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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія прогнозування концентрації нітратів у річковій воді Південного Бугу»

Виконав: студент 2 курсу, групи 2ІСТ-21м
спеціальності 126 – «Інформаційні системи та технології»
_____Лісовський Р.Р.
Керівник: к.т.н., доц. каф. САІТ
_____Жуков С.О.
«02» ____12____ 2022 р.
Опонент: к.т.н., доц. каф. ЗІ
_____Куперштейн Л.М.
«16» ____12____ 2022 р.

Допущено до захисту
Завідувач кафедри САІТ
_____д.т.н., проф. Мокін В. Б.
«05» ____12____ 2022 р.

Вінниця ВНТУ – 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – ІІ-й (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 126 Інформаційні системи та технології
Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Мокін В. Б.
«16» 09 2022 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Лісовському Ростиславу Руслановичу

1. Тема роботи: «Інформаційна технологія прогнозування концентрації нітратів у річковій воді Південного Бугу»,
керівник роботи: Жуков С.О., к.т.н., доц. каф. САІТ,
затверджені наказом закладу вищої освіти від «14» 09 2022 року №203
2. Строк подання студентом роботи «01» 12 2022 року
3. Вихідні дані до роботи:
   - Дані конкурсу «River Water Quality EDA and Forecasting» платформи Kaggle;
   - Електронна карта Вінницької області;
4. Зміст текстової частини:
   - Вибір оптимальних інформаційних технологій та розвідувальний аналіз даних;
   - Розроблення інформаційної технології та ідентифікація оптимальної моделі для прогнозування;
   - Застосування розробленої інформаційної технології та прогнозування змін концентрації нітратів у воді річки Південний Буг.
   - економічна частина.
5. Перелік ілюстративного матеріалу:
   - матриця коефіцієнта кореляції;
   - графік аномальних даних;
   - блок-схема алгоритму роботи інформаційної технології;
   - похибки моделей за різними метриками;
   - графік прогнозування за допомогою оптимальної моделі;
   - діаграма важливості ознак;

6. Консультанти розділів МКР

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Бурєннійкова Н.В., д.е.н., проф. каф. ЕПВМ	05.11.22	20.11.22

7. Дата видачі завдання «16» 09 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МКР	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	09.2022	
2	Огляд проблем створення інформаційної технології прогнозування концентрації нітратів у річковій воді Південного Бугу	09.2022	
3	Аналіз даних концетрації нітратів	09.2022	
4	Реалізація інформаційної технології	10.2022	
5	Аналіз результатів прогнозування	10.2022	
6	Економічна частина	11.2022	
7	Оформлення матеріалів до захисту МКР	11.2022	

Студент _____ Лісовський Р.Р.

Керівник роботи _____ Жуков С.О.

АНОТАЦІЯ

УДК 004.09

Лісовський Р.Р. Інформаційна технологія прогнозування концентрації нітратів у річковій воді Південного Бугу. Магістерська кваліфікаційна робота зі спеціальності 126 – інформаційні системи та технології, освітньо-професійна програма – інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2022. 126 с.

На укр. мові. Бібліогр.: 27 назв; рис.: 87; табл.: 5.

В магістерській кваліфікаційній роботі звернено увагу на проблему якості води, а саме концентрацію нітратів у річковій воді Південного Бугу. Об’єктом досліджень є процес прогнозування вмісту нітратів у водах річки Південний Буг. За допомогою запропонованих технологій можна аналізувати, дані, які отримано із датасету з сервісу Kaggle, та прогнозувати концентрацію нітратів. В наслідок чого, було реалізовано інформаційну технологію, яка дозволяє прогнозувати концентрацію нітратів у річковій воді Південного Бугу за певний період часу. Галузь застосування – екологічні установи та організації, які займаються аналізом якості водних ресурсів.

Ілюстративна частина складається з 6 плакатів.

У розділі економічної частини розглянуто питання про доцільність розробки та впровадження інформаційної технології аналізу та прогнозування концентрації нітратів у річковій воді.

Ключові слова: інформаційна технологія, моніторинг, якість водних ресурсів.

ABSTRACT

Lisovsky R.R. Information technology for forecasting the concentration of nitrates in the river water of the Southern Bug. Master's qualification thesis on specialty 126 - information systems and technologies, educational and professional program - information technologies of data and image analysis. Vinnytsia: VNTU, 2022. 126 p.

In Ukrainian speech Bibliography: 27 titles; Fig.: 87; tab.: 5.

In the master's thesis, attention is paid to the problem of water quality, namely the concentration of nitrates in the river water of the Southern Bug. The object of research is the process of forecasting the content of nitrates in the waters of the South Bug River. With the help of the proposed technologies, it is possible to analyze the data obtained from the dataset from the Kaggle service and predict the concentration of nitrates. As a result, information technology was implemented that allows you to forecast the concentration of nitrates in the river water of the Southern Bug for a certain period of time. The field of application is environmental institutions and organizations engaged in the analysis of the quality of water resources.

The illustrative part consists of 6 posters.

In the section of the economic part, the issue of the feasibility of developing and implementing information technology for analyzing and forecasting the concentration of nitrates in river water is considered.

Keywords: information technology, monitoring, quality of water resources.

ЗМІСТ

ВСТУП

Актуальність теми. На сьогоднішній день тема екології стоїть дуже гостро. Є досить актуальною, тому що з кожним роком рівень життя та здоров'я людей погіршується. З'являються все нові і нові хвороби. До цього призводить забруднення планети, а саме повітря, ґрунту та води. Якість води сильно впливає на стан здоров'я людей. Виливаючи отруйні відходи, орошаючи поля добривами, людина сильно забруднює річкові та підземні води, які в подальшому потрапляють до споживання.

Дивлячись на дану проблему, виникає необхідність у створенні інформаційної технології прогнозування забруднення нітратами річки Південний Буг. Вирішення даної проблеми є досить актуальним, є потреба в створені точнішої, в порівнянні з аналогами, системи для прогнозування нітратів у річковій воді Південного Бугу.

Мета і завдання роботи. Метою роботи є підвищення точності прогнозування концентрації нітратів у воді річки Південний Буг на основі заданого датасету з використанням моделей часових рядів.

Щоб досягнути поставлену мету, потрібно виконати такі задачі:

- проаналізувати предметну область, зробити розвідувальний аналіз якості річкових вод України;
- зробити попередній аналіз даних, за та вибрати оптимальну модель для прогнозування концентрації нітратів у річковій воді;
- розробити інформаційну технологію для прогнозування концентрації нітратів у річковій воді.

Об'єктом дослідження магістерської кваліфікаційної роботи є процес аналізу даних та прогнозування концентрації нітратів у воді басейну річки Південний Буг на території Вінницької області.

Предметом дослідження магістерської кваліфікаційної роботи є методи машинного навчання, які ґрунтуються на сучасних інформаційних технологіях щодо аналізу та прогнозування концентрації нітратів у воді.

Новизна одержаних результатів. Дістала подальший розвиток інформаційна технологія прогнозування концентрації нітратів у воді басейну річки Південний Буг на території Вінницької області, яка дозволяє, завдяки методам машинного навчання, підвищити точність цього прогнозування. Це було досягнуто за рахунок використання складніших бібліотек для попереднього аналізу даних та складніших моделей з використанням часових рядів.

Практичне значення. Практичне значення роботи полягає в тому, що за даними зі створів вище по течії, інформаційна технологія дозволяє отримати точний прогноз концентрації нітратів нижче по течії на певний період часу.

Апробація результатів магістерської кваліфікаційної роботи.

Результати кваліфікаційної роботи доповідались на Всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (Вінниця, 2022-2023 рр.)».

Публікації результатів магістерської кваліфікаційної роботи.

Під час виконання магістерської кваліфікаційної роботи опубліковано тези у збірнику матеріалів Всеукраїнської конференції «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, 2022-2023 рр.)" [1].

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ОБ'ЄКТУ ДОСЛІДЖЕНЬ

1.1 Опис об'єкта досліджень

Вода є однією з найпотрібніших речовин для існування живих організмів. Без неї не існували б тварини, рослини і навіть деяким бактеріям потрібна рідина для існування. Тому великою перевагою є те, що на землі води є вдосталь для існування живих організмів. Проте є і такі місця, де є нагальна проблема з постачанням. Найчастіше, це регіони Африки, де живуть племена, в яких немає доступу до води через засуху, або ж інші кліматичні умови. Тому людство активно займається проблематикою постачання цієї живильної рідини в місця, де вона є в дефіциті [2].

Проте, запаси прісної води зосереджені не тільки в річках, озерах та ставках, є ще так звані підземні води, льодовики, та деяка частина зосереджена в атмосфері. Її можна побачити, у вигляді туману, або ж ранкової роси, яка найчастіше випадає на рослини влітку. На рисунку 1.1, можна побачити, який відсоток об'єму займає вода, та розподіл води на планеті.

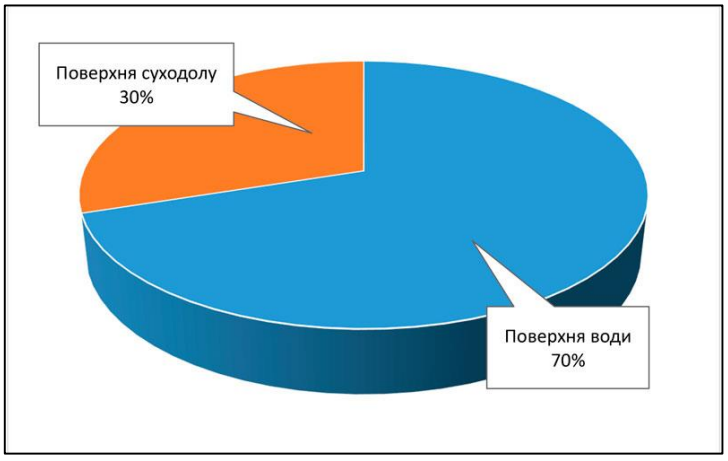


Рисунок 1.1 – Співвідношення води та суші

На планеті Земля, 70% поверхні займає водяний покрив. На підрахунок, це приблизно 1.5 млрд. кубічних метрів. Проте придатною для споживання рослин, тварин, та інших живих організмів є лише 2 - 3%, яка є прісною (рис. 1.2).

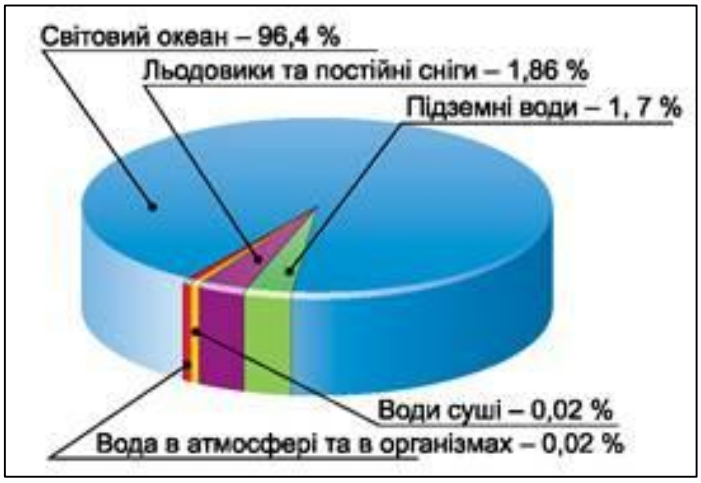


Рисунок 1.2 – Співвідношення води на планеті

На рисунку 1.3, можна побачити діаграму розподілу прісної і солоної води.

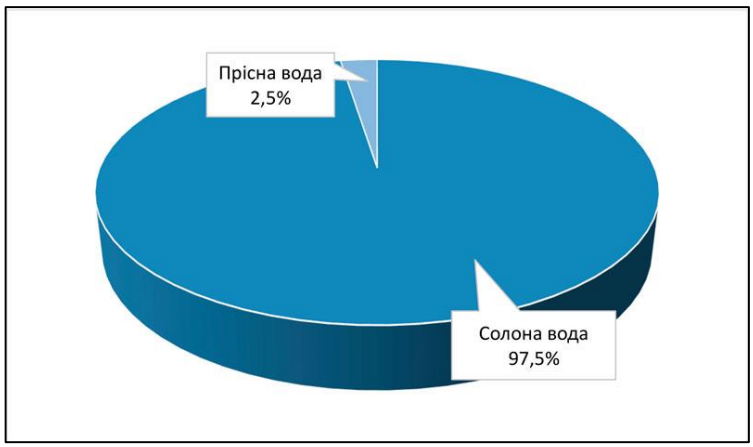


Рисунок 1.3 – Розподіл води на планеті

Прісні водоймища поділяють на статичні, та на такі, які безперервно відновлюються. До статичних можна віднести: деякі озера, підземні води, та льодовики. Тобто вони не зазнають змін щорічно. До безперервно відновлюваних ж віднесено всі інші водоймища. Наприклад, для відновлення річки достатньо в середньому близько 20 – 25 днів. Проте в останній час питання забруднення прісних водойм набуває досить високого значення, насамперед через те, що досить багато підприємств не хочуть витрачати кошти на утилізацію шкідливих хімікатів, які виділяються під час виробництва. Вихід вони знаходять у тому, щоб зливати всі нечистоти у річки та озера, які знаходяться поблизу [2].

Також, значною мірою мають негативний вплив і фермерські господарства. Це зумовлено тим, що для того щоб звільнити свій врожай від шкідників, фермери використовують пестициди, якими окропляють поля та городи. В свою чергу, ці речовини після дощу потрапляють у ґрунт, а звідти у підземні води, які впадають у річки. Таким чином відбувається досить сильне забруднення.

У наш час існує два види використання водних ресурсів. До них відносять водоспоживання, та водовикористання. В першому випадку, вода вилучається для певних промислових потреб, таких як зрошування полів, та іншого використання у господарстві. В другому випадку, рідину використовують для спорту, туризму, водовідведення, та теплової енергетики. Тобто вона не вилучається [2].

Якщо ж говорити про Україну, то до основних споживачів відносять промисловість. Вона займає 48% споживання води і являється основним споживачем. Найчастіше використовують для металургії, хімічної промисловості, та енергетичної діяльності. Оскільки наша країна є досить сильно розвинутою в аграрному напрямку, то відповідно 40% споживання займає сільське господарство. Інші ж 12% використовують для житлово-комунальних цілей.

На даний момент забруднення водних ресурсів досягло досить серйозного рівня небезпеки, і стало глобальною проблемою. Втрата якості води, несе за собою більші проблеми ніж кількісне виснаження. Якщо ж привести приклад, то 1 метр кубічний стічних вод, який впадає в річку, робить непридатним 40 – 45 кубічних метрів води.

В результаті халатного відношення до якості води, в океан щорічно потрапляє велика кількість твердого сміття, таких як пластмаси. Період півроспаду даних речовин є досить довгим, тому часто можна помітити в новинах репортажі про те що викид сміття спричинив катастрофу з вимиранням того чи іншого виду тварин, чи рослин. Також внаслідок виливу великої кількості отрутохімікатів, таких: пестициди, мінеральні добрива, ртуті, нафти страждає

досить велика територія світового океану. До прикладу, найбільш забрудненими нафтою територіями рахуються ті, що лежать на шляхах її перевезення. До них відносяться прилеглі території до Африки, та Америки. За статистикою, близько 800 млн людей не мають доступу до питної води [2].

Наступне, на що хотілося б звернути увагу, це те, що здоров'я людини залежить цілком і повністю від деяких факторів. Так, до них можна віднести: здорову і правильну їжу, чисту та придатну для споживання воду, та свіже повітря. Ще з дитинства нам відомо, що тіло людини складається на 60% з рідини, якою є вода. Тому досить критичним є постійно поповнювати свій запас. Тому що він постійно витрачається на потовиділення, виділення, та для відводу шкідливих речовин в результаті травлення. Зрозуміло, що в такому випадку якість води грає значну роль в формуванні імунітету та здоров'я цілком. Також вода потрібна для гарної розумової активності нашого мозку і для генерації необхідної кількості енергії. За порадою лікарів, вдень потрібно споживати близько 4% від спільної маси тіла. А це означає, що на добу середньостатистична людина масою 75 кг має вживати 2 – 2.5 літри води. На рисунку 1.4 зображена таблиця, яка дає інформацію про те, скільки води потрібно споживати, в залежності від маси тіла [3].

Вага, кг	Низька фізична активність	Помірна фізична активність	Висока фізична активність
до 50	1,55	2	2,3
51-60	1,85	2,3	2,65
61-70	2,2	2,55	3
71-80	2,5	2,95	3,3
81-90	2,8	3,3	3,6
більше 90	3,1	3,6	3,9

Рисунок 1.4 – Таблиця кількості води, яку потрібно споживати впродовж дня

Якщо ж вживати таку значу кількість води, яка є забрудненою шкідливими речовинами, або ж має недостатню якість, то є великий ризик захворіти на різні серйозні хвороби. Тому можна зробити такий висновок, що якість води напряму впливає на процеси які відбуваються всередині людського тіла, і відповідно відображається на стані здоров'я. В наслідок цього, можна сказати, що якість води в річках досить сильно впливає на стан здоров'я населення [3].

Якщо ж брати до уваги саме Вінницьку область, то час від часу можна побачити погіршення якості води в річні Південний Буг. Це є критичним, тому що в переважній більшості, звідти береться вода для постачання в більшість міст, та сіл. Така зміна якості обумовлена тим, що вздовж річки розташовані підприємства, та господарства, які зливають шкідливі хімікати у воду. Як було раніше зазначено, після зрошування полів добривами, хімікати потрапляють у підземні води, а звідти у річку. Як наслідок, можна побачити, що відкривши кран вода має жовтувате забарвлення, та досить неприємний запах. Також від потрапляння пестицидів та добрив, у річковій воді відбувається аномальний ріст водоростей. Що також призводить до забруднення [3].

На даний час однією з найбільш актуальних проблем є постачання у прилеглі міста чистої води та якісної води, яка б відповідала вимогам, які можуть надаватись до питної води.

На жаль в більшій кількості, очисні споруди не здатні якісно відфільтрувати воду, яка подається в будинки через водопостачання. Внаслідок цього, відкривши кран відчувається неприємний запах, та зміна забарвлення з прозорого на жовтуватий колір. Іноді, якщо залишити в склянці, навіть може випадати осад (рис. 1.5).

Рисунок 1.5 – Погіршення якості води

Хоча більша частина населення не використовує її як питну, проте знаходять люди, які використовують таку воду в господарських цілях. А саме для гігієни, та навіть приготування їжі. Як наслідок, спостерігається така тенденція, що в певний період збільшується захворюваність жителів області.

Також, однією з серйозних проблем являється те, що щоб очистити воду, очисні організації вдаються до хлорування останньої, що в свою чергу ні до чого хорошого не призводить. А навпаки, внаслідок вживання хлорованої рідини, в організмі людини створюються небезпечні сполуки [4].

Основними забруднювачами водойм є:

– Відходи промисловості – в переважній більшості це миючі засоби (рис.1.6);

Рисунок 1.6 – Побутові відходи

– Тверді відходи – речовини, які є нерозчинними у воді (рис. 1.7);

Рисунок 1.7 – Тверді відходи

– Стічні води – до вмісту входять: мінеральні солі, нафтопродукти, кислоти та різні мікроорганізми (рис. 1.8);

Рисунок 1.8 – Стічні води

– Відходи фермерської діяльності – пестициди, добрива та нітрати (рис.1.9);

Рисунок 1.9 – Фермерські відходи

Вживання води, в якій наявні нітрати, веде до досить серйозних проблем, одним з яких є водно-нітратна метгемоглобінемія. Суть полягає в тому, що відбувається кисневе голодування тканин. Досліджено, що саме дитячі

організми, які віком до 3х років є дуже чутливими до високого вмісту нітратів у воді. Статистика говорить, що 7 – 9% випадків, нітратне отруєння закінчується смертю [4].

Дивлячись на це, доцільним буде розглянути, та дослідити, не тільки питні водозабори м. Вінниці, а й водозабори, які знаходяться поблизу м. Ладижин. Тому що саме там знаходиться спортивно-оздоровчий табір ВНТУ під назвою «Супутник» (рис. 1.10).

Рисунок 1.10 – Табір «Супутник»

1.2 Огляд існуючих методів аналізу якості вод

Розглянуто ноутбук в Kaggle, який проводить аналіз концентрації Амонію NH4 у воді [5].

Даний ноутбук це досить гарне та просте рішенням для проведення аналізу даних. Останні в свою чергу, беруться з певних станцій, які розташовані вздовж течії Південного Бугу. Провівши аналіз даних, робиться прогноз, який вказує на те, коли вода в річці зазнає найбільшого забруднення. Дані витягуються з порталу «Моніторинг та екологічна оцінка водних ресурсів» [6]. Інтерфейс порталу зображено на рисунку 1.11.

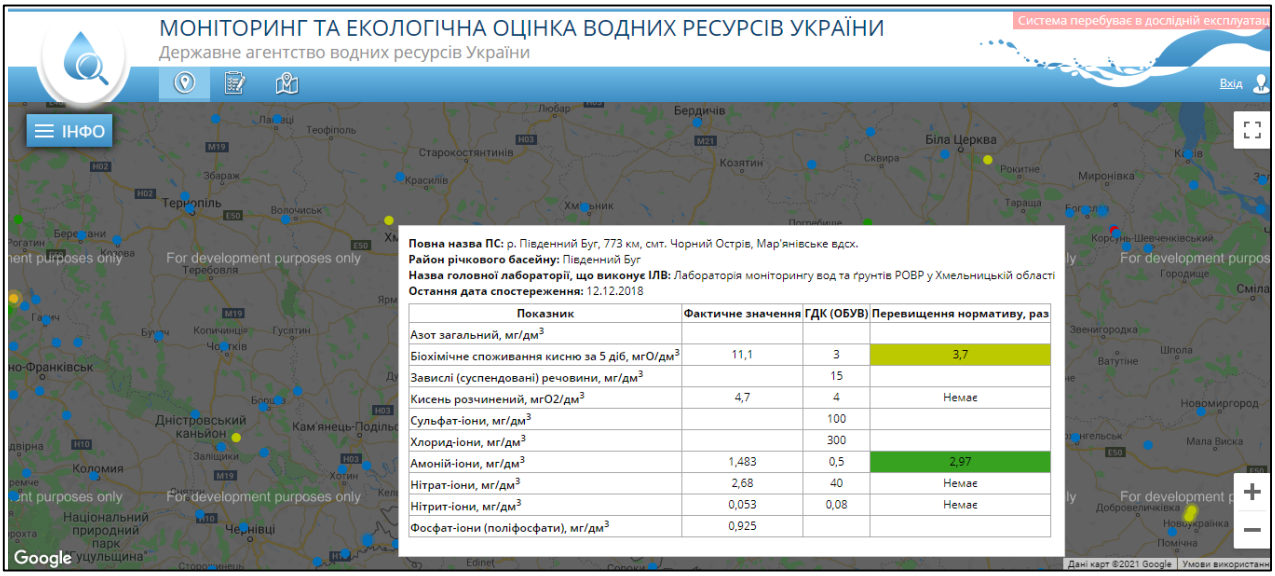


Рисунок 1.11 – Інтерфейс порталу

Дивлячись на зображення, одразу все зрозуміло, тому що все досить інформативно. Достатньо лише навести курсор миші на потрібну ділянку, одразу ж генерується інформація, яка відповідає стану води.

В цьому ноутбуці реалізований аналіз частина річки Південний Буг, яка протікає через Вінницьку область. На наступних рисунках зображені області на карті, з яких беруться дані для аналізу (рис.1.12, 1.13).

Рисунок 1.12 – Область аналізу даних

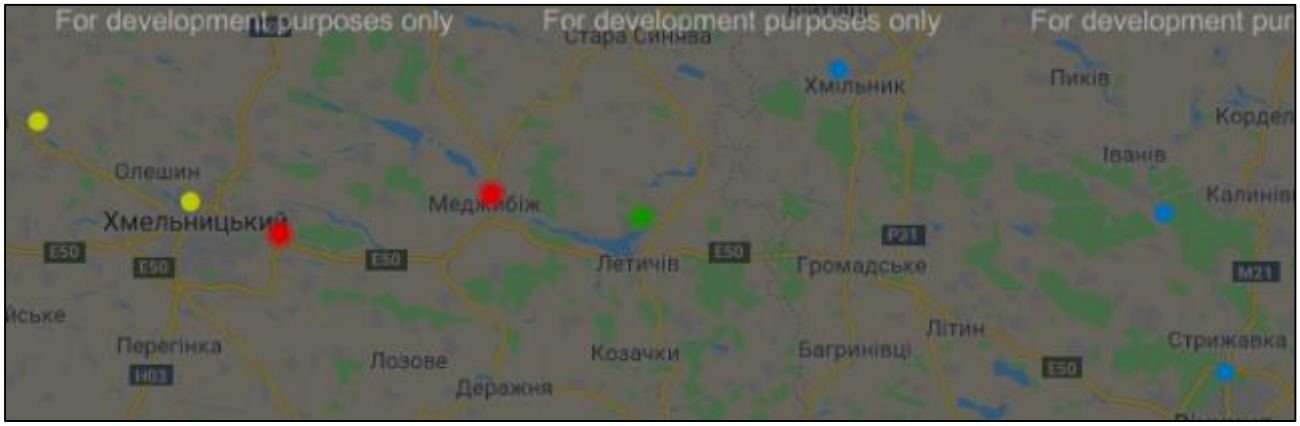


Рисунок 1.13 – Місця з яких беруться дані

Як видно на карті, річка тече з ліва направо. Останнім постом водозабору є водоканал міста Вінниця. Звідти і береться вода, яка постачається у будинки вінничан. Дані були оброблені, та відтворені у вигляді датасету. Дата сет підтягується та підключаються до ноутбуку (рис.1.14).

```
2]: # Download training data
    train = pd.read_csv('/kaggle/input/ammonium-prediction-in-river-water/train.csv')
```

TASK: Display the first 5 rows of the training dataframe.

```
3]: # Display the first 5 rows of the training dataframe.
    train.head()
```

3]:

	Id	target	1	2	3	4	5	6	7
0	0	1.10	0.69	1.04	NaN	NaN	NaN	NaN	NaN
1	3	0.41	0.71	0.72	NaN	NaN	NaN	NaN	NaN
2	4	1.70	2.21	2.21	NaN	NaN	NaN	NaN	NaN
3	5	0.62	0.60	0.68	NaN	NaN	NaN	NaN	NaN
4	6	0.60	0.60	0.90	NaN	NaN	NaN	NaN	NaN

Рисунок 1.14 – Підключення даних

Після приєднання даних, проводиться їх аналіз за допомогою моделі лінійної регресії Linear Regression. Наступним кроком, видно якою ж є точність використаної моделі. Вона становить 80.1% (рис.1.15).

```
Accuracy of Linear Regression model is 80.1%
```

Рисунок 1.15 – Точність моделі лінійної регресії

Далі проводиться наступний аналіз даних, та виведення їх у вигляді графіка. На ньому можна побачити червону лінію, яка відображає границю допустимого вмісту амонію у воді. Норма, яка є допустимою становить 0.5% (рис. 1.16). Дивлячись на графік робиться висновок, що вміст амонію в воді перевищує допустиме значення в окремих частинах аналізованої ділянки.

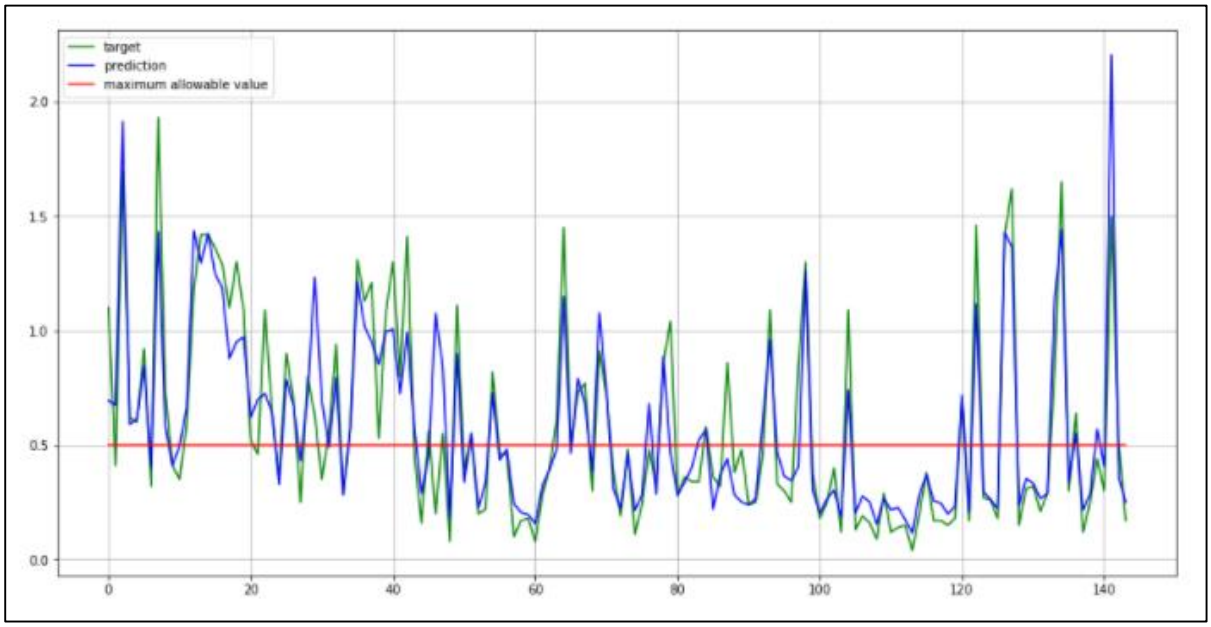


Рисунок 1.16 – Графік вмісту амонію в воді

1.3 Огляд та вибір інформаційних технологій для розробки системи прогнозування концентрації нітратів у річковій воді

Для початку роботи з даними, найперше що потрібно зробити, це їх проаналізувати.

Логічні дії, які є послідовними, які інтерпретують дані на статистичні форми, які згодом будуть потрібні для виведення певного висновку, називають аналізом [7]. Його поділяють на такі етапи:

- Етап збору;
- Етап роботи над ними;
- Етап аналізу;
- Етап пояснення результатів.

Характеристики, які будуть використовуватись для подальшого аналізу, отримуються на етапі обробки. Їх використовують:

- Для того, щоб обчислити характеристики;
- Для перевірки певних гіпотез;
- Для перевірки стохастичності даних;
- Для того, щоб виявити і видалити аномальні спостереження.

При проведені етапу спостереження, дані представляють у більш зручному вигляді для аналізу. Найчастіше це діаграми, графіки та таблиці.

Кореляційний аналіз – виявляє залежність між змінними, та показує, чи є статистична залежність між випадковими величинами [8].

Поділяється на такі завдання:

- Оцінює коефіцієнти кореляції за вибірковими даними;
- Перевіряє вибіркові коефіцієнти кореляції на предмет значущості;
- Оцінює, на скільки близький до лінійного виявлений зв’язок;
- Створює для коефіцієнтів кореляції довірчий інтервал.

Метод, який аналізує результати, залежні від якісних ознак називається дисперсійний аналіз [9].

Задача даного методу полягає у тому, щоб дослідити вплив окремих факторів на мінливість середніх. Виявляється досить ефективним, якщо досліджуються декілька факторів.

Аналіз часових рядів – це вивчення сукупності математичних та статистичних методів, які потрібні для виявлення структури часових рядів, для їх прогнозування. Прогноз, який отримується використовується для прийняття рішень [10].

Виділяють такі цілі часових рядів:

– Ціль природи ряду;

– Ціль прогнозу.

Щоб мати змогу застосувати дані цілі, модель ряду потрібно ідентифікувати, та формально описати.

Регресійний аналіз – це насамперед розділ статистики, який виявляє залежність однієї величини від іншої, і спрямований в напрямку методів [11]. Цей метод використовують не для того, щоб з’ясувати чи істотний зв’язок, а для пошуку моделі зв’язку. Найчастіше використовується, якщо у деякій комбінації змінних виражається відношення між ними. Така комбінація використовується для того, щоб передбачити таке значення, яке може приймати змінна.

Головним завданням даного виду аналізу є визначення того, як окремі фактори впливають на результативний показник.

Для вирішення задачі, яка стоїть при виконанні даної роботи, доцільно буде використати регресійні моделі. Тому що для передбачення концентрації нітратів у воді Південного Бугу, маючи набір даних, потрібно буде передбачити значення певної ознаки в майбутньому часі. А саме, на скільки зміниться рівень концентрації нітратів у воді через певний період часу.

Оскільки, головною ідеєю роботи є системний аналіз існуючих даних, побудова графіків, діаграм а також використання моделей штучного інтелекту,

машинного навчання, та обчислення часових рядів, було вирішено обрати мову програмування Python.

Чому ж саме Python? Все досить просто, за останній час ця мова програмування набула досить великої популярності за рахунок того, що його застосування можливе на всіх відомих платформах, крім мобільних пристроїв [12].

На протязі останнього відрізку часу, дана мова програмування стала більш відомою за рахунок Machine Learning та Data Science, а саме, завдяки гнучкості, яка саме так потрібна в даному сегменті.

Можна виділити такі ознаки, в яких мова Python переважає :

- Простий синтаксис через те, що розробниками було прибрано все лишнє, тому є досить легким для читання;
- Кросплатформеність, тобто працювати можна на різних платформах ;
- Застосовується в широкому спектрі розробки. За допомогою Python розробляють додатки, ігри, використовують для машинного навчання та математичних підрахунків;
- Має в наявності велику кількість бібліотек, тому не потрібно вигадувати щось нове, достатньо просто знайти потрібну бібліотеку для вирішення задачі.

Також, є досить багато джерел та інформації, за якими можна вивчати дану мову програмування.;

Динамічна типізація – найчастіше використовують новачки, тому що є можливість для спрощення коду, а значить і зменшення кількості помилок при написанні програм;

Підключення бібліотек C – для поліпшення швидкості роботи, є можливість підключення попередньо написаних бібліотек на C, та можливість компіляції коду в байт-код., що дозволяє поліпшити роботу програми.

Також, великий об’єм готових бібліотек спрощує роботу над проектами.

NumPy – це бібліотека, яка використовується в Python, для підтримки:

- Багатовимірних масивів;

- Математичних функцій.

NumPy, призначена для реалізації і роботи з багатовимірними масивами, та їх оптимізації [13].

Pandas – бібліотека, яка використовується для обробки і аналізу даних. Працює в парі з бібліотекою NumPy. Є інструментом більш низького рівня. Дана бібліотека дає змогу маніпулювати числовими рядами та таблицями. Найчастіше використовується для моделювання даних та їх аналізу. А також для збору та їх очистки [14].

Pandas_profiling – використовують для розвідувального аналізу даних. Найчастіше використовують для збору даних про інформацію яка досліджується.

Matplotlib – бібліотека, яка виступає гарним рішенням для візуалізації даних. Зображення отримані бібліотекою часто використовують в наукових публікаціях, інтерактивній графіці, веб додатках, де є потреба в побудові діаграм та графіків [15].

Пакет бібліотеки має змогу візуалізувати багато видів зображень:

– Діаграми;

– Стовпчасті діаграми;

– Спектральні діаграми.

– Графіки;

– Контурні графіки.

Sweetviz – це бібліотека Python, яка за допомогою всього лише двох рядків коду генерує зручні звіти з візуалізацією для використання в розвідувальному аналізі. За допомогою цієї бібліотеки є можливість швидко створити досить докладний звіт з характеристик, які представлені в наборі даних.

Можливостями бібліотеки є: цільовий аналіз, порівняння двох частин дата сету, порівняння двох незалежних дата сетів. Також є можливість виявляти кореляцію та асоціації. Також є можливість створення звітів в форматі HTML.

Для запуску бібліотеки, достатньо прописати команду «pip install sweetviz», та після встановлення модуля, підключити його до проекту за допомогою команди «import sweetviz as sv» [16].

Autoviz – бібліотека, яка застосовується в переважній більшості для візуалізації зв’язку між даними. Має змогу знаходити найефективніші функції та за допомогою лише одного рядка коду створює творчу візуалізацію [17].

Scikit-learn – бібліотека, яка використовується в машинному навчанні. За допомогою даної бібліотеки можна створювати велику кількість алгоритмів для кластеризації, класифікації, регресії таких як лінійна регресія, Random Rorest та градієнтний бустинг. Бібліотека також працює в парі з NumPy. І можна сміливо сказати, що вона є однією з популярних бібліотек які використовуються в машинному навчанні [18].

1.4 Огляд існуючих методів та підходів щодо прогнозування концентрації нітратів у річковій воді

Існує, щонайменше 5 методів машинного навчання [19]. До них відносять:

- Регресійні методи;
- Нейронні мережі;
- Бустинг та багінг;
- Random Forest;
- Дерева прийняття рішень.

Для передбачення концентрації нітратів у воді Південного Бугу, доцільно буде використати методи регресії. Оскільки їх основною задачею є передбачення її значення ознаки, а не тільки опис впливу ознак. Тому, доцільно буде використати алгоритми дерев рішень.

Дерева рішень – це засоби, які використовуються в машинному навчанні, статистиці або аналізі даних для прийняття певних рішень. Ціль полягає в тому, що створюється модель, зможе передбачити значення цільової змінної, яка на основі декількох змінних на вході [20].

Дерева бувають двох типів:

- Дерева для класифікації;

– Дерева для Регресії.

Також є методи, за допомогою яких можна створювати більше одного дерева рішень [21].

До них відносять:

– Беггінг;

– Random Forest;

– Бустинг.

Для вирішення нашої задачі по передбаченню нітратів в річковій воді, можна використати такі методи, як Random Forest та XGBoost [22].

Також, на даний момент є ще один популярний метод машинного навчання, який називається Linear Regression (лінійна регресія). Дану модель, найчастіше застосовують в статистиці. Суть в тому, що вона виявляє залежність однієї змінної, від іншої з лінійною функцією залежності [23].

Support Vector Machines, або ж метод опорних векторів – використовується для задач класифікації і регресійного аналізу. Головна ідея в тому, щоб перевести вхідні вектори у прості більш високої розмірності, та знаходження роздільної гіперплощини з найбільшим зазором у просторі. Дві паралельні гіперплощини будуються по обидва боки гіперплощини, що розділяє класи. Гіперплощиною, яка буде розділяти, буде гіперплощина, яка створює найбільшу відстань до двох паралельних гіперплощин. Алгоритм заснований на припущенні, що чим відстань між паралельними гіперплощинами, тим меншою буде середня помилка класифікатора.

Bagging regressors – ціллю є навчити на випадковій підмножині початкового навчального набору кожну регресорну модель та агрегувати прогнози. Оскільки цільова змінна є числовою, то згодом агрегація усереднюється за ітераціями. Реалізується за допомогою бібліотеки scikit-learn [24].

Також, для вирішення задачі передбачення концентрації нітратів у воді Південного Бугу, доцільно буде використати часові ряди.

Часовий ряд – це послідовність значень яка є впорядкованою в часі. Спостереження часового ряду, називають рівнями. Певному моменту часу відповідає свій рівень. Порядок розташування рівнів є характеристикою ряду і не змінюється довільно. Для того, щоб отримати багатовимірний часовий ряд, кожному моменту часу приводять у відповідність декілька значень різних показників досліджуваного об’єкта.

До основних компонентів часових рядів відносять:

– Сезонність;
– Тенденція;
– Нерегулярність;
– Циклічність.

Для того, щоб працювати з часовими рядами існує досить багато методів. Вони спрямовані на точність, та на те, щоб не допустити найменшої кількості помилок. Зараз, існує декілька методів, які є досить точними і мають непогану обчислювальну актуальність.

До таких методів відноситься:

– Експоненціальна модель згладжування;
– ARIMA/SARIMA;
– Наївна модель.

У нашому випадку, доцільно буде використати модель ARIMA.

ARIMA/SARIMA – інтегрована модель авто регресії. Ідея даного методу полягає тому, щоб залежно від попередніх значень часового ряду, побудувати часову модель. За основу беруться, переважно показники кількох місяців, уважно проаналізувавши їх, становлять подальшу модель розвитку подій [25].

Найчастіше набір даних методів використовується за умов:

– Коли ряд є стаціонарним. Це означає, що дисперсія і середня є незмінними в часі.
– Коли вхідні дані, які були отримані є одномірними. Це зумовлено тим, що набір методів передбачає для передбачення майбутніх показників використання попередніх показників.

Назвою самої моделі є абревіатура. Вона складається з:

– AR - авторегресія;

– I – інтеграцію;

– MA – ковзне середнє.

У яких же випадках модель ARIMA дає найкращий результат.

Для отримання справді добрих прогнозів необхідно для кожного дня взяти значення попередніх періодів. Причому чим більше тимчасове вікно, тим кращі результати.

Обумовлюється це тим, що для більш легкого аналізу часового ряду, та більш точного передбачення показників, потрібно більше інформації

Що таке тимчасовий ряд:

Тимчасовий ряд – це статистичні дані про показники будь-яких параметрів, які зібрані в різний період часу. При дослідженні часового ряду потрібно врахувати не лише статистичні відмінності та характеристики вибірки, а й взаємозв'язок вимірів з часом.

Стаціонарний часовий ряд. Як говорилось раніше, часовий ряд визначається послідовністю значень чи кінцевою кількістю випадкових величин. Він ділиться на два види: одновимірний та багатовимірний.

Стаціонарним є одномірний ряд, в якому є незмінними імовірнісні характеристики та показники випадкової величини. Якщо хоча б одна з характеристик змінюється в окремих відрізках часу, тоді ряд стає нестаціонарним.

Характеристика стаціонарного ряду: відсутність автокореляції; однакове середнє значення; Постійна дисперсія.

Що таке Тренд. Тренд це довготривала тенденція зміни аналізованого часового ряду. Також його називають тимчасовим трендом.

Тренди характеризуються за різноманітними рівняннями: лінійними, логарифмічними, статечними та іншими. Визначають, яким буде мати вигляд тимчасовий тренд залежно від функціональної моделі: згладжуванням початкового ряду або ж статистичними способами.

Що таке сезонність. Сезонність передбачає проміжки часу, у яких попит споживачів певну продукцію підвищується чи знижується [25].

Наприклад, одночасно попит може бути як 5%, так і в інший момент часу він зросте до 75%.

Саме поняття характеризується по-різному. У виробництві сезонністю називають нерівномірність випуску товарів, пов'язану з пори року, тобто сезоном. У маркетингу поняття визначається фактором, що впливає на збут, рекламу та діяльність компаній залежно від сезону. Але це стосується економіки, отже, всі визначення підходять.

Сезонні дані передбачають строгу структуру. У місячній статистиці з річною сезонною структурою показники для однакових місяців у різні роки мають бути взаємопов'язані між собою. Тобто взаємозв'язок має бути не тільки у конкретних спостережень протягом року, а й у спостережень з періодом, який крадений одного року. Для того, щоб зробити прогнозування з урахуванням сезонності потрібно 2 і більше року даних. Проте сезонний компонент не використовуватиметься якщо даних буде менше.

Для вирішення задач з прогнозування, також використовують ще одну модель, Facebook Prophet.

Prophet - це бібліотека від компанії Facebook, яка має відкритий код. За допомогою неї виконується прогнозування часових рядів. Як кажуть розробники, бібліотека добре працює з рядами, які мають яскраво виражені сезонні ефекти, а також мають кілька таких періодів. Prophet досить добре справляється з викидами і є стійким до відсутності даних [26].

Бібліотека багато в чому успадковує «стиль» sklearn зі своїми fit та predict. Prophet – це технологія прогнозування даних часових рядів, заснована на адитивній моделі, де нелінійні тенденції відповідають річній, тижневій та щоденній сезонності, плюс святкові ефекти, реалізована для мов програмування Python та R. Найкраще використовувати для часових рядів,які мають сильний сезонний ефект та кілька сезонних історичних даних.

Prophet добре обробляє викиди, через те що є стійким відсутніх даних та змін у тренді, і.

До головних переваг відносять :

– Працює з періодичними і неперіодичними рядами;

– Працює, коли навіть є велика кількість пропущених, або ж відсутніх даних [26];

– виконує прогнозування саме для вказаного інтервалу;

– будує зони невизначеності;

– використання як лінійної, так і логістичної моделі тренду, які можна обмежити точками зміни тренду;

– За допомогою рядів Фур’є заданого порядку, можна додавати складові сезонності;

– За допомогою даних типу «holidays», дає змогу вказувати значення і дати подій, які так або ж інакше впливали на значення у відповідній точці.

1.5 Висновки

Проведено аналіз предметної області, розглянуто аналогічні рішення наявні на даний момент, обґрунтувано доцільність створення технології для прогнозування концентрації нітратів у воді річки Південний Буг.

Провівши аналіз інформаційних технологій які дозволяють виконати поставлену задачу, вдалось визначитись насамперед з мовою програмування. Обрано мову Python, оскільки вона є досить популярною на даний момент, має велику кількість бібліотек для аналізу, обробки та прогнозування даних. Було визначено бібліотеки, методи, та моделі які будуть використовуватись для аналізу даних та прогнозування концентрації нітратів у воді річки Південний Буг.

Так, для аналізу даних буде використано бібліотеки: NumPy, Pandas, Matplotlib, Sweetviz та Autoviz.

Для прогнозування буде використано моделі Facebook Prophet, ARIMA/SARIMA, Linear Regression, Support Vector Machines, Random Forest Regressor, Bagging Regressor, XGB Regressor.

2 АНАЛІЗ ДАНИХ КОНЦЕНТРАЦІЇ НІТРАТІВ У РІЧКОВІЙ ВОДІ ПІВДЕННОГО БУГУ

2.1 Попередній аналіз даних

Для початку роботи було імпортовано пакети маніпуляцій, обробки даних та необхідні бібліотеки для створення графіків, та рисунків для наглядного відображення та аналізу даних (рис. 2.1).

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

# 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)
import pandas_profiling as pp

# Time Series - EDA and Modelling
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller

# Sweetviz
import sweetviz as sv
from IPython.display import IFrame

import warnings
warnings.filterwarnings("ignore")
```

Рисунок 2.1 – Імпорт необхідних пакетів

Хотілося б зазначити, що дані для роботи було взято з датасету «River Water Quality EDA and Forecasting» [27]. Тут містяться дані з 21 станції, про якість річкової води Південного Бугу на вміст нітратів, нітритів, фосфатів, та інших речовин, які забруднюють навколишнє середовище. Вони оновлювались в період з 2000 по 2021 рік. (рис.2.2)

date date of observation [null] 100%	NH4 NH4 2861 total values	BSK5 BOD for 5 days 2861 total values	Suspended Suspended substances 2861 total values	O2 Oxygen 2861 total values
02.03.2001	0	3.03	48.8	14.69
07.06.2001	0.02	4.02	34	10.61
10.09.2001	0.863	3.91	147	10.96
06.11.2001	0.06	2.97	71.2	13.47
12.03.2002	0.168	4.15	27	17.82
06.06.2002	0.001	7.11	74.4	19.28
15.07.2002	0.668	6.81	80	10.74
07.11.2002	0.082	3	51.4	13.08
24.02.2003	0.603	3.2	42	14.97
13.06.2003	0.058	2.8	9.3	12.3
16.09.2003	0.053	2.3	31.2	10.8
14.10.2003	0.017	4.76	80	13.77
15.03.2004	0.014	2.61	30.8	15.51
10.06.2004	0.035	3.38	59.2	10.47
06.09.2004	0.057	2.83	25.6	6.97

Рисунок 2.2 – Дані якості річкової води

На даному рисунку зображена таблиця з інформацією про дату отримання того чи іншого показника. Всього в датасеті їх 8: NH4, BSK5, NO3, NO2, SO4, PO4, CL.

Наступним кроком було завантажено дані з датасету, та відформатовано для зручності у вигляді таблиці, на якій зображено перелік постів, номер відповідає порядковому номеру вниз по течії річки Південний Буг (рис. 2.3).

	id	length	name_station
20	21	773.0	р. Південний Буг, 773 км, смт. Чорний Острів, Мар'янівське вдсх.
19	20	755.0	р. Південний Буг, 755 км, м. Хмельницький , Хмельницьке вдсх.
18	19	744.0	р. Південний Буг, 744 км, с. Копистин, нижче м.Хмельницький
17	18	711.0	р. Південний Буг, 711 км, смт. Меджибіж, Меджибіжське вдсх.
16	17	692.0	р. Південний Буг, 692 км, с. Щедрове, Щедрівське вдсх.
15	16	652.0	р. Південний Буг, 652 км, м. Хмільник, питний в/з, вище міста
14	15	607.0	р. Південний Буг, 607 км, с. Гущинці, нижче села , питний водозабір м.Калинівка
13	14	582.0	р. Південний Буг, 582 км, м. Вінниця, Сабарівське вдсх, питний в/з міста, вище міста
12	13	569.5	р. Південний Буг, 569.5 км, 500 м нижче скиду ВОКВП ВКГ "Вінницяводоканал" (1,5 км нижче греблі Сабарівського вдсх.)
11	12	537.0	р. Південний Буг, 537 км, смт. Сутиски, Сутиське вдсх., н/б`єф
9	10	413.0	р. Південний Буг, 413 км, с. Маньківка, вище села, питний в/з м.Ладижин
8	9	400.0	р. Південний Буг, 400 км, м. Ладижин, Ладижинське вдсх.
7	8	372.0	р. Південний Буг, 372 км, с. Глибочок, Глибочекське вдсх.
6	7	327.0	р. Південний Буг, 327 км, с. Ставки, кордон Вінницької та Кіровоградської обл.
5	6	316.0	р. Південний Буг, 316 км, м.Гайворон, Гайворонське вдсх.
4	5	237.0	р. Південний Буг, 237 км, питний водозабір смт Побузьке
3	4	206.0	р. Південний Буг, 206 км, м. Первомайськ, Первомайське вдсх.
2	3	153.0	р. Південний Буг, 153 км, с. Олексіївка, питний в/з м. Південно-Українськ
1	2	136.0	р. Південний Буг, 136 км, с. Олександрівка, Олександрівське вдсх.
21	22	97.0	р. Південний Буг, 97 км, м. Вознесенськ, пит.в/з м. Вознесенськ, 2 км до в'їзду у м. Вознесенськ по трасі з м. Миколаїв
10	11	50.0	р. Південний Буг, 50 км, с. Ковалівка, Південно-Бузька ЗС
0	1	0.5	р. Південний Буг, 0.5 км, м. Миколаїв, Бузький лиман, тех. в/з Миколаївської ТЕЦ (ліва частина морського порту)

Рисунок 2.3 – Таблиця з даними

2.2 Розвідувальний аналіз даних

Оскільки завданням роботи є прогнозування якості води на вміст нітратів у річковій воді Південного Бугу, доцільніше всього буде робити аналіз біля питного водозабору м. Вінниці. Тому було обрано пост №14, як цільовий, який буде досліджуватись за постами №15 та №16, які знаходиться вище по течії. Так буде можливість зрозуміти, чи є забруднювачі нижче по течії річки, які впливають на якість води (рис 3.4 – 3.5).

```
# Set id of stations
id_target_station = 14
# all_id_station as int: 1-21 (21 - river outlet, 1 - river mouth,
#                        stations numbering - against the flow of the river)
id_feature_station = [15, 16]  # if it is necessary to forecast the data,
                               # taking into account the data of other stations
```

Рисунок 2.4 – Відокремлення постів

	id	length	name_station
13	14	582.0	р. Південний Буг, 582 км, м. Вінниця, Сабарівське вдсх, питний в/з міста, вище міста
14	15	607.0	р. Південний Буг, 607 км, с. Гущинці, нижче села , питний водозабір м.Калинівка
15	16	652.0	р. Південний Буг, 652 км, м. Хмільник, питний в/з, вище міста

Рисунок 2.5 – Необхідні пости

Далі, з набору даних було виокремлено необхідні для аналізу речовини. Такі як: NO3, NH4, NO2, CL. Нітрати NO3, оскільки вони є цільовими, тобто їх вміст буде прогнозуватись, а інші, тому що вони досить добре взаємодіють з ними та між собою у воді, тому зміна концентрації однієї з них впливає на вміст інших (рис. 2.6, 2.7).

```
# Set indicator names (see dataset https://www.kaggle.com/datasets/vbmokin/wq-southern-bug-river-01052021)
# all_indicator_names as str : 'NH4', 'BSK5', 'NO3', 'NO2', 'SO4', 'PO4', 'CL'
target_indicator_name = 'NO3'
feature_indicator_names = ['NH4', 'NO2', 'CL']  # if it is necessary to forecast the data,
                                                # taking into account the data of other indicators
```

Рисунок 2.6 – Відокремлення речовин

	ds	15_NH4	15_NO2	15_CL	15_NO3	16_NH4	16_NO2	16_CL	16_NO3	14_NH4	14_NO2	14_CL	14_NO3
0	2000-01-02	31.30	30.4	26.80	2.20	2.20	2.40	0.270	0.130	0.130	8.80	8.40	7.70
1	2000-01-03	28.60	26.8	25.00	0.68	0.87	0.54	0.090	0.170	0.270	8.80	9.10	8.80
2	2000-01-08	24.10	25.8	25.80	0.37	0.25	0.14	0.240	0.150	0.160	1.50	7.00	0.90
3	2000-04-04	23.20	22.3	22.30	0.81	1.22	0.51	0.090	0.140	0.090	4.60	4.90	3.50
4	2000-04-07	23.20	23.2	20.50	0.10	0.07	0.14	0.170	0.320	0.360	2.30	2.10	1.70
...	...	...	...	...	...	...	...	...	...	...	...	...	...
232	2020-09-15	36.70	34.2	31.70	0.26	0.25	0.16	0.064	0.051	0.072	1.03	0.57	0.84
233	2020-10-06	40.77	48.5	47.00	0.38	0.28	0.30	0.037	0.038	0.052	0.24	0.44	0.96
234	2020-12-08	29.62	38.9	29.61	0.57	0.28	0.20	0.061	0.036	0.046	0.68	1.03	0.71
235	2021-03-16	36.10	30.7	36.10	1.03	1.38	2.29	0.122	0.134	6.790	10.90	8.56	7.70
236	2021-06-04	38.80	39.7	37.90	0.43	0.17	0.16	0.129	0.179	0.164	6.57	8.07	6.38

237 rows × 13 columns

Рисунок 2.7 – Датасет з відібраними показниками якості води з відповідних пунктів спостереження від 14-го до 16-го

Наступним кроком, за допомогою тесту «Augmented Dickey-Fuller (ADF) test» було перевірено ряд на стаціонарність та сезонність (рис. 2.8).

```
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 2.7 – Застосування тесту ADF

Для цього, функція була використана для самого ряду (рис. 2.9), потім для його першої різниці (рис.2.10), а далі для його другої різниці (рис. 2.11).

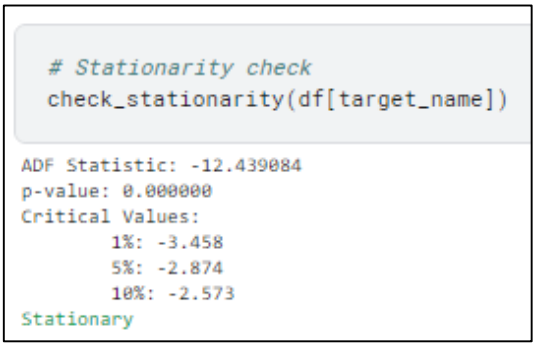


Рисунок 2.9 – Використання ф-ї для ряду

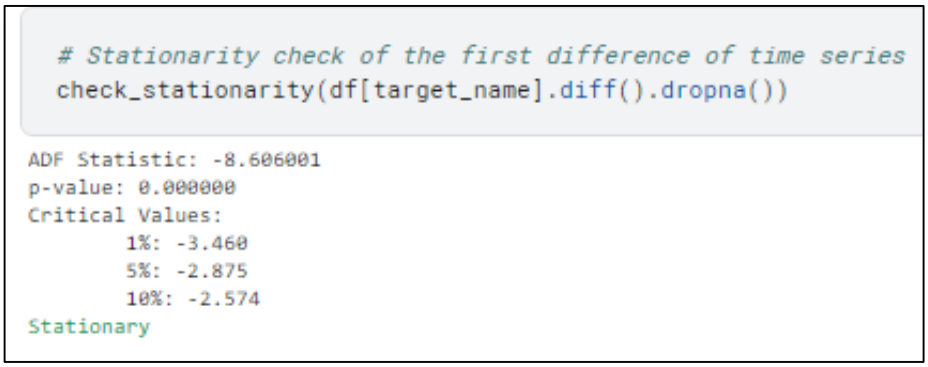


Рисунок 2.10 – Використання ф-ї для першої різниці

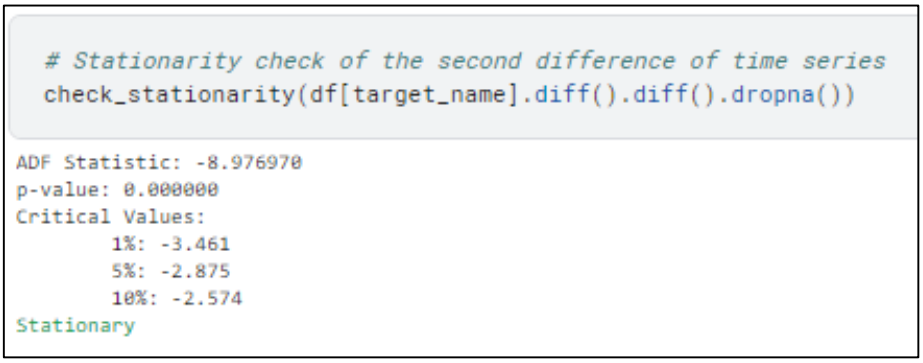
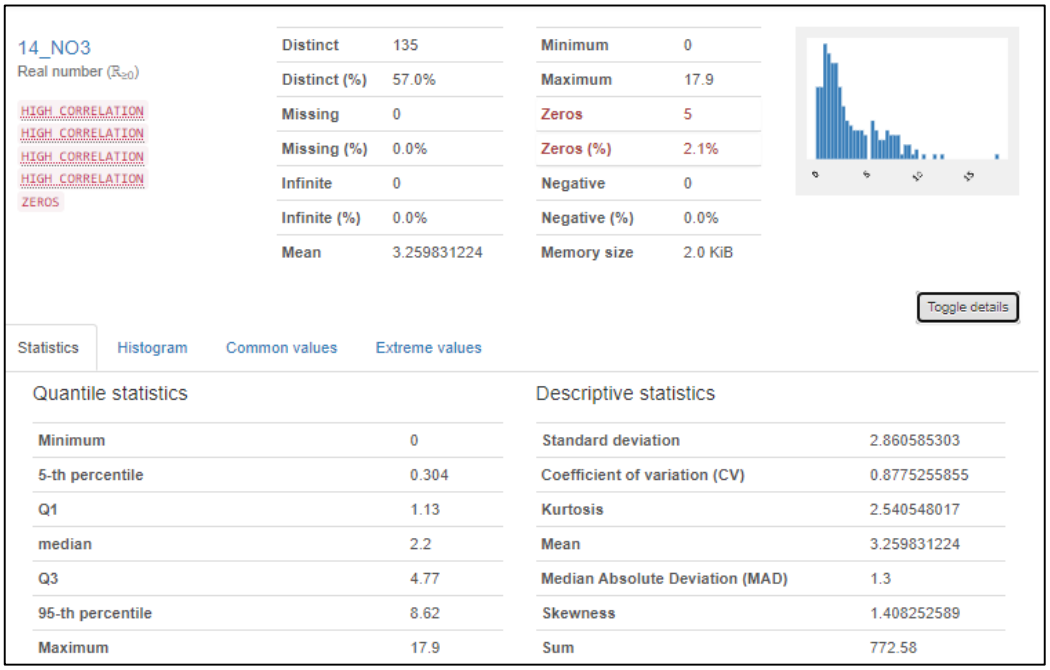


Рисунок 2.11 – Використання ф-ї для другої різниці

Сенс криється в тому, що функція спів ставляє отримане значення з порогом -5, якщо значення ряду більше, то він є не стаціонарним, якщо ж більше, то навпаки. Як видно на рисунках, ряд є стаціонарним.

Щоб більш детально розглянути дані, їх було відображено у вигляді графіків. Для цього було використано бібліотеку Pandas Profiling. Інформація про показники нітратів та хлору з поста №14 зображена на рисунках 2.12 – 2.13.



14_NO3
Real number (ℝ≥0)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
ZEROS

Distinct	135
Distinct (%)	57.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	3.259831224

Minimum	0
Maximum	17.9
Zeros	5
Zeros (%)	2.1%
Negative	0
Negative (%)	0.0%
Memory size	2.0 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

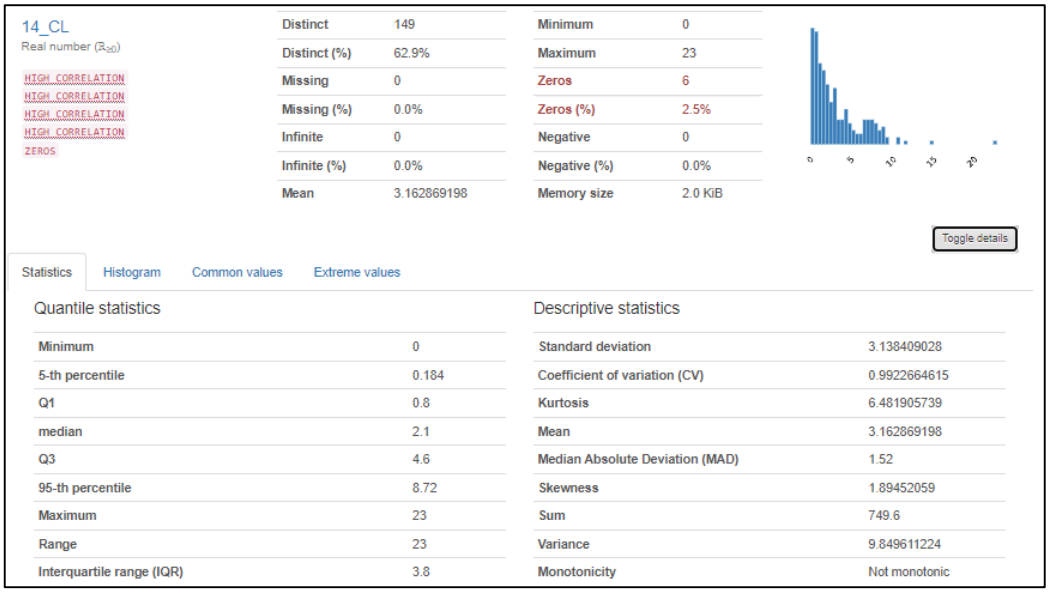
Quantile statistics

Minimum	0
5-th percentile	0.304
Q1	1.13
median	2.2
Q3	4.77
95-th percentile	8.62
Maximum	17.9

Descriptive statistics

Standard deviation	2.860585303
Coefficient of variation (CV)	0.8775255855
Kurtosis	2.540548017
Mean	3.259831224
Median Absolute Deviation (MAD)	1.3
Skewness	1.408252589
Sum	772.58

Рисунок 2.12 – Вміст нітратів



14_CL
Real number (ℝ≥0)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
ZEROS

Distinct	149
Distinct (%)	62.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	3.162869198

Minimum	0
Maximum	23
Zeros	6
Zeros (%)	2.5%
Negative	0
Negative (%)	0.0%
Memory size	2.0 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0
5-th percentile	0.184
Q1	0.8
median	2.1
Q3	4.6
95-th percentile	8.72
Maximum	23
Range	23
Interquartile range (IQR)	3.8

Descriptive statistics

Standard deviation	3.138409028
Coefficient of variation (CV)	0.9922664615
Kurtosis	6.481905739
Mean	3.162869198
Median Absolute Deviation (MAD)	1.52
Skewness	1.89452059
Sum	749.6
Variance	9.849611224
Monotonicity	Not monotonic

Рисунок 2.13 – Вміст хлору

Інформація про показники нітритів та амонію з поста №14 зображена на рисунках 2.14 – 2.15.

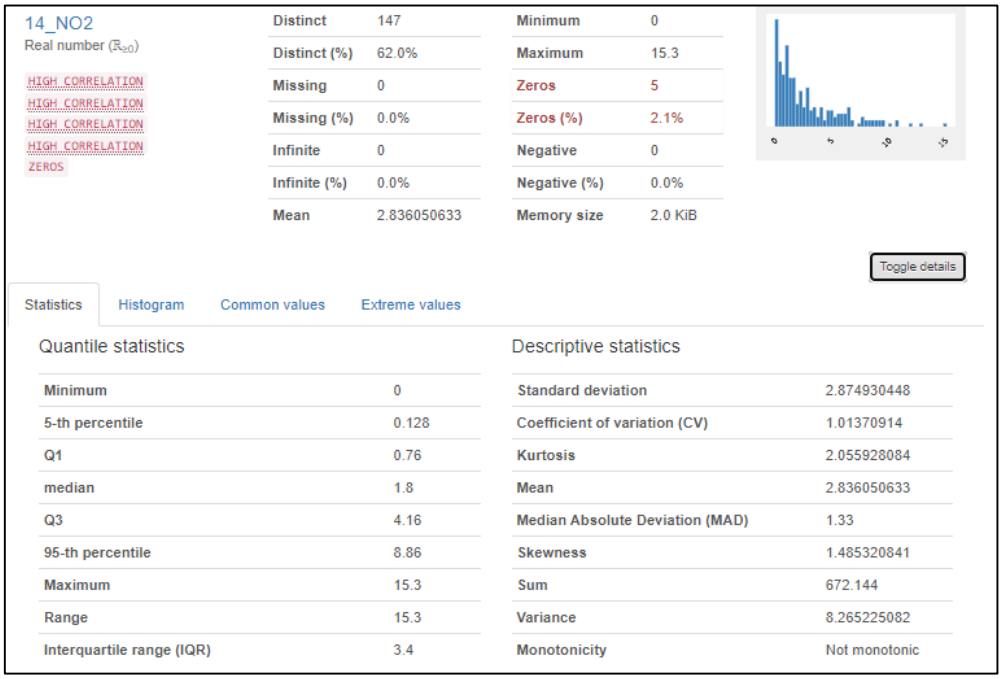


Рисунок 2.14 – Вміст нітритів

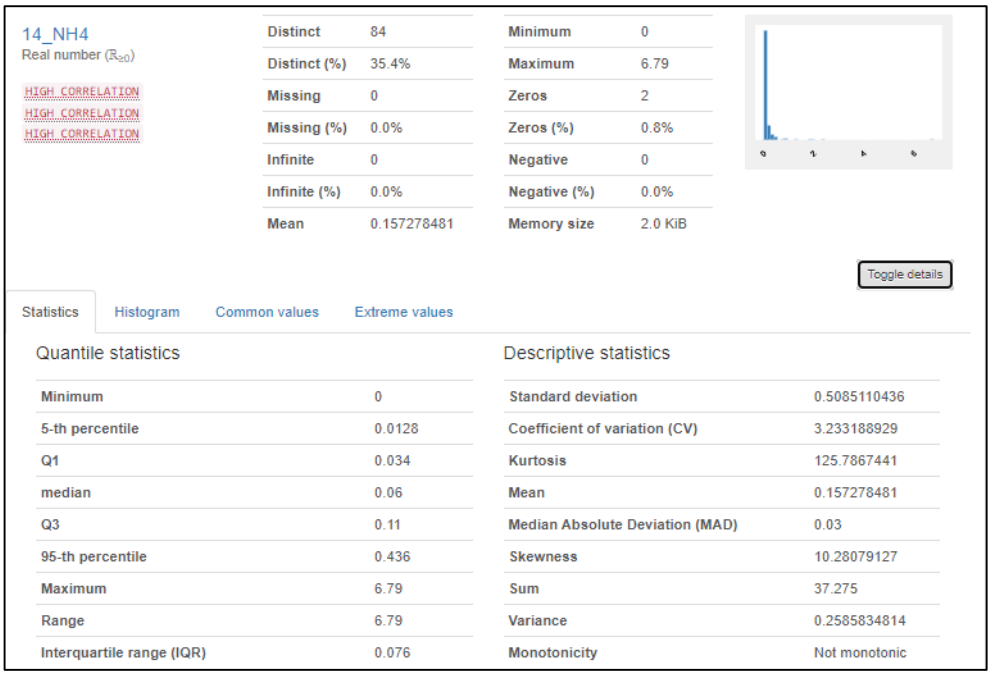
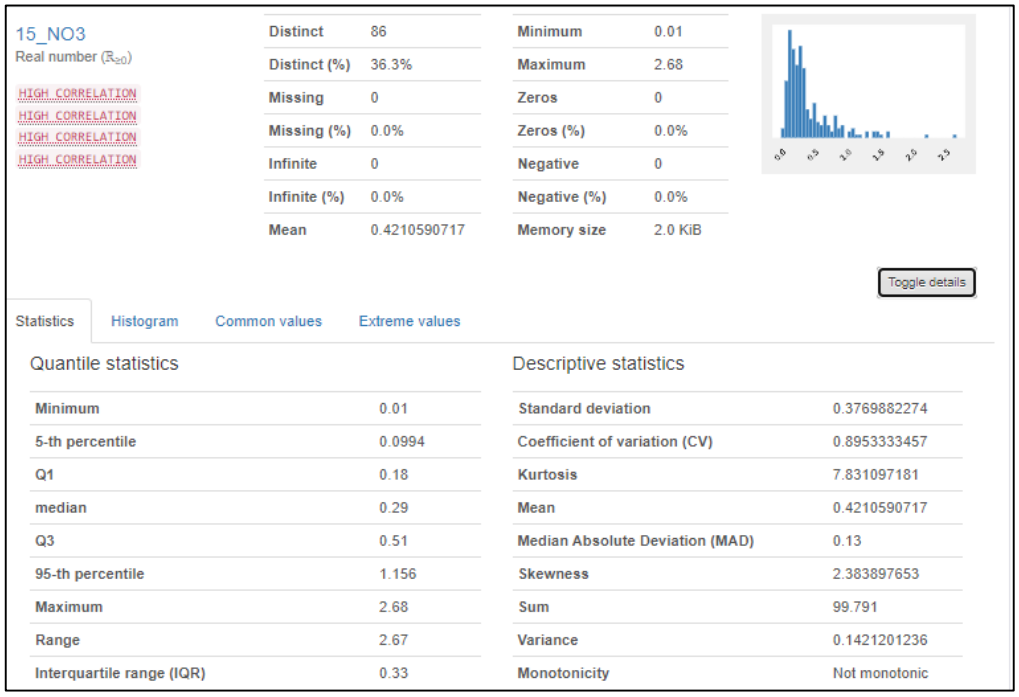


Рисунок 2.15 – Вміст амонію

Далі буде огляд інформації про вміст нітратів та хлору з поста №15 (рис. 2.16 – 2.17).



15_NO3
Real number (ℝ≥0)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	86	Minimum	0.01
Distinct (%)	36.3%	Maximum	2.68
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	0.4210590717	Memory size	2.0 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

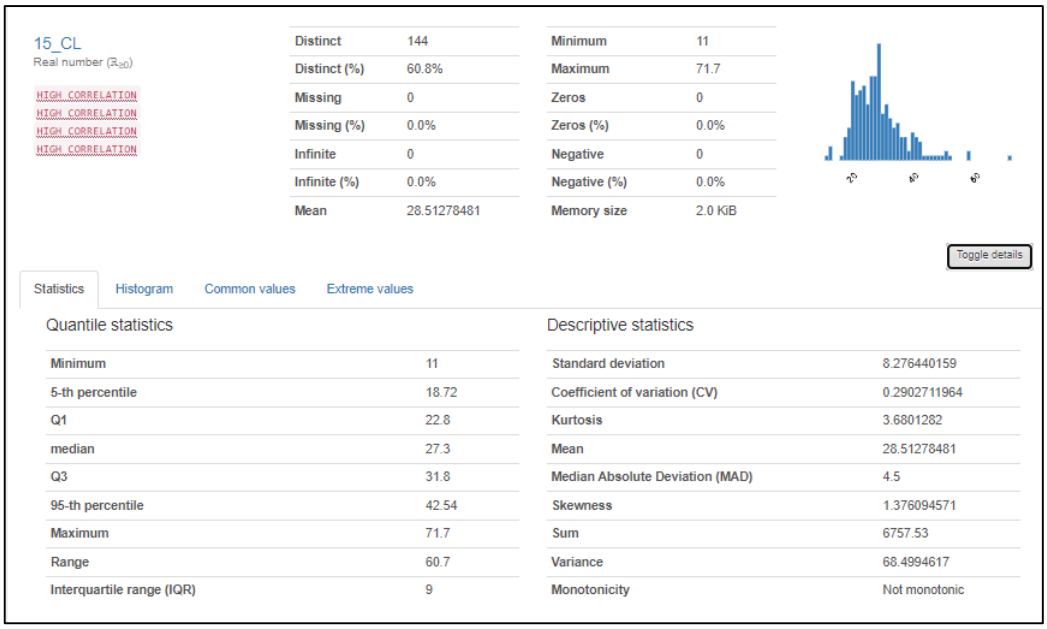
Quantile statistics

Minimum	0.01
5-th percentile	0.0994
Q1	0.18
median	0.29
Q3	0.51
95-th percentile	1.156
Maximum	2.68
Range	2.67
Interquartile range (IQR)	0.33

Descriptive statistics

Standard deviation	0.3769882274
Coefficient of variation (CV)	0.8953333457
Kurtosis	7.831097181
Mean	0.4210590717
Median Absolute Deviation (MAD)	0.13
Skewness	2.383897653
Sum	99.791
Variance	0.1421201236
Monotonicity	Not monotonic

Рисунок 2.16 – Вміст нітратів



15_CL
Real number (ℝ≥0)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	144	Minimum	11
Distinct (%)	60.8%	Maximum	71.7
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	28.51278481	Memory size	2.0 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

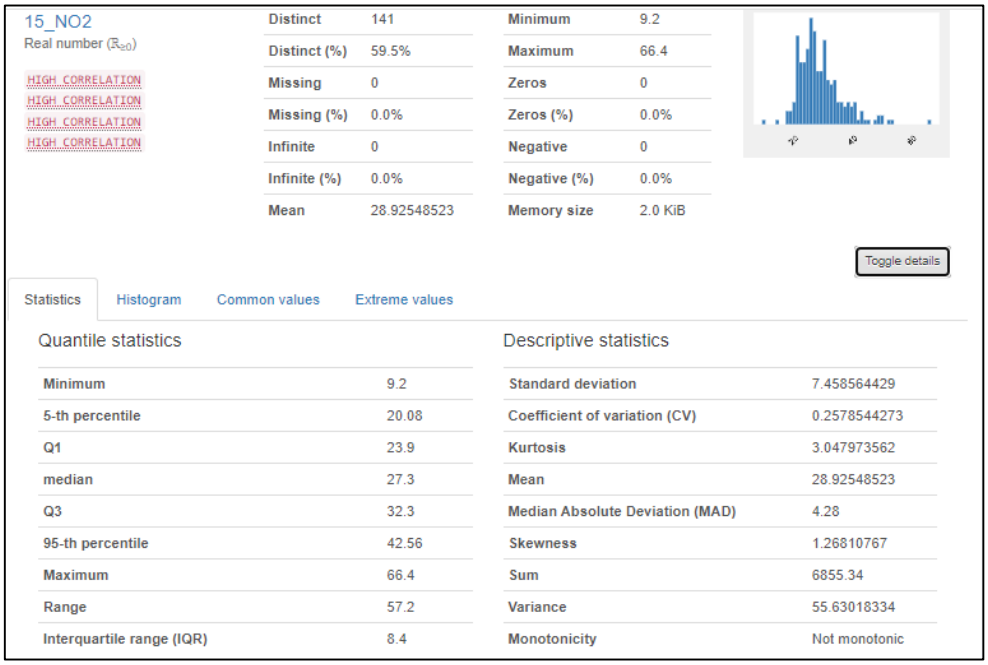
Minimum	11
5-th percentile	18.72
Q1	22.8
median	27.3
Q3	31.8
95-th percentile	42.54
Maximum	71.7
Range	60.7
Interquartile range (IQR)	9

Descriptive statistics

Standard deviation	8.276440159
Coefficient of variation (CV)	0.2902711964
Kurtosis	3.6801282
Mean	28.51278481
Median Absolute Deviation (MAD)	4.5
Skewness	1.376094571
Sum	6757.53
Variance	68.4994617
Monotonicity	Not monotonic

Рисунок 2.17 – Вміст хлору

Інформація про показники нітритів та амонію з поста №15 зображена на рисунках 2.18 – 2.19.



15_NO2
Real number (ℝ≥0)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	141	Minimum	9.2
Distinct (%)	59.5%	Maximum	66.4
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	28.92548523	Memory size	2.0 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

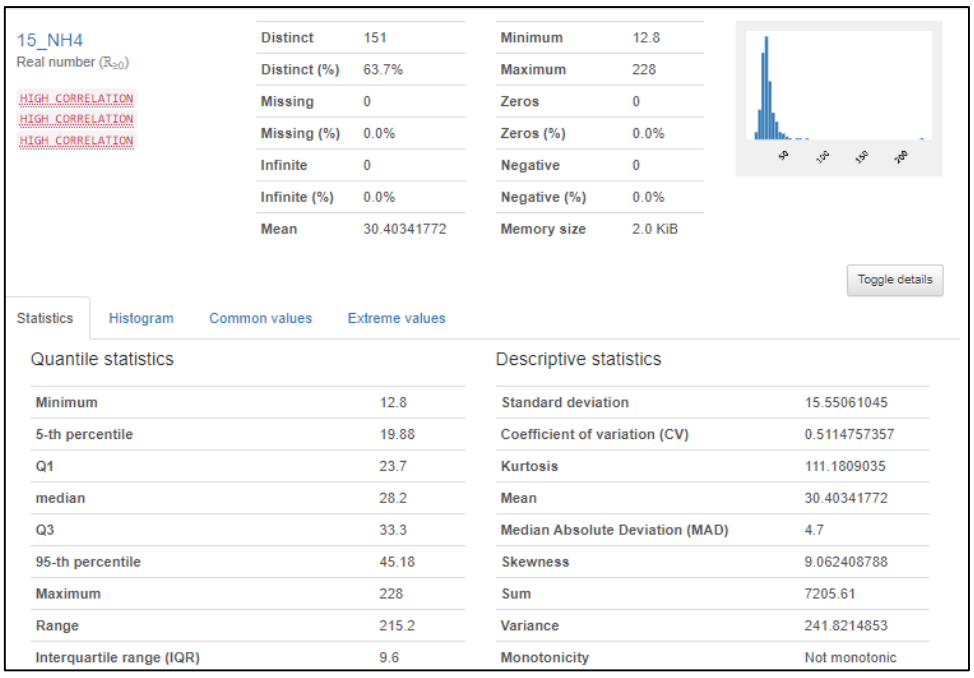
Quantile statistics

Minimum	9.2
5-th percentile	20.08
Q1	23.9
median	27.3
Q3	32.3
95-th percentile	42.56
Maximum	66.4
Range	57.2
Interquartile range (IQR)	8.4

Descriptive statistics

Standard deviation	7.458564429
Coefficient of variation (CV)	0.2578544273
Kurtosis	3.047973562
Mean	28.92548523
Median Absolute Deviation (MAD)	4.28
Skewness	1.26810767
Sum	6855.34
Variance	55.63018334
Monotonicity	Not monotonic

Рисунок 2.18 – Вміст нітритів



15_NH4
Real number (ℝ≥0)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	151	Minimum	12.8
Distinct (%)	63.7%	Maximum	228
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	30.40341772	Memory size	2.0 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

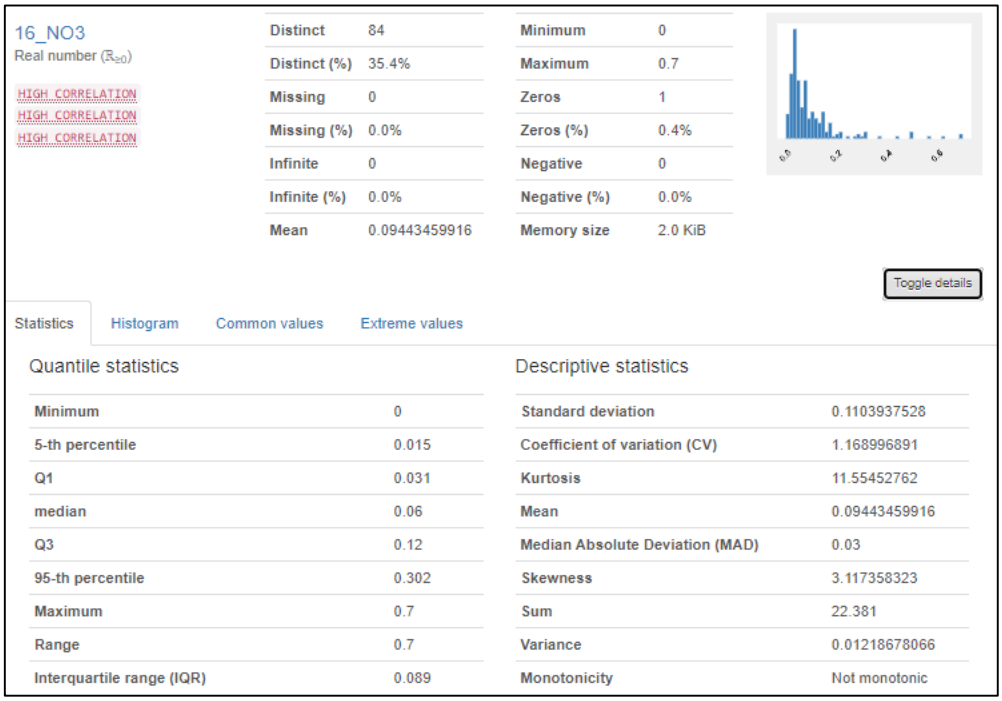
Minimum	12.8
5-th percentile	19.88
Q1	23.7
median	28.2
Q3	33.3
95-th percentile	45.18
Maximum	228
Range	215.2
Interquartile range (IQR)	9.6

Descriptive statistics

Standard deviation	15.55061045
Coefficient of variation (CV)	0.5114757357
Kurtosis	111.1809035
Mean	30.40341772
Median Absolute Deviation (MAD)	4.7
Skewness	9.062408788
Sum	7205.61
Variance	241.8214853
Monotonicity	Not monotonic

Рисунок 2.19 – Вміст амонію

Далі буде огляд інформації про вміст нітратів та хлору з поста №16 (рис. 2.20 –2.21).



16_NO3
Real number (ℝ≥0)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	84
Distinct (%)	35.4%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	0.09443459916

Minimum	0
Maximum	0.7
Zeros	1
Zeros (%)	0.4%
Negative	0
Negative (%)	0.0%
Memory size	2.0 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

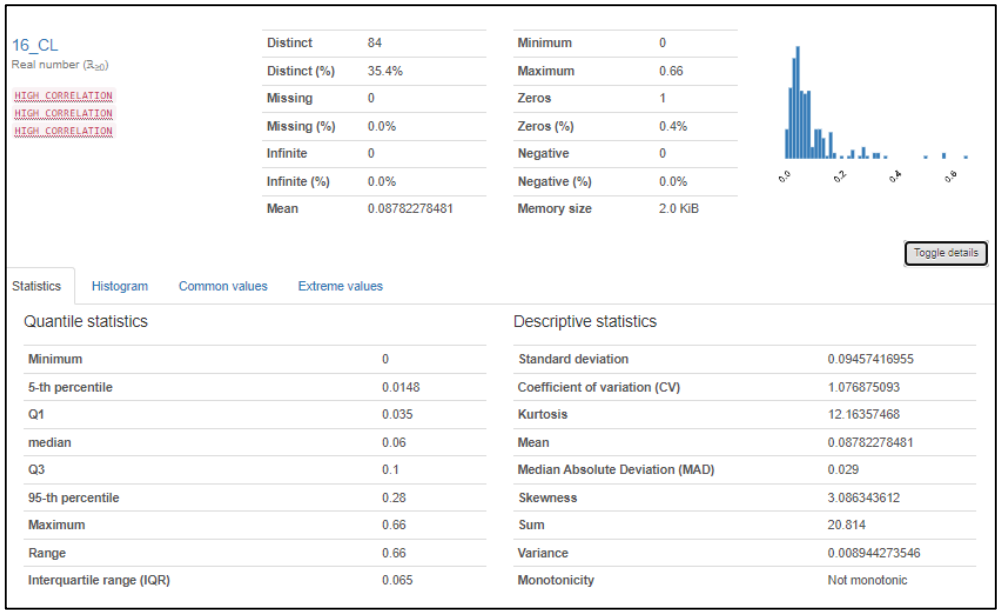
Quantile statistics

Minimum	0
5-th percentile	0.015
Q1	0.031
median	0.06
Q3	0.12
95-th percentile	0.302
Maximum	0.7
Range	0.7
Interquartile range (IQR)	0.089

Descriptive statistics

Standard deviation	0.1103937528
Coefficient of variation (CV)	1.168996891
Kurtosis	11.55452762
Mean	0.09443459916
Median Absolute Deviation (MAD)	0.03
Skewness	3.117358323
Sum	22.381
Variance	0.01218678066
Monotonicity	Not monotonic

Рисунок 2.20 – Вміст нітратів



16_CL
Real number (ℝ≥0)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	84
Distinct (%)	35.4%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	0.08782278481

Minimum	0
Maximum	0.66
Zeros	1
Zeros (%)	0.4%
Negative	0
Negative (%)	0.0%
Memory size	2.0 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

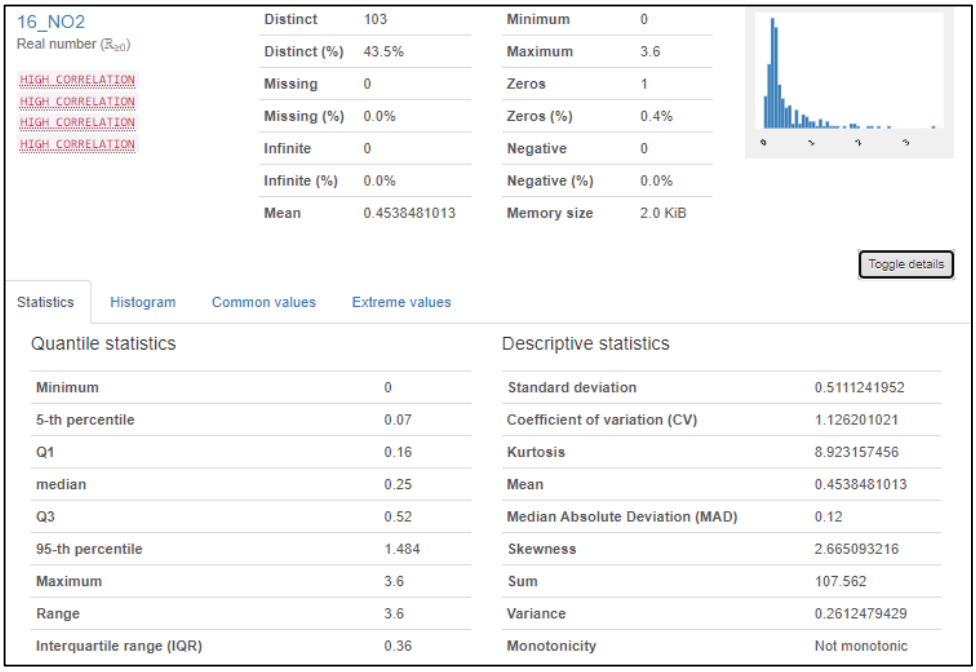
Minimum	0
5-th percentile	0.0148
Q1	0.035
median	0.06
Q3	0.1
95-th percentile	0.28
Maximum	0.66
Range	0.66
Interquartile range (IQR)	0.065

Descriptive statistics

Standard deviation	0.09457416955
Coefficient of variation (CV)	1.076875093
Kurtosis	12.16357468
Mean	0.08782278481
Median Absolute Deviation (MAD)	0.029
Skewness	3.086343612
Sum	20.814
Variance	0.008944273546
Monotonicity	Not monotonic

Рисунок 2.21 – Вміст хлору

Інформація про показники нітритів та амонію з поста №16 зображена на рисунках 2.22 – 2.23.



16_NO2
Real number (ℝ≥0)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	103	Minimum	0
Distinct (%)	43.5%	Maximum	3.6
Missing	0	Zeros	1
Missing (%)	0.0%	Zeros (%)	0.4%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	0.4538481013	Memory size	2.0 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

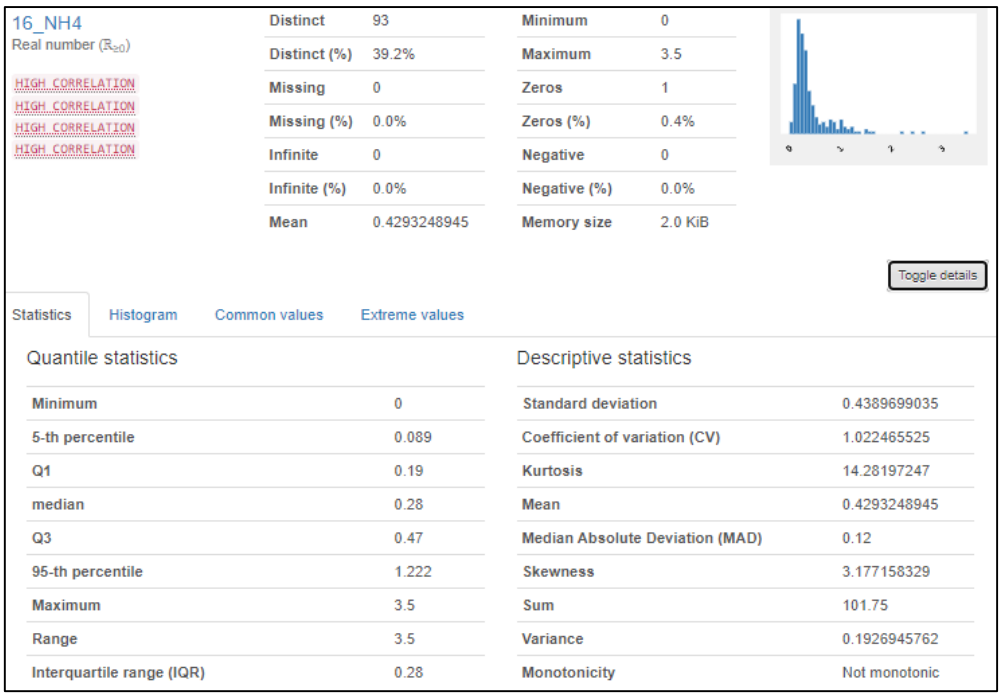
Quantile statistics

Minimum	0
5-th percentile	0.07
Q1	0.16
median	0.25
Q3	0.52
95-th percentile	1.484
Maximum	3.6
Range	3.6
Interquartile range (IQR)	0.36

Descriptive statistics

Standard deviation	0.5111241952
Coefficient of variation (CV)	1.126201021
Kurtosis	8.923157456
Mean	0.4538481013
Median Absolute Deviation (MAD)	0.12
Skewness	2.665093216
Sum	107.562
Variance	0.2612479429
Monotonicity	Not monotonic

Рисунок 2.22 – Вміст нітритів



16_NH4
Real number (ℝ≥0)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	93	Minimum	0
Distinct (%)	39.2%	Maximum	3.5
Missing	0	Zeros	1
Missing (%)	0.0%	Zeros (%)	0.4%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	0.4293248945	Memory size	2.0 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0
5-th percentile	0.089
Q1	0.19
median	0.28
Q3	0.47
95-th percentile	1.222
Maximum	3.5
Range	3.5
Interquartile range (IQR)	0.28

Descriptive statistics

Standard deviation	0.4389699035
Coefficient of variation (CV)	1.022465525
Kurtosis	14.28197247
Mean	0.4293248945
Median Absolute Deviation (MAD)	0.12
Skewness	3.177158329
Sum	101.75
Variance	0.1926945762
Monotonicity	Not monotonic

Рисунок 2.23 – Вміст амонію

Проаналізувавши виведені графіки, можна зробити висновок, що концентрація нітратів у воді на постах №14, №15, №16 є приблизно однаковою. Це може свідчити про те, що вище по течії немає сильних забруднювачів.

Щоб мати остаточне розуміння того, чи взаємодіють речовини між собою, та чи є залежність в якості води одного поста від іншого, було розглянуто

матрицю кореляції Спірмена, яка була побудована за допомогою тієї ж бібліотеки Pandas Profiling (рис. 2.24).

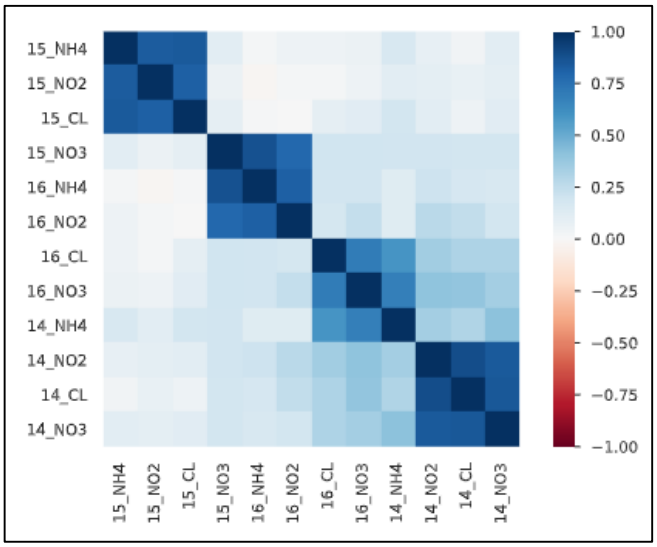


Рисунок 2.24 – Матриця кореляції

Розглянувши дану матрицю, помітно, що 0 вказує на те, що кореляція не відбувається. Проте наближуючись до 1 вона стає сильнішою. Тому судячи по кольору, можна зробити висновок, що деякі речовини досить сильно корелюють між собою. Проте кореляція, яка відбувається між фактором і цільовою змінною знаходиться в межах 45%.

Для того, щоб коректно зробити прогнозування, потрібно проаналізувати датасет на вміст аномальних даних. Для цього створюється таблиця, і з неї виділяються або занадто високі, або ж занадто низькі показники (рис. 2.25, 2.26).

	index	ds	15_NH4	15_NO2	15_CL	15_NO3	16_NH4	16_NO2	16_CL	16_NO3	14_NH4	14_NO2	14_CL	14_NO3
0	0	2000-01-02	31.30	30.4	26.80	2.20	2.20	2.40	0.270	0.130	0.130	8.80	8.40	7.70
1	1	2000-01-03	28.60	26.8	25.00	0.68	0.87	0.54	0.090	0.170	0.270	8.80	9.10	8.80
2	2	2000-01-08	24.10	25.8	25.80	0.37	0.25	0.14	0.240	0.150	0.160	1.50	7.00	0.90
3	3	2000-04-04	23.20	22.3	22.30	0.81	1.22	0.51	0.090	0.140	0.090	4.60	4.90	3.50
4	4	2000-04-07	23.20	23.2	20.50	0.10	0.07	0.14	0.170	0.320	0.360	2.30	2.10	1.70
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
232	232	2020-09-15	36.70	34.2	31.70	0.26	0.25	0.16	0.064	0.051	0.072	1.03	0.57	0.84
233	233	2020-10-06	40.77	48.5	47.00	0.38	0.28	0.30	0.037	0.038	0.052	0.24	0.44	0.96
234	234	2020-12-08	29.62	38.9	29.61	0.57	0.28	0.20	0.061	0.036	0.046	0.68	1.03	0.71
235	235	2021-03-16	36.10	30.7	36.10	1.03	1.38	2.29	0.122	0.134	6.790	10.90	8.56	7.70
236	236	2021-06-04	38.80	39.7	37.90	0.43	0.17	0.16	0.129	0.179	0.164	6.57	8.07	6.38

237 rows × 14 columns

Рисунок 2.25 – Таблиця з даними

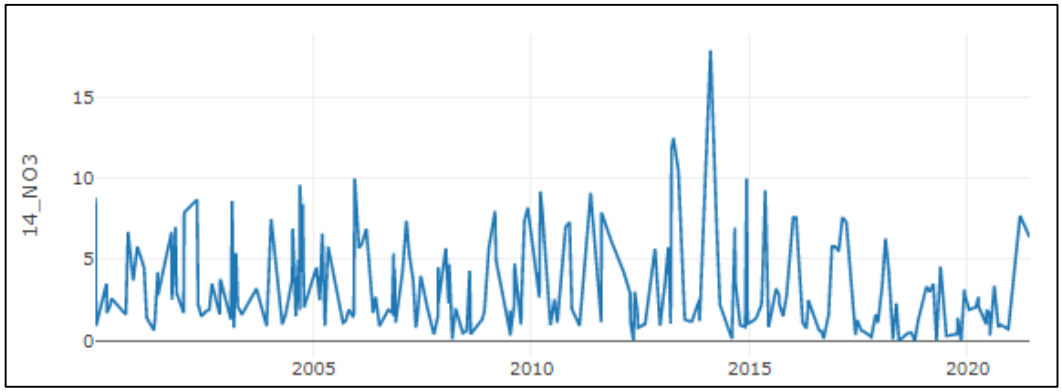


Рисунок 2.26 – Графік даних

Судячи з графіку, видно що за останній час є дві чітких аномальні дати (рис. 2.27, 2.28)

```
# Synthesis dataframe with anomalous dates for Facebook Prophet
if is_anomalies:
    anomalous_dates = ['2014-02-09', '2013-04-04']
    holidays_df = pd.DataFrame(columns = ['ds', 'lower_window', 'upper_window', 'prior_scale'])
    holidays_df['ds'] = anomalous_dates
    holidays_df['holiday'] = 'anomalous_dates'
    holidays_df['lower_window'] = 0
    holidays_df['upper_window'] = 0
    holidays_df['prior_scale'] = 10
    display(holidays_df)
```

	ds	lower_window	upper_window	prior_scale	holiday
0	2014-02-09	0	0	10	anomalous_dates
1	2013-04-04	0	0	10	anomalous_dates

Рисунок 2.27 – Синтез аномальних даних

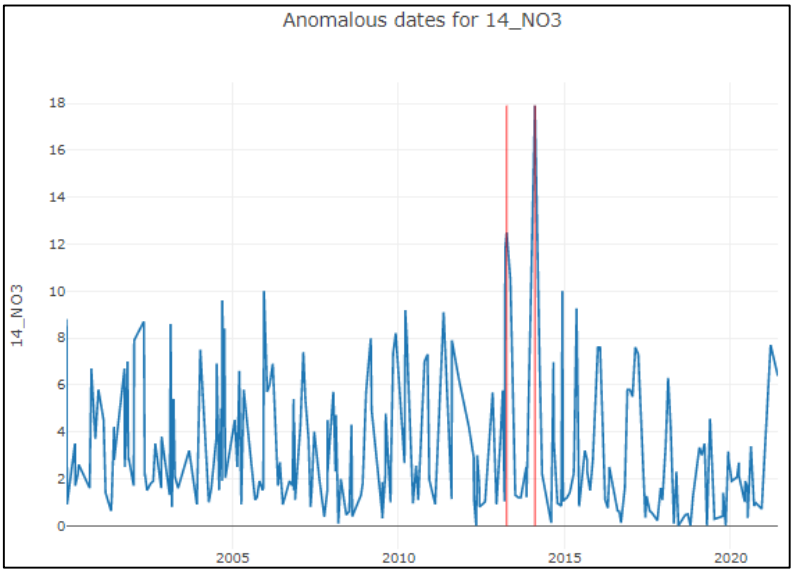


Рисунок 2.28 – Графік аномальних даних

Для того, щоб ще краще можна було провести аналіз даних, було створено два дата сети. Перший, в період з 2016 по 2018, а другий з 2019 по 2021. Порівнявши їх можна буде краще і зрозуміти тенденцію зміни концентрації нітратів у воді (рис. 2.29, 2.30).

```
# Set time interval of data
final_data = 2021
years_num_period = 2
year1_start = 2016
year1_end = year1_start + years_num_period
year2_start = year1_start + years_num_period + 1
year2_end = year2_start + years_num_period if (year2_start + years_num_period) < final_data else final_data
print(f"Interval 1 - from {year1_start} to {year1_end}, Interval 2 - from {year2_start} to {year2_end}")
```

```
Interval 1 - from 2016 to 2018, Interval 2 - from 2019 to 2021
```

Рисунок 2.29 – Створення дата сетів

	14_NH4	14_NO2	14_CL	14_NO3
0	0.100	9.70	8.00	7.60
1	0.110	8.20	7.60	7.60
2	0.050	0.17	0.16	1.10
3	0.038	0.53	0.47	0.75
4	0.070	1.20	0.20	1.91
5	0.140	3.30	3.00	2.50
6	0.000	0.38	0.55	0.63
7	0.040	0.24	0.46	0.60
8	0.040	0.24	0.30	0.12
9	0.020	0.84	1.30	1.61
10	0.050	2.90	4.03	5.80
11	0.060	5.60	6.80	5.80

	14_NH4	14_NO2	14_CL	14_NO3
0	0.064	2.50	2.80	3.33
1	0.140	6.20	0.53	3.00
2	0.080	3.30	3.10	3.50
3	0.072	0.00	0.00	0.00
4	0.330	3.68	3.18	4.56
5	0.047	0.28	0.27	0.28
6	0.075	0.25	0.28	0.40
7	0.083	4.70	5.70	1.40
8	0.017	0.23	0.15	0.81
9	0.010	0.00	0.00	0.00
10	0.950	5.52	4.37	3.16
11	0.060	2.20	1.50	2.80

Рисунок 2.30 – Згенеровані дата сети

Дані, які було виокремлено з спільного дата сету можна більш детально розглянути на наступному графіку (рис.2.31).

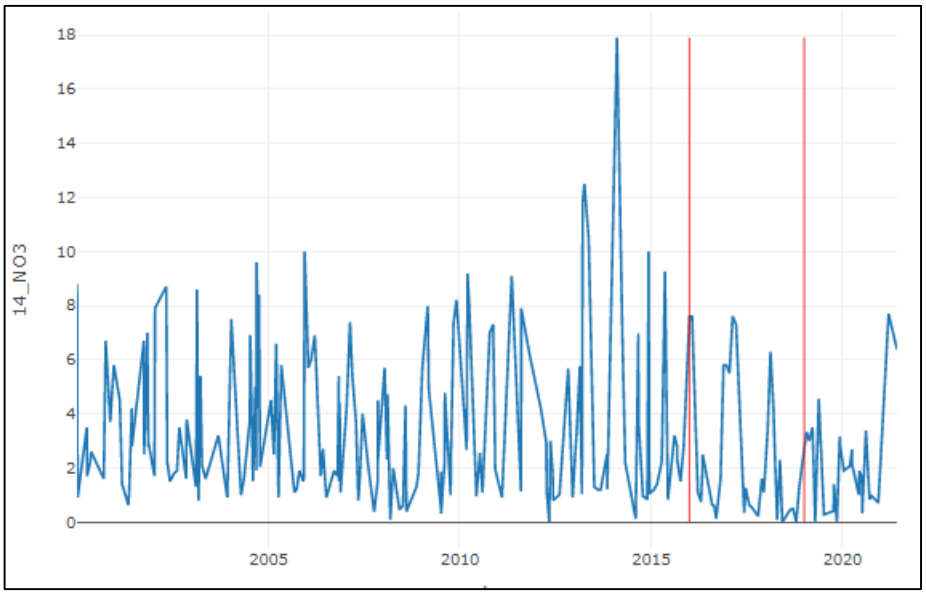


Рисунок 2.31 – Графік виокремлених даних

Розглянувши графік, можна побачити, з якого по який період було взято дані для порівняння.

Для того, щоб коректно візуалізувати, і одразу ж порівняти дані, було використано бібліотеку Sweetviz (рис. 2.32).

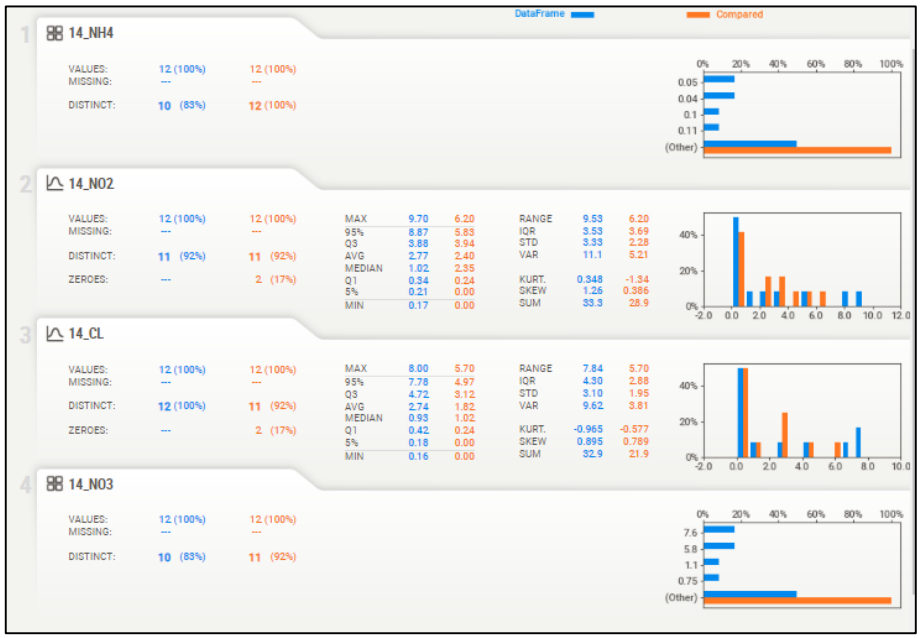


Рисунок 2.32 – Графіки порівняння даних

Дивлячись на рисунок 2.32, можна зрозуміти, що в другому періоді вміст Амонію та нітратів у воді не змінився, фактично залишився таким же. Розглядаючи ж графіки інших речовин, можна побачити таку тенденцію, що концентрація їх, поступово збільшується.

Наступним кроком було використано бібліотеку AutoViz. Таким чином, Отримані два дата сети, об’єднуються в один спільний, і на основі нього, система генерує графіки (рис. 2.33 – 2.34).

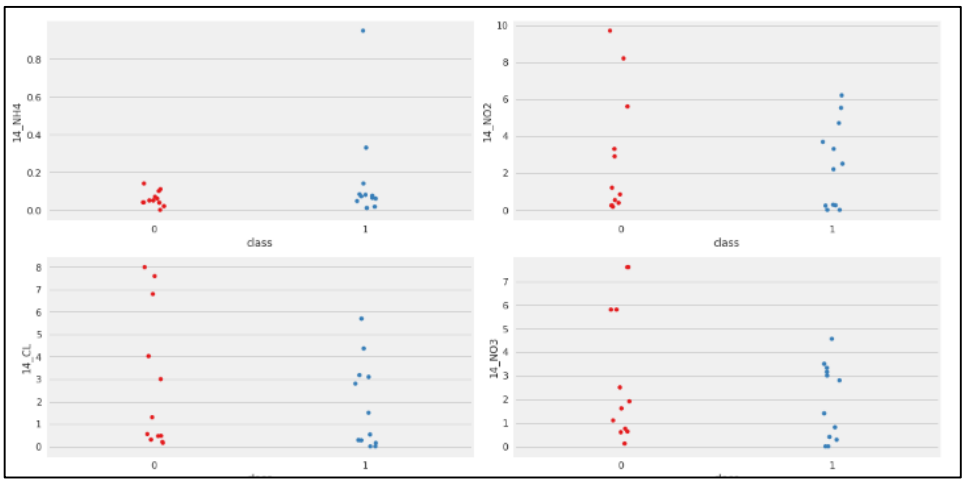


Рисунок 2.33 – Графіки розкиду значень

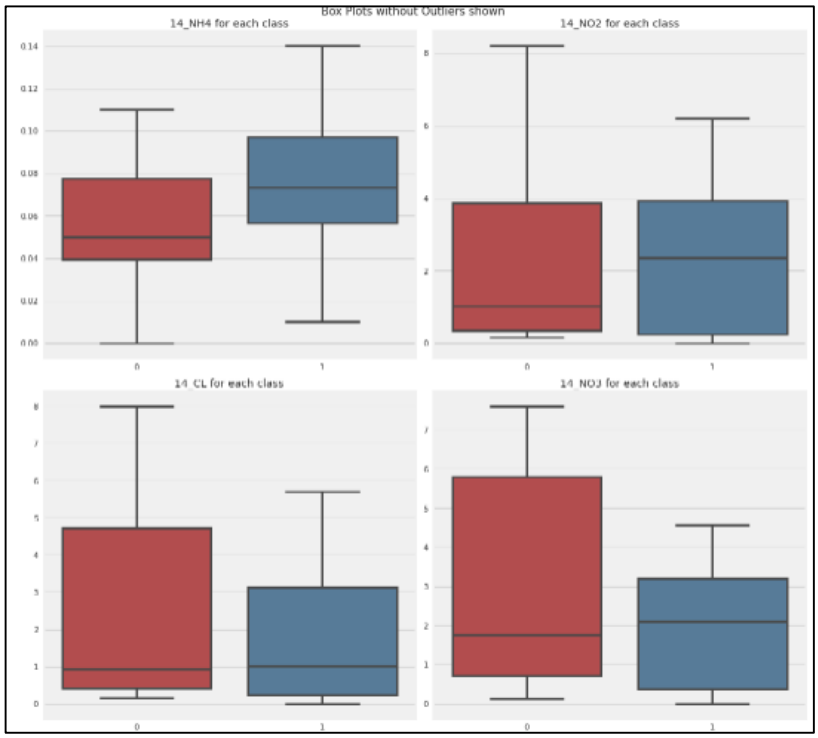


Рисунок 2.34 – Графіки розкиду значень

На останньому рисунку видно графіки у вигляді так званих «Вусатих» діаграм, краї яких відображають значення, які є аномальними. Далі було побудовано гістограми, на яких видно, що дані, які отримані за останній час є досить схожими (рис 2.35)

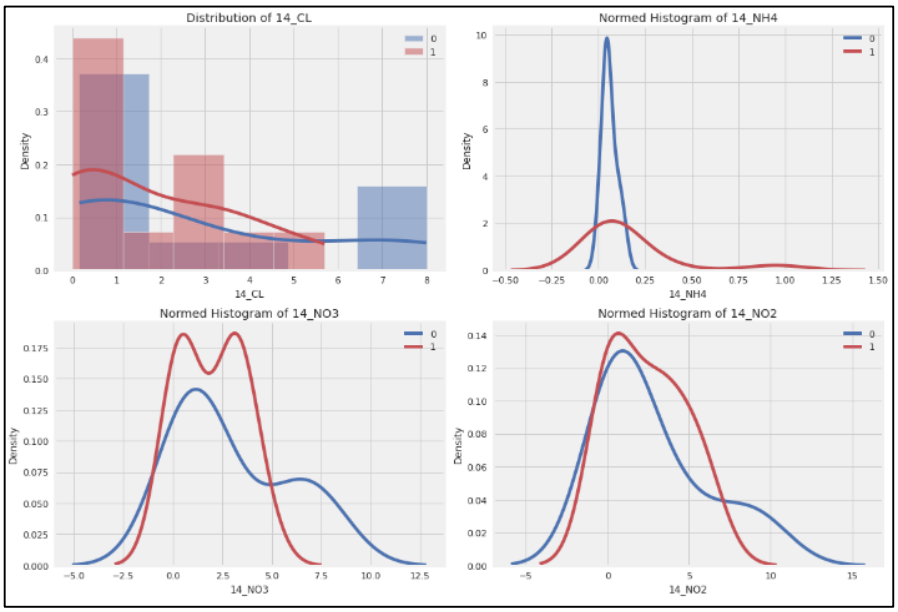


Рисунок 2.35 – Гістограми

Також, бібліотека згенерувала матрицю кореляції, яка показує взаємодію між речовинами (рис. 3.36).

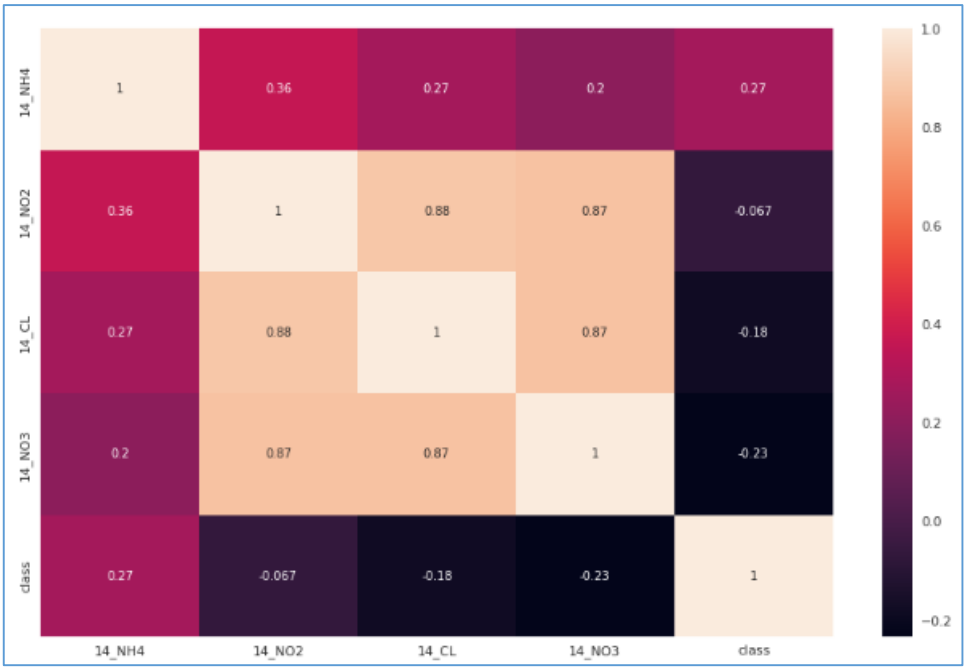


Рисунок 2.36 – Матриця коефіцієнта кореляції

На даній матриці, коефіцієнт кореляції тепловий, тому чим світліше область, тим сильнішою є кореляція. Як і на матриці кореляції Спірмена можна чітко побачити, що відбувається взаємодія між обраними речовинами NO3, NH4, NO2, CL.

2.3 Висновки

Провівши попередній аналіз даних, було визначено, що в наявному датасеті є аномалії та зашумлені дані. Відфільтрувавши їх було створено ще одну вибірку з даними, використавши яку, значно покращиться точність прогнозування.

Візуалізувавши дані у вигляді графіків, за допомогою бібліотек Auto Viz, Sweet Wiz та Pandas Profiling Report стало зрозуміло, що обрані для аналізу речовини (NH4, NO2, CL) взаємодіють у воді з нітратами NO3 та між собою, також має місце незначне забруднення вниз по течії.

Отже, в даному розділі було підготовлено дані для подальшої роботи.

3 ПРОГНОЗУВАННЯ КОНЦЕНТРАЦІЇ НІТРАТІВ У РІЧКОВІЙ ВОДІ

3.1 Створення архітектури інформаційної технології

Для розробки інформаційної технології було створено наступний алгоритм (рис.3.1):

- Попередній аналіз даних;
- створення вибірки з цільовими ознаками;
- стандартизація ознак;
- розподіл даних на тестові та валідаційні;
- навчання обраних моделей;
- вибір оптимальної моделі для прогнозування;
- тренування найкращої моделі;
- прогноз;
- проведення аналізу важливості ознак.

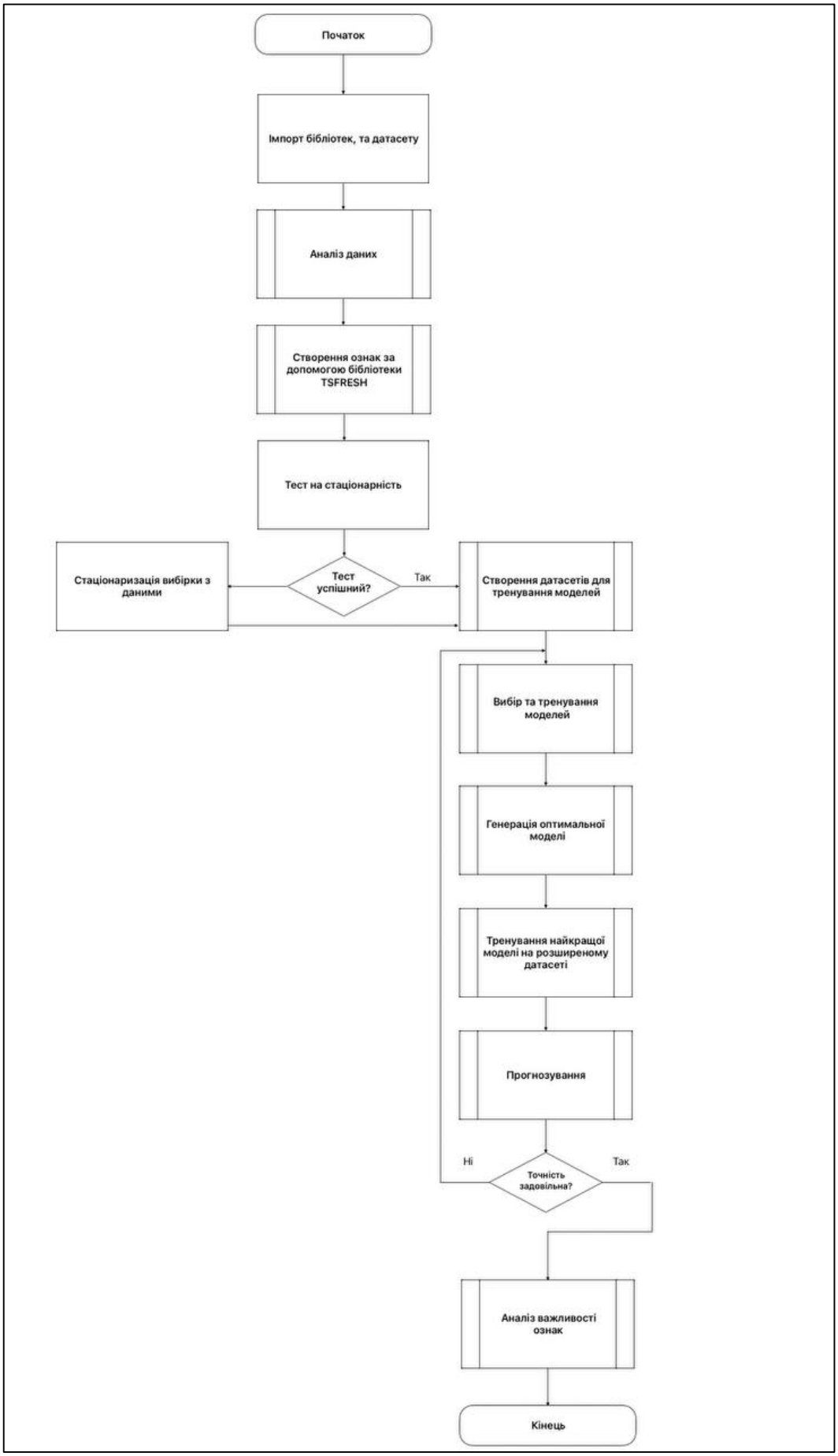


Рисунок 3.1 – Блок-схема інформаційної технології

3.2 Тренування моделей

Перш ніж розпочати тренування моделей штучного інтелекту, використавши бібліотеку TSFRESH, з датасету було автоматично згенеровано безліч статистичних ознак (рис 3.2, 3.3).

```python
def get_tsfresh_features(data):
    # Get statistic features using library TSFRESH
    # Thanks to https://www.kaggle.com/code/vbmokin/btc-growth-forecasting-with-advanced-fe-for-ohlc

    # Extract features
    extracted_features = extract_features(data, column_id="ds", column_sort="ds")

    # Drop features with NaN
    extracted_features_clean = extracted_features.dropna(axis=1, how='all').reset_index(drop=True)

    # Drop features with constants
    cols_std_zero  = []
    for col in extracted_features_clean.columns:
        if extracted_features_clean[col].std()==0:
            cols_std_zero.append(col)
    extracted_features_clean = extracted_features_clean.drop(columns = cols_std_zero)

    extracted_features_clean['ds'] = data['ds']   # For the merging

    return extracted_features_clean
```

Рисунок 3.2 – Генерація статистичних ознак

	14_NO3__sum_values	14_NO3__abs_energy	14_NO3__median	14_NO3__mean	14_NO3__root_mean_square	14_NO3__maximum	14_NO3__absolute_maximum	14_NO3__minimum	14_NO3__benford_corr
0	7.70	59.2900	7.70	7.70	7.70	7.70	7.70	7.70	-0.
1	8.80	77.4400	8.80	8.80	8.80	8.80	8.80	8.80	-0.
2	0.90	0.8100	0.90	0.90	0.90	0.90	0.90	0.90	-0.
3	3.50	12.2500	3.50	3.50	3.50	3.50	3.50	3.50	0.
4	1.70	2.8900	1.70	1.70	1.70	1.70	1.70	1.70	0.
...	...	...	...	...	...	...	...	...	
232	0.84	0.7056	0.84	0.84	0.84	0.84	0.84	0.84	-0.
233	0.96	0.9216	0.96	0.96	0.96	0.96	0.96	0.96	-0.
234	0.71	0.5041	0.71	0.71	0.71	0.71	0.71	0.71	-0.
235	7.70	59.2900	7.70	7.70	7.70	7.70	7.70	7.70	-0.
236	6.38	40.7044	6.38	6.38	6.38	6.38	6.38	6.38	-0.

237 rows × 28 columns

Рисунок 3.3 – Таблиця з статистичними ознаками

Як видно на рисунку 3.3, бібліотека згенерувала 28 ознак.

Продовжуючи роботу, датасет було розділено на три частини, тренувальні, валідаційні, та тестові дані. Оскільки, для аналізу і прогнозування

використовуються моделі часових рядів, і машинного навчання, які працюють по різному принципу, то потрібно створювати різні набори даних (рис. 3.4, 3.5).

```
def get_train_valid_test_ts(df, forecasting_days, target=target_name):
    # Get training, validation and test datasets with target for Time Series models

    # Data prepairing
    df = df.dropna(how="any").reset_index(drop=True)
    df = df[['ds', target_name]]
    df.columns = ['ds', 'y']
    y = None

#

    N = len(df)
    train, _ = cut_data(df, y, 0, N-2*forecasting_days-1)
    valid, _ = cut_data(df, y, N-2*forecasting_days, N-forecasting_days-1)
    test, _ = cut_data(df, y, N-forecasting_days, N)

    # Train+valid - for optimal model training
    train_valid = pd.concat([train, valid])

    print(f'Origin dataset has {len(df)} rows and {len(df.columns)} features')
    print(f'Get training dataset with {len(train)} rows')
    print(f'Get validation dataset with {len(valid)} rows')
    print(f'Get test dataset with {len(test)} rows')

    return train, valid, test, train_valid
```

Рисунок 3.4 – Розділення даних для моделей часових рядів

```
    # Get training, validation and test datasets with target for multi-features ML models

    df = df.drop(columns = ['ds']).dropna(how="any").reset_index(drop=True)

    # Save and drop target
    y = df.pop(target)

    # Get starting points for the recovering target_name from target_name_shigted
    N = len(df)
    #print(f"Total - {N}, Valid start index = {N-forecasting_days-1}, Test start index = {N-1}")
    start_points = {'valid_start_point' : df.loc[N-forecasting_days-1, target_name],
                    'test_start_point' : df.loc[N-1, target_name]}

    # Standartization data
    scaler = StandardScaler()
    df = pd.DataFrame(scaler.fit_transform(df), columns = df.columns)


    train, ytrain = cut_data(df.copy(), y, 0, N-2*forecasting_days-1)
    valid, yvalid = cut_data(df.copy(), y, N-2*forecasting_days, N-forecasting_days-1)
    test, ytest = cut_data(df.copy(), y, N-forecasting_days, N)


    # Train+valid - for optimal model training
    train_valid = pd.concat([train, valid])
    y_train_valid = pd.concat([ytrain, yvalid])

    print(f'Origin dataset has {len(df)} rows and {len(df.columns)} features')
    print(f'Get training dataset with {len(train)} rows')
    print(f'Get validation dataset with {len(valid)} rows')
    print(f'Get test dataset with {len(test)} rows')

    return train, ytrain, valid, yvalid, test, ytest, train_valid, y_train_valid, start_points
```

Рисунок 3.5 – Розділення даних для моделей машинного навчання

Виділивши необхідні дані, можна перейти до тренування моделей.

Аналізуватись моделі будуть по таким трьом метрикам(рис.3.6):

- «r2_score» - дозволяє оцінити, в якому напрямку йде точність прогнозу;
- «rmse» - корінь з квадрату середньої похибки;
- «mape» - відносна похибка від значень у відсотках.

```python
def calc_metrics(type_score, list_true, list_pred):
    # Calculation score with type=type_score for list_true and list_pred
    if type_score=='r2_score':
        score = r2_score(list_true, list_pred)
    elif type_score=='rmse':
        score = mean_squared_error(list_true, list_pred, squared=False)
    elif type_score=='mape':
        score = mean_absolute_percentage_error(list_true, list_pred)
    return score
```

```python
def result_add_metrics(result, n, y_true, y_pred):
    # Calculation and addition metrics into dataframe result[n,:]

    result.loc[n,'r2_score'] = calc_metrics('r2_score', y_true, y_pred)
    result.loc[n,'rmse'] = calc_metrics('rmse', y_true, y_pred)      # in coins
    result.loc[n,'mape'] = 100*calc_metrics('mape', y_true, y_pred)  # in %

    return result
```

Рисунок 3.6 – Метрики для аналізу найкращої моделі

3.2.1 Тренування моделі Facebook Prophet

Першою моделлю, яка буде тренуватись буде Facebook Prophet. Для її тренування генеруються три дата сети (рис.3.7).

```python
# Get datasets
if is_Prophet:
    train_ts, valid_ts, test_ts, train_valid_ts = get_train_valid_test_ts(df.copy(), forecasting_days, target=target_name)

    if not is_anomalies:
        holidays_df = None
```

```
Origin dataset has 234 rows and 2 features
Get training dataset with 220 rows
Get validation dataset with 7 rows
Get test dataset with 7 rows
```

Рисунок 3.7 – Створення дата сетів

Як видно на рисунку 3.7, набір даних було поділено на три частини, серед яких 7 – тестові, 7 – валідаційні, решта призначені для навчання.

На відмінну від моделей машинного навчання, для дослідження часових рядів системою Facebook Prophet, не достатньо, просто виділити якийсь відсоток даних для тренування, потрібно знати що буде потім, а не що було колись, за якийсь період. Тому тестові і валідаційні дані обираються саме вкінці діапазону.

Далі було запущено прямий перебір, де 2, 3, 5, 10 – це сезонність, а 3, 12 – порядок Фур'є (рис. 3.8).

```
%%time
# Models tuning
if is_Prophet:
    for period_days in [2, 3, 5, 10]:
        for fourier_order_seasonality in [3, 12]:
            result, _ = prophet_modeling(result,
                                         target_indicator_name,
                                         train_ts,
                                         valid_ts,
                                         holidays_df,
                                         period_days,
                                         fourier_order_seasonality,
                                         forecasting_days,
                                         f'{period_days}_days_{fourier_order_seasonality}_order',
                                         'valid')
```

Рисунок 3.8 – Запуск перебору

Після чого модель видає набір даних у вигляді графіків. Перший набір графіків, за два дні (рис. 3.9 – 3.12).

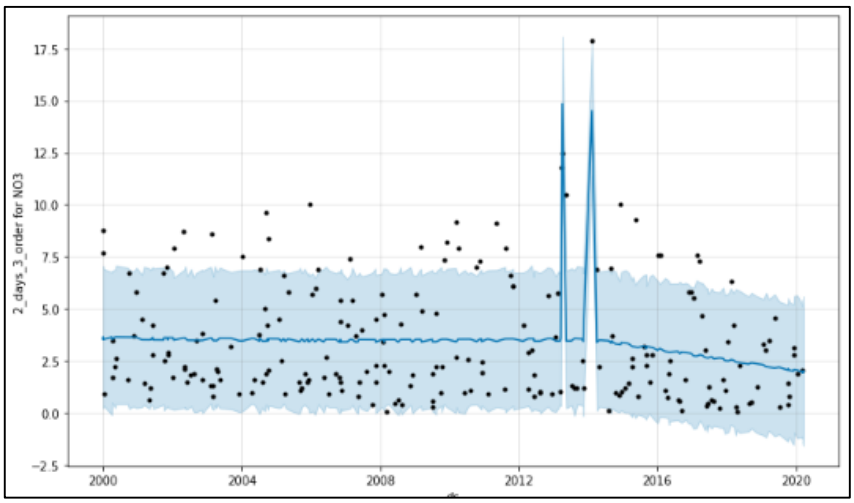


Рисунок 3.9 – Опис набору даних

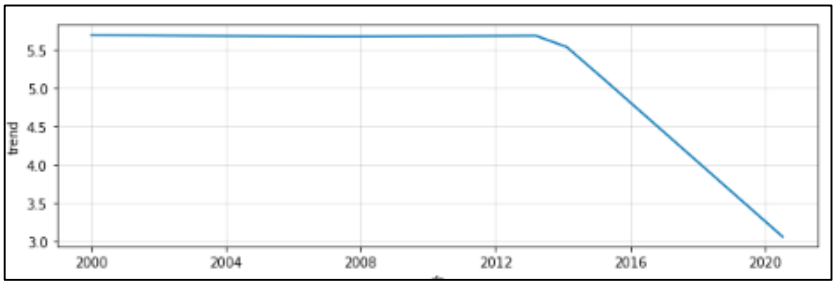


Рисунок 3.10 – Опис тренду

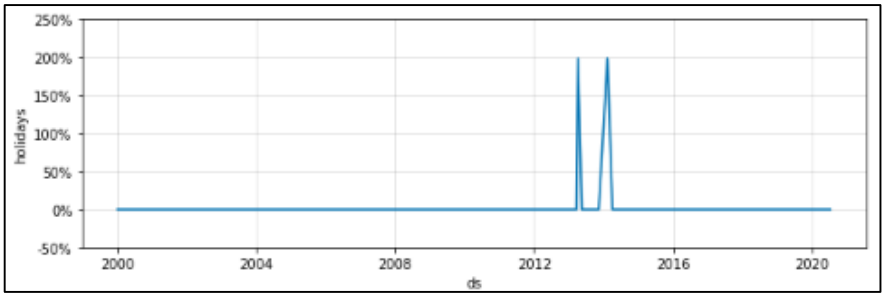


Рисунок 3.11 – Опис аномальних дат

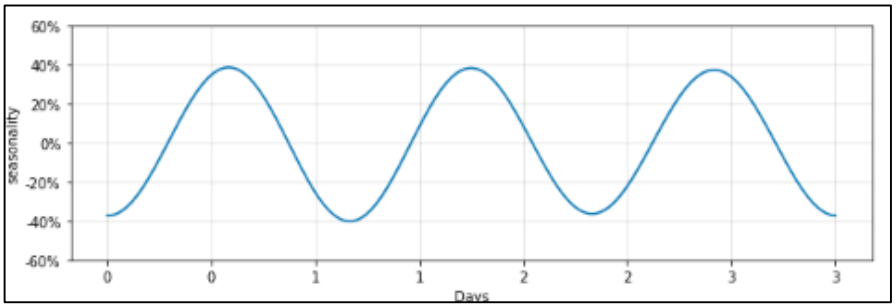


Рисунок 3.12 – Опис сезонності

Наступний набір графіків, показує дані за три дні (рис. 3.13 – 3.16).

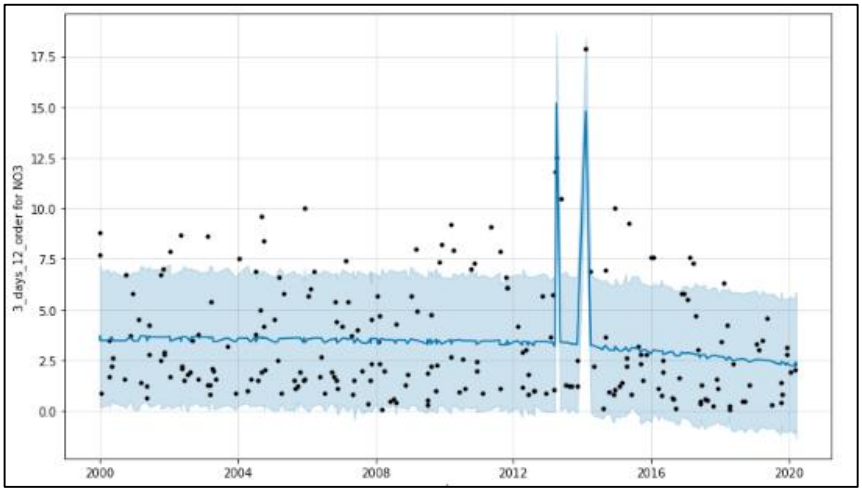


Рисунок 3.13 – Опис набору даних

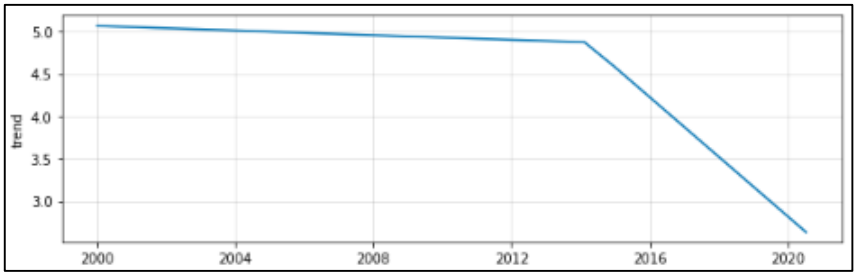


Рисунок 3.14 – Опис тренду

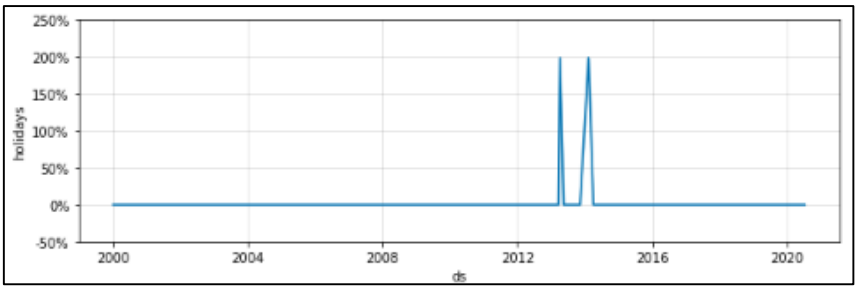


Рисунок 3.15 – Опис аномальних дат

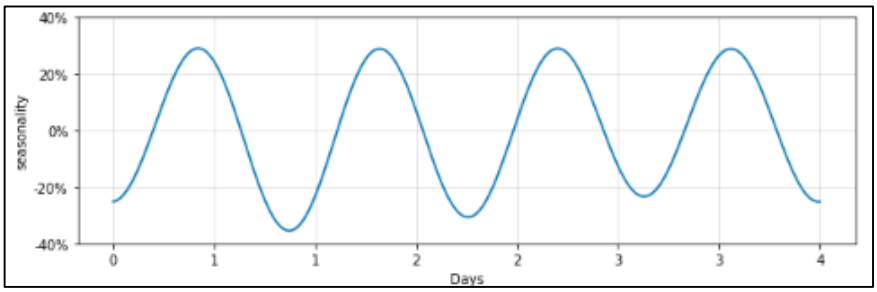


Рисунок 3.16 – Опис сезонності

Далі, буде доцільним розглянути набір даних, які зображені у вигляді графіків, за 5 днів (рис.3.17 – 3.18).

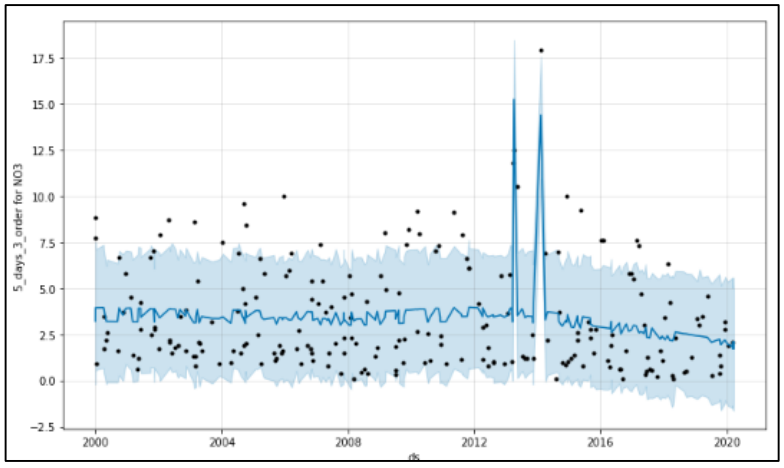


Рисунок 3.17 – Опис набору даних

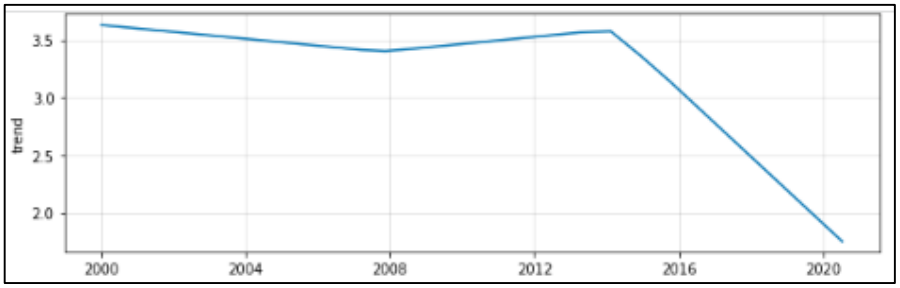


Рисунок 3.18 – Опис тренду

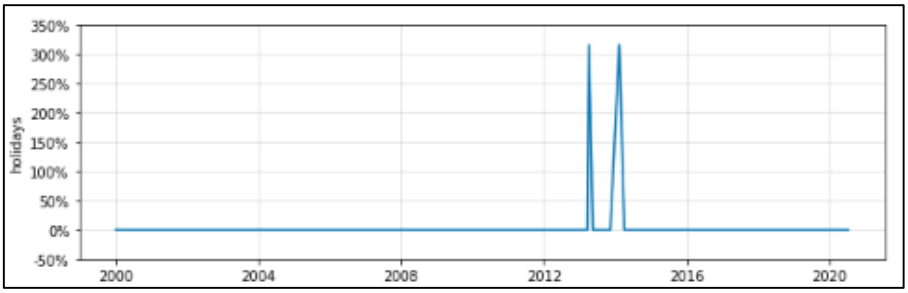


Рисунок 3.19 – Опис аномальних дат

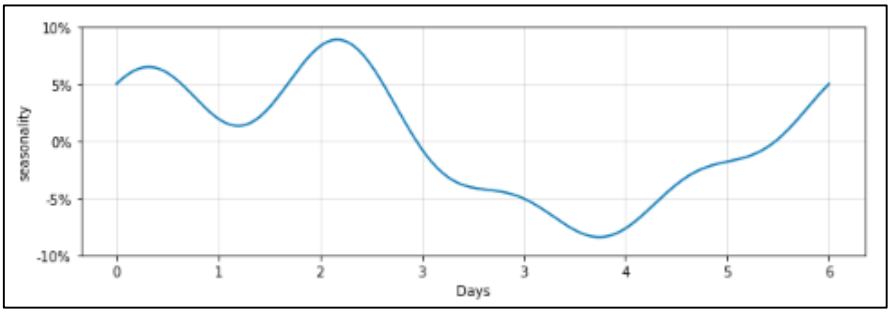


Рисунок 3.20 - Опис сезонності

Слідуючим етапом, розглянуто набір даних, які зображені у вигляді графіків, за 12 днів (рис.3.21 – 3.24).

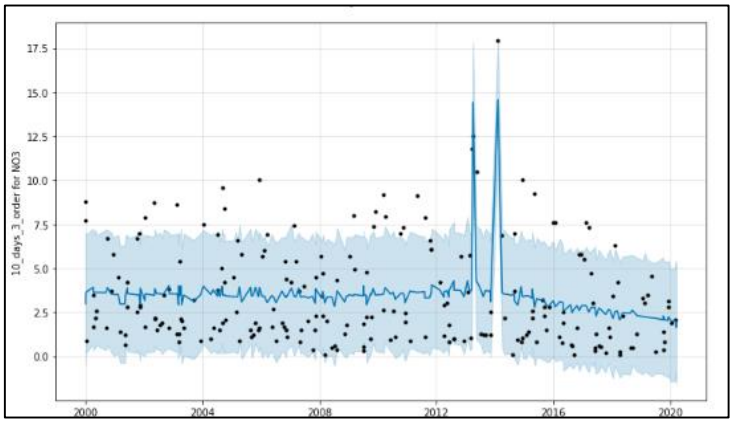


Рисунок 3.21 – Опис набору даних

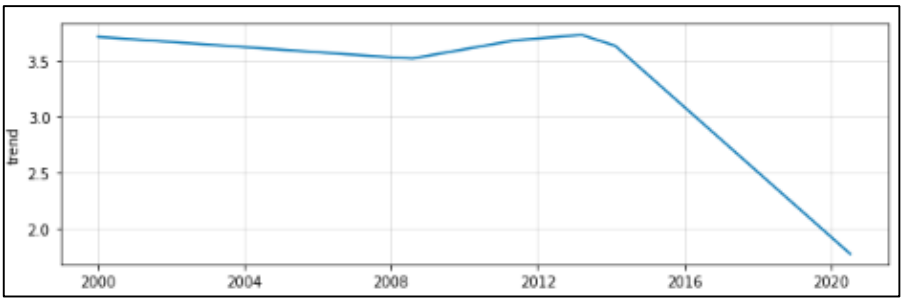


Рисунок 3.22 – Опис тренду

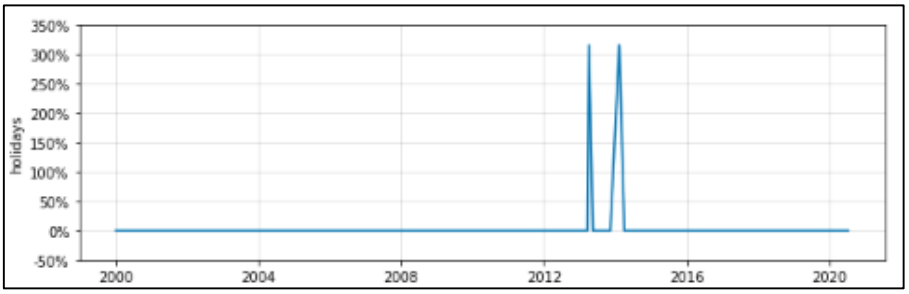


Рисунок 3.23 – Опис аномальних дат

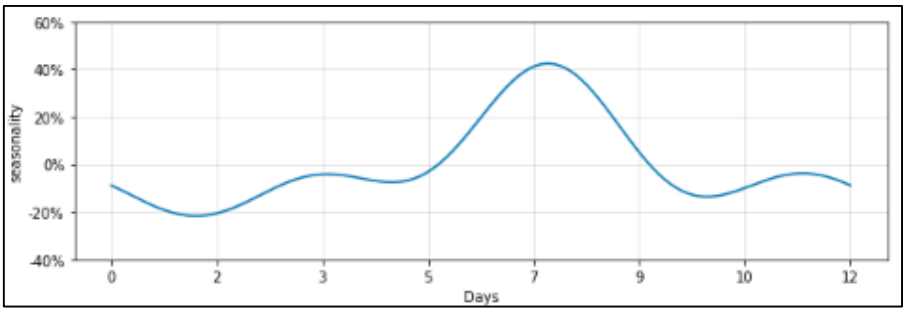


Рисунок 3.24 – Опис сезонності

Проглянувши, та порівнявши дані графіки на початку роботи, було визначено, що точність є не достатньо гарною, тому підправивши сезонність вдалось отримати кращі результати прогнозування.

3.2.2 Тренування моделі ARIMA

Наступною буде тренуватись модель ARIMA. Аналогічно з системою Facebook Prophet, щоб зробити аналіз часових рядів за допомогою системи ARIMA, створюються три дата сети, де виділяється: 7 – тестових, 7 – валідаційних даних та інші, які призначені для навчання моделі (рис.3.25).

```
# Get datasets
if is_ARIMA:
    train_ts, valid_ts, test_ts, train_valid_ts = get_train_valid_test_ts(df.copy(), forecasting_days, target=target_name)
```

```
Origin dataset has 234 rows and 2 features
Get training dataset with 220 rows
Get validation dataset with 7 rows
Get test dataset with 7 rows
```

Рисунок 3.25 – Генерація даних

Запустивши модель, після раніше уже згадуваного тесту «Augmented Dickey Fuller test», будується авто кореляційна, та частково кореляційна функції, з яких видно, що і при першій і при другій різниці, дані не чітко потрапляють в діапазон 5%. Проте порівнявши їх, можна зробити висновок, що перша різниця є кращою (рис. 3.26 – 3.28).

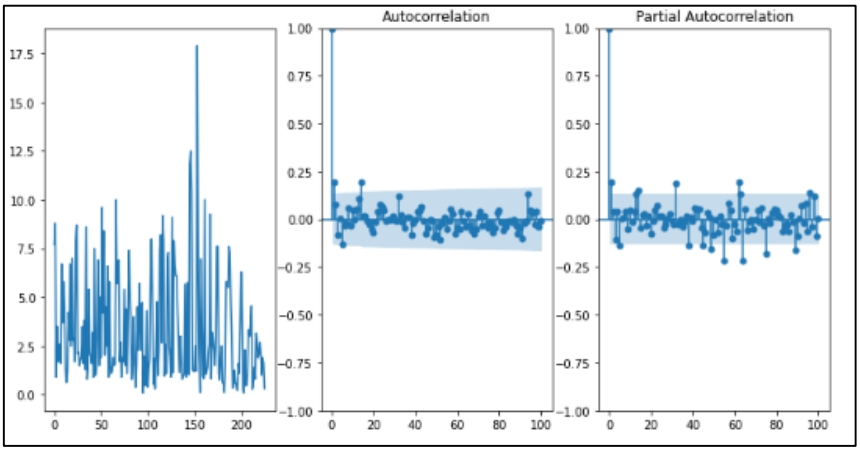


Рисунок 3.26 – Кореляційна і частково кореляційна функції

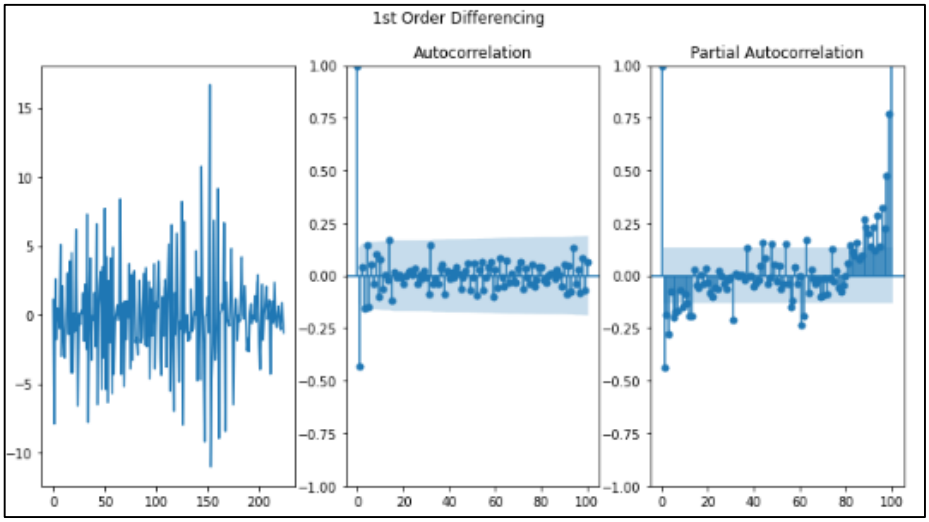


Рисунок 3.27 – Перша різниця

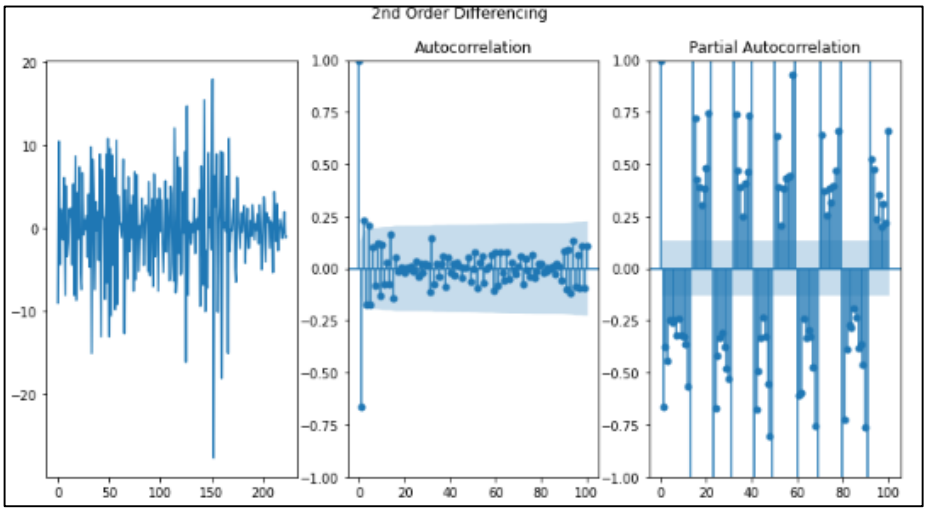


Рисунок 3.28 – Друга різниця

Після чого було отримано результат ручного налаштування моделі ARIMA (рис. 3.29).

```
Best model:  ARIMA(1,0,2)(0,0,0)[0] intercept
Total fit time: 9.852 seconds
                               SARIMAX Results
==========================================================================================
Dep. Variable:                           y   No. Observations:                  220
Model:               SARIMAX(1, 0, 2)   Log Likelihood                -536.979
Date:                Sun, 04 Dec 2022   AIC                           1083.957
Time:                        03:46:25   BIC                           1100.925
Sample:                             0   HQIC                          1090.809
                                  - 220
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      6.3221      0.709      8.915      0.000       4.932       7.712
ar.L1         -0.8699      0.111     -7.860      0.000      -1.087      -0.653
ma.L1          1.0720      0.115      9.310      0.000       0.846       1.298
ma.L2          0.2498      0.067      3.726      0.000       0.118       0.381
sigma2         7.7142      0.631     12.234      0.000       6.478       8.950
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):               145.84
Prob(Q):                              0.99   Prob(JB):                         0.00
Heteroskedasticity (H):               1.26   Skew:                             1.37
Prob(H) (two-sided):                  0.33   Kurtosis:                         5.90
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
CPU times: user 20.4 s, sys: 16.4 s, total: 36.8 s
Wall time: 9.86 s
```

Рисунок 3.29 – Результат

На рисунку 3.29 можна побачити, що отримана модель є (1, 0, 2). Якщо ж подивитись на параметр P, то можна побачити, що він є меншим за 0.005, отже модель нормальна.

Далі запускається діагностика, яка виводить графіки похибок (3.30).

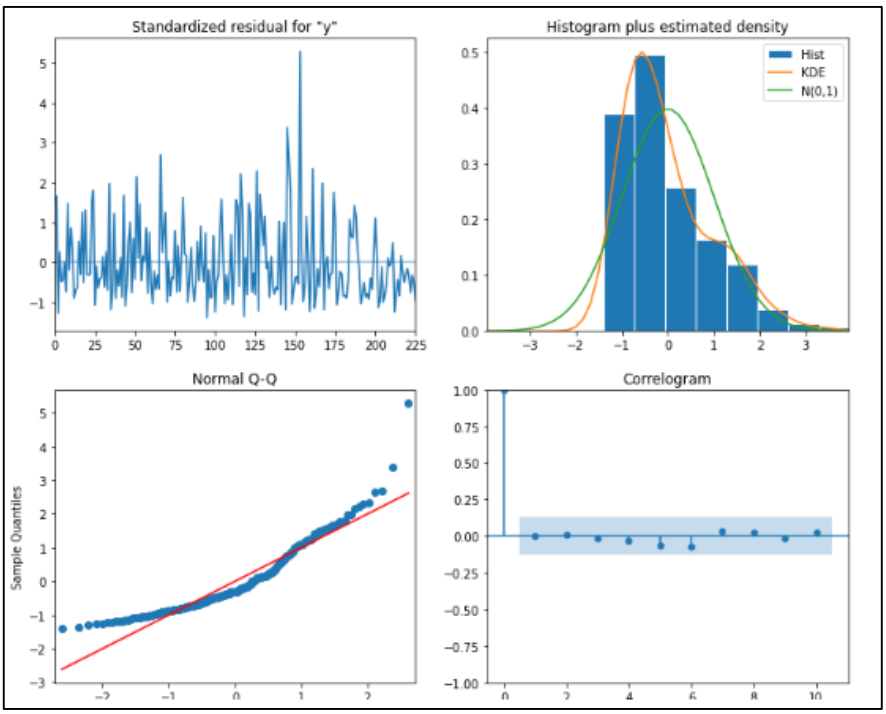


Рисунок 3.30 – Графіки похибок

Дивлячись на рисунок 3.30, можна зробити висновок, що похибки знаходяться приблизно навколо нуля, проте має місце деякі викиди, що може свідчити про те, що дисперсія не всюди однакова. Дивлячись на графік нормального закону розподілу, видно, що похибки вписуються в нормальний закон розподілу, проте досить добре видно, що є зміщення вправо. Якщо ж розглядати, на скільки квантилів похибки зміщені вздовж червоної лінії, можна сказати, що є відхилення. Розглядаючи ж корелограму, видно, що палички на ній не виходять з заданого діапазону. Тому можна зробити висновок, що для даного завдання модель ARIMA не дасть високу точність.

3.2.3 Тренування ML моделей

Наступним кроком було обрано п‘ять моделей машинного навчання, які будуть використовуватись для аналізу даних. Серед них: Linear Regression, Support Vector Machines, Random Forest Regressor, Bagging Regressor, XGB Regressor (рис. 3.31, 3.32).

```
# Linear Regression
n = len(models)
models.loc[n, 'name'] = 'Linear Regression'
models.at[n, 'model'] = LinearRegression()
models.at[n, 'param_grid'] = {'fit_intercept' : [True, False]}


# Support Vector Machines
n = len(models)
models.loc[n, 'name'] = 'Support Vector Machines'
models.at[n, 'model'] = SVR()
models.at[n, 'param_grid'] = {'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
                              'C': np.linspace(1, 15, 15),
                              'tol': [1e-3, 1e-4]
                             }

    # Random Forest Classifier
n = len(models)
models.loc[n, 'name'] = 'Random Forest Regressor'
models.at[n, 'model'] = RandomForestRegressor()
models.at[n, 'param_grid'] = {'n_estimators': [40, 50, 60, 80],
                              'min_samples_split': [30, 40, 50, 60],
                              'min_samples_leaf': [10, 12, 15, 20, 50],
                              'max_features': ['auto'],
                              'max_depth': [3, 4, 5, 6]
                             }

# Bagging Classifier
n = len(models)
models.loc[n, 'name'] = 'Bagging Regressor'
models.at[n, 'model'] = BaggingRegressor()
models.at[n, 'param_grid'] = {'max_features': np.linspace(0.05, 0.8, 1),
                              'n_estimators': [3, 4, 5, 6],
                              'warm_start' : [False]
                             }

# XGB Classifier
n = len(models)
models.loc[n, 'name'] = 'XGB Regressor'
models.at[n, 'model'] = xgb.XGBRegressor()
models.at[n, 'param_grid'] = {'n_estimators': [50, 70, 90],
                              'learning_rate': [0.01, 0.05, 0.1, 0.2],
                              'max_depth': [3, 4, 5]
                             }
```

Рисунок 3.31 – Моделі машинного навчання

	name	model	param_grid
0	Linear Regression	LinearRegression()	{'fit_intercept': [True, False]}
1	Support Vector Machines	SVR()	{'kernel': ['linear', 'poly', 'rbf', 'sigmoid'], 'C': [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0], 'tol': [0.001, 0.0001]}
2	Random Forest Regressor	RandomForestRegressor()	{'n_estimators': [40, 50, 60, 80], 'min_samples_split': [30, 40, 50, 60], 'min_samples_leaf': [10, 12, 15, 20, 50], 'max_features': ['auto'], 'max_depth': [3, 4, 5, 6]}
3	Bagging Regressor	BaggingRegressor()	{'max_features': [0.05], 'n_estimators': [3, 4, 5, 6], 'warm_start': [False]}
4	XGB Regressor	XGBRegressor(base_score=None, booster=None, callbacks=None,\n colsample_bylevel=None, colsample_bynode=None,\n colsample_bytree=None, early_stopping_rounds=None,\n ...	{'n_estimators': [50, 70, 90], 'learning_rate': [0.01, 0.05, 0.1, 0.2], 'max_depth': [3, 4, 5]}

Рисунок 3.32 – Додавання моделей

На рисунку 3.31, можна побачити лістинг коду, в якому видно, які моделі додаються, та налаштовуються параметри їх аналізу та прогнозування.

Далі відбувається прогноз на основі валідаційних даних, який згодом складається в один дата фрейм (рис. 3.33).

```
Tuning model 'Linear Regression'
Best parameters: {'fit_intercept': True}

Tuning model 'Support Vector Machines'
Best parameters: {'C': 3.0, 'kernel': 'rbf', 'tol': 0.0001}

Tuning model 'Random Forest Regressor'
Best parameters: {'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 12, 'min_samples_split': 50, 'n_estimators': 50}

Tuning model 'Bagging Regressor'
Best parameters: {'max_features': 0.05, 'n_estimators': 4, 'warm_start': False}

Tuning model 'XGB Regressor'
Best parameters: {'learning_rate': 0.05, 'max_depth': 3, 'n_estimators': 50}

CPU times: user 5min 45s, sys: 2.84 s, total: 5min 48s
Wall time: 3min 41s
```

Рисунок 3.33 – Дата фрейм з результатами прогнозу

3.3 Вибір оптимальної моделі.

Щоб обрати оптимальну модель для розв’язання поставленої задачі, а саме зробити прогноз концентрації нітратів у річковій воді Південного Бугу, всі дані, які було отримано в результаті роботи з моделями: машинного навчання, моделлю ARIMA та Facebook Prophet було поєднано в один дата фрейм. В якому, за трьома основними метриками було обрано найкращу, та найточнішу модель, яку можна використовувати в даному випадку (рис. 3.34).

	name_model	type_data	r2_score	rmse	mape
6	Prophet_10_days_3_order	valid	0.092209	0.449807	20.182881
7	Prophet_10_days_12_order	valid	-0.033513	0.479945	22.262139
4	Prophet_5_days_3_order	valid	-0.138636	0.503762	23.691246
2	Prophet_3_days_3_order	valid	-0.343909	0.547291	28.841
5	Prophet_5_days_12_order	valid	-0.457495	0.56995	28.945301
0	Prophet_2_days_3_order	valid	-0.470179	0.572425	32.297175
3	Prophet_3_days_12_order	valid	-1.228396	0.704741	40.400179
1	Prophet_2_days_12_order	valid	-2.188942	0.843056	49.590321
9	Support Vector Machines	valid	-545.849335	11.039947	609.034317
12	XGB Regressor	valid	-909.302139	14.243803	789.981221
10	Random Forest Regressor	valid	-949.259586	14.553059	801.972235
11	Bagging Regressor	valid	-986.785947	14.837632	819.560146
8	Linear Regression	valid	-1156.331621	16.060611	874.565124

Number of models built - 13

Рисунок 3.34 – Фрейм з даними про моделі

В результаті обробки даних, і порівняння всіх моделей, була обрана найкраща, це модель Facebook Prophet.

Після чого модель було перетреновано на збільшеному дата сеті, який складається із навчальних і валідаційних даних. Оптимальною для прогнозу обрано «Prophet_10_days_3_order». За метрикою r2score вона показала точність 0,09, за метрикою rmse 0,44 моль/м³, за метрикою MAPE 20% (рис 3.35).

```
Optimal model by metrics "r2_score" is "Prophet_10_days_3_order" with type "Prophet" parameters [10, 3]
18:00:15 - cmdstanpy - INFO - Chain [1] start processing
18:00:15 - cmdstanpy - INFO - Chain [1] done processing
```

	name_model	r2_score	rmse	mape	params
6	Prophet_10_days_3_order	0.092209	0.449807	20.182881	[10, 3]

```
18:00:17 - cmdstanpy - INFO - Chain [1] start processing
18:00:17 - cmdstanpy - INFO - Chain [1] done processing
Optimal model by metrics "rmse" is "Prophet_10_days_3_order" with type "Prophet" parameters [10, 3]
```

	name_model	r2_score	rmse	mape	params
6	Prophet_10_days_3_order	0.092209	0.449807	20.182881	[10, 3]

```
18:00:18 - cmdstanpy - INFO - Chain [1] start processing
18:00:18 - cmdstanpy - INFO - Chain [1] done processing
Optimal model by metrics "mape" is "Prophet_10_days_3_order" with type "Prophet" parameters [10, 3]
```

Рисунок 3.35 – Виведення найкращої моделі

Отримані результати було візуалізовано для кращого сприйняття, (рис 3.36 - 3.38).

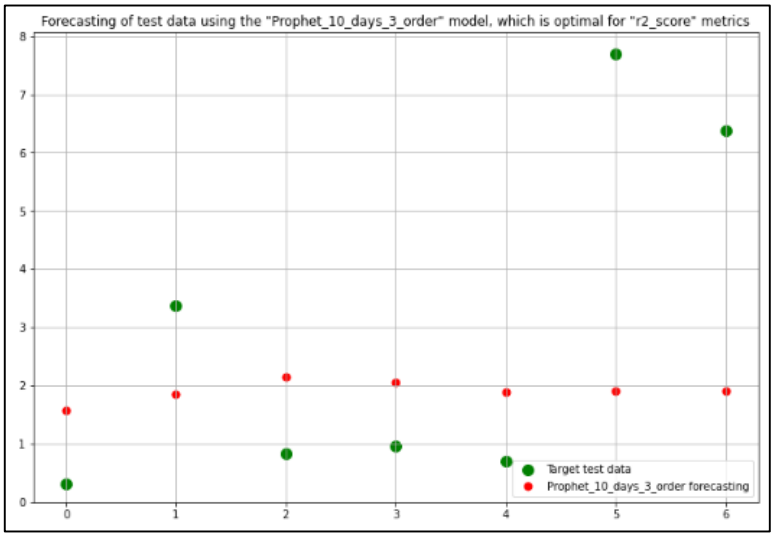


Рисунок 3.36 – Графік прогнозу концентрації нітратів за моделлю, оптимальною за метрикою r2score

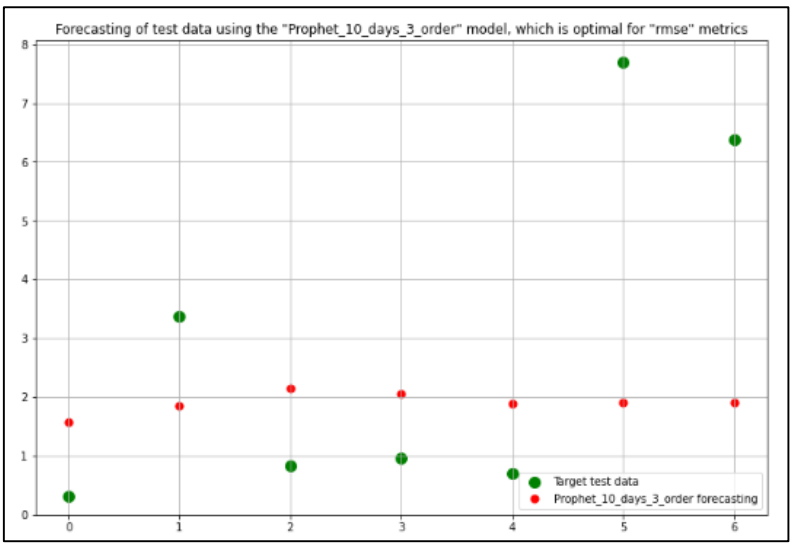


Рисунок 3.37 – Графік прогнозу концентрації нітратів за моделлю, оптимальною за метрикою rmse

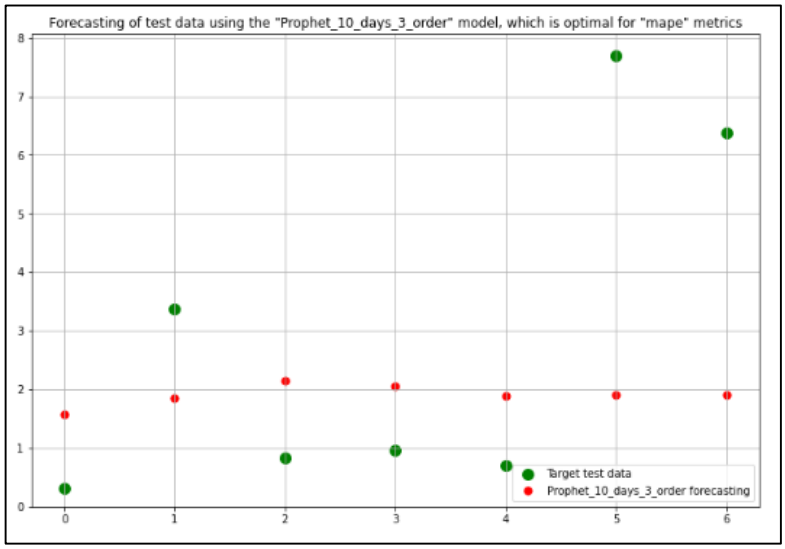


Рисунок 3.38 – Графік прогнозу концентрації нітратів за моделлю, оптимальною за метрикою mape

Дивлячись на графіки прогнозу, можна побачити, що непогано передбачається значення за перші 5 днів, проте напрямок зміни не вгадується. Тому можна зробити висновок, що дані, які були отримані з порталу моніторингу водних ресурсів, хоч і відносно свіжі, проте досить сильно зашумлені, і на основі них досить важко зробити гарний прогноз. Тому для отримання кращого рішення даної проблеми, потрібні або нові фічери або ж дані за нові дати.

Наступним кроком було проведено аналіз важливості ознак. Це було зроблено для того, щоб оцінити важливість та цінність ознак для моделі, яка була обрана найкращою.

Для цього, за допомогою бібліотеки SHAP були побудовані діаграми важливості ознак, щоб змоделювати вміст нітратів у воді (рис.3.39, 3.40).

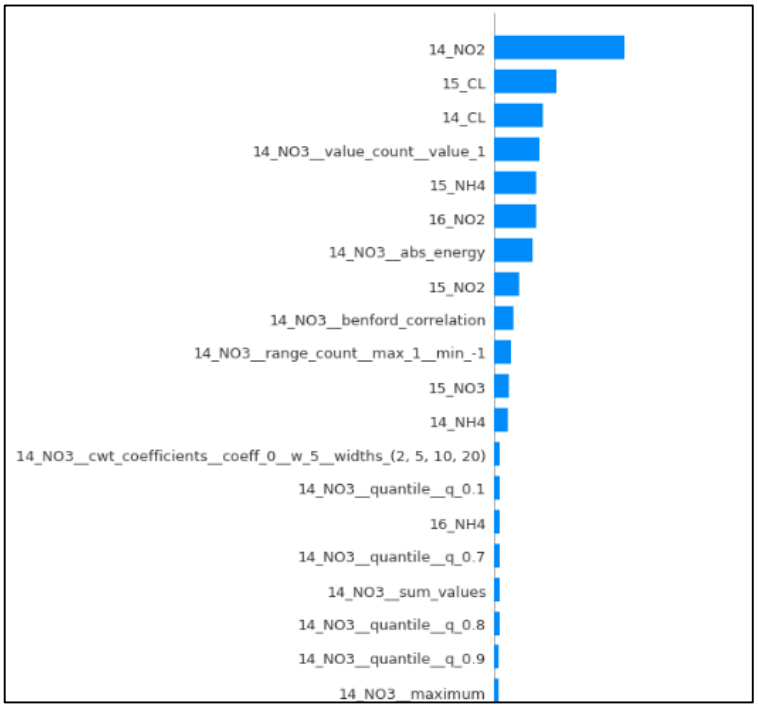


Рисунок 3.39 – Діаграма важливості ознак

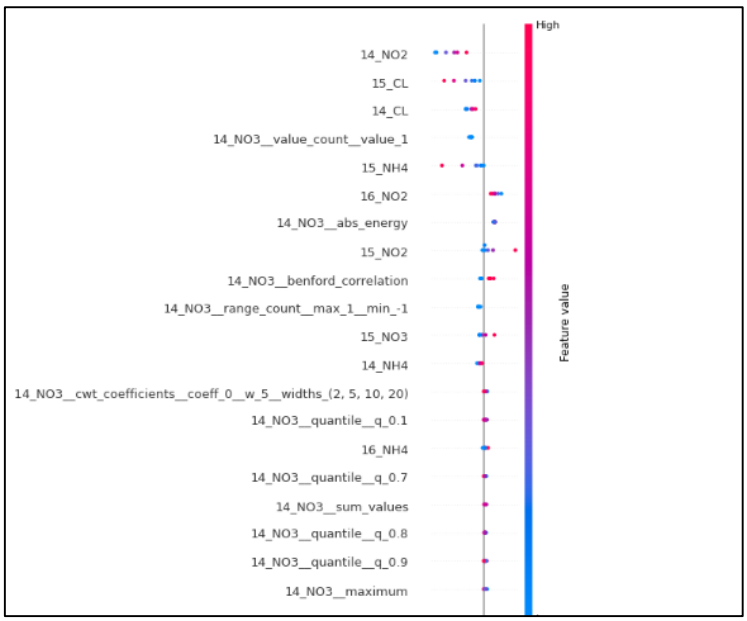


Рисунок 3.40 - Детальна діаграма важливості ознак

Далі, за допомогою бібліотеки ELI5 було побудовано діаграму перестановки для найкращої моделі, для того, щоб змоделювати вміст нітратів у воді (рис.3.41)

Weight	Feature
0.0040 ± 0.0067	14_NO3__benford_correlation
0.0035 ± 0.0084	15_NO3
0.0026 ± 0.0059	14_NH4
0.0009 ± 0.0011	14_NO3__minimum
0.0008 ± 0.0013	14_NO3__cwt_coefficients__coeff_0__w_10__widths_(2, 5, 10, 20)
0.0007 ± 0.0079	16_NO2
0.0006 ± 0.0010	14_NO3__median
0.0006 ± 0.0015	14_NO3__maximum
0.0005 ± 0.0011	14_NO3__quantile__q_0.4
0.0005 ± 0.0018	14_NO3__quantile__q_0.8
0.0004 ± 0.0012	14_NO3__cwt_coefficients__coeff_0__w_20__widths_(2, 5, 10, 20)
0.0004 ± 0.0020	14_NO3__sum_values
0.0004 ± 0.0024	14_NO3__mean
0.0003 ± 0.0016	14_NO3__quantile__q_0.7
0.0002 ± 0.0015	14_NO3__cwt_coefficients__coeff_0__w_2__widths_(2, 5, 10, 20)
0.0002 ± 0.0012	14_NO3
0.0002 ± 0.0004	14_NO3__abs_energy
0.0002 ± 0.0013	14_NO3__quantile__q_0.6
0.0001 ± 0.0006	14_NO3__quantile__q_0.1
0.0001 ± 0.0019	14_NO3__quantile__q_0.2
	... 19 more ...

Рисунок 3.41 – Таблиця важливості ознак

Дивлячись на рисунок 3.41, можна зробити такий висновок, що на даній діаграмі, ознака, яка має найбільшу важливість, це «14_NO3», «15_NO3», «14_NH4», проте найменш цінною виявилась «16_NO2» .

3.4 Прогнозування концентрації нітратів у Ладижинському водосховищі, на якому розташований спортивний табір ВНТУ

Оскільки завданням роботи було спрогнозувати якість води на вміст нітратів у воді річки Південний Буг, тому доцільно буде проаналізувати, та спрогнозувати ще й якість води на території спортивного табору ВНТУ «Супутник», який знаходиться біля с. Степашки. Тому було обрано пост №9, який знаходиться поблизу м. Ладижин як цільовий, який буде досліджуватись за постом №10, який знаходиться вище по течії. Так буде можливість зрозуміти, чи є забруднювачі нижче по течії річки, які впливають на якість води. За досліджувані речовини було обрано: NO3, NH4, NO2, CL. Цільовою речовиною виступили нітрати NO3, а інші було обрано, через те, що вони взаємодіють між собою у воді.

Першим кроком було проведено аналіз даних, який показав, що за останній період з 2019 року на посту №9 замірів не проводилось, тобто дані в дата сеті застарівші. Зробивши візуалізацію згенерованої таблиці з даними, за допомогою бібліотеки Pandas Profiling, стало зрозуміло, що хоча даних і не достатньо, проте досить чітко видно, концентрація нітратів у воді на посту №9 є більшою за концентрацію на посту під номером №10. Це може свідчити про те, що вище по течії є підприємство, або ж сільськогосподарське угіддя, яке забруднює воду шкідливими хімікатами. Інформація про нітрати та хлор з поста №9 зображена на рисунках (3.42, 3.43).

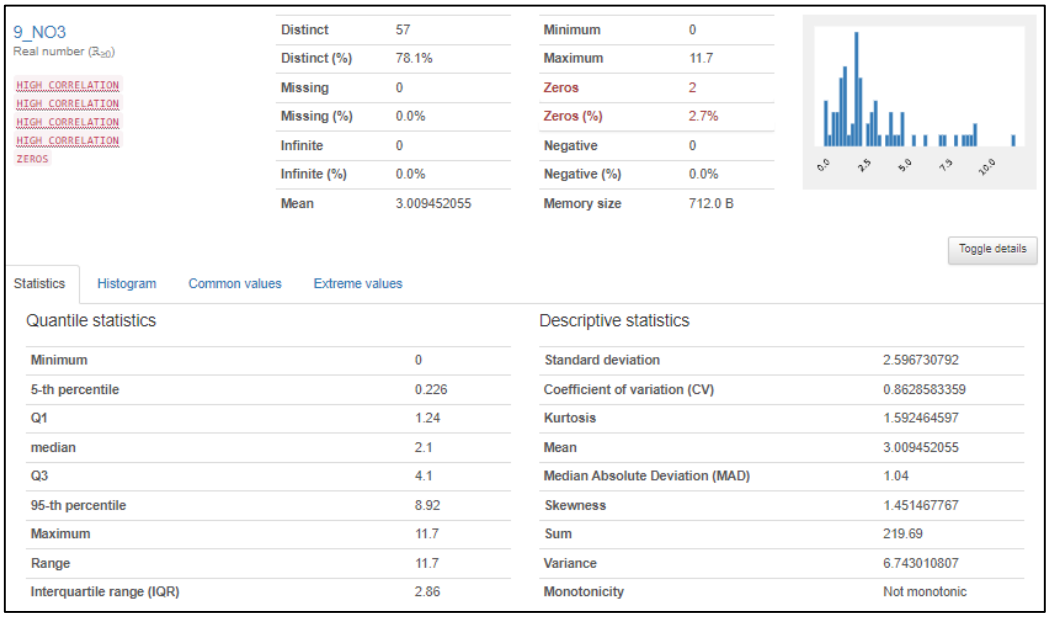


Рисунок 3.42 – Вміст нітратів

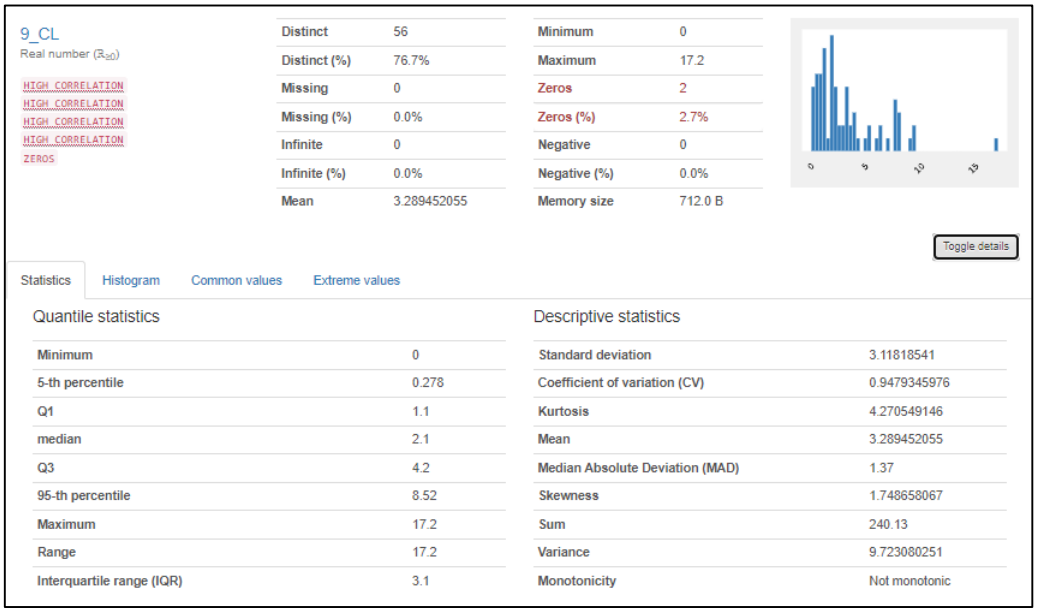


Рисунок 3.43 – Вміст хлору

Інформація про нітрити та амоній з поста №9 зображена на рисунках (3.44, 3.45).

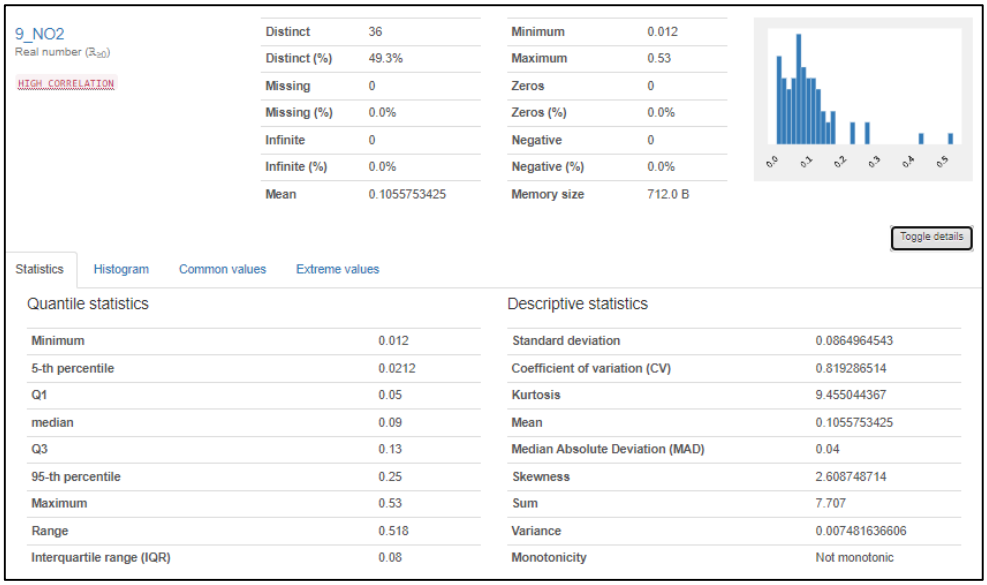


Рисунок 3.44 – Вміст нітритів

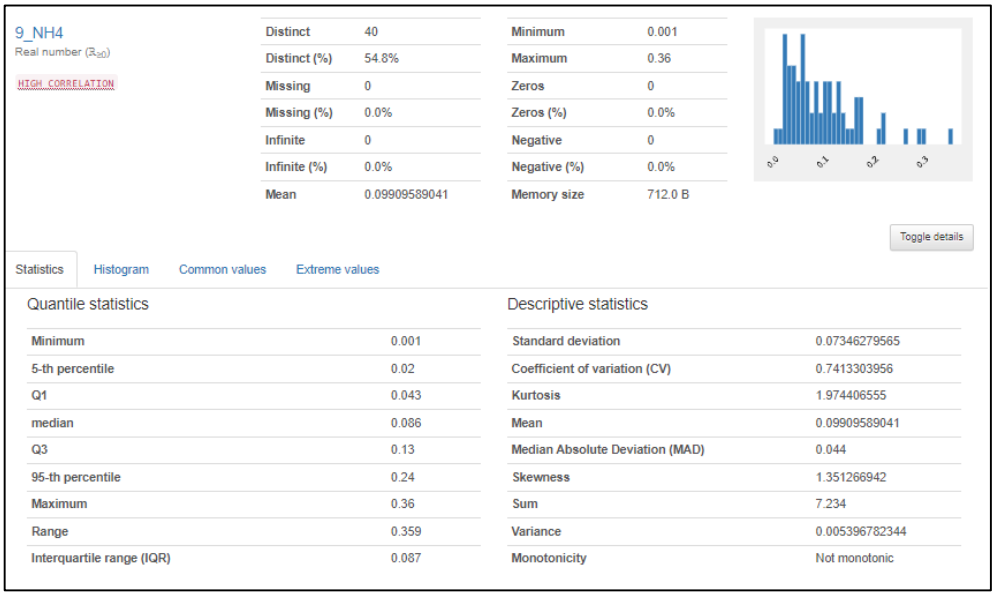


Рисунок 3.45 – Вміст амонію

Далі буде огляд інформації про нітратів та хлору з поста №10 (рис. 3.46 – 3.47).

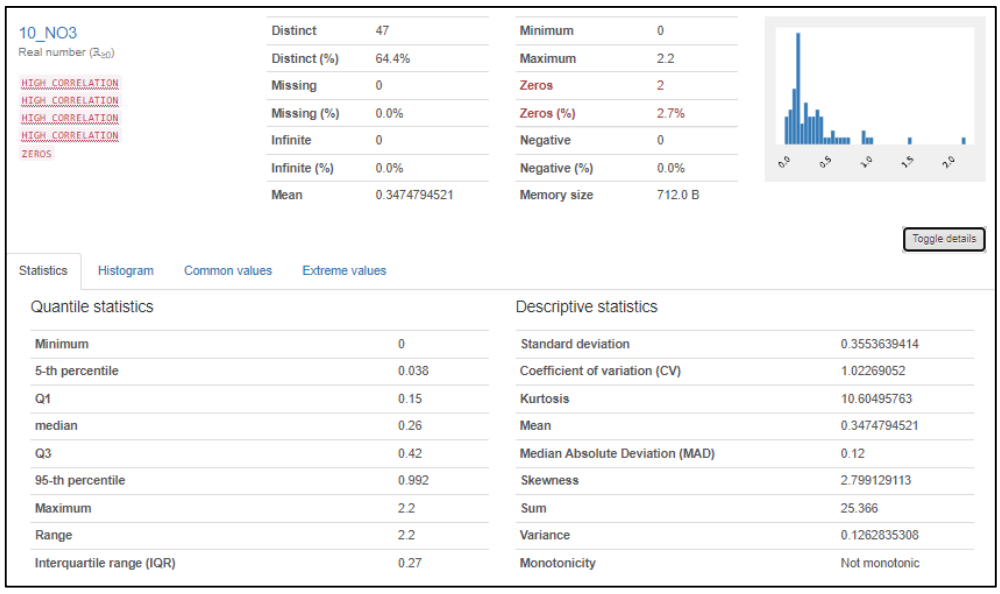


Рисунок 3.46 – Вміст нітратів

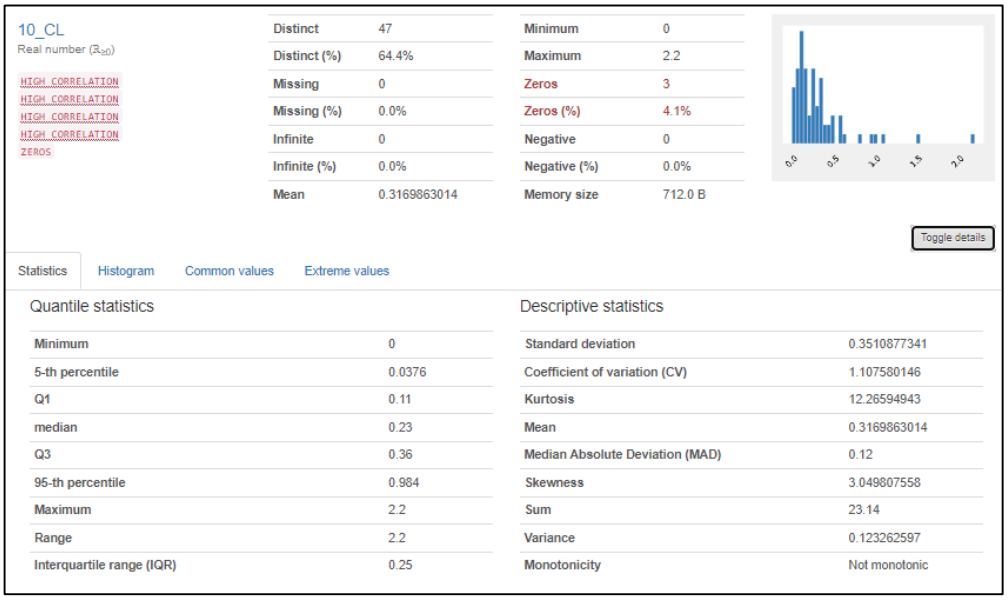


Рисунок 3.47 – Вміст хлору

Інформація про нітрити та амоній з поста №10 зображена на рисунках (3.48, 3.49)

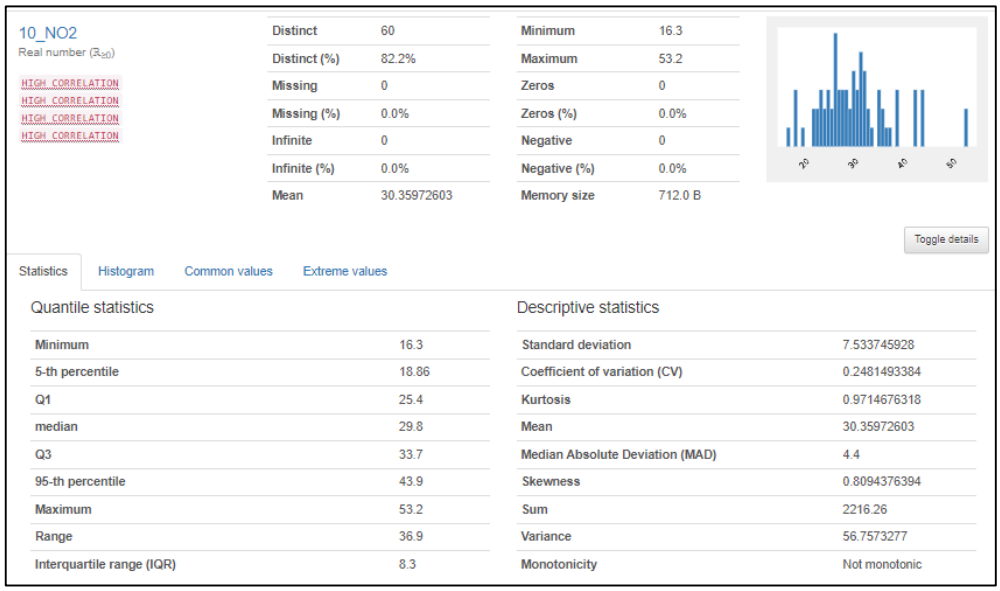


Рисунок 3.48 – Вміст нітритів

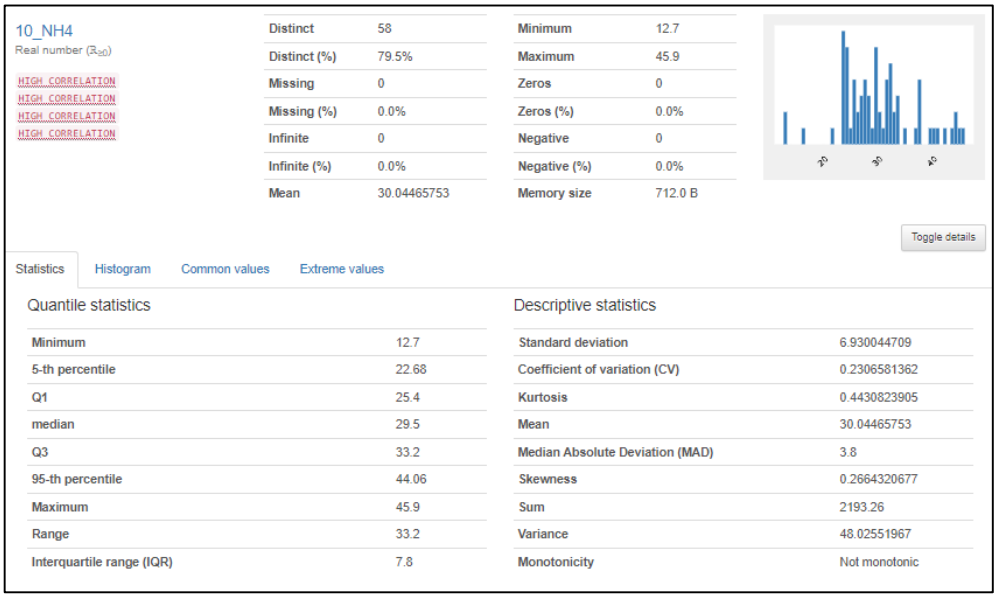


Рисунок 3.49 – Вміст амонію

Проаналізувавши виведені графіки, можна зробити висновок, що концентрація нітратів у воді на посту №9 є більшою за концентрацію на посту під номером №10. Це може свідчити про те, що вище по течії є підприємство, або ж сільськогосподарське угіддя, яке забруднює воду шкідливими хімікатами.

Наступним кроком, відбулась перевірка дата сету на вміст аномальних дат. Для цього створюється таблиця, і з неї виділяються або занадто високі, або ж занадто низькі показники (рис. 3.50).

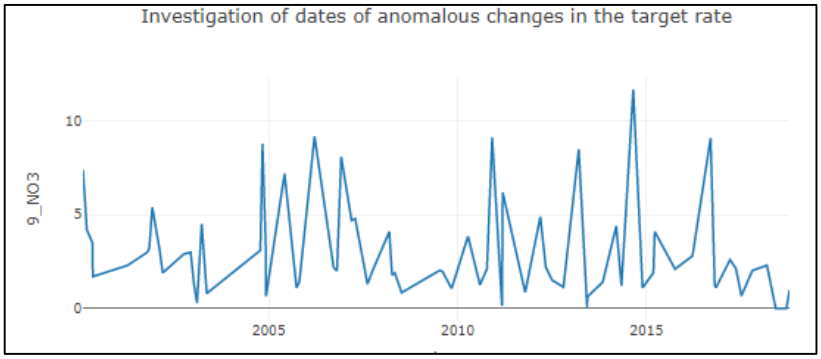


Рисунок 3.50– Графік даних

Судячи з графіку, видно що за останній час є п’ять чітких аномальних дат (рис. 3.51, 3.52)

```
# Synthesis dataframe with anomalous dates for Facebook Prophet
if is_anomalies:
    anomalous_dates = ['2016-09-02', '2014-08-04', '2013-03-04', '2010-12-04', '2006-03-15']
    holidays_df = pd.DataFrame(columns = ['ds', 'lower_window', 'upper_window', 'prior_scale'])
    holidays_df['ds'] = anomalous_dates
    holidays_df['holiday'] = 'anomalous_dates'
    holidays_df['lower_window'] = 0
    holidays_df['upper_window'] = 0
    holidays_df['prior_scale'] = 10
    display(holidays_df)
```

	ds	lower_window	upper_window	prior_scale	holiday
0	2016-09-02	0	0	10	anomalous_dates
1	2014-08-04	0	0	10	anomalous_dates
2	2013-03-04	0	0	10	anomalous_dates
3	2010-12-04	0	0	10	anomalous_dates
4	2006-03-15	0	0	10	anomalous_dates

Рисунок 3.51 – Синтез аномальних даних

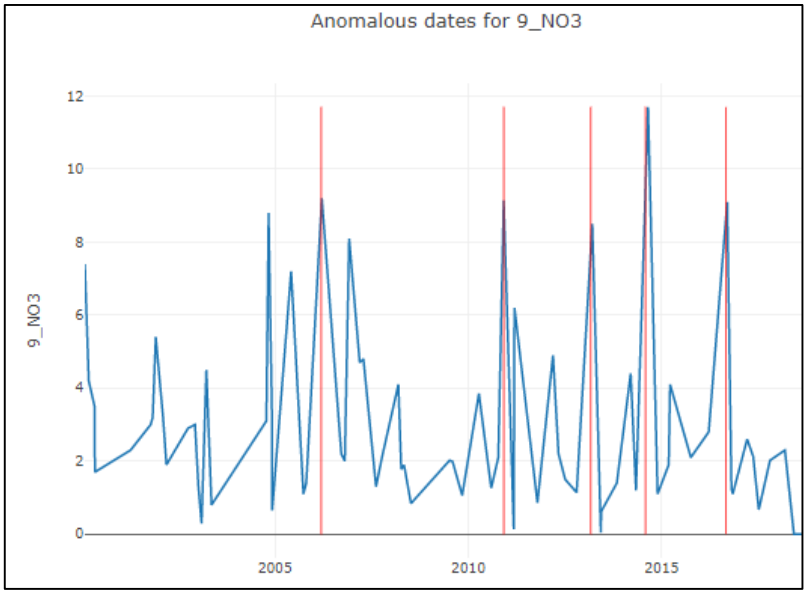


Рисунок 3.52 – Графік аномальних даних

За проаналізованими даними проведено навчання моделей, та зроблено прогноз використовуючи такі моделі, як ARIMA, Facebook Prophet, та моделей машинного навчання, так як Linear Regression, Support Vector Machines, Random Forest Regressor, Bagging Regressor, XGB Regressor.

В результаті, модель ARIMA не змогла запуститись, через малу кількість наявних даних, тому довелось її відключити. Проте моделі машинного навчання, та Facebook Prophet змогли дати певний результат (рис. 3.53)

	name_model	type_data	r2_score	rmse	mape
6	Prophet_10_days_3_order	valid	0.197355	2.224246	48.620928
1	Prophet_2_days_12_order	valid	0.051431	2.417995	52.760426
0	Prophet_2_days_3_order	valid	0.050465	2.419226	52.822802
2	Prophet_3_days_3_order	valid	-0.048348	2.541989	54.944578
3	Prophet_3_days_12_order	valid	-0.049375	2.543233	54.950177
7	Prophet_10_days_12_order	valid	0.163517	2.270646	54.96594
4	Prophet_5_days_3_order	valid	-0.04068	2.532675	63.726221
5	Prophet_5_days_12_order	valid	-0.063076	2.559782	66.379203
12	XGB Regressor	valid	-13.999587	9.615251	414.227708
9	Support Vector Machines	valid	-22.304243	11.985012	517.751066
8	Linear Regression	valid	-29.12435	13.626358	575.126259
11	Bagging Regressor	valid	-34.541706	14.800967	621.177079
10	Random Forest Regressor	valid	-35.748466	15.050141	641.179689

Рисунок 3.53 – Результат прогнозування

В результаті обробки даних, і порівняння всіх моделей, була обрана найкраща, це модель Facebook Prophet. Дані виявились дуже зашумлені, тому за метрикою r2score, показник склав – 0.19, за метрикою rmse – 2.22м³, а відносна похибка mape – 48.6% (рис. 3.54)

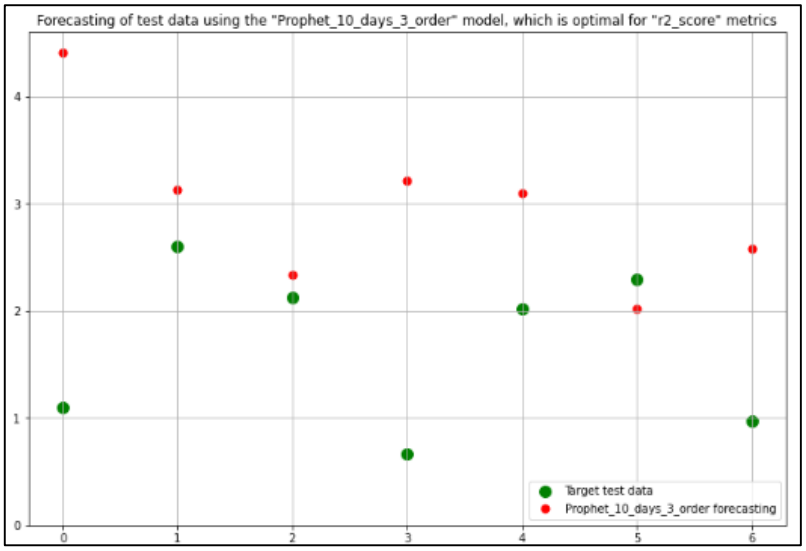


Рисунок 3.54 – Графік прогнозу концентрації нітратів

Провівши аналіз важливості ознак, за допомогою діаграми важливості ознак стало зрозуміло, що найбільшу важливість, має «10_NO3», «10CL», «9NH4» проте найменш цінною виявилась «10_NO2» (рис. 3.55)

Weight	Feature
0.0748 ± 0.2028	10_NO3
0.0550 ± 0.0745	9_NH4
0.0227 ± 0.3479	10_CL
0.0006 ± 0.0093	10_NO2
0 ± 0.0000	9_NO3__value_count__value_0
0 ± 0.0000	9_CL
0 ± 0.0000	9_NO3__sum_values
0 ± 0.0000	9_NO3__abs_energy
0 ± 0.0000	9_NO3__median
0 ± 0.0000	9_NO3__mean
0 ± 0.0000	9_NO3__maximum
0 ± 0.0000	9_NO3__absolute_maximum
0 ± 0.0000	9_NO3__minimum
0 ± 0.0000	9_NO3__quantile__q_0.1
0 ± 0.0000	9_NO3__quantile__q_0.2
0 ± 0.0000	9_NO3__benford_correlation
0 ± 0.0000	9_NO3__quantile__q_0.4
0 ± 0.0000	9_NO3__fft_coefficient__attr_"real"__coeff_0
0 ± 0.0000	9_NO3__root_mean_square
0 ± 0.0000	9_NO3__count_below__t_0
	14 more

Рисунок 3.55 – Таблиця важливості ознак

3.5 Висновки

Для того, щоб краще зрозуміти принцип дії інформаційної технології, розроблено алгоритм роботи, та побудовано його блок-схему.

Реалізувавши технологію по прогнозуванню концентрації нітратів у воді річки Південний Буг, було виявлено оптимальну модель для вирішення цієї задачі. Нею виявилась Facebook Prophet, яка показала за метрикою r2score точність 0,09, за метрикою rmse 0,44 моль/м³, за метрикою MAPE 20%. Візуалізувавши графіки прогнозу, зроблено висновки, що непогано передбачається значення за перші 5 днів, проте напрямок зміни не вгадується. Це пов’язано з тим, що дані, які були отримані з порталу моніторингу водних ресурсів, хоч і відносно свіжі, проте досить сильно зашумлені і на основі цих даних досить важко зробити гарний прогноз. Також було побудовано діаграми важливості ознак для визначення того, які з ознак найбільше впливають на результат, щоб згодом відсіяти непотрібні. Найважливішими ознаками виявились «14_NO3», «15_NO3», «14_NH4», проте найменш цінною виявилась «16_NO2».

Паралельно з цим було проведено розвідувальний аналіз даних та прогнозування концентрації нітратів нітратів у Ладижинському водосховищі, на якому розташований спортивний табір ВНТУ «Супутник». Попередній аналіз даних показав, що дані є застарілими, зашумленими та мають велику кількість аномалій. Зробивши прогноз, найкраща модель для прогнозування Facebook Prophet показала низьку точність, що підтвердило результати попереднього аналізу. Провівши аналіз важливості ознак, стало зрозуміло, які з ознак найбільше впливають на результат, а саме «10_NO3», «10_CL», «9_NH4», проте найменш цінною виявилась «10_NO2»

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нової інформаційної технології прогнозування концентрації нітратів у річковій воді Південного Бугу. Особливістю розробки є те, що інформаційна технологія прогнозування концентрації нітратів у воді басейну річки Південний Буг на території Вінницької області дозволяє, завдяки методам машинного навчання, підвищити точність цього прогнозування, що було досягнуто за рахунок використання складніших бібліотек для попереднього аналізу даних та складніших моделей з використанням часових рядів на основі заданого датасету. За аналог було взято технологію аналізу концентрації амонію у річковій воді Південного Бугу «WQ SB river : EDA and Forecasting» з використанням 3х моделей машинного навчання. Таких як: Linear Regression, Random Forest Regressor, XGBoost Regressor. Орієнтовна вартість розробки такої систем 1500$ або 60000 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1 [28].

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність в реаль-них умовах

Продовження табл. 4.1

Ринкові переваги					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 4.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 4.2

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Радецький О.В.	Пасічнюк Д.В.	Лопухов Б.Л.
	Бали		
Технічна здійсненність концепції	3	3	4
Наявність аналогів на ринку	3	3	4
Цінова політика	3	4	3
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	3	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	4	4	3
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	3	3
Супровідна документація	3	3	4
Сума	41	40	42
Середньоарифметична сума балів	(41+40+42) / 3 = 41		

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок того, що інформаційна технологія прогнозування концентрації нітратів у воді басейну річки Південний Буг на території Вінницької області дозволяє, завдяки методам машинного навчання, підвищити точність цього прогнозування, що було досягнуто за рахунок використання складніших бібліотек для попереднього аналізу даних та складніших моделей з використанням часових рядів на основі заданого датасету з використанням моделей часових рядів.

4.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t, \quad (4.1)$$

де М – місячний посадовий оклад конкретного розробника (дослідника), грн.;

$Т_р$ – число робочих днів за місяць, 22 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 4.4.

Таблиця 4.4 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	35000	1590,91	38	60454,545
Програміст	32000	1454,55	38	55272,727
Всього				115727,27

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

Додаткова заробітна плата розробників, які брати участь в розробці обладнання/програмного продукту.

Додаткову заробітну плату прийнято розраховувати як 11 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 11\ \% / 100\ \% \quad (4.2)$$

$З_д = (115727,27 \cdot 11\ \% / 100\ \%) = 12730,00$ (грн.)

Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \quad (4.3)$$

$Н_з = (115727,27 + 12730,00) \cdot 22\ \% / 100\ \% = 28260,60$ (грн.)

Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (4.4)$$

де Ц – балансова вартість обладнання, грн.;

Т – термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ – термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 22500 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,73 міс.

$$A_{обл} = \frac{22500}{2} \times \frac{1{,}73}{12} = 1919{,}318 \; грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.5.

Так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних активів є безкоштовною, то $B_{нем.ак.}$ = 0 грн.

Таблиця 4.5 – Амортизаційні відрахування на матеріальні та нематеріальні ресурси для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія (ACER Aspire 3 A315-23)	22500	2	1,73	1619,318
Офісне обладнання (меблі)	20000	4	1,73	719,697
Приміщення	1080000	20	1,73	7772,727
Всього				10111,74

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (4.5)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,4 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9.

$В_е = 0,9 \cdot 0,4 \cdot 8 \cdot 38 \cdot 6,2 = 678,528$ (грн.)

Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.6)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$I_в$ = 115727,27 * 58% / 100% = 67121,82 (грн.)

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.7)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$Н_{нзв}$ = 115727,27 * 122 % / 100 % = 141187 (грн.)

4.2.9 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$B_{заг}$ = 115727,27+12730,00+28260,60+10111,74+678,53+67121,82+
+141187 = 375817,23 грн.

Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} \text{ (грн)}, \qquad (4.8)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$ЗВ$ = 375817,23 / 0,5 = 751634 грн.

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів

тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

в) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб'єктів (зацікавлених осіб) цього попиту;

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов'язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

- абсолютного економічного ефекту (чистого дисконтованого доходу);
- внутрішньої економічної дохідності (внутрішньої норми дохідності);
- терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об'єднати, враховуючи визначені ситуації з відповідними умовами.

Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.9)$$

де $\pm\Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$;

Цб – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

ρ – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 25000 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *2000* грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 850 шт., протягом другого року – на 550 шт., протягом третього року на 300 шт. До

моменту впровадження результатів наукової розробки реалізації продукту не було:

*ΔП₁ = (0*2000 + (25000 + 2000)*850)* 0,8333* 0,27) * (1 - 0,18) = 3920624,843 грн.*

*ΔП₂ = (0*2000 + (25000 + 2000)*(850+550)* 0,8333* 0,27) * (1 - 0,18) = 6974099,721 грн.*

*ΔП₃ = (0*2000 + (25000 + 2000)*(850+550+300)* 0,8333* 0,27) * (1 - 0,18) = 8468549,661 грн.*

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 19363274,23 грн.

Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t – період часу (в роках).

Збільшення прибутку ми отримаємо, починаючи з першого року:

ПП = (3920624,843/(1+0,1)[1])+(6974099,721/(1+0,1)[2])+(8468549,661/
/(1+0,1)[3]) = 3564204,40 + 5763718,778 + 6362546,703 = 15690469,88 грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (4.11)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$PV = 2 * 751634 = 1503268{,}94$ *грн.*

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \quad (4.12)$$

$E_{абс} = 15690469{,}88 - 1503268{,}94 = 14187200{,}95$ грн.

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так

званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.13}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 14187200{,}95/1503268{,}94} - 1 = 1{,}185$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{4.14}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,09…0,14);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0{,}14 + 0{,}05 = 0{,}19.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \tag{4.15}$$

$T_{ок}$ = 1 / 1,185 = 0,84 р.

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,84 роки, то фінансування даної наукової розробки є доцільним.

4.4 Висновки

Економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 751634 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,84 роки.

ВИСНОВКИ

В магістерській кваліфікаційній роботі розроблено інформаційну технологію аналізу та прогнозування концентрації нітратів у воді річки Південний Буг.

Проведено аналіз предметної області, розглянуто аналогічні рішення наявні на даний момент, обґрунтувано доцільність створення технології для прогнозування концентрації нітратів у воді річки Південний Буг.

Проведено аналіз інформаційних технологій які дозволяють виконати поставлену задачу. Обрано мову програмування Python, оскільки вона є досить популярною на даний момент, має велику кількість бібліотек для аналізу, обробки та прогнозування даних. Визначено бібліотеки, методи, та моделі які використовувались для аналізу даних та прогнозування концентрації нітратів у воді річки Південний Буг.

Так, для аналізу даних використано бібліотеки: NumPy, Pandas, Matplotlib, Sweetviz та Autoviz. Для прогнозування використано моделі Facebook Prophet, ARIMA/SARIMA, Linear Regression, Support Vector Machines, Random Forest Regressor, Bagging Regressor, XGB Regressor.

Проведено попередній аналіз даних, в результаті чого виявлено, що в наявному датасеті є аномалії та зашумлені дані. Відфільтрувавши їх створено ще одну вибірку з даними, використавши яку, отримано точніший результат.

Візуалізовано дані у вигляді графіків, за допомогою бібліотек Auto Viz, Sweet Wiz та Pandas Profiling Report по яким стало зрозуміло, що обрані для аналізу речовини (NH4, NO2, CL) взаємодіють у воді з нітратами NO3 та між собою, також мало місце незначне забруднення вниз по течії.

Для того, щоб краще зрозуміти принцип дії інформаційної технології, розроблено алгоритм роботи, та побудовано його блок-схему.

Реалізовано інформаційну технологію по прогнозуванню концентрації нітратів у воді річки Південний Буг. За результатами прогнозування визначено оптимальну модель для вирішення цієї задачі. Нею виявилась Facebook Prophet,

яка показала за метрикою r2score точність 0,09, за метрикою rmse 0,44 моль/м³, за метрикою MAPE 20%. Візуалізувавши графіки прогнозу, зроблено висновки, що непогано передбачається значення за перші 4 дні, проте напрямок зміни не вгадується. Це пов'язано з тим, що дані, які були отримані з порталу моніторингу водних ресурсів, хоч і відносно свіжі, проте досить сильно зашумлені і на основі цих даних досить важко зробити гарний прогноз. Також було побудовано діаграми важливості ознак. Найважливішими ознаками виявились «14_NO3», «15_NO3», «14_NH4», проте найменш цінною виявилась «16_NO2».

Також проведено розвідувальний аналіз даних та прогнозування концентрації нітратів на території спортивного табору ВНТУ «Супутник». Попередній аналіз даних показав, що дані є застарілими, зашумленими та мають велику кількість аномалій. Зробивши прогноз, найкраща модель для прогнозування Facebook Prophet показала доволі низьку точність, що підтвердило результати попереднього аналізу. Провівши аналіз важливості ознак, стало зрозуміло, які з ознак найбільше впливають на результат, ними виявились: «10_NO3», «10CL», «9NH4», проте найменш цінною виявилась «10_NO2».

Отже, з використанням розробленої інформаційної технології вдалось підвищити точність прогнозування концентрації нітратів у річковій воді Південного Бугу на основі заданого датасету з використанням моделей часових рядів. Але отримана точність свідчить про необхідність залучення більш точних, актуальних та регулярних даних моніторингу.

Проведено розрахунок витрат на розробку нового програмного продукту, сума яких складає 751634 гривень. Спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення. В результаті, можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,84 роки.

За результатами даної роботи була зроблена доповідь на тему «Ідентифікація та вибір оптимальної моделі для прогнозування концентрації нітратів у річковій воді Південного Бугу» на Всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (Вінниця, 2022-2023 рр.)» з публікацією тез.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лісовський Р.Р., Жуков С.О. Ідентифікація та вибір оптимальної моделі для прогнозування концентрації нітратів у річковій воді Південного Бугу. *Всеукраїнська науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи (Вінниця, 2022-2023 рр.)»*. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2023/author/submission/16828
2. Водний баланс Землі URL: https://uk.wikipedia.org/wiki/Водний_баланс_Землі
3. Водні ресурси України URL: https://uk.wikipedia.org/wiki/Водні_ресурси_України
4. Скільки води треба пити дорослим і дітям URL : https://moz.gov.ua/article/health/skilki-vodi-treba-piti-doroslim-i-ditjam
5. AI-ML-DS Training. L1T: NH4 - linear regression URL: https://www.kaggle.com/vbmokin/ai-ml-ds-training-l1t-nh4-linear-regression
6. Моніторинг та екологічна оцінка водних ресурсів України URL: http://monitoring.davr.gov.ua/EcoWaterMon/GDKMap/Index
7. Аналіз даних URL: https://uk.wikipedia.org/wiki/Аналіз_даних
8. Кореляційний аналіз URL: https://uk.wikipedia.org/wiki/Кореляційний_аналіз
9. Bruce E. Pease Leading Intelligence Analysis / Pease. E. Bruce Leading Analysis. – 2019.
10. Аналіз часових рядів URL: https://uk.wikipedia.org/wiki/Аналіз_часових_рядів
11. Регресійний аналіз URL: https://uk.wikipedia.org/wiki/Регресіний_аналіз
12. Python URL: https://ru.wikipedia.org/wiki/Python
13. NumPy URL: https://ru.wikipedia.org/wiki/NumPy
14. Pandas URL: https://ru.wikipedia.org/wiki/Pandas
15. Matplotlib URL: https://ru.wikipedia.org/wiki/Matplotlib
16. Sweetviz: Automated EDA in Python URL: https://towardsdatascience.com/sweetviz-automated-eda-in-python-a97e4cabacde

17. Autoviz: Automatically Visualize any Dataset URL: https://towardsdatascience.com/autoviz-automatically-visualize-any-dataset-75876a4eede4
18. Scikit-learn URL: https://uk.wikipedia.org/wiki/Scikit-learn
19. Машинне навчання URL: https://ru.wikipedia.org/wiki/Машинное_обучение
20. Дерева рішень URL: https://ru.wikipedia.org/wiki/Дерево_Решений
21. Random Forest URL: https://ru.wikipedia.org/wiki/Random_forest
22. Feng D. C. et al. Interpretable XGBoost-SHAP machine-learning model for shear strength prediction of squat RC walls : *Journal of Structural Engineering*. 2021. T. 147. №. 11. P. 1173.
23. Лінійна регресія URL: https://ru.wikipedia.org/wiki/Линейная_регрессия
24. Kadiyala, Akhil, and Ashok Kumar. "Applications of python to evaluate the performance of bagging methods."*Environmental Progress & Sustainable Energy* 37.5 (2018):P.155-159.
25. Garlapati, Anusha, et al. "Stock Price Prediction Using Facebook Prophet and Arima Models."*2021 6th International Conference for Convergence in Technology (I2CT)*". IEEE, 2021.
26. Time Series Forecasting With Prophet in Python. URL: https://machinelearningmastery.com/time-series-forecasting-with-prophet-in-python/
27. River Water Quality EDA and Forecasting. URL: https://www.kaggle.com/vbmokin/wq-southern-bug-river-01052021
28. Козловський В. О., Лесько О. Й., Кавецький В. В. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт : уклад. Вінниця : ВНТУ, 2021. 42 с.

Додаток А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Мокін В. Б.
«_19_» _____09_____ 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПРОГНОЗУВАННЯ КОНЦЕНТРАЦІЇ НІТРАТІВ У РІЧКОВІЙ ВОДІ ПІВДЕННОГО БУГУ»
08-53.МКР.002.02.000.ТЗ

Керівник: к.т.н., доц. каф. САІТ
_____ Жуков С.О
«_19_» _____09_____ 2022 р.

Розробив: студент гр. 2ІСТ-21м
_____Лісовський Р.Р.
«_19_» _____09_____ 2022 р.

Вінниця 2022

1. Підстава для проведення робіт

Підставою для виконання роботи є наказ № 203 по ВНТУ від «14» _09__ 2022 р., та індивідуальне завдання на МКР, затверджене протоколом № 3 засідання кафедри САІТ від «14» ____09____ 2022 р.

2. Джерела розробки:

– Датасети з Конкурса на платформі Kaggle - WQ SB river : EDA and Forecasting. NEW URL : https://www.kaggle.com/nikaapril/wq-sb-river-eda-and-forecasting-new

– Дані про моніторинг якості річкової води у р. П. Буг URL : http://monitoring.davr.gov.ua/EcoWaterMon/GDKMap/Index

3. Мета і призначення роботи:

Метою роботи є підвищення точності прогнозування концентрації нітратів у воді річки Південний Буг на основі заданого датасету з використанням моделей часових рядів.

4.Вихідні дані для проведення робіт:

– Дані конкурсу «River Water Quality EDA and Forecasting» платформи Kaggle;

– Електронна карта Вінницької області;

5.Методи дослідження:

– розвідувальний аналіз;

– прогнозування даних;

6. Етапи роботи і терміни їх виконання:

1. Аналіз предметної області..... 21.09 –28.09
2. Огляд проблем створення інформаційної технології..... 28.09 – 10.10
3. Аналіз даних концентрації нітратів..... 11.10 – 26.10
4. Реалізація інформаційної технології..... 26.10 – 05.11
5. Аналіз результатів прогнозування ..... 06.11 – 10.11
6. Економічна частина ..... 10.11 – 18.11
7. Оформлення пояснювальної записки..... 19.11 – 30.11

7. Очікувані результати та порядок реалізації:

За даними зі створів вище по течії, отримати прогноз концентрації нітратів нижче по течії за певний період часу, яий в порівнянні з аналогами буде набагато точнішим.

8. Вимоги до розробленої документації

Пояснювальна записка оформлена у відповідності до вимог «Методичних вказівок до виконання та оформлення магістерських кваліфікаційних робіт для студентів спеціальності 126 – «Інформаційні системи та технології» денної форми навчання».

9. Порядок приймання роботи

Публічний захист..... «19» ___грудня____ 2022 р.

Початок розробки.....« 21 » ___вересня___ 2022 р.

Граничні терміни виконання МКР .....«30 » __листопада__ 2022 р.

Розробив студент групи 2ІСТ-21м _____ Лісовський Р.Р.

Додаток Б

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: «Інформаційна технологія прогнозування концентрації нітратів у річковій воді Південного Бугу»
Тип роботи: магістерська кваліфікаційна робота
Підрозділ: кафедра САІТ
Науковий керівник: Жуков С.О. к.т.н., доц. каф. САІТ

Показники звіту подібності Unicheck

Оригінальність	99,7 %
Схожість	0,3 %

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.

Опис прийнятого рішення:

Робота допускається до захисту

Особа, відповідальна за перевірку Жуков С. О.

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

Автор роботи Лісовський Р.Р.

Керівник роботи Жуков С.О.

Додаток В

Лістинг програми

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

# 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 sklearn.linear_model import LinearRegression
from sklearn.svm import SVR, LinearSVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import BaggingRegressor, AdaBoostRegressor
from sklearn.neural_network import MLPRegressor
from prophet import Prophet
import xgboost as xgb
from xgboost import XGBRegressor
import lightgbm as lgb
from lightgbm import LGBMRegressor
import lightgbm as lgb

import warnings
```

```
warnings.filterwarnings("ignore")
```

In [2]:

```
# What EDA & FE techniques use?
is_anomalies = True # or False - Take into account anomalies or no?
```

In [3]:

```
# What type of model to use?
is_Prophet = True   # or False - Facebook Prophet
is_ARIMA = False     # or False - ARIMA and AutoARIMA
is_other_ML = True  # or False - multi-factors models: trees, neural networks, etc.
```

In [4]:

```
# Automatic building ARIMA for Time Series
if is_ARIMA:
    !pip install pmdarima
    import pmdarima as pm
```

In [5]:

```
# Set random state
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)
```

TASK: It is proposed to experiment with forecasting_days

In [6]:

```
# Set the main parameter
forecasting_days = 7  # forecasting_days > 1
```

In [7]:

```
pd.set_option('max_columns',1000)
```

In [8]:

```
# Set indicator names (see dataset https://www.kaggle.com/datasets/vbmokin/wq-southern-bug-river-01052021)
# all_indicator_names as str : 'NH4', 'BSK5', 'NO3', 'NO2', 'SO4', 'PO4', 'CL'
target_indicator_name = 'NO3'
feature_indicator_names = ['NH4', 'NO2', 'CL']  # if it is necessary to forecast the data,
                                                # taking into account the data of other indicators
```

In [9]:

```
# Set id of stations
id_target_station = 14
# all_id_station as int: 1-21 (21 - river outlet, 1 - river mouth,
#                        stations numbering - against the flow of the river)
id_feature_station = [15, 16]  # if it is necessary to forecast the data,
                               # taking into account the data of other stations
```

In [10]:

```
def get_water_data(target_indicator_name : str,
                   id_target_station : int,
                   date_start : str = "2000-01-02",
                   feature_indicator_names : list = [], # list of str
                   id_feature_station : list = [],      # list of int
```

```
                    date_end : str = "2021-06-04"):
    # Get data on the water quality of the Southern Booh (or Bug) River
    # for the given indicators (target and others, if necessary)
    # at the given stations (target and others, if necessary)
    # for the given dates in format "str" inclusive of these dates

    # Indicators names
    all_indicator_names = feature_indicator_names + [target_indicator_name]
    print('Selected indicator names:', all_indicator_names)

    # Information about stations
    pd.set_option('max_colwidth',200)
    all_id_stations = id_feature_station + [id_target_station]
    data_about = pd.read_csv('../input/wq-southern-bug-river-01052021/PB_stations.csv',
sep=';', header=0, encoding='cp1251')
    print('All stations:')
    display(data_about.sort_values(by=['length'], ascending=False))
    print('\nSelected stations:')
    display(data_about[data_about['id'].isin(all_id_stations)])

    # Get data from given id_stations with given indicators and dates
    data = pd.read_csv('../input/wq-southern-bug-river-01052021/PB_All_2000_2021.csv',
sep=';', header=0)
    data['ds'] = pd.to_datetime(data['date'])

    # Data sampling only for selected stations
    df_indicator = data[['id', 'ds'] + all_indicator_names]
    df_indicator = df_indicator[df_indicator['id'].isin(all_id_stations)].dropna().rese
t_index(drop=True)

    # Set new cols
    cols = []
    for station in all_id_stations:
        for feature in all_indicator_names:
            cols.append(str(station) + "_" + feature)

    df = pd.pivot_table(df_indicator, index=["ds"], columns=["id"], values=all_indicato
r_names).dropna()
    df.columns = cols
    df = df.reset_index(drop=False)

    # Set new target name
    new_target_name = str(id_target_station) + "_" + target_indicator_name

    # Get data between given dates
    df = df[(df['ds']>=date_start) & (df['ds']<=date_end)].reset_index(drop=True)

    return df, all_indicator_names, all_id_stations, new_target_name
```

In [11]:

linkcode

```
df, all_indicator_names, all_id_stations, target_name = get_water_data(target_indicator
_name,
                                                                       id_target_st
ation,
                                                                       feature_indi
cator_names = feature_indicator_names,
                                                                       id_feature_s
tation=id_feature_station)
print(f'\nData for processing (target name - "{target_name}"):')
df
```

```
def get_tsfresh_features(data):
    # Get statistic features using library TSFRESH
    # Thanks to https://www.kaggle.com/code/vbmokin/btc-growth-forecasting-with-advance
d-fe-for-ohlc

    # Extract features
    extracted_features = extract_features(data, column_id="ds", column_sort="ds")

    # Drop features with NaN
    extracted_features_clean = extracted_features.dropna(axis=1, how='all').reset_index
(drop=True)

    # Drop features with constants
    cols_std_zero  = []
    for col in extracted_features_clean.columns:
        if extracted_features_clean[col].std()==0:
            cols_std_zero.append(col)
    extracted_features_clean = extracted_features_clean.drop(columns = cols_std_zero)

    extracted_features_clean['ds'] = data['ds']   # For the merging

    return extracted_features_clean
```

In [13]:

```
linkcode
%%time
# FE with TSFRESH
extracted_features_clean = get_tsfresh_features(df[['ds', target_name]])
extracted_features_clean
```

```
# Synthesis dataframe with anomalous dates for Facebook Prophet
if is_anomalies:
    anomalous_dates = ['2014-02-09', '2013-04-04']
    holidays_df = pd.DataFrame(columns = ['ds', 'lower_window', 'upper_window', 'prior_
scale'])
    holidays_df['ds'] = anomalous_dates
    holidays_df['holiday'] = 'anomalous_dates'
    holidays_df['lower_window'] = 0
    holidays_df['upper_window'] = 0
    holidays_df['prior_scale'] = 10
    display(holidays_df)
```

```
def cut_data(df, y, num_start, num_end):
    # Cutting dataframe df and array or list for [num_start, num_end-1]
    df2 = df[num_start:(num_end+1)]
    y2 = y[num_start:(num_end+1)] if y is not None else None
    return df2, y2
```

In [20]:

```
def get_target_mf(df, forecasting_days, col=target_name):
    # Get target as difference of the df[col]
    # Returns target which is shifted for forecasting_days days in the dataframe df
    # "target_name" -> "target"
    df['target'] = df[target_name].shift(-forecasting_days)

    return df
```

In [21]:

```
def get_train_valid_test_ts(df, forecasting_days, target=target_name):
    # Get training, validation and test datasets with target for Time Series models

    # Data prepairing
```

```
    df = df.dropna(how="any").reset_index(drop=True)
    df = df[['ds', target_name]]
    df.columns = ['ds', 'y']
    y = None

#

    N = len(df)
    train, _ = cut_data(df, y, 0, N-2*forecasting_days-1)
    valid, _ = cut_data(df, y, N-2*forecasting_days, N-forecasting_days-1)
    test, _ = cut_data(df, y, N-forecasting_days, N)

    # Train+valid - for optimal model training
    train_valid = pd.concat([train, valid])

    print(f'Origin dataset has {len(df)} rows and {len(df.columns)} features')
    print(f'Get training dataset with {len(train)} rows')
    print(f'Get validation dataset with {len(valid)} rows')
    print(f'Get test dataset with {len(test)} rows')

    return train, valid, test, train_valid
```

In [22]:

```
def get_train_valid_test_mf(df, forecasting_days, target='target'):
    # Get training, validation and test datasets with target for multi-features ML mode
ls

    df = df.drop(columns = ['ds']).dropna(how="any").reset_index(drop=True)

    # Save and drop target
    y = df.pop(target)

    # Get starting points for the recovering target_name from target_name_shigted
    N = len(df)
    #print(f"Total - {N}, Valid start index = {N-forecasting_days-1}, Test start index
= {N-1}")
    start_points = {'valid_start_point' : df.loc[N-forecasting_days-1, target_name],
                    'test_start_point' : df.loc[N-1, target_name]}

    # Standartization data
    scaler = StandardScaler()
    df = pd.DataFrame(scaler.fit_transform(df), columns = df.columns)


    train, ytrain = cut_data(df.copy(), y, 0, N-2*forecasting_days-1)
    valid, yvalid = cut_data(df.copy(), y, N-2*forecasting_days, N-forecasting_days-1)
    test, ytest = cut_data(df.copy(), y, N-forecasting_days, N)


    # Train+valid - for optimal model training
    train_valid = pd.concat([train, valid])
    y_train_valid = pd.concat([ytrain, yvalid])

    print(f'Origin dataset has {len(df)} rows and {len(df.columns)} features')
    print(f'Get training dataset with {len(train)} rows')
    print(f'Get validation dataset with {len(valid)} rows')
    print(f'Get test dataset with {len(test)} rows')

    return train, ytrain, valid, yvalid, test, ytest, train_valid, y_train_valid, start
_points

def calc_metrics(type_score, list_true, list_pred):
    # Calculation score with type=type_score for list_true and list_pred
```

```
    if type_score=='r2_score':
        score = r2_score(list_true, list_pred)
    elif type_score=='rmse':
        score = mean_squared_error(list_true, list_pred, squared=False)
    elif type_score=='mape':
        score = mean_absolute_percentage_error(list_true, list_pred)
    return score
```

In [24]:

```
def result_add_metrics(result, n, y_true, y_pred):
    # Calculation and addition metrics into dataframe result[n,:]

    result.loc[n,'r2_score'] = calc_metrics('r2_score', y_true, y_pred)
    result.loc[n,'rmse'] = calc_metrics('rmse', y_true, y_pred)      # in coins
    result.loc[n,'mape'] = 100*calc_metrics('mape', y_true, y_pred)  # in %

    return result
```

In [25]:

```
# Results of all models
result = pd.DataFrame(columns = ['name_model', 'type_data', 'r2_score', 'rmse', 'mape',
'params', 'ypred'])

# Get datasets
if is_Prophet:
    train_ts, valid_ts, test_ts, train_valid_ts = get_train_valid_test_ts(df.copy(), fo
recasting_days, target=target_name)

    if not is_anomalies:
        holidays_df = None
```

```
Origin dataset has 232 rows and 2 features
Get training dataset with 218 rows
Get validation dataset with 7 rows
Get test dataset with 7 rows
```

In [27]:

```
def prophet_modeling(result,
                     indicator,
                     train,
                     test,
                     holidays_df,
                     period_days,
                     fourier_order_seasonality,
                     forecasting_period,
                     name_model,
                     type_data):
    # Performs FB Prophet model training for given train dataset, holidays_df and seaso
nality_mode
    # Performs forecasting with period by this model, visualization and error estimatio
n
    # df - dataframe with real data in the forecasting_period
    # can be such combinations of parameters: train=train, test=valid or train=train_va
lid, test=test
    # Save results into dataframe result

    # Build Prophet model with parameters and structure
    model = Prophet(daily_seasonality=False,
                    weekly_seasonality=False,
                    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 {indicator
}")

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

    # Ouput the prediction for the next time on forecasted_days
    #forecast[['yhat_lower', 'yhat', 'yhat_upper']] = forecast[['yhat_lower', 'yhat', '
yhat_upper']].round(1)
    #forecast[['ds', 'yhat_lower', 'yhat', 'yhat_upper']].tail(forecasting_period)

    # Forecasting data by the model
    ypred = forecast['yhat'][-forecasting_period:]
    #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
    #result = result_add_metrics(result, n, test['y'], y_pred)

    return result, ypred
```

TASK : It is proposed to experiment with models parameters

```
In [28]:
linkcode
%%time
# Models tuning
if is_Prophet:
    for period_days in [2, 3, 5, 10]:
        for fourier_order_seasonality in [3, 12]:
            result, _ = prophet_modeling(result,
                                         target_indicator_name,
                                         train_ts,
                                         valid_ts,
                                         holidays_df,
                                         period_days,
                                         fourier_order_seasonality,
                                         forecasting_days,
                                         f'{period_days}_days_{fourier_order_seasonalit
y}_order',
                                         'valid')

def 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()
```

In [31]:

```
if is_ARIMA:
    # ACF and PACF
    lag_num = 100
    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 Differ
encing')
    acf_pacf_draw(train_ts['y'].diff().diff().dropna(), lag_num, True, True, '2nd Order
Differencing')
```

In [32]:

```
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
```

In [33]:

```
def get_residual_errors(model):
    # Calculation and drawing the plot residual errors for ARIMA model
    residuals = pd.DataFrame(model.resid)
    fig, ax = plt.subplots(1,2, figsize=(12,6))
    residuals.plot(title="Residuals", ax=ax[0])
    residuals.plot(kind='kde', title='Density', ax=ax[1])
    plt.show()
```

In [34]:

```
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
```

In [35]:

```
%%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 Seasonal
ity)
                               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     # use the stepwise algorithm outlined
in Hyndman and Khandakar (2008)
                                                 # to identify the optimal model parame
ters.
                                                 # The stepwise algorithm can be signif
icantly faster than fitting all
                                                 # hyper-parameter combinations and is
less likely to over-fit the model
                                )

    print(model_auto.summary())
```

```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 5.72 µs
```

```
In [36]:
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]))
```

```
In [37]:
if is_ARIMA:
    # Best model from AutoARIMA
    fig = model_auto.plot_diagnostics(figsize=(12,10))
    plt.show()
```

- The residual errors seem fine with near zero mean and uniform variance.

```
In [38]:
linkcode
if is_ARIMA:
    # Valid forecasting and save result
    result = arima_forecasting(result, model_auto, arima_orders_best, 'ARIMA_auto', val
id_ts, 'valid')
```

```
# Get datasets
if is_other_ML:
    df2 = get_target_mf(df, forecasting_days, col=target_name)
    train_mf, ytrain_mf, valid_mf, yvalid_mf, test_mf, ytest_mf, train_valid_mf, y_trai
n_valid_mf, starting_point = \
                                    get_train_valid_test_mf(df2.copy(), forecasting_day
s, target='target')
```

```
Origin dataset has 225 rows and 39 features
```

```
Get training dataset with 211 rows
Get validation dataset with 7 rows
Get test dataset with 7 rows
if is_other_ML:
    # Set parameters of models
    models = pd.DataFrame(columns = ['name', 'model', 'param_grid'])

    # Linear Regression
    n = len(models)
    models.loc[n, 'name'] = 'Linear Regression'
    models.at[n, 'model'] = LinearRegression()
    models.at[n, 'param_grid'] = {'fit_intercept' : [True, False]}


    # Support Vector Machines
    n = len(models)
    models.loc[n, 'name'] = 'Support Vector Machines'
    models.at[n, 'model'] = SVR()
    models.at[n, 'param_grid'] = {'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
                                  'C': np.linspace(1, 15, 15),
                                  'tol': [1e-3, 1e-4]
                                 }

        # Random Forest Classifier
    n = len(models)
    models.loc[n, 'name'] = 'Random Forest Regressor'
    models.at[n, 'model'] = RandomForestRegressor()
    models.at[n, 'param_grid'] = {'n_estimators': [40, 50, 60, 80],
                                  'min_samples_split': [30, 40, 50, 60],
                                  'min_samples_leaf': [10, 12, 15, 20, 50],
                                  'max_features': ['auto'],
                                  'max_depth': [3, 4, 5, 6]
                                 }

    # Bagging Classifier
    n = len(models)
    models.loc[n, 'name'] = 'Bagging Regressor'
    models.at[n, 'model'] = BaggingRegressor()
    models.at[n, 'param_grid'] = {'max_features': np.linspace(0.05, 0.8, 1),
                                  'n_estimators': [3, 4, 5, 6],
                                  'warm_start' : [False]
                                 }

    # XGB Classifier
    n = len(models)
    models.loc[n, 'name'] = 'XGB Regressor'
    models.at[n, 'model'] = xgb.XGBRegressor()
    models.at[n, 'param_grid'] = {'n_estimators': [50, 70, 90],
                                  'learning_rate': [0.01, 0.05, 0.1, 0.2],
                                  'max_depth': [3, 4, 5]
                                 }


models
def model_prediction(result, models, train_features, valid_features, train_labels, vali
d_labels):
    # Models training and data prediction for all models from DataFrame models
    # Saving results for validation dataset into dataframe result

    def calc_add_score(res, n, type_score, list_true, list_pred, feature_end):
        # Calculation score with type=type_score for list_true and list_pred
        # Adding score into res.loc[n,...]
```

```
        res.loc[i, type_score + feature_end] = calc_metrics(type_score, list_true, list
_pred)
        return res

    # Results
    model_all = []

    for i in range(len(models)):
        # Training
        print(f"Tuning model '{models.loc[i, 'name']}'")
        model = GridSearchCV(models.at[i, 'model'], models.at[i, 'param_grid'])
        model.fit(train_features, train_labels)
        model_all.append(model)
        print(f"Best parameters: {model.best_params_}\n")

        # Prediction
        ypred = model.predict(valid_features)

        # Scoring and saving results into the main dataframe result
        n = len(result)
        result.loc[n,'name_model'] = f"{models.loc[i, 'name']}"
        result.loc[n,'type_data'] = "valid"
        result.at[n,'params'] = model.best_params_
        result.at[n,'ypred'] = ypred
        #result = result_add_metrics(result, n, valid_labels, valid_pred)

    return result, model_all
```

In [43]:

```
%%time
if is_other_ML:
    # Models tuning and the forecasting
    result, model_all = model_prediction(result, models, train_mf, valid_mf, ytrain_mf,
yvalid_mf)
```

```
def recovery_prediction(y, starting_point):
    # Recovering prediction of multi-factors model for shifted col to col in the datafr
ame df
    # y has type np.array
    # starting_point is dictionary with start values for the recovering data
    # Returns y (np.array) with recovering data

    return np.insert(y, 0, starting_point).cumsum()[1:]
```

In [45]:

```
def result_recover_and_metrics(result, df_ts, type_data, start_points):
    # Recovering prediction: from shifted_Close 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=='va
lid' 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
```

In [46]:

```
linkcode
# Dispay and save all results for validation dataset
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', '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_day
s}.csv')
else:
    print('There are no tuned models!')
```

```
def get_model_opt(name_model, params):
    # Model tuning for the name_model

    print(name_model)
    if name_model=='Linear Regression':
        model = LinearRegression(**params)


    elif name_model=='Support Vector Machines':
        model = SVR(**params)


    elif name_model=='Random Forest Regressor':
        model = RandomForestRegressor(**params)

    elif name_model=='Bagging Regressor':
        model = BaggingRegressor(**params)


    elif name_model=='XGB Regressor':
        model = xgb.XGBRegressor(**params)

    else: model = None

    return model
```

In [48]:

```
def get_params_optimal_model(result, main_metrics):
    # Get parameters of the optimal model from dataframe result by main_metrics

    # Set the data type to float (just in case)
    result[main_metrics] = result[main_metrics].astype('float')
```

```
    # Choose the optimal model
    opt_result = result[result['type_data']=='valid'].reset_index(drop=True)
    if main_metrics=='r2_score':
        opt_model = opt_result.nlargest(1, main_metrics)
    else:
        # 'mape' or 'rmse'
        opt_model = opt_result.nsmallest(1, main_metrics)
    display(opt_model[['name_model', 'r2_score', 'rmse', 'mape', 'params']])

    # Get parameters of the optimal model
    opt_name_model = opt_model['name_model'].tolist()[0]
    opt_type_model = opt_model['type_model'].tolist()[0]
    opt_params_model = opt_model['params'].tolist()[0]
    print(f'Optimal model by metrics "{main_metrics}" is "{opt_name_model}" with type "
{opt_type_model}" parameters {opt_params_model}')

    return opt_name_model, opt_type_model, opt_params_model
```

In [49]:

```
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,
                                    target_indicator_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))

    else:
```

```
        # Other ML model
        # Training ML optimal model for training+valid dataset
        print(f"Tuning model '{name_model}'")
        models_opt_number = models[models['name']==name_model].index.tolist()[0]
        #print(f"Model - {models.at[models_opt_number,'model']} with parameters {params
}")
        if type_test=='1':
            model_opt = GridSearchCV(models.at[models_opt_number,'model'], models.at[mo
dels_opt_number,'param_grid'])
        else:
            # type_test=='2'
            model_opt = get_model_opt(models.at[models_opt_number,'name'], params)
        model_opt.fit(df, y)

        # Forecasting
        ypred = model_opt.predict(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
```

In [50]:

```
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(res
ult,
                                                                                    mai
n_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_tr
ain_valid,
                                                              test, ytest,
                                                              opt_name_model, opt_type_
model,
```

```
                                                        opt_params_model, '1')

        # Calculation metrics for recovering prediction ypred for test dataset by the o
ptimal 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 target_name
            ytest_plot = recovery_prediction(ytest.values, start_points['test_start_poi
nt'])
            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
```

In [51]:

```
# 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)
```

```
# Training ML optimal model for training+valid dataset
# Get parameters of the optimal model from dataframe result (without Time Series models
) by main_metrics
if is_other_ML:
    main_metrics = 'r2_score'
    if (len(result) > 0) and (len(models) > 0):
        result_nonTS = result[(result['type_model']!='Prophet') & (result['type_model']
!='ARIMA')].reset_index(drop=True)
        opt_name_model2, opt_type_model2, opt_params_model2 = get_params_optimal_model(
result_nonTS,
                                                                main_me
trics)

        result, model_opt, ypred = model_training_forecasting(result,
                                                              train_valid_mf,
                                                              y_train_valid_mf,
                                                              test_mf,
                                                              ytest_mf,
                                                              opt_name_model2,
                                                              opt_type_model2,
                                                              opt_params_model2,
                                                              '2')

# All features names
if is_other_ML:
    coeff = pd.DataFrame(train_valid_mf.columns)
    coeff.columns = ['feature']
```

```
In [54]:
def add_fi_coeff(coeff, col, list_new_fi_coeff=None, df_new_fi_coeff=None):
    # Adds new importance of features as feature col
    # from list list_new_fi_coeff or dataframe df_new_fi_coeff
    # to the resulting dataframe coeff with feature names
    # Missed importance values are replaced by zero

    if list_new_fi_coeff is not None:
        df_new_fi_coeff = coeff[['feature']].copy()
        df_new_fi_coeff["score"] = pd.Series(list_new_fi_coeff)

    if df_new_fi_coeff is not None:
        # Rename df_new_fi_coeff
        df_new_fi_coeff.columns = ['feature', 'score']   # to the plot drawing
        df_new_fi_coeff[col] = df_new_fi_coeff['score']  # to the merging and saving

        # Merging dataframes - coeff of all features with new_fi_coeff
        coeff = coeff.merge(df_new_fi_coeff[['feature', col]], on='feature', how='left'
).fillna(0)

        is_score = True
    else:
        print(f'Data is absent fol {col}')
        is_score = False
        coeff = None

    return coeff, df_new_fi_coeff, is_score
```

```
In [55]:
# Feature importance diagram with SHAP
if is_other_ML:
    if (len(result) > 0) and (len(models) > 0):
        print('Feature importance diagram with SHAP:')
        try:
            # Trees
            explainer = shap.TreeExplainer(model_opt)
            shap_values = explainer.shap_values(test_mf)
            shap.summary_plot(shap_values, test_mf, plot_type="bar", feature_names=coef
f['feature'].tolist())
            shap.summary_plot(shap_values, test_mf)

            # Save permutation feature importance values
            coeff, _, is_SHAP_successfully = add_fi_coeff(coeff, 'shap_fi_score', shap_
values)
        except:
            try:
                # Other types of models
                explainer = shap.KernelExplainer(model_opt.predict, train_valid_mf)
                shap_values = explainer.shap_values(test_mf)

                # Plot drawing
                shap.summary_plot(shap_values, test_mf, plot_type="bar", feature_names=
coeff['feature'].tolist())
                shap.summary_plot(shap_values, test_mf)

                # Get feature importance values from shap_values format
                # Thanks to https://stackoverflow.com/a/69523421/12301574
                shap_values_all = pd.DataFrame(shap_values, columns = test_mf.columns)
                vals = np.abs(shap_values_all.values).mean(0)
                shap_importance = pd.DataFrame(list(zip(test_mf.columns, vals)),
                                               columns=['feature','score'])
```

```
                # Saving feature importance values
                coeff, _, is_SHAP_successfully = add_fi_coeff(coeff, 'shap_fi_score', N
one, shap_importance)

            except:
                is_SHAP_successfully = False

        if not is_SHAP_successfully:
            print('Feature importance diagram for this optimal model is not supported i
n SHAP')
# Force plot - Feature importance diagram with SHAP for the certaion row in test_mf
if is_other_ML:
    if (len(result) > 0) and (len(models) > 0):
        row_number_in_test_mf = 0
        print('Feature importance diagram as the Force plot with SHAP:')
        if is_SHAP_successfully:
            shap.initjs()
            shap.force_plot(explainer.expected_value, shap_values[0,:],
                            test_mf.loc[test_mf.index.tolist()[row_number_in_test_mf],:
],
                            feature_names=coeff['feature'].tolist(),
                            matplotlib=True, show=False)
            plt.savefig('force_plot.png')
Feature importance diagram as the Force plot with SHAP:
Creation and drawing the feature importance diagrams
if is_other_ML:
    if (len(result) > 0) and (len(models) > 0):

        # Coefficients
        if opt_name_model2=='XGB Regressor':
            print('Feature importance diagram')
            # Coef. of the feature with nonzero importance
            xgb_coeff = pd.DataFrame.from_dict(model_opt.get_booster().get_score(import
ance_type='weight'), orient='index').reset_index(drop=False)
            coeff, _, is_score = add_fi_coeff(coeff, 'xgb_fi_coeff', None, xgb_coeff)

            # With the library xgboost
            fig =  plt.figure(figsize = (15,15))
            axes = fig.add_subplot(111)
            xgb.plot_importance(model_opt,ax = axes,height = 0.5)
            plt.show()
            plt.close()

        else:
            # With the library sklearn
            try:
                coef_model = model_opt.coef_
                coeff, coeff_new, is_score = add_fi_coeff(coeff, 'lr_fi_score', coef_mo
del)
            except:
                try:
                    coef_model = feature_importances_
                    coeff, coeff_new, is_score = add_fi_coeff(coeff, 'model_fi_score',
coef_model)
                except:
                    print('The importance of the feature could not be obtained')
                    is_score = False

            if is_score:
                # Plot drawing
                coeff_non_zero = coeff_new[coeff_new['score']>0]
                plt.figure(figsize=(12, int(len(coeff_non_zero)*0.4)))
```

```
                coeff_non_zero = coeff_non_zero.sort_values(by='score', ascending=True)
                plt.barh(coeff_non_zero["feature"], coeff_non_zero["score"])
                plt.title("Feature importance diagram")
                plt.axvline(x=0, color=".5")
                plt.xlabel("Coefficient values")
                plt.subplots_adjust(left=0.3)
```

```
The importance of the feature could not be obtained
```

In [58]:

```
# Permutation feature importance diagram
if is_other_ML:
    if (len(result) > 0) and (len(models) > 0):
        try:
            perm_importance = permutation_importance(model_opt, test_mf, ytest_mf)

            # Save permutation feature importance values
            coef_model = perm_importance.importances_mean
            coeff, coeff_new, is_score = add_fi_coeff(coeff, 'perm_fi_score', coef_mode
l)

            print('Permutation feature importance diagram:')
            coeff_non_zero = coeff_new[coeff_new['score'].abs()>1e-4]
            coeff_non_zero = coeff_non_zero.sort_values(by='score', ascending=True)
            plt.figure(figsize=(12, int(len(coeff_non_zero)*0.4)))
            plt.barh(coeff_non_zero["feature"], coeff_non_zero["score"])
            plt.xlabel("Permutation Importance")
            plt.show()
            is_perm_importance = True
        except: print('Permutation feature importance diagram for this optimal model is
not supported')
```

```
# Feature importance diagram with ELI5
if is_other_ML:
    if (len(result) > 0) and (len(models) > 0):
        try:
            print('Feature importance diagram with ELI5:')
            perm = PermutationImportance(model_opt).fit(test_mf,ytest_mf)

            # Save permutation feature importance values
            coef_model = perm.feature_importances_  # Feature importances,
                                                    # computed as mean decrease
                                                    # of the score when a feature
                                                    # is permuted (i.e. becomes noise)
            coeff, _, is_score = add_fi_coeff(coeff, 'eli5_perm_fi_score', coef_model)

            # Display permutation feature importance values with ELI5
            display(eli5.show_weights(perm, feature_names = coeff.feature.tolist()))

        except: print('Feature importance diagram for this optimal model is not support
ed in ELI5')
Feature importance diagram with ELI5:
# Display and saving features importance values
if is_other_ML:
    if coeff.isna().sum().sum()==0:
        print('Feature importance values:')
        fi_cols = coeff.columns.tolist()[1:]
        if len(fi_cols) > 0:
            coeff = coeff.sort_values(by=fi_cols, ascending=False)
            display(coeff)
        coeff.to_csv(f'feature_importance_for_optimal_model_{opt_name_model2}.csv', ind
ex=False)
```

Додаток Г

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПРОГНОЗУВАННЯ КОНЦЕНТРАЦІЇ НІТРАТІВ У РІЧКОВІЙ ВОДІ ПІВДЕННОГО БУГУ

Виконав: студент гр. 2ІСТ-21м

_____ Лісовський Р.Р.

«_01_» _____12____ 2022 р.

Керівник: к.т.н., доц. каф. САІТ

_____ Жуков С.О.

«_02_» _____12_____ 2022 р.

Нормоконтроль: к.т.н., доцент

_____ Жуков С. О.

«_02_» _____12_____ 2022 р.

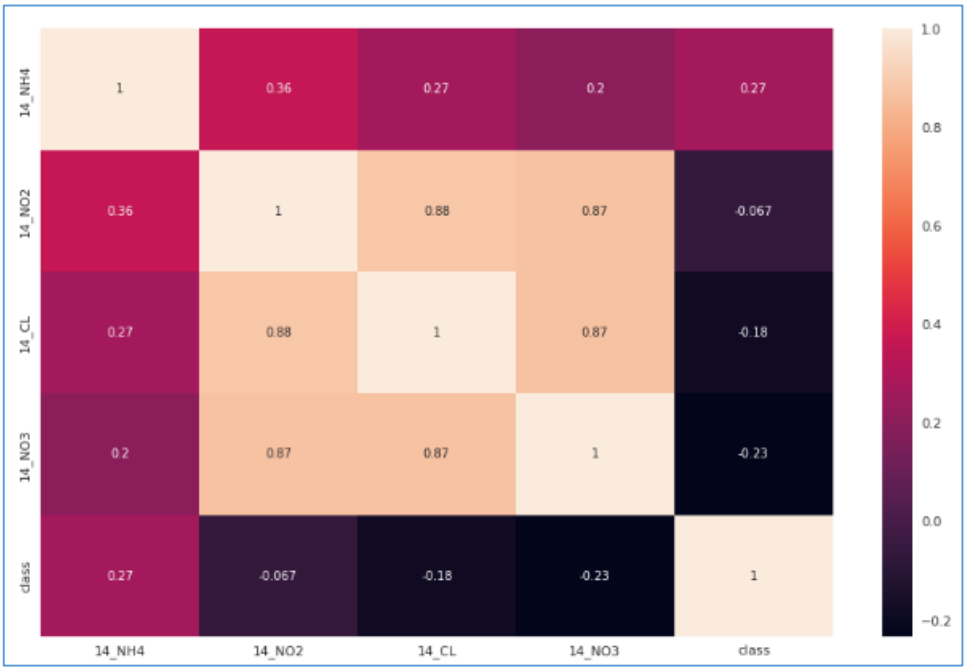


Рисунок Г.1 – Матриця коефіцієнта кореляції

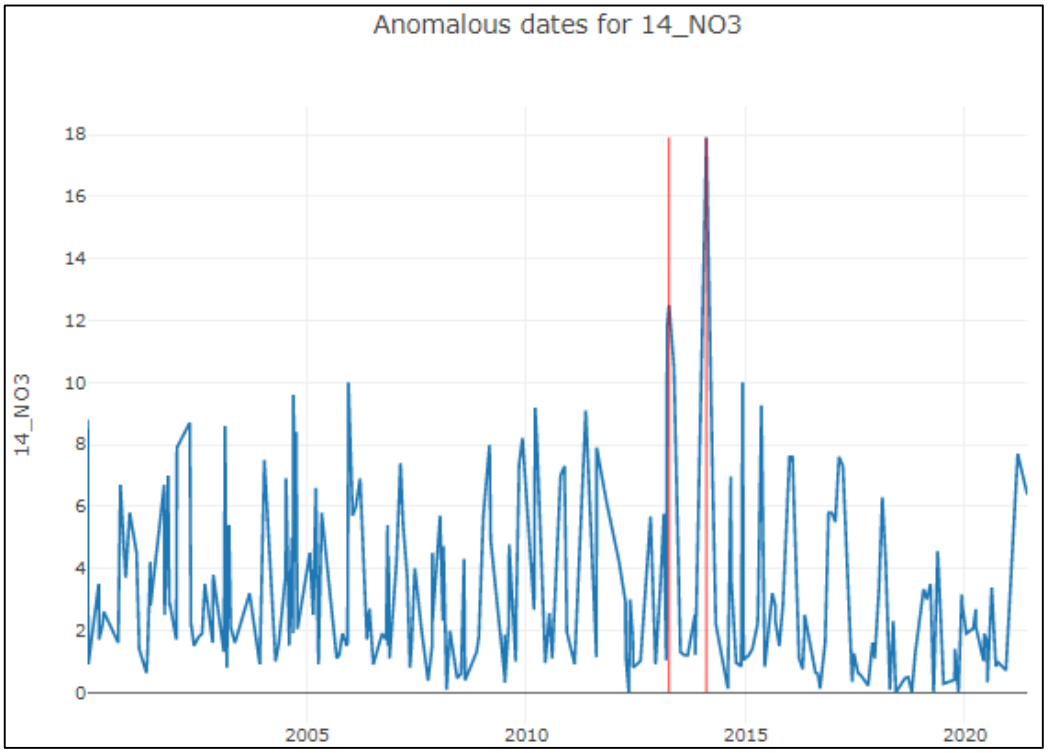


Рисунок Г.2 – Графік аномальних даних

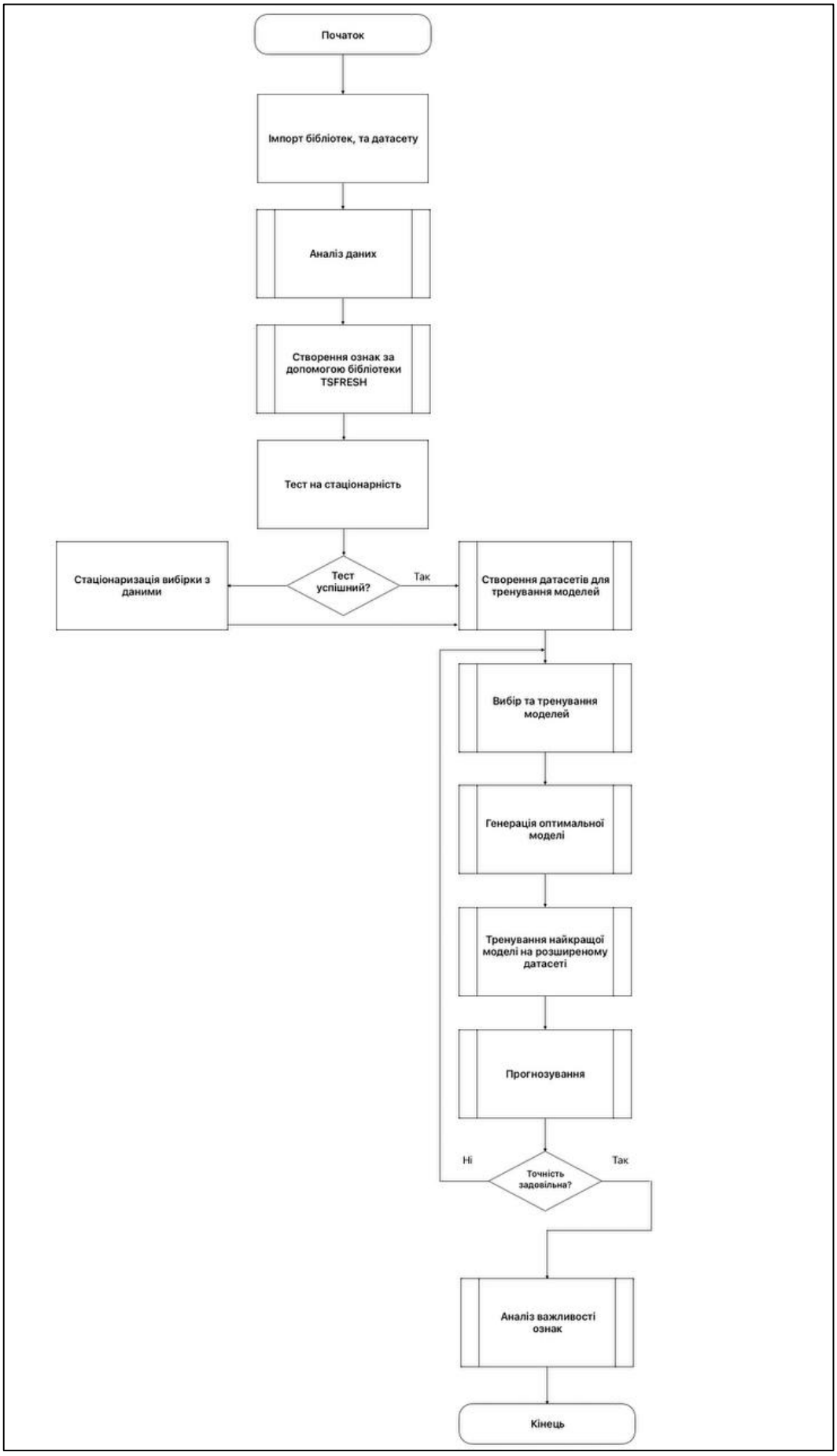


Рисунок Г.3 – Блок-схема алгоритму роботи інформаційної технології

	name_model	type_data	r2_score	rmse	mape
6	Prophet_10_days_3_order	valid	0.092209	0.449807	20.182881
7	Prophet_10_days_12_order	valid	-0.033513	0.479945	22.262139
4	Prophet_5_days_3_order	valid	-0.138636	0.503762	23.691246
2	Prophet_3_days_3_order	valid	-0.343909	0.547291	28.841
5	Prophet_5_days_12_order	valid	-0.457495	0.56995	28.945301
0	Prophet_2_days_3_order	valid	-0.470179	0.572425	32.297175
3	Prophet_3_days_12_order	valid	-1.228396	0.704741	40.400179
1	Prophet_2_days_12_order	valid	-2.188942	0.843056	49.590321
9	Support Vector Machines	valid	-545.849335	11.039947	609.034317
12	XGB Regressor	valid	-909.302139	14.243803	789.981221
10	Random Forest Regressor	valid	-949.259586	14.553059	801.972235
11	Bagging Regressor	valid	-986.785947	14.837632	819.560146
8	Linear Regression	valid	-1156.331621	16.060611	874.565124

Number of models built - 13

Рисунок Г.4 – Похибки моделей за різними метриками

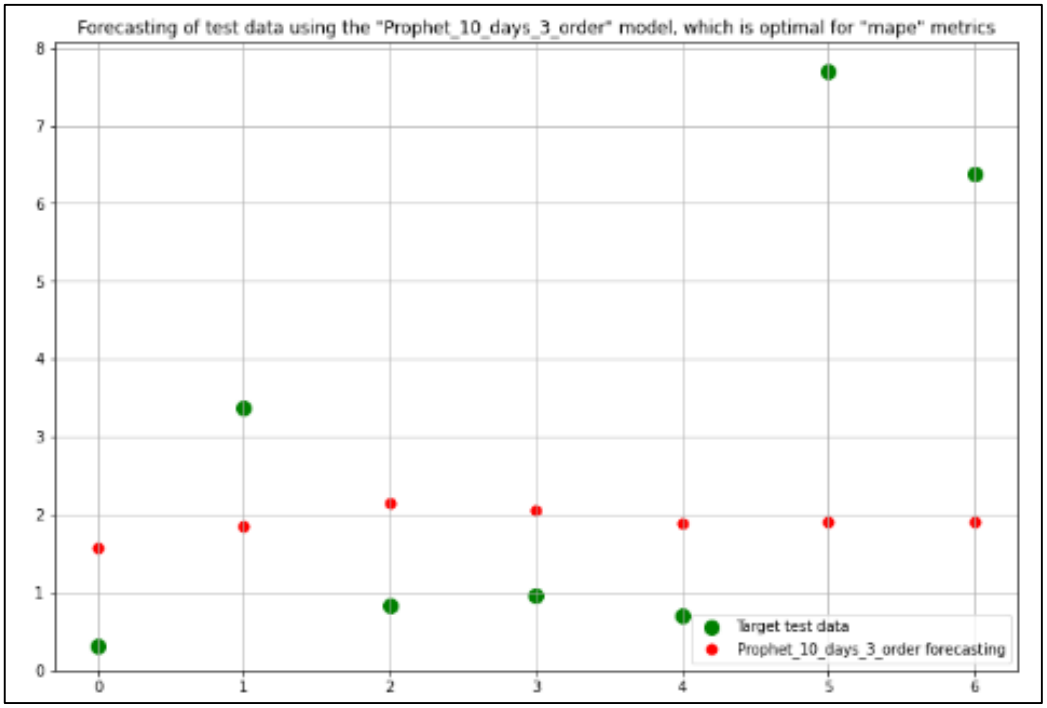


Рисунок Г.5 - Графік прогнозування за допомогою оптимальної моделі Facebook Prophet

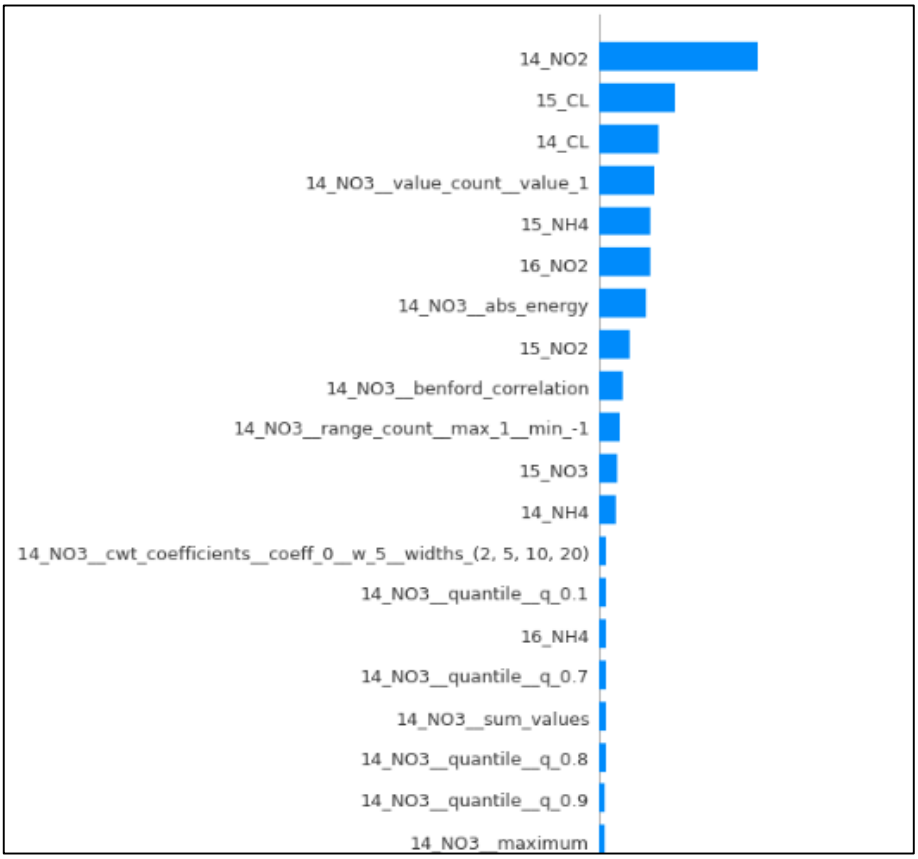


Рисунок Г.6 – Діаграма важливості ознак