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

Бакалаврська кваліфікаційна робота на тему:

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ВОДОГОСПОДАРСЬКИХ БАЛАНСІВ ДІЛЯНОК БАСЕЙНУ ПІВДЕННОГО БУГУ»

Виконав: студент 4 курсу, групи 2ІСТ-21б спеціальності 126 – «Інформаційні системи та технології»
_____Роман ВОВК
Керівник: к.т.н., ас. каф. САІТ
_____ Ігор ШТЕЛЬМАХ
«03» 06 2025 р.
Рецензент: к.т.н., ас. каф. КН
_____Ілона БОГАЧ
«07» 06 2025 р.

Допущено до захисту

Завідувач кафедри САІТ
_____д.т.н., проф. Віталій МОКІН
«06» 06 2025 р.

Вінниця ВНТУ – 2025 рік

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

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
д.т.н., проф. Віталій МОКІН
"28" 03 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Вовку Роману Сергійовичу

1. Тема роботи: «Інформаційна технологія аналізу водогосподарських балансів ділянок басейну Південного Бугу»
керівник роботи: Штельмах І.М., ас. каф. САІТ
затверджені наказом ВНТУ від «20» 03 2025 року № 97
2. Термін подання студентом роботи 31.05.25
3. Зміст текстової частини:
1) Загальна характеристика обєкту досліджень
2) Вибір оптимальної ІТ-інфраструктури.
3) Підготовка та розвідувальний аналіз даних.
4) Розроблення інформаційної технології та прогнозування.
4. Перелік ілюстративного матеріалу:
1) Вигляд створених датасетів;
2) Карти дефіцитів та резервів;
3) Блок-схема алгоритму інформаційної технології аналізу водогосподарських балансів ділянок басейну Південного Бугу;
4) Результати прогнозування витрат води по гідропостам с.Лелітка та с.Тростянчик.

6. Консультанти розділів роботи:

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

7. Дата видачі завдання 28.03.25

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Загальна характеристика обєкту досліджень	01.04	10.04	вик
2	Вибір оптимальної IT-інфраструктури	10.04	20.04	вик
3	Підготовка та розвідувальний аналіз даних	20.04	30.04	вик
4	Розроблення інформаційної технології та прогнозування	1.05	15.05	вик
5	Оформлення матеріалів до захисту БКР	15.05	30.05	вик

Студент Роман ВОВК

Керівник роботи Ігор ШТЕЛЬМАХ

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 92 сторінок формату А4, на яких є 67 рисунків, список використаних джерел містить 29 найменувань.

Метою роботи є розроблення інформаційної технології для аналізу водогосподарських балансів ділянок басейну Південного Бугу з метою виявлення проблемних ділянок, що мають дефіцит води, та прогнозування їхніх витрат для підтримки управлінських рішень.

У межах роботи створено інформаційну технологію з використанням мови програмування Python та бібліотек Pandas, NumPy, GeoPandas, Matplotlib, Seaborn, Prophet, SARIMA, LSTM. Реалізовано алгоритм для обробки даних, розвідувальний аналіз із статистичним описом і візуалізаціями (графіки, гістограми, теплові карти), просторовий аналіз із картами дефіциту (Wдеф) і резервів (Wрез) води, а також прогнозування витрат води для ділянок с. Лелітка та с. Тростянчик. Розроблена технологія забезпечує інтеграцію з геопросторовими даними та адаптована для аналізу часових рядів водного балансу.

Ключові слова: інформаційна технологія, водогосподарський баланс, Python, розвідувальний аналіз, просторовий аналіз, прогнозування, басейн Південного Бугу.

ABSTRACT

The bachelor's qualification work consists of 92 A4 pages, including 67 figures, and the list of references contains 29 items.

The purpose of the diploma work is to develop information technology for analyzing water management balances of sections of the Southern Bug Basin, aimed at identifying problem areas with water deficits and forecasting their water consumption to support management decisions.

Within the work, information technology was developed using the Python programming language and libraries such as Pandas, NumPy, GeoPandas, Matplotlib, Seaborn, Prophet, SARIMA, LSTM. An algorithm was implemented for data processing, exploratory analysis with statistical descriptions and visualizations (graphs, histograms, heatmaps), spatial analysis with maps of water deficits (Wдеф) and reserves (Wрез), and forecasting water consumption for the sections of Lelytka and Trostyanets. The developed technology ensures integration with geospatial data and is adapted for analyzing time series of water balance.

Keywords: information technology, water management balance, Python, exploratory analysis, spatial analysis, forecasting, Southern Bug Basin.

ЗМІСТ

ВСТУП

Актуальність теми. Раціональне управління водними ресурсами є однією з ключових задач екологічної безпеки, соціального добробуту та сталого розвитку регіонів. Особливої актуальності ця проблема набуває в умовах зміни клімату, зростання водоспоживання та локального дефіциту води в окремих водогосподарських ділянках. Басейн річки Південний Буг є важливим водним артерійним регіоном України, що забезпечує потреби населення, сільського господарства та промисловості. За даними Басейнового управління водних ресурсів (БУВР) Південного Бугу, у 2024 році зафіксовано значні коливання водозабезпеченості, що впливають на водний баланс. У зв'язку з цим постає потреба у створенні сучасних інформаційних технологій, які дозволяють аналізувати водогосподарський баланс, виявляти критичні ділянки та прогнозувати ризики нестачі води, що є актуальним для сталого розвитку регіону.

Мета і задачі дослідження. Метою бакалаврської кваліфікаційної роботи є розроблення інформаційної технології для аналізу водогосподарських балансів ділянок басейну Південного Бугу з метою виявлення проблемних ділянок, що мають дефіцит води, та прогнозування витрат води для підтримки управлінських рішень.

Для досягнення поставленої мети необхідно розв'язати такі задачі:

- Сформувати датасет шляхом збору та обробки даних про водогосподарські баланси басейну Південного Бугу, отриманих із PDF-файлів, із конвертацією у структурований CSV-формат.
- Розробити інформаційну технологію для аналізу водогосподарських балансів ділянок басейну Південного Бугу.
- Провести розвідувальний і просторовий аналіз даних із використанням відповідних бібліотек Python для виявлення проблемних ділянок.
- Здійснити прогнозування витрат води для виявлених проблемних ділянок за допомогою моделей машинного навчання.

Об'єктом дослідження є процес створення інформаційної технології для

аналізу водогосподарських балансів ділянок басейну Південного Бугу.

Предметом дослідження є методи та програмні засоби створення інформаційної технології для аналізу водогосподарських балансів ділянок басейну Південного Бугу, включаючи розвідувальний і просторовий аналіз, а також прогнозування витрат води.

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ОБ'ЄКТУ ДОСЛІДЖЕНЬ

1.1 Опис об'єкта досліджень

Об'єктом дослідження є інформаційна технологія аналізу водогосподарських балансів річкового басейну Південного Бугу з подальшим прогнозуванням критичних ситуацій на окремих ділянках. Ця технологія спрямована на створення комплексного аналітичного інструменту, який інтегрує обробку великих обсягів просторових і часових даних, візуалізацію результатів та моделювання для підтримки прийняття обґрунтованих управлінських рішень у сфері водного господарства. Предметом дослідження є процес виявлення дефіцитів і резервів води, побудова інтерактивних візуалізацій для оцінки водного балансу, а також підготовка аналітичної бази для реалізації прогнозних моделей на основі сучасних методів обробки даних. Основна мета дослідження полягає у розробці такого аналітичного інструменту, який дозволить органам водного господарства ефективно оцінювати поточний стан водних ресурсів, прогнозувати ймовірні дефіцити та оптимізувати їхнє використання на основі фактичних даних і аналітичних моделей[1].

Наступний рисунок 1.1 ілюструє Райони басейнів річок України.

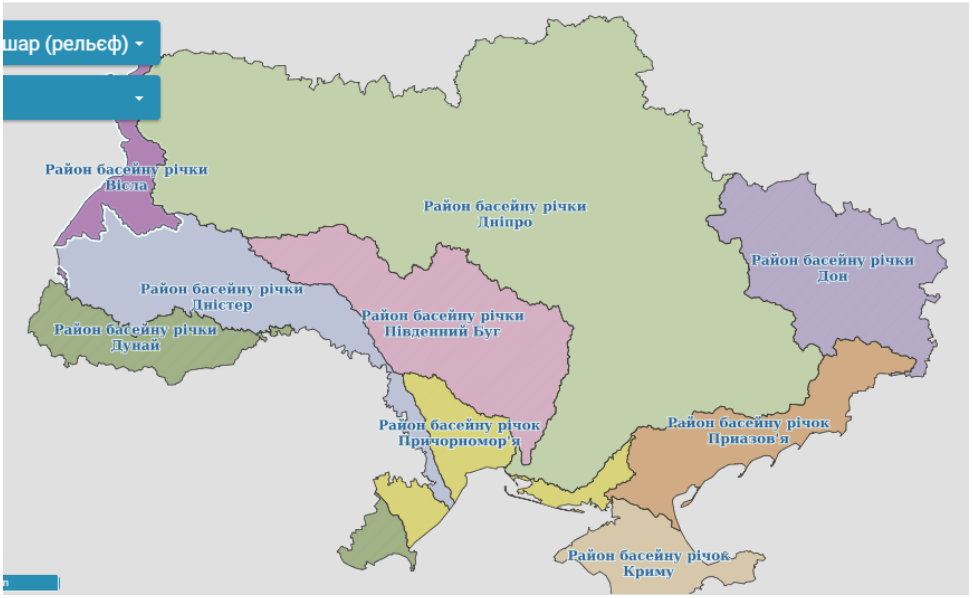

Рисунок 1.1 - Райони басейнів річок України, включаючи басейн річки Південний Буг

Водогосподарські баланси відіграють ключову роль у менеджменті водних ресурсів, забезпечуючи оцінку співвідношення між доступними водними обсягами (природними припливами, штучними джерелами) та потребами у воді для різних секторів: питного водопостачання, зрошення земель, промислового виробництва, енергетики та екологічних потреб. У сучасних умовах зростання впливу кліматичних змін, таких як посухи та нестабільні опади, а також антропогенного навантаження, актуальність аналізу водних балансів значно зростає. Особливо це стосується регіонів України, де водозабезпечення зазнає значних коливань, зокрема у південних областях, де дефіцит води може мати серйозні соціально-економічні наслідки.

Вибір басейну річки Південний Буг як об’єкта дослідження є виправданим з огляду на його стратегічне значення. Ця річка, що протікає через території Вінницької, Одеської, Миколаївської та Кіровоградської областей, забезпечує водними ресурсами численні населені пункти, промислові підприємства та сільськогосподарські угіддя. Однак через складні гідрологічні умови, включаючи сезонні коливання рівня води, вплив гідротехнічних споруд та інтенсивне використання ресурсів, у басейні виникають ситуації локального дефіциту води. Водночас у деяких ділянках спостерігаються надлишкові резерви, які не завжди використовуються раціонально, що підкреслює потребу в детальному аналізі та оптимізації водокористування.

У контексті сучасних викликів зростає необхідність у впровадженні автоматизованих інформаційних технологій для моніторингу, аналізу та управління водними ресурсами. Такі технології повинні вміти обробляти великі масиви просторово-часових даних, будувати географічні карти дефіциту та резервів, а також здійснювати прогнозування критичних ситуацій. У межах цього дослідження реалізовано повний цикл аналітичного процесу: від створення власного датасету шляхом конвертації первинних даних із PDF-файлів у структурований CSV-формат до проведення розвідувального аналізу та просторової візуалізації. Наступний рисунок 1.2 ілюструє структуру реалізації інформаційної технології аналізу [2].

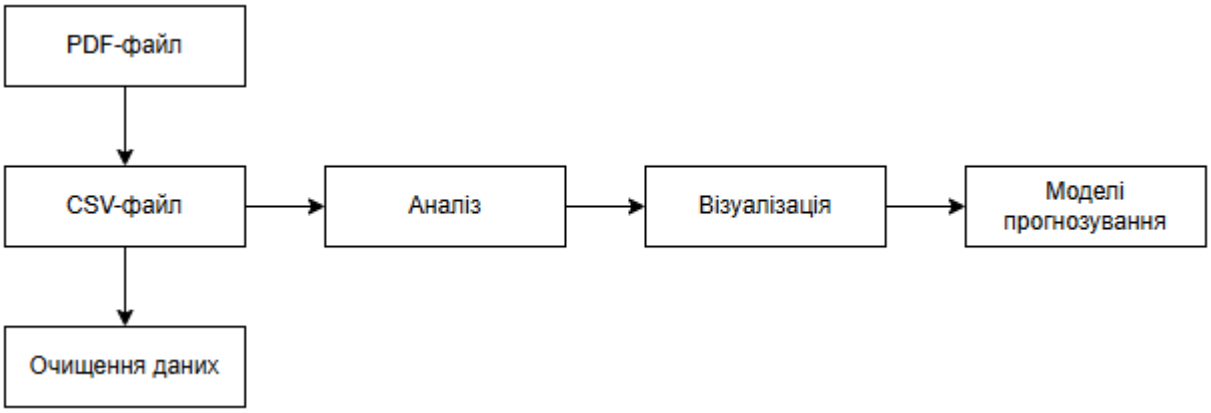

Рисунок 1.2 – Структура реалізації інформаційної технології аналізу водного балансу

Робота виконана в середовищі Kaggle з використанням бібліотек Python, таких як Pandas для обробки даних, GeoPandas для геопросторового аналізу, Matplotlib і Seaborn для побудови інтерактивних графікових зображень (теплові карти, лінійні графіки, карти дефіциту), а також закладено основу для подальшого впровадження методів машинного навчання, зокрема для прогнозування водних витрат [3].

Інформаційна технологія аналізу водного балансу, розроблена в рамках роботи, має не лише практичну цінність, дозволяючи органам водного господарства оперативно реагувати на зміни, але й значний науковий потенціал. Зокрема, вона може бути розширена до системи раннього попередження про водний дефіцит, інтегруючи прогнози на основі історичних даних та кліматичних моделей. Такий підхід є вкрай актуальним для національного водного менеджменту, сприяючи сталому розвитку регіонів і адаптації до кліматичних викликів у довгостроковій перспективі [28].

1.2 Огляд подібних досліджень

Питання аналізу водогосподарських балансів річкових басейнів, зокрема басейну Південного Бугу, є важливим для ефективного управління водними ресурсами в умовах змін клімату та антропогенного впливу. У цьому підрозділі розглянуто основні дослідження, які стосуються застосування інформаційних

технологій, геоінформаційних систем (ГІС) та методів аналізу водних ресурсів у басейні Південного Бугу. Наведено порівняння з розробленою у цій роботі інформаційною технологією, що вирізняється прогнозуванням витрат води за допомогою методів машинного навчання [4].

Одне з ключових досліджень у цій сфері було проведено у Вінницькому національному технічному університеті (ВНТУ) у 2004–2005 роках на замовлення Держводгоспу України. У рамках цієї роботи, під керівництвом Мокіна В.Б. та Крижановського Є.М., створено систему підтримки прийняття управлінських рішень для басейну річки Південний Буг із використанням геоінформаційних технологій. ГІС-система мала масштаб 1:200 000 і включала дані державного моніторингу поверхневих вод, паспортні характеристики річок, водосховищ і ставків, а також інформацію про водокористування у різних водогосподарських ділянках. Основною метою було створення єдиної інформаційної бази, яка б дозволила імпортувати дані з відомчих систем моніторингу та водного кадастру для подальшого аналізу. Для реалізації використано ГІС-програмне забезпечення, що забезпечило просторову візуалізацію даних і можливість оцінки водогосподарського стану. Однак, на відміну від розробленої у цій роботі інформаційної технології, у згаданому дослідженні не застосовувалися методи машинного навчання для прогнозування витрат води. Натомість акцент був зроблений на статичному аналізі та картографуванні, тоді як у нашій роботі використано моделі Prophet, SARIMA, LSTM для прогнозування, що дозволяє враховувати динамічні зміни у водному балансі.

Ще одне дослідження, проведене у ВНТУ у 2013 році під керівництвом Мокіна В.Б., стосувалося моделювання водогосподарського балансу басейну Південного Бугу. У рамках цієї роботи було систематизовано дані по водогосподарських ділянках, зокрема інформацію про водозабір, поверхневі та підземні стоки, а також використано для розрахунків балансів у районах із різною водозабезпеченістю. Результатом стала автоматизована система для обчислення водогосподарських балансів, яка допомагала оптимізувати управління водними ресурсами. Для реалізації використано програмне

забезпечення, що дозволяло автоматизувати розрахунки, а також бази даних для зберігання інформації про водокористування. Хоча ця система була корисною для статичних розрахунків, вона не включала прогнозування на основі часових рядів, як у нашій роботі. У нашому дослідженні, крім статичного аналізу, виконано прогнозування витрат води для ділянок с. Лелітка та с. Тростянчик, що дозволяє передбачати дефіцит і резерви води у майбутньому.

У 2021 році було опубліковано дослідження, присвячене оцінці екологічного стану річки Південний Буг у рамках вимог Водної Рамкової Директиви ЄС. У цій роботі проводився аналіз гідрохімічних і біологічних показників для різних ділянок річки від витоку до гирла. Зокрема, визначалися референційні значення загального азоту та фосфору, а також оцінювався вплив забруднень на екологічний стан. Дослідження включало розвідувальний аналіз даних, подібний до того, що використано у нашій роботі, для оцінки якості даних і виявлення закономірностей. Наприклад, було використано статистичні методи для аналізу розподілу показників якості води та виявлення аномалій у даних. Проте це дослідження не передбачало прогнозування витрат води, що є ключовою відмінністю від нашого підходу. У нашій роботі розвідувальний аналіз доповнено прогнозуванням за допомогою методів машинного навчання, що забезпечує не лише оцінку поточного стану, а й передбачення майбутнього водного балансу.

Басейнове управління водних ресурсів (БУВР) Південного Бугу також проводить регулярний моніторинг стану поверхневих вод. У 2024 році було затверджено 50 пунктів моніторингу в басейні, де збираються дані про якість води, водокористування та гідрологічну обстановку. Ці дані використовуються для підготовки аналітичних довідок і розробки режимів роботи водосховищ. Наприклад, БУВР аналізує вплив сезонних коливань на водогосподарський баланс і формує рекомендації для водокористувачів. Хоча ця діяльність не передбачає розробки інформаційних технологій для прогнозування, вона є важливим джерелом даних, які можуть бути інтегровані в системи, подібні до розробленої у нашій роботі. На відміну від підходу БУВР, наша інформаційна технологія зосереджена на автоматизації прогнозування та інтеграції

розвідувального і просторового аналізу, що дозволяє передбачати дефіцит води для конкретних ділянок, таких як с. Лелітка та с. Тростянчик.

На основі огляду подібних досліджень можна зробити висновок, що використання інформаційних технологій для аналізу водогосподарських балансів у басейні Південного Бугу є актуальним і активно розвивається. Проте більшість робіт зосереджені на статичному аналізі, ГІС-картографуванні або моніторингу, тоді як розроблена у цій роботі інформаційна технологія вирізняється застосуванням методів машинного навчання для прогнозування витрат води, інтеграцією розвідувального та просторового аналізу, а також акцентом на автоматизації для підтримки управлінських рішень у вразливих ділянках басейну.

1.3 Аналіз методів прогнозування

Мова програмування Python є потужним інструментом для вирішення задач аналізу даних і прогнозування завдяки своїй широкій екосистемі бібліотек, алгоритмів та інструментів. Вона активно застосовується у різних галузях, таких як економіка, екологія, аграрний сектор, енергетика, метеорологія та водне господарство. Python особливо ефективний для роботи з часовими рядами, статистичними моделями, просторовими даними та великими табличними масивами, що робить його ідеальним вибором для задач аналізу водогосподарських балансів річкового басейну Південного Бугу.

У рамках цієї бкр основною задачею є аналіз і прогнозування показників дефіциту води (Wдеф) та резервів води (Wрез) для водогосподарських ділянок (ВГД) басейну річки Південний Буг. Ці показники є безперервними числовими величинами, що залежать від ряду факторів: місяця, рівня забезпеченості (50%, 75%, 95%), гідрологічного сезону та специфіки кожної ділянки. Таким чином, задача прогнозування зводиться до регресії, де необхідно передбачити числові значення цільових змінних у майбутньому на основі історичних даних. Для реалізації цього використано середовище Kaggle та бібліотеки Python, зокрема Pandas для обробки даних, GeoPandas для просторового аналізу, Matplotlib і

Seaborn для візуалізації, а також низку методів прогнозування, які детально розглянуто нижче [5].

Огляд методів прогнозування

Лінійна регресія є базовим алгоритмом регресійного прогнозування, який моделює лінійну залежність між незалежними змінними (наприклад, місяць, рівень забезпеченості, обсяги припливу Wвх) та цільовими показниками (Wдеф або Wрез). Її перевага полягає у простоті та інтерпретованості: коефіцієнти моделі дозволяють чітко визначити вплив кожної змінної на результат. У контексті водогосподарських балансів лінійна регресія може бути застосована для оцінки впливу сезонних коливань на дефіцит води, хоча її ефективність обмежена у випадках нелінійних залежностей.

Метод опорних векторів (Support Vector Regression, SVR) є розширенням алгоритму опорних векторів для задач регресії. SVR намагається знайти гіперплощину, яка максимально наближається до реальних значень, ігноруючи незначні відхилення в межах заданої "зони нечутливості". Цей метод ефективний для нелінійних залежностей завдяки використанню ядерних функцій (наприклад, RBF-ядра) і добре працює з невеликими наборами даних, що є актуальним для ВГД із обмеженою кількістю історичних записів.

Random Forest Regressor — це ансамблевий метод, який базується на побудові великої кількості дерев рішень на випадкових підмножинах даних. Усереднення результатів дозволяє досягти високої точності прогнозування та зменшити ризик перенавчання. Random Forest стійкий до викидів і шумів у даних, а також здатний оцінити важливість кожної ознаки, що корисно для аналізу впливу факторів, таких як бічний приплив (Wбіч) чи транзит (Wтранзит), на дефіцит води.

На рисунку 1.3 представлено принцип дії моделі Random Forest, що ґрунтується на побудові ансамблю рішень з подальшою агрегацією результатів.

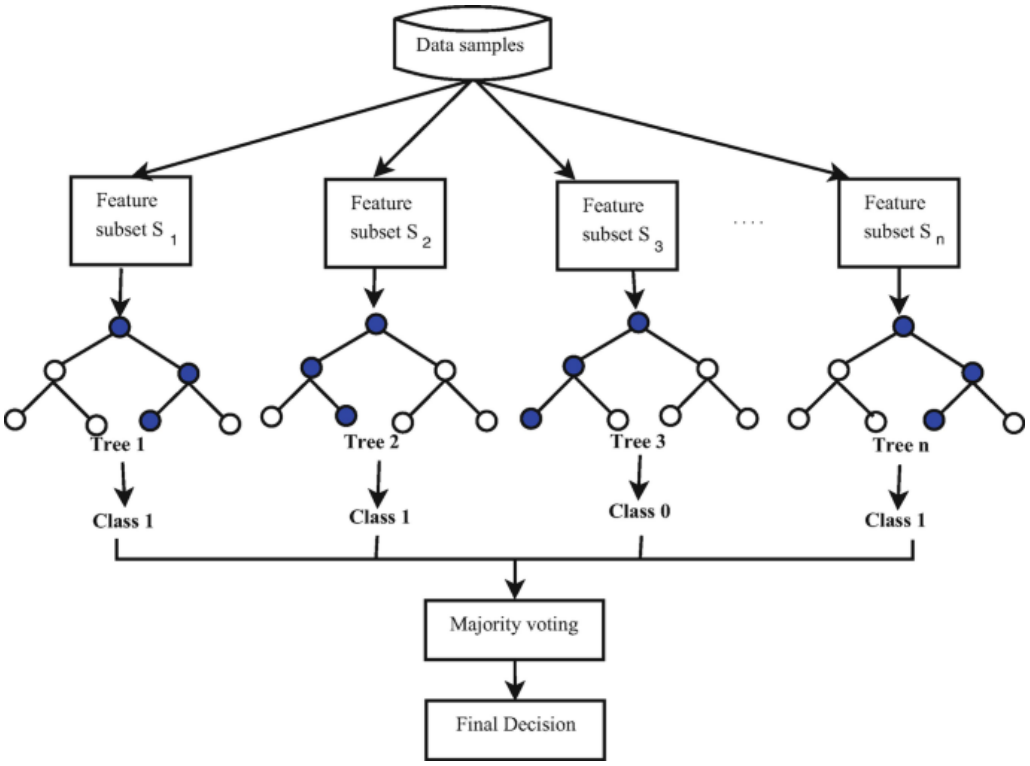

Рисунок 1.3 – Принцип роботи моделі Random Forest

Extra Trees (Extremely Randomized Trees) є модифікацією Random Forest, де випадковість додатково посилюється при виборі точок розділення у вузлах дерева. Це зменшує ймовірність перенавчання та покращує узагальнення моделі, що робить метод ефективним для роботи з даними, які мають високу мультиколінеарність або обмежений обсяг. Extra Trees часто застосовуються у гідрологічних задачах, де потрібно враховувати складні взаємозв'язки між змінними.

Decision Tree Regression базується на побудові дерева рішень, яке розбиває дані на підмножини з мінімальною внутрішньою дисперсією. Простота структури дерева забезпечує високу інтерпретованість моделі, що дозволяє легко зрозуміти логіку прогнозування. Метод ефективний для локальних прогнозів, наприклад, для окремих кластерів ВГД із подібними характеристиками.

Методи бустингу (AdaBoost, XGBoost) працюють за принципом послідовного навчання слабких моделей, де кожна наступна модель коригує помилки попередньої. XGBoost, зокрема, є одним із найпотужніших алгоритмів, який забезпечує високу точність завдяки оптимізації градієнтного спуску та регуляризації. Принцип роботи XGBoost наочно зображено на рисунку 1.4.

Кожне наступне дерево враховує помилки попередніх, а остаточний прогноз є результатом їхньої агрегації.

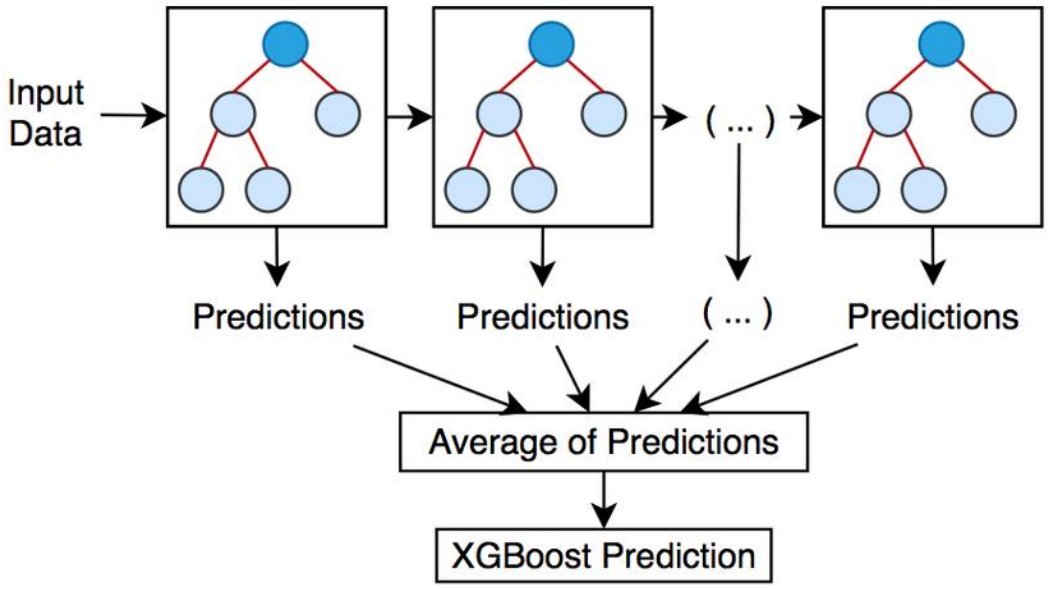

Рисунок 1.4 – Схема роботи алгоритму XGBoost

Багатошаровий перцептрон (Multi-Layer Perceptron, MLP) є представником нейронних мереж, здатним моделювати глибокі нелінійні залежності. MLP складається з кількох прихованих шарів нейронів, які трансформують вхідні дані для прогнозування. Цей метод потребує великої кількості даних і ретельного масштабування, але його потенціал для складних задач, таких як прогнозування часових рядів, є значним.

Prophet — це бібліотека від Facebook, спеціально розроблена для прогнозування часових рядів із урахуванням сезонності, трендів і аномалій. У роботі Prophet було використано для прогнозування Wдеф у Лелітці та Тростянчику, де модель врахувала сезонні коливання (зимові та літні місяці) та рівні забезпеченості, показавши хороші результати для довгострокових прогнозів [17].

Наступний рисунок 1.5 ілюструє процес прогнозування Prophet.

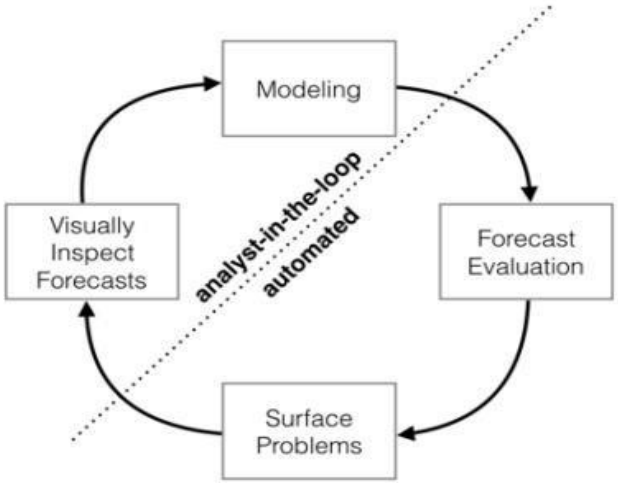

Рисунок 1.5 - Діаграма процесу прогнозування Prophet

SARIMA (Seasonal ARIMA) — це статистична модель для прогнозування часових рядів із сезонними компонентами. SARIMA ефективна для даних із чіткою періодичністю, як у випадку з місячними показниками водного балансу. У роботі модель була застосована до тих самих ділянок, враховуючи сезонність та тренди, що дозволило отримати точні прогнози для 2014 року на основі даних 2013 року [16].

Алгоритм автоматизованого підбору параметрів SARIMA відображено на рисунку 1.6. Він дозволяє обрати оптимальні значення параметрів (p, d, q, P, D, Q), що забезпечують мінімальну похибку прогнозу.

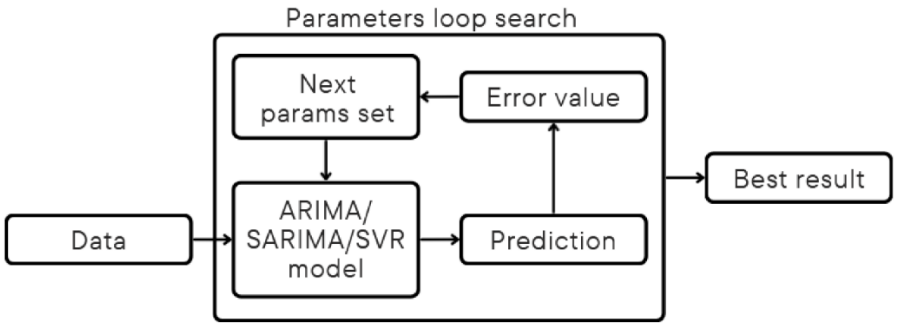

Рисунок 1.6 – Схема підбору параметрів моделі SARIMA з використанням циклу оптимізації похибки

LSTM (Long Short-Term Memory) — це тип рекурентних нейронних мереж, який добре підходить для моделювання часових рядів із довготривалими залежностями. LSTM було використано у дослідженні для прогнозування Wдеф,

враховуючи історичні дані та сезонні закономірності, що забезпечило високу точність для складних нелінійних залежностей [22].

Практична реалізація

У рамках роботи виконано повний цикл аналізу та прогнозування. Початкові дані з PDF-файлу було конвертовано у CSV-формат, після чого проведено розвідувальний аналіз за допомогою Pandas (перевірка пропусків, статистичний опис, кореляційний аналіз). Для візуалізації використано Matplotlib і Seaborn, що дозволило побудувати теплові карти кореляцій, лінійні графіки трендів та парні діаграми. Геопросторовий аналіз реалізовано за допомогою GeoPandas із використанням Shapefile-даних, що дало змогу створити карти дефіциту та резервів води, виявивши проблемні ділянки, такі як Лелітка та Тростянчик [6].

Для прогнозування використано три моделі: Prophet, SARIMA, LSTM. Кожна модель була протестована на даних 2013 року для прогнозування значень Wдеф у 2014 році, після чого результати порівняно за метриками точності (MAE, RMSE).

1.4 Висновки

У цьому розділі проведено ґрунтовний аналіз проблематики предметної області, пов’язаної з управлінням водними ресурсами річкового басейну Південного Бугу. Визначено мету дослідження, яка полягає у розробці інформаційної технології для аналізу водогосподарських балансів із виявленням проблемних ділянок, що мають дефіцит води, та оцінкою їхніх резервів. Об’єктом дослідження виступає інформаційна технологія, а предметом — процеси виявлення дефіцитів і резервів води, побудови візуалізацій та підготовки бази для прогнозних моделей. Аналіз підкреслив актуальність теми через вплив кліматичних змін, сезонних коливань та антропогенного навантаження на водозабезпечення басейну, що підтверджує необхідність автоматизованих аналітичних інструментів.

2 ВИБІР ОПТИМАЛЬНОЇ ІТ-ІНФРАСТРУКТУРИ

2.1 Вибір мови програмування

Для реалізації інформаційної технології прогнозування дефіциту та резервів води в басейні Південного Бугу було обрано мову програмування Python, що стало ключовим рішенням у контексті даної роботи. Цей вибір обґрунтований рядом переваг, які роблять Python ідеальним інструментом для аналізу даних, машинного навчання та обробки часових рядів — основних складових проєкту. Python вирізняється своєю простотою синтаксису, що дозволяє швидко розробляти прототипи, а також має розвинену екосистему відкритих бібліотек, які підтримують широкий спектр завдань, від завантаження даних до складного прогнозування. Крім того, мова має активну спільноту розробників, що забезпечує регулярні оновлення та доступ до документації, що є важливим для студентських проєктів із обмеженими ресурсами [23].

Популярність Python серед інших мов програмування підтверджується численними дослідженнями. Наприклад, за даними індексу TIOBE та опитувань Stack Overflow, Python посідає перші місця серед мов програмування для аналізу даних і машинного навчання протягом останніх років. На рисунку 2.1 представлено графік популярності Python у порівнянні з іншими мовами програмування, що ілюструє його домінування у 2024–2025 роках.

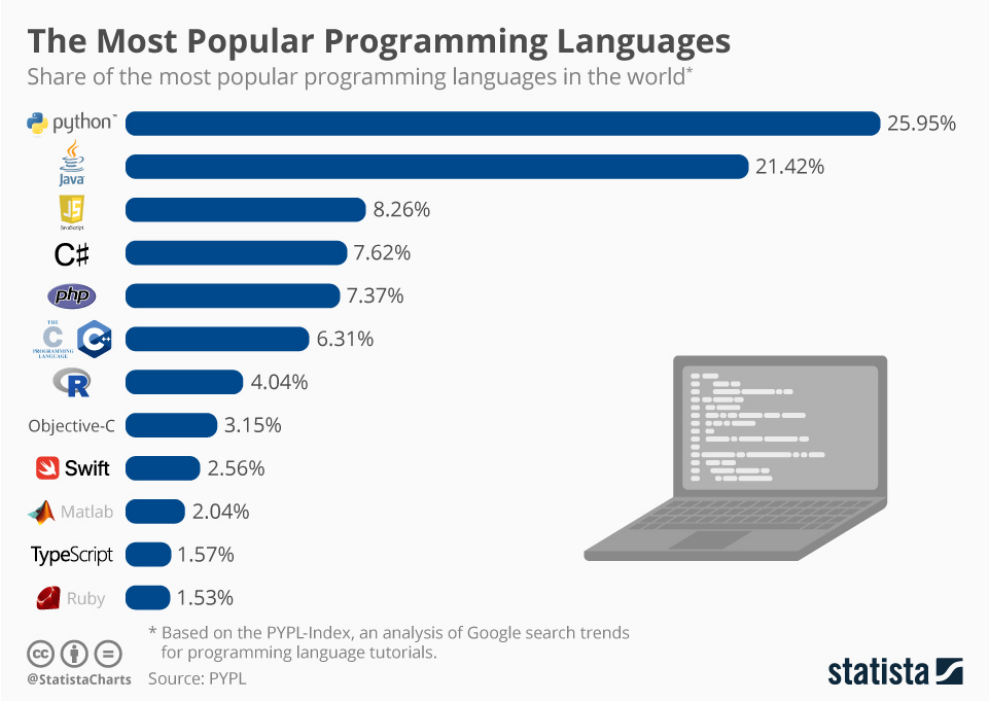

Рисунок 2.1 – Популярність Python серед мов програмування (2024–2025)

У процесі розробки інформаційної технології використано низку спеціалізованих бібліотек, кожна з яких виконувала унікальну роль:

- Pandas — бібліотека для завантаження, очищення та трансформації гідрологічних даних у форматі часових рядів. Вона дозволила ефективно обробляти дані с. Лелітка та с. Тростянчик, включаючи перетворення у формат long і створення індексів дат.
- NumPy — забезпечувала швидкі числові обчислення та роботу з масивами, що було критично важливим для масштабування обчислень під час підготовки даних для моделей [9].
- Matplotlib та Seaborn — використовувалися для створення детальних візуалізацій, таких як графіки часових рядів, гістограми для аналізу розподілу витрат води та boxplot для оцінки сезонних коливань. Ці інструменти допомогли виявити тренди та аномалії в даних [7].
- Prophet — спеціалізована бібліотека від Facebook, яка застосовувалася для моделювання сезонних коливань і довгострокових трендів у витратах води, враховуючи річну сезонність.

- pmdarima — розширення для бібліотеки statsmodels, яке забезпечило автоматичний підбір параметрів для моделі SARIMA, що значно спростило процес налаштування моделі для прогнозування [15].
- TensorFlow — потужна бібліотека для глибинного навчання, яка використовувалася для побудови моделі LSTM, адаптованої до складних нелінійних залежностей у часових рядах [12].
- Scikit-learn — універсальний інструмент для обчислення метрик оцінки моделей (MAE, RMSE, MAPE, R^2), а також для підготовки даних до тренування [10].

Вибір цих бібліотек обґрунтований їхньою спеціалізацією та інтеграцією з Python. Наприклад, Prophet і SARIMA ефективно моделюють сезонні закономірності, характерні для гідрологічних даних, тоді як LSTM доповнює аналіз, адаптуючись до динамічних змін. Для с. Лелітка SARIMA досягла найвищої точності з $R^2 = 0.94$, тоді як Prophet показав $R^2 = 0.92$. Для с. Тростянчик Prophet виявився найбільш точним із $R^2 = 0.86$, тоді як SARIMA мала $R^2 = 0.84$. Альтернативи, такі як R або MATLAB, розглядалися, але Python переміг завдяки своїй відкритості та доступності бібліотек для машинного навчання [11].

2.2 Вибір середовища

Для розробки, тестування та виконання інформаційної технології прогнозування витрат води обрано хмарне середовище Kaggle, яке стало оптимальним рішенням завдяки своїм функціональним можливостям та зручності використання. Kaggle надає інтерактивне середовище на базі Jupyter Notebook, що дозволяє одночасно писати код, візуалізувати результати та документувати процес аналізу, що є ідеальним для ітеративного підходу до прогнозування. Ця платформа стала основним інструментом завдяки інтеграції з Python та доступу до безкоштовних обчислювальних ресурсів, включаючи GPU, що значно прискорило тренування моделей, таких як LSTM, які потребують значних обчислювальних потужностей.

Важливо зазначити, що набори даних для ділянок с. Лелітка та с. Тростянчик не були взяті з готових джерел на Kaggle, а були створені самостійно шляхом обробки PDF-файлів із гідрологічними даними. Для цього використано спеціалізовані інструменти, такі як бібліотека PyPDF2 для Python, яка дозволила витягти табличні дані з PDF-документів. Далі ці дані було перенесено в CSV-формат за допомогою Pandas, що включало етапи очищення (видалення нечислових значень, заповнення пропусків), трансформації (переведення у формат long із мапуванням місяців) та сортування за датами. Цей процес був трудомістким, але забезпечив високу точність і відповідність даних реальним показникам витрат води, що є критично важливим для якісного прогнозування.

Переваги використання Kaggle для роботи з цими самостійно створеними датасетами включають:

- Обчислювальні ресурси: Kaggle пропонує безкоштовний доступ до GPU та TPU, що прискорило тренування моделей, особливо LSTM, яка потребує значних обчислень для навчання на часових рядах.
- Інтерактивність: Середовище Jupyter Notebook на Kaggle дозволило швидко тестувати код для обробки даних, побудови графіків і тренування моделей, а також документувати кожен етап роботи.
- Спільнота та документація: Наявність прикладів коду та навчальних матеріалів на Kaggle допомогла адаптувати рішення для обробки даних і прогнозування, зокрема при налаштуванні моделей Prophet і SARIMA.
- Масштабованість: Платформа дозволяє легко масштабувати обчислення при роботі з великими датасетами, що було корисним під час ітеративного вдосконалення моделей.

Альтернативними варіантами розглядалися локальне середовище на базі Anaconda з Jupyter Notebook, хмарна платформа Google Colab та комерційні сервіси, такі як AWS або Azure. Локальне середовище було відхилене через обмеженість обчислювальних ресурсів, особливо для тренування моделей LSTM, які потребують GPU. Google Colab розглядалося як додаткова опція, але Kaggle обрано через зручність завантаження власних датасетів у CSV-форматі та безкоштовний доступ до обчислень. Комерційні платформи, такі як AWS EC2

або Azure Machine Learning, могли б бути корисними для масштабування проєкту в комерційних цілях, але їхня вартість і складність налаштування перевищують потреби студентського дослідження.

Використання Kaggle у поєднанні з самостійно створеними датасетами дозволило ефективно організувати процес розробки, від обробки PDF-файлів до фінального прогнозування, і забезпечило високу продуктивність на всіх етапах роботи.

2.3 Рекомендація по апаратному забезпеченню

Для ефективної реалізації інформаційної технології прогнозування витрат води та забезпечення її стабільної роботи в реальних умовах необхідно врахувати вимоги до апаратного забезпечення, особливо при роботі з великими обсягами гідрологічних даних та тренуванні моделей машинного навчання. На основі проведених обчислень і специфіки виконаних завдань сформульовано наступні мінімальні рекомендації щодо апаратного забезпечення:

– Процесор: Багатоядерний процесор із високою продуктивністю, наприклад, Intel Core i5 11-го покоління або AMD Ryzen 5 5600X із частотою не менше 2.5–3.0 ГГц. Такі процесори забезпечують достатню швидкість обчислень для тренування моделі SARIMA, а також для попередньої обробки даних у Pandas.

– Оперативна пам’ять: Мінімум 16 ГБ оперативної пам’яті, бажано 32 ГБ, для комфортної роботи з великими наборами даних і одночасного виконання кількох моделей. Наприклад, модель LSTM потребує значного обсягу RAM через складність обчислень.

– Відеокарта: Для прискорення навчання моделей глибинного навчання, таких як LSTM, рекомендується використання GPU, наприклад, NVIDIA GTX 1650 із 4 ГБ пам’яті або NVIDIA RTX 3060 із 6–8 ГБ. У разі використання хмарних платформ, таких як Kaggle, GPU надається вбудовано, що знижує вимоги до локального обладнання.

– Накопичувач: SSD об’ємом не менше 256 ГБ, бажано 512 ГБ, для швидкого доступу до даних, зберігання наборів даних (наприклад, CSV-файлів для Лелітки та Тростянчика), збережених моделей та результатів прогнозування. SSD значно прискорює завантаження даних порівняно з HDD.

– Операційна система: Рекомендується використовувати Windows 10/11, Linux (наприклад, Ubuntu 20.04) або macOS, які сумісні з Python та всіма зазначеними бібліотеками. Linux є особливо зручним для роботи з відкритими даними та серверами.

– Мережеве з’єднання: Стабільний інтернет із швидкістю не менше 10 Мбіт/с для роботи з хмарними платформами, такими як Kaggle, і завантаження оновлень бібліотек.

Ці рекомендації базуються на вимогах до обчислень, які виникли під час роботи з даними с. Лелітка та с. Тростянчик, де модель LSTM вимагала значних ресурсів. Якщо планується масштабування проєкту для обробки даних у реальному часі або роботи з більшими наборами даних (наприклад, для інших ділянок басейну Південного Бугу), доцільно розглянути використання хмарних сервісів, таких як AWS EC2 із конфігурацією t3.large (2 vCPU, 8 GB RAM) або Google Cloud Compute Engine із GPU-опціями. Такі платформи дозволяють адаптувати обчислювальні ресурси до поточних потреб і забезпечують резервування даних для подальшого аналізу.

2.4 Висновки

В другому розділі проведено детальний аналіз і обґрунтування вибору ключових компонентів для реалізації інформаційної технології прогнозування витрат води в басейні Південного Бугу. Основним рішенням стало використання мови програмування Python завдяки її універсальності, простоті синтаксису та інтеграції з бібліотеками для аналізу даних і машинного навчання. Для моделювання обрано методи Prophet, SARIMA та LSTM, які ефективно справляються з сезонними закономірностями та динамічними змінами в гідрологічних даних.

Для розробки використано хмарне середовище Kaggle, яке забезпечило необхідні обчислювальні ресурси, зокрема GPU, для тренування моделей машинного навчання, а також зручний інтерфейс на базі Jupyter Notebook для ітеративного аналізу. Kaggle став оптимальним вибором для роботи з самостійно створеними датасетами, підготовленими шляхом витягнення даних із PDF-файлів та їхньої подальшої обробки, що демонструє гнучкість обраного середовища для нестандартних завдань.

Щодо апаратного забезпечення, сформульовано рекомендації, які включають багатоядерний процесор (Intel Core i5 або AMD Ryzen 5), 16–32 ГБ оперативної пам’яті, GPU (NVIDIA GTX 1650 або RTX 3060), SSD-накопичувач об’ємом 256–512 ГБ та стабільне мережеве з’єднання. Ці вимоги враховують обчислювальні потреби моделей, таких як LSTM, і дозволяють масштабувати систему за допомогою хмарних платформ, таких як AWS або Google Cloud, у разі розширення проєкту.

Таким чином, обрана IT-інфраструктура є оптимальною для вирішення поставленого завдання, забезпечуючи ефективну обробку даних, високу продуктивність і можливість подальшого розвитку інформаційної технології для прогнозування водних ресурсів.

3 ПІДГОТОВКА ТА РОЗВІДУВАЛЬНИЙ АНАЛІЗ ДАНИХ

3.1 Розвідувальний аналіз даних

Першим етапом аналізу стало ознайомлення з даними, які були зібрані та сформовані у власний датасет. Первинною формою даних був PDF-документ, що містив табличну інформацію про щомісячні водогосподарські баланси по 11 водогосподарських ділянках (ВГД) басейну річки Південний Буг. Для зручності обробки та подальшого аналізу дані було вручну перенесено та структуровано у формат CSV відповідно до заданої логіки.

Сформований датасет включає такі ключові стовпці:

- Код ВГД – унікальний ідентифікатор кожної водогосподарської ділянки;
- Річка – назва річки або її відгалуження, що відповідає ділянці;
- Забезпеченість, % – рівень водозабезпечення (50, 75 або 95% забезпеченості);
- Місяць – номер місяця, за який проводився облік;
- Кількість днів – тривалість облікового періоду у днях;
- Wвх, Wпзв, Wрез, Wдеф, Wресурси, Wвсього – числові показники водного балансу в млн м³ (вхід, поповнення, резерв, дефіцит, ресурси тощо).

Таким чином, кожен рядок датасету відповідає певній ділянці, за певного місяця і рівня забезпеченості. Така структура дозволяє не лише аналізувати ситуацію в окремих ділянках, а й виконувати групування, агрегацію, побудову динаміки та прогнозів на основі часових рядів.

Сформований датасет зображено на рисунку 3.1.

	Річка	ВГД	Код ВГД	Забезпеченість, %	Місяць	Кількість днів	Wвх, млн куб. м	Wбіч, млн куб. м	Wпзв, млн куб. м	Wзв, млн куб. м	Wдот, млн куб. м	Wрез_ΔV, млн куб. м	Wресурси, млн куб. м	Wвип, млн куб. м	Wф, млн куб. м	Ws, млн куб. м	Wпер, млн куб. м	Wвкр, млн куб. м	We, млн куб. м	Wвсього, млн куб. м	Wдеф, млн куб. м	Wрез, млн куб. м	Wтранзит, млн куб. м
0	Південний Буг	від витоку до гирла р. Іква	M5.4.0.01	50	1	31	0.0000	34.7656	0.5124	0.9915	0.0	0.7909	37.0604	0.0000	0.0	0.0512	0.0	0.4430	17.2551	17.7493	0.0	19.3111	36.5662
1	Південний Буг	від витоку до гирла р. Іква	M5.4.0.01	50	2	28	0.0000	31.4012	0.5130	0.9915	0.0	-0.7909	32.1148	0.0000	0.0	0.0513	0.0	0.4548	15.5853	16.0914	0.0	16.0234	31.6087
2	Південний Буг	від витоку до гирла р. Іква	M5.4.0.01	50	3	31	0.0000	34.7656	0.5061	0.9915	0.0	0.3264	36.5897	0.0006	0.0	0.0506	0.0	2.1449	18.4132	20.6093	0.0	15.9804	34.3936
3	Південний Буг	від витоку до гирла р. Іква	M5.4.0.01	50	4	30	0.0000	33.6442	0.7844	0.9915	0.0	-0.3264	35.0937	0.0029	0.0	0.0784	0.0	2.5690	17.8192	20.4695	0.0	14.6241	32.4433
4	Південний Буг	від витоку до гирла р. Іква	M5.4.0.01	50	5	31	0.0000	34.7656	0.7964	0.9915	0.0	0.0000	36.5536	0.0043	0.0	0.0796	0.0	1.7039	18.4132	20.2011	0.0	16.3525	34.7657
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
391	Бузький лиман	Бузький лиман	M5.4.0.11	95	8	31	165.0040	0.0000	0.1780	2.6817	0.0	0.0000	167.8636	0.0144	0.0	0.0178	0.0	0.9530	0.0000	0.9852	0.0	166.8785	166.8785
392	Бузький лиман	Бузький лиман	M5.4.0.11	95	9	30	159.6813	0.0000	0.1610	2.6817	0.0	0.0000	162.5239	0.0101	0.0	0.0161	0.0	0.7480	0.0000	0.7742	0.0	161.7497	161.7497
393	Бузький лиман	Бузький лиман	M5.4.0.11	95	10	31	165.0040	0.0000	0.1340	2.6817	0.0	0.0000	167.8196	0.0059	0.0	0.0134	0.0	0.9260	0.0000	0.9453	0.0	166.8743	166.8743
394	Бузький лиман	Бузький лиман	M5.4.0.11	95	11	30	159.6813	0.0000	0.1210	2.6817	0.0	0.0000	162.4839	0.0025	0.0	0.0121	0.0	1.2340	0.0000	1.2486	0.0	161.2353	161.2353
395	Бузький	Бузький	M5.4.0.11	95	12	31	165.0040	0.0000	0.1010	2.6817	0.0	0.0000	167.7866	0.0000	0.0	0.0101	0.0	1.1580	0.0000	1.1681	0.0	166.6185	166.6185

Рисунок 3.1 – Сформований датасет

Першим етапом розвідувального аналізу стало зчитування CSV-файлу з платформи Kaggle, у якому містяться оброблені дані водогосподарських балансів по річці Південний Буг. Після успішного зчитування було здійснено перевірку типів даних та кількості рядків і стовпців у таблиці.

Розмір таблиці становить 396 рядків та 23 колонки. Дані охоплюють різні показники для кожної водогосподарської ділянки: назва річки, код ВГД, забезпеченість (%), місяць, кількість днів, обсяги водних надходжень та витрат за різними каналами (вхідні, бічні, транзитні, випаровування тощо), дефіцит і резерв води, а також інші аналітичні параметри.

Також було виконано ручне налаштування ширини відображення таблиці та формат чисел з плаваючою комою до чотирьох знаків після коми — для зручності аналізу числових характеристик. Окремо було усунено конфлікт версій бібліотеки shapely, яка буде використана на наступних етапах для роботи з просторовими даними.

На рисунку 3.2 зображено перегляд типів даних.

```
✅ Розмір таблиці: (396, 23)
Річка                    object
ВГД                      object
Код ВГД                  object
Забезпеченість, %         int64
Місяць                    int64
Кількість днів            int64
Wвх, млн куб. м         float64
Wбіч, млн куб. м        float64
Wпзв, млн куб. м        float64
Wзв, млн куб. м         float64
Wдот, млн куб. м        float64
Wрез_ΔV, млн куб. м     float64
Wресурси, млн куб. м    float64
Wвип, млн куб. м        float64
Wф, млн куб. м          float64
Wз, млн куб. м          float64
Wпер, млн куб. м        float64
Wвкр, млн куб. м        float64
We, млн куб. м          float64
Wвсього, млн куб. м     float64
Wдеф, млн куб. м        float64
Wрез, млн куб. м        float64
Wтранзит, млн куб. м    float64
dtype: object
```

Рисунок 3.2 – Зчитування датасету та перегляд типів даних

У рамках розвідувального аналізу було виконано статистичний опис числових змінних у сформованому датасеті. Метод describe() дозволив отримати базові характеристики кожного числового стовпця: кількість значень (count), середнє арифметичне (mean), стандартне відхилення (std), мінімальне та максимальне значення, а також значення відповідно до квартилів (25%, 50%, 75%).

На основі цього можна побачити загальний розподіл числових даних. Наприклад, середній обсяг водних ресурсів (Wресурси) становить приблизно 86 млн м³, при цьому спостерігається висока дисперсія, що свідчить про значну варіативність у різних ВГД. Значення дефіциту (Wдеф) також мають розкид – від 0 до 7.1 млн м³, що є важливою характеристикою для подальшого аналізу дефіцитних ділянок. Особливо варто звернути увагу на стовпець Wрез_ΔV, у якому є як позитивні, так і від’ємні значення, що свідчить про наявність як приросту, так і зменшення запасів у водогосподарських одиницях.

Крім того, було здійснено перевірку на пропущені значення. Результат показав, що всі колонки заповнені повністю, відсутні будь-які NaN значення. Це свідчить про коректну підготовку даних і дозволяє проводити подальший аналіз без додаткового очищення.

На рисунку 3.3 зображено результат перевірки даних на пропуски.

	Забезпеченість, %	Місяць	Кількість днів	Wвх, млн куб. м	Wбіч, млн куб. м	Wпзв, млн куб. м	Wзв, млн куб. м	Wдот, млн куб. м	Wрез_ΔV, млн куб. м	Wресурси, млн куб. м	Wвип, млн куб. м	Wф, млн куб. м	Wз, млн куб. м	Wпер, млн куб. м	Wвкр, млн куб. м	We, млн куб. м	Wвсього, млн куб. м	Wдеф, млн куб. м	Wрез, млн куб. м	Wтранзит, млн куб. м
count	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000	396.0000
mean	73.3333	6.5000	30.4167	65.4664	18.2813	0.4657	1.8855	0.0000	0.0000	86.0990	0.0032	0.0000	0.0466	0.0000	3.1752	16.2637	19.4887	0.0449	66.6103	82.8741
std	18.4322	3.4564	0.8631	80.3176	16.7929	0.2618	1.2778	0.0000	1.5120	77.4664	0.0032	0.0000	0.0262	0.0000	3.1429	17.3265	18.4197	0.4770	69.1110	76.9753
min	50.0000	1.0000	28.0000	0.0000	-19.5453	0.0861	0.3145	0.0000	-9.4449	9.6692	0.0000	0.0000	0.0086	0.0000	0.2335	0.0000	0.2449	0.0000	-7.1086	-3.9022
25%	50.0000	3.7500	30.0000	0.0000	4.2940	0.1623	0.5052	0.0000	0.0000	24.5784	0.0003	0.0000	0.0163	0.0000	0.9206	3.4018	5.3749	0.0000	15.6413	22.1104
50%	75.0000	6.5000	31.0000	23.2114	17.6627	0.4721	1.6805	0.0000	0.0000	42.2759	0.0027	0.0000	0.0472	0.0000	1.9454	8.5864	11.0725	0.0000	34.4009	39.7540
75%	95.0000	9.2500	31.0000	135.8663	28.2100	0.6371	2.8447	0.0000	0.0000	141.2812	0.0050	0.0000	0.0637	0.0000	4.7287	32.4347	36.2375	0.0000	99.7663	137.1975
max	95.0000	12.0000	31.0000	280.4781	55.0808	0.9745	4.0165	0.0000	8.4939	283.3378	0.0144	0.0000	0.0974	0.0000	22.5020	51.3875	57.3713	7.1086	282.4593	282.4593

✅ Усі дані заповнені — пропусків немає.

Рисунок 3.3 – Статистичний опис числових колонок та перевірка на пропуски

На цьому етапі розвідувального аналізу було досліджено кількість унікальних значень у ключових категоріях датасету, зокрема: по кодах водогосподарських ділянок (ВГД), рівнях забезпеченості та місяцях спостережень.

Спершу було визначено унікальні значення в полі Код ВГД. Загалом присутні 11 різних ділянок, кожна з яких має по 36 записів, що свідчить про рівномірність у зборі даних по кожній ВГД протягом 12 місяців на 3 рівнях забезпеченості.

Далі проаналізовано рівні забезпеченості водними ресурсами. Всього маємо три категорії: 50%, 75% і 95%. Кожен рівень представлено однаковою кількістю записів (по 132), що знову ж таки підтверджує збалансовану структуру даних.

Також було перевірено розподіл записів за місяцями. Кожен місяць (від 1 до 12) представлений у датасеті по 33 рази, що забезпечує коректність щомісячного аналізу та виключає зміщення по сезонам.

Загалом, така структура набору даних дозволяє здійснювати коректний порівняльний аналіз між ділянками, місяцями та рівнями забезпеченості, без необхідності додаткового балансування або обробки нестачі даних.

На рисунку 3.4 зображено розподіл по ВГД, рівнях забезпеченості та місяцях.

```
Унікальні коди ВГД:
['M5.4.0.01' 'M5.4.0.02' 'M5.4.0.03' 'M5.4.0.04' 'M5.4.0.05' 'M5.4.0.06'
 'M5.4.0.07' 'M5.4.0.08' 'M5.4.0.09' 'M5.4.0.10' 'M5.4.0.11']

Розподіл по ВГД:
Код ВГД
M5.4.0.01    36
M5.4.0.02    36
M5.4.0.03    36
M5.4.0.04    36
M5.4.0.05    36
M5.4.0.06    36
M5.4.0.07    36
M5.4.0.08    36
M5.4.0.09    36
M5.4.0.10    36
M5.4.0.11    36
Name: count, dtype: int64

Рівні забезпеченості:
Забезпеченість, %
50    132
75    132
95    132
Name: count, dtype: int64

Розподіл по місяцях:
Місяць
1     33
2     33
3     33
4     33
5     33
6     33
7     33
8     33
9     33
10    33
11    33
12    33
Name: count, dtype: int64
```

Рисунок 3.4 – Розподіл по ВГД, рівнях забезпеченості та місяцях

Наступним кроком у розвідувальному аналізі стала побудова теплової карти (heatmap), яка дозволяє наочно представити середній дефіцит води (Wдеф) по кожній водогосподарській ділянці (ВГД) у розрізі місяців.

Для цього було згруповано дані за допомогою зведеної таблиці, де в якості індексу використано код ВГД, а стовпцями виступають місяці. Значенням служить середнє значення показника Wдеф, млн куб. м.

На отриманій тепловій карті кольором позначено рівень дефіциту: чим темніший колір, тим більший дефіцит води у відповідному місяці для конкретної ВГД. Завдяки такій візуалізації можна швидко виявити ділянки та періоди з найбільш критичним дефіцитом.

Аналізуючи карту, можна помітити, що дефіцит не є рівномірним, а має чітко виражені пік-місяці. Наприклад, ділянка M5.4.0.01 має найвищий рівень дефіциту у листопаді (місяць 11), а M5.4.0.03 – у травні (місяць 5). Інші ділянки

показують або нульові значення, або незначний дефіцит, що вказує на локалізовану проблему у конкретних сегментах річки Південний Буг.

Таке представлення є надзвичайно ефективним для виявлення просторово-часових аномалій у водному балансі та слугує основою для подальшого прогнозування.

На рисунку 3.5 зображено теплову карту середнього дефіциту води.

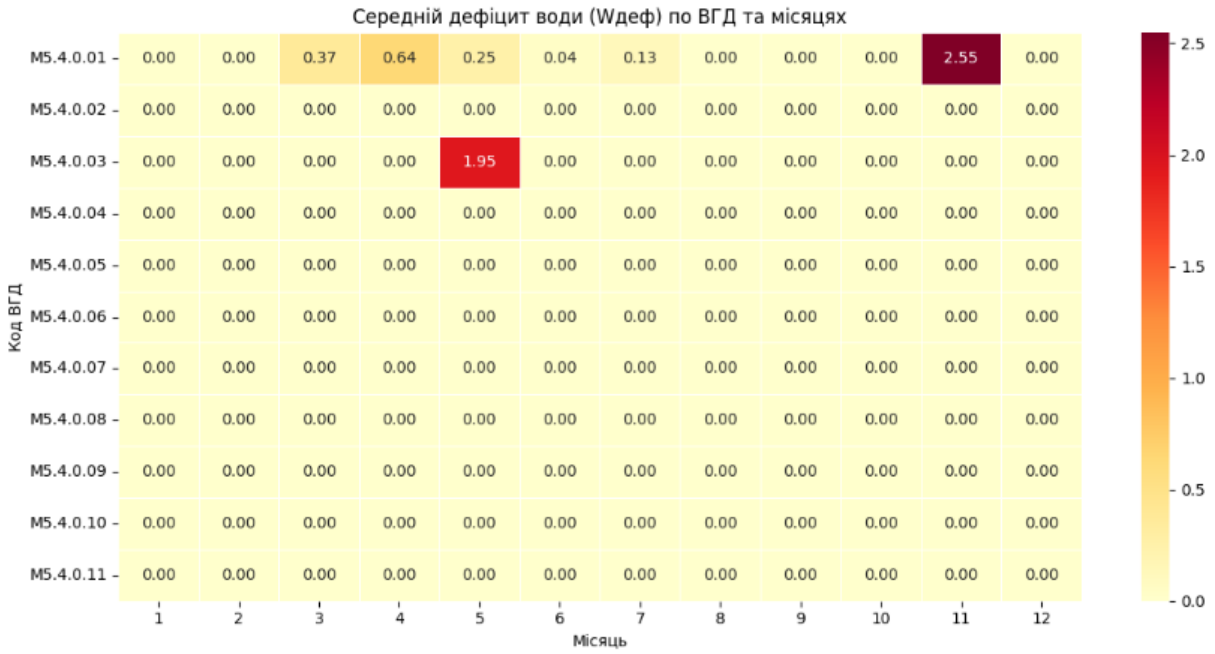

Рисунок 3.5 – Теплова карта середнього дефіциту води (Wдеф) по ВГД та місяцях

У даному етапі аналізу побудовано лінійний графік середнього значення показника Wвсього, млн куб. м (загальна кількість води) для кожного рівня забезпеченості (50%, 75% і 95%) у розрізі календарних місяців.

Попередньо дані було агреговано за допомогою групування по стовпцях Місяць та Забезпеченість, %, після чого обчислено середнє значення Wвсього для кожної комбінації. Отримані дані візуалізовано на графіку за допомогою бібліотеки Seaborn [24].

На графіку видно характер змін об’єму води впродовж року для різних рівнів забезпеченості. Кожна лінія відповідає певному рівню забезпеченості та дозволяє порівняти сезонні коливання обсягів води. Наприклад, пік

спостерігається у травні (місяць 5), що може бути пов'язано з весняним водопіллям, характерним для цього періоду. У літні місяці об’єм води знижується, що пояснюється активним водоспоживанням та зменшенням притоку.

Цей графік дозволяє виявити загальні тренди, а також порівняти поведінку водного балансу при різних рівнях забезпеченості, що важливо для прийняття рішень у водогосподарському плануванні.

На рисунку 3.6 зображено середнє значення Wвсього по місяцях.

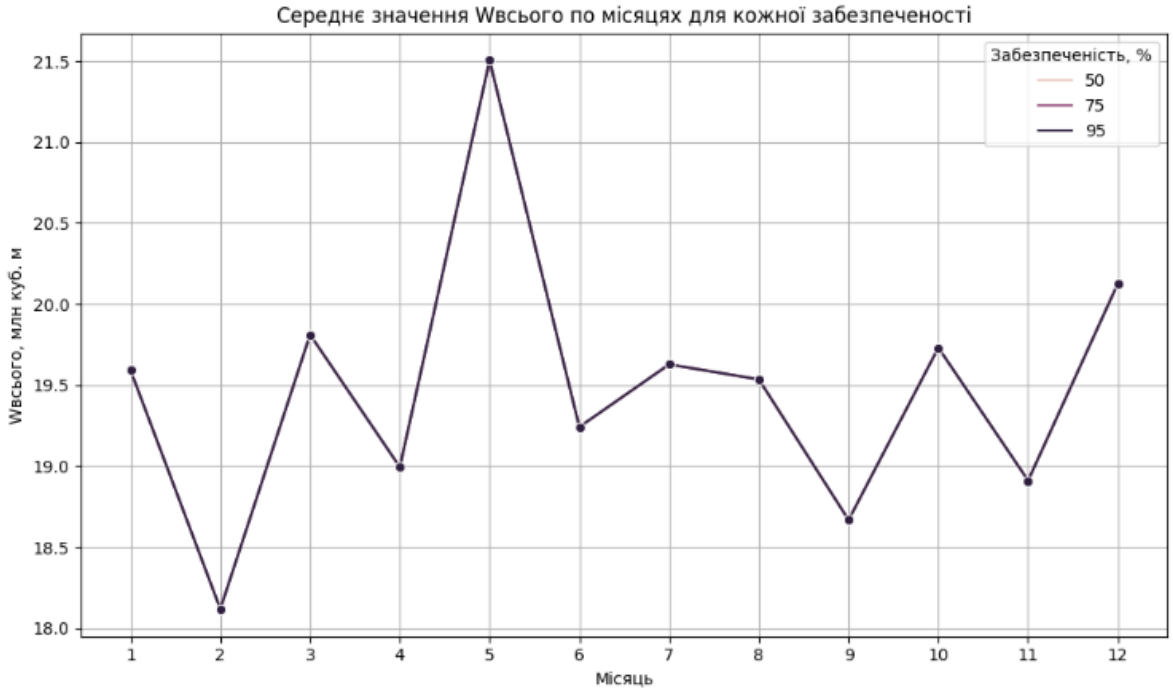

Рисунок 3.6 – Середнє значення Wвсього по місяцях для кожного рівня забезпеченості

Наступним етапом розвідувального аналізу стало дослідження дефіциту води (Wдеф, млн куб. м) у розрізі місяців для кожного рівня забезпеченості (50%, 75% і 95%).

Було виконано групування даних за ознаками Місяць та Забезпеченість, % з подальшим розрахунком середнього значення дефіциту води (Wдеф). Це дозволило сформувати агрегований набір даних, який візуалізовано за допомогою лінійного графіка.

На графіку чітко відображено місяці, в яких спостерігається підвищений дефіцит води. Зокрема, при 50% забезпеченості спостерігаються помітні піки дефіциту у травні та листопаді, що свідчить про ймовірне навантаження на водогосподарські системи у ці періоди. Водночас, при 95% забезпеченості дефіцит практично відсутній, що демонструє надійність водопостачання у відповідних умовах.

Такий підхід дозволяє виявити критичні періоди, коли дефіцит досягає максимальних значень, і може бути використаний для обґрунтування управлінських рішень, пов’язаних з регулюванням водоспоживання у певні місяці.

На рисунку 3.7 зображено середній дефіцит води по місяцям.

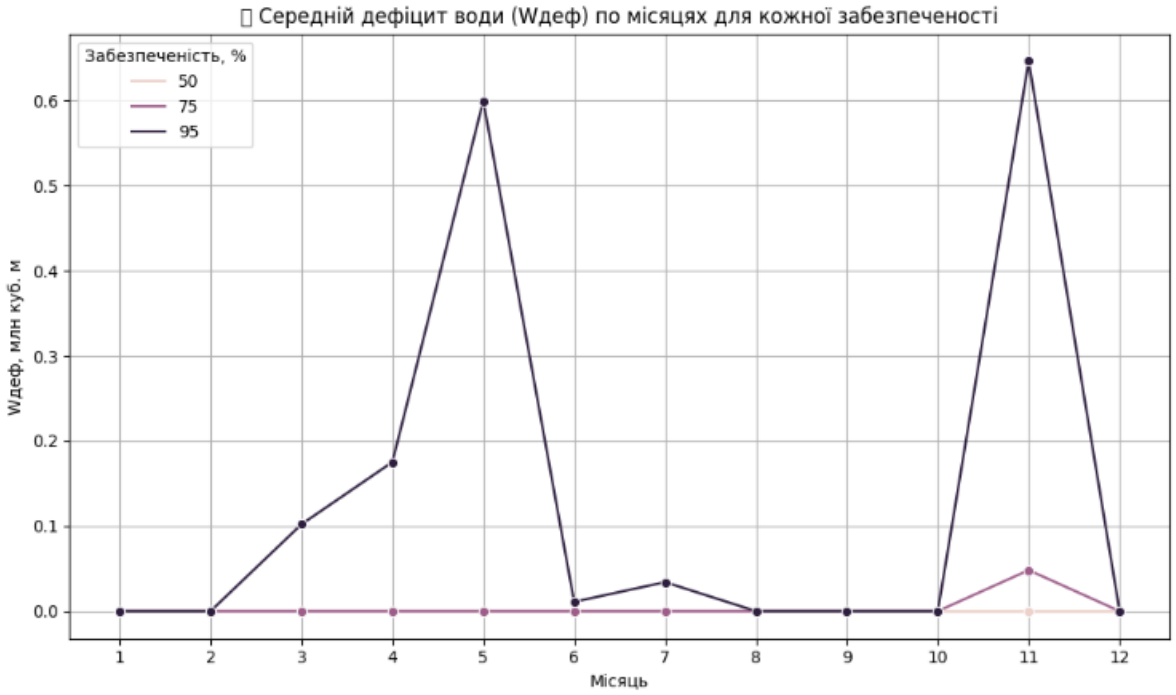

Рисунок 3.7 – Середній дефіцит води (Wдеф) по місяцях для кожного рівня забезпеченості

На завершення розвідувального аналізу було побудовано кореляційну матрицю між усіма числовими ознаками датасету. Для цього з набору даних були відібрані лише числові колонки, після чого за допомогою методу corr() обчислено коефіцієнти кореляції Пірсона між кожною парою змінних.

Кореляційна матриця візуалізована у вигляді теплової карти (heatmap), де значення варіюються від -1 до 1. Синій колір вказує на від’ємну кореляцію, червоний – на додатну, а насиченість кольору свідчить про силу зв’язку.

З аналізу матриці можна зробити кілька важливих висновків:

– Існує сильний позитивний зв’язок між змінними Wвсього, млн куб. м, Wресурси, млн куб. м, Wвип, млн куб. м та Wрез, млн куб. м. Це логічно, адже ці показники формуються на основі одних і тих самих об’ємів водних надходжень.

– Спостерігається висока кореляція між Wдеф, млн куб. м та Wтранзит, млн куб. м, що може свідчити про залежність обсягів дефіциту від кількості транзитної води.

– Також помітно, що місяць (Місяць) має незначну кореляцію з іншими параметрами, що дозволяє використовувати його як нейтральну змінну у подальшому моделюванні.

Таке дослідження дозволяє зрозуміти, які змінні можуть впливати одна на одну, виявити мультиколінеарність, а також обрати релевантні ознаки для побудови моделей прогнозування.

На рисунку 3.8 зображена кореляційна матриця між числовими ознаками.

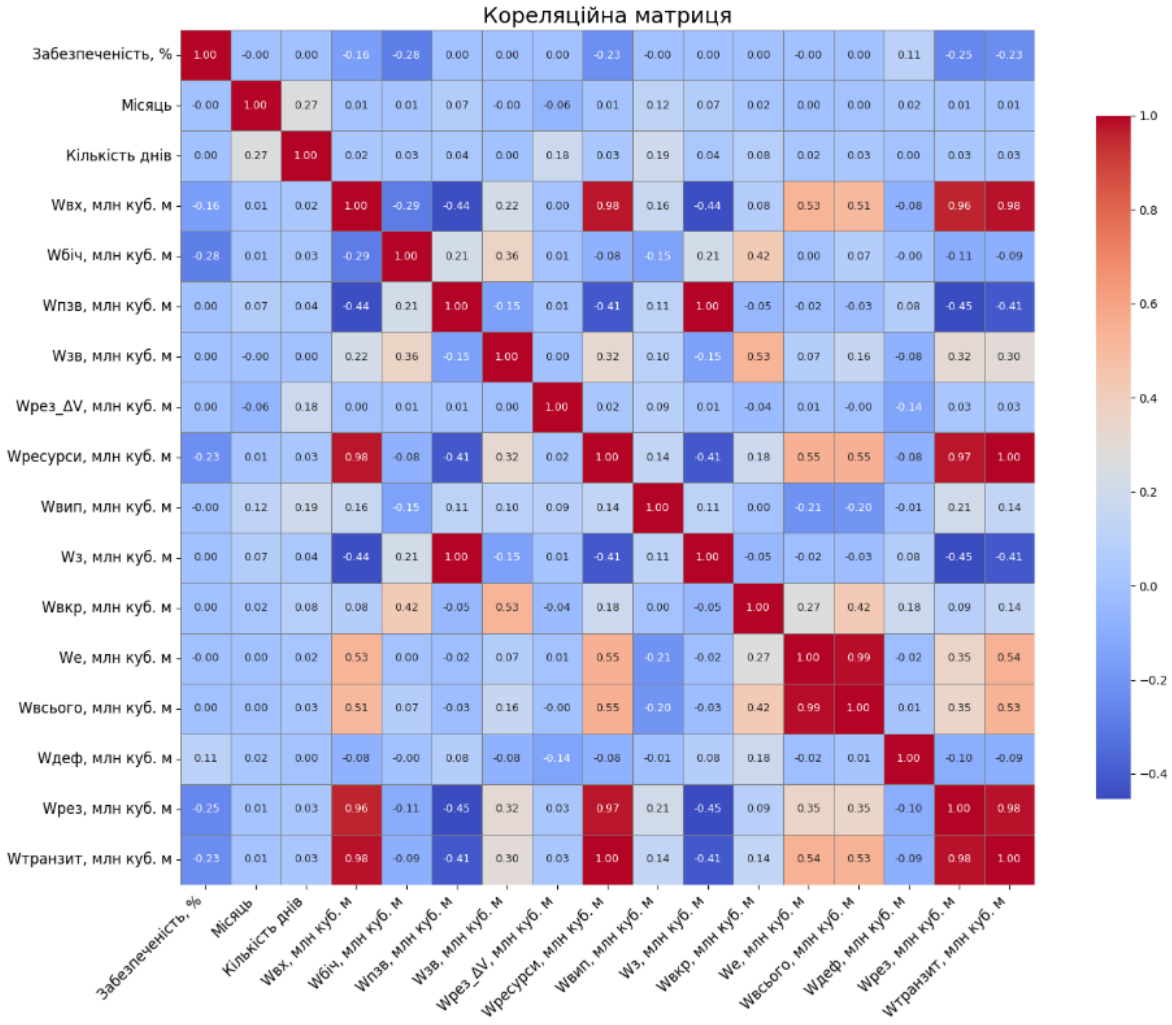

Рисунок 3.8 – Кореляційна матриця між числовими ознаками

З метою оцінки обсягів водних ресурсів на різних ділянках річки Південний Буг було проведено агрегування показника Wвсього, млн куб. м за кодами водогосподарських ділянок (ВГД). Для цього застосовано групування значень за стовпцем "Код ВГД" з подальшим обчисленням суми показника Wвсього для кожного унікального коду.

Візуалізація даних реалізована у вигляді стовпчикової діаграми (barplot), яка дозволяє наочно порівняти сумарні обсяги води для кожної ВГД. Це дозволяє швидко ідентифікувати ділянки з найбільшими або найменшими сумарними водними ресурсами.

Згідно з побудованим графіком, найбільший сумарний обсяг води спостерігається для ділянок з кодами M5.4.0.08 та M5.4.0.07, що свідчить про їхню потенційну важливість у регіональному водному балансі. У той час як деякі інші ділянки, такі як M5.4.0.10 або M5.4.0.11, мають в рази менші показники, що

може бути пов’язано з географічним розташуванням або гідрологічними особливостями.

Такий аналіз є важливою частиною розвідувального етапу, оскільки дозволяє сформувати уявлення про нерівномірність розподілу водних ресурсів у межах досліджуваного басейну.

На рисунку 3.9 зображено сумарну величину Wвсього по кожному коду ВГД.

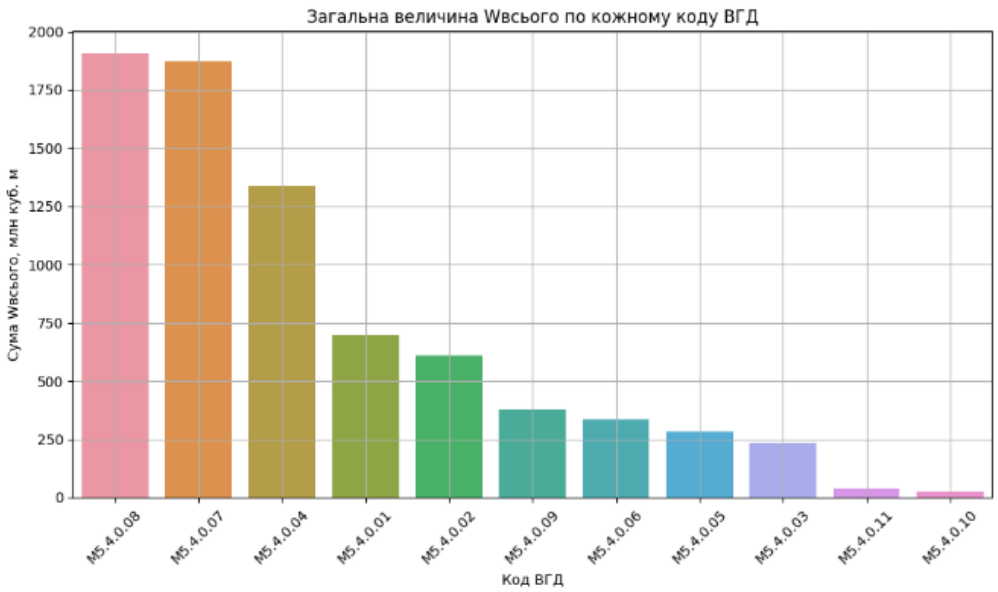

Рисунок 3.9 – Сумарна величина Wвсього по кожному коду ВГД

3.2 Просторовий аналіз дефіциту та резервів води

У межах бкр було реалізовано просторову аналітику дефіциту та резервів води для водогосподарських ділянок річки Південний Буг. Основною метою стало виявлення регіонів, що мають підвищене водне навантаження або, навпаки, потенціал для акумуляції водних ресурсів. Для досягнення цієї мети було здійснено побудову тематичних карт на основі геоданих та розширеної статистики, зібраної у ході обробки CSV-даних.

На першому етапі аналізу ми побудували карту, що відображає дефіцит води (Wдеф, млн куб. м) у розрізі кожної ВГД. Для цього застосовано метод .plot() із GeoDataFrame gdf_def, попередньо сформованого шляхом об’єднання

основної таблиці даних з геометрією (границями ділянок). Завдяки аргументу column='Wдеф, млн куб. м' ми задали показник, що використовується для тематичного зафарбування ділянок, а колірна палітра cmap='Reds' дозволила інтуїтивно відобразити рівень дефіциту — від світло-рожевого (мінімум) до темно-червоного (максимум).

Ця візуалізація надзвичайно інформативна: вона дозволяє швидко ідентифікувати проблемні ділянки, що мають суттєве відставання у водному балансі. Завдяки просторовому сприйняттю можна оцінити не лише рівень дефіциту, а й географічну близькість до інших критичних регіонів, що важливо для управлінських рішень у сфері водокористування.

На рисунку 3.10 зображено тематичну карту дефіциту води.

Рисунок 3.10 – Тематична карта дефіциту води (Wдеф) по водогосподарських ділянках

Наступним кроком стало створення карти, яка ілюструє резерв води (Wрез, млн куб. м) по тих самих ВГД. Візуалізація виконана аналогічно попередній, однак із застосуванням палітри "Blues" — ділянки з найбільшими запасами мають темно-синій відтінок, тоді як мінімальні значення представлені світлими тонами. Це дає змогу візуально оцінити потенціал кожної ділянки до збереження води — один із ключових факторів при плануванні водогосподарських заходів.

Важливо зазначити, що карта резервів є дзеркальним доповненням до карти дефіциту: разом вони створюють повну картину водного балансу в басейні річки. Також така візуалізація дозволяє виділити регіони, які могли б виступати донорами ресурсів за наявності відповідної інфраструктури.

На рисунку 3.11 зображено тематичну карту резрву води.

Рисунок 3.11 – Тематична карта резерву води (Wрез) по водогосподарських ділянках

3.3 Висновки

У другому розділі проведено підготовку та розвідувальний аналіз даних для вирішення задачі прогнозування дефіциту та резервів води в басейні Південного Бугу. На основі зібраних даних із первинних PDF-файлів створено відповідні датасети, що містять інформацію про водні ресурси.

Зібрано дані, які включають ідентифікатори водогосподарських ділянок (ВГД), дати збору даних, обсяги припливу (Wвх), бічного припливу (Wбіч), дефіциту (Wдеф), резервів (Wрез) та рівні забезпеченості (50%, 75%, 95%). Окремо створено датасет із геопросторовими даними (Shapefile) для прив’язки ВГД до координат.

Розроблено ноутбук для візуалізації даних по конкретних датах, що допомагає зрозуміти просторовий розподіл дефіциту та резервів у басейні

Південного Бугу. Графіки, створені на основі цих даних, дозволяють виявити проблемні зони, такі як Лелітка та Тростянчик, та проводити подальший аналіз, забезпечуючи наочне представлення рівнів дефіциту на різний час.

Ці результати закладають основу для подальших кроків у розробці інформаційної технології прогнозування та дозволяють більш глибоко зрозуміти та аналізувати проблему водного балансу в регіоні.

4 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ТА ПРОГНОЗУВАННЯ

4.1 Розроблення інформаційної технології

Для розробки інформаційної технології прогнозування дефіциту та резервів води в басейні Південного Бугу пропонується такий алгоритм (рис. 4.1):

1. Визначення цільової ознаки, яку необхідно прогнозувати, а саме дефіцит води (Wдеф) або резерви (Wрез), залежно від мети прогнозування.

2. Аналіз даних для виявлення закономірностей та трендів у водних балансах, таких як сезонні коливання чи залежність від припливу.

3. Виявлення та фільтрація аномальних даних, зокрема значень Wдеф, що перевищують 95% максимуму, для забезпечення якості даних перед моделюванням [29].

4. Перевірка обсягу даних після фільтрації: якщо кількість залишених даних становить менше 70% від початкового обсягу, процес завершується з повідомленням про недостатність даних; інакше — перехід до наступного етапу.

5. Створення тренувального, валідаційного та тестового наборів даних для підготовки моделей прогнозування.

6. Тренування моделей на тренувальному наборі даних із подальшою оцінкою їхньої продуктивності на валідаційному наборі за допомогою метрик, таких як MAE чи R^2.

7. Прогнозування значень Wдеф на тестовому наборі даних та оцінка точності прогнозів за обраними метриками.

8. Аналіз результатів: якщо точність моделі (наприклад, $R^2 > 0.8$) відповідає встановленим вимогам, модель вважається оптимальною; інакше — повернення до етапу налаштування моделей для повторного тюнінгу. Цей цикл повторюється до досягнення необхідної точності або вичерпання кількості ітерацій.

Рисунок 4.1 – Блок-схема алгоритму інформаційної технології аналізу водогосподарських балансів ділянок басейну Південного Бугу

4.2 Прогнозування витрат води по гідропосту с.Лелітка

Підготовка даних для аналізу часового ряду

Процес прогнозування розпочато з підготовки даних для гідропоста с. Лелітка. Спочатку визначено список місяців українською мовою та створено мапу для їхнього перетворення у числові значення від 1 до 12. На рисунку 4.2 зображено фрагмент коду, що реалізує ці визначення.

```
# Колонки місяців для Лелітки
month_columns_lelitka = ['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
                         'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень']

# Мапа назв місяців у номер для Лелітки
month_map_lelitka = {
    'Січень': 1, 'Лютий': 2, 'Березень': 3, 'Квітень': 4,
    'Травень': 5, 'Червень': 6, 'Липень': 7, 'Серпень': 8,
    'Вересень': 9, 'Жовтень': 10, 'Листопад': 11, 'Грудень': 12
}

# --- Підготовка даних ---
# Завантажуємо CSV для Лелітки
lelitka_df = pd.read_csv("/kaggle/input/gidropost-lelitka/lelitka_cleaned.csv")

# Перетворюємо з широкого формату в довгий
lelitka_long = lelitka_df.melt(id_vars="Рік", var_name="Місяць", value_name="Витрата")
lelitka_long["Місяць"] = lelitka_long["Місяць"].map(month_map_lelitka)
lelitka_long = lelitka_long.dropna(subset=["Місяць"])

# Видаляємо нечислові значення в колонці "Рік"
lelitka_long = lelitka_long[lelitka_long["Рік"].astype(str).str.isnumeric()]
lelitka_long["Рік"] = lelitka_long["Рік"].astype(int)
lelitka_long["Місяць"] = lelitka_long["Місяць"].astype(int)

# Створюємо колонку з датою
lelitka_long["date"] = pd.to_datetime(dict(year=lelitka_long["Рік"], month=lelitka_long["Місяць"], day=1))
lelitka_long = lelitka_long.sort_values("date").reset_index(drop=True)

# Готова таблиця часового ряду
ts_df_lelitka = lelitka_long[["date", "Витрата"]].copy()
ts_df_lelitka = ts_df_lelitka.set_index("date").asfreq("MS").ffill()

# Перевірка на пропуски
print("Пропуски в ts_df_lelitka:", ts_df_lelitka.isna().sum().values[0])

# Налаштування стилю графіків
plt.style.use('seaborn')
```

Рисунок 4.2 – Програмний код для визначення місяців і їхньої мапи

Після підготовки часового ряду даних для с. Лелітка виконано візуалізацію витрат води (Wдеф) для аналізу трендів і сезонних коливань. Для цього побудовано графік часового ряду за допомогою бібліотеки Matplotlib, де осі X і Y позначено як "Дата" та "Витрата (млн м^3)" відповідно, а також додано сітку для

зручності читання. На рисунку 4.3 зображено фрагмент коду для створення графіка.

```
plt.figure(figsize=(12, 4))
ts_df_lelitka["Витрата"].plot()
plt.title("Часовий ряд витрат води (Лелітка)")
plt.xlabel("Дата")
plt.ylabel("Витрата (млн м³)")
plt.grid(True)
plt.tight_layout()
plt.show()
print("Відображено: Часовий ряд витрат води (Лелітка)")
plt.close()
```

Рисунок 4.3 – Програмний код для побудови графіка часового ряду

Результатом виконання коду є графік, який відображає динаміку витрат води для с. Лелітка. Після відображення графіка виконано його закриття для уникнення накопичення об’єктів у пам’яті. На рисунку 4.4 представлено приклад графіка, отриманого в результаті.

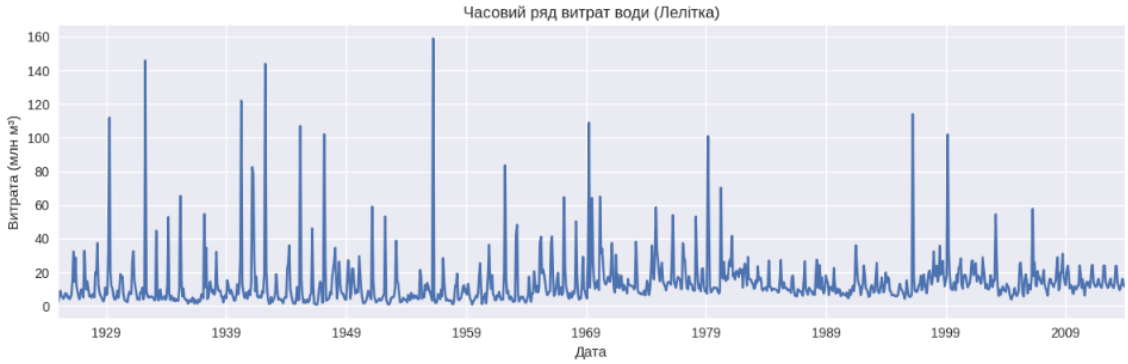

Рисунок 4.4 – Графік часового ряду витрат води для с. Лелітка

Для глибшого розуміння розподілу витрат води на ділянці с. Лелітка побудовано гістограму, яка відображає частоту значень витрат (Wдеф). Використано 50 бінів для деталізації розподілу, а графік оформлено з позначками осей "Витрата (млн м³)" та "Частота" та сіткою для зручності аналізу. На рисунку 4.5 зображено фрагмент коду для створення гістограми.

```
plt.figure(figsize=(10, 4))
ts_df_lelitka["Витрата"].plot(kind="hist", bins=50)
plt.title("Гістограма витрат води (Лелітка)")
plt.xlabel("Витрата (млн м³)")
plt.ylabel("Частота")
plt.grid(True)
plt.tight_layout()
plt.show()
print("Відображено: Гістограма витрат води (Лелітка)")
plt.close()
```

Рисунок 4.5 – Програмний код для побудови гістограми витрат води.

Результатом виконання коду є гістограма, яка дозволяє оцінити розподіл витрат води. Після відображення графіка виконано його закриття для оптимізації пам’яті. На рисунку 4.6 представлено приклад гістограми, отриманої в результаті.

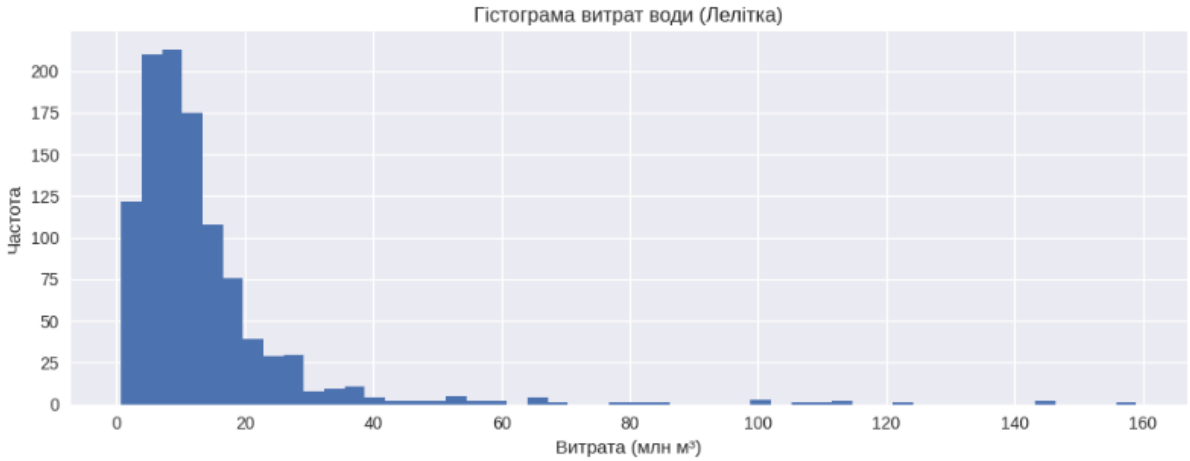

Рисунок 4.6 – Гістограма витрат води для с. Лелітка

Далі для виявлення трендів у даних розраховано ковзне середнє витрат води з вікном у 6 місяців. Графік ковзного середнього побудовано з використанням методу rolling().mean(), із зазначенням осей "Дата" та "Витрата (млн м³)" та додаванням сітки. На рисунку 4.7 зображено фрагмент коду для створення графіка ковзного середнього.

```
plt.figure(figsize=(12, 4))
ts_df_lelitka["Витрата"].rolling(window=6).mean().plot()
plt.title("Ковзне середнє витрат (6 місяців, Лелітка)")
plt.xlabel("Дата")
plt.ylabel("Витрата (млн м³)")
plt.grid(True)
plt.tight_layout()
plt.show()
print("Відображено: Ковзне середнє витрат (6 місяців, Лелітка)")
plt.close()
```

Рисунок 4.7 – Програмний код для побудови графіка ковзного середнього

Результатом є графік, що відображає згладжений тренд витрат води для с. Лелітка. Після відображення графіка виконано його закриття. На рисунку 4.8 представлено приклад графіка ковзного середнього, отриманого в результаті.

Рисунок 4.8 – Графік ковзного середнього витрат води для с. Лелітка

Для оцінки сезонних коливань витрат води на ділянці с. Лелітка проведено аналіз розподілу витрат (Wдеф) за місяцями. Спочатку до часового ряду додано стовпець "Місяць", витягнувши значення місяця з індексу дат. Потім побудовано діаграму розподілу (boxplot) за допомогою бібліотеки Seaborn, де по осі X відображено місяці, а по осі Y — витрати води в мільйонах кубічних метрів. На рисунку 4.9 зображено фрагмент коду для створення boxplot.

```
ts_df_lelitka["Місяць"] = ts_df_lelitka.index.month
plt.figure(figsize=(12, 5))
sns.boxplot(x="Місяць", y="Витрата", data=ts_df_lelitka.reset_index())
plt.title("Boxplot витрат води по місяцях (Лелітка)")
plt.xlabel("Місяць")
plt.ylabel("Витрата (млн м³)")
plt.grid(True)
plt.tight_layout()
plt.show()
print("Відображено: Boxplot витрат води по місяцях (Лелітка)")
plt.close()
```

Рисунок 4.9 – Програмний код для побудови boxplot витрат води по місяцях

Результатом виконання коду є діаграма, яка дозволяє виявити сезонні закономірності та можливі аномалії у витратах води для с. Лелітка. Після відображення графіка виконано його закриття для оптимізації пам’яті. На рисунку 4.10 представлено приклад boxplot, отриманого в результаті.

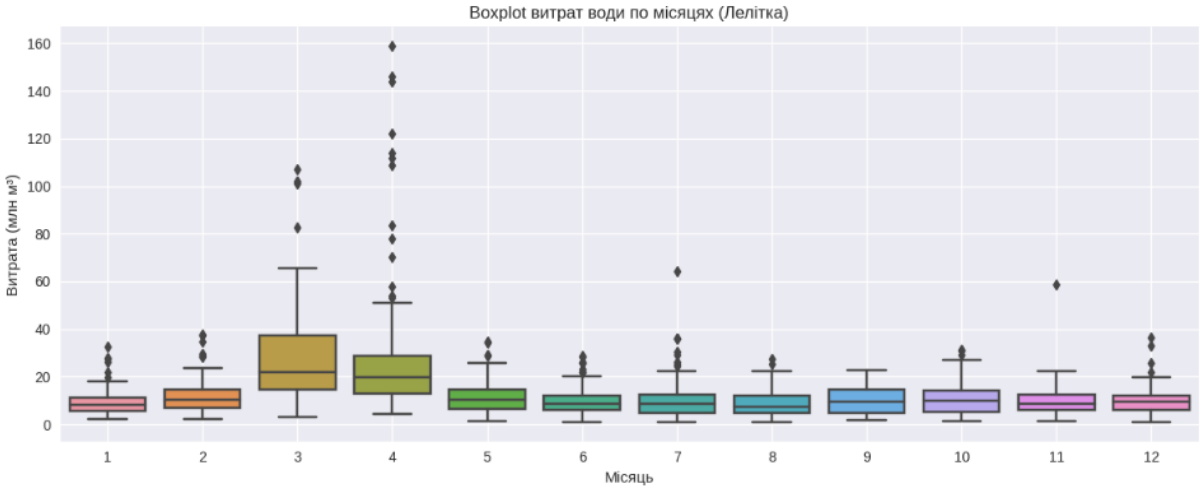

Рисунок 4.10 – Boxplot витрат води по місяцях для с. Лелітка

Після аналізу часового ряду для с. Лелітка виконано прогнозування витрат води (Wдеф) з використанням моделей Prophet і SARIMA.

Спочатку підготовлено дані для моделі Prophet. Для цього створено копію часового ряду, виконано логарифмування значень витрат із використанням np.log1p для стабілізації дисперсії, а також додано регресори (lag1, lag2, lag3, rolling_mean_3, spring_peak), які враховують історичні лаги та сезонні піки (березень-квітень). Дані відформатовано у структуру, сумісну з Prophet, із стовпцем дат "ds" і значень "y". Далі ініціалізовано модель Prophet із увімкненою

річною сезонністю, адитивним режимом, налаштовано параметри (changepoint_prior_scale=1.5, n_changepoints=25, seasonality_prior_scale=40.0) та додано кастомну сезонність із періодом 12 і fourier_order=40. Модель натреновано, а прогноз виконано на 24 місяці (2013–2014 роки) із ітеративним оновленням лагів і ковзного середнього для 2014 року. Результати зворотньо трансформовано з логарифмічного формату за допомогою np.expm1. На рисунках 4.11 і 4.12 зображено фрагменти коду для підготовки даних, навчання моделі та прогнозування.

```
df_lelitka = pd.read_csv("/kaggle/input/gidropost-lelitka/lelitka_cleaned.csv")
df_lelitka = df_lelitka[~df_lelitka['Рік'].isin(['Середнє', 'Найбільше', 'Найменше'])]
df_lelitka['Рік'] = df_lelitka['Рік'].astype(int)
for month in ['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
              'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень']:
    df_lelitka[month] = pd.to_numeric(df_lelitka[month], errors='coerce')
month_map = {
    'Січень': 1, 'Лютий': 2, 'Березень': 3, 'Квітень': 4, 'Травень': 5, 'Червень': 6,
    'Липень': 7, 'Серпень': 8, 'Вересень': 9, 'Жовтень': 10, 'Листопад': 11, 'Грудень': 12
}
df_long_lelitka = df_lelitka.melt(id_vars=["Рік"],
                                  value_vars=['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
                                              'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень'],
                                  var_name="Місяць_назва", value_name="Витрата")
df_long_lelitka["Місяць"] = df_long_lelitka["Місяць_назва"].map(month_map)
df_long_lelitka = df_long_lelitka.dropna(subset=["Витрата"])
df_long_lelitka["date"] = pd.to_datetime(dict(year=df_long_lelitka["Рік"], month=df_long_lelitka["Місяць"], day=1))
df_long_lelitka = df_long_lelitka.sort_values("date").reset_index(drop=True)
# Додавання ознак
df_long_lelitka['lag1'] = df_long_lelitka['Витрата'].shift(1)
df_long_lelitka['lag2'] = df_long_lelitka['Витрата'].shift(2)
df_long_lelitka['lag3'] = df_long_lelitka['Витрата'].shift(3)
df_long_lelitka['rolling_mean_3'] = df_long_lelitka['Витрата'].rolling(window=3, min_periods=1).mean()
df_long_lelitka['spring_peak'] = ((df_long_lelitka["Місяць"].isin([3, 4]))).astype(int)
# Виправлення пропущених значень
df_long_lelitka['lag1'] = df_long_lelitka['lag1'].fillna(0)
df_long_lelitka['lag2'] = df_long_lelitka['lag2'].fillna(0)
df_long_lelitka['lag3'] = df_long_lelitka['lag3'].fillna(0)
df_long_lelitka['rolling_mean_3'] = df_long_lelitka['rolling_mean_3'].fillna(df_long_lelitka['Витрата'])
# Розбиття на тренувальні та тестові дані
train_data = df_long_lelitka[df_long_lelitka["date"].dt.year < 2013].copy()
test_data_2013 = df_long_lelitka[df_long_lelitka["date"].dt.year == 2013].copy()
# Формат для Prophet
prophet_df = pd.DataFrame({
    "ds": train_data["date"],
    "y": np.log1p(train_data["Витрата"]),
    "lag1": np.log1p(train_data["lag1"]),
    "lag2": np.log1p(train_data["lag2"]),
    "lag3": np.log1p(train_data["lag3"]),
    "rolling_mean_3": np.log1p(train_data["rolling_mean_3"]),
    "spring_peak": train_data["spring_peak"]
})
# Модель Prophet
model_prophet = Prophet(
    growth='linear',
    yearly_seasonality=True,
    seasonality_mode='additive',
    changepoint_prior_scale=1.5,
    n_changepoints=25,
    seasonality_prior_scale=40.0
)
model_prophet.add_regressor('lag1')
model_prophet.add_regressor('lag2')
model_prophet.add_regressor('lag3')
model_prophet.add_regressor('rolling_mean_3')
model_prophet.add_regressor('spring_peak')
model_prophet.add_seasonality(name='yearly_custom', period=12, fourier_order=40)
model_prophet.fit(prophet_df)
# Створення майбутнього фрейму для 2013-2014 років
future_dates = pd.date_range(start='2013-01-01', end='2014-12-01', freq='MS')
future_df = pd.DataFrame({'ds': future_dates})
```

Рисунок 4.11 – Програмний код для підготовки даних та прогнозування моделлю Prophet (1 частина)

```
future_df['spring_peak'] = ((future_df['ds'].dt.month.isin([3, 4]))).astype(int)
# Ініціалізація лагів і ковзного середнього
last_train_data = train_data.iloc[-1]
future_df.loc[future_df.index[0], 'lag1'] = np.log1p(last_train_data['Витрата'])
future_df.loc[future_df.index[0], 'lag2'] = np.log1p(last_train_data['Витрата'])
future_df.loc[future_df.index[0], 'lag3'] = np.log1p(last_train_data['Витрата'])
future_df.loc[future_df.index[0], 'rolling_mean_3'] = np.log1p(last_train_data['rolling_mean_3'])
# Ітеративний прогноз із оновленням лагів
forecast_list = []
lag_buffer = [np.log1p(last_train_data['Витрата'])] * 3
rolling_buffer = [np.log1p(last_train_data['rolling_mean_3'])]
for i, date in enumerate(future_dates):
    temp_df = future_df[future_df['ds'] == date].copy()
    # Для 2013 року використовуємо історичні лаги, якщо доступні
    if date in test_data_2013['date'].values:
        idx = test_data_2013[test_data_2013['date'] == date].index[0]
        temp_df['lag1'] = np.log1p(test_data_2013.loc[idx, 'lag1'])
        temp_df['lag2'] = np.log1p(test_data_2013.loc[idx, 'lag2'])
        temp_df['lag3'] = np.log1p(test_data_2013.loc[idx, 'lag3'])
        temp_df['rolling_mean_3'] = np.log1p(test_data_2013.loc[idx, 'rolling_mean_3'])
    else:  # Для 2014 року використовуємо оновлені лаги
        temp_df['lag1'] = lag_buffer[0]
        temp_df['lag2'] = lag_buffer[1]
        temp_df['lag3'] = lag_buffer[2]
        temp_df['rolling_mean_3'] = np.mean(rolling_buffer) if len(rolling_buffer) >= 1 else np.log1p(last_train_data['rolling_mean_3'])

    temp_forecast = model_prophet.predict(temp_df)
    forecast_list.append(temp_forecast)

    # Оновлюємо буфери для наступного кроку (лише для 2014 року)
    if date not in test_data_2013['date'].values and i > 0:
        yhat = np.expm1(temp_forecast['yhat'].iloc[0])
        lag_buffer[2] = lag_buffer[1]
        lag_buffer[1] = lag_buffer[0]
        lag_buffer[0] = np.log1p(yhat)
        rolling_buffer.append(np.log1p(yhat))
        if len(rolling_buffer) > 3:
            rolling_buffer.pop(0)
# Об'єднуємо прогнози
forecast = pd.concat(forecast_list)
forecast["yhat"] = np.expm1(forecast["yhat"])
forecast["yhat_lower"] = np.expm1(forecast["yhat_lower"])
forecast["yhat_upper"] = np.expm1(forecast["yhat_upper"])
# Оцінка R² за 2013 рік
actual_2013 = test_data_2013["Витрата"]
pred_2013 = forecast[forecast["ds"].dt.year == 2013]["yhat"]
r2_prophet = r2_score(actual_2013, pred_2013)
print(f"Prophet R² for 2013: {r2_prophet:.4f}")
# Графік
plt.figure(figsize=(12, 6))
plt.plot(test_data_2013["date"], actual_2013, label="Історичні дані 2013", marker="o", color="black")
plt.plot(forecast["ds"], forecast["yhat"], label="Прогноз Prophet", linestyle="--", color="blue")
plt.fill_between(forecast["ds"], forecast["yhat_lower"], forecast["yhat_upper"], alpha=0.3, color="lightblue", label="Довірчий інтервал")
plt.title("Прогноз витрат води (Лелітка): Prophet, 2013-2014")
plt.xlabel("Дата")
plt.ylabel("Витрата (млн м³)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Рисунок 4.12 – Програмний код для підготовки даних та прогнозування моделлю Prophet (2 частина)

Результатом є графік, що відображає прогноз витрат води для с. Лелітка на 2013–2014 роки, із зазначенням довірчого інтервалу та реальних даних 2013 року для порівняння. Після відображення графіка виконано його закриття. На рисунку 4.13 показано отриманий графік.

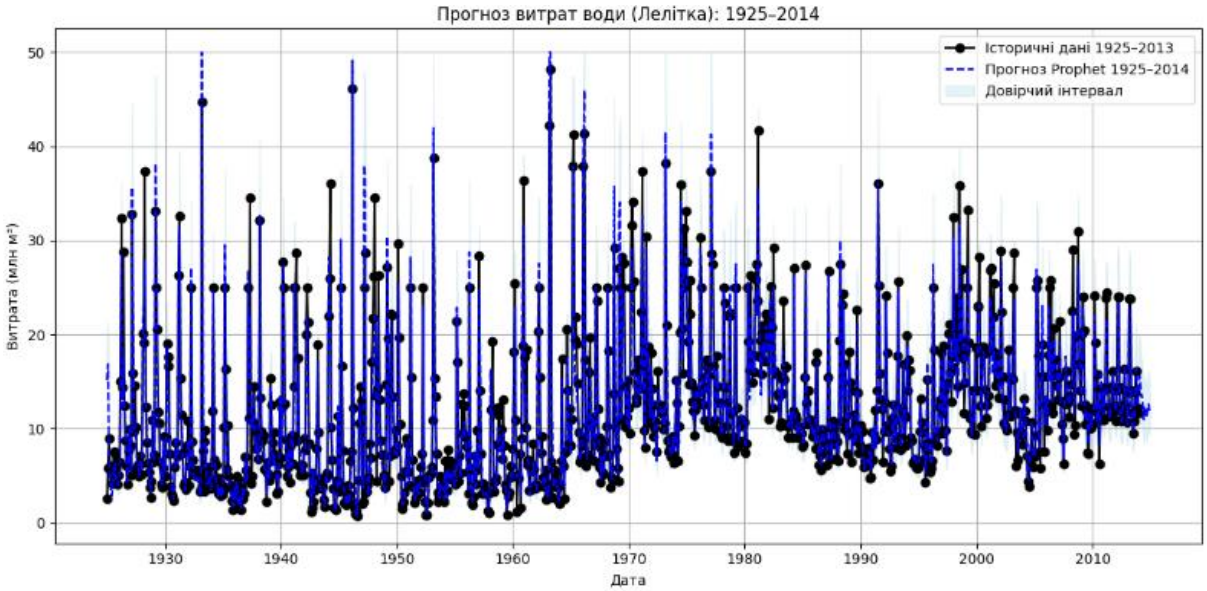

Рисунок 4.13 (А) – Графік прогнозу витрат води моделлю Prophet для с. Лелітка (2013–2014)

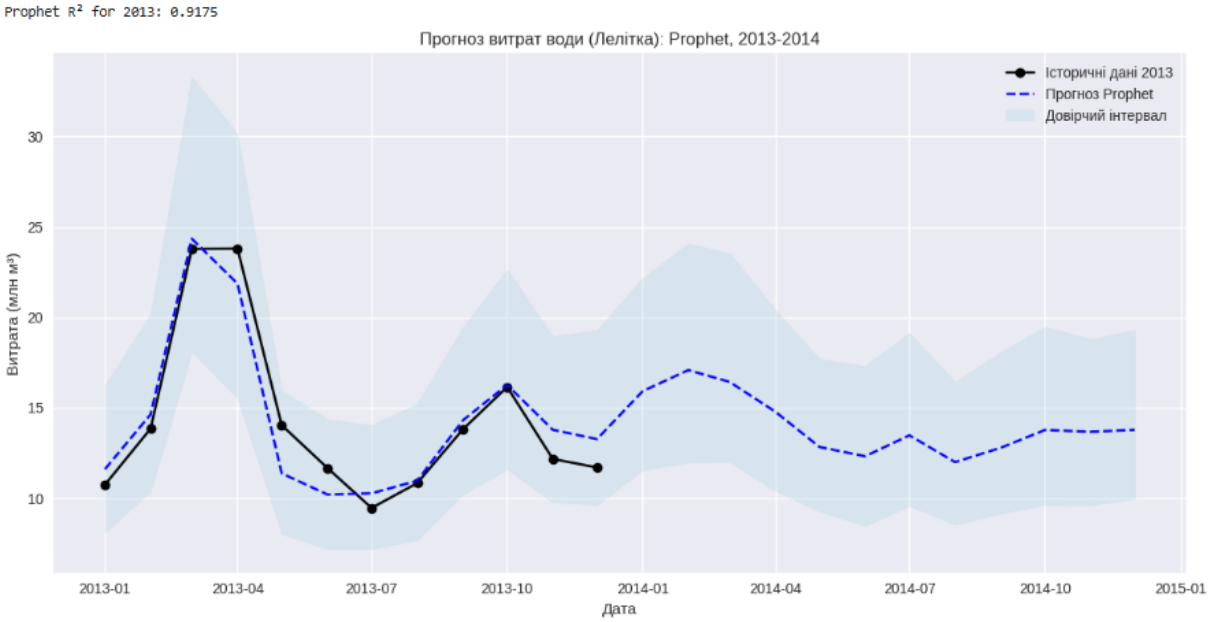

Рисунок 4.13 (Б) – Графік прогнозу витрат води моделлю Prophet для с. Лелітка (2013–2014)

Далі виконано прогнозування з використанням моделі SARIMA. Для цього відібрано часовий ряд витрат води до кінця 2012 року, застосовано логарифмічну трансформацію значень із np.log для стабілізації дисперсії та вручну налаштовано модель SARIMA з параметрами order=(2,1,2) і seasonal_order=(1,1,2,12). Модель натреновано, а прогноз виконано на 24 місяці

(2013–2014 роки) із розрахунком довірчих інтервалів. Результати зворотньо трансформовано з логарифмічного формату за допомогою np.exp і збережено у DataFrame із відповідними датами. На рисунку 4.14 і 4.15 зображено фрагменти коду для навчання моделі SARIMA та прогнозування.

```
# Завантаження та підготовка даних
df_lelitka = pd.read_csv("/kaggle/input/gidropost-lelitka/lelitka_cleaned.csv")
df_lelitka = df_lelitka[~df_lelitka['Рік'].isin(['Середнє', 'Найбільше', 'Найменше'])]
df_lelitka['Рік'] = df_lelitka['Рік'].astype(int)
for month in ['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
              'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень']:
    df_lelitka[month] = pd.to_numeric(df_lelitka[month], errors='coerce')

month_map = {
    'Січень': 1, 'Лютий': 2, 'Березень': 3, 'Квітень': 4, 'Травень': 5, 'Червень': 6,
    'Липень': 7, 'Серпень': 8, 'Вересень': 9, 'Жовтень': 10, 'Листопад': 11, 'Грудень': 12
}
df_long_lelitka = df_lelitka.melt(id_vars=["Рік"],
                                  value_vars=['Січень', 'Лютий', 'Березень', 'Квітень', 'Травен
ь', 'Червень',
                                              'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листоп
ад', 'Грудень'],
                                  var_name="Місяць_назва", value_name="Витрата")
df_long_lelitka["Місяць"] = df_long_lelitka["Місяць_назва"].map(month_map)
df_long_lelitka = df_long_lelitka.dropna(subset=["Витрата"])
df_long_lelitka["date"] = pd.to_datetime(dict(year=df_long_lelitka["Рік"], month=df_long_lelitka
["Місяць"], day=1))
df_long_lelitka = df_long_lelitka.sort_values("date").reset_index(drop=True)

# Створення часового ряду
ts_df_lelitka = df_long_lelitka.set_index('date')['Витрата']

# Вибираємо часовий ряд для Лелітки з 1980 року для тренування
sarima_series_lelitka = ts_df_lelitka["1980-01-01":"2012-12-31"].copy()

# Трансформація даних (логарифм)
sarima_series_lelitka_log = np.log(sarima_series_lelitka.replace(0, np.nan).dropna())

# Ручне налаштування SARIMA
model_sarima_lelitka = pm.ARIMA(order=(2,1,2), seasonal_order=(1,1,2,12), suppress_warnings=True)
model_sarima_lelitka.fit(sarima_series_lelitka_log)

# Прогноз на 24 місяці (2013-2014) з довірчими інтервалами
sarima_forecast_lelitka_log = model_sarima_lelitka.predict(n_periods=24, return_conf_int=True)
forecast_values_log = sarima_forecast_lelitka_log[0]
conf_int_lower_log = sarima_forecast_lelitka_log[1][:, 0]
conf_int_upper_log = sarima_forecast_lelitka_log[1][:, 1]
```

Рисунок 4.14 – Програмний код для прогнозування моделлю SARIMA (1 частина)

```
# Перетворення прогнозу назад із логарифма
forecast_values = np.exp(forecast_values_log)
conf_int_lower = np.exp(conf_int_lower_log)
conf_int_upper = np.exp(conf_int_upper_log)

# Створюємо DataFrame для прогнозу
forecast_index_lelitka = pd.date_range(start="2013-01-01", periods=24, freq="MS")
sarima_df_lelitka = pd.DataFrame({
    "date": forecast_index_lelitka,
    "forecast": forecast_values,
    "lower": conf_int_lower,
    "upper": conf_int_upper
})

# Обчислення R²
r2_sarima = r2_score(ts_df_lelitka["2013-01-01":"2013-12-31"], sarima_df_lelitka[sarima_df_lelitk
a["date"].dt.year == 2013]["forecast"])
print(f"SARIMA R² for 2013: {r2_sarima:.4f}")

# Побудова графіка
plt.figure(figsize=(12, 5))
plt.plot(ts_df_lelitka["2013-01-01":"2013-12-31"].index, ts_df_lelitka["2013-01-01":"2013-12-3
1"].values, label="Історичні дані 2013", marker="o", color="black")
plt.plot(sarima_df_lelitka["date"], sarima_df_lelitka["forecast"], label="SARIMA прогноз", linest
yle="--", color="blue")
plt.fill_between(sarima_df_lelitka["date"], sarima_df_lelitka["lower"], sarima_df_lelitka["uppe
r"],
                 alpha=0.3, color="lightblue", label="Довірчий інтервал")
plt.title("Прогноз витрат води (Лелітка): SARIMA, 2013–2014")
plt.xlabel("Дата")
plt.ylabel("Витрата (млн м³)")
plt.legend()
plt.grid(True)
plt.xlim(pd.Timestamp('2013-01-01'), pd.Timestamp('2014-12-31'))
plt.tight_layout()
plt.show()
print("Відображено: Прогноз витрат води (Лелітка): SARIMA, 2013–2014")
plt.close()
```

Рисунок 4.15 – Програмний код для прогнозування моделлю SARIMA (2 частина)

Результатом є графік, що дозволяє порівняти прогнозовані значення з реальними даними 2013 року для оцінки точності моделі SARIMA. Після відображення графіка виконано його закриття. На рисунку 4.16 показано отриманий графік.

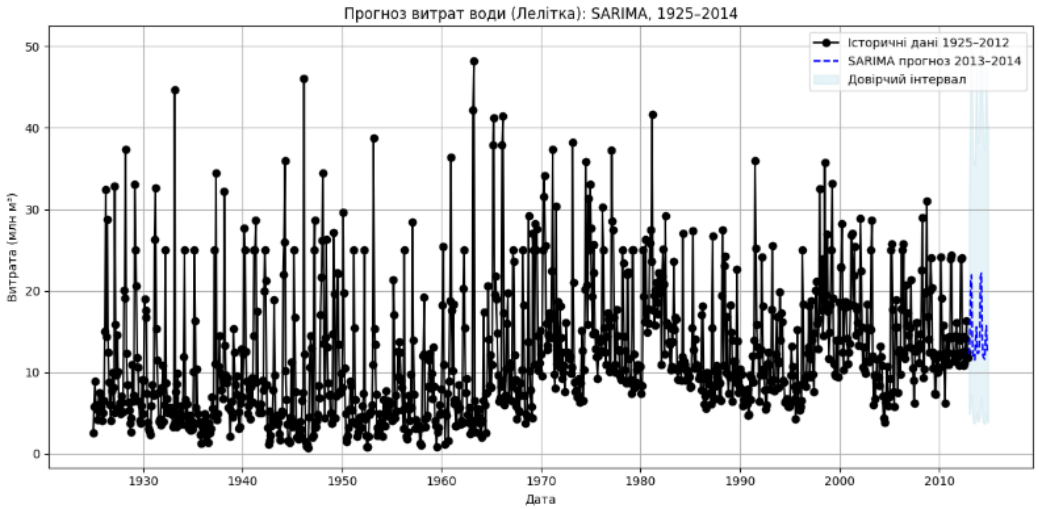

Рисунок 4.16 (А) – Графік прогнозу витрат води моделлю SARIMA для с. Лелітка (2013–2014)

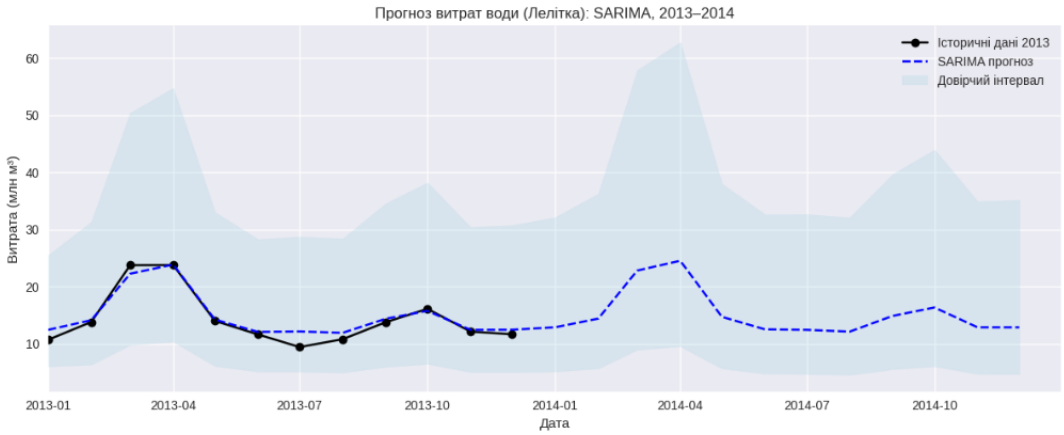

Рисунок 4.16 (Б) – Графік прогнозу витрат води моделлю SARIMA для с. Лелітка (2013–2014)

Для подальшого прогнозування витрат води (Wдеф) на ділянці с. Лелітка використано модель LSTM.

Спочатку підготовлено дані для моделі LSTM. Вибрано значення витрат води до кінця 2012 року, виконано масштабування за допомогою MinMaxScaler для нормалізації даних, а також створено послідовності з глибиною огляду 12 місяців для навчання. Побудовано модель LSTM із двома шарами (перший із 50 нейронами з активацією ReLU і return_sequences=True, другий із 50 нейронами), додано dropout (0.2) для уникнення перетренування, скомпілювано з функцією втрат MSE та оптимікатором Adam, і натреновано на 50 епох із валідаційною вибіркою 10%. Прогноз виконано на 24 місяці (2013–2014 роки) із подальшим зворотним масштабуванням результатів. На рисунку 4.17 і 4.18 зображено фрагменти програмного коду для підготовки даних, побудови та навчання моделі LSTM.

```python
# Завантаження та підготовка даних
df_lelitka = pd.read_csv("/kaggle/input/gidropost-lelitka/lelitka_cleaned.csv")
df_lelitka = df_lelitka[~df_lelitka['Рік'].isin(['Середнє', 'Найбільше', 'Найменше'])]
df_lelitka['Рік'] = df_lelitka['Рік'].astype(int)
for month in ['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
              'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень']:
    df_lelitka[month] = pd.to_numeric(df_lelitka[month], errors='coerce')

month_map = {
    'Січень': 1, 'Лютий': 2, 'Березень': 3, 'Квітень': 4, 'Травень': 5, 'Червень': 6,
    'Липень': 7, 'Серпень': 8, 'Вересень': 9, 'Жовтень': 10, 'Листопад': 11, 'Грудень': 12
}
df_long_lelitka = df_lelitka.melt(id_vars=["Рік"],
                                  value_vars=['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
                                              'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень'],
                                  var_name="Місяць_назва", value_name="Витрата")
df_long_lelitka["Місяць"] = df_long_lelitka["Місяць_назва"].map(month_map)
df_long_lelitka = df_long_lelitka.dropna(subset=["Витрата"])
df_long_lelitka["date"] = pd.to_datetime(dict(year=df_long_lelitka["Рік"], month=df_long_lelitka["Місяць"], day=1))
df_long_lelitka = df_long_lelitka.sort_values("date").reset_index(drop=True)

# Створення часового ряду
ts_df_lelitka = df_long_lelitka.set_index('date')['Витрата']

# Нормалізація даних
scaler = MinMaxScaler()
ts_df_lelitka_scaled = scaler.fit_transform(ts_df_lelitka.values.reshape(-1, 1))

# Створення послідовностей для LSTM
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i + seq_length])
        y.append(data[i + seq_length])
    return np.array(X), np.array(y)

seq_length = 12  # Ковзне вікно в 12 місяців
X, y = create_sequences(ts_df_lelitka_scaled, seq_length)

# Розділення на тренувальну та тестову вибірки
train_size = len(ts_df_lelitka) - 24  # 24 місяці (2013-2014) для тесту
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Побудова моделі LSTM
model = Sequential([
    LSTM(50, activation='relu', input_shape=(seq_length, 1), return_sequences=True),
    Dropout(0.2),
    LSTM(50, activation='relu'),
    Dropout(0.2),
    Dense(1)
])
```

Рисунок 4.17 – Програмний код для підготовки даних та побудови моделі LSTM (1 частина)

```python
model.compile(optimizer=Adam(learning_rate=0.001), loss='mse')

# Тренування моделі
model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.1, verbose=0)

# Прогноз
last_sequence = ts_df_lelitka_scaled[-seq_length:]
future_predictions = []
current_sequence = last_sequence

for _ in range(24):  # Прогноз на 24 місяці
    next_pred = model.predict(current_sequence.reshape(1, seq_length, 1), verbose=0)
    future_predictions.append(next_pred[0, 0])
    current_sequence = np.roll(current_sequence, -1)
    current_sequence[-1] = next_pred[0, 0]

future_predictions = np.array(future_predictions)
forecast_values = scaler.inverse_transform(future_predictions.reshape(-1, 1))
historical_values = scaler.inverse_transform(y_test.reshape(-1, 1))

# Обчислення R²
r2_lstm = r2_score(historical_values[:12], forecast_values[:12])  # Порівняння лише за 2013
print(f"LSTM R² for 2013: {r2_lstm:.4f}")

# Побудова графіка
forecast_index_lelitka = pd.date_range(start="2013-01-01", periods=24, freq="MS")
plt.figure(figsize=(12, 5))
plt.plot(pd.date_range(start="2013-01-01", periods=12, freq="MS"), historical_values[:12], label="Історичні дані 2013", marker="o", color="black")
plt.plot(forecast_index_lelitka, forecast_values, label="LSTM прогноз", linestyle="--", color="blue")
plt.title("Прогноз витрат води (Лелітка): LSTM, 2013–2014")
plt.xlabel("Дата")
plt.ylabel("Витрата (млн м³)")
plt.legend()
plt.grid(True)
plt.xlim(pd.Timestamp('2013-01-01'), pd.Timestamp('2014-12-31'))
plt.tight_layout()
plt.show()
print("Відображено: Прогноз витрат води (Лелітка): LSTM, 2013–2014")
plt.close()
```

Рисунок 4.18 – Програмний код для підготовки даних та побудови моделі LSTM (2 частина)

Для візуалізації результатів побудовано графік, на якому показано історичні дані за 2013 рік та прогнозовані значення за 2013–2014 роки. На рисунку 4.19 представлено графік, отриманий в результаті.

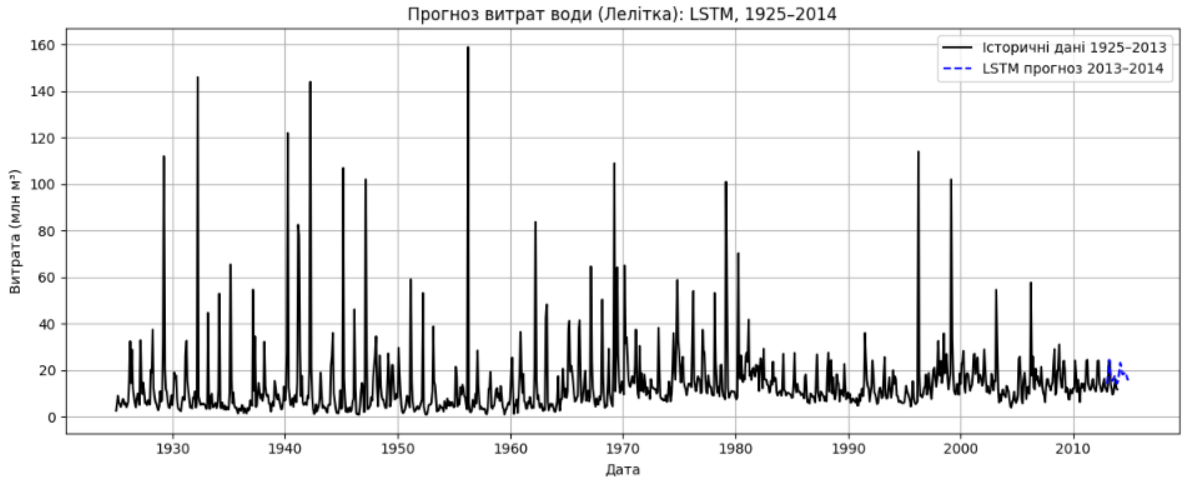

Рисунок 4.19 (А) – Графік прогнозу витрат води моделлю LSTM для с. Лелітка (2013–2014)

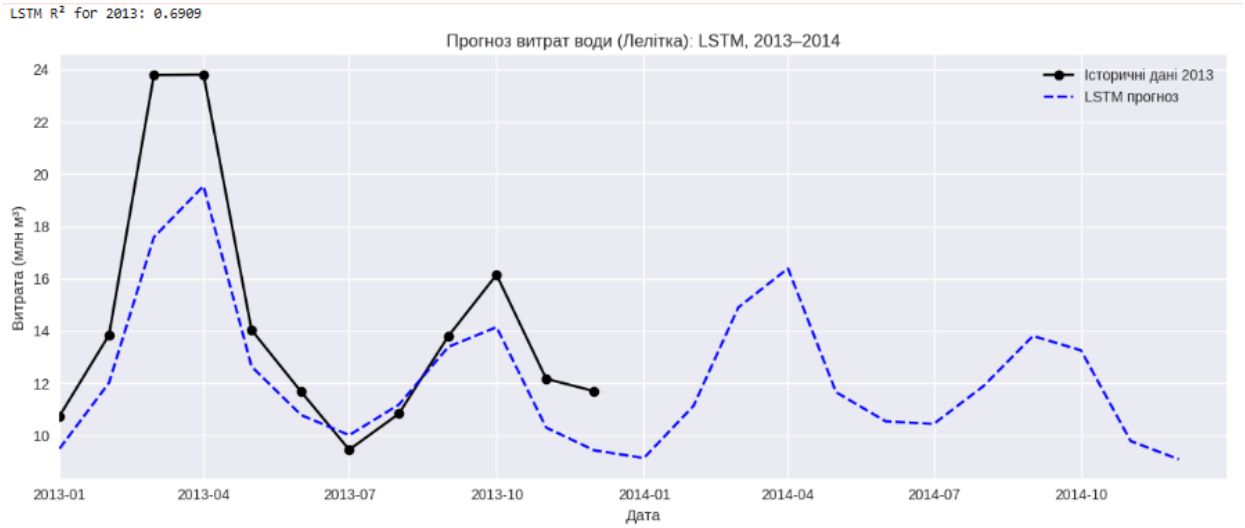

Рисунок 4.19 (Б) – Графік прогнозу витрат води моделлю LSTM для с. Лелітка (2013–2014)

Оцінка моделей та аналіз результатів

Для оцінки точності моделей прогнозування витрат води (Wдеф) на ділянці с. Лелітка виконано порівняння їхніх прогнозів за 2013 рік із реальними даними. Спочатку витягнуто історичні значення витрат за 2013 рік, а створено функцію

для обчислення метрик: MAE, MSE, RMSE, MAPE та R^2. Для кожної моделі (Prophet, SARIMA, LSTM) обчислено метрики на основі прогнозів за 2013 рік, результати зведено у таблицю та збережено у форматі Markdown. На рисунку 4.20 зображено програмний код для обчислення метрик і створення таблиці результатів.

```
true_values_2013_lelitka = ts_df_lelitka["2013-01-01":"2013-12-31"].values

# Функція для обчислення метрик
def calculate_metrics(true, predicted):
    if len(true) != len(predicted):
        raise ValueError(f"Довжина прогнозу ({len(predicted)}) не відповідає довжині реальних даних ({len(true)})")
    mae = mean_absolute_error(true, predicted)
    mse = mean_squared_error(true, predicted)
    rmse = math.sqrt(mse)
    mape = np.mean(np.abs((true - predicted) / np.where(true == 0, 1e-10, true))) * 100
    r2 = r2_score(true, predicted)
    return mae, mse, rmse, mape, r2

# Prophet: Отримуємо прогнози за 2013 рік
prophet_forecast_2013 = forecast[forecast["ds"].dt.year == 2013]["yhat"].values
prophet_mae, prophet_mse, prophet_rmse, prophet_mape, prophet_r2 = calculate_metrics(true_values_2013_lelitka, prophet_forecast_2013)

# SARIMA: Отримуємо прогнози за 2013 рік
sarima_forecast_2013 = sarima_df_lelitka[sarima_df_lelitka["date"].dt.year == 2013]["forecast"].values
sarima_mae, sarima_mse, sarima_rmse, sarima_mape, sarima_r2 = calculate_metrics(true_values_2013_lelitka, sarima_forecast_2013)

# LSTM: Отримуємо прогнози за 2013 рік
lstm_forecast_2013 = forecast_values[:12].flatten()  # Беремо перші 12 місяців (2013)
lstm_mae, lstm_mse, lstm_rmse, lstm_mape, lstm_r2 = calculate_metrics(true_values_2013_lelitka, lstm_forecast_2013)

# Перевірка довжин
print("Довжини прогнозів:", len(prophet_forecast_2013), len(sarima_forecast_2013), len(lstm_forecast_2013))
print("Реальні дані Лелітка (2013):", true_values_2013_lelitka[:5])

# Створюємо таблицю з результатами
results_lelitka = {
    "Модель": ["Prophet", "SARIMA", "LSTM"],
    "MAE": [prophet_mae, sarima_mae, lstm_mae],
    "MSE": [prophet_mse, sarima_mse, lstm_mse],
    "RMSE": [prophet_rmse, sarima_rmse, lstm_rmse],
    "MAPE (%)": [prophet_mape, sarima_mape, lstm_mape],
    "R²": [prophet_r2, sarima_r2, lstm_r2]
}
results_df_lelitka = pd.DataFrame(results_lelitka)
results_df_lelitka[["MAE", "MSE", "RMSE", "MAPE (%)", "R²"]] = results_df_lelitka[["MAE", "MSE", "RMSE", "MAPE (%)", "R²"]].round(2)

# Виведення і збереження таблиці в Markdown
markdown_table = results_df_lelitka.to_markdown(index=False)
markdown_output = "# Результати порівняння точності всіх моделей прогнозування ділянки с. Лелітка\n\n" + markdown_table
with open("model_comparison_lelitka.md", "w") as f:
    f.write(markdown_output)
print(markdown_output)
```

Рисунок 4.20 – Програмний код для обчислення метрик і створення таблиці результатів

Результатом виконання коду є таблиця, що містить значення метрик для всіх моделей, що дозволяє порівняти їхню точність. На рисунку 4.21 представлено таблицю з результатами порівняння моделей.

```
Довжини прогнозів: 12 12 12
Реальні дані Лелітка (2013): [10.74097911 13.84526068 23.7838969  23.80066444 14.03609557]
# Результати порівняння точності всіх моделей прогнозування ділянки с. Лелітка

| Модель   |   MAE |   MSE |   RMSE |   MAPE (%) |   R² |
|:---------|------:|------:|-------:|-----------:|-----:|
| Prophet  |  1.08 |  1.72 |   1.31 |       8.04 | 0.92 |
| SARIMA   |  0.85 |  1.29 |   1.14 |       7.12 | 0.94 |
| LSTM     |  3.65 | 14.84 |   3.85 |      28.18 | 0.29 |
```

Рисунок 4.21 – Таблиця порівняння точності моделей для гідропоста с. Лелітка

Далі проведено аналіз історичних і прогнозованих витрат води. Спочатку витягнуто історичні дані за 2013 рік та сформовано таблицю з витратами по місяцях. Потім для кожної моделі отримано прогнози на 2014 рік, об’єднано їх у єдину таблицю та збережено у форматі Markdown із назвами місяців англійською мовою. На рисунку 4.22 зображено програмний код для створення таблиць історичних і прогнозованих даних.

```python
historical_2013_lelitka = ts_df_lelitka["2013-01-01":"2013-12-31"].reset_index()
if len(historical_2013_lelitka) != 12:
    raise ValueError(f"Очікується 12 місяців за 2013 рік, але отримано {len(historical_2013_lelitka)} записів.")
historical_2013_lelitka["Місяць"] = historical_2013_lelitka["date"].dt.strftime("%B")  # Назви місяців англійською
historical_2013_table = historical_2013_lelitka[["Місяць", "Витрата"]].copy()
historical_2013_table["Витрата"] = historical_2013_table["Витрата"].round(2)

# Виведення таблиці у Markdown
historical_markdown = "# Історичні витрати води за 2013 рік (Лелітка)\n\n" + historical_2013_table.to_markdown(index=False)
print(historical_markdown)
with open("historical_2013_lelitka.md", "w") as f:
    f.write(historical_markdown)

# --- Прогнозовані витрати води за 2014 рік (всі моделі) ---
# Прогнози за 2014 рік для всіх моделей
prophet_forecast_2014 = forecast[forecast["ds"].dt.year == 2014][["ds", "yhat"]].copy()
prophet_forecast_2014.rename(columns={"ds": "date", "yhat": "Prophet"}, inplace=True)
if len(prophet_forecast_2014) != 12:
    raise ValueError(f"Очікується 12 місяців прогнозів Prophet за 2014 рік, але отримано {len(prophet_forecast_2014)} записів.")

sarima_forecast_2014 = sarima_df_lelitka[sarima_df_lelitka["date"].dt.year == 2014][["date", "forecast"]].copy()
sarima_forecast_2014.rename(columns={"forecast": "SARIMA"}, inplace=True)
if len(sarima_forecast_2014) != 12:
    raise ValueError(f"Очікується 12 місяців прогнозів SARIMA за 2014 рік, але отримано {len(sarima_forecast_2014)} записів.")

# Для LSTM: створюємо DataFrame із forecast_values для 2014 року (місяці 13-24)
forecast_index_2014 = pd.date_range(start="2014-01-01", periods=12, freq="MS")
if len(forecast_values) < 24:
    raise ValueError(f"Очікується щонайменше 24 значення в forecast_values, але отримано {len(forecast_values)}.")
lstm_forecast_2014 = pd.DataFrame({
    "date": forecast_index_2014,
    "LSTM": forecast_values[12:24].flatten()  # Беремо прогнози для 2014 року
})
if len(lstm_forecast_2014) != 12:
    raise ValueError(f"Очікується 12 місяців прогнозів LSTM за 2014 рік, але отримано {len(lstm_forecast_2014)} записів.")

# Об'єднуємо прогнози всіх моделей
forecast_2014_lelitka = prophet_forecast_2014.merge(sarima_forecast_2014, on="date").merge(lstm_forecast_2014, on="date")
if len(forecast_2014_lelitka) != 12:
    raise ValueError(f"Очікується 12 місяців у фінальній таблиці, але отримано {len(forecast_2014_lelitka)} записів.")
forecast_2014_lelitka["Місяць"] = forecast_2014_lelitka["date"].dt.strftime("%B")  # Назви місяців англійською
forecast_2014_table = forecast_2014_lelitka[["Місяць", "Prophet", "SARIMA", "LSTM"]].copy()
forecast_2014_table[["Prophet", "SARIMA", "LSTM"]] = forecast_2014_table[["Prophet", "SARIMA", "LSTM"]].round(2)

# Виведення таблиці у Markdown
forecast_markdown = "# Прогнозовані витрати води за 2014 рік (Лелітка)\n\n" + forecast_2014_table.to_markdown(index=False)
print(forecast_markdown)
with open("forecast_2014_lelitka.md", "w") as f:
    f.write(forecast_markdown)
```

Рисунок 4.22 – Програмний код для створення таблиць історичних і прогнозованих витрат

Результатом є дві таблиці: одна з історичними витратами за 2013 рік, інша з прогнозованими значеннями на 2014 рік для всіх моделей. На рисунку 4.23 представлено ці таблиці.

Історичні витрати води за 2013 рік (Лелітка)

Місяць	Витрата
January	10.74
February	13.85
March	23.78
April	23.8
May	14.04
June	11.67
July	9.46
August	10.84
September	13.81
October	16.15
November	12.17
December	11.7

Прогнозовані витрати води за 2014 рік (Лелітка)

Місяць	Prophet	SARIMA	LSTM
January	15.92	12.92	16.23
February	17.08	14.44	17.14
March	16.43	22.84	18.08
April	14.78	24.56	18.11
May	12.83	14.75	17.31
June	12.32	12.56	16.96
July	13.48	12.48	17.15
August	12	12.16	17.63
September	12.79	14.92	18.07
October	13.77	16.38	17.91
November	13.67	12.92	17.33
December	13.78	12.91	17.29

Рисунок 4.23 – Таблиці витрат води за 2013-2014 роки для с. Лелітка

4.3 Прогнозування витрат води по гідропосту с.Тростянчик

Підготовка та аналіз даних для Тростянчика.

Процес прогнозування витрат води (Wдеф) для водогосподарської ділянки с. Тростянчик розпочато з підготовки даних. Завантажено очищені дані з CSV-файлу та перетворено їх із широкого формату у довгий за допомогою функції melt. Створено мапу місяців для перетворення назв місяців у числові значення, видалено нечислові записи в колонці "Рік", а також сформовано колонку з датами у форматі datetime. Дані відсортовано за датами, створено часовий ряд із частотою "MS" (початок місяця) та заповнено пропуски методом прямого заповнення. На рисунку 4.24 зображено програмний код для завантаження, обробки та підготовки часового ряду.

```python
# ✅ Завантаження даних для Тростянчика
df_trost = pd.read_csv("/kaggle/input/gidropost-trostyanchik/trostyanchik_cleaned.csv")
df_trost.head()

# 📅 Перетворимо з wide в long формат
month_columns = ['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
                 'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень']

# 🔄 Melt
trost_long = df_trost.melt(id_vars=["Рік"], value_vars=month_columns,
                           var_name="Місяць_назва", value_name="Витрата")

# 📌 Мапа місяців у номер
month_map = {
    'Січень': 1, 'Лютий': 2, 'Березень': 3, 'Квітень': 4,
    'Травень': 5, 'Червень': 6, 'Липень': 7, 'Серпень': 8,
    'Вересень': 9, 'Жовтень': 10, 'Листопад': 11, 'Грудень': 12
}
trost_long["Місяць"] = trost_long["Місяць_назва"].map(month_map)

# 🔍 Видаляємо нечислові значення в колонці "Рік"
trost_long = trost_long[trost_long["Рік"].astype(str).str.isnumeric()]

# 🔢 Перетворюємо типи
trost_long["Рік"] = trost_long["Рік"].astype(int)
trost_long["Місяць"] = trost_long["Місяць"].astype(int)

# 🗓️ Формуємо колонку з датою
trost_long["date"] = pd.to_datetime(dict(year=trost_long["Рік"], month=trost_long["Місяць"], day=1))

# 🗂️ Сортуємо
trost_long = trost_long.sort_values("date")

# Формуємо time series
ts_df = trost_long[["date", "Витрата"]].copy()
ts_df = ts_df.set_index("date")
ts_df = ts_df.asfreq("MS")  # MS = Month Start

# Заповнюємо пропуски
ts_df = ts_df.ffill()

# Перевірка на пропуски
print("Кількість пропусків:", ts_df.isna().sum().values[0])

# Побудова графіка для перевірки
ts_df.plot(figsize=(12, 4), title="Часовий ряд витрат води (Тростянчик)")

# Гістограма
ts_df["Витрата"].plot(kind="hist", bins=50, figsize=(10, 4), title="Гістограма витрат води (Тростянчик)")
plt.xlabel("Витрата")
plt.grid(True)
plt.show()

# Ковзне середнє
ts_df["Витрата"].rolling(window=6).mean().plot(figsize=(12, 4), title="Ковзне середнє витрат (6 міс)")
plt.xlabel("Дата")
plt.ylabel("Витрата")
plt.grid(True)
plt.show()
```

Рисунок 4.24 – Програмний код для підготовки часового ряду для с. Тростянчик

Для аналізу даних побудовано графік часового ряду витрат води, щоб оцінити загальні тренди та коливання. На рисунку 4.25 представлено графік часового ряду, отриманий в результаті.

Рисунок 4.25 – Графік часового ряду витрат води для с. Тростянчик

Для виявлення сезонних закономірностей додано стовпець "Місяць" та побудовано boxplot, який показує розподіл витрат води по місяцях. На рисунку 4.26 представлено boxplot витрат води по місяцях.

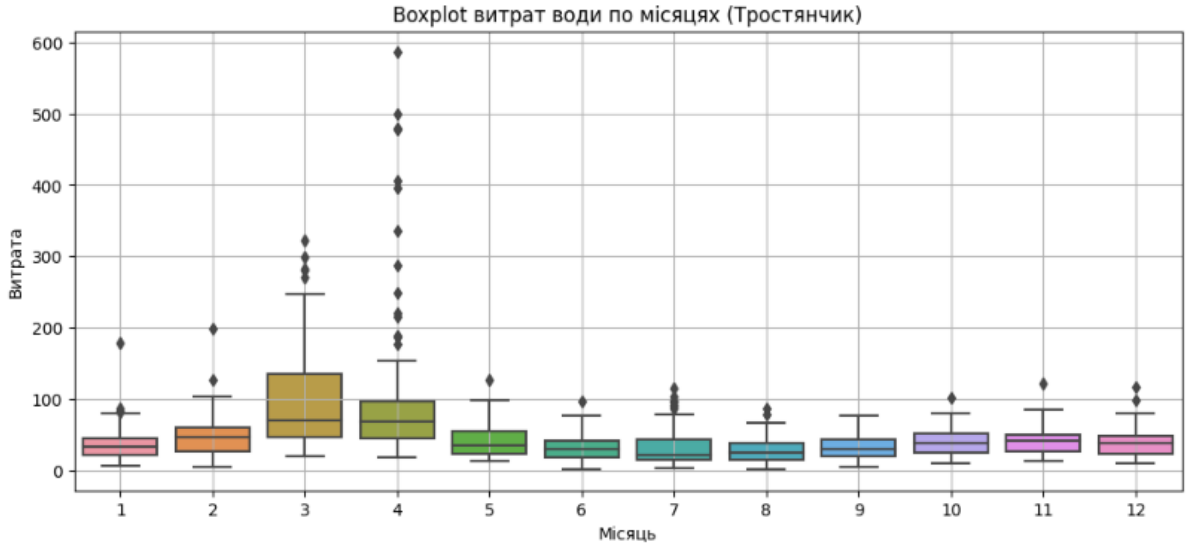

Рисунок 4.26 – Boxplot витрат води по місяцях для с. Тростянчик

Прогнозування з використанням моделей Prophet та SARIMA

Для прогнозування витрат води (Wдеф) на ділянці с. Тростянчик використано моделі Prophet і SARIMA.

Спочатку підготовлено дані для моделі Prophet. Створено копію часового ряду, виконано логарифмування значень витрат із використанням np.log1p для

стабілізації дисперсії, а також додано регресори (lag1, lag2, lag3, rolling_mean_3, spring_peak), які враховують історичні лаги та сезонні піки (березень-квітень). Дані відформатовано у структуру з колонками "ds" (дати) і "y" (значення), сумісну з Prophet. Модель ініціалізовано з адитивним режимом сезонності, увімкненою річною сезонністю, налаштовано параметри (changepoint_prior_scale=1.5, n_changepoints=25, seasonality_prior_scale=40.0) та додано кастомну сезонність із періодом 12 і fourier_order=40. Модель натреновано, а прогноз виконано на 24 місяці (2013–2014 роки) із ітеративним оновленням лагів і ковзного середнього для 2014 року. Результати зворотньо трансформовано з логарифмічного формату за допомогою np.expm1. На рисунках 4.27 і 4.28 зображено фрагменти програмного коду для підготовки даних, навчання моделі Prophet та прогнозування.

```
ts_df = pd.read_csv("/kaggle/input/gidropost-trostyanchik/trostyanchik_cleaned.csv")
# Перетворення в довгий формат
# Фільтрація нечислових значень у колонці 'Рік'
ts_df = ts_df[pd.to_numeric(ts_df['Рік'], errors='coerce').notna()]
ts_df['Рік'] = ts_df['Рік'].astype(int)
# Перетворення в довгий формат
month_map = {
    'Січень': 1, 'Лютий': 2, 'Березень': 3, 'Квітень': 4, 'Травень': 5, 'Червень': 6,
    'Липень': 7, 'Серпень': 8, 'Вересень': 9, 'Жовтень': 10, 'Листопад': 11, 'Грудень': 12
}
ts_df_long = ts_df.melt(
    id_vars=["Рік"],
    value_vars=['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
                'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень'],
    var_name="Місяць_назва",
    value_name="Витрата"
)
ts_df_long["Місяць"] = ts_df_long["Місяць_назва"].map(month_map)
ts_df_long = ts_df_long.dropna(subset=["Витрата"])
ts_df_long["ds"] = pd.to_datetime(dict(year=ts_df_long["Рік"], month=ts_df_long["Місяць"], day=1))
ts_df_long = ts_df_long.sort_values("ds").reset_index(drop=True)
# Додавання ознак
ts_df_long['lag1'] = ts_df_long['Витрата'].shift(1)
ts_df_long['lag2'] = ts_df_long['Витрата'].shift(2)
ts_df_long['lag3'] = ts_df_long['Витрата'].shift(3)
ts_df_long['rolling_mean_3'] = ts_df_long['Витрата'].rolling(window=3, min_periods=1).mean()
ts_df_long['spring_peak'] = ((ts_df_long["Місяць"].isin([3, 4]))).astype(int)
# Виправлення пропущених значень
ts_df_long['lag1'] = ts_df_long['lag1'].fillna(0)
ts_df_long['lag2'] = ts_df_long['lag2'].fillna(0)
ts_df_long['lag3'] = ts_df_long['lag3'].fillna(0)
ts_df_long['rolling_mean_3'] = ts_df_long['rolling_mean_3'].fillna(ts_df_long['Витрата'])
# Розбиття на тренувальні та тестові дані
train_data = ts_df_long[ts_df_long["ds"].dt.year < 2013].copy()
test_data_2013 = ts_df_long[ts_df_long["ds"].dt.year == 2013].copy()
# Формат для Prophet
prophet_df = pd.DataFrame({
    "ds": train_data["ds"],
    "y": np.log1p(train_data["Витрата"]),
    "lag1": np.log1p(train_data["lag1"]),
    "lag2": np.log1p(train_data["lag2"]),
    "lag3": np.log1p(train_data["lag3"]),
    "rolling_mean_3": np.log1p(train_data["rolling_mean_3"]),
    "spring_peak": train_data["spring_peak"]
})
# Модель Prophet
model_trostianchyk = Prophet(
    growth='linear',
    yearly_seasonality=True,
    seasonality_mode='additive',
    changepoint_prior_scale=1.5,
    n_changepoints=25,
    seasonality_prior_scale=40.0
)
model_trostianchyk.add_regressor('lag1')
model_trostianchyk.add_regressor('lag2')
model_trostianchyk.add_regressor('lag3')
model_trostianchyk.add_regressor('rolling_mean_3')
model_trostianchyk.add_regressor('spring_peak')
model_trostianchyk.add_seasonality(name='yearly_custom', period=12, fourier_order=40)
model_trostianchyk.fit(prophet_df)
```

Рисунок 4.27 – Програмний код для підготовки даних та прогнозування моделлю Prophet (1 частина)

```
future_dates = pd.date_range(start='2013-01-01', end='2014-12-01', freq='MS')
future_df = pd.DataFrame({'ds': future_dates})
future_df['spring_peak'] = ((future_df['ds'].dt.month.isin([3, 4]))).astype(int)
# Ініціалізація лагів і ковзного середнього
last_train_data = train_data.iloc[-1]
future_df.loc[future_df.index[0], 'lag1'] = np.log1p(last_train_data['Витрата'])
future_df.loc[future_df.index[0], 'lag2'] = np.log1p(last_train_data['Витрата'])
future_df.loc[future_df.index[0], 'lag3'] = np.log1p(last_train_data['Витрата'])
future_df.loc[future_df.index[0], 'rolling_mean_3'] = np.log1p(last_train_data['rolling_mean_3'])
# Ітеративний прогноз із оновленням лагів
forecast_list = []
lag_buffer = [np.log1p(last_train_data['Витрата'])] * 3
rolling_buffer = [np.log1p(last_train_data['rolling_mean_3'])]
for i, date in enumerate(future_dates):
    temp_df = future_df[future_df['ds'] == date].copy()
    # Для 2013 року використовуємо історичні лаги, якщо доступні
    if date in test_data_2013['ds'].values:
        idx = test_data_2013[test_data_2013['ds'] == date].index[0]
        temp_df['lag1'] = np.log1p(test_data_2013.loc[idx, 'lag1'])
        temp_df['lag2'] = np.log1p(test_data_2013.loc[idx, 'lag2'])
        temp_df['lag3'] = np.log1p(test_data_2013.loc[idx, 'lag3'])
        temp_df['rolling_mean_3'] = np.log1p(test_data_2013.loc[idx, 'rolling_mean_3'])
    else:  # Для 2014 року використовуємо оновлені лаги
        temp_df['lag1'] = lag_buffer[0]
        temp_df['lag2'] = lag_buffer[1]
        temp_df['lag3'] = lag_buffer[2]
        temp_df['rolling_mean_3'] = np.mean(rolling_buffer) if len(rolling_buffer) >= 1 else np.log1p(last_train_data['rolling_mean_3'])
    temp_forecast = model_trostianchyk.predict(temp_df)
    forecast_list.append(temp_forecast)
    # Оновлюємо буфери для наступного кроку (лише для 2014 року)
    if date not in test_data_2013['ds'].values and i > 0:
        yhat = np.expm1(temp_forecast['yhat'].iloc[0])
        lag_buffer[2] = lag_buffer[1]
        lag_buffer[1] = lag_buffer[0]
        lag_buffer[0] = np.log1p(yhat)
        rolling_buffer.append(np.log1p(yhat))
        if len(rolling_buffer) > 3:
            rolling_buffer.pop(0)
# Об'єднуємо прогнози
forecast_trostianchyk = pd.concat(forecast_list)
forecast_trostianchyk["yhat"] = np.expm1(forecast_trostianchyk["yhat"])
forecast_trostianchyk["yhat_lower"] = np.expm1(forecast_trostianchyk["yhat_lower"])
forecast_trostianchyk["yhat_upper"] = np.expm1(forecast_trostianchyk["yhat_upper"])
# Оцінка R² за 2013 рік
actual_2013 = test_data_2013["Витрата"]
pred_2013 = forecast_trostianchyk[forecast_trostianchyk["ds"].dt.year == 2013]["yhat"]
r2_trostianchyk = r2_score(actual_2013, pred_2013)
print(f"Trostianchyk Prophet R² for 2013: {r2_trostianchyk:.4f}")
# Графік
plt.figure(figsize=(12, 6))
plt.plot(test_data_2013["ds"], actual_2013, label="Історичні дані 2013", marker="o", color="black")
plt.plot(forecast_trostianchyk["ds"], forecast_trostianchyk["yhat"], label="Прогноз Prophet", linestyle="--", color="blue")
plt.fill_between(forecast_trostianchyk["ds"], forecast_trostianchyk["yhat_lower"], forecast_trostianchyk["yhat_upper"], alpha=0.3, color="lightblue", label="Довірчий інтервал")
plt.title("Прогноз витрат води (Тростянчик): Prophet, 2013-2014")
plt.xlabel("Дата")
plt.ylabel("Витрата (млн м³)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Рисунок 4.28 – Програмний код для підготовки даних та прогнозування моделлю Prophet (2 частина)

Для візуалізації результатів побудовано графік, на якому показано прогнозовані значення витрат води з довірчим інтервалом, а також історичні дані за 2013 рік. На рисунку 4.29 представлено графік, отриманий в результаті.

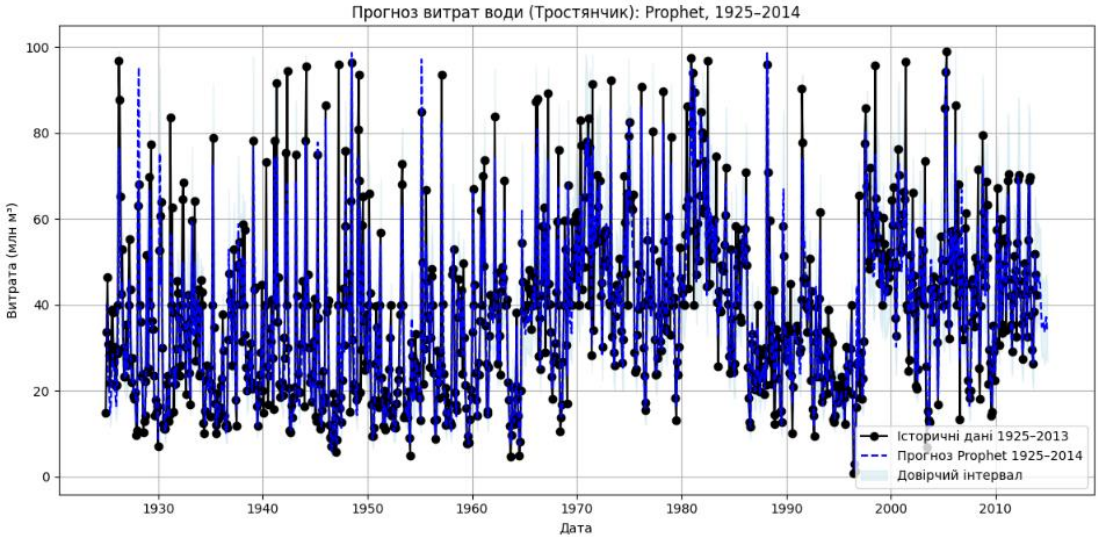

Рисунок 4.29 (А) – Графік прогнозу витрат води моделлю Prophet для с. Тростянчик (2013–2014)

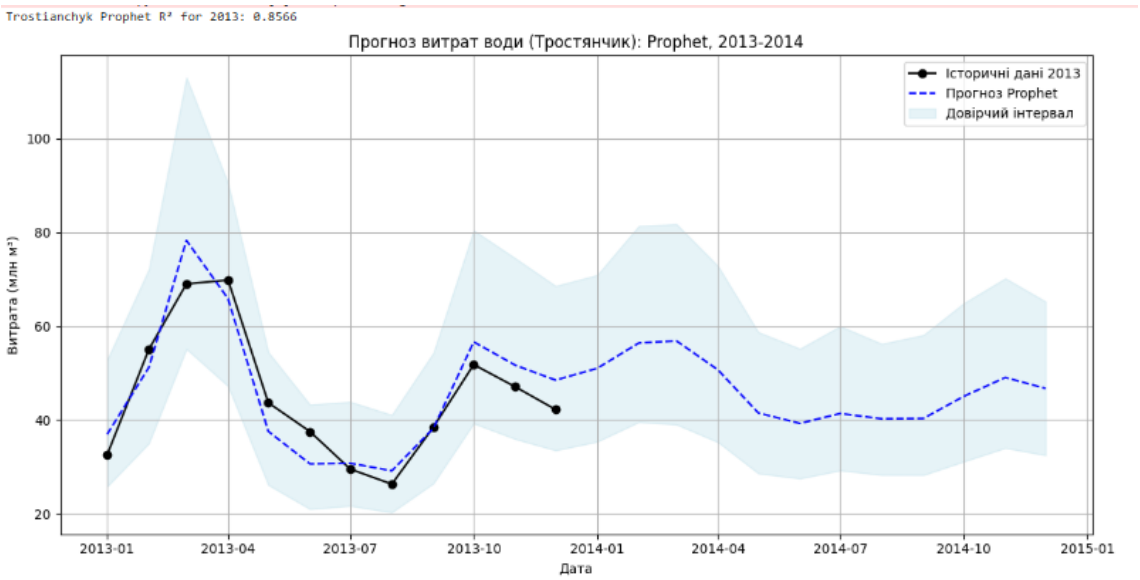

Рисунок 4.29 (Б) – Графік прогнозу витрат води моделлю Prophet для с. Тростянчик (2013–2014)

Далі виконано прогнозування з використанням моделі SARIMA. Використано часовий ряд до кінця 2012 року, застосовано логарифмічну трансформацію значень із np.log для стабілізації дисперсії та вручну налаштовано модель SARIMA з параметрами order=(2,1,2) і seasonal_order=(1,1,2,12). Модель натреновано, а прогноз виконано на 24 місяці (2013–2014 роки) із розрахунком довірчих інтервалів. Результати зворотньо трансформовано з логарифмічного формату за допомогою np.exp і збережено у

DataFrame із відповідними датами. На рисунках 4.30 і 4.31 зображені фрагменти програмного коду для навчання моделі SARIMA та прогнозування.

```
# Завантаження та підготовка даних
df_trostianchyk = pd.read_csv("/kaggle/input/gidropost-trostyanchik/trostyanchik_cleaned.csv")  # Оновіть шлях до файлу
df_trostianchyk = df_trostianchyk[~df_trostianchyk['Рік'].isin(['Середнє', 'Найбільше', 'Найменше'])]
df_trostianchyk['Рік'] = df_trostianchyk['Рік'].astype(int)
for month in ['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
              'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень']:
    df_trostianchyk[month] = pd.to_numeric(df_trostianchyk[month], errors='coerce')

# Перетворення в довгий формат
month_map = {
    'Січень': 1, 'Лютий': 2, 'Березень': 3, 'Квітень': 4, 'Травень': 5, 'Червень': 6,
    'Липень': 7, 'Серпень': 8, 'Вересень': 9, 'Жовтень': 10, 'Листопад': 11, 'Грудень': 12
}
ts_df_long = df_trostianchyk.melt(
    id_vars=["Рік"],
    value_vars=['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
                'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень'],
    var_name="Місяць_назва",
    value_name="Витрата"
)
ts_df_long['Рік'] = ts_df_long['Рік'].astype(int)
ts_df_long["Місяць"] = ts_df_long["Місяць_назва"].map(month_map)
ts_df_long = ts_df_long.dropna(subset=["Витрата"])
ts_df_long["date"] = pd.to_datetime(dict(year=ts_df_long["Рік"], month=ts_df_long["Місяць"], day=1))
ts_df_long = ts_df_long.sort_values("date").reset_index(drop=True)

# Створення часового ряду
ts_df_trostianchyk = ts_df_long.set_index('date')['Витрата']

# Вибираємо часовий ряд з 1980 року для тренування
sarima_series_trostianchyk = ts_df_trostianchyk["1980-01-01":"2012-12-31"].copy()

# Трансформація даних (логарифм)
sarima_series_trostianchyk_log = np.log(sarima_series_trostianchyk.replace(0, np.nan).dropna())

# Ручне налаштування SARIMA
model_sarima_trostianchyk = pm.ARIMA(order=(2,1,2), seasonal_order=(1,1,2,12), suppress_warnings=True)
model_sarima_trostianchyk.fit(sarima_series_trostianchyk_log)

# Прогноз на 24 місяці (2013-2014) з довірчими інтервалами
sarima_forecast_trostianchyk_log = model_sarima_trostianchyk.predict(n_periods=24, return_conf_int=True)
forecast_values_log = sarima_forecast_trostianchyk_log[0]
conf_int_lower_log = sarima_forecast_trostianchyk_log[1][:, 0]
conf_int_upper_log = sarima_forecast_trostianchyk_log[1][:, 1]

# Перетворення прогнозу назад із логарифма
forecast_values = np.exp(forecast_values_log)
conf_int_lower = np.exp(conf_int_lower_log)
conf_int_upper = np.exp(conf_int_upper_log)

# Створюємо DataFrame для прогнозу
forecast_index_trostianchyk = pd.date_range(start="2013-01-01", periods=24, freq="MS")
sarima_df_trostianchyk = pd.DataFrame({
    "date": forecast_index_trostianchyk,
    "forecast": forecast_values,
    "lower": conf_int_lower,
    "upper": conf_int_upper
})
```

Рисунок 4.30 – Програмний код для підготовки даних та прогнозування моделлю SARIMA (1 частина)

```python
# Обчислення R²
r2_sarima = r2_score(ts_df_trostianchyk["2013-01-01":"2013-12-31"], sarima_df_trostianchyk[sarima_df_trostianchyk["date"].dt.year == 2013]["forecast"])
print(f"SARIMA R² for 2013: {r2_sarima:.4f}")

# Побудова графіка
plt.figure(figsize=(12, 5))
plt.plot(ts_df_trostianchyk["2013-01-01":"2013-12-31"].index, ts_df_trostianchyk["2013-01-01":"2013-12-31"].values, label="Історичні дані 2013", marker="o", color="black")
plt.plot(sarima_df_trostianchyk["date"], sarima_df_trostianchyk["forecast"], label="SARIMA прогноз", linestyle="--", color="blue")
plt.fill_between(sarima_df_trostianchyk["date"], sarima_df_trostianchyk["lower"], sarima_df_trostianchyk["upper"],
                 alpha=0.3, color="lightblue", label="Довірчий інтервал")
plt.title("Прогноз витрат води (Тростянчик): SARIMA, 2013–2014")
plt.xlabel("Дата")
plt.ylabel("Витрата (млн м³)")
plt.legend()
plt.grid(True)
plt.xlim(pd.Timestamp('2013-01-01'), pd.Timestamp('2014-12-31'))
plt.tight_layout()
plt.show()
```

Рисунок 4.31 – Програмний код для підготовки даних та прогнозування моделлю SARIMA (2 частина)

Для візуалізації результатів побудовано графік, на якому показано історичні дані за 2013 рік та прогнозовані значення за 2013–2014 роки. На рисунку 4.32 представлено графік, отриманий в результаті.

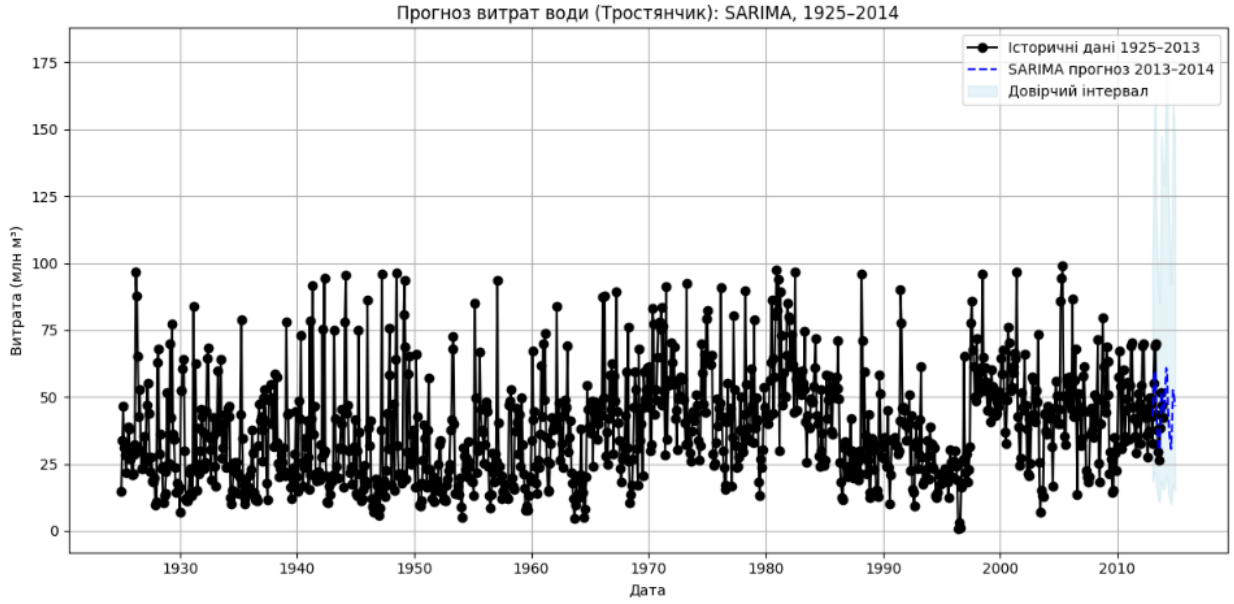

Рисунок 4.32 (А) – Графік прогнозу витрат води моделлю SARIMA для с. Тростянчик (2013–2014)

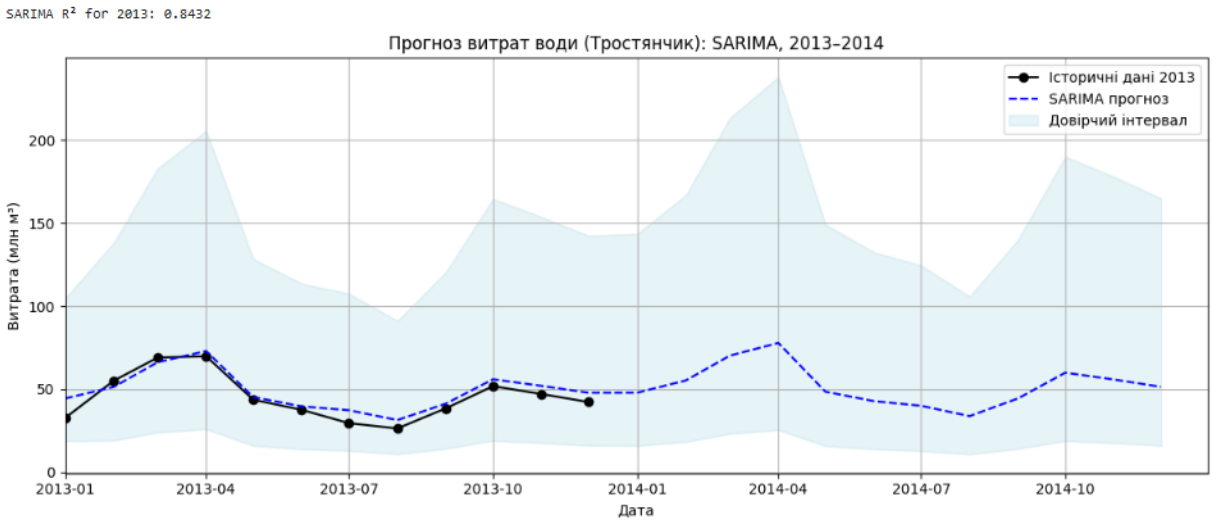

Рисунок 4.32 (Б) – Графік прогнозу витрат води моделлю SARIMA для с. Тростянчик (2013–2014)

Для подальшого прогнозування витрат води (Wдеф) на ділянці с. Тростянчик використано модель LSTM.

Спочатку підготовлено дані для моделі LSTM. Вибрано значення витрат води до кінця 2012 року, виконано масштабування за допомогою MinMaxScaler для нормалізації даних, а також створено послідовності з глибиною огляду 12 місяців для навчання. Побудовано модель LSTM із двома шарами (перший із 50 нейронами з активацією ReLU і return_sequences=True, другий із 50 нейронами), додано dropout (0.2) для уникнення перетренування, скомпілювано з функцією втрат MSE та оптимікатором Adam, і натреновано на 50 епох із валідаційною вибіркою 10% та розміром батча 32. Прогноз виконано на 24 місяці (2013–2014 роки) із подальшим зворотним масштабуванням результатів. На рисунках 4.33 і 4.34 зображені фрагменти програмного коду для підготовки даних, побудови та навчання моделі LSTM.

```
df_trostianchyk = pd.read_csv("/kaggle/input/gidropost-trostyanchik/trostyanchik_cleaned.csv")
df_trostianchyk = df_trostianchyk[~df_trostianchyk['Рік'].isin(['Середнє', 'Найбільше', 'Найменше'])]
df_trostianchyk['Рік'] = df_trostianchyk['Рік'].astype(int)
for month in ['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
              'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень']:
    df_trostianchyk[month] = pd.to_numeric(df_trostianchyk[month], errors='coerce')
month_map = {
    'Січень': 1, 'Лютий': 2, 'Березень': 3, 'Квітень': 4, 'Травень': 5, 'Червень': 6,
    'Липень': 7, 'Серпень': 8, 'Вересень': 9, 'Жовтень': 10, 'Листопад': 11, 'Грудень': 12
}
ts_df_long = df_trostianchyk.melt(
    id_vars=["Рік"],
    value_vars=['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
                'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень'],
    var_name="Місяць_назва",
    value_name="Витрата"
)
ts_df_long['Рік'] = ts_df_long['Рік'].astype(int)
ts_df_long["Місяць"] = ts_df_long["Місяць_назва"].map(month_map)
ts_df_long = ts_df_long.dropna(subset=["Витрата"])
ts_df_long["date"] = pd.to_datetime(dict(year=ts_df_long["Рік"], month=ts_df_long["Місяць"], day=1))
ts_df_long = ts_df_long.sort_values("date").reset_index(drop=True)
# Створення часового ряду та діагностика
ts_df_trostianchyk = ts_df_long.set_index('date')['Витрата']
print("Діапазон даних:", ts_df_trostianchyk.index.min(), "до", ts_df_trostianchyk.index.max())
print("Кількість даних:", len(ts_df_trostianchyk))
print("Мінімум і максимум:", ts_df_trostianchyk.min(), ts_df_trostianchyk.max())
# Нормалізація даних
scaler = MinMaxScaler()
ts_df_trostianchyk_scaled = scaler.fit_transform(ts_df_trostianchyk.values.reshape(-1, 1))
# Створення послідовностей для LSTM
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i + seq_length])
        y.append(data[i + seq_length])
    return np.array(X), np.array(y)
seq_length = 12
X, y = create_sequences(ts_df_trostianchyk_scaled, seq_length)
# Розділення на тренувальну та тестову вибірки
train_size = len(ts_df_trostianchyk) - 24  # Використаємо всі дані до 2012 року
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
# Побудова моделі LSTM
model = Sequential([
    LSTM(50, activation='relu', input_shape=(seq_length, 1)),
    Dropout(0.2),
    Dense(1)
])
```

Рисунок 4.33 – Програмний код для підготовки даних та побудови моделі LSTM (1 частина)

```
model.compile(optimizer=Adam(learning_rate=0.001), loss='mse')
# Тренування моделі з ранньою зупинкою
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_split=0.1, verbose=1, callbacks=[early_stopping])
# Прогноз
last_sequence = ts_df_trostianchyk_scaled[-seq_length:]
future_predictions = []
current_sequence = last_sequence
for _ in range(24):  # Прогноз на 24 місяці
    next_pred = model.predict(current_sequence.reshape(1, seq_length, 1), verbose=0)
    future_predictions.append(next_pred[0, 0])
    current_sequence = np.roll(current_sequence, -1)
    current_sequence[-1] = next_pred[0, 0]
future_predictions = np.array(future_predictions)
forecast_values = scaler.inverse_transform(future_predictions.reshape(-1, 1))
historical_values = scaler.inverse_transform(y_test.reshape(-1, 1))
# Обчислення R²
r2_lstm = r2_score(historical_values[:12], forecast_values[:12])  # Порівняння лише за 2013
print(f"LSTM R² for 2013: {r2_lstm:.4f}")
# Побудова графіка
forecast_index_trostianchyk = pd.date_range(start="2013-01-01", periods=24, freq="MS")
plt.figure(figsize=(12, 5))
plt.plot(pd.date_range(start="2013-01-01", periods=12, freq="MS"), historical_values[:12], label="Історичні дані 2013", marker="o", color="black")
plt.plot(forecast_index_trostianchyk, forecast_values, label="LSTM прогноз", linestyle="--", color="green")
plt.title("Прогноз витрат води (Тростянчик): LSTM, 2013–2014")
plt.xlabel("Дата")
plt.ylabel("Витрата (млн м³)")
plt.legend()
plt.grid(True)
plt.xlim(pd.Timestamp('2013-01-01'), pd.Timestamp('2014-12-31'))
plt.tight_layout()
plt.show()
```

Рисунок 4.34 – Програмний код для підготовки даних та побудови моделі LSTM (2 частина)

Для візуалізації результатів побудовано графік, на якому показано історичні дані за 2013 рік та прогнозовані значення за 2013–2014 роки. На рисунку 4.35 представлено графік, отриманий в результаті.

Рисунок 4.35 (А) – Графік прогнозу витрат води моделлю LSTM для с. Тростянчик (2013–2014)

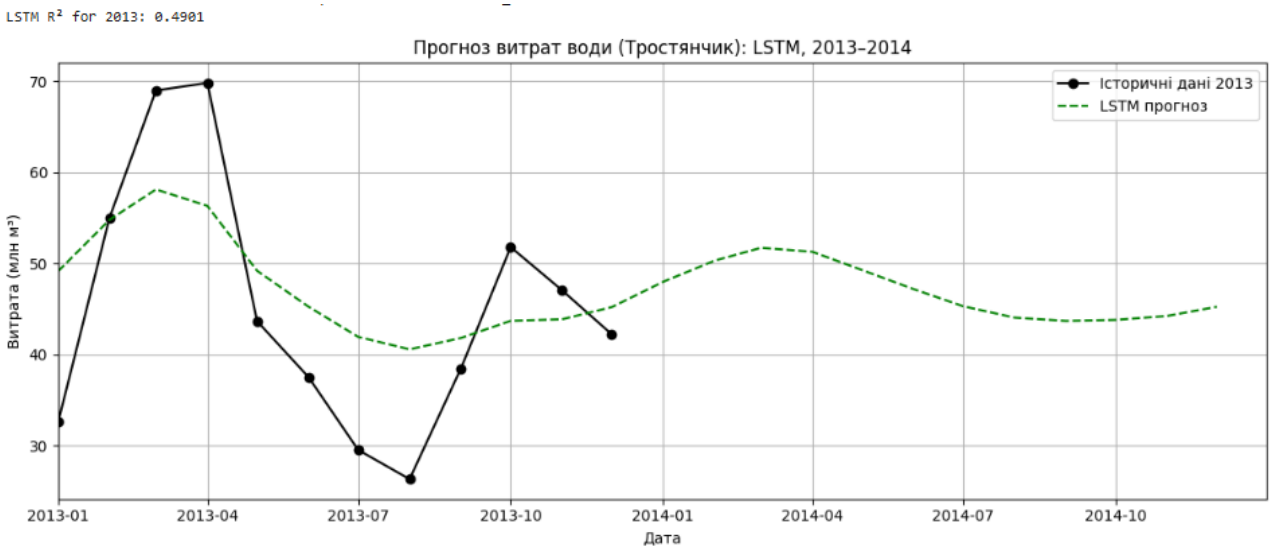

Рисунок 4.35 (Б) – Графік прогнозу витрат води моделлю LSTM для с. Тростянчик (2013–2014)

Оцінка моделей та аналіз результатів

Для оцінки точності моделей прогнозування витрат води (Wдеф) на ділянці с. Тростянчик проведено порівняння їхніх прогнозів за 2013 рік із реальними даними. Спочатку витягнуто історичні значення витрат за 2013 рік, а створено

функцію для обчислення метрик: MAE, MSE, RMSE, MAPE та R^2. Для кожної моделі (Prophet, SARIMA, LSTM) обчислено метрики на основі прогнозів за 2013 рік, результати зведено у таблицю та збережено у форматі Markdown. На рисунку 4.36 зображено програмний код для обчислення метрик і створення таблиці результатів.

```
true_values_2013_trostianchyk = ts_df_trostianchyk["2013-01-01":"2013-12-31"].values

# Функція для обчислення метрик
def calculate_metrics(true, predicted):
    if len(true) != len(predicted):
        raise ValueError(f"Довжина прогнозу ({len(predicted)}) не відповідає довжині реальних даних ({len(true)})")
    mae = mean_absolute_error(true, predicted)
    mse = mean_squared_error(true, predicted)
    rmse = math.sqrt(mse)
    mape = np.mean(np.abs((true - predicted) / np.where(true == 0, 1e-10, true))) * 100
    r2 = r2_score(true, predicted)
    return mae, mse, rmse, mape, r2

# Prophet: Отримуємо прогнози за 2013 рік
prophet_forecast_2013 = forecast_trostianchyk[forecast_trostianchyk["ds"].dt.year == 2013]["yhat"].values
prophet_mae, prophet_mse, prophet_rmse, prophet_mape, prophet_r2 = calculate_metrics(true_values_2013_trostianchyk, prophet_forecast_2013)

# SARIMA: Отримуємо прогнози за 2013 рік
sarima_forecast_2013 = sarima_df_trostianchyk[sarima_df_trostianchyk["date"].dt.year == 2013]["forecast"].values
sarima_mae, sarima_mse, sarima_rmse, sarima_mape, sarima_r2 = calculate_metrics(true_values_2013_trostianchyk, sarima_forecast_2013)

# LSTM: Отримуємо прогнози за 2013 рік
lstm_forecast_2013 = forecast_values[:12].flatten()  # Беремо перші 12 місяців (2013)
lstm_mae, lstm_mse, lstm_rmse, lstm_mape, lstm_r2 = calculate_metrics(true_values_2013_trostianchyk, lstm_forecast_2013)

# Перевірка довжин
print("Довжини прогнозів:", len(prophet_forecast_2013), len(sarima_forecast_2013), len(lstm_forecast_2013))
print("Реальні дані Тростянчик (2013):", true_values_2013_trostianchyk[:5])

# Створюємо таблицю з результатами
results_trostianchyk = {
    "Модель": ["Prophet", "SARIMA", "LSTM"],
    "MAE": [prophet_mae, sarima_mae, lstm_mae],
    "MSE": [prophet_mse, sarima_mse, lstm_mse],
    "RMSE": [prophet_rmse, sarima_rmse, lstm_rmse],
    "MAPE (%)": [prophet_mape, sarima_mape, lstm_mape],
    "R²": [prophet_r2, sarima_r2, lstm_r2]
}
results_df_trostianchyk = pd.DataFrame(results_trostianchyk)
results_df_trostianchyk[["MAE", "MSE", "RMSE", "MAPE (%)", "R²"]] = results_df_trostianchyk[["MAE", "MSE", "RMSE", "MAPE (%)", "R²"]].round(2)

# Виведення і збереження таблиці в Markdown
markdown_table = results_df_trostianchyk.to_markdown(index=False)
markdown_output = "# Результати порівняння точності всіх моделей прогнозування ділянки с. Тростянчик\n\n" + markdown_table
with open("model_comparison_trostianchyk.md", "w") as f:
    f.write(markdown_output)
print(markdown_output)
```

Рисунок 4.36 – Програмний код для обчислення метрик і створення таблиці результатів

Результатом виконання коду є таблиця, що містить значення метрик для всіх моделей, що дозволяє порівняти їхню точність. На рисунку 4.37 представлено таблицю з результатами порівняння моделей.

```
Довжини прогнозів: 12 12 12
Реальні дані Тростянчик (2013): [32.59680841 55.04516482 69.00009475 69.83355313 43.66839674]
# Результати порівняння точності всіх моделей прогнозування ділянки с. Тростянчик

| Модель   |   MAE |   MSE |   RMSE |   MAPE (%) |   R² |
|:---------|------:|------:|-------:|-----------:|-----:|
| Prophet  |  4.55 | 26.21 |   5.12 |      10.14 | 0.86 |
| SARIMA   |  4.62 | 28.66 |   5.35 |      12.1  | 0.84 |
| LSTM     |  8.23 | 93.19 |   9.65 |      21.16 | 0.49 |
```

Рисунок 4.37 – Таблиця порівняння точності моделей для с. Тростянчик

Далі проведено аналіз історичних і прогнозованих витрат води. Спочатку витягнуто історичні дані за 2013 рік та сформовано таблицю з витратами по місяцях. Потім для кожної моделі отримано прогнози на 2014 рік, об’єднано їх у єдину таблицю та збережено у форматі Markdown із назвами місяців англійською мовою. На рисунку 4.38 зображено програмний код для створення таблиць історичних і прогнозованих даних.

```python
# --- Історичні витрати води за 2013 рік ---
# Витяги історичних даних за 2013 рік
historical_2013_trostianchyk = ts_df_trostianchyk["2013-01-01":"2013-12-31"].reset_index()
if len(historical_2013_trostianchyk) != 12:
    raise ValueError(f"Очікується 12 місяців за 2013 рік, але отримано {len(historical_2013_trostianchyk)} записів.")
historical_2013_trostianchyk["Місяць"] = historical_2013_trostianchyk["date"].dt.strftime("%B")  # Назви місяців англійською
historical_2013_table = historical_2013_trostianchyk[["Місяць", "Витрата"]].copy()
historical_2013_table["Витрата"] = historical_2013_table["Витрата"].round(2)
# Виведення таблиці у Markdown
historical_markdown = "# Історичні витрати води за 2013 рік (Тростянчик)\n\n" + historical_2013_table.to_markdown(index=False)
print(historical_markdown)
with open("historical_2013_trostianchyk.md", "w") as f:
    f.write(historical_markdown)
# --- Прогнозовані витрати води за 2014 рік (всі моделі) ---
# Прогнози за 2014 рік для всіх моделей
prophet_forecast_2014 = forecast_trostianchyk[forecast_trostianchyk["ds"].dt.year == 2014][["ds", "yhat"]].copy()
prophet_forecast_2014.rename(columns={"ds": "date", "yhat": "Prophet"}, inplace=True)
if len(prophet_forecast_2014) != 12:
    raise ValueError(f"Очікується 12 місяців прогнозів Prophet за 2014 рік, але отримано {len(prophet_forecast_2014)} записів.")
sarima_forecast_2014 = sarima_df_trostianchyk[sarima_df_trostianchyk["date"].dt.year == 2014][["date", "forecast"]].copy()
sarima_forecast_2014.rename(columns={"forecast": "SARIMA"}, inplace=True)
if len(sarima_forecast_2014) != 12:
    raise ValueError(f"Очікується 12 місяців прогнозів SARIMA за 2014 рік, але отримано {len(sarima_forecast_2014)} записів.")
# Для LSTM: створюємо DataFrame із forecast_values для 2014 року (місяці 13-24)
forecast_index_2014 = pd.date_range(start="2014-01-01", periods=12, freq="MS")
if len(forecast_values) < 24:
    raise ValueError(f"Очікується щонайменше 24 значення в forecast_values, але отримано {len(forecast_values)}.")
lstm_forecast_2014 = pd.DataFrame({
    "date": forecast_index_2014,
    "LSTM": forecast_values[12:24].flatten()  # Беремо прогнози для 2014 року
})
if len(lstm_forecast_2014) != 12:
    raise ValueError(f"Очікується 12 місяців прогнозів LSTM за 2014 рік, але отримано {len(lstm_forecast_2014)} записів.")
# Об'єднуємо прогнози всіх моделей
forecast_2014_trostianchyk = prophet_forecast_2014.merge(sarima_forecast_2014, on="date").merge(lstm_forecast_2014, on="date")
if len(forecast_2014_trostianchyk) != 12:
    raise ValueError(f"Очікується 12 місяців у фінальній таблиці, але отримано {len(forecast_2014_trostianchyk)} записів.")
forecast_2014_trostianchyk["Місяць"] = forecast_2014_trostianchyk["date"].dt.strftime("%B")  # Назви місяців англійською
forecast_2014_table = forecast_2014_trostianchyk[["Місяць", "Prophet", "SARIMA", "LSTM"]].copy()
forecast_2014_table[["Prophet", "SARIMA", "LSTM"]] = forecast_2014_table[["Prophet", "SARIMA", "LSTM"]].round(2)
# Виведення таблиці у Markdown
forecast_markdown = "# Прогнозовані витрати води за 2014 рік (Тростянчик)\n\n" + forecast_2014_table.to_markdown(index=False)
print(forecast_markdown)
with open("forecast_2014_trostianchyk.md", "w") as f:
    f.write(forecast_markdown)
```

Рисунок 4.38 – Програмний код для створення таблиць історичних і прогнозованих витрат

Результатом є дві таблиці: одна з історичними витратами за 2013 рік, інша з прогнозованими значеннями на 2014 рік для всіх моделей. На рисунку 4.39 представлено таблиці витрат за 2013-2014 роки.

Історичні витрати води за 2013 рік (Тростянчик)

Місяць	Витрата
January	32.6
February	55.05
March	69
April	69.83
May	43.67
June	37.53
July	29.53
August	26.32
September	38.49
October	51.8
November	47.06
December	42.22

Прогнозовані витрати води за 2014 рік (Тростянчик)

Місяць	Prophet	SARIMA	LSTM
January	51.01	47.87	48.53
February	56.41	55.26	49.74
March	56.87	70.17	49.68
April	50.67	77.84	48.4
May	41.5	48.49	46.47
June	39.27	42.77	45.04
July	41.37	40	44.1
August	40.25	33.77	43.95
September	40.33	44.47	44.56
October	45.03	59.92	45.36
November	49.05	55.81	46.04
December	46.72	51.41	46.8

Рисунок 4.39 – Таблиці витрат води за 2013-2014 роки для с. Тростянчик

4.4 Висновки

На основі проведеного аналізу моделей прогнозування витрат води (Wдеф) для ділянок с. Лелітка та с. Тростянчик визначено, що ефективність моделей залежить від специфіки даних кожної ділянки. Для с. Лелітка найкращі результати продемонструвала модель SARIMA, яка досягла R^2 0.94, що свідчить про високу точність прогнозування. Модель Prophet також показала добрі результати з R^2 0.92, тоді як LSTM виявилася менш ефективною з R^2 0.69. Для с. Тростянчик найкращі показники точності продемонструвала модель Prophet із R^2 0.86, тоді як SARIMA досягла R^2 0.84, а LSTM показала найнижчий результат із R^2 0.49.

Отже, для практичного використання в управлінні водними ресурсами рекомендується застосовувати модель SARIMA для с. Лелітка завдяки її найвищій точності, а для с. Тростянчик — модель Prophet для оптимального балансу точності. Прогнозовані значення за 2014 рік для обох ділянок можуть бути використані для планування водогосподарських заходів, з урахуванням виявлених відмінностей у поведінці моделей на різних наборах даних.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи проведено аналіз предметної області, представлено суть проблеми та доведено актуальність питання аналізу водогосподарських балансів ділянок басейну Південного Бугу. Досліджено вплив кліматичних змін, сезонних коливань та антропогенних факторів на водний баланс, а також розглянуто можливості існуючих методів прогнозування водних ресурсів.

Для аналізу зібрано та оброблено дані про водогосподарські баланси басейну Південного Бугу, отримані з PDF-файлів. За допомогою бібліотек PyPDF2 і Pandas проведено їхню конвертацію у структурований CSV-формат, що включало очищення даних, заповнення пропусків і трансформацію у формат, придатний для подальшого моделювання. Це дозволило сформувати якісний датасет для роботи з даними.

Розроблено інформаційну технологію для аналізу водогосподарських балансів ділянок басейну Південного Бугу. Технологія включає етапи підготовки даних, моделювання та оцінки результатів. Запропоновано оптимальну IT-інфраструктуру, обґрунтовано вибір мови програмування Python та бібліотек Pandas, Scikit-learn, Prophet, pmdarima, TensorFlow для обробки часових рядів і прогнозування.

Проведено розвідувальний і просторовий аналіз даних із використанням бібліотек Python (Pandas, NumPy, GeoPandas, Matplotlib, Seaborn). Розвідувальний аналіз включав статистичний опис, аналіз кореляцій, побудову графіків часових рядів, гістограм і boxplot для виявлення трендів, сезонності та аномалій. Просторовий аналіз виконано за допомогою GeoPandas та Shapefile-даних, що дозволило створити карти дефіциту (Wдеф) і резервів (Wрез) води та виявити проблемні ділянки с. Лелітка та с. Тростянчик.

Здійснено прогнозування витрат води для виявлених проблемних ділянок с. Лелітка та с. Тростянчик за допомогою моделей машинного навчання Prophet, SARIMA та LSTM. Результати порівняно за метриками R^2: для с. Лелітка

найкращі результати показала модель SARIMA ($R^2 = 0.94$), а для с. Тростянчик — Prophet ($R^2 = 0.86$).

На основі розробленої інформаційної технології створено інструмент, який дозволяє здійснювати розвідувальний і просторовий аналіз та прогнозування даних за інформацією про водогосподарський баланс басейну Південного Бугу. Цей інструмент забезпечує комплексний підхід до аналізу водних ресурсів, сприяючи підвищенню точності прогнозів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гідрологічний аналіз басейну Південного Бугу: методологія та практика. URL: https://waterresources.gov.ua/hydrological-analysis-south-bug

2. Прогнозування водних ресурсів: принципи та методи. URL: https://www.hydrology.org/resources/forecasting-methods

3. Hyndman R. J., Athanasopoulos G. *Forecasting: principles and practice*. 2nd ed. Oxford : Oxford University Press, 2014. – URL: https://otexts.com/fpp2/

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

5. Бібліотека Pandas. URL: https://pandas.pydata.org

6. Підручник Python Pandas: DataFrame, діапазон дат, використання Pandas. URL: https://www.guru99.com/uk/python-pandas-tutorial.html

7. Бібліотека Matplotlib. URL: https://matplotlib.org/

8. Matplotlib. URL: https://kkite.pnu.edu.ua/wp-content/uploads/sites/50/2020/04/Matplotlib.pdf

9. Бібліотека NumPy. URL: https://numpy.org/

10. Бібліотека Scikit-learn. URL: https://scikit-learn.org/stable/

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

12. Бібліотека TensorFlow. URL: https://www.tensorflow.org/

13. Практичне застосування TensorFlow для прогнозування часових рядів. URL: https://www.tensorflow.org/tutorials/structured_data/time_series

14. Бібліотека XGBoost. URL: https://xgboost.readthedocs.io/en/stable/

15. Бібліотека pmdarima. URL: https://alkaline-ml.com/pmdarima/

16. SARIMA моделі для аналізу часових ряді. URL: https://www.statsmodels.org/stable/examples/notebooks/generated/sarima.html

17. Бібліотека Prophet. URL: https://pypi.org/project/prophet/

18. Мокін В. Б., Дратований М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних : електронний навчальний посібник комбінованого

(локального та мережевого) використання [Електронний ресурс]. Вінниця : ВНТУ, 2024, 258 с. URL: https://docs.vntu.edu.ua/card.php?id=8163

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

20. Гідрологічні моделі для управління водними ресурсами. URL: https://www.hydrology-and-earth-system-sciences.net/

21. Дослідження кліматичних змін у басейні Південного Бугу. URL: https://climate-adapt.eea.europa.eu/knowledge/tools/south-bug-basin

22. Часові ряди та їхнє застосування у гідрології. URL: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/time-series-analysis

23. Прогнозування дефіциту води в Україні: сучасні підходи. URL: https://water.ua.gov.ua/forecasting-water-deficit

24. Seaborn: бібліотека для статистичної візуалізації даних. URL: https://seaborn.pydata.org/

25. Water Resources Management in the South Bug Basin: Challenges and Opportunities. URL: https://www.iwmi.cgiar.org/publications/south-bug-water-management

26. Box G. E. P., Jenkins G. M., Reinsel G. C. Time Series Analysis: Forecasting and Control. 5th ed. Wiley, 2015. – URL: https://www.wiley.com/en-us/Time+Series+Analysis%3A+Forecasting+and+Control%2C+5th+Edition-p-9781118675021.

27. Прогнозування сезонних коливань водних ресурсів: методи та інструменти. URL: https://www.hydrology-journal.org/seasonal-forecasting-water

28. Оцінка впливу кліматичних змін на водний баланс України. URL: https://www.ecology.gov.ua/climate-impact-water-balance

29. Моделювання водного дефіциту з використанням машинного навчання. URL: https://www.jhydrol.com/machine-learning-water-deficit-modeling

Додаток А
(обов'язковий)

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

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
"____"_____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на бакалаврську кваліфікаційну роботу
ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ВОДОГОСПОДАРСЬКИХ БАЛАНСІВ ДІЛЯНОК БАСЕЙНУ ПІВДЕННОГО БУГУ
08-34.БКР.003.00.000 ТЗ

Керівник: доц.
_____ Ігор ШТЕЛЬМАХ
«__» _____ 2025 р.
Розробив студент гр. 2ІСТ-21б
_____ Роман ВОВК
«__» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2025р., та індивідуальне завдання на БКР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2025р.

2. Джерела розробки:

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

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

3. Мета і призначення роботи.

Метою роботи є розроблення інформаційної технології для аналізу водогосподарських балансів ділянок басейну Південного Бугу, що дозволяє виявляти проблемні ділянки з дефіцитом води та прогнозувати їхні витрати для підтримки управлінських рішень.

4. Методи дослідження:

Вивчення водогосподарських балансів, аналіз даних про водні ресурси басейну Південного Бугу, розроблення моделі водного балансу, створення карт і графіків, інтеграція просторових даних та прогнозування витрат води з оцінкою точності.

5. Етапи роботи і терміни їх виконання:

a) Загальна характеристика обєкту досліджень _____ – _____
b) Вибір оптимальної інфраструктури _____ – _____
c) Підготовка та розвідувальний аналіз даних _____ – _____
d) Розроблення інформаційної технології та прогнозування _____ – _____
e) Оформлення матеріалів до захисту БКР _____ – _____

7. Очікувані результати та порядок реалізації

Отримання інформаційної технології для аналізу водогосподарських балансів ділянок басейну Південного Бугу, що забезпечить виявлення проблемних ділянок із дефіцитом води та прогнозування їхніх витрат.

8. Вимоги до розробленої документації

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

9. Порядок приймання роботи

Публічний захист «___» _____ 2025 р.
Початок розробки «___» _____ 2025 р.
Граничні терміни виконання БКР «___» _____ 2025 р.
Розробив студент групи 2ІСТ-21б _____ Роман ВОВК

Додаток Б

(обов’язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Інформаційна технологія аналізу водогосподарських балансів ділянок басейну Південного Бугу»

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІІТА, гр. 2ІСТ-21б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,6 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

- ■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- □ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- □ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Віталій МОКІН, зав. каф. САІТ _____ (підпис)

Євгеній КРИЖАНОВСЬКИЙ, доц. каф. САІТ _____ (підпис)

Особа, відповідальна за перевірку _____ (підпис) Сергій ЖУКОВ

З висновком експертної комісії ознайомлений(-на)

Керівник _____ (підпис) Ігор ШТЕЛЬМАХ, к.т.н., ас. каф. САІТ

Здобувач _____ (підпис) Роман ВОВК

Додаток В
(довідниковий)

Фрагмент лістингу програми

EDA

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import display
!pip uninstall shapely -y
!pip install shapely==1.8.5.post1
# Налаштування виводу
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 50)
pd.set_option('display.float_format', '{:.4f}'.format)
# Завантаження CSV (шлях оновлюй під свій датасет)
df = pd.read_csv('/kaggle/input/water-balance-southern-buh/pivdennyi_buh_FINAL_corrected.csv')
# Перевіримо розмір та типи
print(" Розмір таблиці:", df.shape)
display(df.dtypes)
# Статистичний опис числових колонок
display(df.describe())
# Перевірка на пропущені значення
missing = df.isnull().sum()
missing = missing[missing > 0]
if not missing.empty:
    print("🔍 Пропущені значення:")
    display(missing)
else:
    print(" Усі дані заповнені — пропусків немає.")
# Унікальні коди ВГД
print(" Унікальні коди ВГД:")
print(df["Код ВГД"].unique())

# Кількість записів по кожному коду ВГД
print("\n Розподіл по ВГД:")
display(df["Код ВГД"].value_counts())
# Рівні забезпеченості
print("\n Рівні забезпеченості:")
```

```
display(df["Забезпеченість, %"].value_counts())
# Розподіл по місяцях
print("\n Розподіл по місяцях:")
display(df["Місяць"].value_counts().sort_index())
# Групуємо середній Wдеф по коду ВГД та місяцю
pivot_def = df.pivot_table(
    index="Код ВГД",
    columns="Місяць",
    values="Wдеф, млн куб. м",
    aggfunc="mean"
)
# Побудова heatmap
plt.figure(figsize=(12, 6))
sns.heatmap(pivot_def, annot=True, fmt=".2f", cmap="YlOrRd", linewidths=0.5)
plt.title("Середній дефіцит води (Wдеф) по ВГД та місяцях")
plt.ylabel("Код ВГД")
plt.xlabel("Місяць")
plt.tight_layout()
plt.show()
import warnings
warnings.filterwarnings("ignore")
# Групуємо дані: середній Wвсього по місяцях і забезпеченості
grouped = df.groupby(["Місяць", "Забезпеченість, %"])["Wвсього, млн куб. м"].mean().reset_index()
# Побудова графіка
plt.figure(figsize=(10, 6))
sns.lineplot(data=grouped, x="Місяць", y="Wвсього, млн куб. м", hue="Забезпеченість, %", marker="o")
plt.title("Середнє значення Wвсього по місяцях для кожної забезпеченості")
plt.xlabel("Місяць")
plt.ylabel("Wвсього, млн куб. м")
plt.xticks(range(1, 13))
plt.grid(True)
plt.legend(title="Забезпеченість, %")
plt.tight_layout()
plt.show()
grouped.pivot(index="Місяць", columns="Забезпеченість, %", values="Wвсього, млн куб. м")
# Групування: середній Wдеф по місяцях і забезпеченості
group_def = df.groupby(["Місяць", "Забезпеченість, %"])["Wдеф, млн куб. м"].mean().reset_index()
# Побудова графіка
plt.figure(figsize=(10, 6))
sns.lineplot(data=group_def, x="Місяць", y="Wдеф, млн куб. м", hue="Забезпеченість, %", marker="o")
```

```
plt.title(" Середній дефіцит води (Wдеф) по місяцях для кожної забезпеченості")
plt.xlabel("Місяць")
plt.ylabel("Wдеф, млн куб. м")
plt.xticks(range(1, 13))
plt.grid(True)
plt.legend(title="Забезпеченість, %")
plt.tight_layout()
plt.show()
# Вибираємо тільки числові колонки
numeric_df = df.select_dtypes(include=np.number)
# Розрахунок кореляційної матриці
corr = numeric_df.corr()
# Побудова heatmap кореляцій
plt.figure(figsize=(14, 10))
sns.heatmap(corr, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
plt.title("Кореляційна матриця між числовими ознаками")
plt.tight_layout()
plt.show()
# Щоб не було забагато — беремо ключові 5 змінних
pair_df = df[["Wпзв, млн куб. м", "Wвсього, млн куб. м", "Wдеф, млн куб. м", "Wрез, млн куб. м", "Wтранзит, млн куб. м"]]
# Побудова pairplot
sns.pairplot(pair_df)
plt.suptitle(" Парні порівняння показників", y=1.02)
plt.show()
# Сума Wвсього по кожному ВГД
total_by_vgd = df.groupby("Код ВГД")["Wвсього, млн куб. м"].sum().sort_values(ascending=False)
# Побудова барчарту
plt.figure(figsize=(10, 6))
sns.barplot(x=total_by_vgd.index, y=total_by_vgd.values)
plt.title("Загальна величина Wвсього по кожному коду ВГД")
plt.xlabel("Код ВГД")
plt.ylabel("Сума Wвсього, млн куб. м")
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
import geopandas as gpd
gdf = gpd.read_file("/kaggle/input/maps-in-shapes/vgd.shp")
gdf["Код_обрізаний"] = gdf["SEM373"].str.strip().str.strip("").str[:10]
df["Код ВГД"] = df["Код ВГД"].str.strip().str.replace("М", "M", regex=False).str[:10]
```

```
agg_def = df.groupby("Код ВГД")[["Wдеф, млн куб. м", "Wрез, млн куб. м"]].mean().reset_index()
gdf_def = gdf.merge(agg_def, left_on="Код_обрізаний", right_on="Код ВГД")
print(" Рядків після злиття:", gdf_def.shape[0])
print(" Валідна геометрія:", gdf_def["geometry"].notnull().sum())
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 8))
gdf_def.plot(
    column="Wдеф, млн куб. м",
    cmap="Reds",
    legend=True,
    edgecolor="black"
)
plt.title("Карта дефіциту води (Wдеф) по водогосподарських ділянках")
plt.axis("off")
plt.show()
# Побудова карти резервів води (Wрез)
plt.figure(figsize=(10, 8))
gdf_def.plot(
    column="Wрез, млн куб. м",
    cmap="Blues",
    legend=True,
    edgecolor="black"
)
plt.title("Карта резерву води (Wрез) по водогосподарських ділянках")
plt.axis("off")
plt.show()
# ТОП-5 ділянок з найбільшим дефіцитом
top_deficit = gdf_def.sort_values(by="Wдеф, млн куб. м", ascending=False).head(5)
print("🏆 ТОП-5 ділянок з найбільшим дефіцитом:")
display(top_deficit[["Код ВГД", "Wдеф, млн куб. м"]])
# ТОП-5 ділянок з найменшим резервом
lowest_reserve = gdf_def.sort_values(by="Wрез, млн куб. м", ascending=True).head(5)
print("📉 ТОП-5 ділянок з найменшим резервом:")
display(lowest_reserve[["Код ВГД", "Wрез, млн куб. м"]])
```

Forecasting

```
!pip install prophet
!pip install pmdarima
import numpy as np
import pandas as pd
import matplotlib
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from prophet import Prophet
import pmdarima as pm
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import math
df_trost = pd.read_csv("/kaggle/input/gidropost-trostyanchik/trostyanchik_cleaned.csv")
df_trost.head()
month_columns = ['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
                 'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень']
trost_long = df_trost.melt(id_vars=["Рік"], value_vars=month_columns,
                           var_name="Місяць_назва", value_name="Витрата")
month_map = {
    'Січень': 1, 'Лютий': 2, 'Березень': 3, 'Квітень': 4,
    'Травень': 5, 'Червень': 6, 'Липень': 7, 'Серпень': 8,
    'Вересень': 9, 'Жовтень': 10, 'Листопад': 11, 'Грудень': 12
}
trost_long["Місяць"] = trost_long["Місяць_назва"].map(month_map)
trost_long = trost_long[trost_long["Рік"].astype(str).str.isnumeric()]
trost_long["Рік"] = trost_long["Рік"].astype(int)
trost_long["Місяць"] = trost_long["Місяць"].astype(int)
trost_long["date"] = pd.to_datetime(dict(year=trost_long["Рік"], month=trost_long["Місяць"], day=1))
trost_long = trost_long.sort_values("date")
# Формуємо time series
ts_df = trost_long[["date", "Витрата"]].copy()
ts_df = ts_df.set_index("date")
ts_df = ts_df.asfreq("MS")  # MS = Month Start
# Заповнюємо пропуски
ts_df = ts_df.ffill()
# Перевірка на пропуски
print("Кількість пропусків:", ts_df.isna().sum().values[0])
# Побудова графіка для перевірки
ts_df.plot(figsize=(12, 4), title="Часовий ряд витрат води (Тростянчик)")
# Гістограма
```

```
ts_df["Витрата"].plot(kind="hist", bins=50, figsize=(10, 4), title="Гістограма витрат води (Тростянчик)")
plt.xlabel("Витрата")
plt.grid(True)
plt.show()
# Ковзне середнє
ts_df["Витрата"].rolling(window=6).mean().plot(figsize=(12, 4), title="Ковзне середнє витрат (6 міс)")
plt.xlabel("Дата")
plt.ylabel("Витрата")
plt.grid(True)
plt.show()
# Додаємо колонку з місяцем
ts_df["Місяць"] = ts_df.index.month
# Boxplot по місяцях
plt.figure(figsize=(12, 5))
sns.boxplot(x="Місяць", y="Витрата", data=ts_df.reset_index())
plt.title("Boxplot витрат води по місяцях (Тростянчик)")
plt.grid(True)
plt.show()
# ⏱️ 1. Підготовка даних для Тростянчика
prophet_df_trostianchyk = ts_df[["Витрата"]].copy()
prophet_df_trostianchyk["y"] = np.log1p(prophet_df_trostianchyk["Витрата"])
prophet_df_trostianchyk["ds"] = prophet_df_trostianchyk.index
prophet_df_trostianchyk = prophet_df_trostianchyk[["ds", "y"]]
model_trostianchyk = Prophet(seasonality_mode='additive', yearly_seasonality=True)
model_trostianchyk.random_state = 42
model_trostianchyk.fit(prophet_df_trostianchyk)
future_trostianchyk = model_trostianchyk.make_future_dataframe(periods=24, freq="MS")
forecast_trostianchyk = model_trostianchyk.predict(future_trostianchyk)
forecast_trostianchyk["yhat"] = np.expm1(forecast_trostianchyk["yhat"])
forecast_trostianchyk["yhat_lower"] = np.expm1(forecast_trostianchyk["yhat_lower"])
forecast_trostianchyk["yhat_upper"] = np.expm1(forecast_trostianchyk["yhat_upper"])
combined_df_trostianchyk = forecast_trostianchyk[(forecast_trostianchyk["ds"] >= "2013-01-01") & (forecast_trostianchyk["ds"] <= "2014-12-31")].copy()
actual_trostianchyk = prophet_df_trostianchyk[(prophet_df_trostianchyk["ds"] >= "2013-01-01") & (prophet_df_trostianchyk["ds"] <= "2013-12-31")].copy()
actual_trostianchyk["y"] = np.expm1(actual_trostianchyk["y"])
plt.figure(figsize=(12, 5))
plt.plot(combined_df_trostianchyk["ds"], combined_df_trostianchyk["yhat"], label="Прогноз", color="blue")
plt.fill_between(combined_df_trostianchyk["ds"], combined_df_trostianchyk["yhat_lower"], combined_df_trostianchyk["yhat_upper"],
                 alpha=0.3, label="Довірчий інтервал", color="lightblue")
```

```
plt.scatter(actual_trostianchyk["ds"], actual_trostianchyk["y"], color="black", label="Історичні дані", s=20)
plt.title("Прогноз витрат води (Тростянчик): 2013–2014")
plt.xlabel("Дата")
plt.ylabel("Витрата (млн м³)")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig('prophet_forecast_trostianchyk.png')
# Використаємо time series без логарифмування для Тростянчика
sarima_series_trostianchyk = ts_df["Витрата"].copy()
# Тренуємо SARIMA вручну з параметрами для Тростянчика
sarima_model_trostianchyk = SARIMAX(sarima_series_trostianchyk,
                                    order=(0,1,1),
                                    seasonal_order=(0,1,1,12),
                                    enforce_stationarity=False,
                                    enforce_invertibility=False)
sarima_result_trostianchyk = sarima_model_trostianchyk.fit(disp=False)
# Прогноз на 24 місяці (2013-2014) для Тростянчика
n_months = 24
sarima_forecast_trostianchyk = sarima_result_trostianchyk.get_forecast(steps=n_months)
sarima_pred_trostianchyk = sarima_forecast_trostianchyk.predicted_mean
forecast_index_trostianchyk = pd.date_range(start="2013-01-01", periods=n_months, freq="MS")
sarima_df_trostianchyk = pd.DataFrame({"date": forecast_index_trostianchyk, "forecast": sarima_pred_trostianchyk})
historical_trostianchyk = ts_df["Витрата"]["2013-01-01":"2013-12-01"]
plt.figure(figsize=(12, 5))
plt.plot(historical_trostianchyk.index, historical_trostianchyk.values, label="Історичні дані", marker="o", color="black")
plt.plot(sarima_df_trostianchyk["date"], sarima_df_trostianchyk["forecast"], label="SARIMA прогноз", linestyle="--", color="blue")
plt.title("Прогноз витрат води (Тростянчик): SARIMA, 2013–2014")
plt.xlabel("Дата")
plt.ylabel("Витрата (млн м³)")
plt.legend()
plt.grid(True)
plt.xlim(pd.Timestamp('2013-01-01'), pd.Timestamp('2014-12-31'))
plt.savefig('sarima_forecast_trostianchyk.png')
# Масштабуємо дані для Тростянчика
scaler_trostianchyk = MinMaxScaler()
scaled_data_trostianchyk = scaler_trostianchyk.fit_transform(ts_df[["Витрата"]])
# Вікно (12 місяців) + створення X та y для Тростянчика
window_trostianchyk = 12
```

```
X_trostianchyk, y_trostianchyk = [], []
for i in range(window_trostianchyk, len(scaled_data_trostianchyk)):
    X_trostianchyk.append(scaled_data_trostianchyk[i - window_trostianchyk:i, 0])
    y_trostianchyk.append(scaled_data_trostianchyk[i, 0])
X_trostianchyk, y_trostianchyk = np.array(X_trostianchyk), np.array(y_trostianchyk)
# Переформатовуємо під LSTM для Тростянчика
X_trostianchyk = np.reshape(X_trostianchyk, (X_trostianchyk.shape[0], X_trostianchyk.shape[1], 1))
# Модель LSTM для Тростянчика
model_lstm_trostianchyk = Sequential()
model_lstm_trostianchyk.add(LSTM(units=64, return_sequences=False, input_shape=(X_trostianchyk.shape[1], 1)))
model_lstm_trostianchyk.add(Dense(1))
model_lstm_trostianchyk.compile(optimizer='adam', loss='mean_squared_error')
# Навчаємо модель для Тростянчика
model_lstm_trostianchyk.fit(X_trostianchyk, y_trostianchyk, epochs=50, batch_size=32, verbose=1)
# Починаємо з останніх 12 місяців (2012-12-01 до 2013-11-01) для Тростянчика
last_12_trostianchyk = scaled_data_trostianchyk[-12:]
input_seq_trostianchyk = np.reshape(last_12_trostianchyk, (1, 12, 1))
# Прогноз на 24 місяці (2013-2014) для Тростянчика
lstm_predictions_trostianchyk = []
for _ in range(24):
    next_pred_trostianchyk = model_lstm_trostianchyk.predict(input_seq_trostianchyk, verbose=0)[0][0]
    lstm_predictions_trostianchyk.append(next_pred_trostianchyk)
    input_seq_trostianchyk = np.append(input_seq_trostianchyk[:, 1:, :], [[[next_pred_trostianchyk]]], axis=1)
# Масштаб назад для Тростянчика
lstm_pred_scaled_trostianchyk =
scaler_trostianchyk.inverse_transform(np.array(lstm_predictions_trostianchyk).reshape(-1, 1)).flatten()
# Створимо DataFrame для Тростянчика
lstm_dates_trostianchyk = pd.date_range(start="2013-01-01", periods=24, freq="MS")
lstm_df_trostianchyk = pd.DataFrame({"date": lstm_dates_trostianchyk, "forecast": lstm_pred_scaled_trostianchyk})
# Історичні дані за 2013 рік для Тростянчика
historical_trostianchyk = ts_df["Витрата"]["2013-01-01":"2013-12-31"]
# Побудова графіка для Тростянчика
plt.figure(figsize=(12, 5))
plt.plot(historical_trostianchyk.index, historical_trostianchyk.values, label="Історичні дані", marker="o",
color="black")
plt.plot(lstm_df_trostianchyk["date"], lstm_df_trostianchyk["forecast"], label="LSTM прогноз", linestyle="--",
color="green")
plt.title("Прогноз витрат води (Тростянчик): LSTM, 2013–2014")
plt.xlabel("Дата")
plt.ylabel("Витрата (млн м³)")
plt.legend()
```

```
plt.grid(True)
plt.xlim(pd.Timestamp('2013-01-01'), pd.Timestamp('2014-12-31'))
plt.savefig('lstm_forecast_trostianchyk.png')
# Завантажуємо CSV
df = pd.read_csv("/kaggle/input/gidropost-trostyanchik/trostyanchik_cleaned.csv")
# Список місяців
month_columns = ['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень', 'Червень',
                 'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень']
# Melt wide → long
df_long = df.melt(id_vars=["Рік"], value_vars=month_columns,
                  var_name="Місяць_назва", value_name="Витрата")
# Видаляємо нечислові значення в колонці "Рік" та "Витрата"
df_long = df_long[pd.to_numeric(df_long["Рік"], errors="coerce").notna()]
df_long = df_long[pd.to_numeric(df_long["Витрата"], errors="coerce").notna()]
df_long = df_long.dropna()

# Мапа місяців
month_map = {
    'Січень': 1, 'Лютий': 2, 'Березень': 3, 'Квітень': 4,
    'Травень': 5, 'Червень': 6, 'Липень': 7, 'Серпень': 8,
    'Вересень': 9, 'Жовтень': 10, 'Листопад': 11, 'Грудень': 12
}
df_long["Місяць"] = df_long["Місяць_назва"].map(month_map)
# Перетворюємо типи
df_long["Рік"] = df_long["Рік"].astype(int)
df_long["Місяць"] = df_long["Місяць"].astype(int)
df_long["Витрата"] = df_long["Витрата"].astype(float)
# Створюємо дату
df_long["date"] = pd.to_datetime(dict(year=df_long["Рік"], month=df_long["Місяць"], day=1))
df_long = df_long.sort_values("date").reset_index(drop=True)
# Готова time-series таблиця
ts_df_xgb = df_long[["date", "Витрата"]].copy()
ts_df_xgb.set_index("date", inplace=True)
# Створимо нові колонки для моделі
df_long["year"] = df_long["date"].dt.year
df_long["month"] = df_long["date"].dt.month
X = df_long[["year", "month"]]
y = df_long["Витрата"]
X_train = X[df_long["year"] < 2013]
y_train = y[df_long["year"] < 2013]
```

```
X_test = X[df_long["year"] == 2013]
y_test = y[df_long["year"] == 2013]
model = XGBRegressor(n_estimators=100, learning_rate=0.1, random_state=42)
model.fit(X_train, y_train)
y_pred_2013 = model.predict(X_test)
future_dates = pd.DataFrame({
    "year": [2014]*12,
    "month": list(range(1, 13))
})
future_preds = model.predict(future_dates)
future_dates["forecast"] = future_preds
future_dates["date"] = pd.to_datetime(dict(year=future_dates["year"], month=future_dates["month"], day=1))
plt.figure(figsize=(12, 6))
plt.plot(df_long[df_long["year"] == 2013]["date"], y_test, label="Реальні 2013")
plt.plot(df_long[df_long["year"] == 2013]["date"], y_pred_2013, label="Прогноз 2013", linestyle="--")
plt.plot(future_dates["date"], future_dates["forecast"], label="Прогноз 2014", linestyle="--")
plt.title("XGBoost прогноз для Тростянчика (2013–2014)")
plt.xlabel("Дата")
plt.ylabel("Витрата")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
true_values_2013 = ts_df["Витрата"]["2013-01-01":"2013-12-31"].values
def calculate_metrics(true, predicted):
    mae = mean_absolute_error(true, predicted)
    mse = mean_squared_error(true, predicted)
    rmse = math.sqrt(mse)
    # MAPE: уникаємо ділення на нуль
    mape = np.mean(np.abs((true - predicted) / np.where(true == 0, 1e-10, true))) * 100
    r2 = r2_score(true, predicted)
    return mae, mse, rmse, mape, r2
# 1. Prophet: Отримуємо прогноз за 2013 рік
prophet_forecast = forecast_trostianchyk[(forecast_trostianchyk["ds"] >= "2013-01-01") &
                                         (forecast_trostianchyk["ds"] <= "2013-12-31")]["yhat"].values
prophet_mae, prophet_mse, prophet_rmse, prophet_mape, prophet_r2 = calculate_metrics(true_values_2013,
prophet_forecast)
# 2. SARIMA: Отримуємо прогноз за 2013 рік
sarima_forecast_2013 = sarima_df_trostianchyk[sarima_df_trostianchyk["date"].between("2013-01-01", "2013-12-
31")]["forecast"].values
```

```
sarima_mae, sarima_mse, sarima_rmse, sarima_mape, sarima_r2 = calculate_metrics(true_values_2013, sarima_forecast_2013)

# 3. LSTM: Отримуємо прогноз за 2013 рік

lstm_forecast_2013 = lstm_df_trostianchyk[lstm_df_trostianchyk["date"].between("2013-01-01", "2013-12-31")]["forecast"].values

lstm_mae, lstm_mse, lstm_rmse, lstm_mape, lstm_r2 = calculate_metrics(true_values_2013, lstm_forecast_2013)

# 4. XGBoost: Отримуємо прогноз за 2013 рік

xgboost_forecast_2013 = y_pred_2013  # y_pred_2013 з коду XGBoost

xgboost_mae, xgboost_mse, xgboost_rmse, xgboost_mape, xgboost_r2 = calculate_metrics(true_values_2013, xgboost_forecast_2013)

# Створюємо таблицю з результатами

results = {

   "Модель": ["Prophet", "SARIMA", "LSTM", "XGBoost"],

   "MAE": [prophet_mae, sarima_mae, lstm_mae, xgboost_mae],

   "MSE": [prophet_mse, sarima_mse, lstm_mse, xgboost_mse],

   "RMSE": [prophet_rmse, sarima_rmse, lstm_rmse, xgboost_rmse],

   "MAPE (%)": [prophet_mape, sarima_mape, lstm_mape, xgboost_mape],

   "R²": [prophet_r2, sarima_r2, lstm_r2, xgboost_r2]

}

results_df = pd.DataFrame(results)

# Форматування чисел до 2 знаків після коми

results_df["MAE"] = results_df["MAE"].round(2)

results_df["MSE"] = results_df["MSE"].round(2)

results_df["RMSE"] = results_df["RMSE"].round(2)

results_df["MAPE (%)"] = results_df["MAPE (%)"].round(2)

results_df["R²"] = results_df["R²"].round(2)

# Виводимо таблицю у форматі Markdown

markdown_table = results_df.to_markdown(index=False)

# Додаємо заголовок

markdown_output = "# Результати порівняння точності всіх моделей прогнозування ділянки Тростянчик**\n\n**" + markdown_table

# Зберігаємо у файл

with open("model_comparison_trostianchyk.md", "w") as f:

   f.write(markdown_output)

print(markdown_output)

# --- Аналіз витрат води за 2013 рік (історичні дані) ---

# Витяги історичних даних за 2013 рік

historical_2013_trostianchyk = ts_df["Витрата"]["2013-01-01":"2013-12-31"].reset_index()

historical_2013_trostianchyk["Місяць"] = historical_2013_trostianchyk["date"].dt.strftime("%B")

historical_2013_table = historical_2013_trostianchyk[["Місяць", "Витрата"]].copy()

historical_2013_table["Витрата"] = historical_2013_table["Витрата"].round(2)

# Виведення таблиці у Markdown
```

```
historical_markdown = "# Історичні витрати води за 2013 рік (Тростянчик)\n\n" + historical_2013_table.to_markdown(index=False)

print(historical_markdown)

with open("historical_2013_trostianchyk.md", "w") as f:
    f.write(historical_markdown)

# --- Прогнозовані витрати води за 2014 рік (всі моделі) ---

# Прогнози за 2014 рік для всіх моделей

prophet_forecast_2014 = forecast_trostianchyk[(forecast_trostianchyk["ds"] >= "2014-01-01") & (forecast_trostianchyk["ds"] <= "2014-12-31")][["ds", "yhat"]].copy()

prophet_forecast_2014.rename(columns={"ds": "date", "yhat": "Prophet"}, inplace=True)

sarima_forecast_2014 = sarima_df_trostianchyk[sarima_df_trostianchyk["date"].between("2014-01-01", "2014-12-31")][["date", "forecast"]].copy()

sarima_forecast_2014.rename(columns={"forecast": "SARIMA"}, inplace=True)

lstm_forecast_2014 = lstm_df_trostianchyk[lstm_df_trostianchyk["date"].between("2014-01-01", "2014-12-31")][["date", "forecast"]].copy()

lstm_forecast_2014.rename(columns={"forecast": "LSTM"}, inplace=True)

xgboost_forecast_2014 = future_dates[["date", "forecast"]].copy()

xgboost_forecast_2014.rename(columns={"forecast": "XGBoost"}, inplace=True)

# Об'єднуємо прогнози всіх моделей

forecast_2014_trostianchyk = prophet_forecast_2014.merge(sarima_forecast_2014, on="date").merge(lstm_forecast_2014, on="date").merge(xgboost_forecast_2014, on="date")

forecast_2014_trostianchyk["Місяць"] = forecast_2014_trostianchyk["date"].dt.strftime("%B")

forecast_2014_table = forecast_2014_trostianchyk[["Місяць", "Prophet", "SARIMA", "LSTM", "XGBoost"]].copy()

forecast_2014_table[["Prophet", "SARIMA", "LSTM", "XGBoost"]] = forecast_2014_table[["Prophet", "SARIMA", "LSTM", "XGBoost"]].round(2)

# Виведення таблиці у Markdown

forecast_markdown = "# Прогнозовані витрати води за 2014 рік (Тростянчик)\n\n" + forecast_2014_table.to_markdown(index=False)

print(forecast_markdown)

with open("forecast_2014_trostianchyk.md", "w") as f:
    f.write(forecast_markdown)
```

Додаток Г
(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ВОДОГОСПОДАРСЬКИХ БАЛАНСІВ ДІЛЯНОК БАСЕЙНУ ПІВДЕННОГО БУГУ

Нормоконтроль: к.т.н., доцент
_____ Сергій ЖУКОВ
«___» _____ 2025 р.

	Річка	ВГД	Код ВГД	Забезпеченість, %	Місяць	Кількість днів	Wвх, млн куб. м	Wбіч, млн куб. м	Wпзв, млн куб. м	Wзв, млн куб. м	Wдоп, млн куб. м	Wрез_ΔV, млн куб. м	Wресурси, млн куб. м	Wвип, млн куб. м	Wф, млн куб. м	Wз, млн куб. м	Wпер, млн куб. м	Wвкр, млн куб. м	We, млн куб. м	Wвсього, млн куб. м	Wдеф, млн куб. м	Wрез, млн куб. м	Wтранзит, млн куб. м
0	Південний Буг	від витоку до гирла р. Інгва	M5.4.0.01	50	1	31	0.0000	34.7656	0.5124	0.9915	0.0	0.7909	37.0604	0.0000	0.0	0.0512	0.0	0.4430	17.2551	17.7493	0.0	19.3111	36.5662
1	Південний Буг	від витоку до гирла р. Інгва	M5.4.0.01	50	2	28	0.0000	31.4012	0.5130	0.9915	0.0	-0.7909	32.1148	0.0000	0.0	0.0513	0.0	0.4548	15.5853	16.0914	0.0	16.0234	31.6087
2	Південний Буг	від витоку до гирла р. Інгва	M5.4.0.01	50	3	31	0.0000	34.7656	0.5061	0.9915	0.0	0.3264	36.5897	0.0006	0.0	0.0506	0.0	2.1449	18.4132	20.6093	0.0	15.9804	34.3936
3	Південний Буг	від витоку до гирла р. Інгва	M5.4.0.01	50	4	30	0.0000	33.6442	0.7844	0.9915	0.0	-0.3264	35.0937	0.0029	0.0	0.0784	0.0	2.5690	17.8192	20.4695	0.0	14.6241	32.4433
4	Південний Буг	від витоку до гирла р. Інгва	M5.4.0.01	50	5	31	0.0000	34.7656	0.7964	0.9915	0.0	0.0000	36.5536	0.0043	0.0	0.0796	0.0	1.7039	18.4132	20.2011	0.0	16.3525	34.7657
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
391	Бузький лиман	Бузький лиман	M5.4.0.11	95	8	31	165.0040	0.0000	0.1780	2.6817	0.0	0.0000	167.8636	0.0144	0.0	0.0178	0.0	0.9530	0.0000	0.9852	0.0	166.8785	166.8785
392	Бузький лиман	Бузький лиман	M5.4.0.11	95	9	30	159.6813	0.0000	0.1610	2.6817	0.0	0.0000	162.5239	0.0101	0.0	0.0161	0.0	0.7480	0.0000	0.7742	0.0	161.7497	161.7497
393	Бузький лиман	Бузький лиман	M5.4.0.11	95	10	31	165.0040	0.0000	0.1340	2.6817	0.0	0.0000	167.8196	0.0059	0.0	0.0134	0.0	0.9260	0.0000	0.9453	0.0	166.8743	166.8743
394	Бузький лиман	Бузький лиман	M5.4.0.11	95	11	30	159.6813	0.0000	0.1210	2.6817	0.0	0.0000	162.4839	0.0025	0.0	0.0121	0.0	1.2340	0.0000	1.2486	0.0	161.2353	161.2353
395	Бузький лиман	Бузький лиман	M5.4.0.11	95	12	31	165.0040	0.0000	0.1010	2.6817	0.0	0.0000	167.7866	0.0000	0.0	0.0101	0.0	1.1580	0.0000	1.1681	0.0	166.6185	166.6185

Рисунок Г.1 – Створений датасет water-balance-southern-buh

Рік	Січень	Лютий	Березень	Квітень	Травен
92 unique values	5.77 178	4.96 199	19.6 323	18.1 586	12.1
1925	14.8	33.6	46.5	30.9	28.2
1926	21.3	28.6	139	96.8	87.7
1927	25.8	27.7	134	55.2	43.7
1928	10.8	63.1	68	143	36.1
1929	24.4	25.2	69.8	479	77.3
1930	7	52.6	52.6	60.6	63.9
1931	13	14.5	83.7	216	62.7
1932	36.8	23.4	25.8	405	64.6
1933	24.1	16.8	188	59.7	37.1
1934	24.1	45.7	175	42.9	12.4
1935	13.9	20.2	247	78.9	34.7
1936	21.1	22.8	37.8	29.3	14.2
1937	37.7	25.9	246	53	126
1938	31.6	58.7	131	57.4	33.1

Рисунок Г.2 – Створений датасет gidropost_trostyanchik

Рік	Січень	Лютий	Березень	Квітень	Травен
92 unique values	2.02 – 32.5	2.19 – 37.9	3.14 – 107	4.41 – 159	1.16
1925	2.55	5.79	8.96	5.98	5.45
1926	5	6.5	15	32.4	14.4
1927	8.19	5.05	32.8	15.9	9.69
1928	5.17	20.1	19.1	37.4	12.3
1929	6.47	7.22	33.1	112	20.6
1930	4.95	9.05	19	16.7	17.6
1931	7.26	7.14	26.3	32.6	15.3
1932	10.8	3.93	5.98	146	8.66
1933	3.33	3.63	44.7	7.6	3.33
1934	4.56	11.9	52.9	6.64	3.59
1935	3.04	10.1	65.4	16.3	5.74
1936	3.31	2.49	4.93	4.6	1.57
1937	5.56	4.67	54.6	11.1	34.5
1938	6.76	8.79	32.2	13.3	9.7

Рисунок Г.3 – Створений датасет gidropost_lelitka

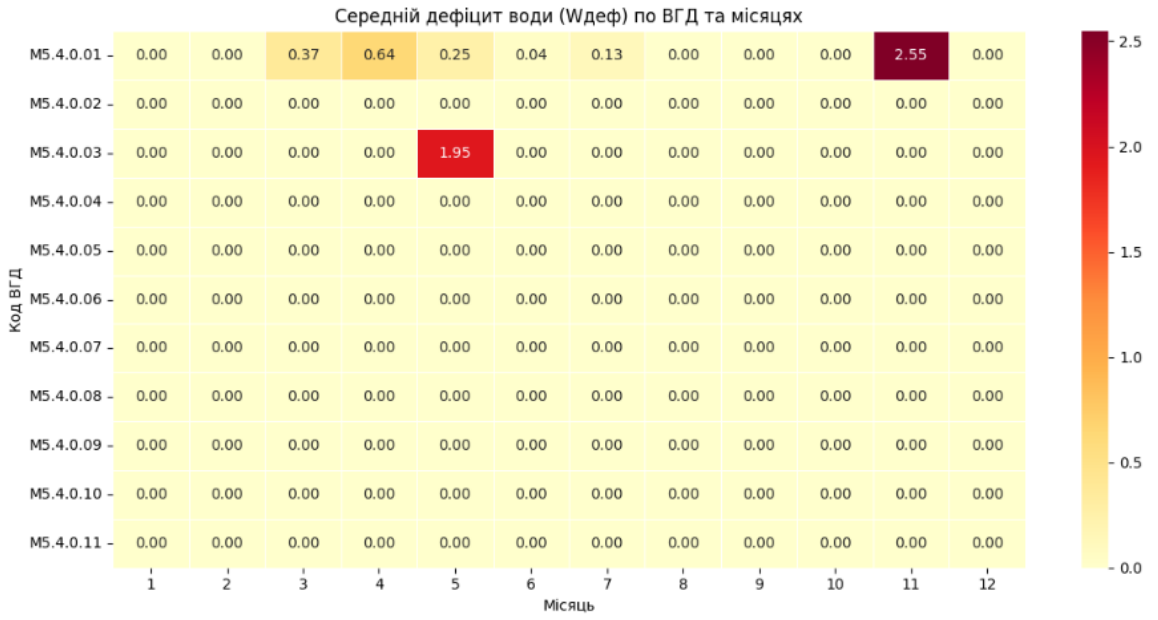

Рисунок Г.4 – Теплова карта дефіциту води

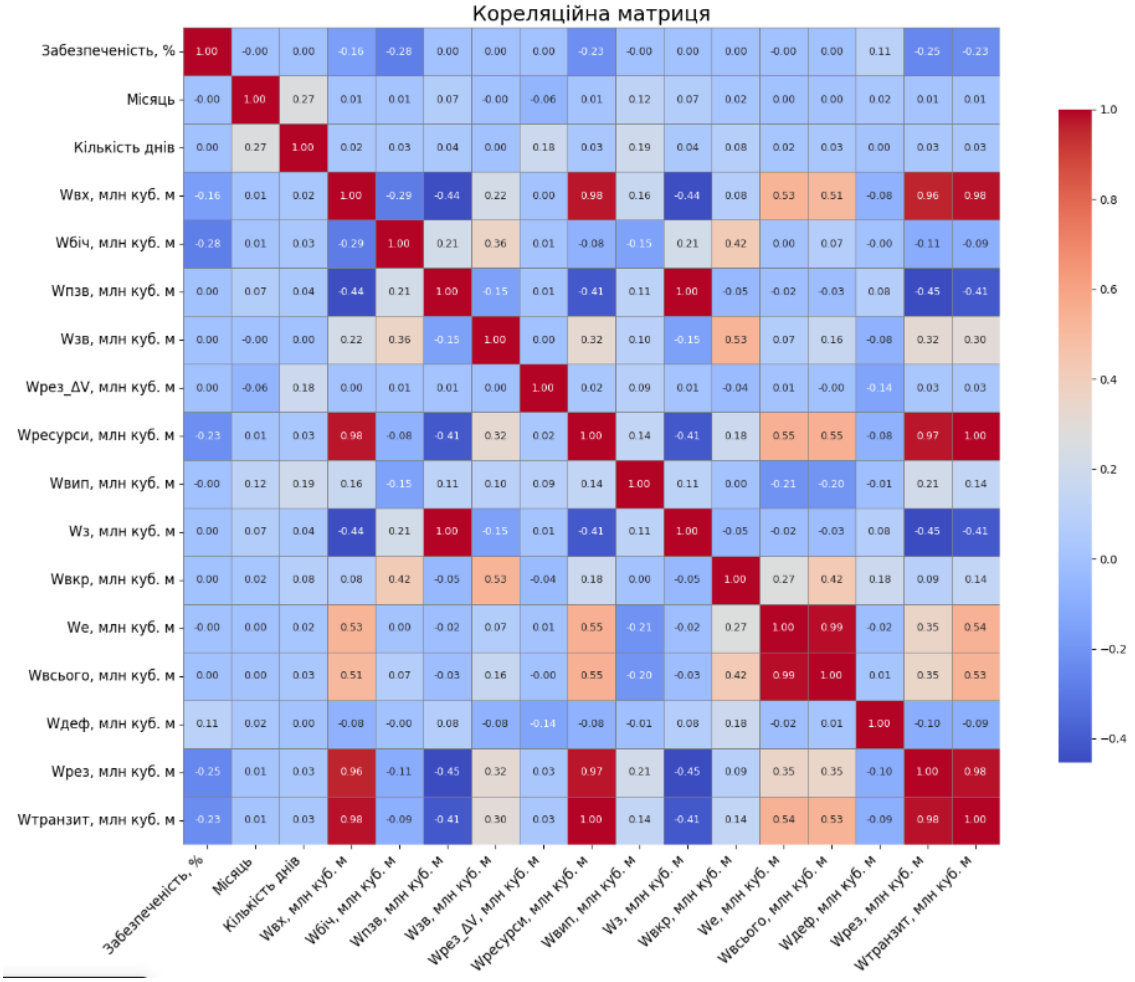

Рисунок Г.5 – Кореляційна матриця між числовими ознаками

Рисунок Г.6 – Карта дефіциту води

Рисунок Г.7 – Карта резерву води

Рисунок Г.8 – Блок-схема алгоритму інформаційної технології аналізу водогосподарських балансів ділянок басейну Південного Бугу

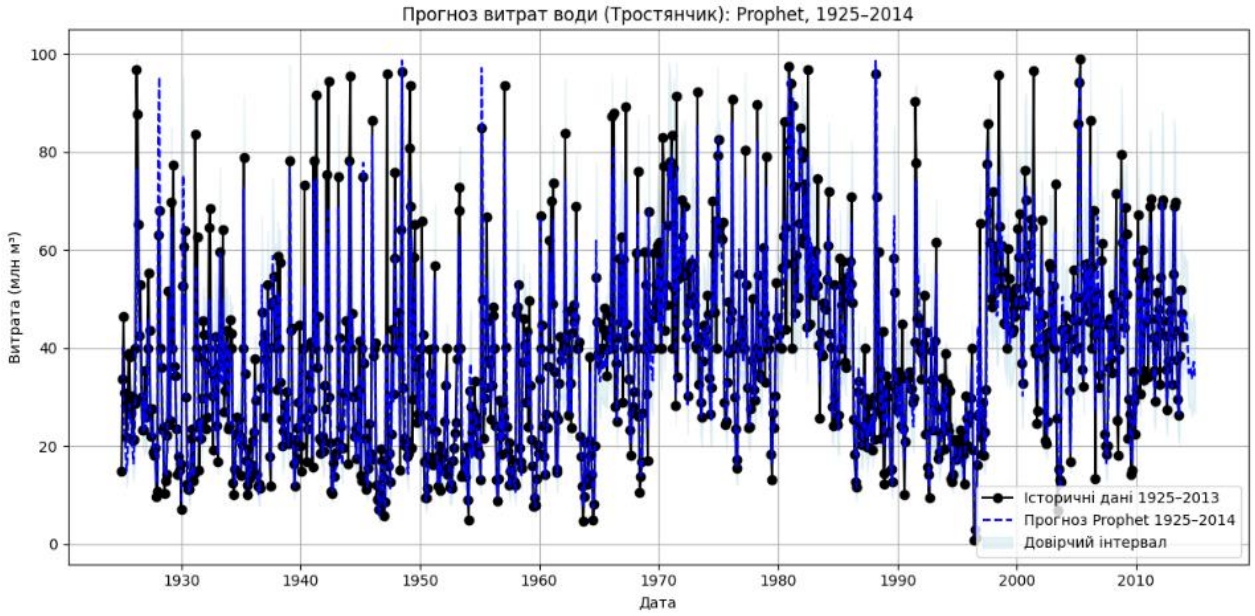

Рисунок Г.9 (А) – Результат прогнозування витрат води за найкращою моделлю Prophet для ділянки с.Тростянчик

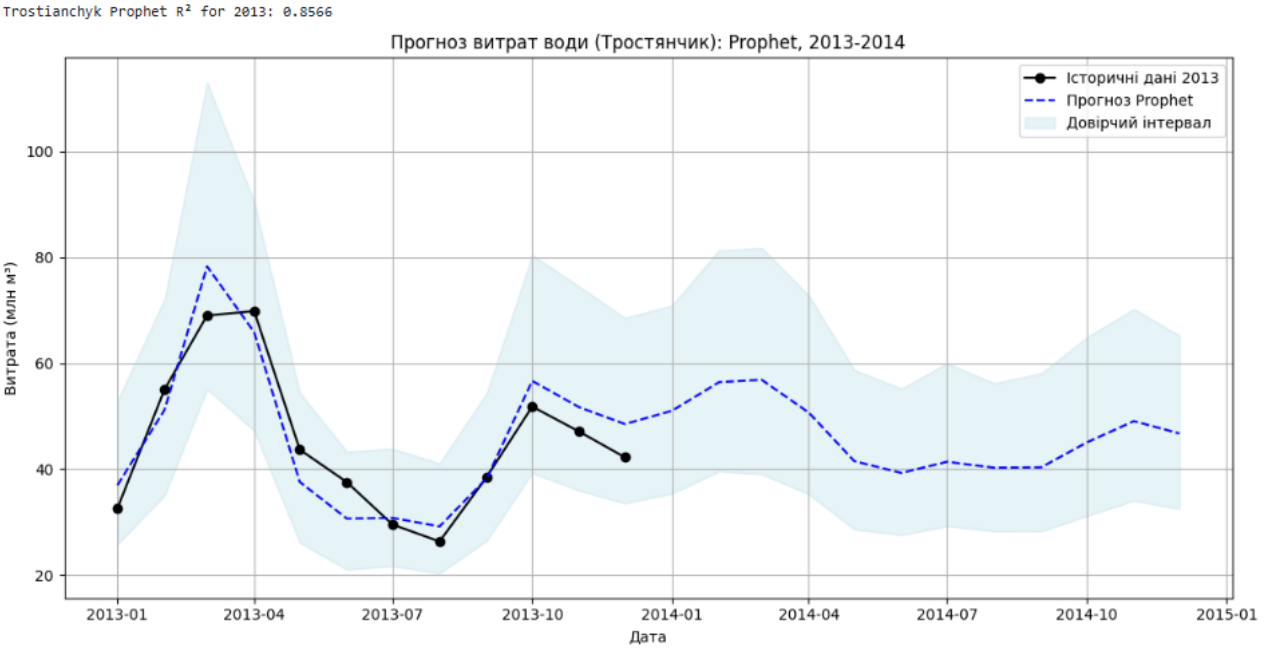

Рисунок Г.9 (Б) – Результат прогнозування витрат води за найкращою моделлю Prophet для ділянки с.Тростянчик

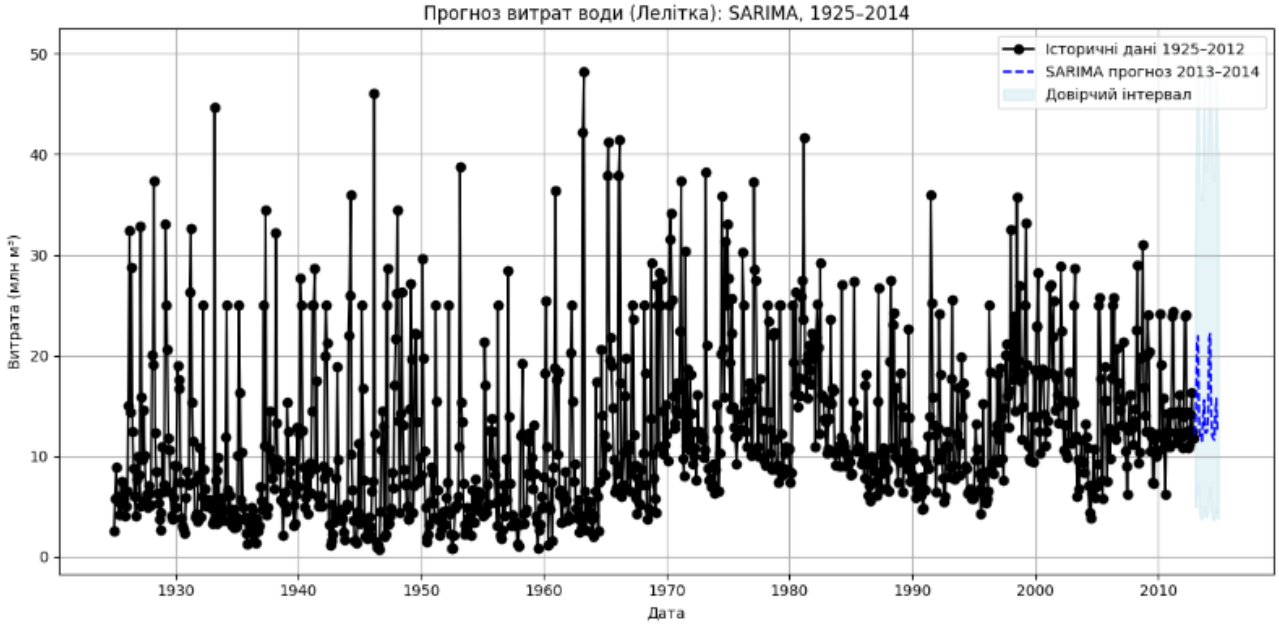

Рисунок Г.10 (А) – Результат прогнозування витрат води за найкращою моделлю SARIMA для ділянки с.Лелітка

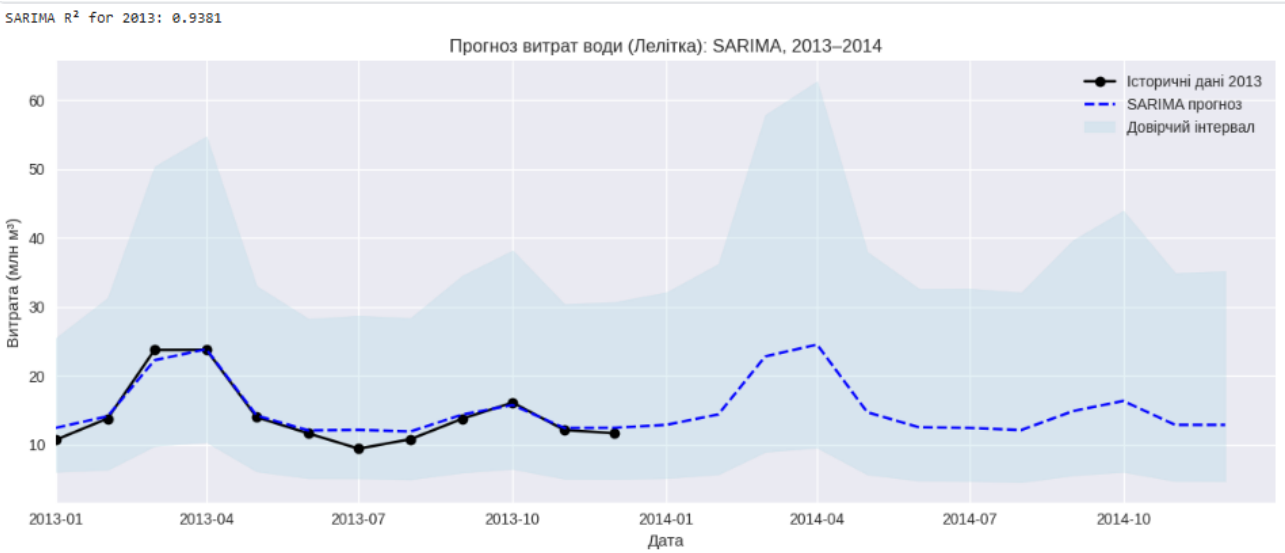

Рисунок Г.10 (Б) – Результат прогнозування витрат води за найкращою моделлю SARIMA для ділянки с.Лелітка