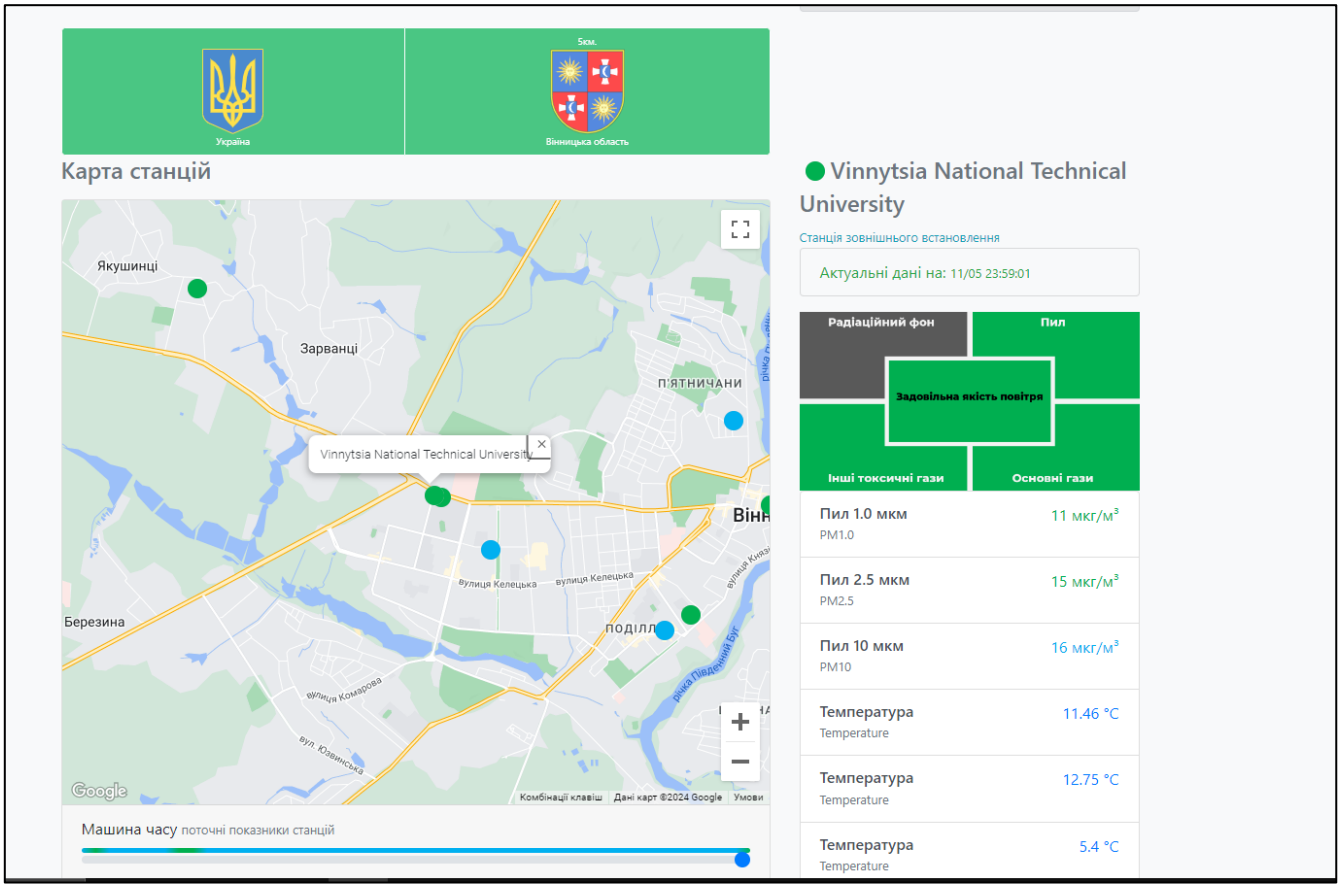


Рисунок 1.7 – Дані із сайту EcoCity

Рекомендації щодо здоров'я: На веб-сайті надаються рекомендації щодо здоров'я на основі поточного рівня якості повітря. Наприклад, людям з проблемами дихання може рекомендуватися уникати напруженої діяльності на свіжому повітрі, коли рівень забруднення повітря високий.

Попередження про якість повітря: Користувачі можуть зареєструватися, щоб отримувати попередження електронною поштою або SMS, коли рівень якості повітря в їхньому регіоні перевищує певні граничні значення.

Публічні дані: Дані проекту доступні громадськості безкоштовно для використання в дослідженнях, освіті та інших цілях.

Окрім онлайн-платформи, EcoCity також має мобільний додаток, який дозволяє користувачам отримувати доступ до інформації про якість повітря на ходу [6].

Загалом, EcoCity - це цінний ресурс для всіх, хто хоче бути в курсі якості повітря в Україні. Відданість проекту прозорості та доступності даних для громадськості робить його важливим інструментом для захисту довкілля та залучення громадян.

Причиною забруднення можуть бути як природні явища і техногенні наслідки.

До природних чинників можна віднести вулканічний пил, космічний пил, пилові бурі тощо. До техногенних відносять викиди транспорту, викиди промисловості і сільське господарство [7].

У рамках даної дипломної роботи я хочу зупинитись більше на пилових бурях, а саме на пилу із Сахари, який уже двічі цього року накривав Україну. Африканський пил може подорожувати по всій земній кулі до деяких частин Європи, Південної Америки, Центральної Америки, Карибського басейну та Сполучених Штатів. Це явище не є новим для євроатлантичного регіону, воно уже відбувалось раніше. Також в Україні його фіксували у липні 2021 року [8].

Що ж собою являє пил із Сахари? Це тверді частинки, які знаходяться в повітрі пустелі Сахара і переносяться по світу вітрами, циклонами та антициклонами. Присутність пилу в атмосфері викликає зміну забарвлення неба, що зображено на рисунку 1.8.

Рисунок 1.8 – Зміна забарвлення неба після впливу пилу з Сахари

Опади з пилом із Сахари осідають у вигляді грязьового дощу, або ж як його називають – кривавого дощу. На рисунку 1.9 показані наслідки таких опадів.

Рисунок 1.9 – Наслідки опадів із пилом Сахари

Також може бути сухе осідання пилу, відбувається воно тоді, коли концентрація частинок висока і ці частинки падають на поверхню землі під дією сили тяжіння.

У 2024 році Україну двічі накривало Сахарським пилом, а саме на початку квітня та в кінці квітня. Піки забруднення та проходження пилу територією України припадають на 1 квітня та 24 квітня відповідно. Причинами цих явищ стали погодні умови та рух повітряних мас планетою. З 31 березня по 1 квітня був південний вітер із швидкістю від 7 до 12 м\с в західних регіонах України ця швидкість могла ся-гати до 30 м\с. Щодо 24 квітня, тоді вітер був південно східний з швидкістю до 10 м\с, небо було затягнуте хмарами та очікувались опади. На рисунку 1.10 та 1.11 зображені супутникові знімок станом на 1 та 24 квітня відповідно [9].

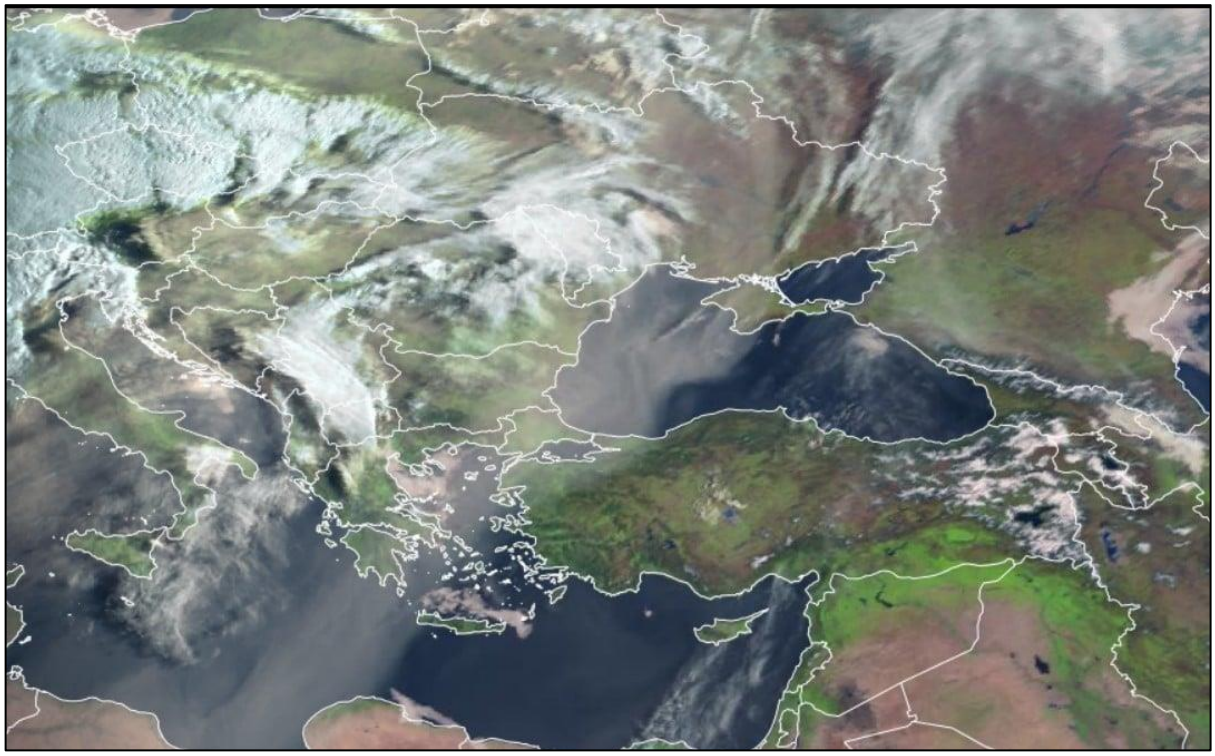


Рисунок 1.10 – Супутниковий знімок станом на 1 квітня 2024 року

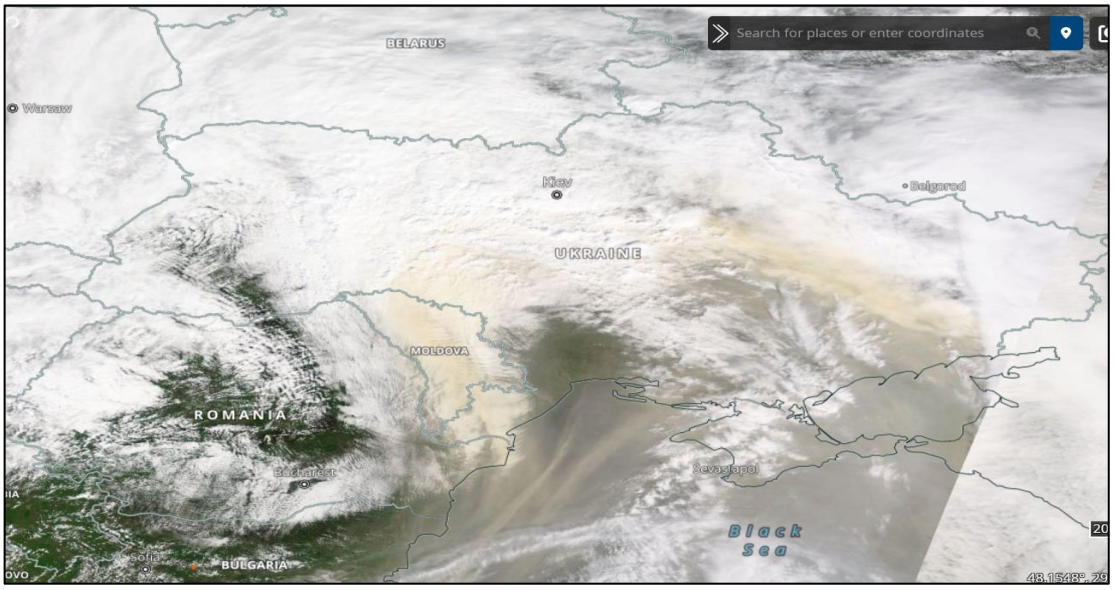


Рисунок 1.11 – Супутниковий знімок станом на 24 квітня 2024 року

На цих знімках видно рух повітряних мас в ті дні та те, звідки приходив пил Сахари на територію України.

1.3 Аналіз моделей та бібліотек Python для прогнозування даних

Мова Python була створена співробітником голландського інституту CWI Гвідо ван Россумом у 1991 році. Python має велику спільноту по всьому світу і випускає нові версії доволі часто.

Python - це універсальна мова програмування високого рівня, яка використовується в широкому спектрі галузей, включаючи науку про дані, машинне навчання, веб-розробку, автоматизацію та багато іншого [10].

Основними характеристиками мови Python є:

– Універсальність: Python можна використовувати для широкого спектру завдань, від простих застосунків та веб-розробки до науки про дані та машинного навчання.

– Простота: Python має лаконічний синтаксис, схожий на англійську мову, що робить його легким для вивчення та читання.

– Читабельність: код добре структурований, використовуються відступи для позначення блоків коду, що спрощує читання.

– Різноманітність бібліотек: Бібліотеки Python — це попередньо написані модулі та функції, що полегшують розробку. Не потрібно створювати кожного разу логіку та алгоритми написанням нового коду, коли можна просто скористатись уже готовими функціями із бібліотек.

– Це інтерпретована мова, що означає, що код не потрібно компілювати перед запуском і це робить його більш гнучким та динамічним.

– Python дозволяє швидко розробляти та писати код, що робить його більш продуктивним для багатьох завдань.

– Python можна використовувати для розробки масштабованих та складних програмних систем.

– Python - це одна з найпопулярніших мов програмування у світі, що робить її більш затребуваною на ринку праці [11].

Ця мова програмування широко використовується у машинному навчанні та роботі з даними. Причинами цього є простота синтаксису та широкий спектр бібліотек для цього, таких як: NumPy, Pandas, Matplotlib, SciPy, Scikit-learn, Plotly.

NumPy [12] вважається найкращою бібліотекою Python для машинного навчання та штучного інтелекту. Це цифрова бібліотека з відкритим вихідним кодом, яку можна використовувати для виконання різних математичних операцій над ма-сивами. NumPy вважається однією з найбільш використовуваних наукових бібліотек, тому багато професіоналів покладаються на неї для аналізу даних.

Масиви NumPy потребують набагато менше місця для зберігання, ніж інші списки Python, і вони швидші та зручніші у використанні. Загалом, NumPy — чудовий варіант для підвищення продуктивності моделей машинного навчання без надто складної роботи [13].

Ось деякі з основних функцій NumPy:

– Високопродуктивний об’єкт N-вимірного масиву;

– Маніпуляція формою;

– Очищення/маніпуляції даними;

– Статистичні операції та лінійна алгебра.

Pandas [14] – це потужна бібліотека Python з відкритим кодом, спеціально розроблена для роботи з табличними даними [14]. Вона базується на фундаменті NumPy, що дозволяє їй ефективно обробляти масиви даних, але пропонує додаткові функціональні можливості для роботи з рядками, стовпцями та таблицями.

Основні структури даних Pandas:

– DataFrame: Основна структура даних Pandas. Вона являє собою двовимірну таблицю з рядками та стовпцями, подібну до електронної таблиці Кожен стовпець може містити дані різного типу (числа, текст, категорії тощо);

– Series: Одновимірний масив даних, схожий на список NumPy, але з мітками для кожного значення. Мітки дозволяють легко отримувати доступ до окремих елементів та виконувати операції над ними [15].

Функціональні можливості Pandas:

– Зчитування та запис даних: Pandas дозволяє легко зчитувати дані з різних джерел, таких як файли CSV, Excel, бази даних SQL та веб-API. Також вона забезпечує зручний запис даних у різні формати;

– Очищення даних: Pandas пропонує інструменти для обробки відсутніх значень, виявлення дублікатів, перетворення типів даних та інші функції для підготовки даних до аналізу;

– Маніпуляція даними: Pandas дозволяє додавати, видаляти, перейменовувати рядки та стовпці, здійснювати об'єднання та розділення таблиць. Ці функції допомагають реструктурувати та перетворити дані для потрібного аналізу;

– Індексація та фільтрація: Pandas надає потужні інструменти для вибірки даних на основі певних умов;

– Агрегація даних: Pandas дозволяє виконувати агрегатні функції, такі як сума, середнє значення, мінімум, максимум тощо, по рядках, стовпцях або цілих таблицях. Це допомагає отримати стислий огляд даних;

– Перевагою використання Pandas є інтеграція з іншими бібліотеками: Pandas чудово інтегрується з іншими популярними бібліотеками Python для аналізу даних, такими як NumPy, Matplotlib, Scikit-learn. Це створює потужний інструментарій для комплексної роботи з даними.

Загалом, Pandas є незамінним інструментом для будь-кого, хто працює з табличними даними в Python.

Matplotlib [16] - це бібліотека Python з відкритим кодом, призначена для створення статичних, анімованих та інтерактивних візуалізацій даних. Вона пропонує спектр інструментів для побудови графіків, діаграм та інших графічних елементів, що допомагає перетворити числові дані у зрозумілий та інформативний формат. Бібліотека підтримує різні види графіків та діаграм, таких як гістограми, графіки, діаграми розсіювання, секторні діаграми, контурні графіки та інші.

Користувач може гнучко налаштувати такі графіки, задавши типи ліній, кольори, маркери, шрифти, осі, масштабування та легенди [17].

Ця бібліотека є незмінним інструментом для візуалізації та аналізу даних, допомагаючи представити дані зрозуміло та зручно.

SciPy [18] це безкоштовна бібліотека з відкритим кодом, яка використовується для вирішення математичних, наукових, інженерних і технічних задач. Вона дозволяє користувачам маніпулювати даними та візуалізувати дані за допомогою широкого спектру високорівневих команд Python. SciPy побудовано на базі NumPy, тому їх зручно використовувати разом [19].

Переваги використання SciPy:

– Широкий спектр функціональних можливостей: SciPy пропонує широкий набір алгоритмів та функцій для вирішення різноманітних наукових та інженерних задач;

– Ефективність: бібліотека використовує оптимізований код для забезпечення високої продуктивності при роботі з великими обсягами даних;

– Інтеграція з NumPy та іншими бібліотеками: SciPy тісно інтегрується з NumPy та іншими науковими бібліотеками Python, створюючи потужний інструментарій для наукових обчислень [20].

Scikit-learn [21] — це безкоштовна бібліотека машинного навчання, написана на Python. Вона надає широкий вибір алгоритмів навчання з учителем і без нього. Одна з основних переваг бібліотеки полягає в тому, що вона працює на основі декількох поширених математичних бібліотек і легко інтегрує їх одна з одною. Ще однією перевагою є велика спільнота користувачів й неабияка документація. Scikit-learn спеціалізується на алгоритмах машинного навчання для вирішення задач навчання з учителем: класифікації й регресії, а також для завдань навчання без учителя: кластеризації, зменшення розмірності й детектування аномалій [22].

Scikit-learn містить реалізації різних алгоритмів, зокрема: лінійна регресія, дерева рішень, методи опорних векторів, наївний Байєс, k-найближчі сусіди, випадкових лісів тощо.

Кожен із цих алгоритмів має унікальні характеристики та параметри, які можна налаштувати для досягнення оптимальних результатів. Scikit-learn також надає корисні інструменти для попередньої обробки даних, включаючи масштабування, видалення відсутніх значень, кодування категоріальних змінних тощо.

Scikit-learn — це потужний інструмент машинного навчання в Python, який є популярним вибором серед науковців і розробників, які займаються аналізом даних завдяки простоті використання, багатим функціональним можливостям і надійності.

Plotly [23] — це потужна бібліотека Python з відкритим кодом, яка дозволяє створювати інтерактивні веб-візуалізації.

На відміну від Matplotlib, який зосереджується на статичних діаграмах, Plotly дозволяє створювати діаграми та графіки, з якими користувачі можуть взаємодіяти. Ця інтерактивність робить Plotly цінним інструментом для дослідження даних, комунікації та інформаційних панелей моніторингу даних.

Основні функції Plotly:

– Інтерактивні діаграми: Ви можете масштабувати, панорамувати та переміщати мишу на діаграмах Plotly, щоб переглянути додаткову інформацію про ваші точки даних. Це дозволяє користувачам глибше копати дані та виявляти закономірності;

– Широкий вибір типів діаграм : Plotly підтримує велику колекцію типів діаграм, включаючи лінійні діаграми, точкові діаграми, стовпчасті діаграми, теплові карти, 3D візуалізації тощо;

– Налаштування: Plotly пропонує багато варіантів налаштування кольорів, маркерів, легенд, макета та анімації. Налаштуйте візуалізацію відповідно до ваших потреб і брендингу.

Plotly - це цінне доповнення до інструментарію будь-якого науковця з даних. Його інтерактивні функції та широкі можливості налаштування роблять його ідеальним для створення проникливих та цікавих візуалізацій даних.

Для прогнозування було обрано такі моделі як FBProphet та ARIMA.

FBProphet, розроблена основною командою Facebook з вивчення даних, являє собою бібліотеку Python з відкритим вихідним кодом, спеціально призначену для прогнозування часових рядів [24].

Її особливості включають:

– Простота: FBProphet має чіткий і лаконічний синтаксис, що робить її простою у вивченні та використанні навіть для недосвідчених фахівців з роботи з даними;

– Автоматизація: автоматизує багато аспектів процесу прогнозування, такі як вибір моделі, налаштування гіперпараметрів і визначення сезонності. Це знижує необхідність ручного втручання й оптимізує робочі процеси;

– Робота з сезонністю: FBProphet відмінно справляється з виявленням сезонності в часових рядах, наприклад, тижневих, місячних і річних трендів. Ці закономірності можуть бути включені в прогнози, що призводить до більш точних результатів, особливо для даних, що демонструють сезонні коливання;

– Вплив свят: можна врахувати вплив державних свят на часові ряди. Включивши святкові дні як додаткові змінні регресії, FBProphet може скоригувати прогнози з урахуванням можливих збоїв або стрибків у даних.

Наступний рисунок 1.12 ілюструє процес прогнозування Prophet [25].

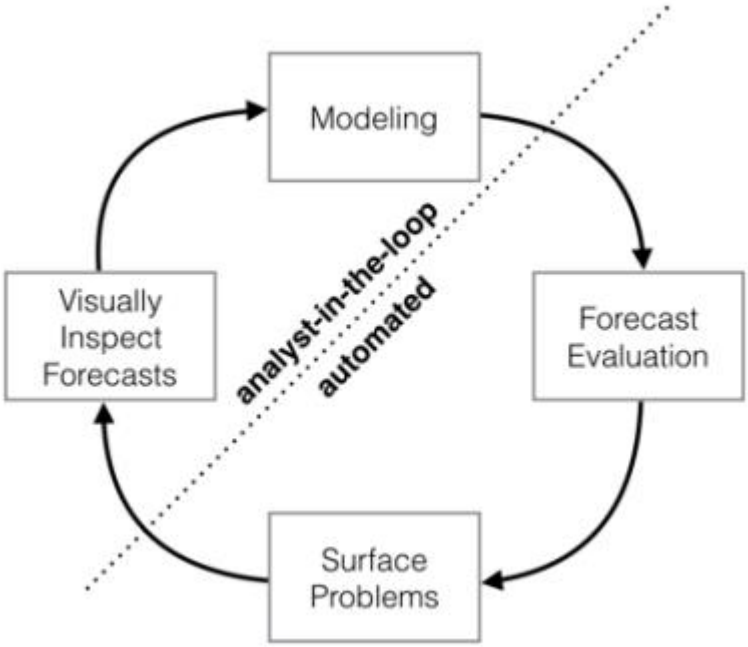


Рисунок 1.12 - Діаграма процесу прогнозування Prophet

Ця діаграма показує цикл розробки та вдосконалення моделі, який складається з чотирьох основних етапів:

– Create/Update Model (Створити/Оновити Модель): на цьому етапі створюється нова модель або вдосконалюється існуюча. Це включає в себе вибір алгоритму, налаштування гіперпараметрів, навчання моделі на тренувальних даних тощо;

– Evaluate Model (Оцінити Модель): після створення або оновлення моделі її необхідно оцінити. Це включає в себе використання валідаційних або тестових даних для оцінки продуктивності моделі. Показники оцінки можуть включати точність, коефіцієнт детермінації (R^2), середню абсолютну похибку (MAPE), корінь середньоквадратичної похибки (RMSE) тощо;

– Surface Problems (Виявлення Проблем): на цьому етапі аналізуються результати оцінки моделі, щоб виявити можливі проблеми або слабкі місця моделі. Це може включати в себе аналіз похибок, виявлення аномалій або виявлення областей, де модель працює недостатньо добре;

– Visually Inspect Model (Візуальна Інспекція Моделі): візуалізація результатів моделі допомагає глибше зрозуміти її поведінку та виявити можливі проблеми. Це може включати в себе побудову графіків фактичних і прогнозованих значень, аналіз залишків, побудову діаграм важливості ознак тощо.

ARIMA — це модель, яка прогнозує майбутні тенденції на основі даних часових рядів. Ця модель є різновидом регресійного аналізу [26].

– AR (Авторегресія): модель, яка показує, що змінна повертається до свого власного відстаючого/попереднього значення.

– I (Інтеграція): Різниця необроблених спостережень, щоб зробити часовий ряд стаціонарним

– MA (ковзне середнє): Залежність між спостереженнями та залишковими помилками для моделей ковзного середнього Для моделей ARIMA, стандарт Позначення: ARIMA з p, d і q.

Ціле значення замінює параметр, що вказує на тип використовуваної моделі ARIMA.

– P: Кількість відкладених спостережень у моделі. Також називається розпорядженням про відстрочку.

– D: у рази більше частоти, з якою необроблені спостереження відрізняються. Також називається ступенем диференціації.

– Q: Розмір вікна ковзного середнього. Також називається ковзним середнім порядком.

1.4 Висновки

У цьому розділі проведено аналіз проблематики предметної області, а саме проблеми забруднення повітря пилу Сахари в Україні. Розглянуто основні сервіси та технології, що використовуються для моніторингу якості повітря, а саме SaveEcoBot та EcoCity. Розглянуто моделі прогнозування часових рядів та

бібліотеки Python для роботи з даними, які будуть використовуватись у подальшій роботі, а саме FBProphet та ARIMA.

2 ПІДГОТОВКА ТА РОЗВІДУВАЛЬНИЙ АНАЛІЗ ДАНИХ

2.1 Розвідувальний аналіз даних

Для виконання задачі було зібрано дані з сервісу EcoCity та сформовано відповідний датасет Air Quality Monitoring from EcoCity та ноутбук в Kaggle [27, 28]. Датасет містить в собі такі значення:

– id_station – ідентифікаційний код станції на сайті EcoCity;

– location – координати відповідної станції моніторингу якості повітря;

– date – дата, коли дані були зібрані;

– time_point - індекс або множник, який допомагає обчислити точний час у певний день;

– indicator – назва індексу концентрації забруднення;

– dimension – міра величини забруднення;

– value – числове значення забруднення.

Сформовано інший датасет з інформацією про кожну станцію Вінницької області. Він буде використаний для співставлення станції із її адресою для більш чіткого розуміння, яка саме станція досліджується.

Зчитаємо дані із CSV файлів та додамо адреси до наявних станцій (рис 2.1)

```
column_names = ['id_station', 'location', 'date', 'time_point', 'Column6', 'indicator', 'dimension', 'value']
stations = pd.read_csv('/kaggle/input/air-quality-monitoring-from-ecocity/ECOCITY_Archive_651_561_2024-03-25_2024-05-10.csv', sep=',', names=column_names)

# Визначення назв колонок для about_station
column_names_a = ["id_saveecobot", "id_ecocity", "network", "locality", "address", "start_date", "lat", "lng", "notes", "source"]
about_stations = pd.read_csv('/kaggle/input/eco-city-stations-about/ecocity_about_stations_2024.csv', header=None, names=column_names_a)

stations['id_station'] = pd.to_numeric(stations['id_station'], errors='coerce').dropna().astype(int)
about_stations['id_ecocity'] = pd.to_numeric(about_stations['id_ecocity'], errors='coerce').dropna().astype(int)
# Перевірка зчитаних даних
print(stations)
stations.info()

result = pd.merge(stations, about_stations[['id_ecocity', 'address']], left_on='id_station', right_on='id_ecocity', how='left')

result = result.drop(columns=['id_ecocity'])

result
```

Рисунок 2.1 – Зчитування даних

Переглянемо наявні дані на рисунку 2.2 та інформацію про них на рисунку 2.3.

	id_station	location	date	time_point	Column6	\
0	1315	49.23327 28.409161	2024-03-25	1	18	
1	1315	49.23327 28.409161	2024-03-25	2	20	
2	1315	49.23327 28.409161	2024-03-25	3	18	
3	1315	49.23327 28.409161	2024-03-25	4	20	
4	1315	49.23327 28.409161	2024-03-25	5	20	
...	...	...	...	...	...	
19642	1876	49.2048561 28.5288355	2024-05-10	68	20	
19643	1876	49.2048561 28.5288355	2024-05-10	69	20	
19644	1876	49.2048561 28.5288355	2024-05-10	70	20	
19645	1876	49.2048561 28.5288355	2024-05-10	71	20	
19646	1876	49.2048561 28.5288355	2024-05-10	72	20	

	indicator	dimension	value
0	PM1.0	ug/m3	26.8422
1	PM1.0	ug/m3	25.4215
2	PM1.0	ug/m3	24.4250
3	PM1.0	ug/m3	22.8410
4	PM1.0	ug/m3	22.6225
...	...	...	...
19642	PM1.0	ug/m3	6.3255
19643	PM1.0	ug/m3	5.8295
19644	PM1.0	ug/m3	3.7260
19645	PM1.0	ug/m3	3.9925
19646	PM1.0	ug/m3	4.0595

Рисунок 2.2 – Наявні дані

```
[19647 rows x 8 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19647 entries, 0 to 19646
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   id_station  19647 non-null  int64
 1   location    19647 non-null  object
 2   date        19647 non-null  object
 3   time_point  19647 non-null  int64
 4   Column6     19647 non-null  int64
 5   indicator   19647 non-null  object
 6   dimension   19647 non-null  object
 7   value       19647 non-null  float64
dtypes: float64(1), int64(3), object(4)
memory usage: 1.2+ MB
```

Рисунок 2.3 – Інформація про дані

Згідно інформації про дані, пропущених записів немає.

Після цього додавання колонки адреси до наявних даних отримуємо таблицю, зображену на рисунку 2.4

[27]:

	id_station	location	date	time_point	Column6	indicator	dimension	value	address
0	1315	49.23327 28.409161	2024-03-25	1	18	PM1.0	ug/m3	26.8422	Vinnytsia, str. Khmelnytsky shose, 95, VNTU, b...
1	1315	49.23327 28.409161	2024-03-25	2	20	PM1.0	ug/m3	25.4215	Vinnytsia, str. Khmelnytsky shose, 95, VNTU, b...
2	1315	49.23327 28.409161	2024-03-25	3	18	PM1.0	ug/m3	24.4250	Vinnytsia, str. Khmelnytsky shose, 95, VNTU, b...
3	1315	49.23327 28.409161	2024-03-25	4	20	PM1.0	ug/m3	22.8410	Vinnytsia, str. Khmelnytsky shose, 95, VNTU, b...
4	1315	49.23327 28.409161	2024-03-25	5	20	PM1.0	ug/m3	22.6225	Vinnytsia, str. Khmelnytsky shose, 95, VNTU, b...
...	...	...	...	...	...	...	...	...	...
19642	1876	49.2048561 28.5288355	2024-05-10	68	20	PM1.0	ug/m3	6.3255	Prov. Bohdana Khmel'nyts'koho, Vinnyts'ki Khutoru
19643	1876	49.2048561 28.5288355	2024-05-10	69	20	PM1.0	ug/m3	5.8295	Prov. Bohdana Khmel'nyts'koho, Vinnyts'ki Khutoru
19644	1876	49.2048561 28.5288355	2024-05-10	70	20	PM1.0	ug/m3	3.7260	Prov. Bohdana Khmel'nyts'koho, Vinnyts'ki Khutoru
19645	1876	49.2048561 28.5288355	2024-05-10	71	20	PM1.0	ug/m3	3.9925	Prov. Bohdana Khmel'nyts'koho, Vinnyts'ki Khutoru
19646	1876	49.2048561 28.5288355	2024-05-10	72	20	PM1.0	ug/m3	4.0595	Prov. Bohdana Khmel'nyts'koho, Vinnyts'ki Khutoru

19647 rows × 9 columns

Рисунок 2.4 – Вигляд таблиці із адресою

Визначимо які станції є в датасеті та дізнаємось їх Id (рис. 3.5).

In [3]:

```
#цей фрагмент коду зручності та розуміння, які станції наявні
unique_id_count = stations['id_station'].nunique()
print("Кількість унікальних значень в стовбці id_station (кількість станцій):", unique_id_count)
unique_stations = stations['id_station'].unique()
print("Id унікальних станцій")
print(unique_stations)
```

```
Кількість унікальних значень в стовбці id_station (кількість станцій): 6
Id унікальних станцій
[1315 1612 1769 1864 1872 1876]
```

Рисунок 2.5 – Визначення унікальних станцій

За допомогою функції обробки даних з певної станції побудуємо (рисунок 2.6) графіки та первинно виявимо аномалії в ряді. Ця функція також перетворює колонки date, time_point в одну datetime; для кожної станції в словнику datasets будує графік значень value по датах (datetime). Потім визначає порогове значення (95% від

максимального значення value) і знаходить аномалії - значення, які перевищують цей поріг. Виводить аномалії на екран і зберігає їх у словнику anomalies_datasets.

```
[29]:
def get_data(id_station):
    if id_station in result['id_station'].unique():
        anomalies_datasets = {}
        station_data = result[result['id_station'] == id_station].copy()
        station_data['datetime'] = pd.to_datetime(station_data['date']) + pd.to_timedelta((station_data['time_point'] - 1) * 20, unit='m')
        station_data['datetime'] = pd.to_datetime(station_data['datetime'])
        station_data.set_index('datetime', inplace=True)
        address = station_data['address'].iloc[0]  # Отримання адреси для станції
        datasets = {id_station: station_data[['id_station', 'value']]}  # Вибір потрібних колонок без 'datetime'
        for station_id, dataset in datasets.items():
            plt.figure(figsize=(12, 6))
            plt.plot(dataset.index, dataset['value'], label=f'station_{station_id}')
            plt.xlabel('Date')
            plt.ylabel('Dimension (PM1)')
            plt.title(f'PM1 Dimension Over Time for station_{station_id}, {address}')
            plt.legend()
            plt.show()

            threshold = 0.95 * dataset['value'].max()
            anomalies = dataset[dataset['value'] > threshold]
            print(f'Anomalies for station_{station_id}:')
            print(anomalies)
            anomalies_datasets[station_id] = anomalies
        df_station = datasets[id_station].reset_index()
        df_anomalies_station = anomalies_datasets[id_station].reset_index()
        return df_station, df_anomalies_station
```

Рисунок 2.6 – Функція get_data

Виклик функції зображено на рисунку 2.7.

```
# Приклад виклику функції
df_station, df_anomalies_station = get_data(1315)
print("="*40)
print("Dataset for the selected station:")
print(df_station)
print("="*40)
print("Anomalies for the selected station:")
print(df_anomalies_station)
```

Рисунок 2.7 – Виклик функції

На рисунку 2.8 зображений графік за даними станції ВНТУ. На ньому видно піки аномалій, у тому числі й у вище згадані дати, а саме 30, 31 березня, 24-26 квітня. Інші пікові точки можуть бути пов’язані із другими причинами забруднення повітря або ж і з тим, що пил Сахари продовжив «подорожувати» територією України.

На рисунках 2.9-2.13 зображені результати по іншим станціям, де ця тенденція зберігається, проте й вплив інших факторів на них чітко видно.

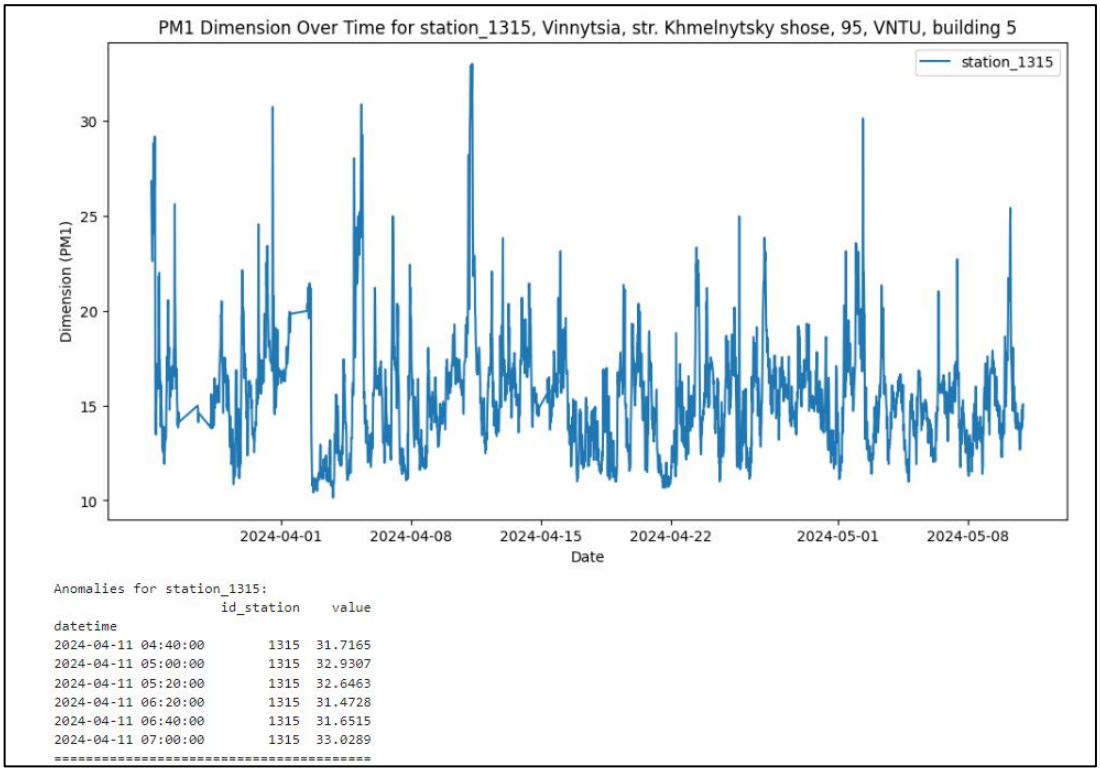


Рисунок 2.8 – Графік даних зі списком дат аномалій станції 1315

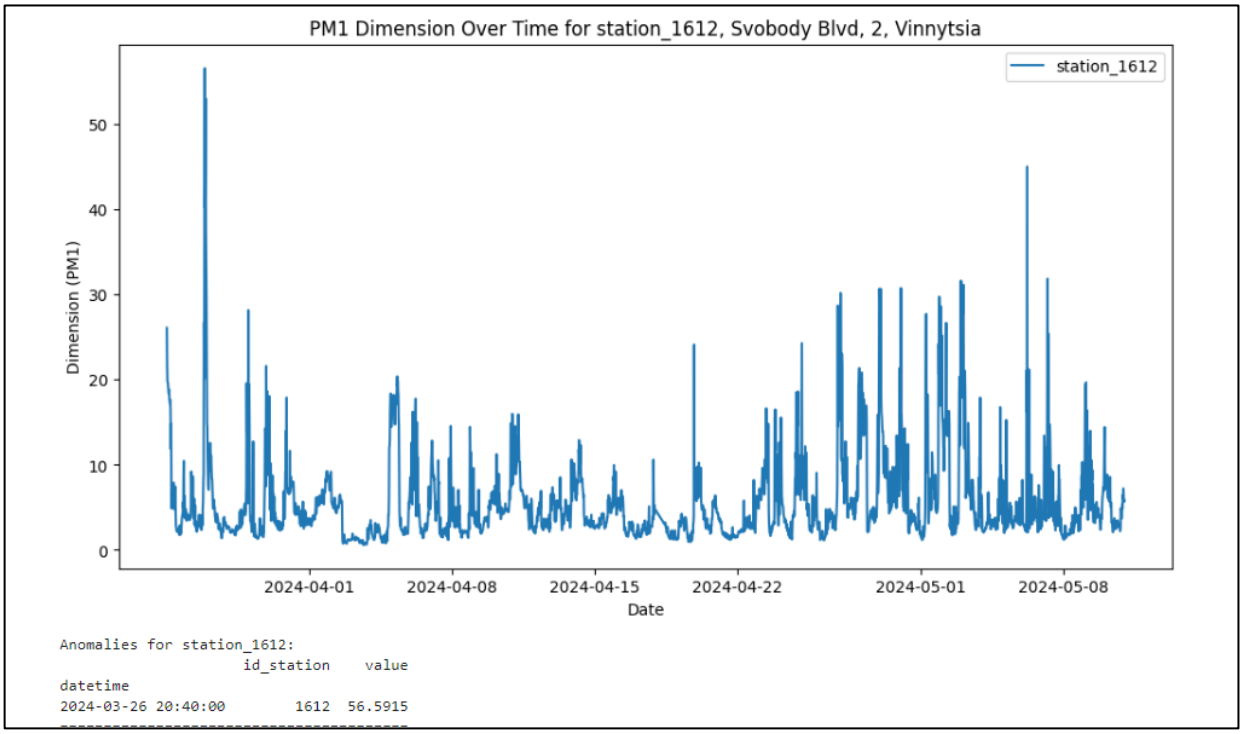
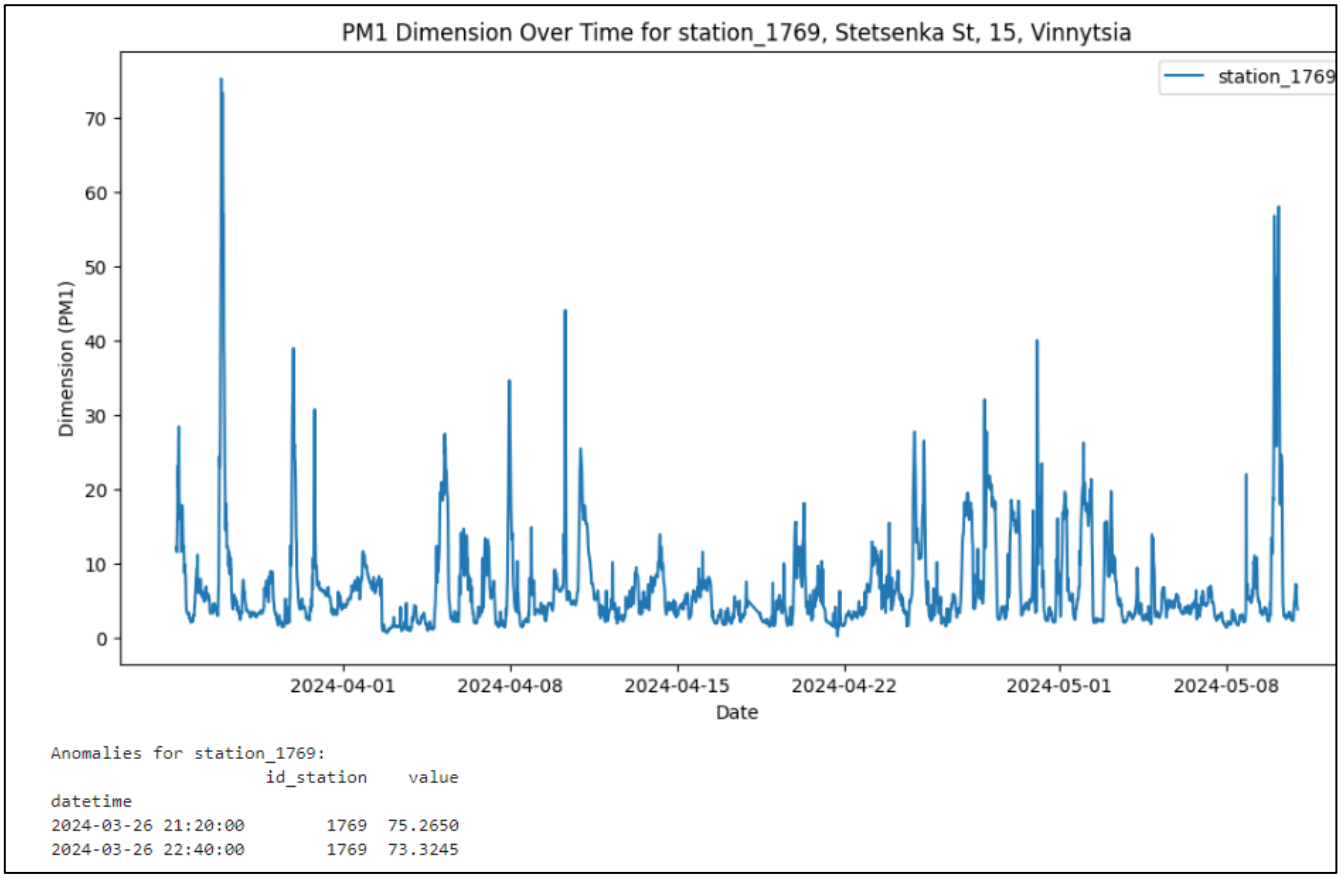


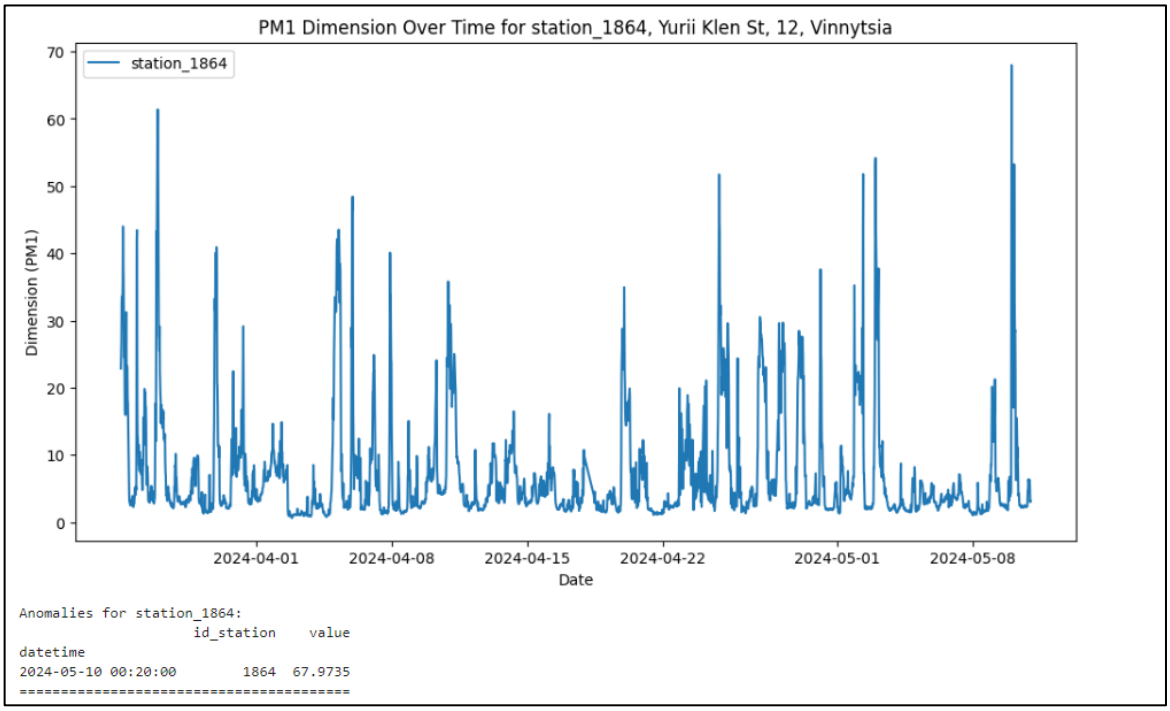
Рисунок 2.9 – Графік даних зі списком дат аномалій станції 1612



Anomalies for station_1769:

datetime	id_station	value
2024-03-26 21:20:00	1769	75.2650
2024-03-26 22:40:00	1769	73.3245

Рисунок 2.10 – Графік даних зі списком дат аномалій станції 1769



Anomalies for station_1864:

datetime	id_station	value
2024-05-10 00:20:00	1864	67.9735

Рисунок 2.11 – Графік даних зі списком дат аномалій станції 1864

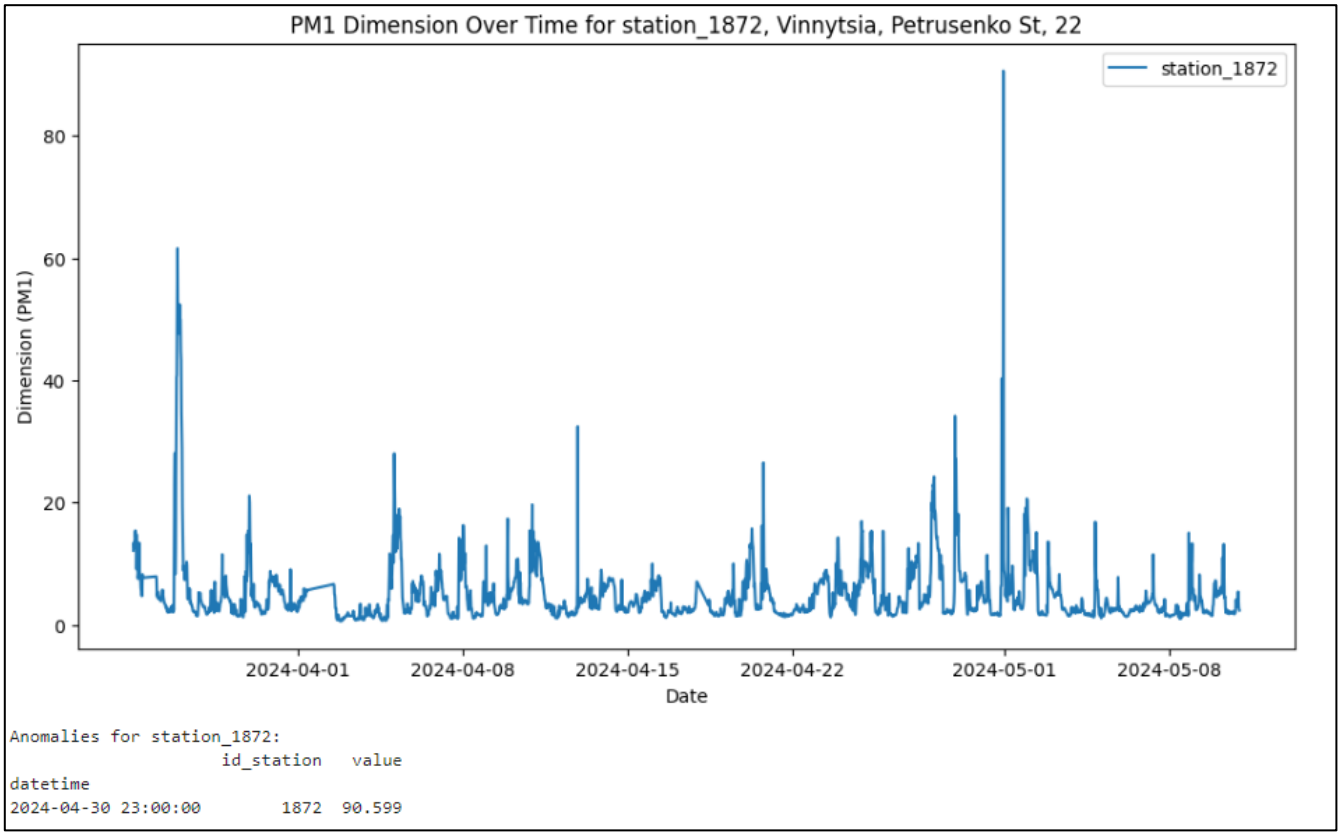


Рисунок 2.12 – Графік даних зі списком дат аномалій станції 1872

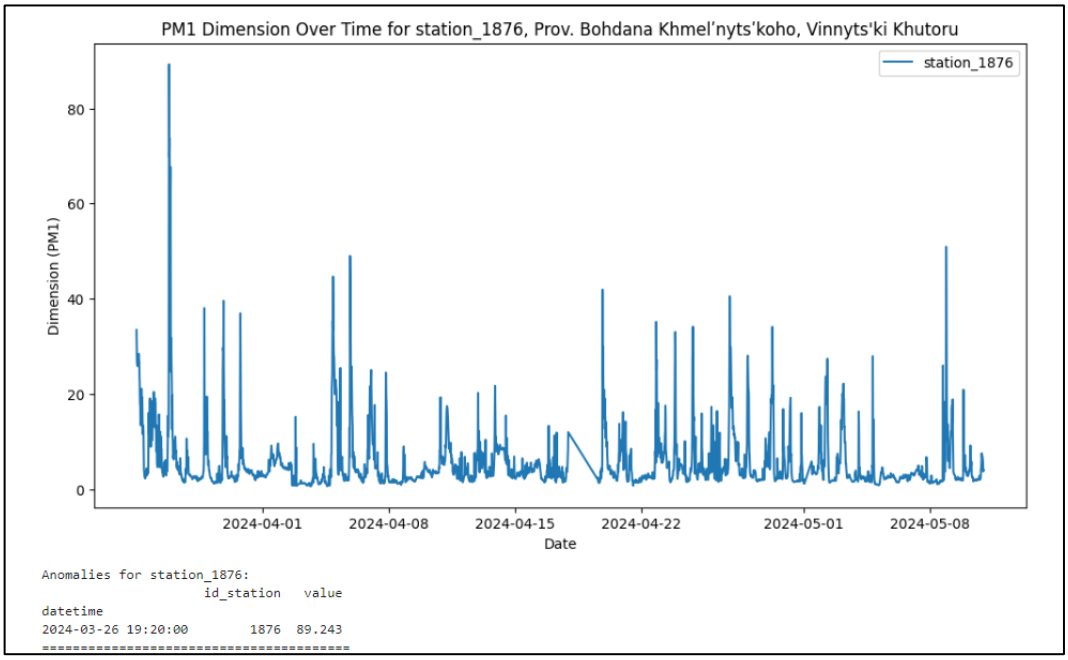


Рисунок 2.13 – Графік даних зі списком дат аномалій станції 1876

2.2 Просторовий аналіз та візуалізація даних

Розроблено ноутбук Sahara's Dust in the Region - 2D Analysis [29], який допомагає візуалізувати дані по конкретній годині та даті. Для створення графіків використовуються дані із датасету Air Quality Monitoring from EcoCity [27].

Як було видно з рисунку 2.2 датасет не має відокремлених координат, тому написано функцію для виділення lat, lng, фрагмент коду зображено на рисунку 2.14.

```
def extract_location_data(df, id_station=None):
    if id_station is not None:
        station_data = df[df['id_station'] == id_station].copy()
        if not station_data.empty:
            # Об'єднання колонок 'date' і 'time_point' у нову колонку 'datetime'
            station_data['datetime'] = pd.to_datetime(station_data['date']) + pd.to_timedelta((sta
tion_data['time_point'] - 1) * 20, unit='m')
            station_data[['lat', 'lng']] = station_data['location'].str.split(expand=True)
            # Видалення колонок 'location', 'date' та 'time_point'
            return station_data.drop(columns=['location', 'date', 'time_point'])
        else:
            print(f"Station ID {id_station} does not exist in the dataset.")
            return pd.DataFrame()
    else:
        df[['lat', 'lng']] = df['location'].str.split(expand=True)
        # Об'єднання колонок 'date' і 'time_point' у нову колонку 'datetime'
        df['datetime'] = pd.to_datetime(df['date']) + pd.to_timedelta((df['time_point'] - 1) * 20,
unit='m')
        # Видалення колонок 'location', 'date' та 'time_point'
        return df.drop(columns=['location', 'date', 'time_point'])
```

Рисунок 2.14 – Функція для розділення координат

Сформуємо новий датафрейм для подальшої роботи з ним в рамках цього ноутбуку, виокремивши тільки потрібні поля саме для просторового аналізу (рис. 2.15)

```
df = data[['id_station', 'datetime', 'value', 'lat', 'lng']]
df.columns = ['id_station', 'ds', 'value', 'lat', 'lng']
df['network'] = 'Eco-City'
df
```

	id_station	ds	value	lat	lng	network
0	1315	2024-03-25 00:00:00	26.8422	49.23327	28.409161	Eco-City
1	1315	2024-03-25 00:20:00	25.4215	49.23327	28.409161	Eco-City
2	1315	2024-03-25 00:40:00	24.4250	49.23327	28.409161	Eco-City
3	1315	2024-03-25 01:00:00	22.8410	49.23327	28.409161	Eco-City
4	1315	2024-03-25 01:20:00	22.6225	49.23327	28.409161	Eco-City
...	...	...	...	...	...	...
19642	1876	2024-05-10 22:20:00	6.3255	49.2048561	28.5288355	Eco-City
19643	1876	2024-05-10 22:40:00	5.8295	49.2048561	28.5288355	Eco-City
19644	1876	2024-05-10 23:00:00	3.7260	49.2048561	28.5288355	Eco-City
19645	1876	2024-05-10 23:20:00	3.9925	49.2048561	28.5288355	Eco-City
19646	1876	2024-05-10 23:40:00	4.0595	49.2048561	28.5288355	Eco-City

Рисунок 2.15 – Створення нового датафрейму

Для того щоб задати час візуалізації використовується стрічка коду datetime_analysis =’ ‘

По заданому часу ми можемо виокремити показники PM_1 з кожної станції (рис 2.16).

```
# Selection data for interpolation
data = df[df['ds']==datetime.datetime.fromisoformat(datetime_analysis)].reset_index(drop=True)
x = data.lng.values
y = data.lat.values
z = data.value.values
fig = plt.figure()
plt.scatter(x, y)
plt.title(f'Stations in Vinnytsia region with data for {indicator_name} in {datetime_analysis}')
display(data)
plt.show()
```

	id_station	ds	value	lat	lng	network
0	1315	2024-03-31	16.8568	49.23327	28.409161	Eco-City
1	1612	2024-03-31	6.5630	49.2177336	28.4497946	Eco-City
2	1769	2024-03-31	6.7130	49.2436876	28.4964322	Eco-City
3	1864	2024-03-31	8.3815	49.2368871	28.5133055	Eco-City
4	1872	2024-03-31	6.4255	49.2419269	28.4620209	Eco-City
5	1876	2024-03-31	4.5600	49.2048561	28.5288355	Eco-City

Рисунок 2.16 – Датафрейм в точно заданий час

Наступним кроком є інтерполяція даних. Інтерполяція даних для просторового аналізу — це процес оцінки значень у точках, де дані не були виміряні, на основі відомих значень у сусідніх точках. Це дозволяє створювати гладкі карти та поверхні, що відображають просторовий розподіл певних характеристик, у нашому випадку – це рівень забруднення. На рисунку 2.17 показаний фрагмент коду інтерполяції даних.

```
data['lng'] = data['lng'].astype(float)
data['lat'] = data['lat'].astype(float)
```

```
# Data interpolation
x = x.astype(float)
y = y.astype(float)
f = interp2d(x, y, z, kind='linear')  # 'linear', 'cubic', 'quintic'
f
```

```
<scipy.interpolate._interpolate.interp2d at 0x790156f17700>
```

```
# Calculation of values for a regular network of points
X = np.linspace(data.lng.min()*0.99995, data.lng.max()*1.00005, 100)
Y = np.linspace(data.lat.min()*0.99995, data.lat.max()*1.00005, 100)
Z = f(X, Y)
Z[0]
```

Рисунок 2.17 – Інтерполяція даних

За побудову графіків відповідає фрагмент коду наведений на рисунку 2.18.

```
# Coordinates of stations - from EcoCity or no
xeco = data['lng'].values
yeco = data['lat'].values
numeco = data['id_station'].astype('str').values


# Visualization
fig = plt.figure(figsize=(12,10))
plt.contourf(X, Y, Z)

plt.scatter(xeco, yeco, c='k', s=100, label='EcoCity')
for i in range(len(xeco)):
    plt.annotate("  "+numeco[i], xy=(xeco[i], yeco[i]), textcoords='data')

plt.axis()
plt.title(f'Stations in Vinnytsia region with hourly average data for {indicator_name} in {datetim
e_analysis} (maximum value = {round(data.value.max(),2)})')
plt.colorbar()
plt.legend(loc='best')
plt.grid()
plt.show()
```

Рисунок 2.18 – Фрагмент коду, який відповідає за побудову графіків

Прикладами роботи ноутбуку є графіки на рисунках 2.19 та 2.20.

– Кольорові області: Графік показує різні рівні показника якості повітря (PM_1) у вигляді кольорових областей. Кожен колір відповідає певному діапазону значень цього показника;

– Контурні лінії: Лінії, що розділяють області різних кольорів, показують межі між різними рівнями значень.

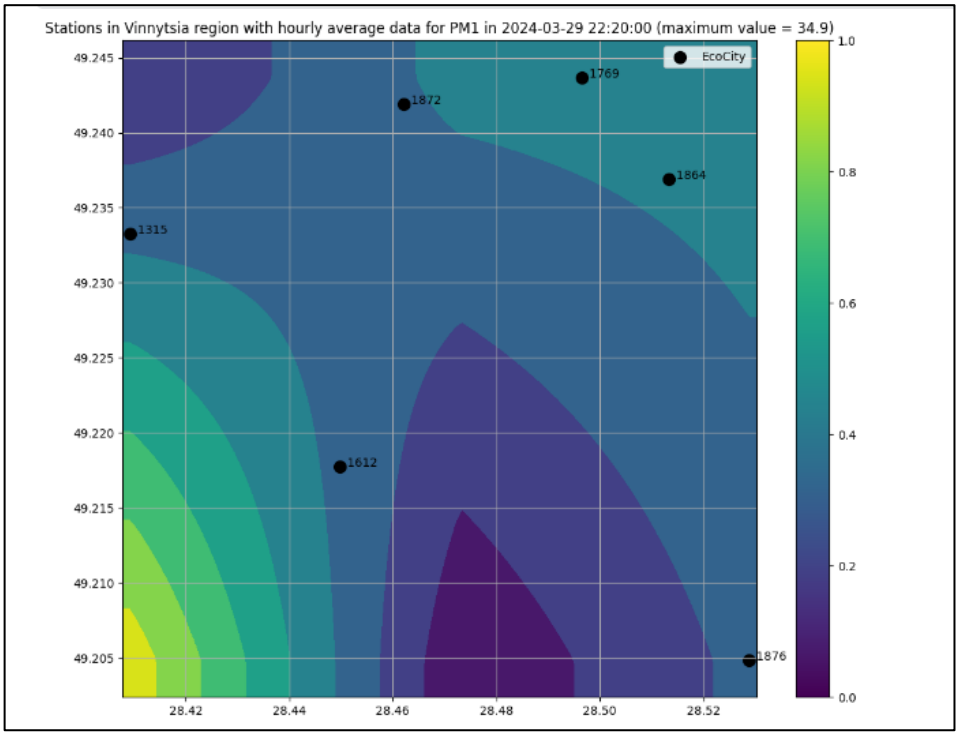


Рисунок 2.19 – Просторовий аналіз даних станом на 22:20 29-03-2024

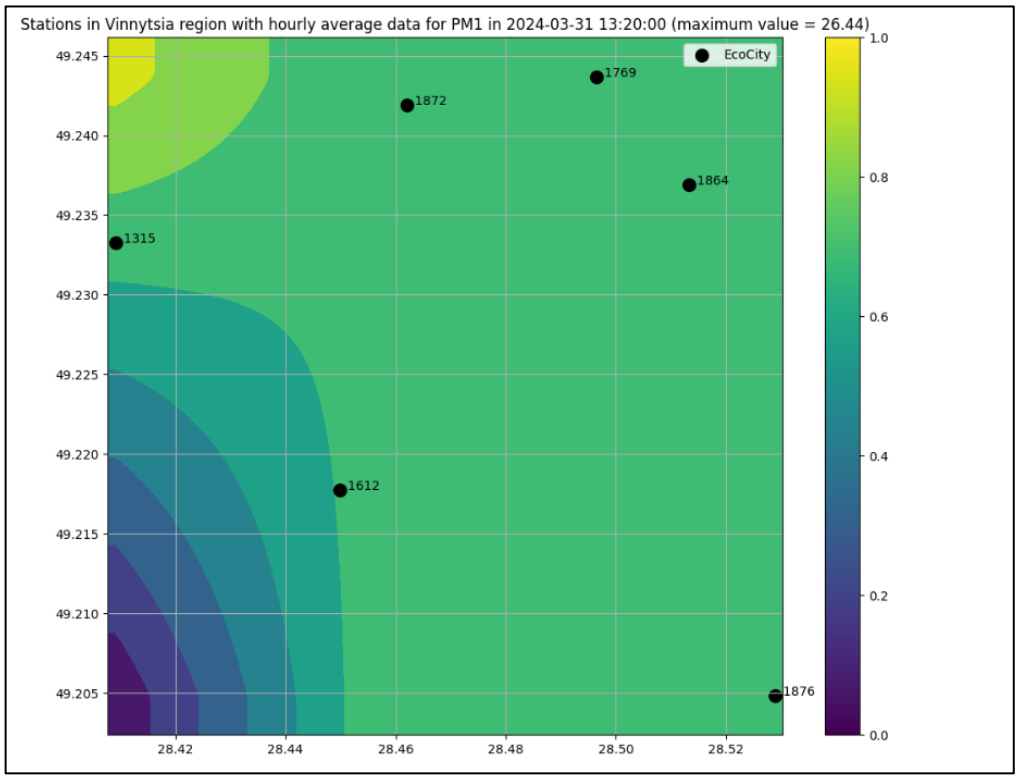


Рисунок 2.20 – Просторовий аналіз даних станом на 13:20 31-03-2024

Графік дозволяє легко і наочно зрозуміти просторовий розподіл показника якості повітря (PM_1) у Вінницькій області на конкретний час, допомагаючи виявити проблемні зони та проводити подальший аналіз.

2.3 Висновки

У другому розділі проведено підготовку та розвідувальний аналіз даних для вирішення задачі прогнозування поширення пилу Сахари в Україні. На основі зібраних даних із сервісу EcoCity створено відповідні датасети, що містять інформацію про концентрацію забруднення повітря.

Зібрано дані із сервісу EcoCity, що включають інформацію про станції моніторингу якості повітря, їх координати, дати та час збору даних, індекси концентрації забруднення, міри величини забруднення та числові значення забруднення.

Окремо створено датасет з інформацією про кожну станцію Вінницької області, що дозволяє точно співвідносити станції з їх адресами.

Розроблено ноутбук для візуалізації даних по конкретних годинах та датах, що допомагає зрозуміти просторовий розподіл показника якості повітря (PM1) у Вінницькій області.

Графіки, створені на основі цих даних, дозволяють виявити проблемні зони та проводити подальший аналіз, забезпечуючи наочне представлення рівнів забруднення на різний час доби.

Ці результати закладають основу для подальших кроків у розробці інформаційної технології прогнозування та дозволяють більш глибоко зрозуміти та аналізувати проблему забруднення повітря пилом Сахари в Україні.

3. РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ТА ПРОГНОЗУВАННЯ

3.1 Розробка інформаційної технології

Для розробки інформаційної технології прогнозування поширення пилу Сахари в Україні пропонується такий алгоритм (рис. 3.1).

1. Визначення цільової ознаки, яку необхідно прогнозувати;
2. Аналіз даних для виявлення закономірностей та трендів;
3. Виявлення та фільтрація аномальних даних для покращення якості моделі;
4. Створення тренувального, валідаційного та тестового наборів для різних типів моделей;
5. Налаштування та тюнінг моделей
6. Тренування моделей на тренувальному датасеті та оцінка їх продуктивності на валідаційному датасеті;
7. Прогнозування значень на тестовому датасеті та оцінка точності прогнозів;
8. Аналіз результатів та вибір оптимальної моделі, яка найкраще відповідає вимогам точності та швидкодії, якщо моделі не отримали задовільних оцінок точності, то налаштовуємо модель знову;

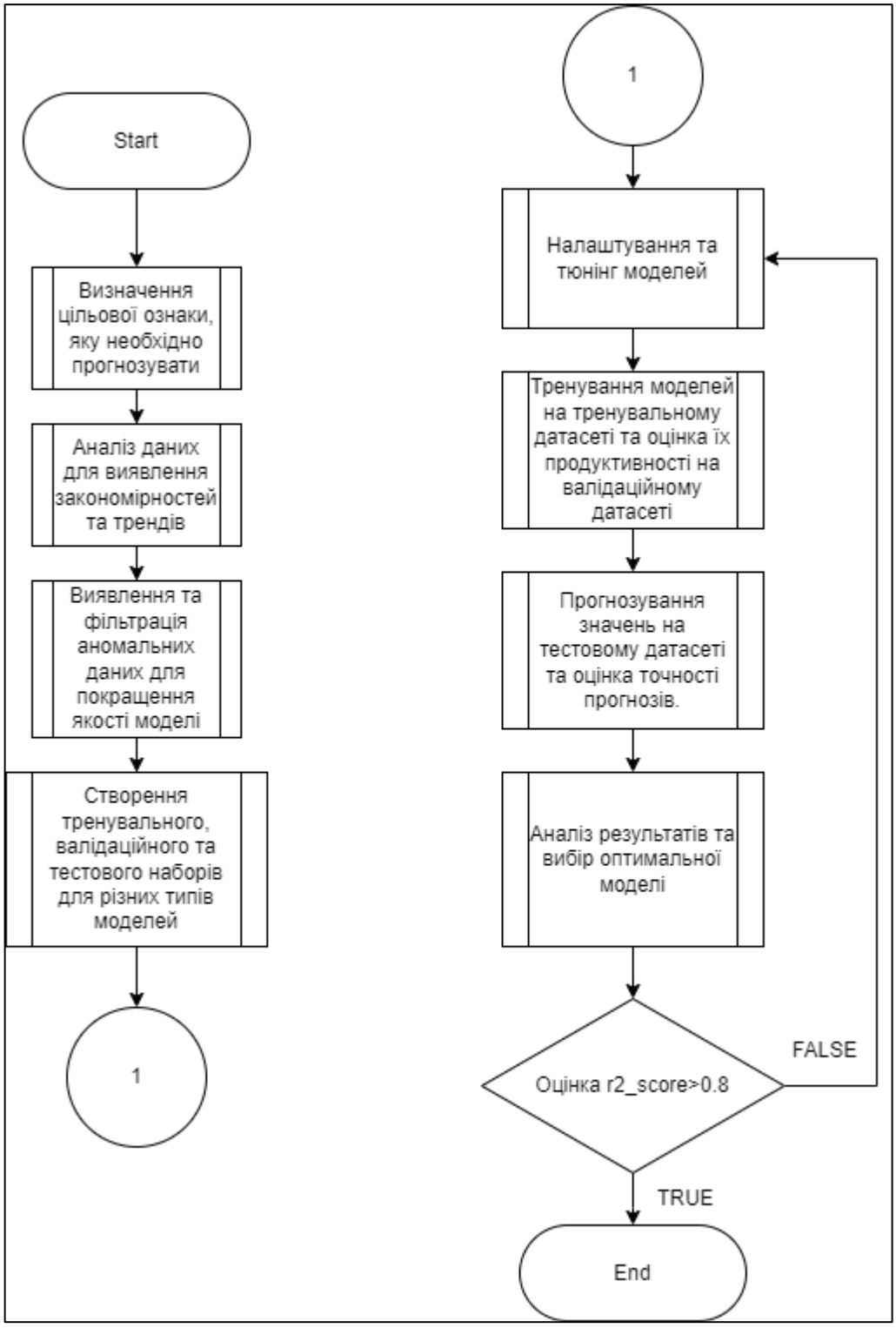


Рисунок 3.1 – Блок-схема алгоритму інформаційної технології аналізу та прогнозування поширення пилу Сахари Україною

3.2 Прогнозування поширення пилу Сахари

Виконаємо прогнозування по даним станції ВНТУ 1315 за допомогою моделей описаних у першому розділі, а саме ARIMA та Facebook Prophet. Перевіримо ряд на стаціонарність.

На рисунку 3.2 зображений фрагмент коду, який відповідає за імпорт потрібних бібліотек.

```
# Import libraries
import random
import os
import numpy as np
import pandas as pd
import requests

# Date
import datetime as dt
from datetime import date, timedelta, datetime

# EDA
import matplotlib.pyplot as plt
from matplotlib.pylab import rcParams
import plotly.express as px
import plotly.graph_objects as go
from plotly.offline import init_notebook_mode
init_notebook_mode(connected=True)

# Time Series - EDA and Modelling
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima_model import ARIMA

# Metrics
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error

# Modeling and preprocessing
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from prophet import Prophet
```

Рисунок 3.2 – Імпорт бібліотек

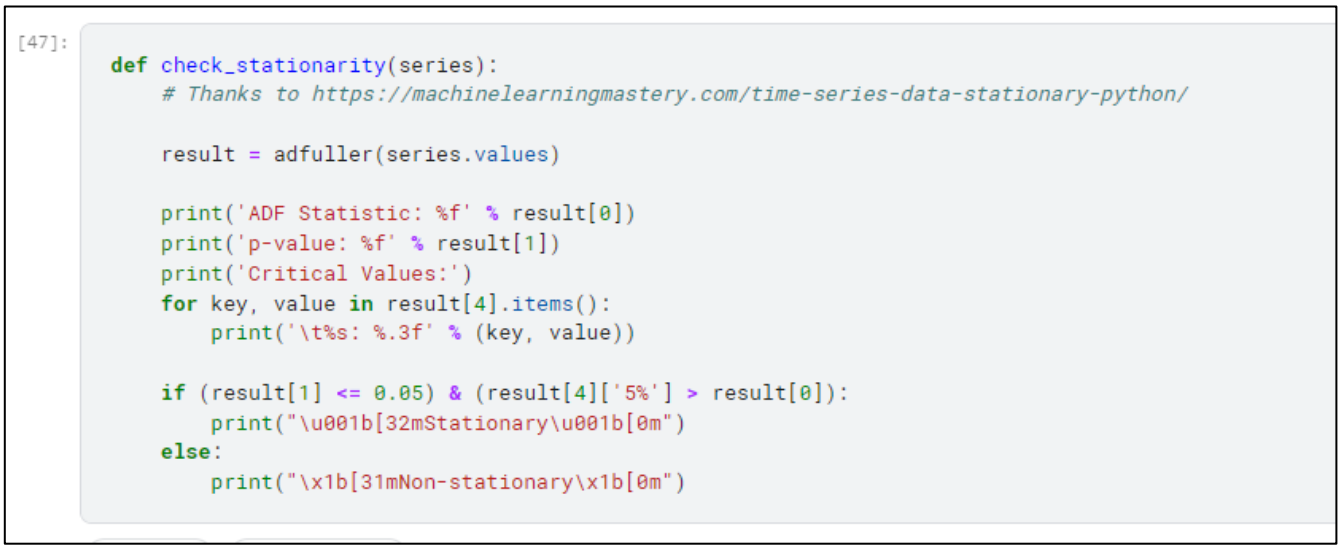
Перевірка стаціонарності часового ряду є важливим кроком у процесі аналізу та моделювання даних з кількох причин [30]. Стаціонарність означає, що статистичні властивості часового ряду, такі як середнє, дисперсія і автокореляція, залишаються постійними з часом. Ось чому перевірка стаціонарності є важливою:

Багато моделей часового ряду, таких як ARIMA (AutoRegressive Integrated Moving Average), припускають, що ряд є стаціонарним. Якщо час ряд не є стаціонарним, результати моделювання можуть бути ненадійними або навіть хибними.

Стаціонарні ряди легше аналізувати, оскільки їхні властивості не змінюються з часом. Це дозволяє застосовувати стандартні інструменти та методи аналізу, такі як автокореляційні функції та спектральний аналіз [30].

Стаціонарні ряди зазвичай забезпечують більш точні прогнози, оскільки їхня поведінка більш передбачувана. Нестаціонарні ряди можуть мати тенденції, сезонні коливання або інші компоненти, які можуть ускладнювати прогнозування.

На рисунку 3.3 зображений фрагмент коду перевірки даних на стаціонарність.



```
[47]:
def check_stationarity(series):
    # Thanks to https://machinelearningmastery.com/time-series-data-stationary-python/

    result = adfuller(series.values)

    print('ADF Statistic: %f' % result[0])
    print('p-value: %f' % result[1])
    print('Critical Values:')
    for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))

    if (result[1] <= 0.05) & (result[4]['5%'] > result[0]):
        print("\u001b[32mStationary\u001b[0m")
    else:
        print("\x1b[31mNon-stationary\x1b[0m")
```

Рисунок 3.3 – Фрагмент коду для перевірки ряду на стаціонарність

На рисунку 3.4 зображено результат перевірки даних на стаціонарність.

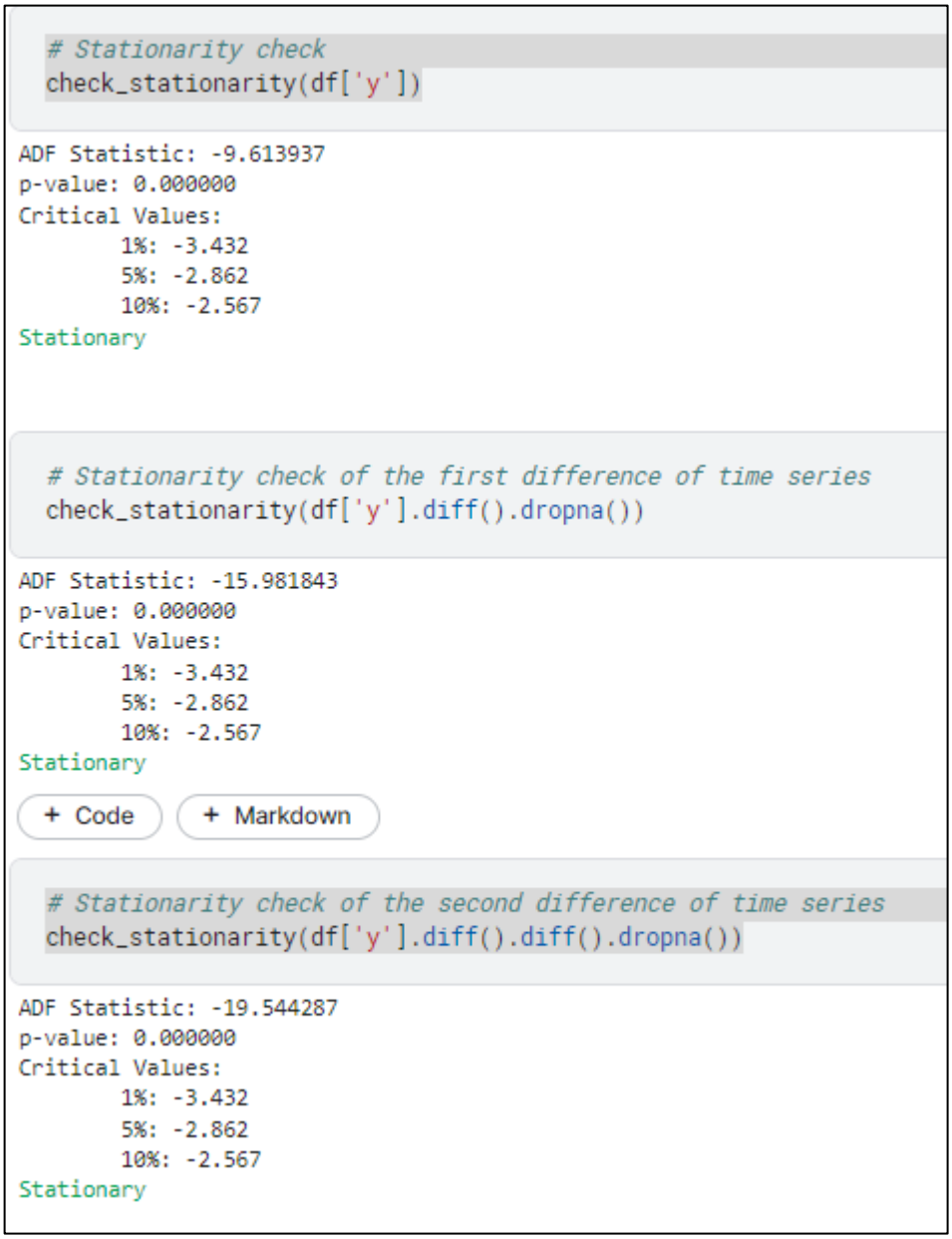


Рисунок 3.4 – Інформація про стаціонарність ряду

Оригінальний часовий ряд є стаціонарним, оскільки значення ADF Statistic (-9.613937) є меншою за критичні значення на всіх рівнях значущості (1%, 5%, 10%). p-value (0.000000) є значно меншою за 0.05, що свідчить про відхилення нульової гіпотези (часовий ряд не має одиничного кореня, тобто є стаціонарним).

Перша різниця часового ряду також є стаціонарною, оскільки: значення ADF Statistic (-15.981843) є меншою за критичні значення на всіх рівнях значущості (1%, 5%, 10%). p-value (0.000000) є значно меншою за 0.05, що свідчить про відхилення нульової гіпотези.

Друга різниця часового ряду також є стаціонарною, оскільки: значення ADF Statistic (-19.544287) є меншою за критичні значення на всіх рівнях значущості (1%, 5%, 10%). p-value (0.000000) є значно меншою за 0.05, що свідчить про відхилення нульової гіпотези.

Дані не мають потреби в додатковому диференціюванні для досягнення стаціонарності. Це означає, що можна безпосередньо використовувати оригінальний ряд для моделювання та прогнозування.

Для прогнозування часових рядів можна використовувати моделі ARIMA та Facebook Prophet.

Прогнозування відбуватиметься на 1 день, оскільки дані мають інтервал в 20 хвилин.

Задаємо початкову конфігурацію, назва набору даних “PM1”, цільову зміну y, та кількість днів прогнозування, яка дорівнює 1 (фрагмент коду рис 3.5)

```
# Set main parameters
series_name = 'PM1'
target = 'y'
forecasting_days = 1
```

Рисунок 3.5 – Початкові налаштування

За допомогою функції зображеної на рисунку 3.6 зчитуємо дані із датасету за ідентифікатором обраної станції та створюємо нову вибірку з колонками ds та y, де ds – це конкретний час в проміжку, а y – показник забруднення PM_1.

```python
def get_data(df, id_station):
    if id_station in df['id_station'].unique():
        station_data = df[df['id_station'] == id_station].copy()
        station_data['datetime'] = pd.to_datetime(station_data['date']) + pd.to_timedelta((station_data['time_point'] - 1) * 20, unit='m')
        new_dataframe = pd.DataFrame({'ds': station_data['datetime'], 'y': station_data['value']})
        return new_dataframe
    else:
        print(f"Station ID {id_station} does not exist in the dataset.")
```

```python
column_names = ['id_station', 'location', 'date', 'time_point', 'Column6','indicator','dimension','value']
df = pd.read_csv('/kaggle/input/air-quality-monitoring-from-ecocity/ECOCITY_Archive_651_561_2024-03-25_2024-05-10.csv', sep=',', names=column_names)

df = get_data(df, 1315)
df
```

Рисунок 3.6 – Функція зчитування та формування нової вибірки

На рисунку 3.7 зображений результат виклику функції.

	ds	y
0	2024-03-25 00:00:00	26.8422
1	2024-03-25 00:20:00	25.4215
2	2024-03-25 00:40:00	24.4250
3	2024-03-25 01:00:00	22.8410
4	2024-03-25 01:20:00	22.6225
5	2024-03-25 01:40:00	25.4483
6	2024-03-25 02:00:00	24.0770
7	2024-03-25 02:20:00	24.0658
8	2024-03-25 02:40:00	28.8442
9	2024-03-25 03:00:00	27.3440
10	2024-03-25 03:20:00	25.9956

Рисунок 3.7 – Дані по станції ВНТУ 1315

Оскільки це дані моніторингу якості повітря з інтервалом в 20 хвилин, на які впливають багато факторів, як погодні умови, технологічні фактори та інші – вибірка виявилась зашумленою.

Зашумленість даних, також відома як шум даних, - це небажане або стороннє втручання в дані, яке може призвести до неточності, помилок або невідповідності інформації [30].

Зашумленість даних може мати негативний вплив на результати аналізу даних і машинного навчання. Наприклад, вона може призвести до:

– Неточні прогнози: Моделі машинного навчання, навчені на зашумлених даних, можуть давати неточні прогнози або приймати неправильні рішення.

– Неправильні висновки: Якщо дані зашумлені, можуть бути неправильні висновки про дані та закономірності, які вони містять.

Для того щоб позбутись зашумленості прийнято рішення агрегувати дані подобово по максимальному значені, що зображено на рисунку 3.8.

```
# Resample to daily with max()
df.set_index('ds', inplace=True)
df = df.resample('D').max()
df.index = df.index.strftime('%Y-%m-%d')
df = df.dropna()
df.reset_index(inplace=True)
display(df)
```

	ds	y
0	2024-03-25	29.1970
1	2024-03-26	25.6225
2	2024-03-27	15.0000
3	2024-03-28	20.5079
4	2024-03-29	22.1432
5	2024-03-30	24.5647
6	2024-03-31	30.7505
7	2024-04-01	19.9521
8	2024-04-02	21.4484
9	2024-04-03	15.5837
10	2024-04-04	28.0432
11	2024-04-05	30.8890
12	2024-04-06	21.5360
13	2024-04-07	24.9865
14	2024-04-08	19.3895

Рисунок 3.8 – Агрегація даних

На рисунку 3.9 зображено графік агрегованих даних.

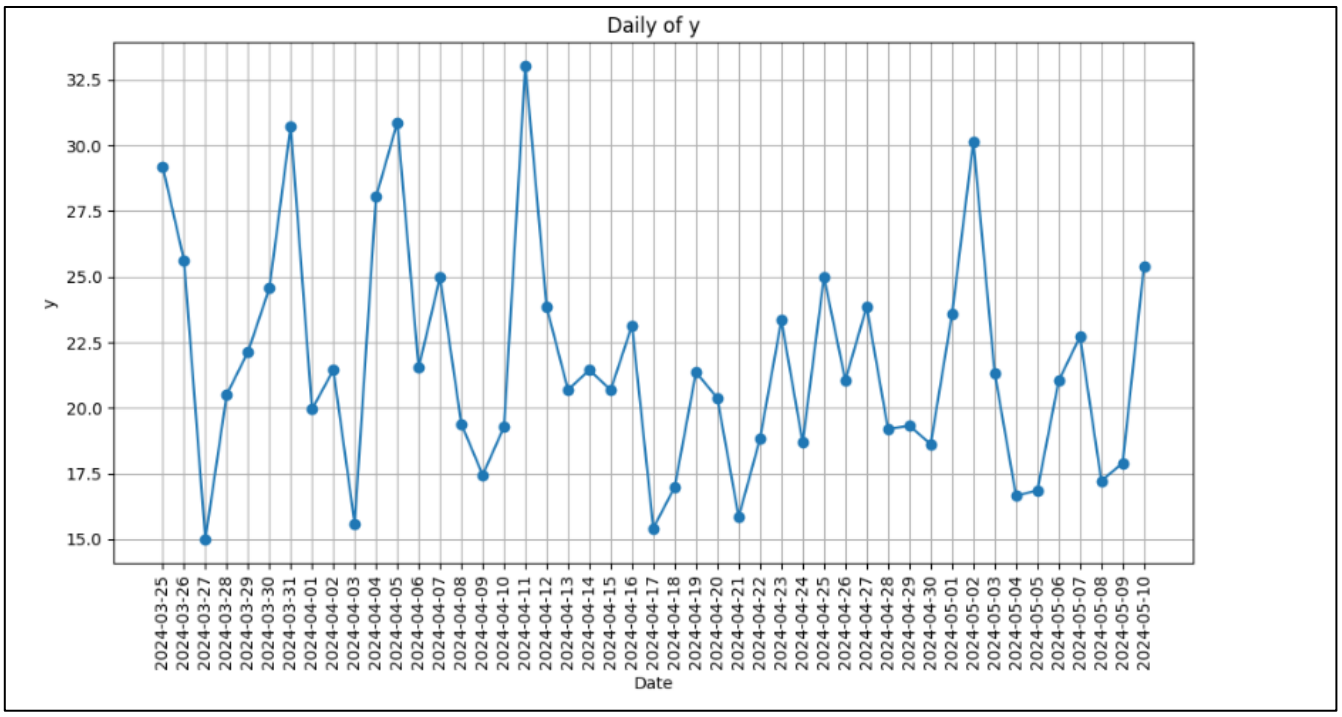


Рисунок 3.9 – Графік агрегованих даних

Виявлення аномалій є важливим кроком у підготовці даних для прогнозування часових рядів. У даній роботі використано бібліотеку FBProphet для прогнозування, а також методи для виявлення аномальних значень у даних. Ці аномалії можуть суттєво впливати на якість прогнозу, тому їх ідентифікація та обробка є критично важливими.

На рисунку 3.10 ми знаходимо дати, коли значення цільової змінної у перевищує поріг 24.9, це значення виявлено після розвідувального аналізу даних. Ці дати вважаються аномальними, оскільки значення у значно вище звичайного діапазону [31].

```
# Get anomalous dates
anomalous_dates = df[df['y'] > 24.9]['ds'].tolist()
anomalous_dates
```

```
['2024-03-25',
 '2024-03-26',
 '2024-03-31',
 '2024-04-04',
 '2024-04-05',
 '2024-04-07',
 '2024-04-11',
 '2024-04-25',
 '2024-05-02',
 '2024-05-10']
```

Рисунок 3.10 – Знаходження аномальних дат

На фрагменті коду, наведеному на рисунку 3.11 зображено створення копії датафрейму й обчислення різниці між поточним і попереднім значенням y, а також графік різниць. Це допомагає виявити великі зміни, які також можуть бути аномальними.

```
# Difference anomalous
df2 = df.copy()
df2['y_diff_anomalous'] = df2['y'].diff().dropna()

# Plot drawing
plt.figure(figsize=(10, 6))
plt.plot(df2['ds'], df2['y_diff_anomalous'], marker='o', linestyle='-')
plt.xlabel('Date')
plt.ylabel('y_diff_anomalous')
plt.title('Daily of y_diff_anomalous')
plt.xticks(rotation=90)
plt.grid(True)
plt.tight_layout()
plt.show()
```

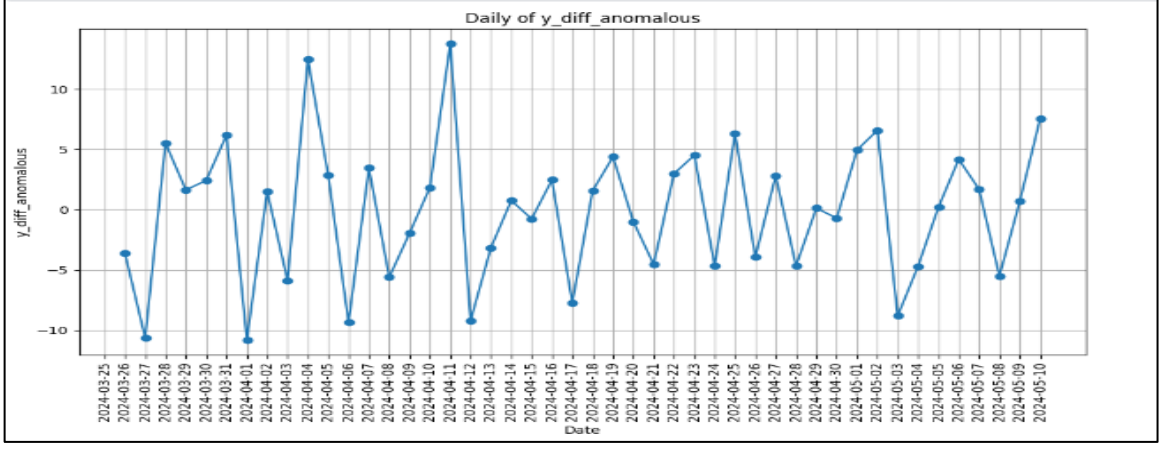


Рисунок 3.11 – Обчислення різниць та графік різниць

На рисунку 3.12 виявляємо дати, коли абсолютна різниця значень “y” між сусідніми днями перевищує 10. Такі зміни вважаються аномальними, оскільки вони є значними відхиленнями від норми.

```
# Get difference anomalous
anomalous_dates_diff_new = df2[df2['y_diff_anomalous'].abs() >= 10]['ds'].tolist()
anomalous_dates_diff_new
```

```
['2024-03-27', '2024-04-01', '2024-04-04', '2024-04-11']
```

Рисунок 3.12 – Виявлення аномальної різниці між сусідніми днями

Після цих дій маємо вибірку аномальних дат, а саме '2024-03-31', '2024-04-07', '2024-04-11', '2024-05-02', '2024-04-25', '2024-05-10', '2024-04-01', '2024-03-26', '2024-04-04', '2024-03-25', '2024-03-27', '2024-04-05. У цей діапазон також попадають дати, коли пилок Сахари був активний на території України, інші ж дати можуть бути спровокованими іншими чинниками або ж «подорожуванням» пилу в повітрі.

На рисунку 3.13 зображено синтез датафрейму з аномальними даними для моделі Facebook Prophet.

```
# Synthesis dataframe with anomalous dates for Facebook Prophet
if is_anomalies:
    holidays_df = pd.DataFrame(columns = ['ds', 'lower_window', 'upper_window', 'prior_scale'])
    holidays_df['ds'] = anomalous_dates_diff      # anomalous_dates
    holidays_df['holiday'] = 'anomalous_dates'
    holidays_df['lower_window'] = -2
    holidays_df['upper_window'] = 2
    holidays_df['prior_scale'] = 50
    display(holidays_df)
```

	ds	lower_window	upper_window	prior_scale	holiday
0	2024-03-31	-2	2	50	anomalous_dates
1	2024-04-07	-2	2	50	anomalous_dates
2	2024-04-11	-2	2	50	anomalous_dates
3	2024-05-02	-2	2	50	anomalous_dates
4	2024-04-25	-2	2	50	anomalous_dates
5	2024-05-10	-2	2	50	anomalous_dates
6	2024-04-01	-2	2	50	anomalous_dates
7	2024-03-26	-2	2	50	anomalous_dates
8	2024-04-04	-2	2	50	anomalous_dates
9	2024-03-25	-2	2	50	anomalous_dates
10	2024-03-27	-2	2	50	anomalous_dates
11	2024-04-05	-2	2	50	anomalous_dates

Рисунок 3.13 – Синтез датафрейму з аномальними даними

Цей кодовий фрагмент демонструє, як можна інтегрувати аномальні дати в модель прогнозування за допомогою FBProphet. Це дозволяє моделі враховувати вплив аномальних подій на майбутні прогнози.

Після цього сформовано тренувальний, валідаційний та тестовий датасети. Вони всі одного розміру, що є нетрадиційно для машинного навчання, але підходить для використання у даній задачі (рис. 3.14)

```
Origin dataset has 47 rows and 2 features
Get training dataset with 47 rows
Get validation dataset with 47 rows
Get test dataset with 47 rows
```

Рисунок 3.14 – Розмір побудованих тренувального, валідаційного та тестового датасетів

Для моделі задамо такі параметри (рис. 3.15):

– Щотижнева сезонність увімкнена для врахування повторюваних змін протягом тижня;

– Щоденна та щорічна сезонність відключені, оскільки вони не є релевантними для даного набору даних;

– Точки зміни тренду: Використовуються всі дані для виявлення змін у тренді;

– Пріор масштабу зміни тренду встановлено на 0.5, що забезпечує більшу гнучкість;

– Аномальні дати враховуються, що дозволяє моделі коректно враховувати їх вплив на прогноз;

– Використовується мультиплікативна модель сезонності, що є більш відповідною для даних, де амплітуда сезонних змін залежить від рівня тренду.

```
def prophet_modeling(result,
                     series_name,
                     train,
                     test,
                     holidays_df,
                     period_days,
                     fourier_order_seasonality,
                     forecasting_period,
                     name_model,
                     type_data):

    # Build Prophet model with parameters and structure
    model = Prophet(daily_seasonality=False,
                    weekly_seasonality=True,
                    yearly_seasonality=False,
                    changepoint_range=1,
                    changepoint_prior_scale = 0.5,
                    holidays=holidays_df,
                    seasonality_mode = 'multiplicative'
                   )
    model.add_seasonality(name='seasonality', period=period_days,
                          fourier_order=fourier_order_seasonality,
                          mode = 'multiplicative', prior_scale = 0.5)
```

Рисунок 3.15 – Задані параметри для моделі

Фрагмент коду, зображений на рисунку 3.16 виконує налаштування моделі прогнозування FBProphet, перевіряючи різні комбінації параметрів сезонності. Кожна комбінація перевіряється в циклі, і результати прогнозування зберігаються для подальшого аналізу. Перевіряються різні періоди сезонності (3, 5, 21 днів), що дозволяє моделі враховувати сезонні коливання різної довжини.

Перевіряються різні порядки перетворення Фур'є (3 і 12), що дозволяє моделі враховувати різні складності сезонних патернів.

```
%%time
# Models tuning
if is_Prophet:
    for period_days in [3, 5, 21]:
        for fourier_order_seasonality in [3, 12]:
            result, _ = prophet_modeling(result,
                                         series_name,
                                         train_ts,
                                         valid_ts,
                                         holidays_df,
                                         period_days,
                                         fourier_order_seasonality,
                                         forecasting_days,
                                         f'{period_days}_days_{fourier_order_seasonality}_order',
                                         'valid')
```

Рисунок 3.16 – Тюнінг моделі

На рисунку 3.17 зображений прогноз якості повітря та поширення пилу Сахари за допомогою FBprophet.

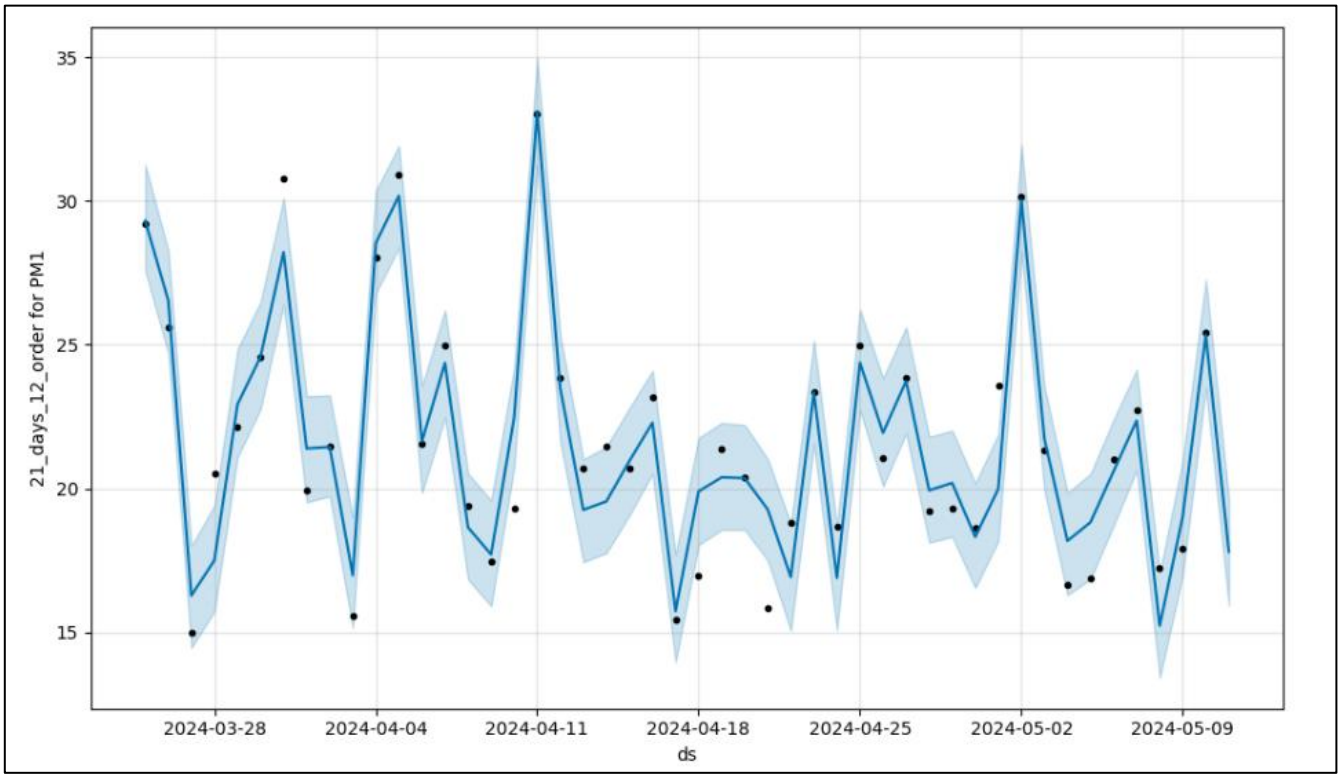


Рисунок 3.17 - Результат прогнозування якості повітря та поширення пилу Сахари за найкращою моделлю за валідаційними даними на основі Facebook Prophet

Prophet дозволяє візуалізувати окремі компоненти: лінію тренду, свята, у нашому випадку це аномальні дати та сезонність, це представлено на рисунку 3.18.

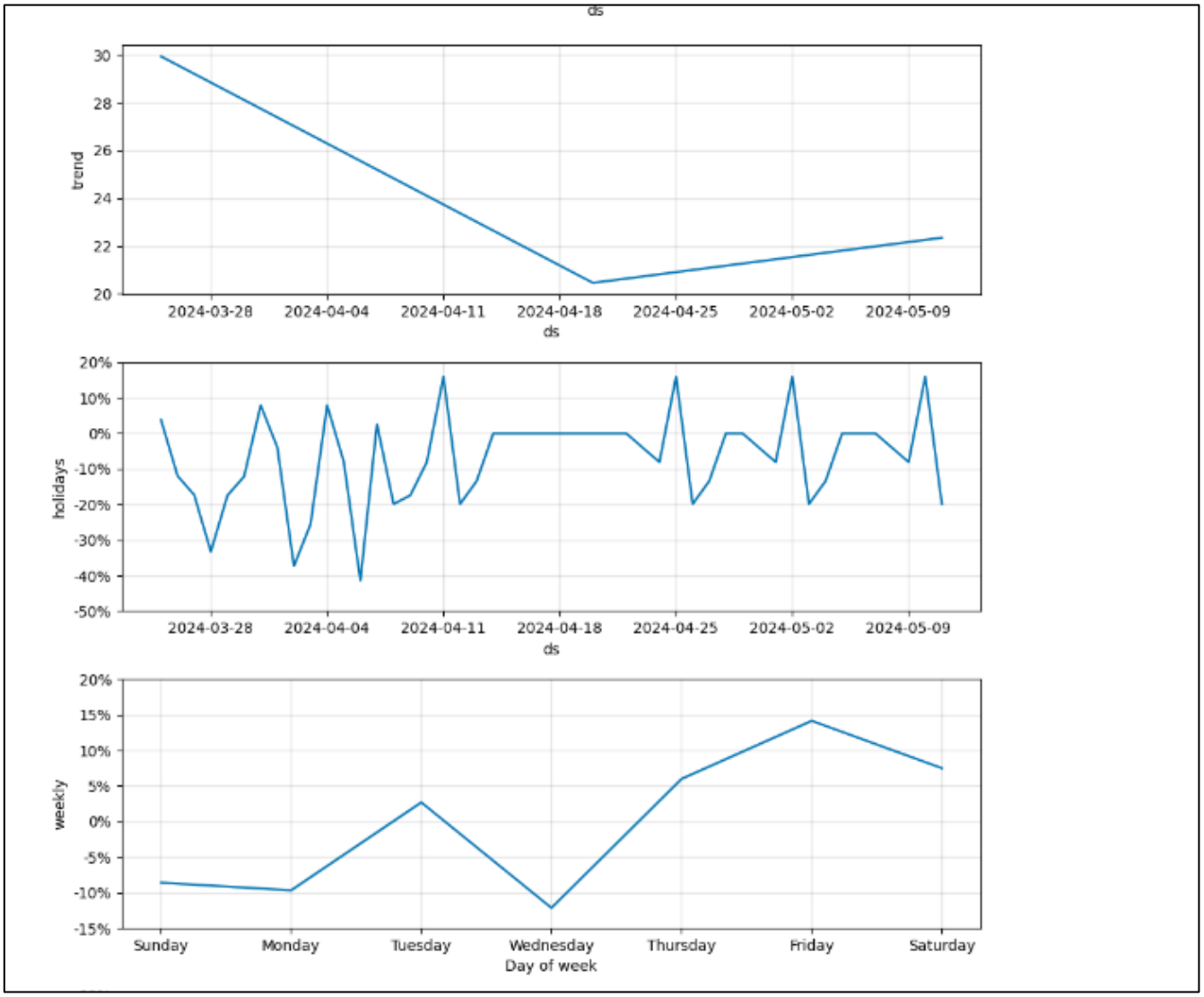


Рисунок 3.18 – Візуалізація графіків компонентів прогнозування за допомогою Prophet

Для прогнозування за допомого моделі ARIMA використаємо автоматизацію процесу побудови моделі ARIMA, а саме автоматичний перебір варіантів параметрів моделі (рис 3.19, 3.20).

```
Performing stepwise search to minimize aic
 ARIMA(4,0,4)(0,0,0)[0]             : AIC=inf, Time=0.69 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=426.482, Time=0.01 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=302.391, Time=0.03 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=379.206, Time=0.07 sec
 ARIMA(2,0,0)(0,0,0)[0]             : AIC=301.248, Time=0.03 sec
 ARIMA(3,0,0)(0,0,0)[0]             : AIC=inf, Time=0.04 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=285.893, Time=0.10 sec
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=284.757, Time=0.04 sec
 ARIMA(1,0,2)(0,0,0)[0]             : AIC=285.171, Time=0.09 sec
 ARIMA(0,0,2)(0,0,0)[0]             : AIC=357.571, Time=0.08 sec
 ARIMA(2,0,2)(0,0,0)[0]             : AIC=inf, Time=0.09 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=275.361, Time=0.12 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=275.525, Time=0.04 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=276.209, Time=0.03 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=275.814, Time=0.17 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=277.097, Time=0.07 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=275.053, Time=0.01 sec

Best model:  ARIMA(0,0,0)(0,0,0)[0] intercept
Total fit time: 1.736 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   47
Model:                        SARIMAX   Log Likelihood                -135.527
Date:                Fri, 31 May 2024   AIC                            275.053
Time:                        16:01:08   BIC                            278.754
Sample:                             0   HQIC                           276.446
                                  - 47
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     21.7006      0.735     29.510      0.000      20.259      23.142
sigma2        18.7130      4.448      4.207      0.000       9.996      27.430
===================================================================================
Ljung-Box (L1) (Q):                   0.84   Jarque-Bera (JB):                 4.23
Prob(Q):                              0.36   Prob(JB):                         0.12
Heteroskedasticity (H):               0.52   Skew:                             0.73
Prob(H) (two-sided):                  0.20   Kurtosis:                         3.05
===================================================================================
```

Рисунок 3.19 – Автоматичний підбір найкращих параметрів та результат роботи найкращої моделі

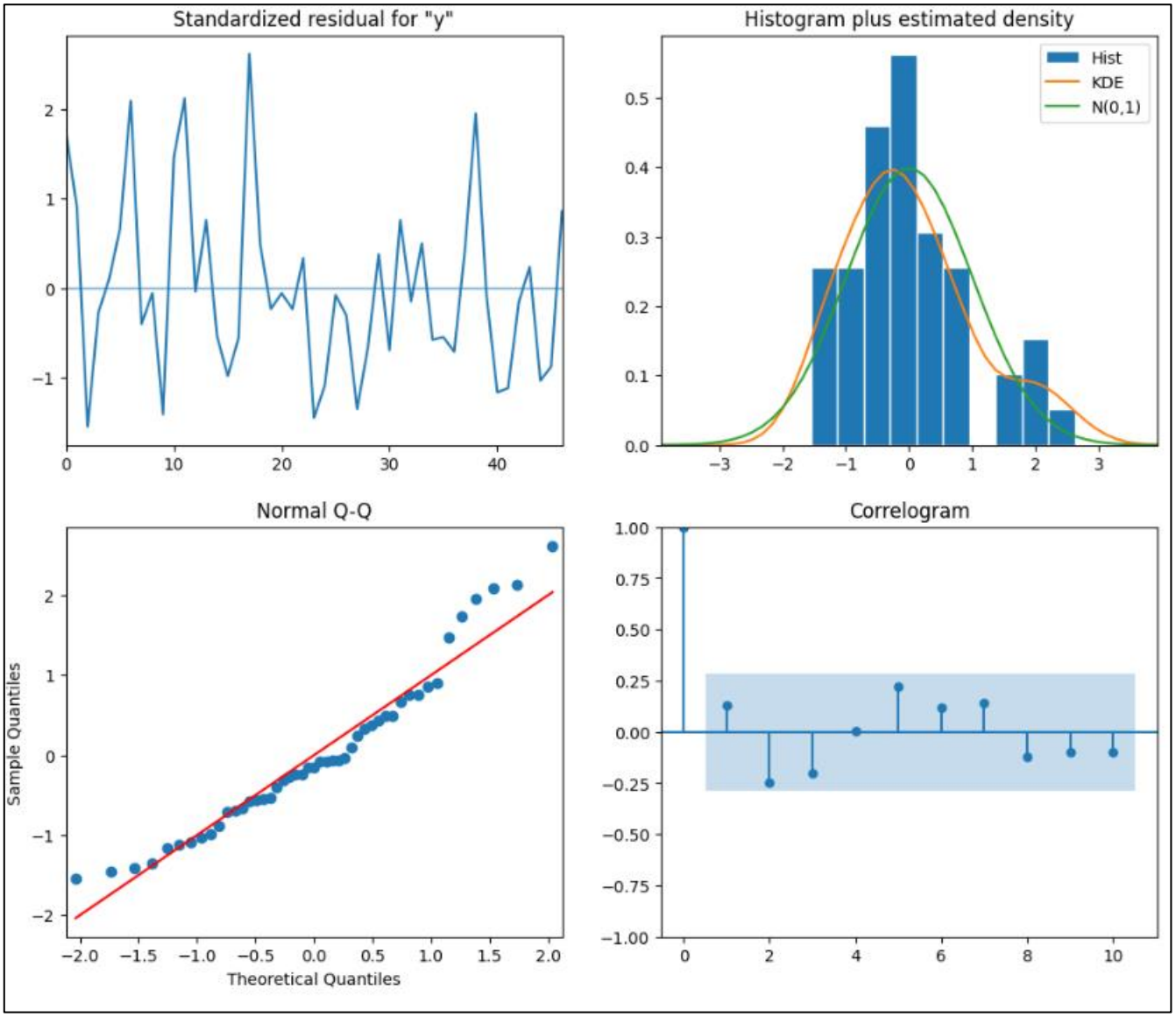


Рисунок 3.20 – Результат діагностики (похибки прогнозу валідаційних даних) оптимальної моделі ARIMA

На рисунку 3.21 зображена таблиця результатів оцінки різних моделей прогнозування для валідаційних даних.

	name_model	type_data	r2_score	rmse	mape
5	Prophet_21_days_12_order	valid	0.890255	1.433061	5.29409
3	Prophet_5_days_12_order	valid	0.833217	1.766637	7.219881
2	Prophet_5_days_3_order	valid	0.811897	1.876156	7.537381
1	Prophet_3_days_12_order	valid	0.763567	2.103415	8.469911
0	Prophet_3_days_3_order	valid	0.755483	2.139074	8.653403
4	Prophet_21_days_3_order	valid	0.731006	2.243583	8.755588
6	ARIMA_auto	valid	-0.0	4.325848	15.844534

Рисунок 3.21 – Таблиця результатів оцінки різних моделей прогнозування

Таблиця з рисунку 3.21 містить наступні стовпці:

– name_model: Назва моделі та її параметрів;

– type_data: Тип даних (в даному випадку всі дані є валідаційними, тобто valid);

– r2_score: Коефіцієнт детермінації, що показує, наскільки добре модель пояснює варіацію даних;

– rmse: Корінь середньоквадратичної помилки (Root Mean Squared Error), що вимірює середню відстань між передбаченими та фактичними значеннями;

– mape: Середня абсолютна відносна помилка (Mean Absolute Percentage Error), що вимірює середню відносну помилку в процентах [30].

Перевірено різні конфігурації періодів сезонності та порядків перетворення Фур'є.

Кращі результати показала модель Prophet_21_days_12_order з показниками r2_score 0,89, RMSE 1,43 MAPE 5,29%, що свідчить про те, що довший період сезонності та вищий порядок перетворення Фур'є краще моделюють сезонні коливання у даних цієї станції.

Автоматична модель ARIMA показала найгірші результати, що свідчить про її непридатність для даного набору даних у порівнянні з моделями Prophet.

Для прогнозування даних інших станцій модель ARIMA не буде враховуватись, оскільки її результати найгірші.

На рисунку 3.22 зображено графік розкиду даних, який показує зв'язок між прогнозованими та фактичними значеннями тестових даних.

Точки, які щільно розташовані навколо лінії тренду свідчать про те, що модель прогнозування дає точні результати.

Точки, які відхиляються від лінії тренду свідчать про те, що модель прогнозування дає неточні результати для цих точок даних.

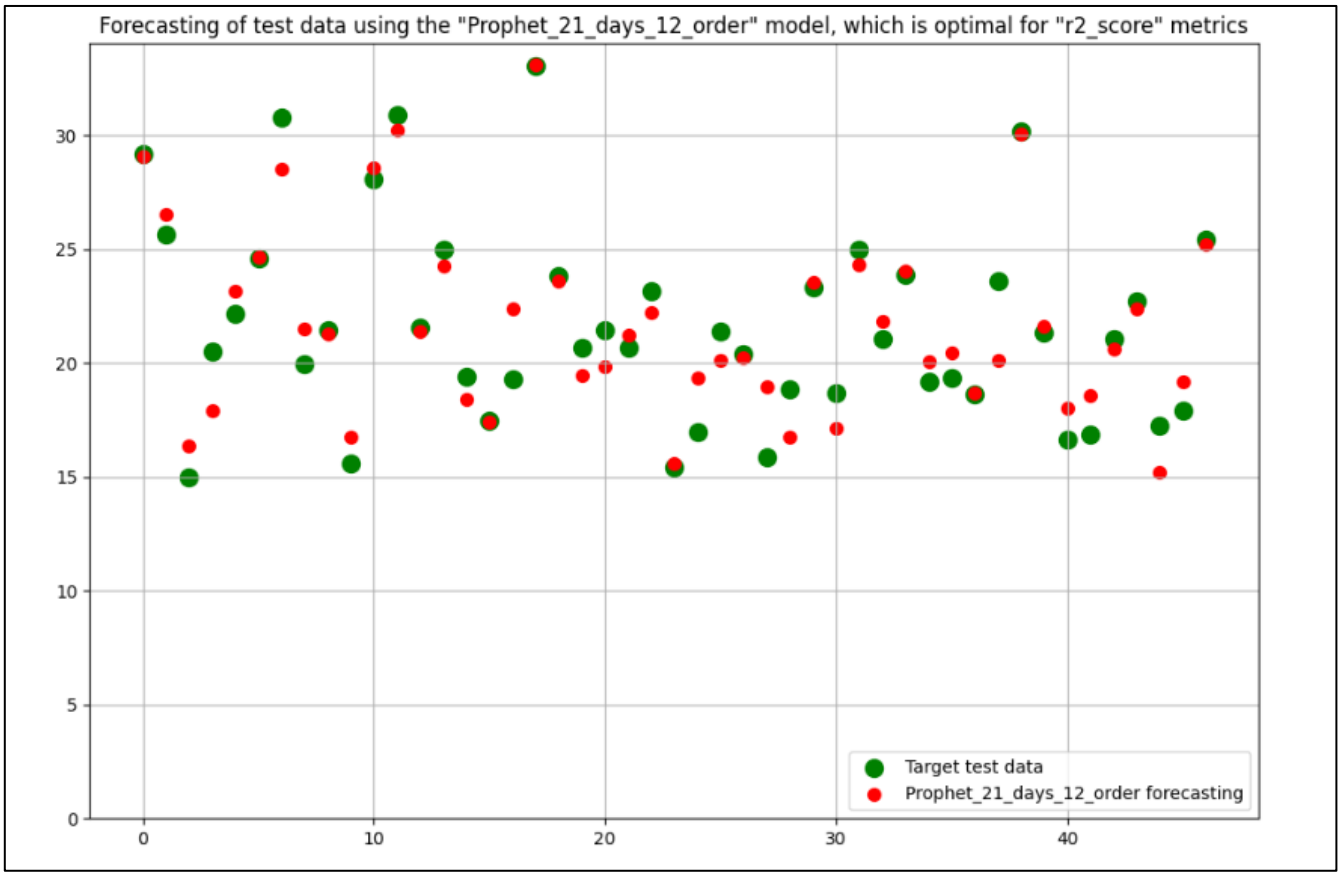


Рисунок 3.22 - Графік розкиду даних, який показує зв'язок між прогнозованими та фактичними значеннями тестових даних

З готовими моделям побудуємо прогнозування і для інших станцій із датасету.

На рисунках 3.23, 3.24 зображені графік розкиду даних та таблиця результатів оцінки різних моделей прогнозування для станції 1612.

Для даних станції 1612 модель Prophet_21_days_12_order показала найкращі результати серед представлених моделей, маючи найвищий r2_score 0,82, найнижчий RMSE 4,5 та найнижчий MAPE. Однак, значення MAPE досить високе (29,3%), що свідчить про значні відхилення прогнозу від реальних значень у відсотках.

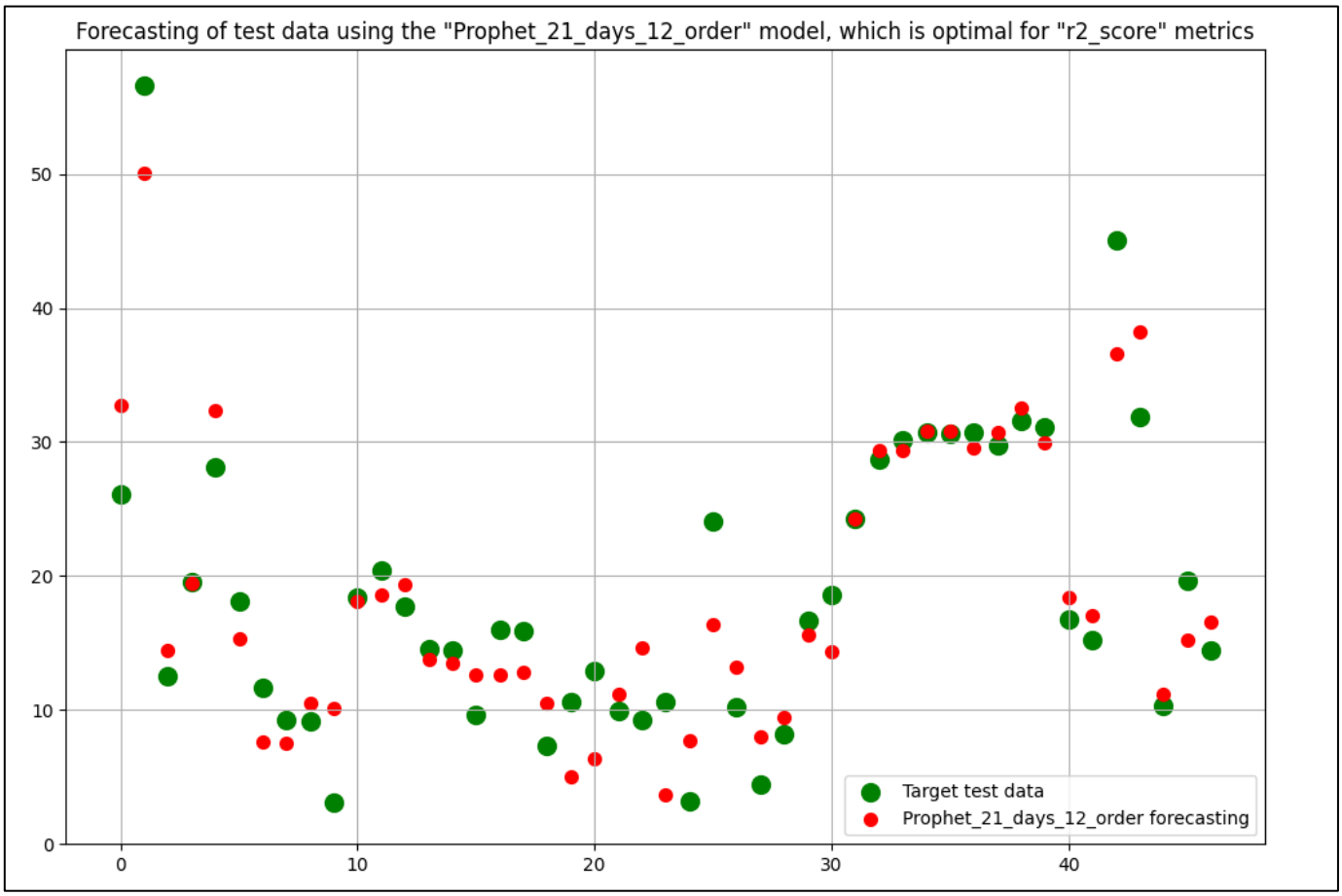


Рисунок 3.23 - Графік розкиду даних, який показує зв'язок між прогнозованими та фактичними значеннями тестових даних для станції 1612

	name_model	type_data	r2_score	rmse	mape
5	Prophet_21_days_12_order	valid	0.825219	4.50362	29.324069
4	Prophet_21_days_3_order	valid	0.771143	5.153433	34.093503
1	Prophet_3_days_12_order	valid	0.768988	5.177647	35.603932
3	Prophet_5_days_12_order	valid	0.767217	5.197445	37.443577
0	Prophet_3_days_3_order	valid	0.727429	5.624124	39.432009
2	Prophet_5_days_3_order	valid	0.743153	5.459485	40.310621
6	ARIMA_auto	valid	-25.746259	55.711683	295.32302

Number of models built - 7

Рисунок 3.24 – Таблиця результатів оцінки різних моделей прогнозування для станції 1612

На рисунках 3.25, 3.26 зображені графік розкиду даних та таблиця результатів оцінки різних моделей прогнозування для станції 1864.

Найкращі результати досягнуті моделлю Prophet_21_days_12_order, що робить її найкращим вибором для прогнозування часових рядів у цьому випадку. Вона показала найвищий R2 0,82 та найнижчі RMSE і MAPE серед всіх моделей.

Ця модель має найкращі показники серед представлених, проте високе значення MAPE (33.31%) вказує на необхідність подальшого покращення моделі, можливо, шляхом врахування додаткових факторів або використання інших методів.

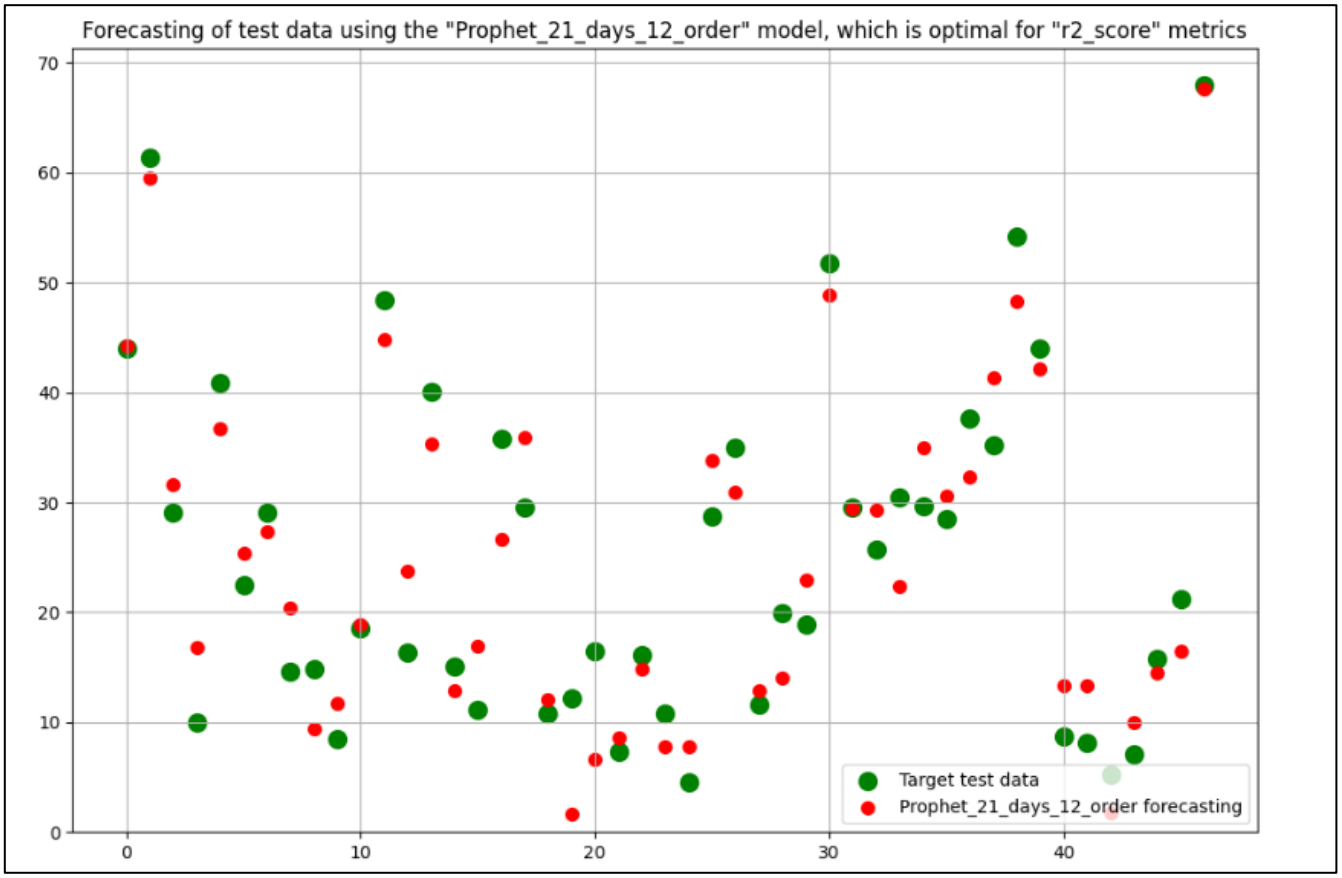


Рисунок 3.25 - Графік розкиду даних, який показує зв'язок між прогнозованими та фактичними значеннями тестових даних для станції 1864

	name_model	type_data	r2_score	rmse	mape
5	Prophet_21_days_12_order	valid	0.821463	6.547193	33.311556
3	Prophet_5_days_12_order	valid	0.767536	7.470825	36.589998
1	Prophet_3_days_12_order	valid	0.761257	7.571051	37.061687
2	Prophet_5_days_3_order	valid	0.759508	7.598734	37.602653
0	Prophet_3_days_3_order	valid	0.752407	7.7101	37.908077
4	Prophet_21_days_3_order	valid	0.750926	7.733131	40.206116
6	ARIMA_auto	valid	-0.060875	15.959649	97.59841

Number of models built - 7

Рисунок 3.26 – Таблиця результатів оцінки різних моделей прогнозування для станції 1864

3.2 Висновки

Розроблено інформаційну технологію прогнозування поширення пилу Сахари Україною.

Розроблено та протестовано кілька моделей прогнозування, зокрема моделі часових рядів.

Найкращі результати показала модель Prophet, яка забезпечила високу точність прогнозів для декількох станцій моніторингу.

В ході тестування різних моделей було оцінено їх ефективність за такими показниками, як R^2, RMSE та MAPE.

Обрано оптимальну модель Prophet_21_days_12_order, яка показала найкращий результат прогнозування з показниками R^2=0,89 для станції 1315, R^2=0,82 для станції 1612 та R^2=0,82 для станції 1864.

ВИСНОВКИ

В ході виконання бакалаврської дипломної роботи було проведено аналіз предметної області, представлено суть проблеми та доведена актуальність питання прогнозування поширення пилу Сахари в повітрі України.

У першому розділі проведено аналіз проблематики предметної області, а саме проблеми забруднення повітря пилом Сахари. Розглянуто основні можливості, переваги і недоліки існуючих інформаційних систем для моніторингу якості повітря. Обрано оптимальні засоби та інформаційні технології для вирішення задачі, здійснено опис та обґрунтування вибору.

У другому розділі проведено підготовку та розвідувальний аналіз даних для вирішення задачі прогнозування поширення пилу Сахари в Україні. Було розроблено ноутбук для візуалізації даних по конкретних годинах та датах, що допомагає зрозуміти просторовий розподіл показника якості повітря (PM_1) у Вінницькій області.

У третьому розділі розроблено інформаційну технологію для прогнозування поширення пилу Сахари в повітрі України на базі алгоритмів обробки даних та моделей прогнозування. Проведено тестування на реальних даних з різних станцій моніторингу. Продемонстровано роботу інформаційної технології в цілому, забезпечуючи точність та своєчасність прогнозування рівнів забруднення повітря пилом Сахари. Проведений відбір оптимальної моделі для прогнозування, а саме Prophet_21_days_12_order, яка показала себе найкраще для усіх станцій з показниками R^2=0,89 для станції 1315, R^2=0,82 для станції 1612, R^2=0,82 для станції 1864.

Таким чином, створена інформаційна технологія, яка дозволяє більш точно прогнозувати поширення пилу Сахари в атмосферному повітрі України за даними громадського моніторингу показника PM_1, який найкраще характеризує вміст цього пилу.

За результатами дослідження готується стаття у співавторстві до подання у науковий фаховий журнал категорії Б зі спеціальності 126.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Air Quality Index Scale and Color Legend [Електронний ресурс]. URL: https://aqicn.org/scale/

2. Advances in air quality research – current and emerging challenges [Електронний ресурс]. URL: https://acp.copernicus.org/articles/22/4615/2022/

3. Air pollution deaths attributable to fossil fuels: observational and modelling study [Електронний ресурс]. URL: https://www.bmj.com/content/383/bmj-2023-077784.

4. Respiratory system [Електронний ресурс]. URL: completehttps://commons.wikimedia.org/wiki/File:Respiratory_system_complete_en.svg

5. SaveEcoBot [Електронний ресурс]. URL: https://www.saveecobot.com/

6. Карта моніторингу якості повітря EcoCity [Електронний ресурс]. URL: https://eco-city.org.ua/

7. 6 Major Causes of Air Pollution and Most Common Pollutants [Електронний ресурс]. URL: https://www.machengineering.com/major-causes-of-air-pollution/

8. Sharp increase in Saharan dust intrusions over the western Euro-Mediterranean in February–March 2020–2022 and associated atmospheric circulation [Електронний ресурс]. URL: https://acp.copernicus.org/articles/24/4083/2024

9. Супутникові знімки Землі WorldView [Електронний ресурс]. URL: https://worldview.earthdata.nasa.gov

10. Що таке мова Python та які сфери її застосування [Електронний ресурс]. URL: https://university.sigma.software/what-is-python/

11. Де використовується Python і чому вам потрібно знати цю мову [Електрон-ний ресурс]. URL: https://genius.space/lab/de-vikoristovuyetsya-python-i-chomu-vam-potribno-znati-tsyu-movu/

12. Бібілотека NumPy [Електронний ресурс]. URL: https://numpy.org/

13. 10 найкращих бібліотек Python для машинного навчання та ШІ [Електронний ресурс]. URL: https://www.unite.ai/uk/10-best-python-libraries-for-machine-learning-ai/

14. Бібілотека pandas [Електронний ресурс]. URL: https://pandas.pydata.org

15. Підручник Python Pandas: DataFrame, діапазон дат, використання Pandas [Електронний ресурс]. URL: https://www.guru99.com/uk/python-pandas-tutorial.html

16. Бібілотека Matplotlib [Електронний ресурс]. URL: https://matplotlib.org/

17. Matplotlib [Електронний ресурс]. URL: https://kkite.pnu.edu.ua/wp-content/uploads/sites/50/2020/04/Matplotlib.pdf

18. Бібілотека SciPy [Електронний ресурс]. URL: https://scipy.org/

19. Підручник SciPy у Python: що таке, бібліотека, функції та приклади [Електронний ресурс]. URL: https://www.guru99.com/uk/scipy-tutorial.html

20. SciPy [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/SciPy

21. Бібілотека scikit-learn [Електронний ресурс]. URL: https://scikit-learn.org/stable/

22. Перші кроки в NLP: розглядаємо Python-бібліотеку scikit-learn в реальному завданні [Електронний ресурс]. URL: https://dou.ua/lenta/articles/first-steps-in-nlp-scikit-learn/

23. Бібілотека plotly [Електронний ресурс]. URL: https://plotly.com/

24. FBProphet [Електронний ресурс]. URL: https://pypi.org/project/fbprophet

25. Taylor S. J., Letham B. Forecasting at scale. PeerJ Preprints. 2017. 5:e3190v2. DOI: https://doi.org/10.7287/peerj.preprints.3190v2

26. Hyndman R. J., Athanasopoulos G. Forecasting: principles and practice. 2nd ed. Oxford : Oxford University Press, 2014..

27. Air Quality Monitoring from EcoCity [Електронний ресурс]. URL: https://www.kaggle.com/datasets/vbmokin/air-quality-monitoring-from-ecocity

28. Anomaly detection Sahara Vinnytsia [Електронний ресурс]. URL: https://www.kaggle.com/code/tarasskrynnyk/anomaly-detection-sahara-vinnytsia

29. Sahara's Dust in the Region - 2D Analysis [Електронний ресурс]. URL: https://www.kaggle.com/code/vbmokin/sahara-s-dust-in-the-region-2d-analysis

30. Мокін В. Б., Дратований М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних : електронний навчальний посібник комбінованого (локального та мережевого) використання [Електронний ресурс]. Вінниця : ВНТУ, 2024, 258 с. URL: https://docs.vntu.edu.ua/card.php?id=8163

31. Шмундяк Д. О., Копняк В. Є. Метод ідентифікації локальних аномалій значень показників стану довкілля з використанням декомпозиції на півхвилі, Вісник Вінницького політехнічного інституту, 2024. № 1, с. 88–100. DOI: https://doi.org/10.31649/1997-9266-2024-172-1-88-100

Додаток А

(обов’язковий)

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ

Завідувач кафедри САІТ

_____ д.т.н., проф. Віталій МОКІН

“____”_____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську дипломну роботу

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ПОШИРЕННЯ ПИЛКУ САХАРИ У ПОВІТРІ УКРАЇНИ ЗА ДАНИМИ ГРОМАДСЬКОГО МОНІТОРИНГУ

08-34.БДР.015.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Сергій ЖУКОВ
«__» _____ 2024 р.
Розробив студент гр. 2ІСТ-20б
_____ Тарас СКРИННИК
«__» _____ 2024 р.

Вінниця 2024

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____ 2024р., та індивідуальне завдання на БДР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2024р.

2. Джерела розробки:

1) Наука про дані: машинне навчання та інтелектуальний аналіз даних : електронний навчальний посібник комбінованого (локального та мережевого) використання [Електронний ресурс] / В. Б. Мокін, М. В. Дратований – Вінниця : ВНТУ, 2024. – 258 с. – Режим доступу: https://docs.vntu.edu.ua/card.php?id=8163

2) Шмундяк Д. О., Копняк В. Є. Метод ідентифікації локальних аномалій значень показників стану довкілля з використанням декомпозиції на півхвилі, Вісник Вінницького політехнічного інституту, 2024. № 1, с. 88–100. DOI: https://doi.org/10.31649/1997-9266-2024-172-1-88-100

3. Мета і призначення роботи.

Метою даної роботи є підвищення точності прогнозування поширення пилку Сахари в атмосферному повітрі України за даними громадського моніторингу шляхом створення інформаційної технології.

4. Вихідні дані для проведення робіт:

Air Quality Monitoring from EcoCity [Електронний ресурс]. URL: https://www.kaggle.com/datasets/vbmokin/air-quality-monitoring-from-ecocity

5. Методи дослідження:

Використання платформи Kaggle та мови програмування Python.

6. Етапи роботи і терміни їх виконання:

a) Загальна характеристика об’єкту дослідження _____ – _____
b) Підготовка та розвідувальний аналіз даних _____ – _____
c) Створення інформаційної технології прогнозування _____ – _____
d) Прогнозування поширення пилу _____ – _____
e) Оформлення матеріалів до захисту БДР _____ – _____

7. Очікувані результати та порядок реалізації

Отримання інформаційної технології аналізу та прогнозування поширення пилу Сахари у повітрі України за даними громадського моніторингу.

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2024 р.
Початок розробки «___» _____ 2024 р.
Граничні терміни виконання БДР «___» _____ 2024 р.

Розробив студент групи 2ІСТ-20б _____ Тарас СКРИННИК

Додаток Б

(обов'язковий)

Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень

Назва роботи: «Інформаційна технологія аналізу та прогнозування поширення пилку Сахари у повітрі України за даними громадського моніторингу»
Тип роботи: бакалаврська дипломна робота
Підрозділ: кафедра САІТ

Показники звіту подібності Unicheck

Оригінальність 93.25 % Схожість 6.75 %

Аналіз звіту подібності (відмітити потрібне)

- ● Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- ○ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
- ○ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку (підпис) Сергій ЖУКОВ

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи (підпис) Тарас СКРИННИК

Керівник роботи (підпис) Сергій ЖУКОВ

Додаток В

(довідниковий)

Фрагмент лістингу програми

Anomaly detection Sahara Vinnytsia

```
import numpy as np
import pandas as pd

# Visualization
import seaborn as sns
import matplotlib.pyplot as plt
column_names = ['id_station', 'location', 'date', 'time_point', 'Column6', 'indicator', 'dimension', 'value']
stations = pd.read_csv('/kaggle/input/air-quality-monitoring-from-ecocity/ECOCITY_Archive_651_561_2024-03-25_2024-05-10.csv', sep=',',
names=column_names)

# Визначення назв колонок для about_station
column_names_a = ["id_saveecobot", "id_ecocity", "network", "locality", "address", "start_date", "lat", "lng", "notes", "source"]
about_stations = pd.read_csv('/kaggle/input/eco-city-stations-about/ecocity_about_stations_2024.csv', header=None, names=column_names_a)

stations['id_station'] = pd.to_numeric(stations['id_station'], errors='coerce').dropna().astype(int)
about_stations['id_ecocity'] = pd.to_numeric(about_stations['id_ecocity'], errors='coerce').dropna().astype(int)
# Перевірка зчитаних даних
print(stations)
stations.info()

result = pd.merge(stations, about_stations[['id_ecocity', 'address']], left_on='id_station', right_on='id_ecocity', how='left')

result = result.drop(columns=['id_ecocity'])

result
#цей фрагмент коду зручності та розуміння, які станції наявні
unique_id_count = result['id_station'].nunique()
print("Кількість унікальних значень в стовбці id_station (кількість станцій):", unique_id_count)
unique_stations = result['id_station'].unique()
print("Id унікальних станцій")
print(unique_stations)
def get_data(id_station):
    if id_station in result['id_station'].unique():
        anomalies_datasets = {}
        station_data = result[result['id_station'] == id_station].copy()
        station_data['datetime'] = pd.to_datetime(station_data['date']) + pd.to_timedelta((station_data['time_point'] - 1) * 20, unit='m')
        station_data['datetime'] = pd.to_datetime(station_data['datetime'])
        station_data.set_index('datetime', inplace=True)
        address = station_data['address'].iloc[0]  # Отримання адреси для станції
        datasets = {id_station: station_data[['id_station', 'value']]}  # Вибір потрібних колонок без 'datetime'
        for station_id, dataset in datasets.items():
            plt.figure(figsize=(12, 6))
            plt.plot(dataset.index, dataset['value'], label=f'station_{station_id}')
            plt.xlabel('Date')
            plt.ylabel('Dimension (PM1)')
            plt.title(f'PM1 Dimension Over Time for station_{station_id}, {address}')
            plt.legend()
            plt.show()
            threshold = 0.95 * dataset['value'].max()
```

```
            anomalies = dataset[dataset['value'] > threshold]
            print(f'Anomalies for station_{station_id}:')
            print(anomalies)
            anomalies_datasets[station_id] = anomalies
        df_station = datasets[id_station].reset_index()
        df_anomalies_station = anomalies_datasets[id_station].reset_index()
        return df_station, df_anomalies_station
df_station, df_anomalies_station = get_data(1315)
print("="*40)
print("Dataset for the selected station:")
print(df_station)
print("="*40)
print("Anomalies for the selected station:")
print(df_anomalies_station)
```

Sahara Influence forecasting and EDA

```
# Import libraries
    import random
    import os
    import numpy as np
    import pandas as pd
    import requests

    # Date
    import datetime as dt
    from datetime import date, timedelta, datetime

    # EDA
    import matplotlib.pyplot as plt
    from matplotlib.pylab import rcParams
    import plotly.express as px
    import plotly.graph_objects as go
    from plotly.offline import init_notebook_mode
    init_notebook_mode(connected=True)

    # FE
    from tsfresh import extract_features, select_features, extract_relevant_features
    from tsfresh.utilities.dataframe_functions import impute
    from sklearn.inspection import permutation_importance
    import eli5
    from eli5.sklearn import PermutationImportance
    import shap

    # Time Series - EDA and Modelling
    import statsmodels.api as sm
    from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
    from statsmodels.tsa.stattools import adfuller
    from statsmodels.tsa.seasonal import seasonal_decompose
    from statsmodels.tsa.arima_model import ARIMA
```

```
# Metrics
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error

# Modeling and preprocessing
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from prophet import Prophet

import warnings
warnings.filterwarnings("ignore")

# What EDA & FE techniques use?
is_anomalies = True # or False - Take into account anomalies or no?
# What type of model to use?
is_Prophet = True   # or False - Facebook Prophet
is_ARIMA = True     # or False - ARIMA and AutoARIMA
is_other_ML = False  # or False - multi-factors models: trees, neural networks, etc.
if is_ARIMA:
   !pip install pmdarima
   import pmdarima as pm
def fix_all_seeds(seed):
   np.random.seed(seed)
   random.seed(seed)
   os.environ['PYTHONHASHSEED'] = str(seed)

random_state = 42
fix_all_seeds(random_state)
series_name = 'PM1'
target = 'y'
forecasting_days = 1
date_start = dt.datetime(2024, 3, 30)
date_end = dt.datetime(2024, 5, 5)
print(f"Time interval: from {date_start} to {date_end}")
def get_data(df, id_station):
   if id_station in df['id_station'].unique():
      station_data = df[df['id_station'] == id_station].copy()
      station_data['datetime'] = pd.to_datetime(station_data['date']) + pd.to_timedelta((station_data['time_point'] - 1) * 20, unit='m')
      new_dataframe = pd.DataFrame({'ds': station_data['datetime'], 'y': station_data['value']})
      return new_dataframe
   else:
      print(f"Station ID {id_station} does not exist in the dataset.")
column_names = ['id_station', 'location', 'date', 'time_point', 'Column6','indicator','dimension','value']
df = pd.read_csv('/kaggle/input/air-quality-monitoring-from-ecocity/ECOCITY_Archive_651_561_2024-03-25_2024-05-10.csv', sep=',',
names=column_names)

df = get_data(df, 1315  )
display(df.head(15))
```

```
df.set_index('ds', inplace=True)
df = df.resample('D'). max()
df.index = df.index.strftime('%Y-%m-%d')
df = df.dropna()
df.reset_index(inplace=True)
display(df)
# Plot drawing
plt.figure(figsize=(10, 6))
plt.plot(df['ds'], df['y'], marker='o', linestyle='-')
plt.xlabel('Date')
plt.ylabel('y')
plt.title('Daily of y')
plt.xticks(rotation=90)
plt.grid(True)
plt.tight_layout()
plt.show()
# Get anomalous dates
anomalous_dates = df[df['y'] > 24.9]['ds'].tolist()
anomalous_dates
if is_anomalies:
    anomalous_dates_diff = anomalous_dates.copy()
    print(anomalous_dates_diff)
# Difference anomalous
df2 = df.copy()
df2['y_diff_anomalous'] = df2['y'].diff().dropna()

# Plot drawing
plt.figure(figsize=(10, 6))
plt.plot(df2['ds'], df2['y_diff_anomalous'], marker='o', linestyle='-')
plt.xlabel('Date')
plt.ylabel('y_diff_anomalous')
plt.title('Daily of y_diff_anomalous')
plt.xticks(rotation=90)
plt.grid(True)
plt.tight_layout()
plt.show()
# Get difference anomalous
anomalous_dates_diff_new = df2[df2['y_diff_anomalous'].abs() >= 10]['ds'].tolist()
anomalous_dates_diff_new
if is_anomalies:
    anomalous_dates_diff = list(set(anomalous_dates + anomalous_dates_diff_new))
    print(anomalous_dates_diff)
# Synthesis a new feature in df for anomalous_dates_diff
if is_anomalies:
    df['y_diff_anomalous'] = df['ds'].isin(anomalous_dates_diff).astype('int')
    display(df)

    # Number of anomalous dates
    print(f"Number of anomalous dates - {df['y_diff_anomalous'].sum()}")
```

```
# Synthesis dataframe with anomalous dates for Facebook Prophet
if is_anomalies:
    holidays_df = pd.DataFrame(columns = ['ds', 'lower_window', 'upper_window', 'prior_scale'])
    holidays_df['ds'] = anomalous_dates_diff      # anomalous_dates
    holidays_df['holiday'] = 'anomalous_dates'
    holidays_df['lower_window'] = -2
    holidays_df['upper_window'] = 2
    holidays_df['prior_scale'] = 50
    display(holidays_df)
if is_Prophet:
    df2 = df.copy()
if is_Prophet:
    train_ts, valid_ts, test_ts, train_valid_ts = get_train_valid_test_ts(df2.copy(), forecasting_days, target='y')

    if not is_anomalies:
        holidays_df = None
def prophet_modeling(result,
                     series_name,
                     train,
                     test,
                     holidays_df,
                     period_days,
                     fourier_order_seasonality,
                     forecasting_period,
                     name_model,
                     type_data):

    # Build Prophet model with parameters and structure
    model = Prophet(daily_seasonality=False,
                    weekly_seasonality=True,
                    yearly_seasonality=False,
                    changepoint_range=1,
                    changepoint_prior_scale = 0.5,
                    holidays=holidays_df,
                    seasonality_mode = 'multiplicative'
                   )
    model.add_seasonality(name='seasonality', period=period_days,
                          fourier_order=fourier_order_seasonality,
                          mode = 'multiplicative', prior_scale = 0.5)
    # Training model for df
    model.fit(train)

    # Make a forecast
    future = model.make_future_dataframe(periods = forecasting_period)
    forecast = model.predict(future)

    # Draw plot of the values with forecasting data
    figure = model.plot(forecast, xlabel = 'ds', ylabel = f"{name_model} for {series_name}")
```

```
    # Draw plot with the components (trend and seasonalities) of the forecasts
    figure_component = model.plot_components(forecast)

    # Forecasting data by the model
    #ypred = forecast['yhat'][-forecasting_period:]
    ypred = forecast['yhat'][:(len(test))]
    #print(ypred)
    # Save results
    n = len(result)
    result.loc[n,'name_model'] = f"Prophet_{name_model}"
    result.loc[n,'type_data'] = type_data
    result.at[n,'params'] = [period_days]+[fourier_order_seasonality]
    result.at[n,'ypred'] = ypred

    return result, ypred

%%time
# Models tuning
if is_Prophet:
    for period_days in [3, 5, 21]:
        for fourier_order_seasonality in [3, 12]:
            result, _ = prophet_modeling(result,
                                         series_name,
                                         train_ts,
                                         valid_ts,
                                         holidays_df,
                                         period_days,
                                         fourier_order_seasonality,
                                         forecasting_days,
                                         f'{period_days}_days_{fourier_order_seasonality}_order',
                                         'valid')
# Get datasets
if is_ARIMA:
    train_ts, valid_ts, test_ts, train_valid_ts = get_train_valid_test_ts(df2.copy(), forecasting_days, target='y')
ef acf_pacf_draw(df, lag_num=40, acf=True, pacf=True, title="", ylim=1):
    # Draw plots named title with ACF and PACF for dataframe df

    num_plots = 1+int(acf)+int(pacf)
    fig, ax = plt.subplots(1,num_plots,figsize=(12,6))
    # 'Original Series'
    ax[0].plot(df.values.squeeze())

    if acf:
        # ACF drawing
        plot_acf(df.values.squeeze(), lags=lag_num, ax=ax[1])
        ax[1].set(ylim=(-ylim, ylim))

        if pacf:
            # PACF drawing
```

```
            plot_pacf(df.values.squeeze(), lags=lag_num, ax=ax[2])
            ax[2].set(ylim=(-ylim, ylim))

        elif pacf:
            # PACF drawing
            plot_pacf(df.values.squeeze(), lags=lag_num, ax=ax[1])
            ax[1].set(ylim=(-ylim, ylim))

        fig.suptitle(title)
        plt.show()
    if is_ARIMA:
        # ACF and PACF
        lag_num = 10
        acf_pacf_draw(train_ts['y'], lag_num, True, True, 'Original Series')
        acf_pacf_draw(train_ts['y'].diff().dropna(), lag_num, True, True, '1st Order Differencing')
        acf_pacf_draw(train_ts['y'].diff().diff().dropna(), lag_num, True, True, '2nd Order Differencing')
        acf_pacf_draw(train_ts['y'].diff().diff().diff().dropna(), lag_num, True, True, '3rd Order Differencing')
    def arima_fit(df, col, order=(1,1,1)):
        # ARIMA model fitting for series df[col]

        model = sm.tsa.arima.ARIMA(df[col].values.squeeze(), order=order)
        model = model.fit()
        return model
    def arima_forecasting(result, model, params, name_model, df, type_data):
        # Data df (validation or test) forecasting on the num days by the model
        # with params and save metrics to result

        ypred = model.forecast(steps=len(df))

        n = len(result)
        result.loc[n,'name_model'] = name_model
        result.loc[n,'type_data'] = type_data
        result.at[n,'params'] = params
        result.at[n,'ypred'] = ypred
        #result = result_add_metrics(result, n, df['y'], y_pred)

        return result
    %%time
    if is_ARIMA:
        # Automatic tuning of the ARIMA model
        model_auto = pm.auto_arima(train_ts['y'].values,
                                   start_p=4,       # start p
                                   start_q=4,       # start q
                                   test='adf',      # use adftest to find optimal 'd'
                                   max_p=5, max_q=5, # maximum p and q
                                   m=1,             # frequency of series (1 - No Seasonality)
                                   d=None,          # let model determine 'd'
                                   seasonal=False,  # No Seasonality
                                   start_P=0,
```

```
                    D=0,
                    start_Q=0,
                    trace=True,
                    error_action='ignore',
                    suppress_warnings=False,
                    stepwise=True
                   )

    print(model_auto.summary())
if is_ARIMA:
    # Get orders of the best model from AutoARIMA
    arima_orders_best = list(model_auto.get_params().get('order'))
    print(f"Optimal parameters are {arima_orders_best}")
    model_auto = arima_fit(train_ts, 'y', order=(arima_orders_best[0],arima_orders_best[1],arima_orders_best[2]))
f is_ARIMA:
    # Best model from AutoARIMA
    fig = model_auto.plot_diagnostics(figsize=(12,10))
    plt.show()
def result_recover_and_metrics(result, df_ts, type_data, start_points):
    # Recovering prediction: from shifted_Close_diff to Close
    # Calculation metrics for recovering ypred forecasting for all models in result
    # ypred real is from df_ts['y']
    # start points value for the recovering is from dictionary start_points
    # type_data = 'valid' or 'test'

    for i in range(len(result)):
        if (result.loc[i, 'type_data']==type_data) and (result.loc[i, 'mape'] is np.nan):
            ypred = result.loc[i, 'ypred']

            # Recovering ypred for multi-factors models
            if not (str(result.loc[i, 'type_model']) in ['Prophet', 'ARIMA']):
                # Multi-factors model
                # Get start points value for the recovering
                start_point_value = start_points['valid_start_point'] if type_data=='valid' else start_points['test_start_point']
                # Recovering prediction
                ypred = recovery_prediction(ypred, start_point_value)

            # Calculation metrics
            result = result_add_metrics(result, i, df_ts['y'], ypred)

    return result
if len(result) > 0:

    # Get type of each model
    result['type_model'] = result['name_model'].str.split('_').str[0]

    # Calculation metrics for recovering prediction ypred for validation dataset by all models
    result = result_recover_and_metrics(result, valid_ts, 'valid', starting_point)
    #display(result[['name_model', 'type_data', 'mape']].sort_values(by=['type_data', 'mape'], ascending=True))
```

```
    display(result[['name_model', 'type_data', 'r2_score', 'rmse', 'mape']].sort_values(by=['type_data', 'mape', 'rmse'], ascending=True))

    # Save results
    num_models = len(result[result['type_data']=='valid']['name_model'].unique().tolist())
    print(f"Number of models built - {num_models}")
    result.to_csv(f'result_of_{num_models}_models_for_forecasting_days_{forecasting_days}.csv')
else:
    print('There are no tuned models!')
def model_training_forecasting(result, df, y, test, ytest,
                               name_model, type_model, params, type_test='1'):
    # Model training for df and y
    # Forecasting ypred
    # type_model = 'Prophet' or "ARIMA" or 'Other ML'
    # type_test = '1' (with find optimal parameters by GridSearchCV)
    # type_test = '2' (with optimal parameters - without GridSearchCV)
    # return params and metrics in the dataframe result

    if type_model=='Prophet':
        season_days_optimal = params[0]
        fourier_order_seasonality_optimal = params[1]
        model_opt = None
        _, ypred = prophet_modeling(result,
                                    series_name,
                                    df,
                                    test,
                                    holidays_df,
                                    season_days_optimal,
                                    fourier_order_seasonality_optimal,
                                    forecasting_days,
                                    f'{type_model}_optimal',
                                    'test')
    elif type_model=='ARIMA':
        season_days_optimal = params[0]
        fourier_order_seasonality_optimal = params[1]
        model_opt = None

        # Training ARIMA optimal model for training+valid dataset
        df['y'] = y
        model_opt = arima_fit(df, 'y', order=(params[0],params[1],params[2]))

        # Model diagnostics
        fig = model_opt.plot_diagnostics(figsize=(12,10))
        plt.show()

        # Plot residual errors
        get_residual_errors(model_opt)

        # Test forecasting and save result
        ypred = model_opt.forecast(steps=len(test))
```

```
# Scoring and saving results into the dataframe result
    n = len(result)-1
    result.loc[n,'name_model'] = f"{type_model}_optimal"
    result.loc[n,'type_data'] = "test"
    result.loc[n,'type_model'] = type_model
    result.at[n,'params'] = params
    result.at[n,'ypred'] = ypred
    #result = result_add_metrics(result, n, ytest, ypred)

    return result, model_opt, ypred
def get_optimal_model_and_forecasting(result, main_metrics, start_points):
    # Choosion the optimal model from dataframe result by main_metrics
    # Tuning optimal model for big dataset train+valid
    # Test forecasting and drawing it
    # Returns the optimal model and it's name


    if len(result) > 0:
        # Get parameters of the optimal model from dataframe result by main_metrics
        opt_name_model, opt_type_model, opt_params_model = get_params_optimal_model(result,
                                                           main_metrics)
        # Set datasets for the final tuning and testing by optimal model
        if (opt_type_model=='Prophet') or (opt_type_model=='ARIMA'):
            train_valid = train_valid_ts.copy()
            y_train_valid = train_valid_ts['y'].copy()
            test = test_ts.copy()
            ytest = test_ts['y'].copy()

        else:
            # Multi-factors ML models
            train_valid = train_valid_mf.copy()
            y_train_valid = y_train_valid_mf.copy()
            test = test_mf.copy()
            ytest = ytest_mf.copy()

        # Optimal model training for train+valid and test forecasting
        result, model_opt, ypred = model_training_forecasting(result, train_valid, y_train_valid,
                                               test, ytest,
                                               opt_name_model, opt_type_model,
                                               opt_params_model, '1')

        # Calculation metrics for recovering prediction ypred for test dataset by the optimal model
        result = result_recover_and_metrics(result, test_ts, 'test', start_points)

        # Drawing plot for prediction for the test data
        if not ((opt_type_model=='Prophet') or (opt_type_model=='ARIMA')):
            # Recovery values "Close"
            ytest_plot = recovery_prediction(ytest.values, start_points['test_start_point'])
            ypred_plot = recovery_prediction(ypred, start_points['test_start_point'])
```

```
        else:
            ytest_plot = ytest.copy()
            ypred_plot = ypred.copy()

        # Drawing
        plt.figure(figsize=(12,8))
        x = np.arange(len(ytest_plot))
        plt.scatter(x, ytest_plot, label = "Target test data", color = 'g', s=100)
        plt.scatter(x, ypred_plot, label = f"{opt_name_model} forecasting", color = 'r', s=50)
        plt.title(f'Forecasting of test data using the "{opt_name_model}" model, which is optimal for "{main_metrics}" metrics')
        plt.ylim(0)
        plt.legend(loc='lower right')
        plt.grid(True)

        return opt_name_model
# Get the optimal model by different metrics
if len(result) > 0:
    for valid_metrics in ['r2_score', 'rmse', 'mape']:
        get_optimal_model_and_forecasting(result, valid_metrics, starting_point)
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ПОШИРЕННЯ ПИЛКУ САХАРИ У ПОВІТРІ УКРАЇНИ ЗА ДАНИМИ ГРОМАДСЬКОГО МОНІТОРИНГУ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

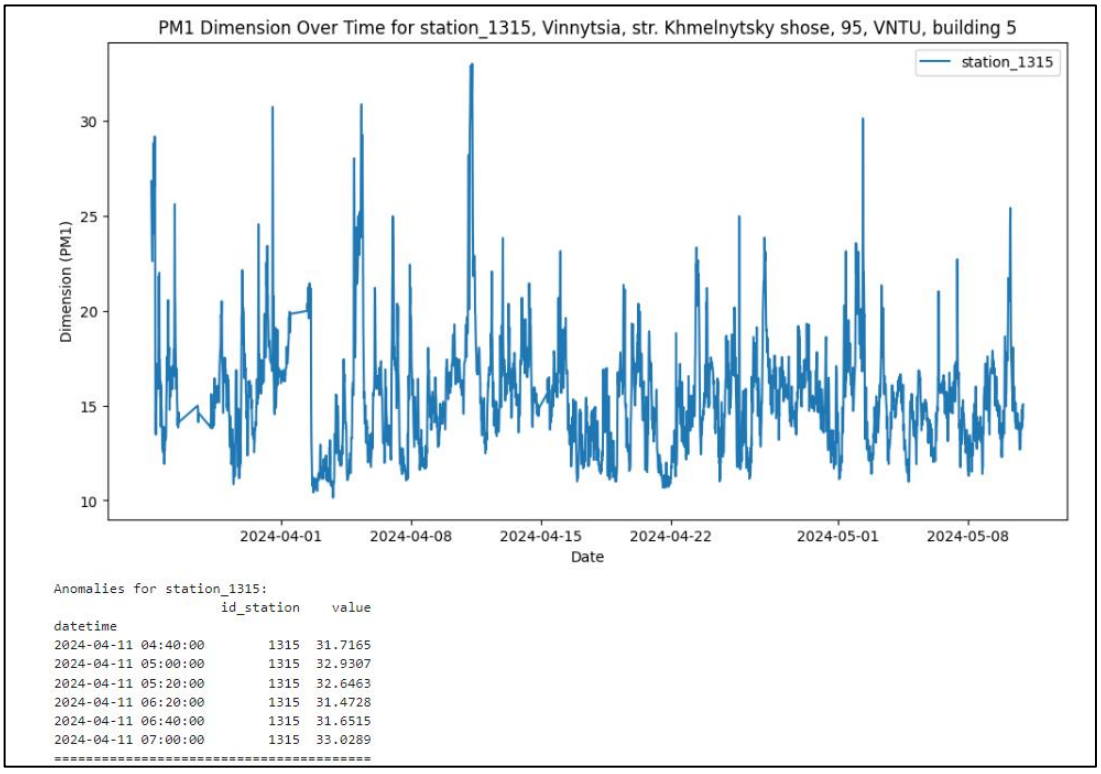
«___» _____ 2024 р.

[27]:

	id_station	location	date	time_point	Column6	indicator	dimension	value	address
0	1315	49.23327 28.409161	2024-03-25	1	18	PM1.0	ug/m3	26.8422	Vinnytsia, str. Khmelnytsky shose, 95, VNTU, b...
1	1315	49.23327 28.409161	2024-03-25	2	20	PM1.0	ug/m3	25.4215	Vinnytsia, str. Khmelnytsky shose, 95, VNTU, b...
2	1315	49.23327 28.409161	2024-03-25	3	18	PM1.0	ug/m3	24.4250	Vinnytsia, str. Khmelnytsky shose, 95, VNTU, b...
3	1315	49.23327 28.409161	2024-03-25	4	20	PM1.0	ug/m3	22.8410	Vinnytsia, str. Khmelnytsky shose, 95, VNTU, b...
4	1315	49.23327 28.409161	2024-03-25	5	20	PM1.0	ug/m3	22.6225	Vinnytsia, str. Khmelnytsky shose, 95, VNTU, b...
...	...	...	...	...	...	...	...	...	...
19642	1876	49.2048561 28.5288355	2024-05-10	68	20	PM1.0	ug/m3	6.3255	Prov. Bohdana Khmel'nyts'koho, Vinnyts'ki Khutoru
19643	1876	49.2048561 28.5288355	2024-05-10	69	20	PM1.0	ug/m3	5.8295	Prov. Bohdana Khmel'nyts'koho, Vinnyts'ki Khutoru
19644	1876	49.2048561 28.5288355	2024-05-10	70	20	PM1.0	ug/m3	3.7260	Prov. Bohdana Khmel'nyts'koho, Vinnyts'ki Khutoru
19645	1876	49.2048561 28.5288355	2024-05-10	71	20	PM1.0	ug/m3	3.9925	Prov. Bohdana Khmel'nyts'koho, Vinnyts'ki Khutoru
19646	1876	49.2048561 28.5288355	2024-05-10	72	20	PM1.0	ug/m3	4.0595	Prov. Bohdana Khmel'nyts'koho, Vinnyts'ki Khutoru

19647 rows × 9 columns

Рисунок Г.1 – Створений датасет Air Quality Monitoring from EcoCity



Anomalies for station_1315:

datetime	id_station	value
2024-04-11 04:40:00	1315	31.7165
2024-04-11 05:00:00	1315	32.9307
2024-04-11 05:20:00	1315	32.6463
2024-04-11 06:20:00	1315	31.4728
2024-04-11 06:40:00	1315	31.6515
2024-04-11 07:00:00	1315	33.0289

==========

Рисунок Г.2 – Графік даних та список аномальних дат для станції 1315

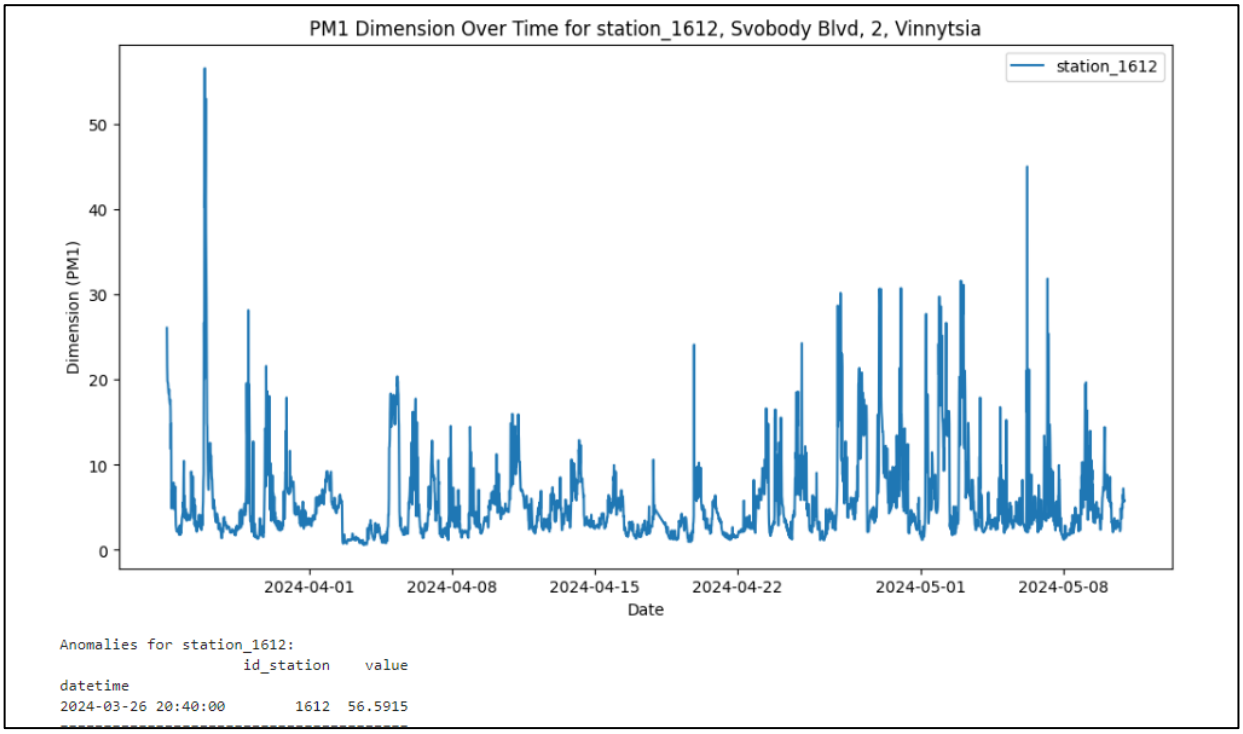


Рисунок Г.3 – Графік даних та список аномальних дат для станції 1612

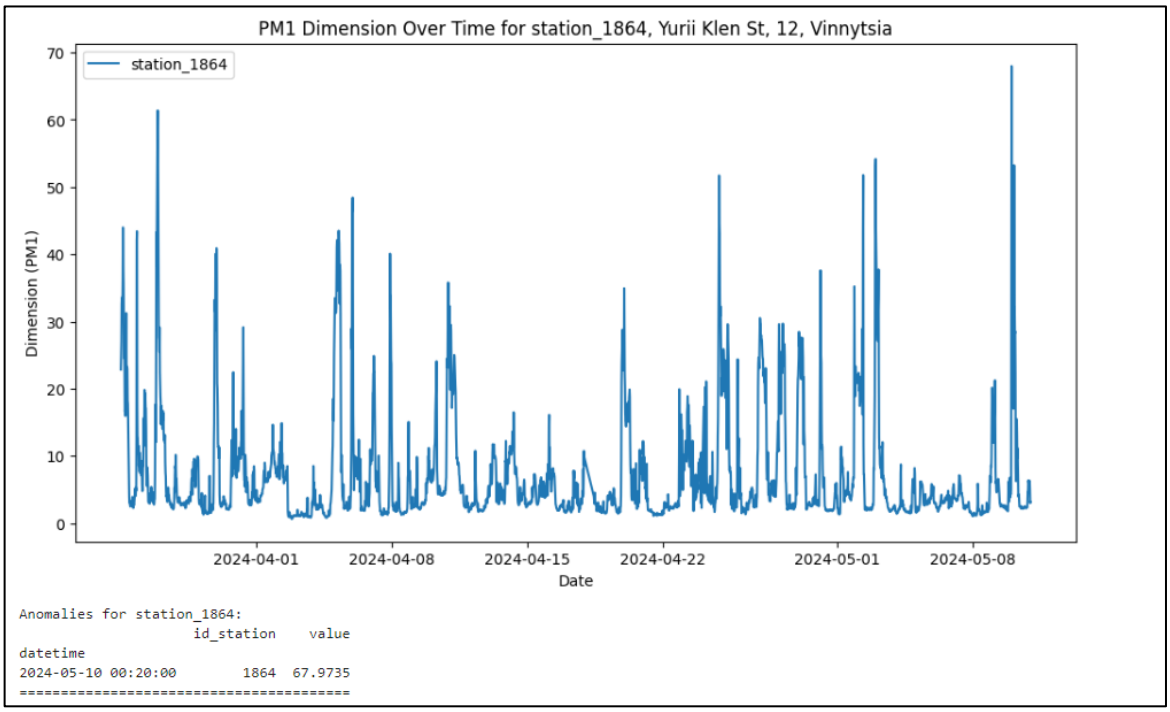


Рисунок Г.4 – Графік даних та список аномальних дат для станції 1864

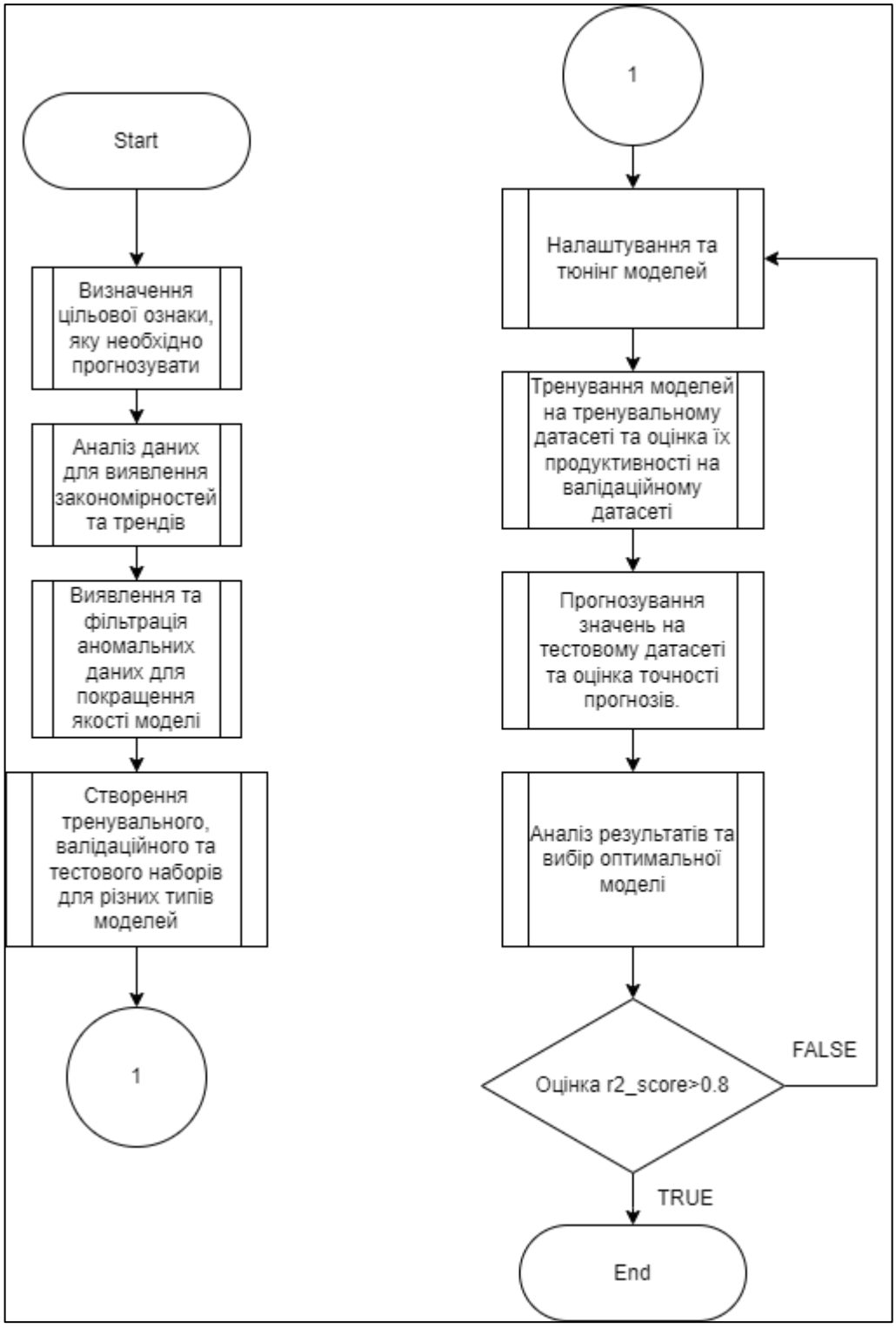


Рисунок Г.5 – Блок-схема алгоритму інформаційної технології аналізу та прогнозування поширення пилу Сахари Україною

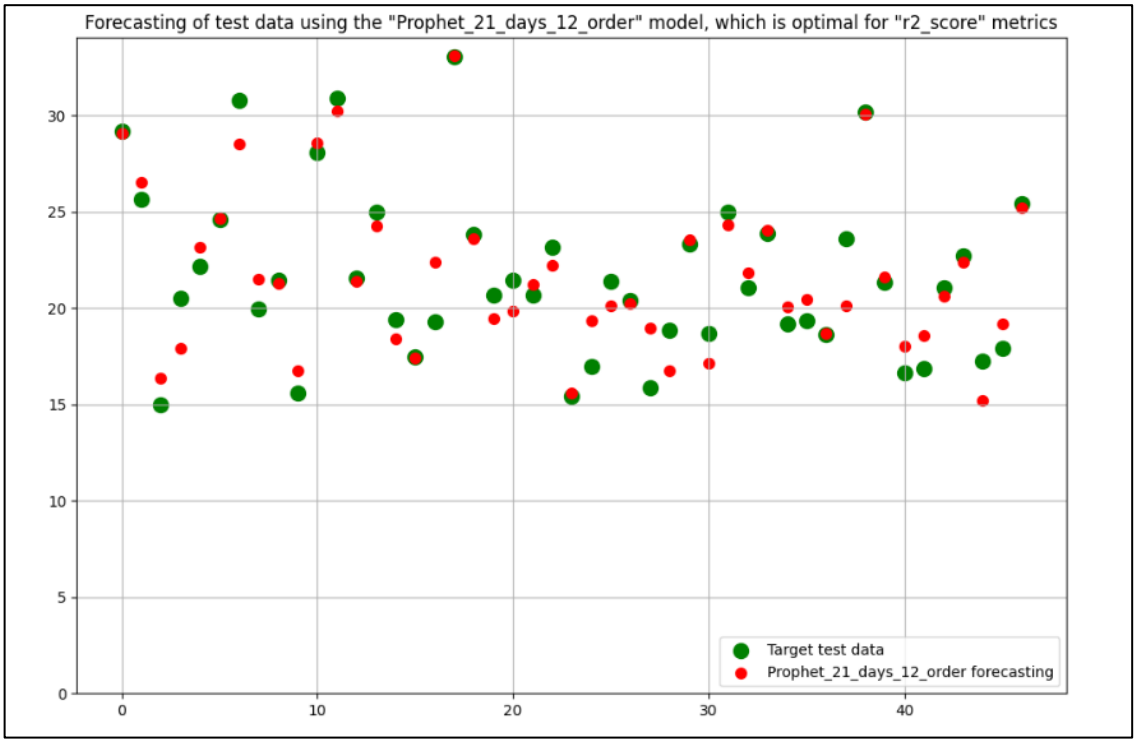


Рисунок Г.6 – Результат прогнозування якості повітря та поширення пилу Сахари за найкращою моделлю Prophet_21_days_12_order для станції 1315

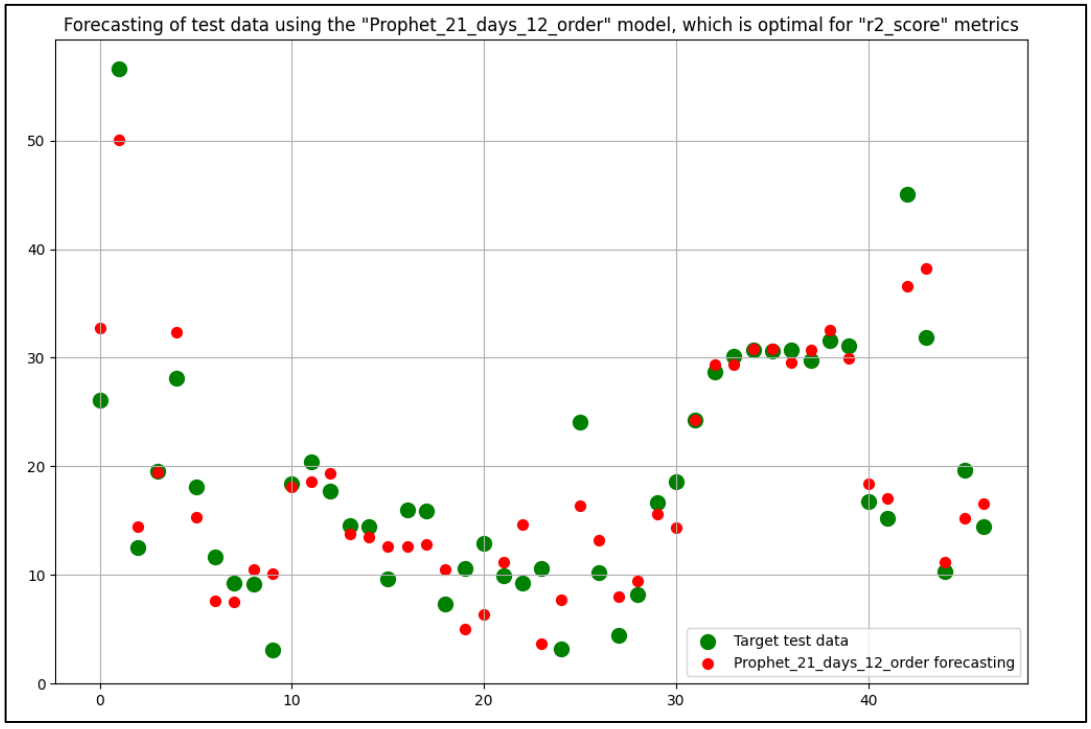


Рисунок Г.7 – Результат прогнозування якості повітря та поширення пилу Сахари за найкращою моделлю Prophet_21_days_12_order для станції 1612

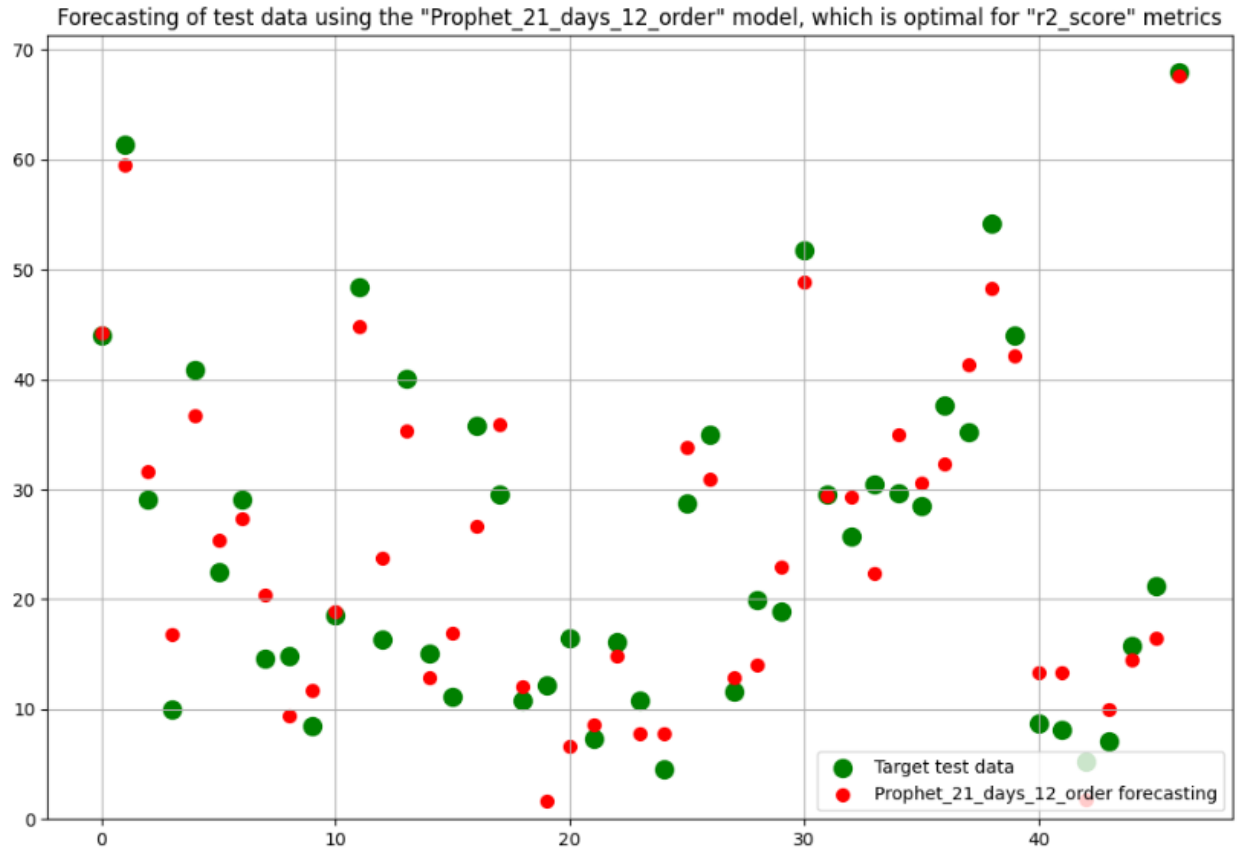


Рисунок Г.8 – Результат прогнозування якості повітря та поширення пилу Сахари за найкращою моделлю Prophet_21_days_12_order для станції 1864