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

Бакалаврська кваліфікаційна робота на тему:

«СИСТЕМНИЙ АНАЛІЗ ВОДОГОСПОДАРСЬКИХ БАЛАНСІВ ДІЛЯНОК БАСЕЙНУ ДНІСТРА»

Виконала: студентка 4 курсу, групи СА-21б
спеціальності 124 – «Системний аналіз»
_____ Анна ГАЙОВИЧ
Керівник: к.т.н., доц. каф. САІТ
_____ Євгеній КРИЖАНОВСЬКИЙ
«__08__» _____06_____ 2025 р.
Рецензент: к.т.н., ас. кафедри КН
_____ Ілона БОГАЧ
«__11__» _____06_____ 2025 р.

Допущено до захисту
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«__09__» _____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 124 – «Системний аналіз»
Освітньо-професійна програма – Системний аналіз

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
д.т.н., проф. Віталій МОКІН
"28" 05 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ
Гайович Анні Михайлівні

1. Тема роботи: «Системний аналіз водогосподарських балансів ділянок басейну Дністра»
керівник роботи: Крижановський Є. М., к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «20» 03 2025 року № 97
2. Строк подання студентом роботи «31» 05 2025 року
3. Вихідні дані до роботи:
1) Дані про водогосподарський баланс для району басейну річки Дністер;
2) Дані автоматичних гідрологічних постів;
3) Шейп-файли водогосподарських ділянок річкових басейнів України.
4. Зміст текстової частини:
1) Загальна характеристика об'єкту досліджень;
2) Підготовка даних, розвідувальний аналіз та інженерія ознак;
3) Прогнозування рівня води методами машинного навчання.
5. Перелік ілюстративного матеріалу:
1) Кореляційна матриця;
2) Візуалізація проблемних ділянок із дефіцитами води;
3) Графік рівнів води у р. Дністер за 2024 рік;
4) Таблиця порівняння оцінювання точності прогнозування моделей;
5) Графік прогнозованих даних.

6. Консультанти розділів роботи:

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

7. Дата видачі завдання «28» 05 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	01.04.2025	20.04.2025	виконано
2	Порівняльний аналіз існуючих аналітичних досліджень	20.04.2025	30.04.2025	виконано
3	Вибір оптимальних інформаційних технологій	01.05.2025	10.05.2025	виконано
4	Системний аналіз водогосподарський балансів ділянок басейну Дністра для підтримки прийняття рішень щодо управління дефіцитом води	10.05.2025	20.05.2025	виконано
5	Оформлення матеріалів до захисту БКР	20.05.2025	30.05.2025	виконано

Студент Анна ГАЙОВИЧ

Керівник роботи Євгеній КРИЖАНОВСЬКИЙ

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 82 сторінок формату А4, на яких є 49 рисунків, список використаних джерел містить 34 найменування.

Метою роботи є системний аналіз водогосподарських балансів ділянок басейну Дністра, включно з візуалізацією, аналізом дефіциту водних ресурсів та прогнозуванням рівня води із використанням моделей машинного навчання.

У роботі проведено системний аналіз водогосподарських балансів ділянок басейну Дністра. Здійснено структуризацію гідрологічних даних, побудовано візуалізації дефіциту води з урахуванням рівня забезпеченості, а також реалізовано прогнозування рівня води за допомогою моделей машинного навчання, зокрема Prophet, ARIMA та мультифакторних регресійних моделей. Проведений системний аналіз водогосподарських балансів ділянок басейну Дністра дозволяє виявляти просторові й часові закономірності водного дефіциту та забезпечує інформаційну підтримку управлінських рішень на рівні басейнового планування.

Ключові слова: системний аналіз, водогосподарський баланс, гідрологічні дані, дефіцит води, машинне навчання, прогнозування, басейн Дністра.

ABSTRACT

The bachelor’s thesis consists of 82 A4 pages, which contain 49 figures, the list of sources used contains 34 titles.

The aim of the work is to conduct a systems analysis of water management balances in the Dniester River basin sections, including visualization, analysis of water resource deficits, and forecasting of water levels using machine learning models.

The paper presents a systems analysis of the water management balances of the Dniester River basin sections. The work includes the structuring of hydrological data, the construction of visualizations of water deficits with consideration of supply reliability, as well as the implementation of water level forecasting using machine learning models such as Prophet, ARIMA, and multifactor regression models. The conducted systems analysis enables the identification of spatial and temporal patterns of water deficit provides information support for decision-making in basin-level water management.

Key words: systems analysis, water management balance, hydrological data, water deficit, machine learning, forecasting, Dniester basin.

ЗМІСТ

ВСТУП

Актуальність теми. Забезпечення раціонального використання водних ресурсів є ключовим викликом сучасного природокористування, особливо в умовах зміни клімату, зростання водоспоживання та антропогенного навантаження на річкові басейни. Басейн річки Дністер має велике екологічне та соціально-економічне значення для західних і південних регіонів України, забезпечуючи водопостачання для населення, сільського господарства та промисловості. У той же час, у межах басейну спостерігаються періодичні дефіцити водних ресурсів на окремих ділянках, які можуть призводити до порушення екологічного балансу, обмеження водокористування та конфліктів між споживачами. Особливої уваги потребує проблема системного управління водогосподарськими балансами з урахуванням просторово-часової мінливості, сезонних факторів, гідрологічної забезпеченості та можливостей впливу на керовані показники (наприклад, забори і скидання води). У зв’язку з цим актуальним є завдання побудови інструментів аналітики та прогнозування, які дозволяють виявити критичні зони, оцінити динаміку дефіциту води та підтримати прийняття обґрунтованих управлінських рішень у басейновому масштабі.

Мета і задачі дослідження. Метою дослідження є системний аналіз водогосподарських балансів ділянок басейну Дністра, включно з візуалізацією, аналізом дефіциту водних ресурсів та прогнозуванням рівня води із використанням моделей машинного навчання.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

– охарактеризувати об’єкт дослідження як складну просторово-часову систему з багатьма взаємопов’язаними параметрами;

– здійснити збір, структуризацію та підготовку вихідних гідрологічних даних для подальшого аналізу;

– реалізувати візуалізацію дефіциту та резервів води на основі даних про водогосподарський баланс з урахуванням рівня забезпеченості;

– побудувати моделі машинного навчання для прогнозування рівня води та порівняти їхню точність;

– сформулювати висновки щодо просторових і часових закономірностей дефіциту та можливих напрямків оптимізації водогосподарського балансу.

Об’єктом дослідження є водогосподарські ділянки басейну річки Дністер та їх баланс водних ресурсів.

Предметом дослідження є методи аналізу, візуалізації та прогнозування дефіциту водних ресурсів на основі спостережуваних гідрологічних даних із використанням аналітичних засобів і моделей машинного навчання.

Публікації. За результатами даної роботи була зроблена доповідь на тему «Системний аналіз водогосподарських балансів ділянок басейну Дністра» на Міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» з публікацією тез [1].

1. ЗАГАЛЬНА ХАРАКТЕРИСТИКА ОБ'ЄКТУ ДОСЛІДЖЕНЬ

1.1 Опис об'єкта досліджень

Об'єктом дослідження є водогосподарська система ділянок басейну Дністра (рис. 1.1), яка розглядається як просторово розподілена динамічна система з великою кількістю взаємопов'язаних елементів. Така система включає в себе сукупність гідрологічно пов'язаних водогосподарських ділянок (ВГД), розташованих уздовж течії річки та її приток, кожна з яких має власні характеристики, режим водокористування, джерела водопостачання та навантаження. З точки зору системного аналізу, вона є багаторівневою, ієрархічною структурою, в якій взаємодіють природні, технічні та управлінські компоненти.

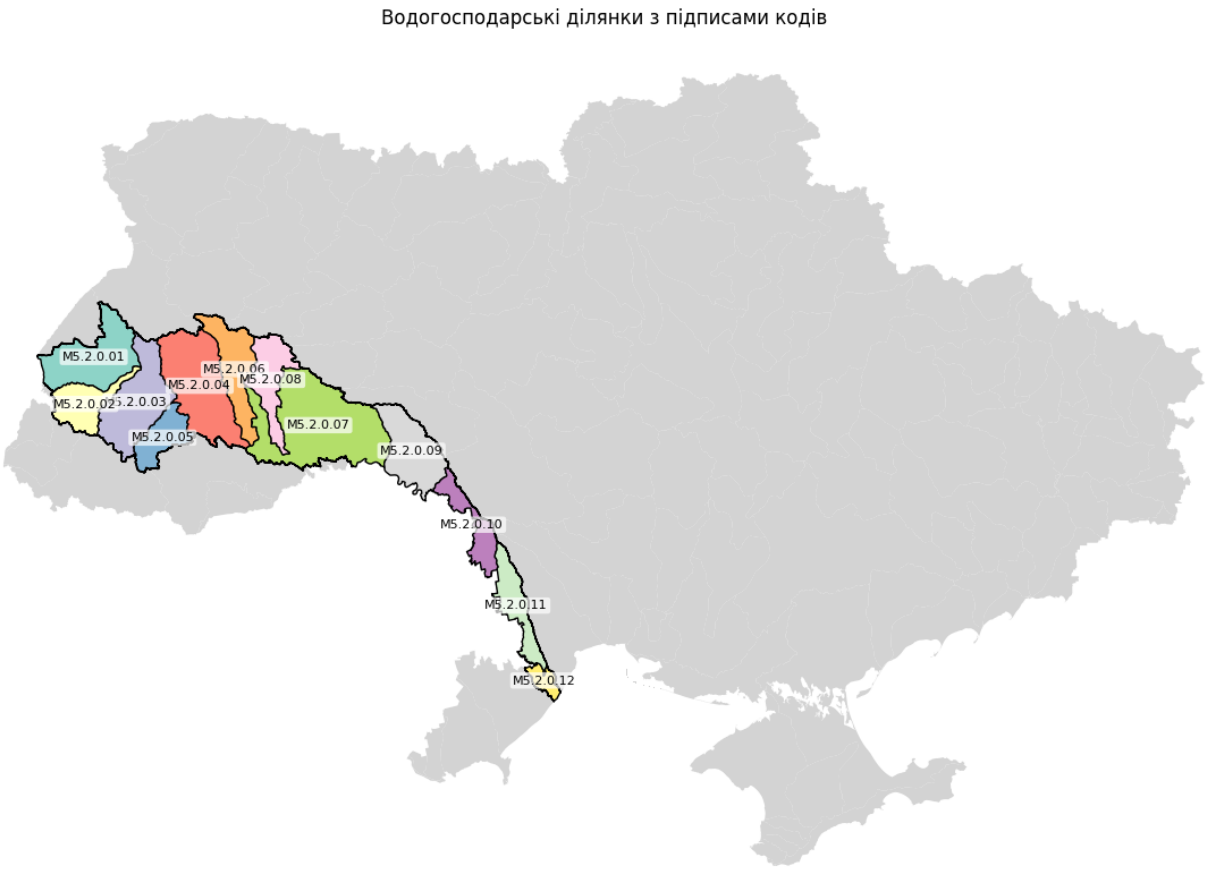


Рисунок 1.1 – Водогосподарські ділянки басейну Дністра

Транскордонний річковий басейн Дністра розташований на території трьох країн: України, Республіки Молдова та Республіки Польщі. Загальна довжина Дністра становить 1362 км, в межах України – 662 км (довжина українсько-молдавської ділянки становить 225 км). Площа водозбору на території України – 53,9 тис. км^2. Район басейну річки (РБР) Дністра покриває 8,7% території України. Район басейну Дністра охоплює територію 7 областей України (Львівська, Івано-Франківська, Чернівецька, Тернопільська, Хмельницька, Вінницька та Одеська). Гідрографічна мережа РБР Дністра включає 499 річок із площею водозабору більше 10 км^2 та 33 водосховища [2]. Така географічна структура обумовлює високу складність координації водогосподарської діяльності між регіонами та підвищує актуальність створення аналітичних інструментів для просторового моделювання.

Структура водогосподарського балансу включає прибуткову (П) та витратну (В) частини, а також результат водогосподарського балансу. Результат водогосподарського балансу характеризується наявністю резервів (П≥В) або дефіцитів (П<В) стоку.

Розрахунок водогосподарського балансу водогосподарської ділянки (далі – ВГД) здійснюється за такою формулою (в одиницях об’єму води за розрахунковий період):

$$ВГБ = W_{вх} + W_{біч} + W_{пзв} + W_{зв} + W_{дот} \pm \Delta V - W_{вип} - W_{ф} - W_{з} - W_{пер} - W_{вкр} - W_{е},$$

де ВГБ – водогосподарський баланс;

$W_{вх}$ – об’єм стоку, що надходить за розрахунковий період з розташованих вище ВГД;

$W_{біч}$ – об’єм стоку, що формується на розрахунковій ВГД (бічний приплив);

$W_{пзв}$ – об’єм водозабору із підземних водних об’єктів;

$W_{зв}$ – об’єм зворотних вод на розрахунковій ВГД;

$W_{дот}$ – дотаційний об’єм води на ВГД (зовнішні та внутрішньобасейнові перекидання);

$\pm \Delta V$ – спрацювання (+), наповнення (-) ставків та водосховищ;

$W_{вип}$ – втрати на додаткове випаровування та льодоутворення з водосховищ (з урахуванням повернення води від розтавання льоду);

$W_{ф}$ – фільтраційні втрати з водосховищ;

$W_{з}$ – зменшення стоку річки, викликане забором гідравлічно зв'язаних з нею підземних вод;

$W_{пер}$ – перекидання частини стоку за межі розрахункової ВГД;

$W_{вкр}$ – забір поверхневих вод;

$W_{е}$ – мінімальний екологічний стік у замикаючому створі ВГД [3].

Система характеризується складною внутрішньою структурою: вона включає як вертикальні (басейнові) взаємозв’язки, так і горизонтальні – між сусідніми ділянками. Вплив змін на одній ВГД (наприклад, збільшення водозабору або зменшення зворотних скидань) може суттєво відобразитися га доступності води для розташованих нижче ділянок. Такий ефект просторової взаємозалежності ускладнює управління системою і вимагає впровадження моделей, що враховують множинність чинників і їхню взаємодію.

Особливістю досліджуваної системи є поєднання керованих і некерованих елементів. До першої групи належать такі параметри, як ліміти водозабору, режим експлуатації водосховищ, рівень зворотних вод, які можуть регулюватися адміністративно або технічно. До некерованих належать кліматичні та природні чинники – опади, температура повітря, природне випаровування, сезонна зміна стоку, які вносять невизначеність у формування балансу. Це поєднання визначає складність побудови систем підтримки прийняття рішень і підвищує значущість аналітичних моделей, які дозволяють досліджувати сценарії, оцінювати ризики та планувати адаптивні заходи управління.

Таким чином, водогосподарська система басейну Дністра виступає типовим прикладом складного об’єкта системного аналізу, в якому необхідна інтеграція математичного моделювання, просторової аналітики, аналізу даних та інформаційних технологій для підтримки прийняття ефективних управлінських рішень у галузі водокористування.

1.2 Огляд існуючих аналітичних досліджень

У сучасних умовах зміни клімату та підвищеного антропогенного навантаження потреба в ефективному управлінні водними ресурсами зростає. Це особливо актуально для великих річкових басейнів, де просторово-часова динаміка гідрологічних характеристик поєднується зі складною структурою водокористування.

Одним із пріоритетних напрямків є побудова моделей прогнозування водогосподарського балансу для виявлення ризиків дефіциту та оптимізації розподілу водних ресурсів у міжмісячному та міжбасейновому розрізі. заслуговує на увагу науково-дослідна робота «Побудова та прогнозування водогосподарського балансу водогосподарської ділянки М5.1.4.45 р. Горинь від витоку до кордону Хмельницької та Рівненської областей» [4]. У шостому розділі цієї роботи наведено результати побудови прогнозів водогосподарського балансу на основі багаторічних гідрологічних спостережень з урахуванням екологічних обмежень. Зокрема, проаналізовано динаміку резервів водних ресурсів та транзитного стоку на нижче розташовану ВГД при трьох рівнях забезпеченості: 50%, 75% та 95% – за умов мінімального екологічного стоку 6,0 $м^3/с$. На відповідних графіках (рис. 1.2 – 1.4) демонструється сезонний розподіл доступного обсягу води, що дозволяє виявити потенційно критичні місяці та сформулювати рекомендації щодо зміни режиму водокористування.

Рисунок 1.2 – Графік прогнозних резервів водних ресурсів та транзиту стоку на розташовану нижче ВГД для 50% забезпеченості при мінімальній екологічній витраті 6,0 м^3/с

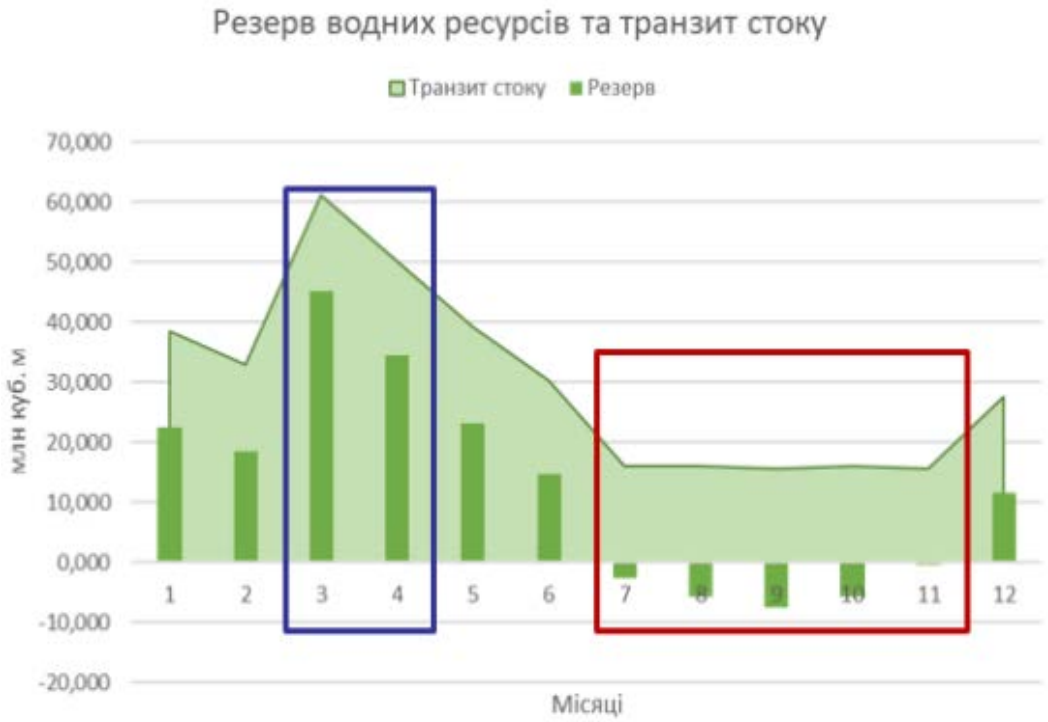


Рисунок 1.3 – Графік прогнозних резервів водних ресурсів та транзиту стоку на розташовану нижче ВГД для 75% забезпеченості при мінімальній екологічній витраті 6,0 м^3/с

Рисунок 1.4 – Графік прогнозних резервів водних ресурсів та транзиту стоку на розташовану нижче ВГД для 95% забезпеченості при мінімальній екологічній витраті 6,0 $м^3/с$

Висновки роботи підкреслюють важливість поєднання розрахункових сценаріїв з аналітичним прогнозуванням для підтримки прийняття рішень у сфері водного менеджменту, зокрема для таких об’єктів, як Хмельницька АЕС, де стабільність водопостачання має критичне значення. Запропонований підхід до моделювання балансу є адаптивним і потенційно придатним для застосування в інших басейнах, зокрема басейні Дністра.

Ще один напрямок сучасного системного аналізу у водній сфері – прогнозування не лише кількісних характеристик, а й показників якості води. У публікації В. Б. Мокіна та колективу авторів «Передбачення показників якості води у річці з використанням моделей часових рядів і методів бібліотеки Prophet» [5] запропоновано підхід до прогнозування рівня біохімічного споживання кисню (БСК5) на річці Південний Буг на основі даних автоматизованих спостережень за 1994 – 2019 роки. Прогнозування здійснено за допомогою бібліотеки Prophet мовою Python, що дало змогу побудувати моделі сезонності із залученням рядів Фур’є порядку 2 та 6 (рис. 1.5 – 1.6).

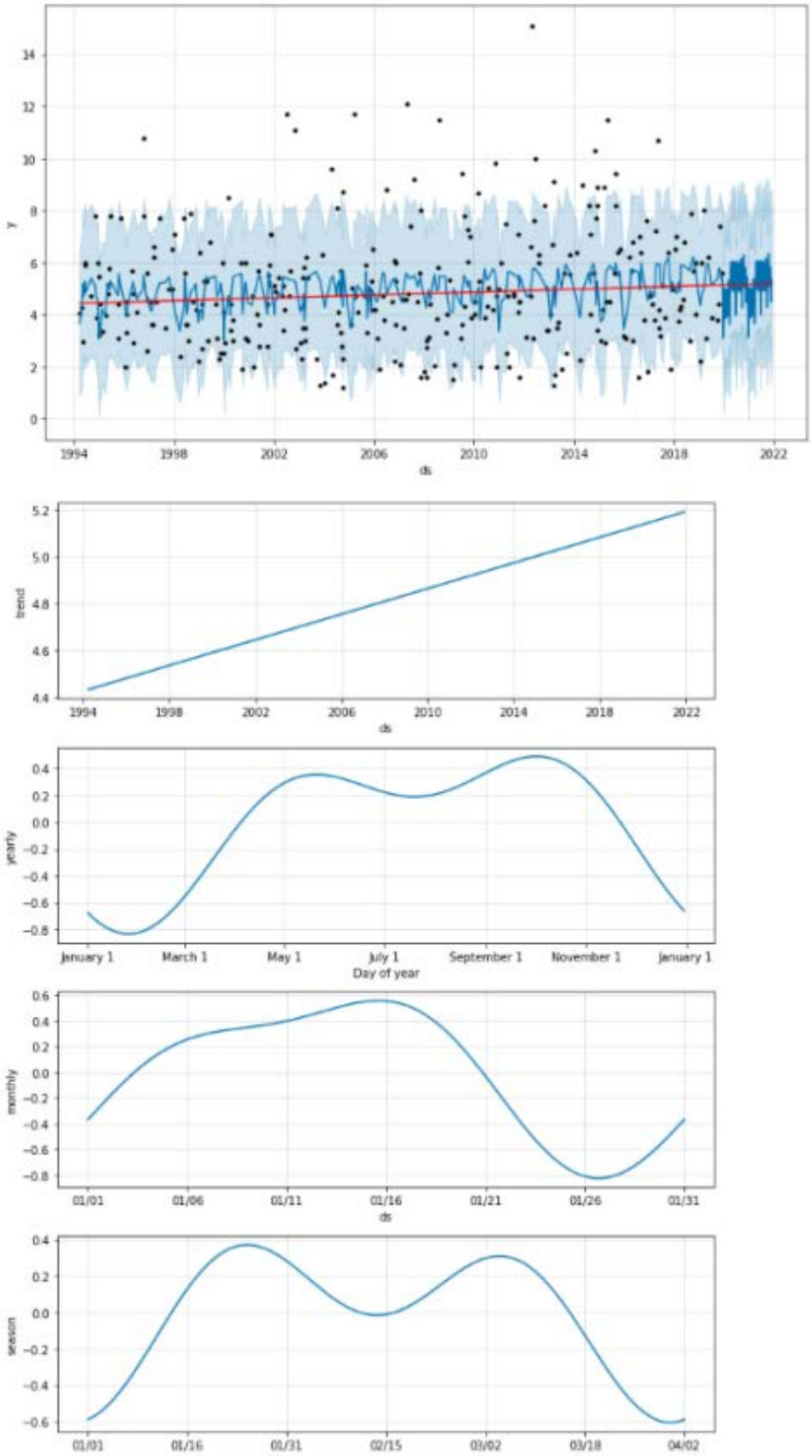


Рисунок 1.5 – Передбачення на 2 роки значення БСК5 біля водозабору КП «Вінницяводоканал» за даними 25 років з лінійним трендом з 4-ма точками зміни тренду та урахуванням сезонності (річної, місячної та по порах року) на основі рядів Фур’є з кількістю членів (порядком) 2

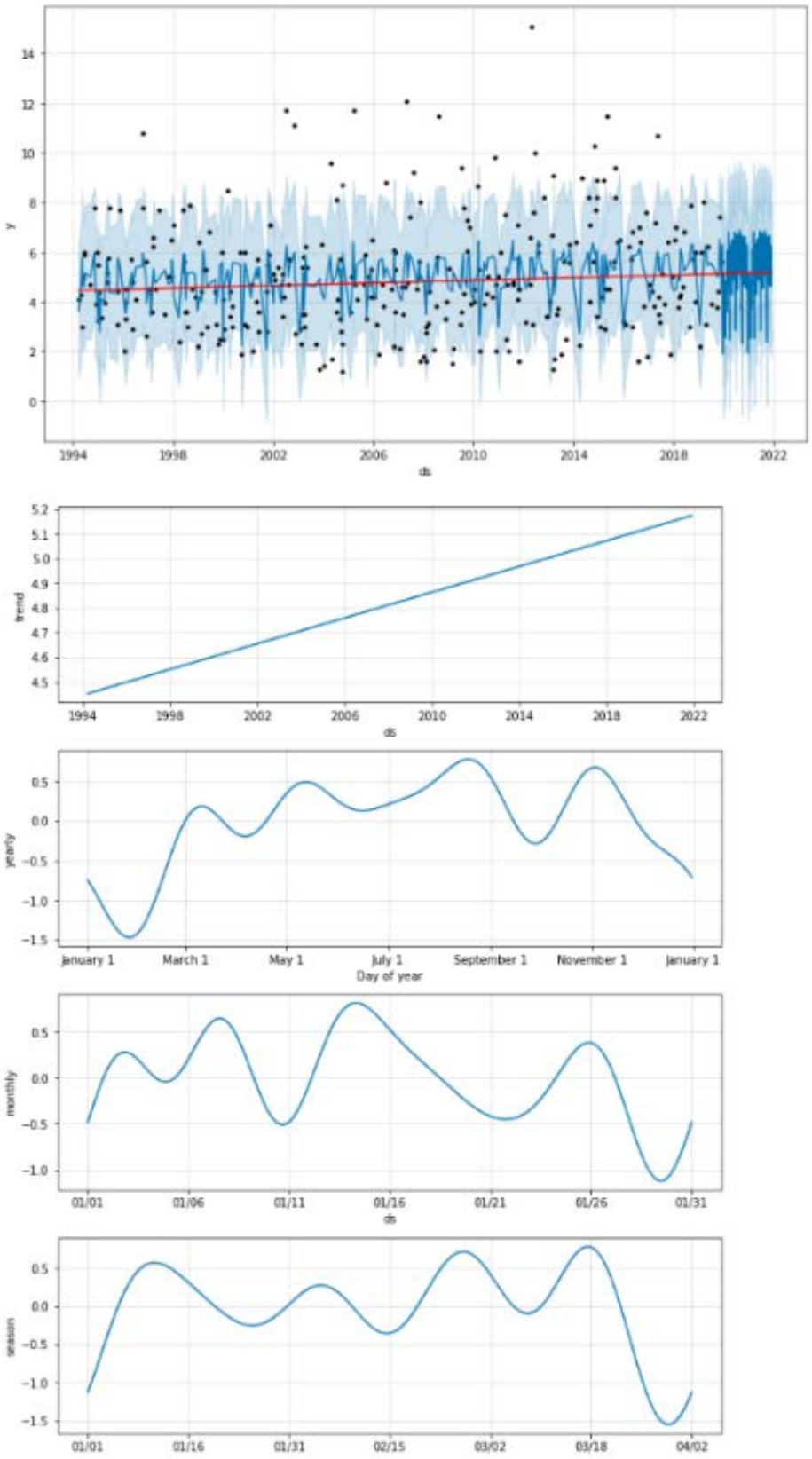


Рисунок 1.6 – Передбачення на 2 роки значення БСК5 біля водозабору КП «Вінницяводоканал» за даними 25 років з лінійним трендом з 4-ма точками зміни тренду та урахуванням сезонності (річної, місячної та по порах року) на основі рядів Фур’є з кількістю членів (порядком) 6

Результати показали чіткий зростаючий тренд забруднення води, наявність регулярних сезонних коливань та відносну стабільність моделі при правильному виборі сезонних компонент. Автори підкреслюють, що адитивна модель із порядком Фур'є 6 є найкращою з точки зору інтерпретації та точності, а сам метод Prophet добре себе зарекомендував у контексті гідроекологічного моделювання. Прогнозування таких показників, як БСК5, є важливим для оперативного реагування на екологічні загрози та побудови довгострокових стратегій забезпечення безпеки водопостачання.

Обидва аналізовані джерела ілюструють значущість використання сучасних цифрових методів, аналітичних платформ і бібліотек для комплексного аналізу гідрологічних процесів. Незалежно від того, чи йдеться про прогнозування дефіциту та резервів води, чи про динаміку забруднення, вирішальне значення мають якість вхідних даних, правильний вибір моделі та її адаптація до специфіки об'єкта дослідження.

1.3 Вибір мови програмування та середовища розробки

Для проведення системного аналізу водогосподарських балансів басейну річки Дністер було обрано мову програмування Python [6]. Це одна з найпопулярніших мов серед дослідників, інженерів та аналітиків, яка поєднує простоту синтаксису з великою функціональністю. Python активно використовується у сферах аналізу даних, статистичного моделювання, геоінформатики, екології, управління природними ресурсами та автоматизованого прогнозування. Її відкритий характер, постійна підтримка з боку наукової спільноти та наявність великої кількості спеціалізованих бібліотек роблять Python і детальним вибором для реалізації міждисциплінарних досліджень, зокрема в галузі водного господарства.

У процесі системного аналізу використовувались сучасні засоби Python для опрацювання гідрологічних даних, просторового аналізу, візуалізації та побудови моделей прогнозування. Базовою платформою для роботи з табличними даними стала бібліотека pandas [7], яка забезпечує зручні функції фільтрації, агрегування,

перетворення, заповнення пропущених значень, а також об’єднання даних з різних джерел. Для побудови візуалізацій використовувались бібліотеки matplotlib [8] та seaborn [9], які дозволили створити графіки сезонної динаміки дефіцитів та резервів води, а також проаналізувати розподіли та кореляції між ключовими показниками.

Важливою частиною здійснення системного аналізу є просторовий компонент. Для роботи з геоданими було використано бібліотеку geopandas [10], яка дозволила відображати водогосподарські ділянки басейну Дністра на мапах, пов’язати їх із відповідними показниками водного балансу та створити інтерактивні картограми. Такий підхід дав змогу не лише аналізувати числові значення, а й візуалізувати просторову структуру ризиків, виявити найуразливіші ділянки та оцінити гідрологічну ситуацію в басейновому контексті.

Для розв’язання задачі прогнозування рівня води було побудовано ряд моделей машинного навчання. Зокрема, використано як класичні статистичні підходи – Prophet і ARIMA_auto, так і сучасні моделі машинного навчання: Support Vector Machines (SVM), Linear SVR, XGBoost Regressor, Linear Regression, Random Forest Regressor, Bagging Regressor, KNeighbors Regressor та MLP Regressor [11 – 21]. Усі моделі було навчено на реальних гідрологічних даних за 2024 рік. Результати прогнозування оцінювались за допомогою метрик RMSE (корінь середньоквадратичної помилки) та MAPE (середня абсолютна відносна похибка) [22], що дало змогу об’єктивно порівняти точність різних алгоритмів та визначити найпридатніші підходи для короткострокового прогнозу рівня води.

Середовищем розробки обрано хмарну платформу Kaggle Notebook [23], яка поєднує в собі функціональність Jupyter Notebook, гнучке середовище програмування на Python та відкритий доступ до публічних датасетів. Платформа надає змогу запускати код у хмарі без потреби встановлення додаткового програмного забезпечення, що особливо зручно для командної роботи, публікації результатів і забезпечення відтворюваності дослідження. Завдяки попередньо встановленим бібліотекам, підтримці GPU-прискорення та інтерактивному інтерфейсу, Kaggle забезпечує повноцінне середовище для реалізації навіть складних обчислювальних задач.

Інтеграція всіх зазначених інструментів дозволила здійснити системний аналіз водогосподарський балансів ділянок басейну Дністра, який охоплює всі ключові етапи – від завантаження та очищення даних до їх візуального аналізу, просторового відображення, побудови прогнозів та оцінювання моделей. Такий підхід дозволив не лише виявити закономірності у динаміці водогосподарський показників, а й створити аналітичний інструмент, придатний до подальшого масштабування, зокрема для включення в системи підтримки прийняття рішень у сфері управління водними ресурсами.

1.4 Висновки

У першому розділі охарактеризовано водогосподарську систему ділянок басейну річки Дністер як складну динамічну систему з просторовою та сезонною варіативністю. Розглянуто основні складові водогосподарського балансу (сток, бічний приплив, водозабори, зворотні води, екологічний стік тощо), а також чинники, які впливають на дефіцит води.

Проведено огляд актуальних аналітичних досліджень у сфері прогнозуваня водогосподарських балансів і якості води з використанням бібліотеки Prophet. Обґрунтовано вибір мови програмування Python та хмарного середовища Kaggle Notebooks як платформи для реалізації моделювання, візуалізації та просторового аналізу даних.

Також розглянуто набір моделей для прогнозування рівня води: Prophet, ARIMA_auto, Support Vector Machines (SVM), Linear SVR, XGBoost Regressor, Linear Regression, Random Forest Regressor, Bagging Regressor, KNeighbors Regressor та MLP Regressor. Дані моделі дозволяють досліджувати часову динаміку рівнів води та оцінити точність прогнозування на різних часових відрізках, що створює підґрунтя для формування ефективної системи підтримки прийняття рішень у сфері управління водними ресурсами.

2. ПІДГОТОВКА ДАНИХ, РОЗВІДУВАЛЬНИЙ АНАЛІЗ ТА ІНЖЕНЕРІЯ ОЗНАК

2.1 Робота з необробленими масивами даних, структуризація та формування загального датасету для роботи

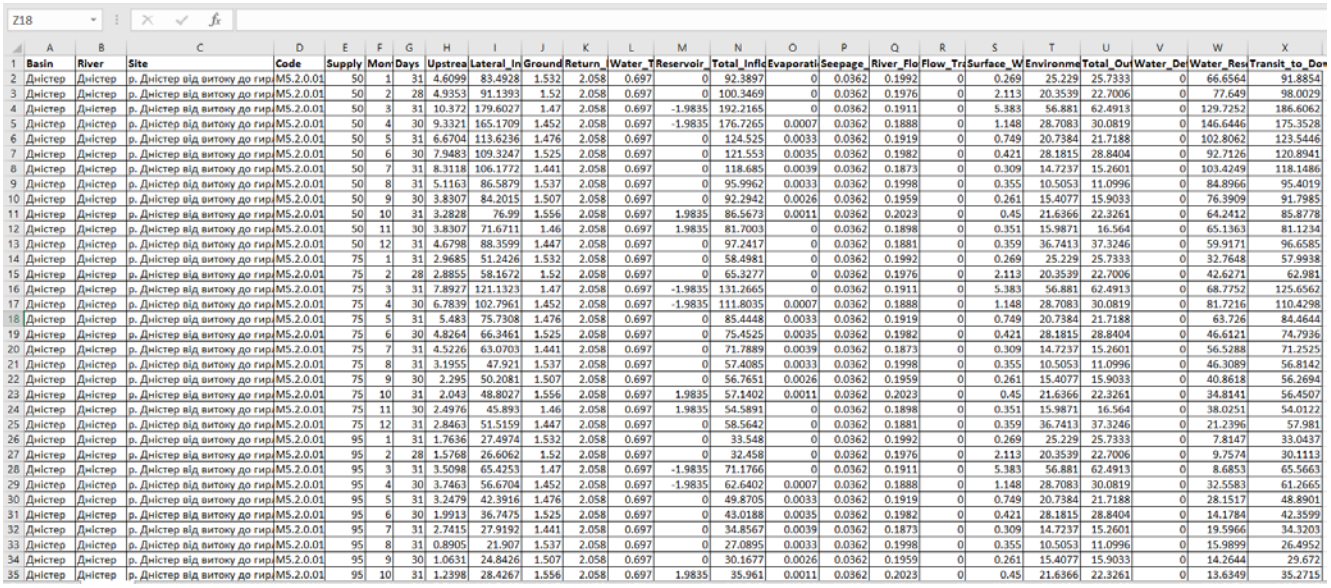
Для проведення системного аналізу водогосподарських балансів басейну Дністра було використано офіційні дані, опубліковані Державним агентством водних ресурсів України у вигляді щомісячного водогосподарського балансу в форматі PDF [24]. З метою подальшої обробки дані були конвертовані у формат XLSX, після чого збережені у вигляді CSV-файлу.

На етапі попередньої обробки було виконано перетворення числових значень до уніфікованого формату з використанням крапки як десяткового роздільника, колонки таблиці було перейменовано з оригінальних довгих назв українською мовою на стислий формат англійською мовою, що спрощує обробку в середовищах програмування та відповідає сучасним вимогам до структурованих даних машинного навчання. Також перевірено коректність структури таблиці, наявність усіх місяців, значень та ідентифікаторів водогосподарських ділянок (ВГД). Нижче наведено вигляд даних до та після обробки (рис. 2.1 – 2.3).

Таблиця 1. Водогосподарський баланс для
р. Дністер від витоку до гирла р. Стрий (код М5.2.0.01)
(назва водогосподарської ділянки)
при 50% забезпеченості стоку

Складові водогосподарського балансу	Розрахункові інтервали часу водогосподарського року (місяці/дні)											
	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII
	31	28	31	30	31	30	31	31	30	31	30	31
I. Прибуткова частина												
1. Об'єм стоку, що надходить за розрахунковий період з розташованих вище ВГД, Wвх, млн куб. м	4,6099	4,9353	10,3723	9,3321	6,6704	7,9483	8,3118	5,1163	3,8307	3,2828	3,8307	4,6798
2. Об'єм стоку, що формується на розрахунковій ВГД (бічний приплив), Wбіч, млн куб. м	83,4928	91,1393	179,6027	165,1709	113,6236	109,3247	106,1772	86,5879	84,2015	76,9900	71,6711	88,3599
3. Об'єм водозабору із підземних водних об'єктів, Wпзв, млн куб. м	1,5320	1,5200	1,4700	1,4520	1,4760	1,5250	1,4410	1,5370	1,5070	1,5560	1,4600	1,4470
4. Об'єм зворотних вод на розрахунковій ВГД, Wзв, млн куб. м	2,0580	2,0580	2,0580	2,0580	2,0580	2,0580	2,0580	2,0580	2,0580	2,0580	2,0580	2,0580
5. Дотаційний об'єм води на ВГД (зовнішні та внутрішньобасейнові перекидання), Wдот, млн куб. м	0,6970	0,6970	0,6970	0,6970	0,6970	0,6970	0,6970	0,6970	0,6970	0,6970	0,6970	0,6970
6. Спрацювання (+), наповнення (-) ставків та водосховищ, ±ΔV, млн куб. м	0,0000	0,0000	-1,9835	-1,9835	0,0000	0,0000	0,0000	0,0000	0,0000	1,9835	1,9835	0,0000
7. Усього по прибутковій частині (наявні ресурси), млн куб. м	92,3897	100,3496	192,2165	176,7265	124,5250	121,5530	118,6850	95,9962	92,2942	86,5673	81,7003	97,2417

Рисунок 2.1 – Приклад вихідних даних балансу у форматі PDF



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	Basin	River	Site	Code	Supply	Mon	Days	Upstrea	Lateral_In	Ground	Return_I	Water_T	Reservoir_	Total_Inflo	Evaporati	Seepage_	River_Flo	Flow_Tra	Surface_W	Environme	Total_Out	Water_De	Water_Res	Transit_to_Dow
2	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	50	1	31	4.6099	83.4928	1.532	2.058	0.697	0	92.3897	0	0.0362	0.1992	0	0.269	25.229	25.7333	0	66.6564	91.8854
3	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	50	2	28	4.9353	91.1393	1.52	2.058	0.697	0	100.3469	0	0.0362	0.1976	0	2.113	20.3539	22.7006	0	77.649	98.0029
4	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	50	3	31	10.372	179.6027	1.47	2.058	0.697	-1.9835	192.2165	0	0.0362	0.1911	0	5.383	56.881	62.4913	0	129.7252	186.6062
5	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	50	4	30	9.3321	165.1709	1.452	2.058	0.697	-1.9835	176.7265	0.0007	0.0362	0.1888	0	1.148	28.7083	30.0819	0	146.6446	175.3528
6	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	50	5	31	6.6704	113.6236	1.476	2.058	0.697	0	124.525	0.0033	0.0362	0.1919	0	0.749	20.7384	21.7188	0	102.8062	123.5446
7	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	50	6	30	7.9483	109.3247	1.525	2.058	0.697	0	121.553	0.0035	0.0362	0.1982	0	0.421	28.1815	28.8404	0	92.7126	120.8941
8	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	50	7	31	8.3118	106.1772	1.441	2.058	0.697	0	118.685	0.0039	0.0362	0.1873	0	0.309	14.7237	15.2601	0	103.4249	118.1486
9	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	50	8	31	5.1163	86.5879	1.537	2.058	0.697	0	95.9962	0.0033	0.0362	0.1998	0	0.355	10.5053	11.0996	0	84.8966	95.4019
10	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	50	9	30	3.8307	84.2015	1.507	2.058	0.697	0	92.2942	0.0026	0.0362	0.1959	0	0.261	15.4077	15.9033	0	76.3909	91.7985
11	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	50	10	31	3.2828	76.99	1.556	2.058	0.697	1.9835	86.5673	0.0011	0.0362	0.2023	0	0.45	21.6366	22.3261	0	64.2412	85.8778
12	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	50	11	30	3.8307	71.6711	1.46	2.058	0.697	1.9835	81.7003	0	0.0362	0.1898	0	0.351	15.9871	16.564	0	65.1363	81.1234
13	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	50	12	31	4.6798	88.3599	1.447	2.058	0.697	0	97.2417	0	0.0362	0.1881	0	0.359	36.7413	37.3246	0	59.9171	96.6585
14	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	75	1	31	2.9685	51.2426	1.532	2.058	0.697	0	58.4981	0	0.0362	0.1992	0	0.269	25.229	25.7333	0	32.7648	57.9938
15	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	75	2	28	2.8855	58.1672	1.52	2.058	0.697	0	65.3277	0	0.0362	0.1976	0	2.113	20.3539	22.7006	0	42.6271	62.981
16	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	75	3	31	7.8927	121.1323	1.47	2.058	0.697	-1.9835	131.2665	0	0.0362	0.1911	0	5.383	56.881	62.4913	0	68.7752	125.6562
17	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	75	4	30	6.7839	102.7961	1.452	2.058	0.697	-1.9835	111.8035	0.0007	0.0362	0.1888	0	1.148	28.7083	30.0819	0	81.7216	110.4298
18	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	75	5	31	5.483	75.7308	1.476	2.058	0.697	0	85.4448	0.0033	0.0362	0.1919	0	0.749	20.7384	21.7188	0	63.726	84.4644
19	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	75	6	30	4.8264	66.3461	1.525	2.058	0.697	0	75.4525	0.0035	0.0362	0.1982	0	0.421	28.1815	28.8404	0	46.6121	74.7936
20	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	75	7	31	4.5226	63.0703	1.441	2.058	0.697	0	71.7889	0.0039	0.0362	0.1873	0	0.309	14.7237	15.2601	0	56.5288	71.2525
21	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	75	8	31	3.1955	47.921	1.537	2.058	0.697	0	57.4085	0.0033	0.0362	0.1998	0	0.355	10.5053	11.0996	0	46.3089	56.8142
22	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	75	9	30	2.295	50.2081	1.507	2.058	0.697	0	56.7651	0.0026	0.0362	0.1959	0	0.261	15.4077	15.9033	0	40.8618	56.2694
23	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	75	10	31	2.043	48.8027	1.556	2.058	0.697	1.9835	57.1402	0.0011	0.0362	0.2023	0	0.45	21.6366	22.3261	0	34.8141	56.4507
24	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	75	11	30	2.4976	45.893	1.46	2.058	0.697	1.9835	54.5891	0	0.0362	0.1898	0	0.351	15.9871	16.564	0	38.0251	54.0122
25	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	75	12	31	2.8463	51.5159	1.447	2.058	0.697	0	58.5642	0	0.0362	0.1881	0	0.359	36.7413	37.3246	0	21.2396	57.981
26	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	95	1	31	1.7636	27.4974	1.532	2.058	0.697	0	33.548	0	0.0362	0.1992	0	0.269	25.229	25.7333	0	7.8147	33.0437
27	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	95	2	28	1.5768	26.6062	1.52	2.058	0.697	0	32.458	0	0.0362	0.1976	0	2.113	20.3539	22.7006	0	9.7574	30.1113
28	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	95	3	31	3.5098	65.4253	1.47	2.058	0.697	-1.9835	71.1766	0	0.0362	0.1911	0	5.383	56.881	62.4913	0	8.6853	65.5663
29	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	95	4	30	3.7463	56.6704	1.452	2.058	0.697	-1.9835	62.6402	0.0007	0.0362	0.1888	0	1.148	28.7083	30.0819	0	32.5583	61.2665
30	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	95	5	31	3.2479	42.3916	1.476	2.058	0.697	0	49.8705	0.0033	0.0362	0.1919	0	0.749	20.7384	21.7188	0	28.1517	48.8901
31	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	95	6	30	1.9913	36.7475	1.525	2.058	0.697	0	43.0188	0.0035	0.0362	0.1982	0	0.421	28.1815	28.8404	0	14.1784	42.3599
32	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	95	7	31	2.7415	27.9192	1.441	2.058	0.697	0	34.8567	0.0039	0.0362	0.1873	0	0.309	14.7237	15.2601	0	19.5966	34.3203
33	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	95	8	31	0.8905	21.907	1.537	2.058	0.697	0	27.0895	0.0033	0.0362	0.1998	0	0.355	10.5053	11.0996	0	15.9899	26.4952
34	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	95	9	30	1.0631	24.8426	1.507	2.058	0.697	0	30.1677	0.0026	0.0362	0.1959	0	0.261	15.4077	15.9033	0	14.2644	29.672
35	Дністер	Дністер	р. Дністер від витоку до гирл	M5.2.0.01	95	10	31	1.2398	28.4267	1.556	2.058	0.697	1.9835	35.961	0.0011	0.0362	0.2023	0	0.45	21.6366	22.3261	0	13.6349	35.2715

Рисунок 2.2 – Вигляд даних балансу після обробки у форматі XLSX

	Basin	River	Site	Code	Supply	Month	Days	Upstream_Inflow	Lateral_Inflow	Groundwater_Intake	...	Evaporation/Ice_Loss
0	Дністер	Дністер	р. Дністер від витоку до гирла р. Стрий	M5.2.0.01	50	1	31	4.6099	83.4928	1.532	...	0.0000
1	Дністер	Дністер	р. Дністер від витоку до гирла р. Стрий	M5.2.0.01	50	2	28	4.9353	91.1393	1.520	...	0.0000
2	Дністер	Дністер	р. Дністер від витоку до гирла р. Стрий	M5.2.0.01	50	3	31	10.3723	179.6027	1.470	...	0.0000
3	Дністер	Дністер	р. Дністер від витоку до гирла р. Стрий	M5.2.0.01	50	4	30	9.3321	165.1709	1.452	...	0.0007
4	Дністер	Дністер	р. Дністер від витоку до гирла р. Стрий	M5.2.0.01	50	5	31	6.6704	113.6236	1.476	...	0.0033

Рисунок 2.3 – Таблиця з даними балансу після зчитування файлу у форматі CSV

Окрім балансових даних по ВГД, аналогічним чином було оброблено гідрологічні спостереження з сайту Укргідрометеоцентру [25] по автоматичному гідропосту «Самбір» за 2024 рік (рис. 2.4 – 2.5)

Таблиця рівнів води за період

Дата/год.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	Max	Min	Average
01.01.2024	215	215	215	214	214	214	214	214	213	213	213	213	213	213	213	213	213	212	212	213	212	212	213	213	215	212	213
02.01.2024	213	213	213	214	215	215	216	216	217	217	217	217	217	217	217	216	216	216	215	215	215	214	214	214	217	213	215
03.01.2024	213	213	213	213	213	212	212	212	212	212	213	213	213	213	213	213	214	214	214	214	213	214	214	213	214	212	213
04.01.2024	213	213	213	214	214	213	213	213	213	213	213	212	212	212	212	212	212	212	212	-	-	-	-	-	214	212	213
05.01.2024	212	212	212	212	212	213	213	214	215	216	217	219	221	223	224	225	225	225	225	225	225	224	224	224	225	212	219
06.01.2024	223	223	223	222	222	222	221	221	221	221	220	220	220	219	219	219	219	218	218	218	218	218	218	218	223	218	220
07.01.2024	218	219	219	221	222	224	225	228	230	233	236	239	242	245	247	248	249	249	249	-	248	247	246	246	249	218	236
08.01.2024	245	244	243	242	242	241	240	239	238	237	237	236	235	235	234	233	233	232	231	230	230	230	229	228	245	228	236
09.01.2024	228	227	227	226	226	225	225	225	225	224	223	222	220	219	217	216	215	214	213	213	213	214	214	214	228	213	220
10.01.2024	215	215	214	214	214	213	212	211	210	209	209	208	208	208	209	209	209	209	208	208	209	209	210	211	215	208	210
11.01.2024	211	212	212	212	212	212	211	210	210	209	209	208	208	208	208	209	209	209	210	210	211	212	214	215	215	208	210
12.01.2024	215	215	215	218	219	217	215	215	214	214	213	213	213	213	212	212	212	212	212	212	211	211	211	210	219	210	214
13.01.2024	210	209	209	208	208	208	207	207	207	207	207	207	207	207	207	207	207	207	207	207	207	-	-	-	210	207	207
14.01.2024	207	207	207	206	206	206	206	206	205	205	205	205	205	205	205	204	204	204	204	204	204	204	204	204	207	204	205
15.01.2024	204	203	203	203	203	203	203	203	203	203	202	202	202	202	202	202	202	202	202	202	202	202	202	201	204	201	202
16.01.2024	201	201	201	201	201	201	201	200	200	200	200	200	200	200	200	200	199	199	199	199	199	199	-	198	201	198	200
17.01.2024	198	198	199	198	198	198	197	196	195	194	194	192	192	192	193	193	194	193	193	193	193	193	194	195	199	192	195
18.01.2024	196	198	199	200	200	200	200	200	200	200	200	200	201	201	202	203	205	206	208	209	212	214	217	220	220	196	204
19.01.2024	222	225	228	230	231	232	233	233	232	232	232	231	231	231	230	229	229	229	228	227	226	226	225	224	233	222	229
20.01.2024	224	223	223	222	221	221	220	220	219	219	218	218	217	216	217	216	216	215	215	215	214	214	214	214	224	214	218
21.01.2024	213	213	213	212	212	212	212	211	211	211	211	210	210	210	210	210	209	209	209	209	208	208	208	208	213	208	210

Рисунок 2.4 – Вихідні дані рівнів води з посту «Самбір»

Дата	0	1	2	3	4	5	6	7	8	...	17	18	19	20	21	22	23	Max	Min	Average
01.01.2024	215	215	215	214	214	214	214	214	213	...	212	212	213	212	212	213	213	215	212	213
02.01.2024	213	213	213	214	215	215	216	216	217	...	216	215	215	215	214	214	214	217	213	215
03.01.2024	213	213	213	213	213	212	212	212	212	...	214	214	214	213	214	214	213	214	212	213
04.01.2024	213	213	213	214	214	213	213	213	213	...	212	212	-	-	-	-	-	214	212	213
05.01.2024	212	212	212	212	212	213	213	214	215	...	225	225	225	225	224	224	224	225	212	219
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
27.12.2024	-	-	-	-	-	-	-	-	181	...	180	180	179	178	178	178	-	181	178	180
28.12.2024	-	-	-	-	-	-	-	-	-	...	182	181	181	181	181	181	-	182	181	182
29.12.2024	-	-	-	-	-	-	-	-	181	...	180	180	181	180	180	180	-	181	180	181
30.12.2024	-	-	-	-	-	-	-	-	180	...	178	177	177	176	-	-	-	180	176	179
31.12.2024	-	-	-	-	-	-	-	-	-	...	176	-	175	-	-	-	-	179	175	177

Рисунок 2.5 – Таблиця з даними рівня води після зчитування файлу у форматі CSV

Також задля підвищення точності просторового аналізу було додатково включено шейп-файли з контурами водогосподарських ділянок (рис. 2.6), що дозволяє будувати карти та здійснювати просторову ідентифікацію кожної ділянки.

	SEM373	SEM116	ObjectCode	ObjectKey	ObjectID	ObjectName	geometry
0	A6.6.2.03	р. Сан та її притоки (в межах України)	94100122	P94100122	51	Водогосподарська ділянка	POLYGON ((23.45920 50.20367, 23.45920 50.20330...
1	A6.6.2.03	р. Сан та її притоки (в межах України)	94100122	P94100122	52	Водогосподарська ділянка	MULTIPOLYGON (((22.84684 49.00330, 22.84670 49...
2	M5.2.0.12	Дністровський лиман	94100122	P94100122	53	Водогосподарська ділянка	POLYGON ((30.22343 46.41694, 30.23530 46.40398...
3	M5.1.3.29	р. Оріль від кордону Харківської та Дніпропетр...	94100122	P94100122	54	Водогосподарська ділянка	POLYGON ((34.89420 49.43250, 34.89420 49.43000...
4	M5.1.3.28	р. Оріль від витоку до кордону Харківської та ...	94100122	P94100122	55	Водогосподарська ділянка	POLYGON ((35.49920 49.79330, 35.49920 49.79170...

Рисунок 2.6 – Вигляд даних із шейп-файлу з контурами водогосподарських ділянок

Усі зазначені компоненти – оброблені таблиці водогосподарських балансів, погодинні спостереження рівнів води, геометрія ВГД – інтегровані у фінальний датасет, завантажений на платформу Kaggle [26]. Така структура забезпечує зручний і публічний доступ до даних, повторюваність експериментів та можливість інтеграції нових джерел інформації в подальшому. У наступних розділах ці дані будуть використані для побудови моделей прогнозування рівня води та аналізу впливу гідрологічних і керованих факторів на водогосподарський баланс.

2.2 Розвідувальний аналіз та візуалізація водогосподарських балансів ділянок басейну Дністра

У процесі системного аналізу даних, реалізованого в середовищі Kaggle Notebooks [27], було побудовано прототип системи оцінювання гідрологічного стану водогосподарських ділянок басейну Дністра. В рамках цього етапу проєкту виконано повноцінний розвідувальний аналіз (EDA), спрямований на дослідження структури даних, виявлення статистичних закономірностей, зв’язків між ключовими змінними та їх просторово-часову динаміку.

Першим кроком стало візуальне дослідження розподілів числових змінних у

вибірці, зокрема таких як Intake, Return, Water_Reserve, Water_Deficit та ін. За допомогою histplot з використанням оцінки щільності розподілу (kde=True) було побудовано серію гістограм, які дозволили візуально оцінити характер розподілу кожної змінної (рис. 2.7). Аналіз розподілів показав, що більшість змінних мають правосторонню асиметрію, зокрема Water_Deficit, що свідчить про наявність значної кількості випадків із низьким дефіцитом і поодиноких – з надзвичайно високими значеннями. Це підтверджує гіпотезу про нерівномірний розподіл дефіциту серед ділянок: лише деякі з них мають критично високі втрати води. Натомість резерв води (Water_Reserve) демонструє більш симетричний розподіл, однак також демонструє значну варіативність значень і наявність асиметрії. Це вказує на те, що хоча більшість ділянок мають середні значення резерву, існують як ділянки з дуже низькими, так і з дуже високими показниками.

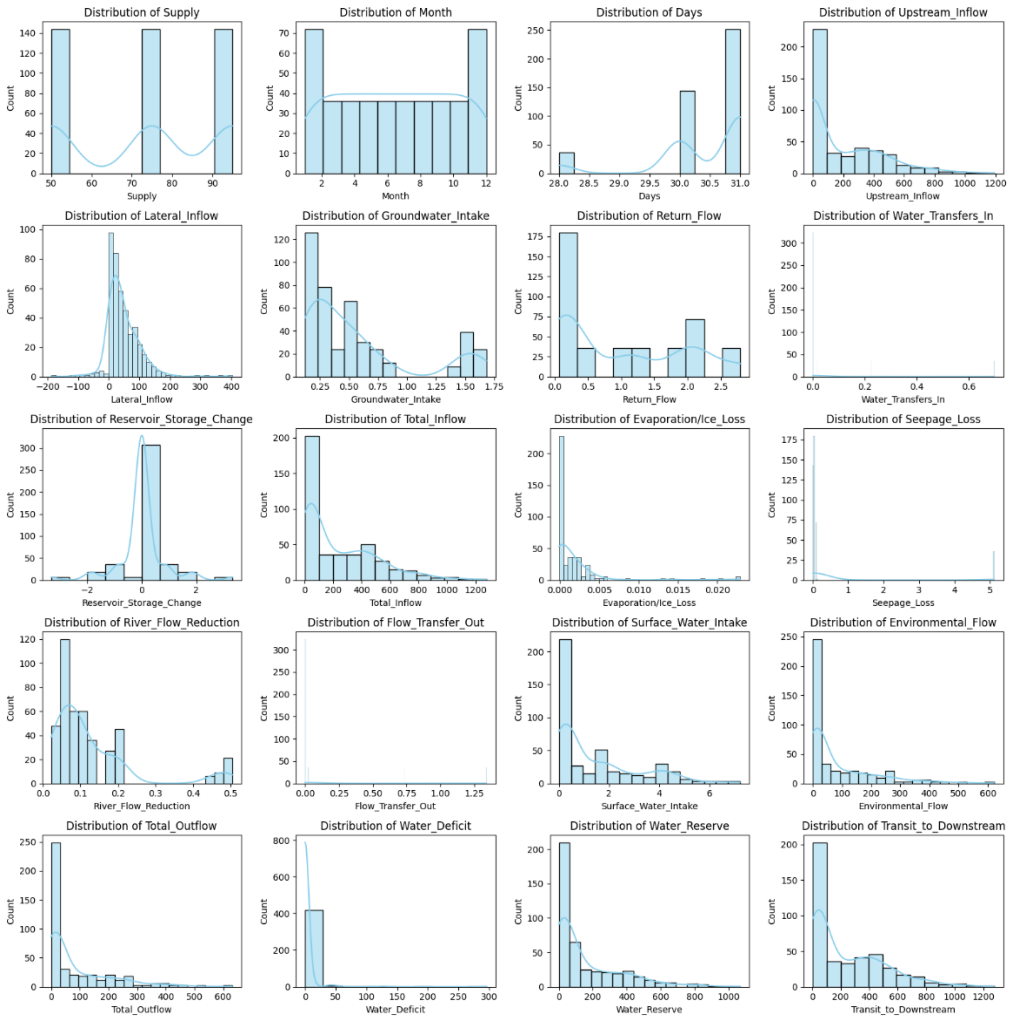


Рисунок 2.7 – Гістограми розподілу числових характеристик

Наступним важливим етапом став кореляційний аналіз, реалізований у вигляді теплової матриці (рис. 2.8). Було обрано лише числові ознаки, після чого побудовано матрицю парних кореляцій Пірсона. Матриця кореляцій – це проста таблиця, яка показує коефіцієнти кореляції між змінними [28]. Візуалізація засобами seaborn.heatmap із кольоровою палітрою Spectral дозволила легко виокремити найбільш значущі зв'язки. Аналіз показав наявність як сильних позитивних, так і негативних кореляцій між окремими змінними, що свідчить про складну структуру взаємодії елементів водогосподарського балансу. Серед цікавих зв'язків слід відзначити сильну позитивну кореляцію між Seepage_Loss та Flow_Transfer_Out (r = 0.87), що може вказувати на технічно зумовлену одночасну передачу води на інші ділянки та втрати через фільтрацію. Помірна позитивна кореляція між Return_Flow та Total_Inflow (r = 0.42) вказує на роль зворотних вод у забезпеченні повторного водокористування.

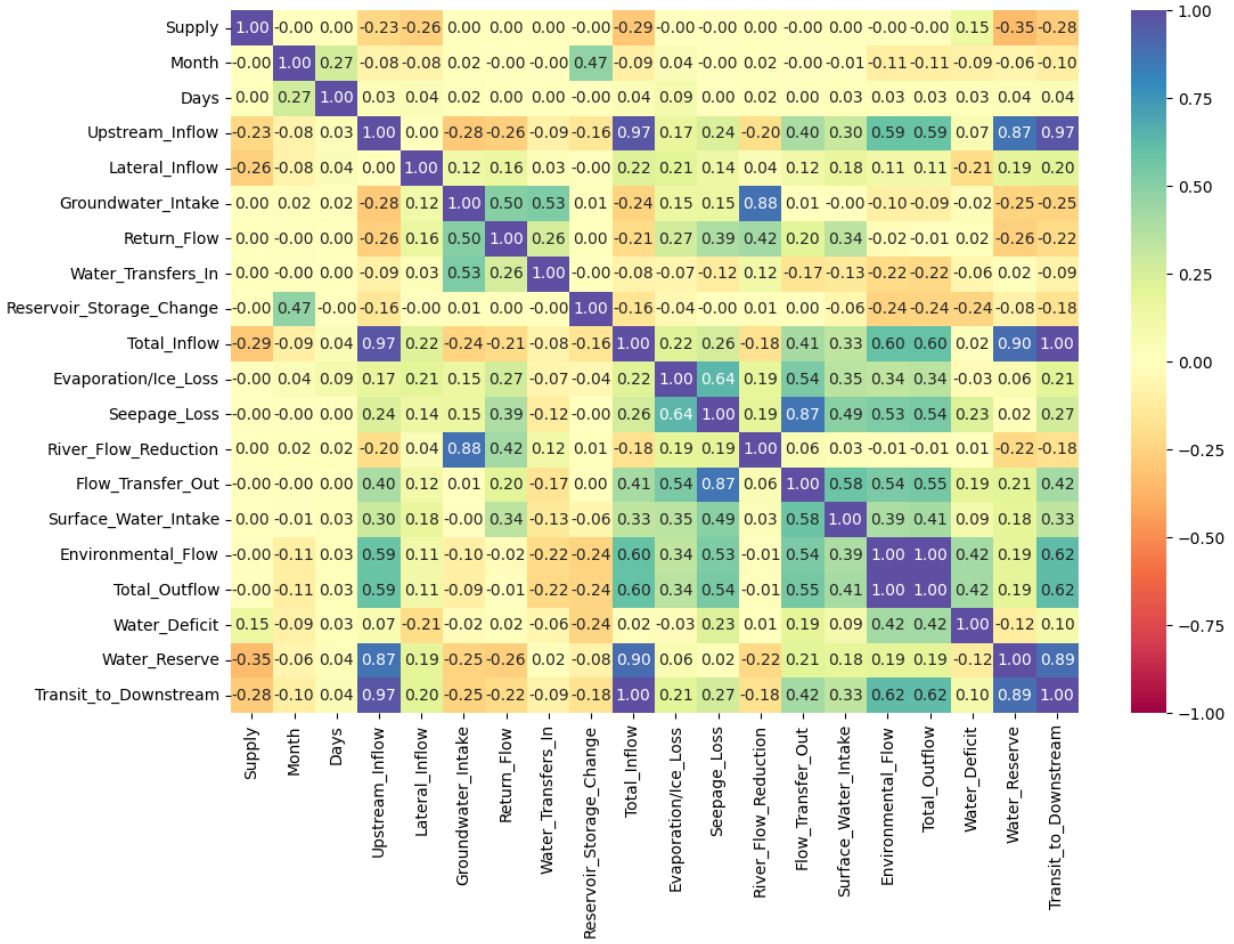


Рисунок 2.8 – Кореляційна матриця

У ході просторово-часового аналізу було побудовано графіки динаміки водного дефіциту для кожної ділянки за трьома рівнями гідрологічної забезпеченості – 50%, 75% та 95% (рис. 2.9). Візуалізації показали чітку просторову неоднорідність та сезонність дефіцитів. Так, на більшості ділянок, де дефіцит фіксується, його пік припадає на весняно-літній період (переважно березень-травень). У зимові місяці дефіцит практично відсутній, що відповідає зниженому водоспоживанню та зростанню стоку.

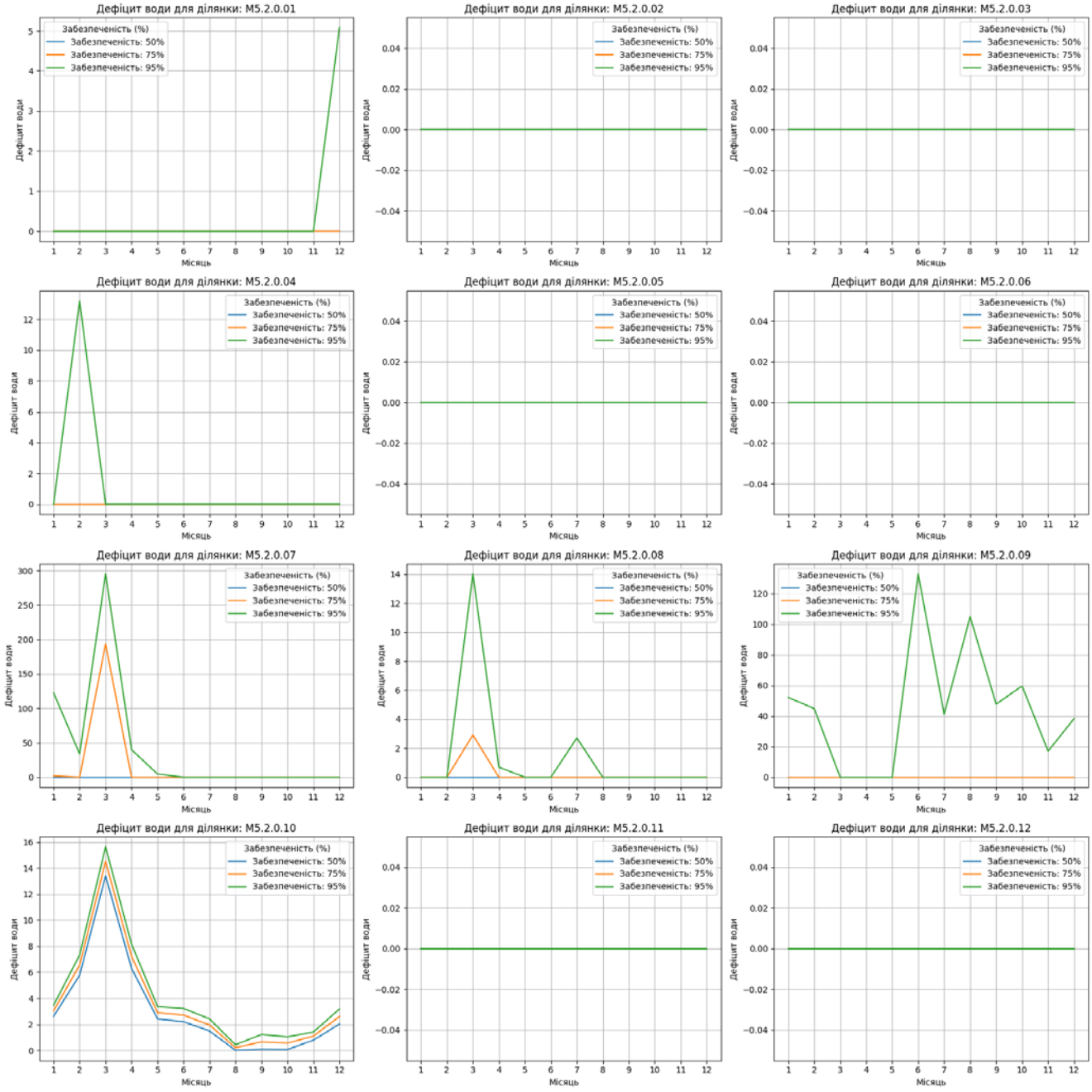


Рисунок 2.9 – Дефіцит води для ділянок басейну Дністра

Особливо критичними виявилися ділянки з кодами M5.2.0.07, M5.2.0.09 та M5.2.0.10. Зокрема, для M5.2.0.07 у березні зафіксовано надзвичайно високий дефіцит води – майже 300 млн м^3 при 95% забезпеченості. Подібна ситуація спостерігається і на ділянці M5.2.0.10, де дефіцит зберігається протягом усього року, навіть за забезпеченості 95%, що може свідчити про хронічне водозабезпечення на фоні сталого попиту. Натомість ділянки M5.2.0.02, M5.2.0.03, M5.2.0.05, M5.2.0.06, M5.2.0.11 та M5.2.0.12 демонструють нульовий або мінімальний дефіцит за всіма сценаріями, що вказує на їхню водогосподарську стабільність.

Аналогічно було побудовано графіки динаміки водного резерву (рис. 2.10). На них простежується зменшення запасів у літні місяці з накопиченням у весняний період. Деякі ділянки демонструють виражену нестабільність резервів між сценаріями забезпеченості, що вказує на вразливість до гідрологічних коливань.

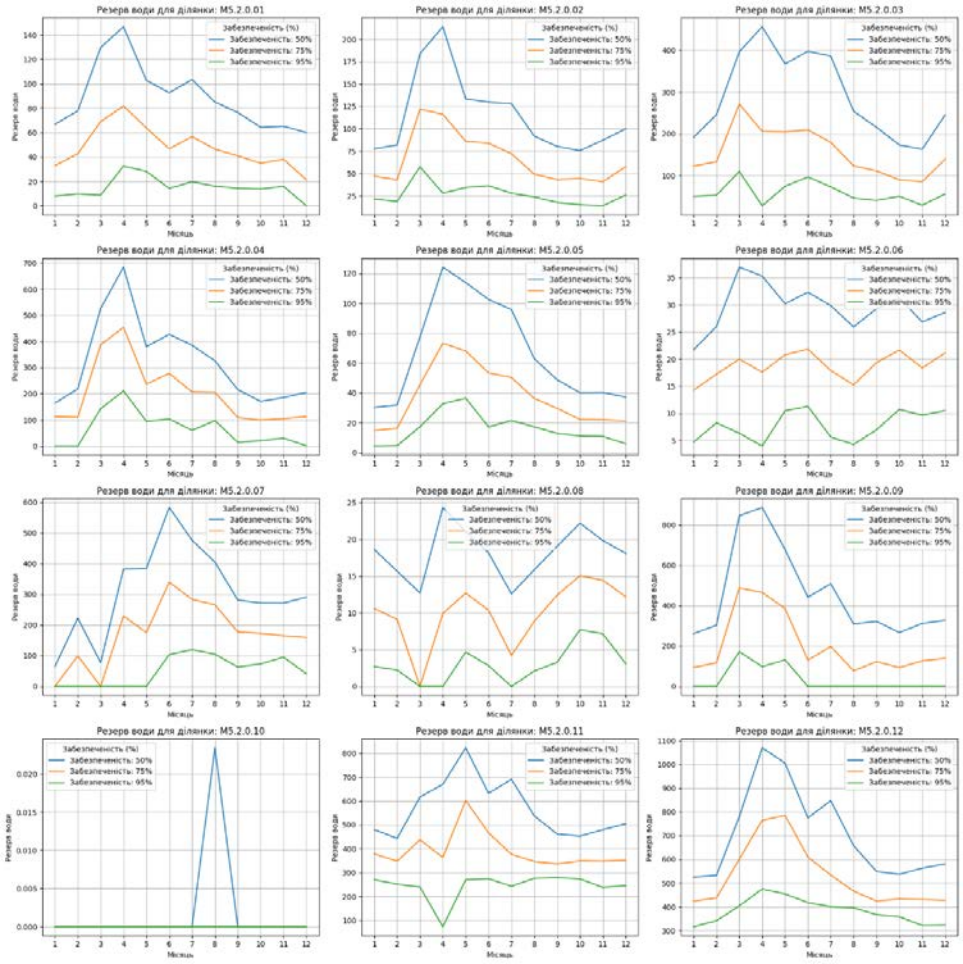


Рисунок 2.10 – Резерв води для ділянок басейну Дністра

З метою просторового оцінювання дефіциту було реалізовано картографічну візуалізацію за допомогою бібліотеки GeoPandas (рис. 2.11). Створено кольорову мапу, на якій кожна ділянка відображає рівень дефіциту у відповідній кольоровій гамі. Крім того, нанесено кругові маркери на центроїдах ділянок, радіус яких пропорційний обсягу резерву. Це дозволило інтегрувати два важливих параметри на одну карту, спростивши аналітичне оцінювання загального стану кожної зони.

Рисунок 2.11 – Карта дефіцитів та резервів води по водогосподарських ділянках

Остаточним виводом візуального модулю стала інтерактивна карта, створена з використанням folium [29]. Вона включає як динамічне відображення значень (tooltip, popup), так і власні HTML-легенди, що пояснюють як шкалу дефіциту, так і градації розміру маркерів за рівнем резерву (рис. 2.12). Інтерфейс дозволяє аналізувати окремі ділянки та отримувати повну картину щодо розподілу водних ресурсів у просторі.

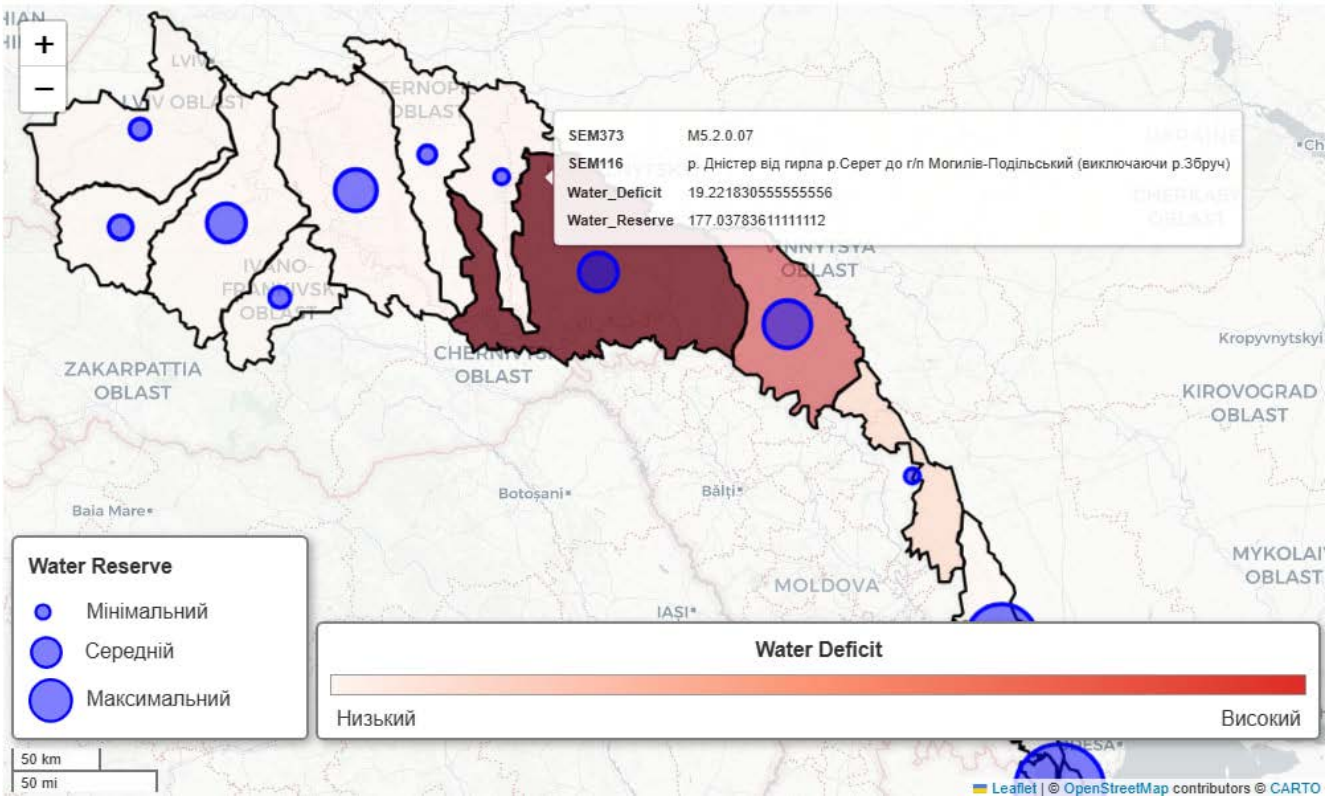


Рисунок 2.12 – Інтерактивна карта дефіцитів та резервів води по водогосподарських ділянках

На основі проведеного просторово-часового аналізу було виявлено ділянку з найвищим середньорічним дефіцитом водних ресурсів – M5.2.0.07 (р. Дністер від гирла р. Серет до г/п Могилів-Подільський (виключаючи р. Збруч)), де середнє значення дефіциту перевищує 19 млн куб. м, попри наявний резерв понад 177 млн куб. м. Такий дисбаланс свідчить про наявність хронічної нестачі води, що потребує подальшого вивчення в контексті управлінських рішень та інфраструктурних заходів. У подальшому, для побудови моделей прогнозування рівня води буде використано дані з автоматичного гідропосту Самбір, який розташовано в межах іншої водогосподарської ділянки. Такий вибір зумовлений доступністю погодинних спостережень і дозволить протестувати ефективність моделей машинного навчання на фактичних гідрологічних даних.

2.3 Розвідувальний аналіз даних про рівень води на гідропості Самбір

Для подальшого прогнозування рівня води в басейні Дністра було проведено розвідувальний аналіз даних (EDA) [30] на основі спостережень гідропосту Самбір, розміщеного на річці Дністер у Львівській області. Дослідження охоплює погодинні спостереження рівня води за 2024 рік, які були отримані з офіційного джерела – сайту Укргідрометеоцентру [25]. Для аналізу було використано середньодобові значення рівня води, що містять у вихідному датасеті. Було виконано попередню обробку даних (рис. 2.13): у таблиці залишено лише стовпці з датою та середнім рівнем, перейменовано їх відповідно до стандарту бібліотеки Prophet (ds – дата, y – значення), а також перетворено тип стовпця ds у формат datetime для коректного розпізнавання часової ознаки.

In [8]:
```
df = df[['Дата', 'Average']]
df.columns = ['ds', 'y']
df['ds'] = pd.to_datetime(df['ds'], format="%d.%m.%Y")
df
```

Out[8]:

	ds	y
0	2024-01-01	213
1	2024-01-02	215
2	2024-01-03	213
3	2024-01-04	213
4	2024-01-05	219
...	...	...
344	2024-12-27	180
345	2024-12-28	182
346	2024-12-29	181
347	2024-12-30	179
348	2024-12-31	177

349 rows × 2 columns

Рисунок 2.13 – Попередня обробка даних рівня води

Першим кроком аналізу стало дослідження середньодобових значень рівня води. На рисунку 2.14 наведено графік, що ілюструє зміну середнього рівня води упродовж 2024 року, сформований на основі середньодобових значень.

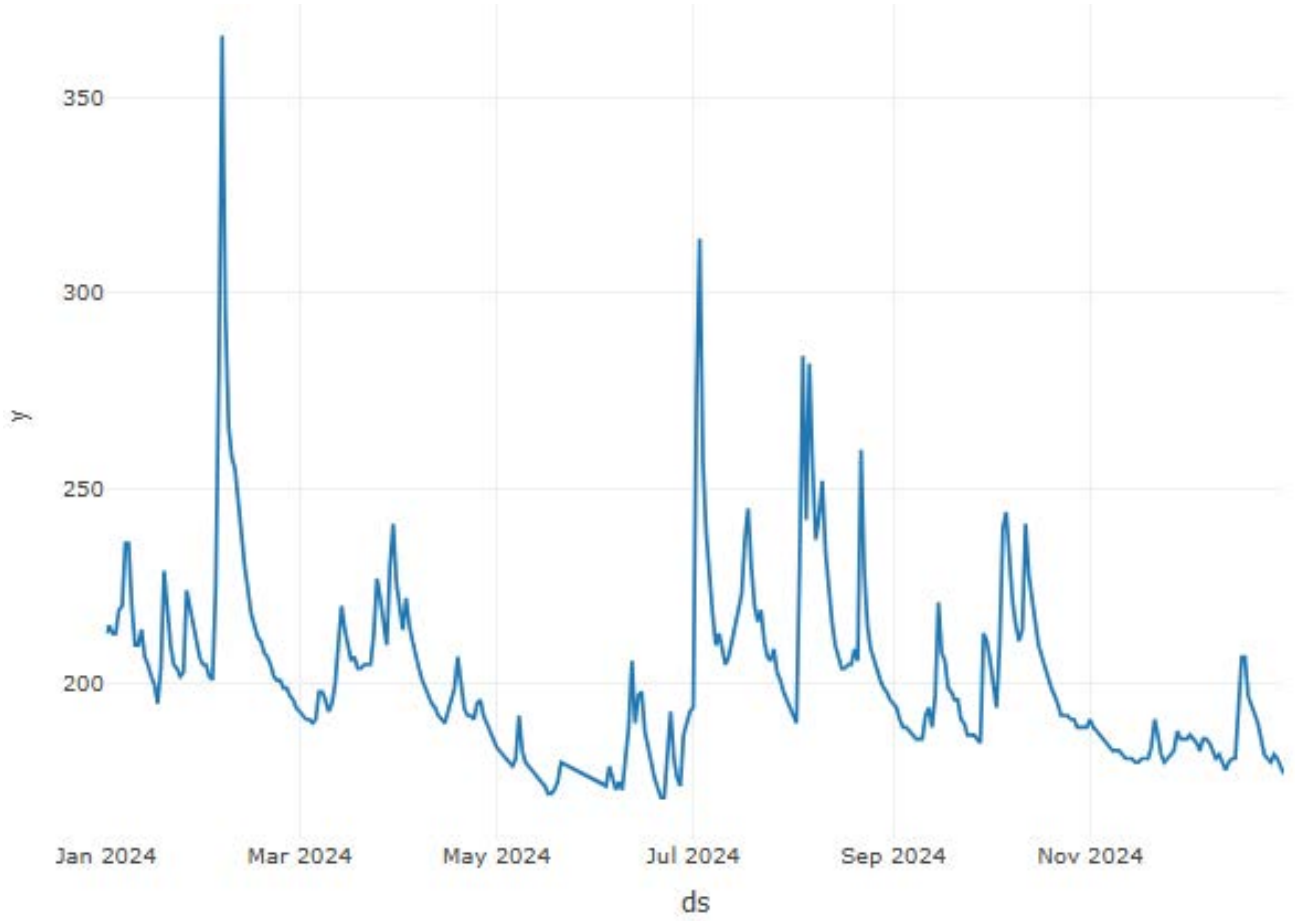


Рисунок 2.14 – Графік середніх добових рівнів води на гідропості Самбір, 2024 рік

На графіку спостерігається виражена сезонна варіація рівня води протягом 2024 року. У січні-лютому утримується переважно середній рівень, після чого в березні зафіксовано стрімке зростання до рекордних значень, що свідчить про початок весняного водопілля. У квітні-травні рівень знижується, проте залишається нестабільним із короткочасними піками. У червні-липні знову фіксується інтенсивне зростання рівня, імовірно пов’язане з дощовими паводками. У другій половині року (серпень-листопад) рівень води загалом знижується, зберігаючи коливання, а в грудні досягає мінімальних значень. Такий характер коливань є типовим для річок з комбінованим (сніговим і дощовим) живленням, до

яких належить і верхня течія самого Дністра. Весняне зростання рівня води зумовлене таненням снігу, а літні піки – інтенсивними опадами.

Для кращого розуміння періодичних закономірностей було проведено сезонну декомпозицію часового ряду. Було проаналізовано сезонність при різних періодах: 7 днів (тиждень), 30 днів (місяць), 92 дні (квартал) та 4 дні. Серед них найбільш чітке повторення сезонного компонента було зафіксовано при періоді в 92 дні – тобто квартальній сезонності (рис. 2.15).

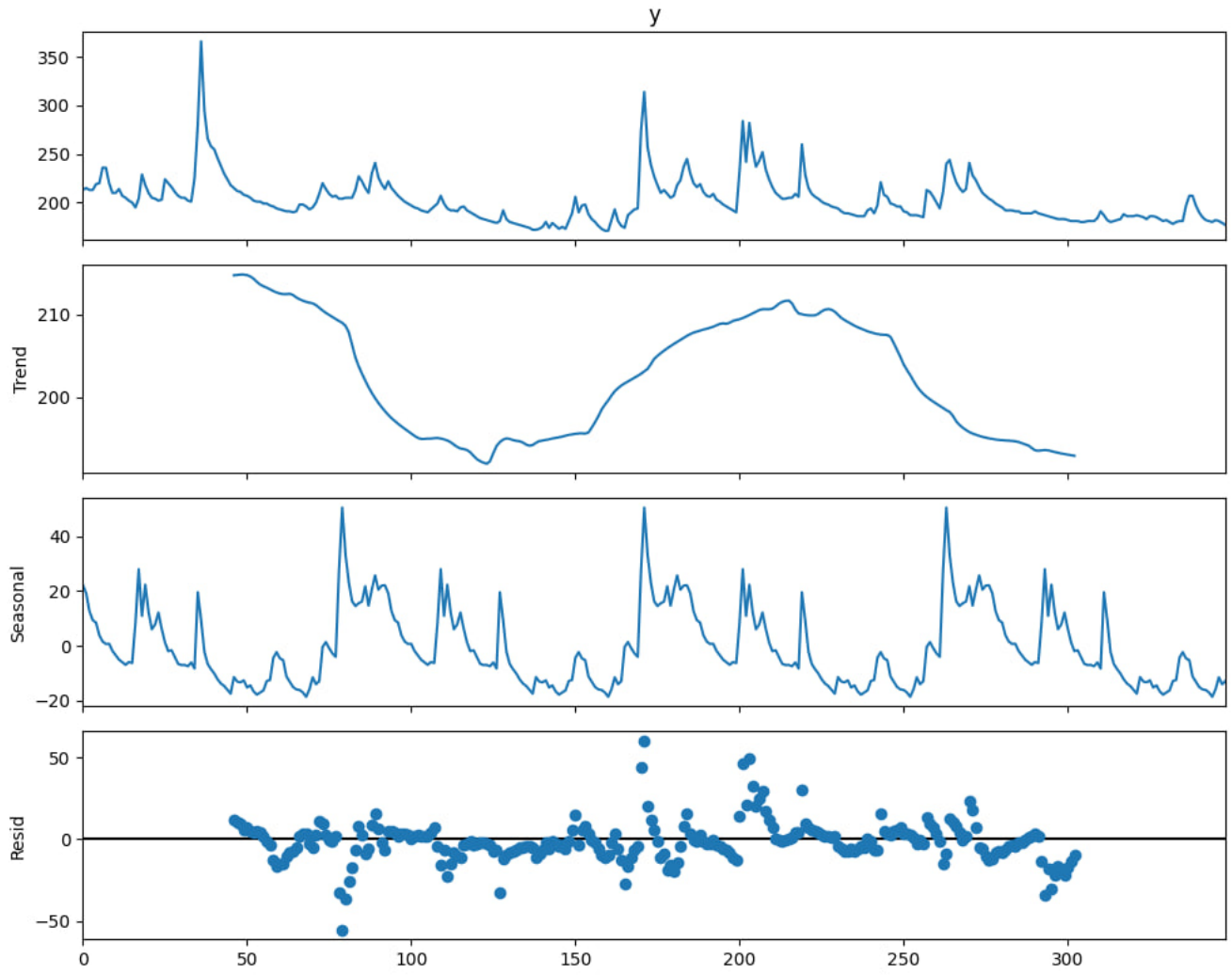


Рисунок 2.15 – Сезонна декомпозиція рівня води на гідропості Самбір при квартальній сезонності (період 92 дні)

Сезонна компонента демонструє стабільний характер із вираженим циклом підняття, піку та наступного зниження рівня води. Амплітуда змін сезонності

становить приблизно 69 см, що підтверджує вагомий внесок сезонного чинника в загальну динаміку рівнів води. Натомість сезонності з коротшими періодами (рис. 2.16 – 2.18) показали слабко виражену або хаотичну структуру, що свідчить про відсутність регулярної короткострокової циклічності.

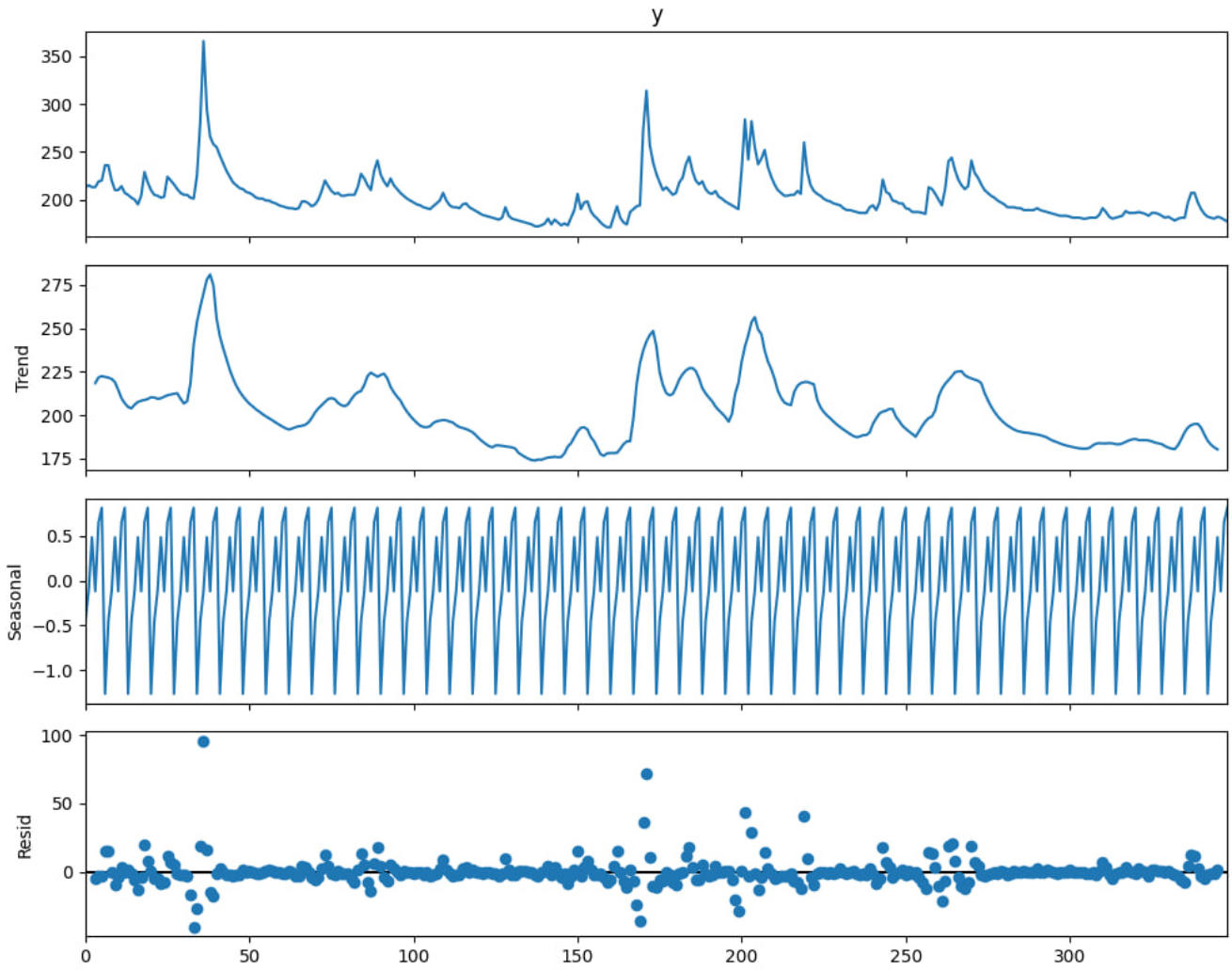


Рисунок 2.16 – Сезонна декомпозиція рівня води на гідропості Самбір при тижневій сезонності (період 7 днів)

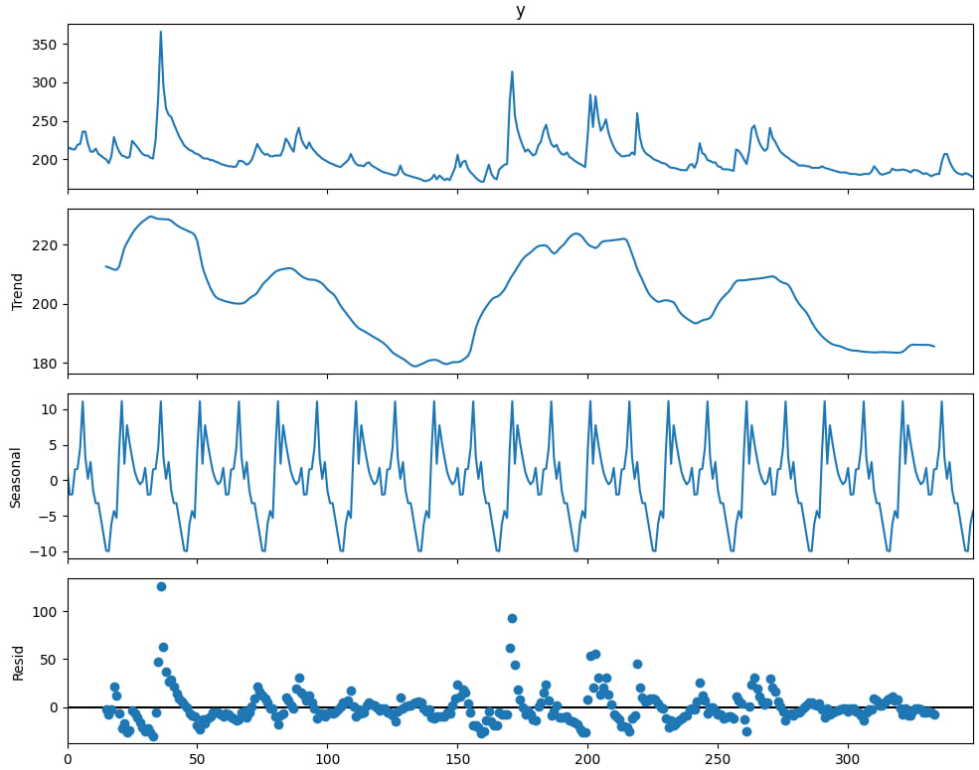


Рисунок 2.17 – Сезонна декомпозиція рівня води на гідропості Самбір при місячній сезонності (період 30 днів)

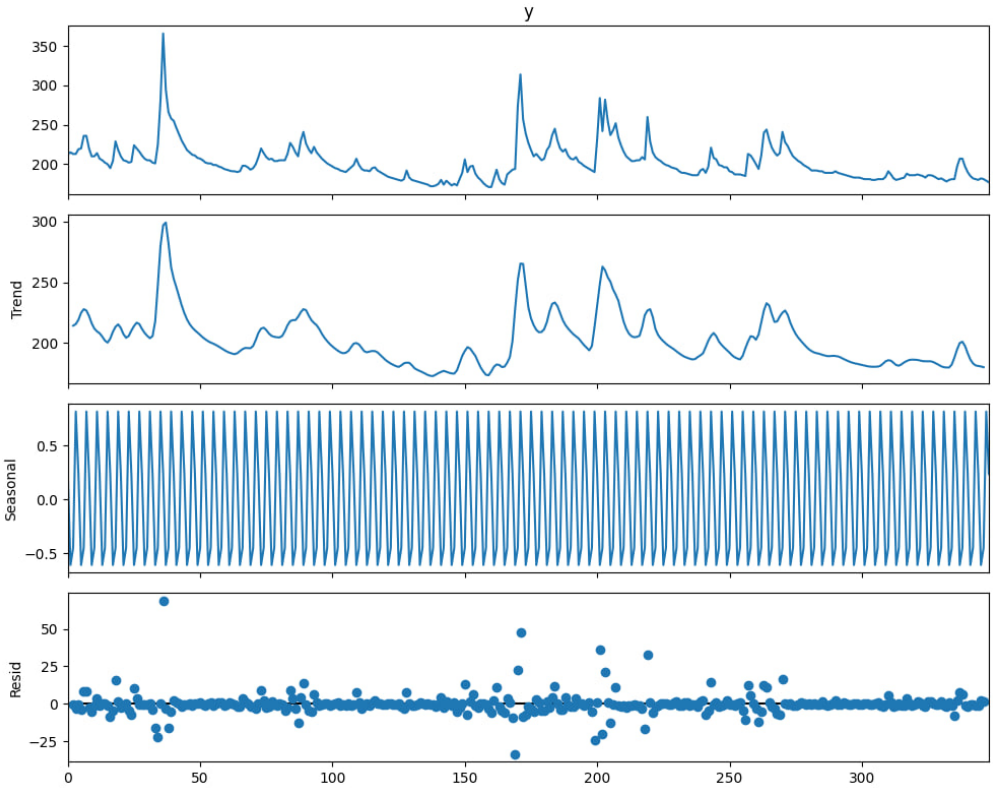


Рисунок 2.18 – Сезонна декомпозиція рівня води на гідропості Самбір при 4-денному періоді

Здійснений аналіз дозволив визначити, що гідропост Самбір демонструє стабільну квартальну динаміку рівня води, яка є релевантною для подальшого використання в побудові моделей прогнозування. Отримані результати дозволяють сформувати гіпотезу про наявність природного сезонного патерну, який може бути формально врахований у регресійних та гібридних моделях інтелектуального аналізу даних. Це створює підґрунтя для використання відповідних сезонних параметрів у побудові моделей типу Prophet, ARIMA та інших, що дозволяє покращити точність коротко- та середньострокових прогнозів рівня води на гідрологічних постах регіону.

2.4 Інженерія ознак для підготовки даних до прогнозування

Для підвищення інформативності моделей прогнозування рівня води на гідропості Самбір було здійснено інженерію ознак з використанням сучасного інструментарію Python. Ключовим етапом стала інтеграція бібліотеки TSFRESH [31] – одного з найпотужніших інструментів для автоматизованого генерування дескрипторів із часових рядів.

Інженерія ознак для часових рядів – це трудомісткий процес, оскільки науковці та інженери повинні враховувати різноманітні алгоритми обробки сигналів і аналізу часових рядів, для ідентифікації та вилучення значущих ознак. Python-бібліотека tsfresh (Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests) прискорює цей процес, поєднуючи 63 методи характеристики часових рядів, які за замовчуванням обчислюють загалом 794 ознаки, із вибором ознак на основі автоматично налаштованих перевірок гіпотез [32].

Для уникнення надмірної кількості ознак і зниження ризику перенавчання моделей було виконано відбір релевантних дескрипторів, який спирається на значущість ознаки у контексті передбачення цільової змінної y. Як результат, після очищення залишилось 77 нових колонок, що є додатковими ознаками для побудови моделей. Нові ознаки було інтегровано до основного датафрейму через операцію об’єднання (merge) за спільною колонкою ds (дата). Після цієї операції розмір

таблиці збільшився з двох колонок до 79, що підтверджує успішне приєднання 77 дескрипторів. Фрагмент оновленої таблиці наведено на рисунку 2.19.

	ds	y	level_0__sum_values	level_0__abs_energy	level_0__median	level_0__mean	level_0__root_mean_square	level_0__r
0	2024-01-01	213	0.0	0.0	0.0	0.0	0.0	0.0
1	2024-01-02	215	1.0	1.0	1.0	1.0	1.0	1.0
2	2024-01-03	213	2.0	4.0	2.0	2.0	2.0	2.0
3	2024-01-04	213	3.0	9.0	3.0	3.0	3.0	3.0
4	2024-01-05	219	4.0	16.0	4.0	4.0	4.0	4.0
...	...	...	...	...	...	...	...	...
344	2024-12-27	180	344.0	118336.0	344.0	344.0	344.0	344.0
345	2024-12-28	182	345.0	119025.0	345.0	345.0	345.0	345.0
346	2024-12-29	181	346.0	119716.0	346.0	346.0	346.0	346.0
347	2024-12-30	179	347.0	120409.0	347.0	347.0	347.0	347.0
348	2024-12-31	177	348.0	121104.0	348.0	348.0	348.0	348.0

349 rows × 79 columns

Рисунок 2.19 – Фрагмент датафрейму після додавання ознак, сформованих за допомогою TSFRESH

Наступним етапом було реалізовано обробку аномальних подій. Для цього вручну визначено перелік дат, у які фіксувались нетипово різкі коливання рівня води. Ці дати було внесено до окремої таблиці holidays_df, яка використовувалась як спеціальний модуль для моделі Prophet. У цій таблиці кожна аномальна дата вказується як «свято», з нульовими вікнами допуску (lower_window = 0, upper_window = 0), але з підвищеним коефіцієнтом ваги (prior_scale = 10), що сигналізує моделі про необхідність врахування цих подій як нетипових. Таблиця holidays_df наведена на рисунку 2.20.

	ds	lower_window	upper_window	prior_scale	holiday
0	2024-02-06	0	0	10	anomalous_dates
1	2024-03-30	0	0	10	anomalous_dates
2	2024-06-18	0	0	10	anomalous_dates
3	2024-07-03	0	0	10	anomalous_dates
4	2024-08-04	0	0	10	anomalous_dates
5	2024-08-06	0	0	10	anomalous_dates
6	2024-08-22	0	0	10	anomalous_dates
7	2024-09-15	0	0	10	anomalous_dates
8	2024-10-06	0	0	10	anomalous_dates
9	2024-10-12	0	0	10	anomalous_dates
10	2024-12-19	0	0	10	anomalous_dates

Рисунок 2.20 – Таблиця holidays_df для врахування аномальних подій у Prophet

Для забезпечення сумісності з класичними моделями машинного навчання (SVM, XGBoost, Random Forest тощо) також було згенеровано двійкову ознаку y_diff_anomalous. Вона вказує, чи відповідає поточна дата одному з ідентифікованих аномальних сплесків. Ця ознака була додана до датафрейму як ще один інформативний предиктор. Рядки з індикатором аномалії відображено на рисунку 2.21.

	ds	y	y_diff_anomalous
36	2024-02-06	366	1
89	2024-03-30	241	1
156	2024-06-18	180	1
171	2024-07-03	314	1
201	2024-08-04	284	1
203	2024-08-06	282	1
219	2024-08-22	260	1
243	2024-09-15	221	1
264	2024-10-06	244	1
270	2024-10-12	241	1
338	2024-12-19	207	1

```
Number of anomalous dates - 11
```

Рисунок 2.21 – Рядки датафрейму з позначенням аномальних дат (y_diff_anomalous = 1)

Для візуального підтвердження коректності розмітки було побудовано графік рівня води з виділенням аномальних точок червоними маркерами (рис. 2.22). Це дозволило переконатись, що обрані дати дійсно відповідають нетиповим коливанням.

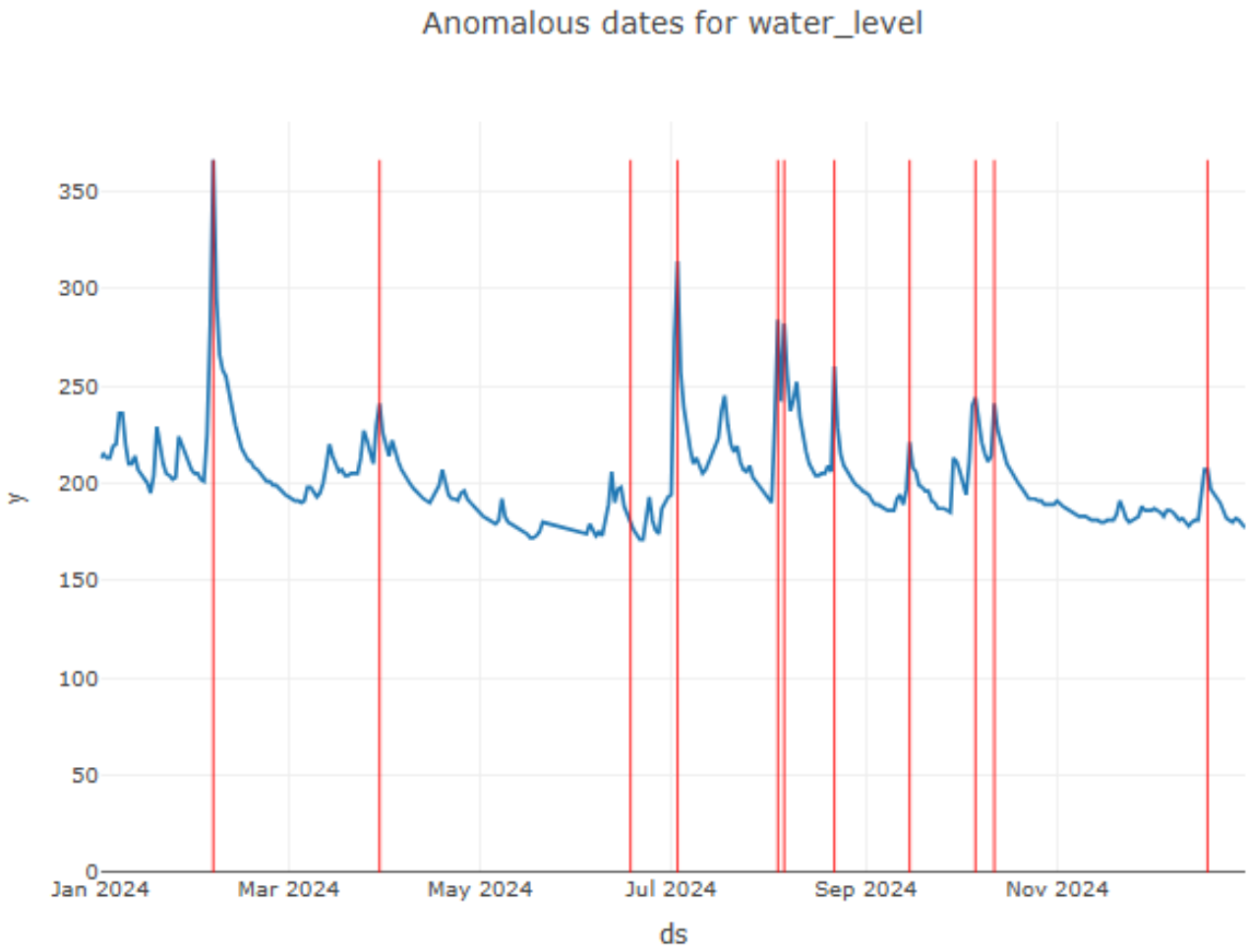


Рисунок 2.22 – Візуалізація аномальних коливань рівня води на часовому графіку

Таким чином, виконана інженерія ознак базується на принципах автоматизованого видобування статистичних дескрипторів із рядів та доповнення даних ознаками про аномальні події. Вона створює основу для побудови точних і надійних моделей прогнозування рівня води в річці Дністер. У результаті отримано насичений набір інформативних ознак, придатних як для лінійних, так і для ансамблевих моделей машинного навчання, що забезпечує адаптивність прогнозування до різних сценаріїв гідрологічної динаміки.

2.5 Висновки

В межах другого розділу було виконано повний цикл підготовки даних – від первинного зчитування неоднорідних джерел до формування насиченого набору ознак, готового для побудови моделей прогнозування. Було конвертовано та уніфіковано таблиці балансів ВГД, погодинні спостереження рівня води на гідропості Самбір і просторові шейп-файли. Усі ці компоненти завантажено на Kaggle [26], що забезпечує відтворюваність дослідження й можливість подальшого розширення датасету.

Розвідувальний аналіз балансових показників виявив суттєву просторову неоднорідність дефіциту та резервів води, визначивши критичну ділянку M5.2.0.07 із середньорічним дефіцитом понад 19 млн куб. м. Аналіз часових рядів рівня води показав стабільну квартальну сезонність із амплітудою близько 69 см, що задає часовий горизонт для подальшого прогнозування.

Застосування TSFRESH додало 77 статистичних дескрипторів, збільшивши розмір датафрейму до 79 колонок. Також було додано 11 аномальних дат, які враховуватимуться окремими регресорами (у Prophet) або як бінарна ознака (у класичних ML-моделях).

У сукупності ці кроки створюють надійну аналітичну базу для наступного етапу – порівняння та відбір моделей машинного навчання з метою підвищення ефективності управління водними ресурсами басейну Дністра.

3. ПРОГНОЗУВАННЯ РІВНЯ ВОДИ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

3.1 Опис моделей машинного навчання для прогнозування рівня води

Для прогнозування рівня води у річці Дністер було використано такі моделі: Facebook Prophet, AutoARIMA, Linear Regression, KNeighbors Regressor, Support Vector Machines, Linear SVR, Random Forest Regressor, Bagging Regressor, XGB Regressor та MLP Regressor. Метою використання широко ряду моделей є подальше оцінювання їх ефективності, що дозволить сформулювати обґрунтований висновок щодо вибору найдоцільнішої архітектури системи прогнозування.

Prophet – це процедура прогнозування даних часових рядів, заснована на адитивній моделі, де нелінійні тренди апроксимуються за допомогою річної, тижневої та денної сезонності, а також ефектів свят [11]. Тобто дана модель розкладає часовий ряд на компоненти: тренд, різні типи сезонності та вплив свят, які потім окремо моделюються та комбінуються для прогнозування. Її ключові переваги полягають в автоматизованому підході та високій інтерпретовності, забезпеченій чітким розділенням компонентів, а також у стійкості до пропусків даних та викидів. Проте, Prophet має обмеження у моделюванні складних автокореляційних залежностей через відсутність авторегресійної компоненти, може демонструвати зниження точності при довгостроковому прогнозуванні та базується на припущенні про адитивний характер взаємодії компонентів, що не завжди є універсальним для всіх типів часових рядів.

Модель ARIMA (autoregressive integrated moving average) є узагальненням моделі авторегресійного ковзного середнього, яка використовується для моделювання часових рядів з метою прогнозування майбутніх значень. Моделі ARIMA можуть бути особливо ефективними у випадках, коли дані демонструють ознаки нестаціонарності [12]. Процес auto-ARIMA спрямований на ідентифікацію найбільш оптимальних параметрів для моделі ARIMA, обираючи одну налаштовану модель ARIMA. Auto-ARIMA працює шляхом проведення тестів на необхідність диференціювання (зокрема, тесту Квятковського-Філіпса-Шмідта-

Шина, доповненого, доповненого тесту Дікі-Фуллера або тесту Філіпса-Перрона), щоб визначити порядок диференціювання d, а потім підбирає моделі в межах заданих діапазонів параметрів start_p, max_p, start_q, max_q. Якщо активовано опцію сезонності (seasonal), auto-ARIMA також намагається визначити оптимальні гіперпараметри P і Q, попередньо провівши тест Канова-Гансена для визначення оптимального порядку сезонного диференціювання D [13]. Ця автоматизація усуває необхідність у ручному виборі порядку моделі, що є трудомістким процесом. Проте, її недоліки полягають у підвищених обчислювальних витратах, оскільки вона перебирає численні комбінації параметрів, що може займати багато часу для дуже великих часових рядів або при пошуку в широкому діапазоні параметрів. Крім того, хоча й auto-ARIMA прагне знайти найкращу модель, вона не гарантує, що ця модель є глобально оптимальною, і іноді може «застрягати» у локальних мінімумах, особливо при наявності аномалій або різких змін у даних.

Linear Regression підлаштовує лінійну модель із коефіцієнтами w = (w1, …, wp), щоб мінімізувати суму квадратів залишків між фактичними значеннями цільової змінної у наборі даних та значеннями, передбаченими за допомогою лінійного наближення [14].

KNeighbors Regressor реалізує навчання на основі k найближчих сусідів кожної точки запиту, де k – це ціле числове значення, задане користувачем. Цільове значення прогнозується за допомогою локальної інтерполяції цільових значень, пов’язаних з найближчими сусідами в тренувальному наборі [15].

Support Vector Machines (SVMs) – це набір методів навчання з учителем, що використовуються для класифікації, регресії та виявлення викидів. Support Vector Regression (SVR) ґрунтується на тих самих принципах, що й Support Vector Classification (SVC), з кількома незначними відмінностями. Вона обмежує похибку передбачення заданим порогом [16].

Linear SVR реалізує лінійну модель підтримуючих векторів для регресії, використовуючи бібліотеку liblinear. Основні відмінності між LinearSVR та SVR полягають у використанні різних функцій втрат за замовчуванням, а також у способі регуляризації вільного члена в цих двох реалізаціях [17].

Випадковий ліс (Random Forest Regressor) – це мета-оцінювач, який навчає низку регресійних дерев рішень на різних підвибірках початкового датасету і використовує усереднення їхніх результатів для підвищення точності прогнозування та контролю перенавчання [18].

Bagging Regressor – це ансамблевий мета-оцінювач, який навчає базові регресори на випадкових підмножинах вихідного датасету, а потім агрегує їхні індивідуальні прогнози (шляхом голосування або усереднення), щоб сформувати фінальний прогноз [19].

XGBRegressor – це клас, сумісний з API бібліотеки scikit-learn, призначений для виконання регресії з використанням алгоритму градієнтного бустингу XGBoost. Він реалізує оптимізовану розподілену бібліотеку градієнтного бустингу, яка розроблена для досягнення високої ефективності, гнучкості та портативності [20].

MLPRegressor реалізує багатошаровий перцептрон (MLP), який навчається за допомогою алгоритму зворотного поширення помилки та не використовує функцію активації в вихідному шарі, що дозволяє розглядати його як стандартну нейромережеву модель для регресії [21].

Розглянуті моделі машинного навчання мають як спільні характеристики, так і специфічні особливості, що визначають їхню придатність до прогнозування часових рядів. Моделі класичної регресії, зокрема Linear Regression, вирізняються простою реалізацією, високою швидкістю навчання та зрозумілою інтерпретованістю. Проте вони передбачають наявність лінійного зв’язку між ознаками і цільовою змінною, що часто не відповідає складності гідрологічних процесів.

Методи, засновані на локальних залежностях, як-от KNeighbors Regressor, забезпечують хорошу адаптацію до нетипових форм розподілу, але демонструють обмежену здатність до узагальнення, особливо за межами навчальної вибірки. Також їх ефективність суттєво знижується на часових рядах, де порядок спостережень має критичне значення, адже модель не враховує часову структуру даних.

Моделі на основі підтримки векторів – Support Vector Machines (SVR) та Linear SVR – є потужними засобами для побудови як лінійних, так і нелінійних залежностей (залежно від використаного ядра). Вони демонструють хорошу здатність до узагальнення навіть при невеликому обсязі навчальної вибірки. Однак їх чутливість до налаштування гіперпараметрів, відносна складність обчислень і низька масштабованість можуть обмежити їх застосування в задачах із великим обсягом даних або високою частотою спостережень. Linear SVR, натомість, краще масштабується, хоча дещо поступається в гнучкості та точності при складних нелінійних зв'язках.

Ансамблеві дерева, такі як Random Forest Regressor та Bagging Regressor, демонструють високу точність, стійкість до перенавчання та здатність моделювати складні взаємозв'язки. Їх ключова перевага – нечутливість до шуму в даних і гнучкість щодо структури ознак. Утім, відсутність вбудованого механізму обліку часової послідовності обмежує їх ефективність у прогнозуванні часових рядів без попередньої інженерії ознак. Аналогічно, модель XGB Regressor, яка реалізує градієнтний бустинг, забезпечує високу точність і гнучкість, але вимагає ретельного налаштування та значних обчислювальних ресурсів.

Багатошарова перцептронна мережа (MLP Regressor) має здатність моделювати складні й нелінійні взаємозв'язки, особливо ефективна при великому обсязі навчальних даних. Її ключовими перевагами є універсальність та здатність до навчання прихованих патернів. Водночас, недоліками є обмежена інтерпретованість, ризик перенавчання без належної регуляризації та чутливість до вибору гіперпараметрів і структури мережі. Крім того, без спеціальної архітектурної модифікації (наприклад, рекурентних елементів) ця модель не враховує послідовність у часі.

Хоча більшість використаних моделей не мають вбудованої підтримки часових залежностей і потребують додаткової інженерії ознак для врахування тимчасових закономірностей, окремо слід виділити моделі Facebook Prophet і AutoARIMA. Вони є спеціалізованими алгоритмами прогнозування часових рядів і забезпечують інтеграцію сезонних та трендових компонент без необхідності ручної

трансформації даних. Тому їх використання є особливо доцільним у задачі гідрологічного прогнозування.

3.2 Тренування моделей машинного навчання

Модель Facebook Prophet була використана як одна з базових інструментів для прогнозування середньодобового рівня води на гідропості Самбір. Prophet є адитивною моделлю часового ряду, яка враховує тренд, сезонність і вплив зовнішніх подій (наприклад, аномальних дат). У розробленому ноутбуці «Water level - EDA & forecasting» [30] було налаштовано мультиплікативну сезонність із заданими періодами 90 та 120 днів, що відповідає природним сезонним циклам коливання рівня води.

Першим кроком було виконано розбиття датасету на тренувальну, валідаційну та тестову вибірки. Результат цієї операції представлено на рисунку 3.1.

```
In [47]:
# Get datasets
if is_Prophet:
    train_ts, valid_ts, test_ts, train_valid_ts = get_
train_valid_test_ts(df2.copy(), forecasting_days, targ
et='y')

    if not is_anomalies:
        holidays_df = None
```

```
Origin dataset has 348 rows and 2 features
Get training dataset with 338 rows
Get validation dataset with 5 rows
Get test dataset with 5 rows
```

Рисунок 3.1 – Розбиття даних на тренувальний, валідаційний і тестовий набори для побудови моделі Prophet

Далі реалізовано тренування моделі Prophet з кастомною сезонністю: до базової структури моделі було додано користувацьку сезонну компоненту з періодом 90 або 120 днів і Фур’є-порядком 3 або 12. Крім того, в моделі було відключено стандартну добову, тижневу та річну сезонність, натомість використовувалась тільки визначена вручну компонента. На вхід моделі також передано таблицю holidays_df з датами аномалій, що дозволило посилити врахування нетипових коливань рівня води.

Графіки на рисунках 3.2 – 3.5 ілюструють прогнозовані значення рівня води разом із фактичними спостереженнями за 2024 рік. Чорні крапки на графіку відповідають реальним значенням рівня води, зібраним протягом року, тоді як суцільна синя лінія – це прогноз моделі, побудований на основі попередніх значень. Навколо синьої лінії зображено світло-синю зону, яка відображає довірчі інтервали, що вказують на межі можливої похибки прогнозу. Ширина цієї зони в різні періоди змінюється, що демонструє зміну рівня невизначеності в оцінюванні моделі. Модель успішно вловлює сезонні коливання рівня води, включаючи характерні весняні та літні підйоми, що свідчить про ефективне навчання на історичних даних. Також модель реагує на окремі пікові значення, хоча деякі з них виходять за межі довірчих інтервалів, що свідчить про присутність аномалій у даних.

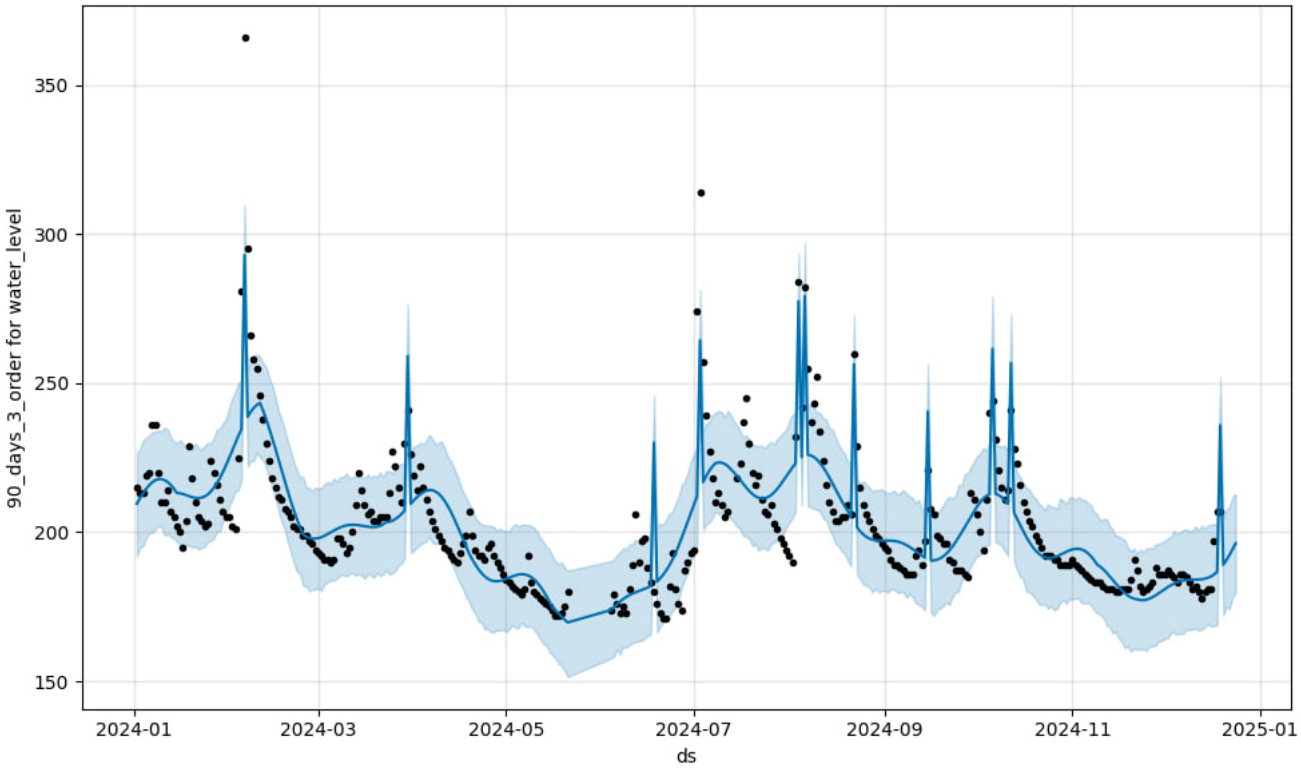


Рисунок 3.2 – Результат тренування моделі Prophet_90_days_3_order

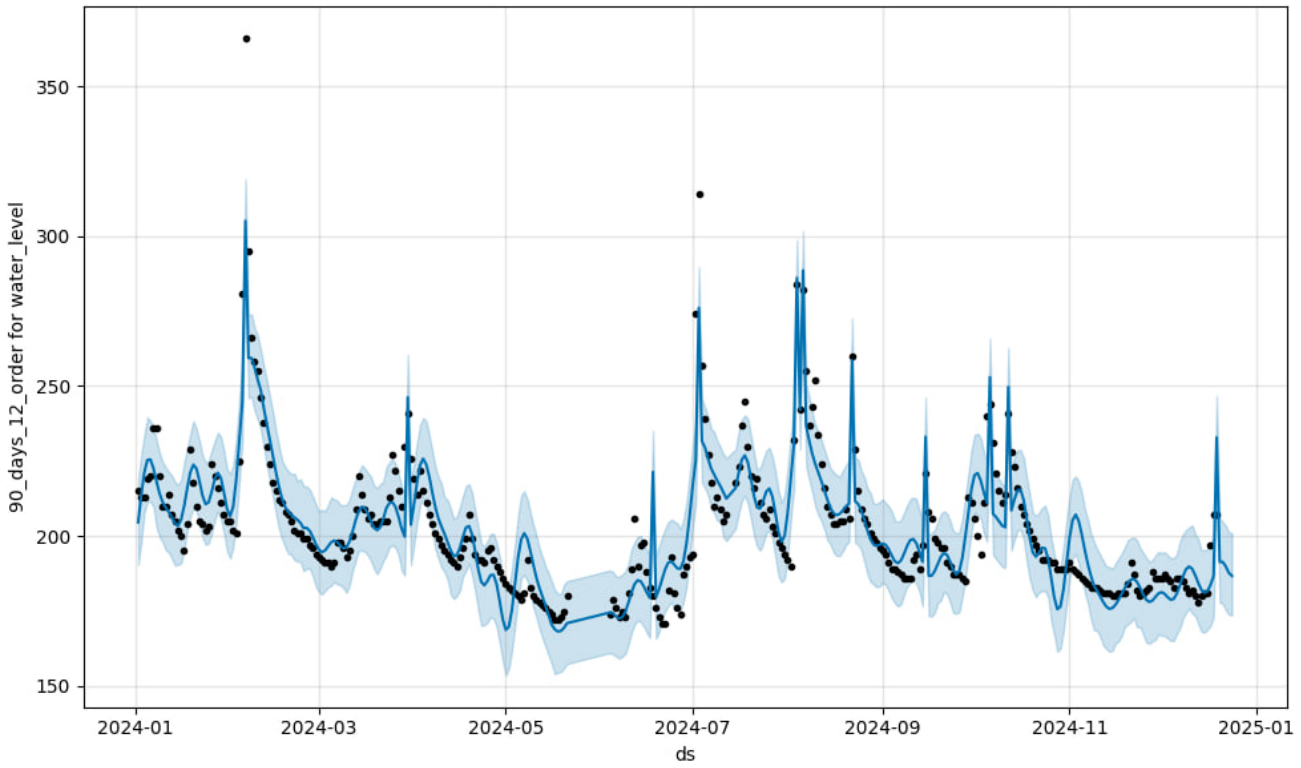


Рисунок 3.3 – Результат тренування моделі Prophet_90_days_12_order

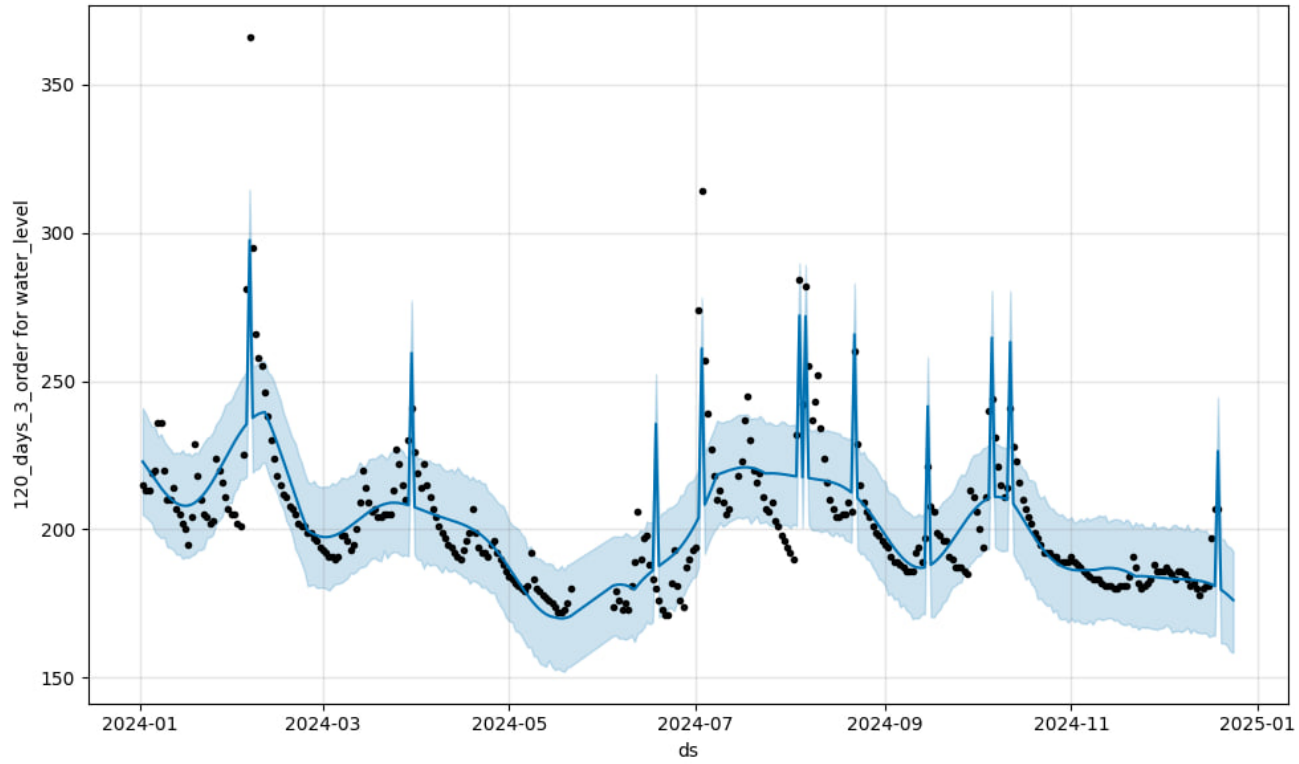


Рисунок 3.4 – Результат тренування моделі Prophet_120_days_3_order

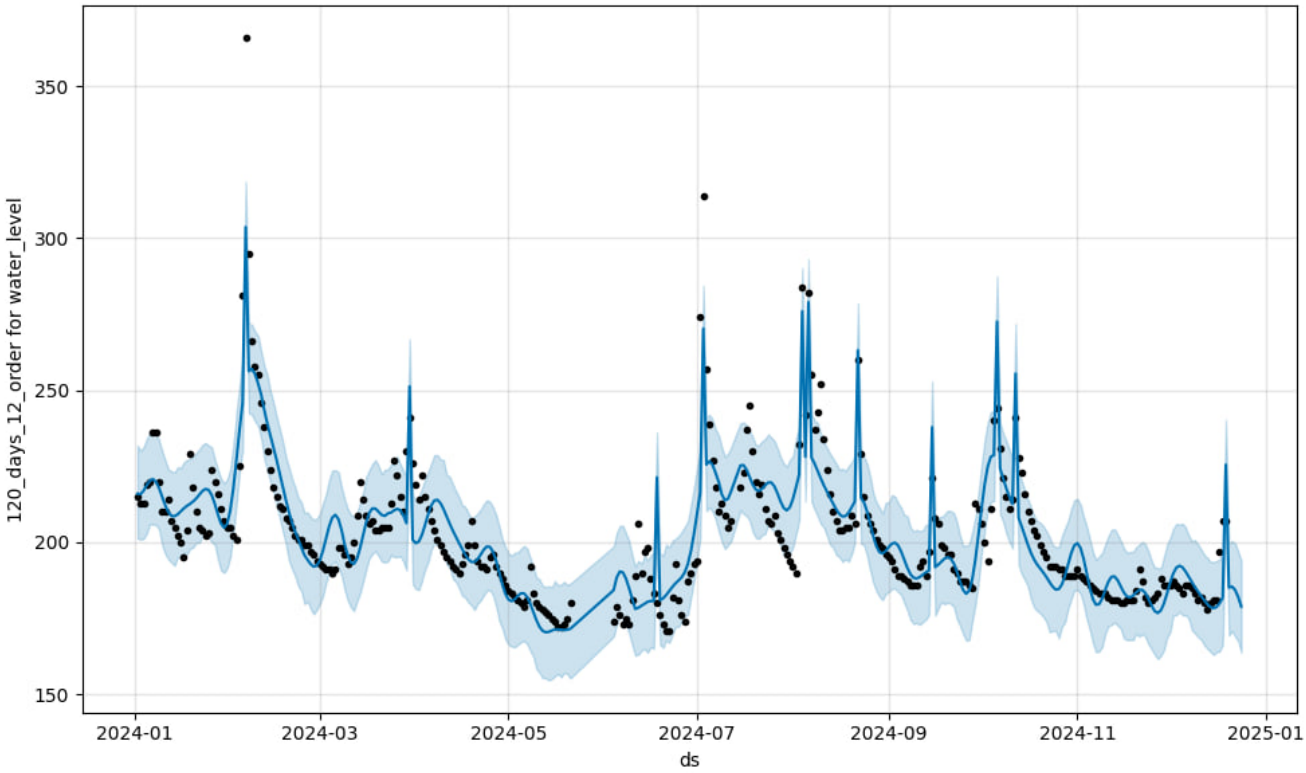


Рисунок 3.5 – Результат тренування моделі Prophet_120_days_12_order

Крім прогнозу, Prophet також генерує графіки компонент прогнозу – тренду та сезонності. На них видно, що сезонна компонента моделі суттєво впливає на кінцевий результат прогнозування. Порівнюючи варіанти із різними параметрами (рис. 3.6 – 3.9), можна оцінити вплив довжини сезонного періоду та кількості гармонік (порядку Фур'є-розкладу) на точність і стабільність передбачення.

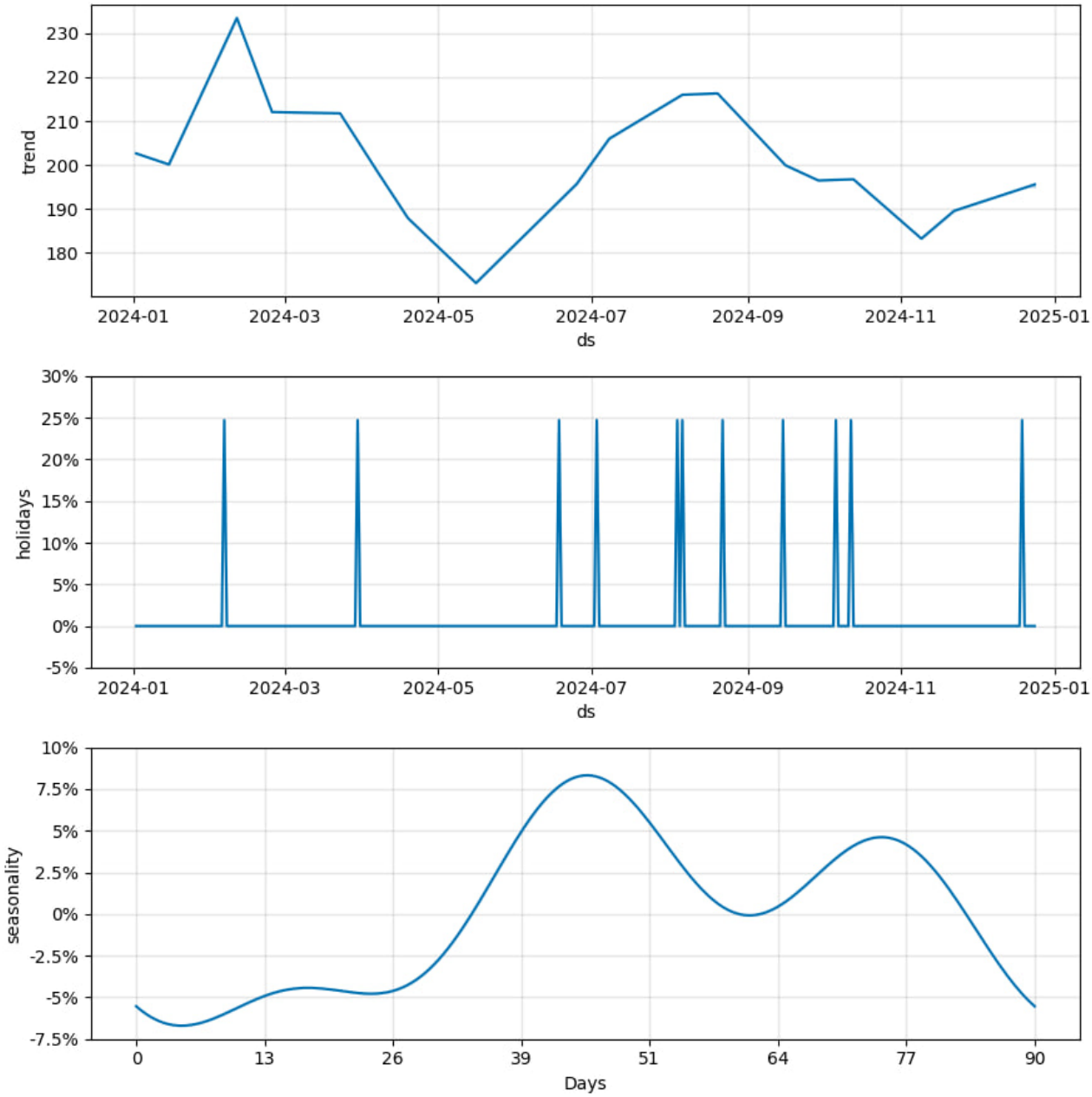


Рисунок 3.6 – Компоненти моделі Prophet_90_days_3_order

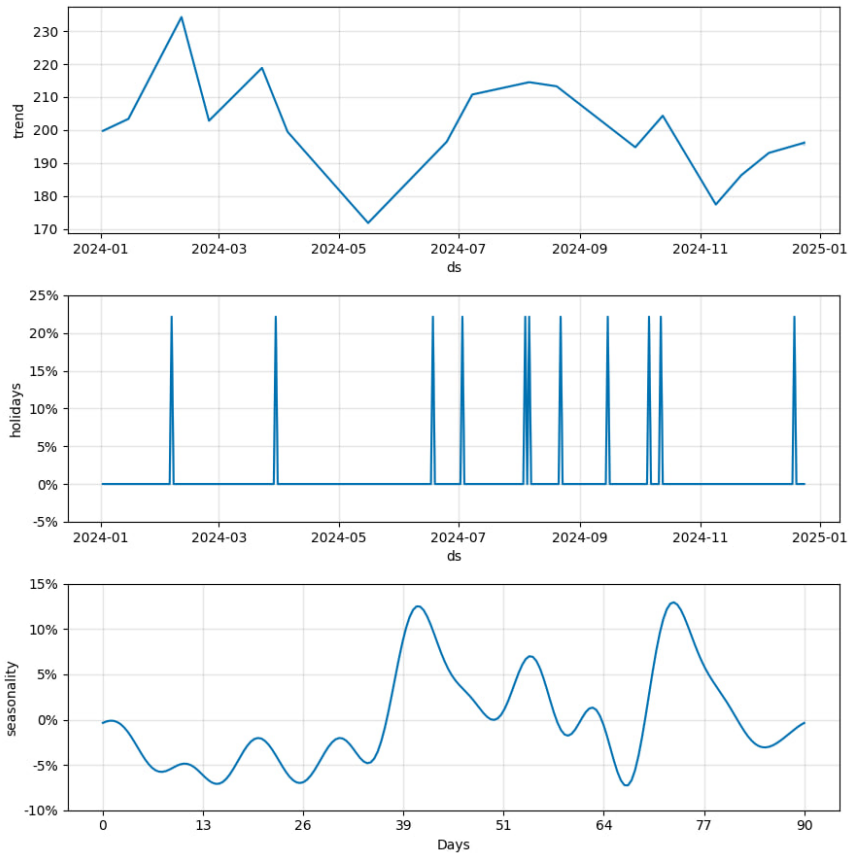


Рисунок 3.7 – Компоненти моделі Prophet_90_days_12_order

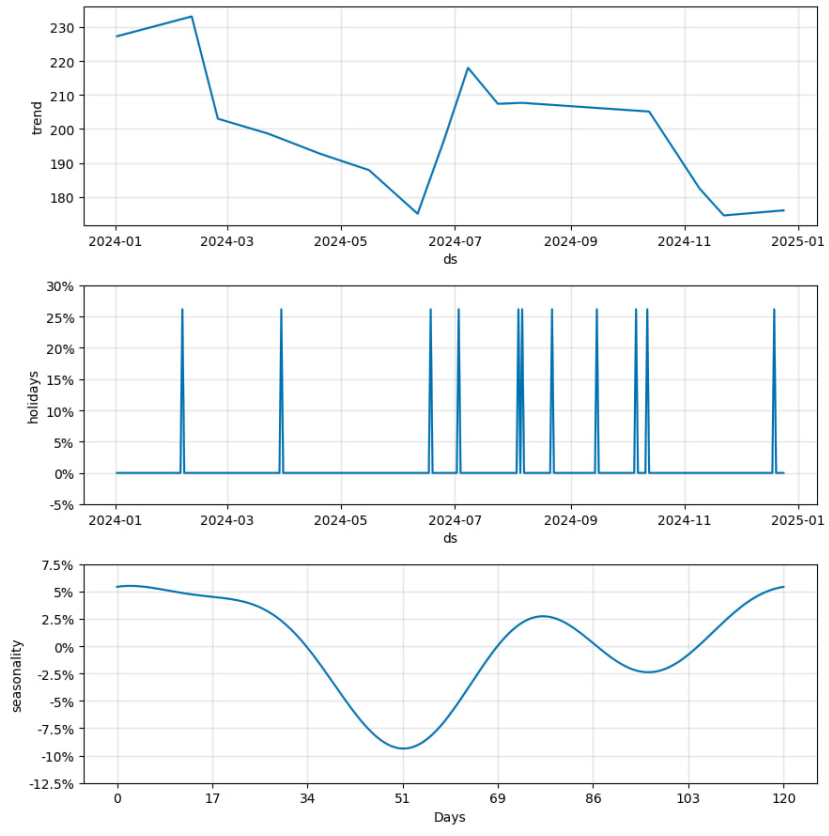


Рисунок 3.8 – Компоненти моделі Prophet_120_days_3_order

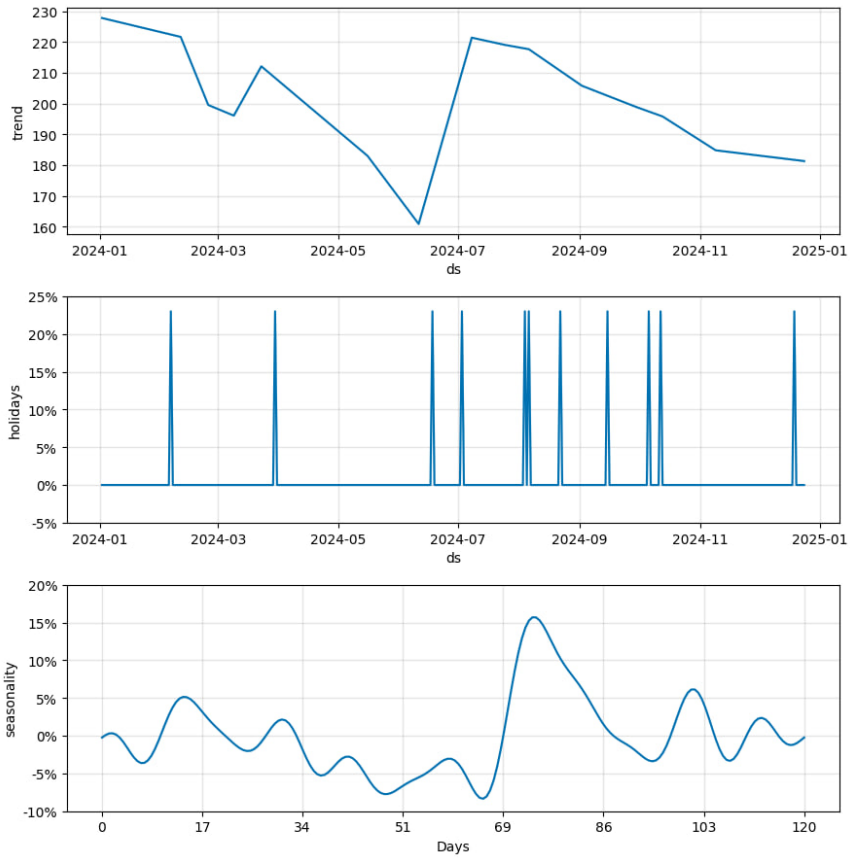


Рисунок 3.9 – Компоненти моделі Prophet_120_days_12_order

Наступною було побудовано та проведено тренування моделі ARIMA. Ця модель використовується для моделювання та передбачення часових рядів, що демонструють автокорельовану природу та потенційну нестабільність. У рамках проведення системного аналізу використовувалась бібліотека pmdarima, яка дозволяє автоматизувати вибір оптимальних параметрів моделі за допомогою функції auto_arima.

Як і при побудові моделі Prophet, спочатку було виконано розбиття даних на тренувальну, валідаційну та тестову вибірки. Як показано на відповідному виводі коду (рис. 3.10), усього використано 348 спостережень: 338 рядків увійшли до тренувального набору, 5 – до валідаційного та 5 до тестового.

In [50]:

```
# Get datasets
if is_ARIMA:
    train_ts, valid_ts, test_ts, train_valid_ts = get_
train_valid_test_ts(df2.copy(), forecasting_days, targ
et='Close')
```

```
Origin dataset has 348 rows and 2 features
Get training dataset with 338 rows
Get validation dataset with 5 rows
Get test dataset with 5 rows
```

Рисунок 3.10 – Розбиття даних на тренувальний, валідаційний і тестовий набори для побудови моделі ARIMA

Далі, для оцінювання необхідності диференціювання та визначення порядків авторегресивної та ковзної середньої компонент було побудовано графіки автокореляційної (ACF) та часткової автокореляційної (PACF) функцій для часового ряду без диференціювання (рис. 3.11 – 3.14).

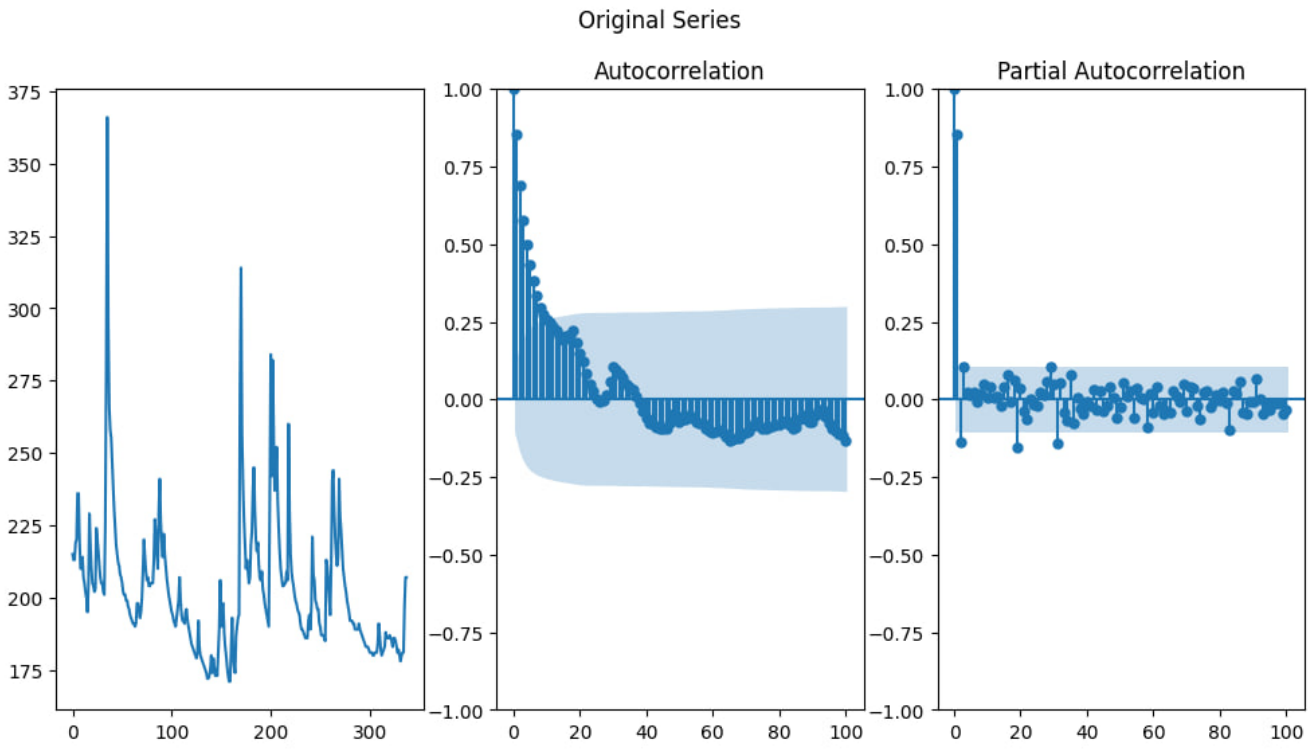


Рисунок 3.11 – ACF та PACF для вихідного часового ряду (Original Series)

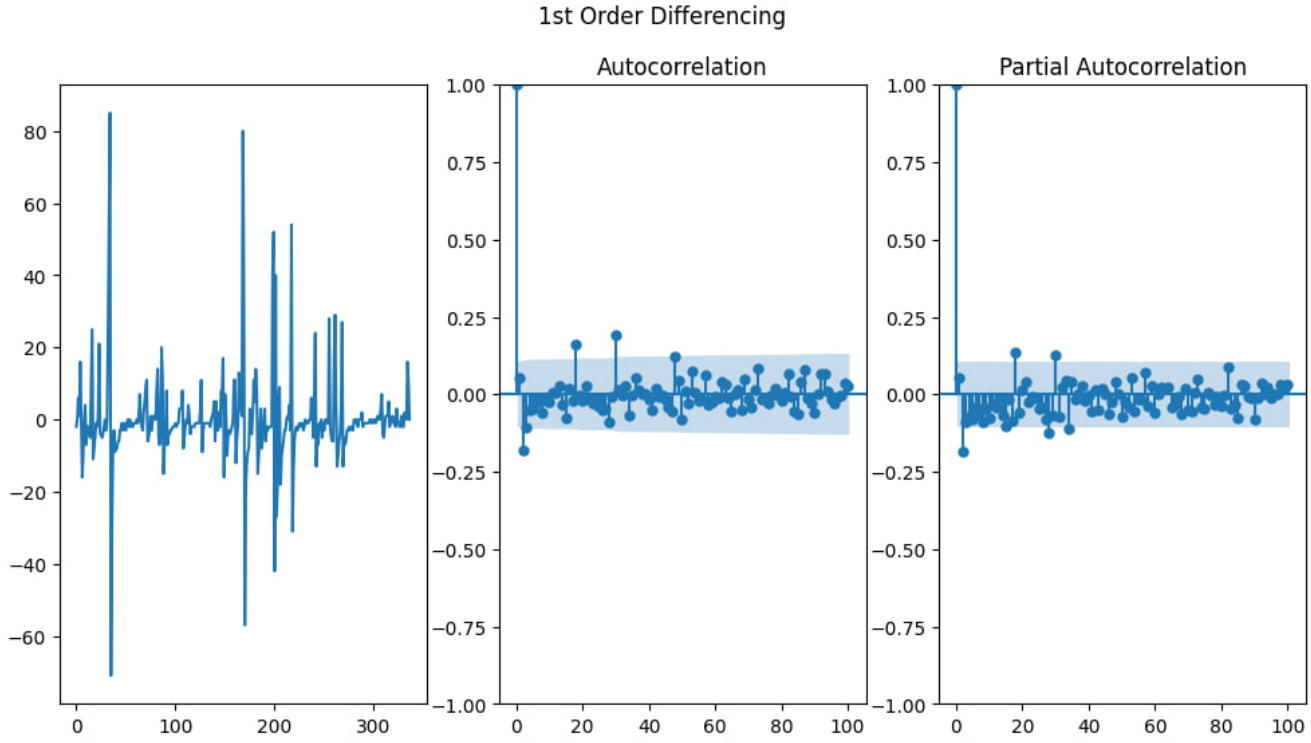


Рисунок 3.12 – ACF та PACF для ряду після першого диференціювання (1st Order Differencing)

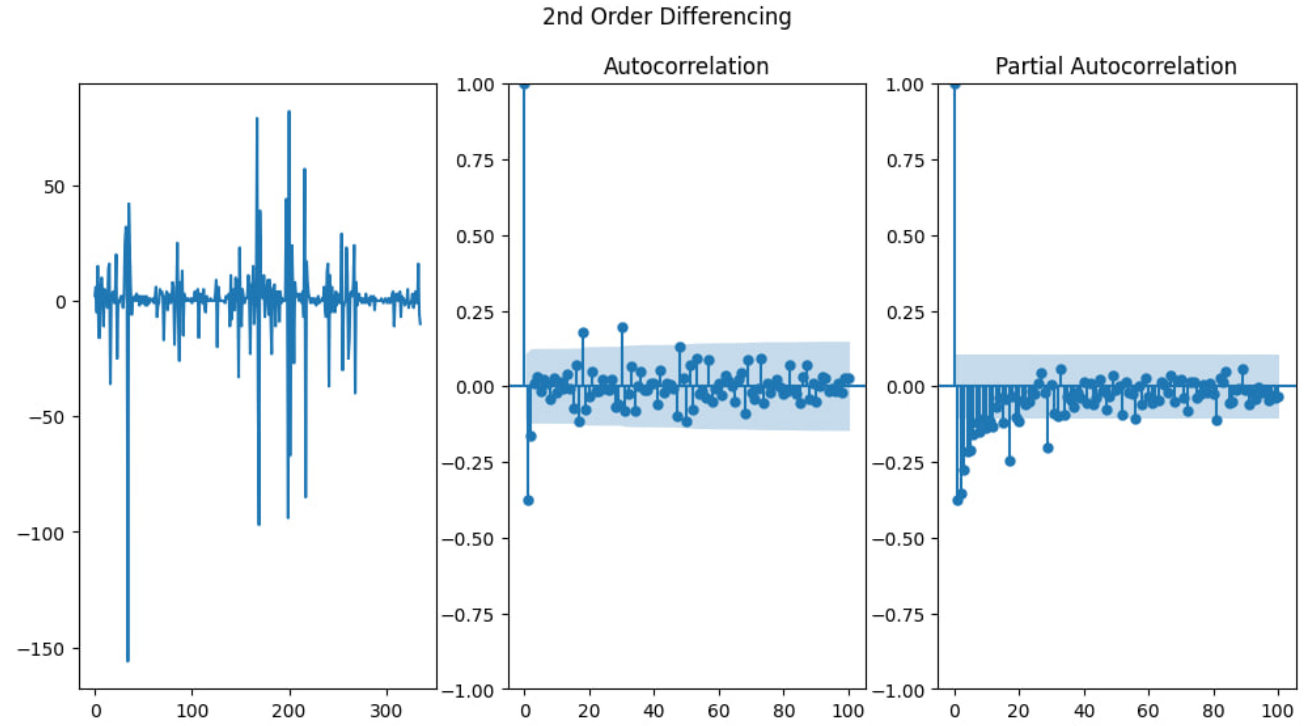


Рисунок 3.13 – ACF та PACF для ряду після другого диференціювання (2nd Order Differencing)

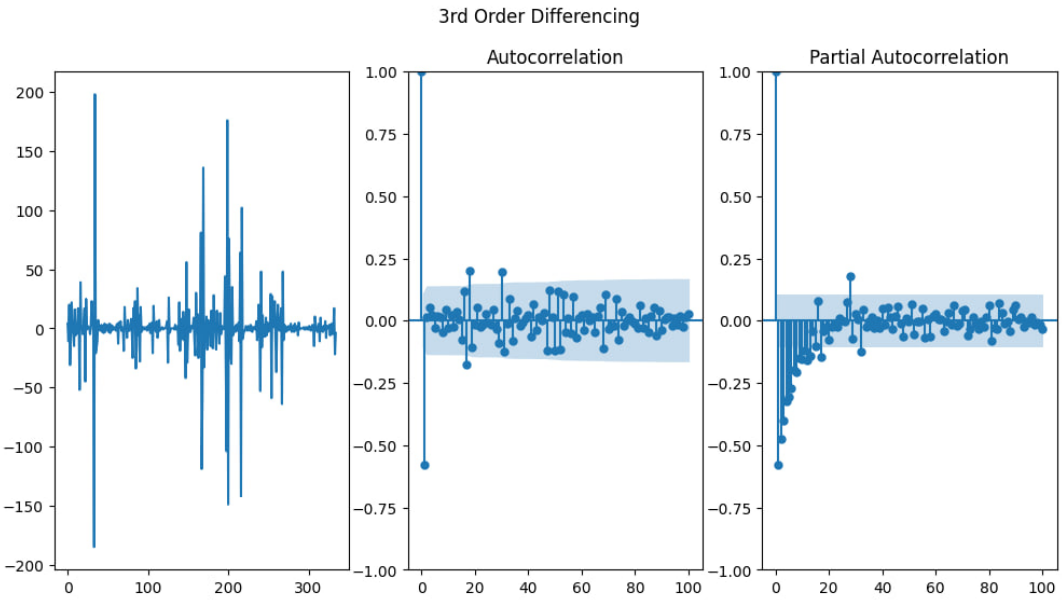


Рисунок 3.14 – ACF та PACF для ряду після третього диференціювання (3rd Order Differencing)

Згідно з результатами візуалізації, ряд уже на першому кроці не демонструє чіткої стаціонарності, що свідчить про потенційну необхідність застосування інтегрування (I-компоненти в ARIMA). Однак на наступних кроках стає очевидним, що надмірне диференціювання призводить до надмірної згладженості та втрати інформації.

Для автоматичного підбору оптимальних гіперпараметрів p, d, q застосовано функцію auto_arima з покроковим перебором (stepwise=True). У процесі перебору моделей було проаналізовано понад 20 комбінацій параметрів, обчислено відповідні критерії AIC для кожної моделі та відібрано найкращу – ARIMA(1,0,1) з константою (intercept) (рис. 3.15).

```
Performing stepwise search to minimize aic
 ARIMA(4,0,4)(0,0,0)[0]             : AIC=inf, Time=1.07 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=4556.595, Time=0.02 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=inf, Time=0.03 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=4120.376, Time=0.20 sec
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=2689.333, Time=0.06 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=2688.474, Time=0.22 sec
 ARIMA(2,0,0)(0,0,0)[0]             : AIC=inf, Time=0.05 sec
 ARIMA(3,0,1)(0,0,0)[0]             : AIC=2664.985, Time=0.43 sec
 ARIMA(3,0,0)(0,0,0)[0]             : AIC=inf, Time=0.08 sec
 ARIMA(4,0,1)(0,0,0)[0]             : AIC=2664.299, Time=0.60 sec
 ARIMA(4,0,0)(0,0,0)[0]             : AIC=inf, Time=0.11 sec
 ARIMA(5,0,1)(0,0,0)[0]             : AIC=2666.260, Time=0.70 sec
 ARIMA(4,0,2)(0,0,0)[0]             : AIC=2686.380, Time=0.89 sec
 ARIMA(3,0,2)(0,0,0)[0]             : AIC=2665.168, Time=0.46 sec
 ARIMA(5,0,0)(0,0,0)[0]             : AIC=inf, Time=0.23 sec
 ARIMA(5,0,2)(0,0,0)[0]             : AIC=inf, Time=0.95 sec
 ARIMA(4,0,1)(0,0,0)[0] intercept   : AIC=2657.714, Time=0.37 sec
 ARIMA(3,0,1)(0,0,0)[0] intercept   : AIC=2655.862, Time=0.39 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=2655.259, Time=0.59 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=2654.015, Time=0.39 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=2832.887, Time=0.12 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=2660.211, Time=0.17 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=2654.562, Time=0.48 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=3095.330, Time=0.02 sec
 ARIMA(0,0,2)(0,0,0)[0] intercept   : AIC=2735.159, Time=0.22 sec
 ARIMA(2,0,0)(0,0,0)[0] intercept   : AIC=2655.639, Time=0.13 sec
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=2656.291, Time=0.31 sec

Best model:  ARIMA(1,0,1)(0,0,0)[0] intercept
Total fit time: 9.323 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  338
Model:               SARIMAX(1, 0, 1)   Log Likelihood               -1323.007
Date:                Tue, 03 Jun 2025   AIC                           2654.015
Time:                        13:14:24   BIC                           2669.307
Sample:                             0   HQIC                          2660.109
                                  - 338
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     41.1903      5.930      6.946      0.000      29.568      52.812
ar.L1          0.7970      0.026     30.997      0.000       0.747       0.847
ma.L1          0.1977      0.035      5.655      0.000       0.129       0.266
sigma2       146.4377      4.792     30.559      0.000     137.046     155.830
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):              5722.79
Prob(Q):                              0.89   Prob(JB):                         0.00
Heteroskedasticity (H):               0.26   Skew:                             2.94
Prob(H) (two-sided):                  0.00   Kurtosis:                        22.28
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
CPU times: user 36.5 s, sys: 39.2 ms, total: 36.5 s
Wall time: 9.35 s
```

Рисунок 3.15 – Автоматичний вибір параметрів ARIMA та діагностика моделі

Як видно з підсумкового звіту, модель має коефіцієнт авторегресії ar.L1 = 0.797 та ковзної середньої ma.L = 0.1977. Усі параметри є статистично значущими (p-value < 0.001), а дисперсія залишків (sigma2 = 146.4) свідчить про відносно

низьку варіативність прогнозних помилок. Статистика Ljung-Box (p = 0.89) підтверджує відсутність автокореляції залишків першого лагу, що є позитивною ознакою адекватності моделі.

Після фіксації оптимальної структури моделі виконано побудову діагностичних графіків (рис. 3.16), які включають залишки, щільність розподілу, нормальність та автокореляцію.

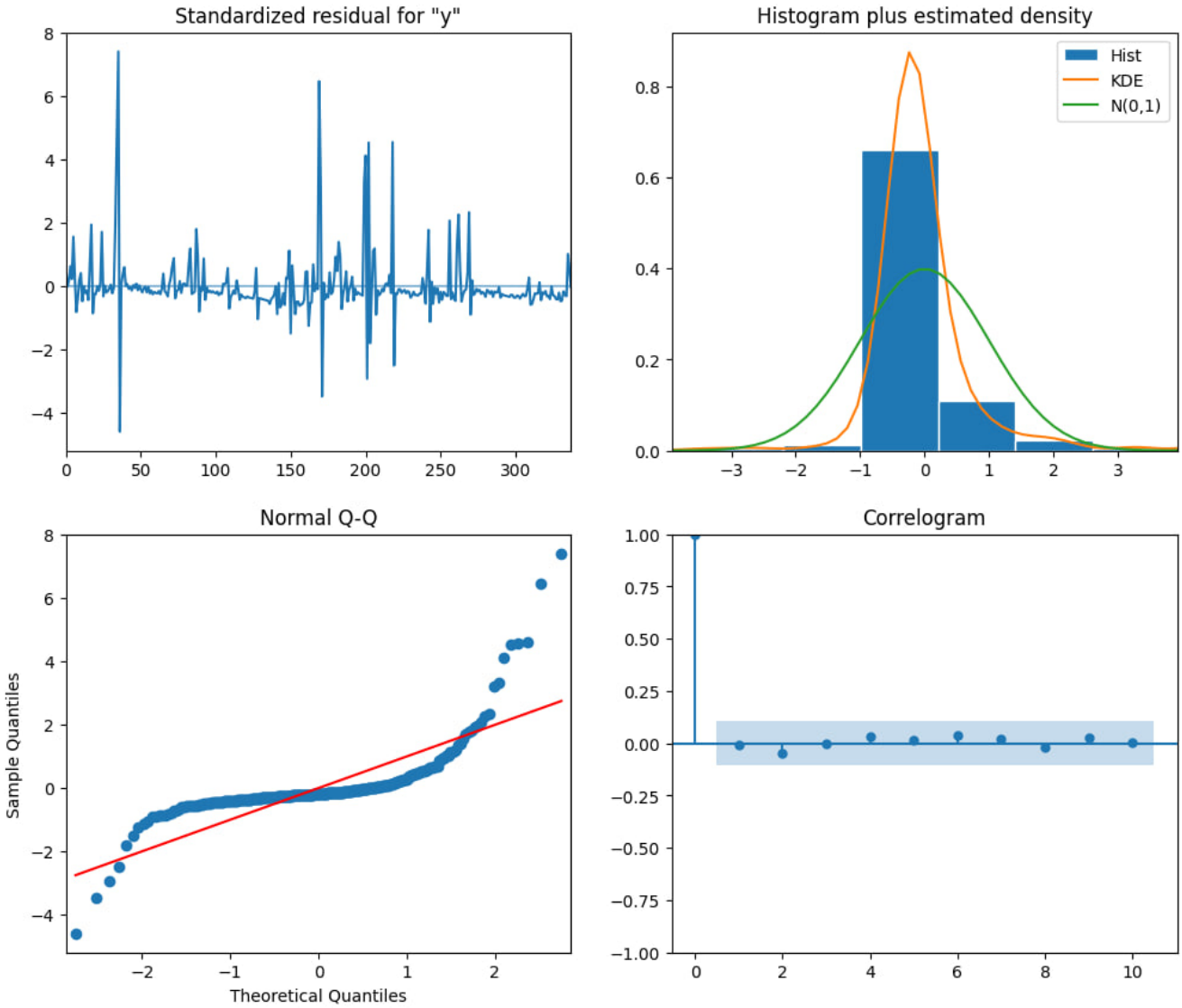


Рисунок 3.16 – Діагностика залишків моделі ARIMA(1,0,1)

Залишки коливаються навколо нульового рівня, що свідчить про відсутність систематичних похибок. Графік щільності демонструє асиметричний розподіл із відхиленням від нормального. Водночас автокореляція залишків не є статистично

значущою, що вказує на їхню часову незалежність і підтверджує адекватність обраної моделі.

Отже, побудована модель ARIMA(1,0,1) виявилася статистично значущою, із задовільними характеристиками залишків та здатністю до прогнозування на валідаційному наборі даних. Вона була використана як одна з базових моделей для подальшого порівняння з іншими алгоритмами, зокрема Prophet та моделями машинного навчання.

Після побудови моделей Facebook Prophet та ARIMA було здійснено побудову та тренування групи моделей машинного навчання, що працюють із багатофакторними ознаками. Для цього на етапі інженерії ознак було сформовано мультифакторний датасет із 79 характеристик за допомогою TSFRESH. Дані було розбито на тренувальну валідаційну та тестову вибірки (рис. 3.17).

```
In [61]: # Get datasets
         if is_other_ML:
             df2 = get_target_mf(df2, forecasting_days, col='y')
             train_mf, ytrain_mf, valid_mf, yvalid_mf, test_mf, y
         test_mf, train_valid_mf, y_train_valid_mf, starting_poin
         t = \
                                              get_train_valid_test
         _mf(df2.copy(), forecasting_days, target='target')

         Origin dataset has 343 rows and 79 features
         Get training dataset with 333 rows
         Get validation dataset with 5 rows
         Get test dataset with 5 rows
```

Рисунок 3.17 – Розбиття даних на тренувальний, валідаційний і тестовий набори для побудови мультифакторних моделей машинного навчання

Було реалізовано навчання восьми моделей: Linear Regression, KNeighbors Regressor, Support Vector Machines (SVR), Linear SVR, Random Forest Regressor, XGB Regressor та MLP Regressor. Для кожної моделі здійснювався підбір

гіперпараметрів методом GridSearchCV, що дозволило отримати оптимальні налаштування на основі точності передбачень на валідаційній вибірці (рис. 3.18).

In [64]:
```
%%time
if is_other_ML:
    # Models tuning and the forecasting
    result, model_all = model_prediction(result, models, train_mf, valid_mf, ytrain_mf, yvalid_m
f)
```

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

Tuning model 'KNeighbors Regressor'
Best parameters: {'leaf_size': 10, 'n_neighbors': 10}

Tuning model 'Support Vector Machines'
Best parameters: {'C': 1.0, 'kernel': 'linear', 'tol': 0.001}

Tuning model 'Linear SVR'
Best parameters: {'C': 7.0}

Tuning model 'Random Forest Regressor'
Best parameters: {'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 5, 'min_samples_
split': 20, 'n_estimators': 60}

Tuning model 'Bagging Regressor'
Best parameters: {'max_features': 0.8061224489795918, 'n_estimators': 5, 'warm_start': False}

Tuning model 'XGB Regressor'
Best parameters: {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 30}

Tuning model 'MLP Regressor'
Best parameters: {'hidden_layer_sizes': 2, 'learning_rate': 'adaptive', 'learning_rate_init':
0.001, 'max_iter': 1000, 'solver': 'sgd'}

CPU times: user 5min 39s, sys: 1.08 s, total: 5min 40s
Wall time: 3min 39s
```

Рисунок 3.18 – Результати підбору оптимальних гіперпараметрів для моделей машинного навчання

Модель лінійної регресії продемонструвала найкращу якість при врахуванні вільного члена (fit_intercept=True), що є типовим у випадках наявності зсуву в часовому ряді. Для моделі KNeighbors Regressor найкращими виявилися параметри leaf_size=10 та n_neighbors=10, що відповідає локальному усередненню на основі 10 найближчих спостережень. Підтримуючі векторні машини (SVR) виявили

найкращі результати за лінійного ядра (kernel=’linear’) з регуляризаційним параметром C=1.0 та точністю tol=0.001. У моделі Linear SVR оптимальним виявилося значення параметра C=7.0, що забезпечило баланс між точністю і стабільністю. У випадку Random Foresr Regressor оптимальні параметри включали max_depth=4, n_estimators=60, min_samples_leaf=5, що відповідає компактним, але ефективним ансамблям дерев. Для Bagging Regressor найкращим виявився ансамбль із 5 оцінювачів при використанні близько 80% ознак (max_features≈0.81). Градієнтний бустинг (XGB Regressor) досяг найкращих результатів при низькій швидкості навчання (learning_rate=0.01), невеликій глибині дерев (max_depth=3) та кількості ітерацій n_estimators=30. Остання модель – багатошарова перцептронна регресія (MLP Regressor) показала найкращий результат при використанні лише 2 нейронів у прихованому шарі, адаптивному режимі навчання (learning_rate=’adaptive’) та стохастичному градієнтному спуску (solver=’sgd’).

Отже, на даному етапі було побудовано та налаштовано різні типи моделей: статистичні (Prophet і ARIMA), а також мультифакторні моделі машинного навчання (регресійні, ансамблеві, нейромережеві). Кожна з моделей проходила автоматичний підбір гіперпараметрів та валідацію на окремому піднаборі даних. Отримані результати заклали основу для подальшого порівняння точності прогнозування та вибору найкращої моделі.

3.3 Вибір оптимальної моделі та результати прогнозування рівня води

З метою порівняння ефективності побудованих моделей прогнозування рівня води проведено оцінювання їхньої точності на валідаційній вибірці. Для цього використовувалися дві основні метрики: середньоквадратична помилка (RMSE) та середня абсолютна відносна похибка (MAPE), які дозволяють кількісно оцінити точність відтворення часових рядів.

RMSE є квадратним коренем з MSE. Математично, він вимірює стандартне відхилення помилки. Аналогічно до MSE, RMSE широко використовується в регресіях та оцінюванні моделей, що вимагають числових прогнозів.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}\,. \quad (3.1)$$

Відповідно до тієї ж логіки, RMSE також схильний до впливу викидів, але значно меншою мірою. Однак, враховуючи цю чутливість до викидів, ця метрика є важливою для їх ідентифікації в моделях. RMSE також можна пояснити простим способом, оскільки він вимірюється в тих самих одиницях, що й прогнозована змінна, таких як грошові одиниці (наприклад, долар, євро). Завдяки своїй легкій зрозумілості, RMSE широко використовується для порівняння продуктивності різних моделей. Іншими словами, якщо є кілька моделей та алгоритмів, найнижче значення RMSE вказує на найточнішу [22].

MAPE обчислює середнє значення абсолютних відсоткових відхилень між прогнозами моделі та фактичними значеннями. Отже, ця метрика виражає середню помилку у відсотках від фактичного значення.

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|. \quad (3.2)$$

MAPE сильніше штрафує негативні помилки (коли прогнозоване значення перевищує фактичне). Це пояснюється тим, що відсоткова похибка не може перевищувати 100% для дуже низьких прогнозів, тоді як для вищих прогнозів немає верхньої межі. Отже, MAPE схильна надавати перевагу моделям, які недооцінюють значення, а не переоцінюють [22].

На рисунку 3.19 наведено таблиця зі значеннями RMSE та MAPE для всіх побудованих моделей на валідаційних даних:

	name_model	type_data	rmse	mape
3	Prophet_120_days_12_order	valid	5.774616	1.990526
1	Prophet_90_days_12_order	valid	4.958168	2.507532
0	Prophet_90_days_3_order	valid	10.004556	4.696355
2	Prophet_120_days_3_order	valid	10.193718	4.752827
4	ARIMA_auto	valid	18.796533	9.748766
7	Support Vector Machines	valid	19.604184	10.238392
8	Linear SVR	valid	20.571144	10.715677
11	XGB Regressor	valid	21.790427	11.294953
5	Linear Regression	valid	22.396888	11.642071
9	Random Forest Regressor	valid	23.039506	11.963893
10	Bagging Regressor	valid	23.459753	12.274336
6	KNeighbors Regressor	valid	23.741567	12.398002
12	MLP Regressor	valid	24.24602	12.571904

Рисунок 3.19 – Порівняльна таблиця точності моделей

Проаналізувавши отриманий результат видно, що найнижче значення RMSE = 4.96 досягнуто для моделі Prophet з параметрами сезонності 90 днів і порядком Фур’є 12, що свідчить про її здатність точно відтворювати абсолютні значення рівня води. Тоді як найменше значення MAPE = 1.99% продемонструвала модель Prophet із сезонністю 120 днів та порядком 12, що свідчить про її вищу стабільність прогнозів у відносному вимірі.

Для остаточного тестування було обрано ці дві конфігурації моделі Prophet як оптимальні відповідно до кожної з метрик. Вони були додатково натреновані на повному об’єднаному наборі train+valid, після чого виконано прогноз на тестовому наборі даних (рис. 3.20).

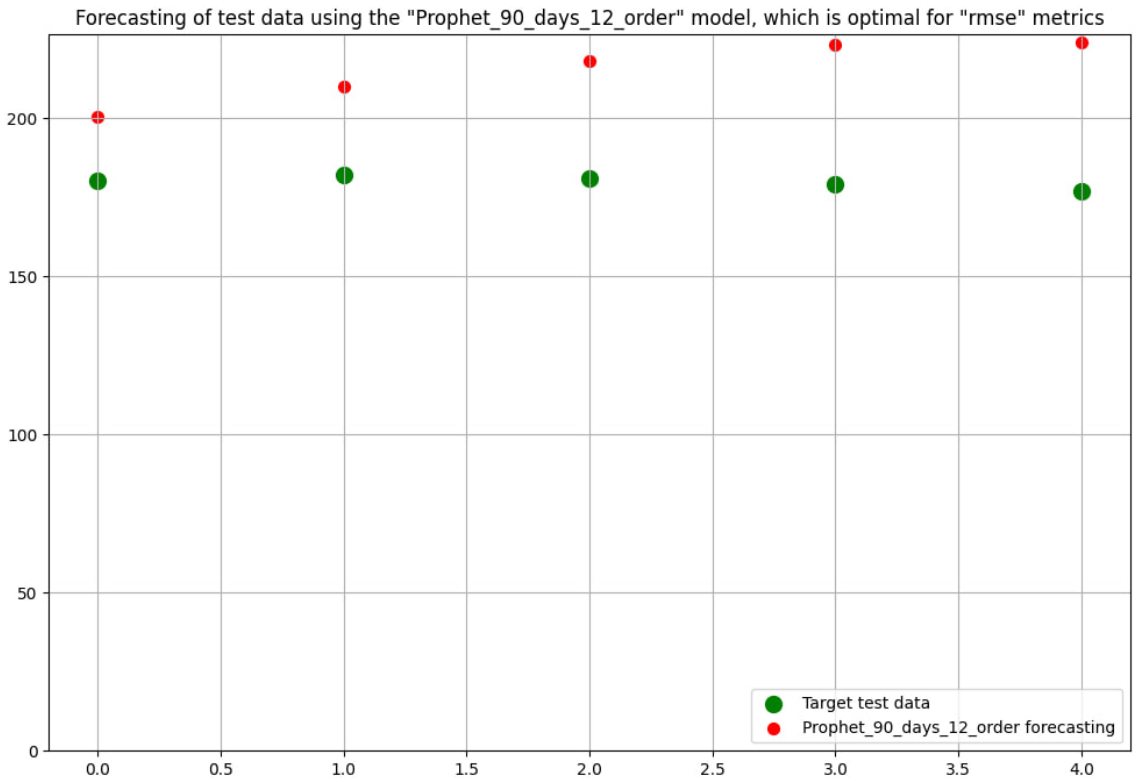


Рисунок 3.20 – Результати 5-денного прогнозування рівня води на тестовому наборі даних за допомогою моделі Prophet_90_days_12_order

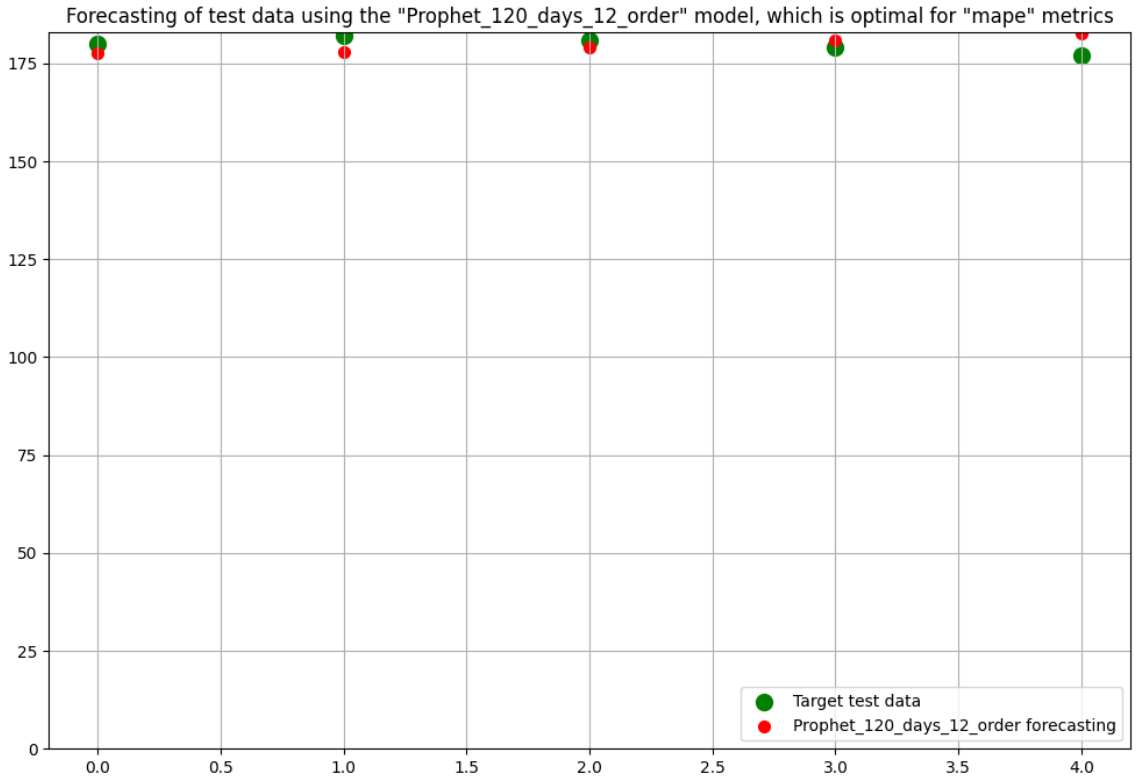


Рисунок 3.21 – Результати 5-денного прогнозування рівня води на тестовому наборі даних за допомогою моделі Prophet_120_days_12_order

Зелені точки на графіках, зображених на рисунку 3.20 – 3.21, відповідають реальним значенням рівня води, зібраним у тестовому наборі, тоді як червоні точки позначають прогноз, згенерований навченими моделями. Обидві моделі відтворюють загальну динаміку змін рівня води з високою точністю, однак візуальний аналіз показує, що модель Prophet з 120-денною сезонністю забезпечує кращу відповідність між прогнозними та фактичними значеннями.

Отже, врахувавши результати аналізу чисельних метрик та графічного представлення результатів прогнозування, можна зробити висновок, що найбільш оптимальною та ефективною для розв’язання задачі короткострокового прогнозування рівня води виявилася модель Prophet із параметрами сезонності 120 днів та Фур’є-порядком 12 (RMSE = 5.77, MAPE = 1.99%). Її застосування дозволяє досягнути збалансованого поєднання абсолютної точності та відносної надійності прогнозів.

3.4 Висновки

У третьому розділі було описано, реалізовано та виконано порівняння ряду моделей, що використовувалися для задачі короткострокового прогнозування рівня води в річці Дністер. До аналізу було залучено як спеціалізовані алгоритми для обробки часових рядів (Facebook Prophet, AutoARIMA), так і універсальні регресійні моделі класичного машинного навчання (Linear Regression, KNeighbors Regressor, Support Vector Machines, Linear SVR, Random Forest Regressor, Bagging Regressor, XGB Regressor та MLP Regressor). Для кожної моделі було проведено автоматичний підбір гіперпараметрів із використанням GridSearchCV або внутрішніх механізмів оптимізації. Усі моделі було навчено та протестовано на однаково розбитих даних, що забезпечило справедливість порівняння.

Результати валідації показали перевагу спеціалізованих моделей, орієнтованих на часові ряди, зокрема Prophet, який у конфігурації з сезонністю 120 днів і Фур’є-порядком 12 досяг найнижчого значення MAPE (1.99%) та продемонстрував стабільну точність на тестовому наборі даних.

Класичні моделі машинного навчання показали задовільні, але гірші

результати, що свідчить про обмеження їх застосування в задачах, де ключовим є збереження часової послідовності.

Таким чином, аналіз точності прогнозів, графічного відтворення та стабільності моделювання дозволяє зробити висновок, що найбільш доцільною для задачі короткострокового прогнозування рівня води є модель Prophet_120_days_12_order. Вона поєднує переваги високої абсолютної та відносної точності, стабільності в умовах сезонних змін і прозорості моделі, що є критично важливим у практичному гідрологічному застосуванні.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи було охарактеризовано водогосподарські ділянки басейну річки Дністер як складну просторово-часову систему, що включає множину взаємопов’язаних параметрів, зокрема обсяги водозабору, природного стоку, втрат, екологічного стоку, дефіциту та резервів. Такий підхід дозволив розглядати водогосподарський баланс не як статичну таблицю показників, а як динамічну модель, що змінюється у часі і просторі, потребуючи багатофакторного аналізу для виявлення закономірностей і критичних зон.

Проведено аналіз проблематики системного управління водними ресурсами в басейні Дністра, зокрема складності врахування просторово-часової мінливості дефіциту води та гідрологічної забезпеченості. Розглянуто існуючі підходи до аналізу водогосподарських балансів, визначено їхні обмеження у частині інтегрованої візуалізації, динамічного аналізу та автоматизованого прогнозування на основі фактичних гідрологічних спостережень.

З метою реалізації поставлених завдань було зібрано та підготовлено гідрологічні дані з відкритих джерел, виконано приведення до уніфікованого формату, а також структуризацію показників водогосподарського балансу за місяцями та водогосподарськими ділянками.

На основі обробленого набору даних було реалізованого інструменти для візуалізації дефіциту та резервів води з урахуванням показника гідрологічної забезпеченості. Проведений системний аналіз дозволяє ідентифікувати закономірності водного дефіциту та виявити критичні ділянки із нестачею води. Це дає змогу ухвалювати обґрунтовані управлінські рішення на рівні окремих водогосподарських ділянок у межах басейну.

Для вирішення задачі прогнозування рівня води було протестовано кілька моделей машинного навчання, зокрема Prophet, ARIMA, Random Forest, XGB та MLP Regressor. Найвищу точність серед усіх моделей продемонструвала модель Prophet з параметрами сезонності 120 днів та порядком Фур’є 12, яка досягла

значень метрик RMSE = 5.77 та MAPE = 1.99%. Такий результат свідчить про високу здатність моделі адаптуватися до сезонних коливань рівня води.

Узагальнення результатів аналізу дозволило виявити характерні просторові та часові закономірності дефіциту водних ресурсів у басейні Дністра, що зумовлені сезонною нерівномірністю надходження води, інтенсивністю заборів та варіативністю гідрологічної забезпеченості. Запропонований підхід може слугувати основою для розроблення системи підтримки прийняття рішень у сфері управління водними ресурсами, зокрема в частині балансування дефіциту, ідентифікації зон ризику та прогнозного планування у водогосподарській практиці.

За результатами даної роботи була зроблена доповідь на тему «Системний аналіз водогосподарських балансів ділянок басейну Дністра» на Міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» з публікацією тез [1].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гайович А. М., Крижановський Є. М. Системний аналіз водогосподарських балансів ділянок басейну Дністра. *Міжнародна науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи (Вінниця, 2024-2025 рр.)»*. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/25476/21148.

2. План управління річковим басейном Дністра на 2025-2030 роки : Затверджено розпорядженням Кабінету Міністрів України від 01.11.2024 р. № 1078-р. URL: https://davr.gov.ua/fls18/PURBDniester7_ukr.pdf.

3. Про затвердження Порядку розроблення водогосподарський балансів : Наказ Міністерства екології та природних ресурсів України від 17.02.2017 р. №232/30100. URL: https://zakon.rada.gov.ua/laws/show/z0232-17#Text.

4. Побудова та прогнозування водогосподарського балансу водогосподарської ділянки М5.1.4.45 р. Горинь від витоку до кордону Хмельницької та Рівненської областей / Керівник: Мокін В. Б. Вінниця : ВНТУ, 2024. 119 с. URL: https://nrat.ukrintei.ua/searchdoc/0224U033394/.

5. Мокін В. Б., Мокін О. Б., Давидюк О. М., Гораш М. А., Лучко А. М., Варчук І. В., Дратований М. В Передбачення показників якості води у річці з використанням моделей часових рядів і методів бібліотеки Prophet : матеріали II Міжнародної науково-практичної конференції, XLIX науково-технічної конференції підрозділів ВНТУ, Вінниця, 27-28 квітня 2020 р. Вінниця, 2020. URL: http://ir.lib.vntu.edu.ua/bitstream/handle/123456789/29808/%D0%9C%D0%BE%D0%BA%D1%96%D0%BD.pdf?sequence=1&isAllowed=y.

6. Python. URL: https://www.python.org/.

7. Pandas documentation. URL: https://pandas.pydata.org/docs/.

8. Matplotlib: Visualization with Python. URL: https://matplotlib.org/.

9. Seaborn: statistical data visualization. URL: https://seaborn.pydata.org/.

10. GeoPandas 1.0.1. URL: https://geopandas.org/en/stable/.

11. Prophet. URL: https://facebook.github.io/prophet/.

12. ARIMA. URL: https://www.sktime.net/en/latest/api_reference/auto_generated/sktime.forecasting.arima.ARIMA.html.

13. pmdarima.arima.auto_arima. URL: https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html.

14. Support Vector Machines. URL: https://scikit-learn.org/stable/modules/svm.html.

15. Linear SVR. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVR.html.

16. Class XGBRegressor (2.4.0). URL: https://cloud.google.com/python/docs/reference/bigframes/latest/bigframes.ml.ensemble.XGBRegressor#methods.

17. LinearRegression. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.

18. RandomForestRegressor. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

19. BaggingRegressor. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html.

20. KNeighborsRegressor. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html.

21. MLPRegressor. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html.

22. Metrics Evaluation: MSE, RMSE, MAE and MAPE. URL: https://medium.com/@jonatasv/metrics-evaluation-mse-rmse-mae-and-mape-317cab85a26b.

23. Kaggle: Your Machine Learning and Data Science Community. URL: https://www.kaggle.com/.

24. Водогосподарський баланс для району басейну річки Дністер. URL: https://www.davr.gov.ua/fls18/balans_dnister.pdf.

25. Дані автоматичних гідрологічних постів. URL: https://www.meteo.gov.ua/ua/Dani-avtomatichnikh-hidrolohichnikh-postiv.

26. Kaggle Dataset «Water Balance Dnister». URL: https://www.kaggle.com/datasets/annhaiovych/dnister.

27. Kaggle Notebook «Dnister Balance EDA & Visualization». URL: https://www.kaggle.com/code/annhaiovych/dnister-balance-eda-visualization.

28. Data Science – Statistics Correlation Matrix. URL: https://www.w3schools.com/datascience/ds_stat_correlation_matrix.asp.

29. Folium. URL: https://python-visualization.github.io/folium/latest/.

30. Kaggle Notebook «Water level – EDA & forecasting». URL: https://www.kaggle.com/code/annhaiovych/water-level-eda-forecasting.

31. Tsfresh. Extract Features on Time Series Easily. URL: https://tsfresh.com/.

32. Maximilian Christ, Nils Braun, Julius Neuffer, Andreas W. Kempa-Liehr. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package), *Neurocomputing*. 2018. № 307. С. 72-77. URL: https://doi.org/10.1016/j.neucom.2018.03.067.

33. Мокін В. Б., Дратований М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних : електронний навчальний посібник комбінованого (локального та мережевого) використання. Вінниця : ВНТУ, 2024. 258 с. URL: https://pdf.lib.vntu.edu.ua/books/2024/Mokin_2024_263.pdf.

34. Шмундяк Д. О., Мокін В. Б. Системний аналіз стану природних середовищ з урахуванням аномалій. *Наукові праці Вінницького національного технічного університету*. Вінниця, 2024. № 2. URL: https://doi.org/10.31649/2307-5376-2024-4-63-73.

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
СИСТЕМНИЙ АНАЛІЗ ВОДОГОСПОДАРСЬКИХ БАЛАНСІВ ДІЛЯНОК БАСЕЙНУ ДНІСТРА
08-34.БКР.002.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Євгеній КРИЖАНОВСЬКИЙ
«__» _____ 2025 р.
Розробила студентка гр. СА-21б
_____ Анна ГАЙОВИЧ
«__» _____ 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.

2) Шмундяк Д. О., Мокін В. Б. Системний аналіз стану природних середовищ з урахуванням аномалій. *Наукові праці Вінницького національного технічного університету*. Вінниця, 2024. № 2.

3. Мета і призначення роботи.

Метою роботи є системний аналіз водогосподарських балансів ділянок басейну Дністра, включно з візуалізацією, аналізом дефіциту водних ресурсів та прогнозуванням рівня води із використанням моделей машинного навчання.

4. Вихідні дані для проведення робіт:

Дані про водогосподарський баланс для району басейну річки Дністер, дані автоматичних гідрологічних постів, шейп-файли водогосподарських ділянок річкових басейнів України.

5. Методи дослідження:

Аналіз гідрологічних даних, візуалізація показників водогосподарського балансу, застосування моделей машинного навчання (Prophet, ARIMA, Random Forest тощо), порівняння прогнозної точності за метриками RMSE і MAPE, програмна реалізація засобів аналізу та прогнозування в середовищі Python.

6. Етапи роботи і терміни їх виконання:

a) Аналіз предметної області _____ – _____
b) Порівняльний аналіз існуючих аналітичних досліджень _____ – _____
c) Вибір оптимальних інформаційних технологій _____ – _____
d) Системний аналіз водогосподарських балансів ділянок басейну Дністра для підтримки прийняття рішень щодо управління дефіцитом води _____ – _____
e) Оформлення матеріалів до захисту БКР _____ – _____

7. Очікувані результати та порядок реалізації

Системний аналіз водогосподарський балансів ділянок басейну Дністра для виявлення та візуалізації ділянок із дефіцитом водних ресурсів та прогнозування рівня води в річці Дністер із застосуванням моделей машинного навчання.

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.

Розробила студентка групи СА-21б _____ Анна ГАЙОВИЧ

Додаток Б

(обов’язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Системний аналіз водогосподарських балансів ділянок басейну Дністра»

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІІТА, гр. СА-21б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 4,12%

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)

Сергій ЖУКОВ, доц. каф. САІТ _____
(підпис)

Особа, відповідальна за перевірку _____ Сергій ЖУКОВ
(підпис)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Євгеній КРИЖАНОВСЬКИЙ, к.т.н., доц. каф. САІТ
(підпис)

Здобувач _____ Анна ГАЙОВИЧ
(підпис)

Додаток В

(довідниковий)

Фрагмент лістингу програми

```
# Set random state
def fix_all_seeds(seed):
    np.random.seed(seed)
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)

random_state = 42
fix_all_seeds(random_state)

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

    result = adfuller(series.values)

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

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

def seasonal_decompose_analysis(df, col, df_period, verbose=1):
    # Decomposition of the series df.y into components with given df_period
    # and analysis of the amplitude fractions of these components
    # relative to the maximum value of the given series

    # Series decompose
    decomp = seasonal_decompose(df[col], period=df_period)
    if verbose==1:
        fig = decomp.plot()
        fig.set_size_inches((10, 8))
        fig.tight_layout()
        plt.show()

    # Maximum value of series
    y_max = df[col].max()

    # Max & min of Trend
    trend_component = decomp.trend
    max_trend = trend_component.max()
    min_trend = trend_component.min()
    part_trend = round((max_trend-min_trend)*100/y_max,2)

    # Max & min of Seasonal
    seasonal_component = decomp.seasonal
    max_seasonal = seasonal_component.max()
    min_seasonal = seasonal_component.min()
    part_seasonal = round((max_seasonal-min_seasonal)*100/y_max,2)

    # Max & min of Resid
    resid_component = decomp.resid
    max_resid = resid_component.max()
    min_resid = resid_component.min()
```

```
    part_resid = round((max_resid-min_resid)*100/y_max,2)

    # Results output
    if verbose==1:
        print('Values:')
        print(f' * of Trend are from {min_trend} to {max_trend}')
        print(f' * of Seasonal are from {min_seasonal} to {max_seasonal}')
        print(f' * of Resid are from {min_resid} to {max_resid}')
        print(f'\nThe share of components from the amplitude of the series {round(y_max)} is:')
        print(f'' * Trend    = {part_trend}%'')
        print(f'' * Seasonal = {part_seasonal}%'')
        print(f'' * Resid    = {part_resid}%'')

    return y_max, part_trend, part_seasonal, part_resid

def seasonal_EDA(df, col):
    # EDA of seasonality of series df[col]
    res = pd.DataFrame(columns = ['period', 'part_trend', 'part_seasonal', 'part_resid'])
    max_period = int(len(df)/2)
    for i in range(1,max_period):
        y_max, part_trend, part_seasonal, part_resid = seasonal_decompose_analysis(df, col, i, verbose=0)
        res.loc[i, 'period'] = i
        res.loc[i, 'part_trend'] = part_trend
        res.loc[i, 'part_seasonal'] = part_seasonal
        res.loc[i, 'part_resid'] = part_resid

    # Result output
    display(res.sort_values(by=['part_seasonal'], ascending=False).head(20))
    res['part_seasonal'].rolling(window=7, closed='both').mean().plot(figsize=(10,8), grid=True)

    return df

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

    data = data.reset_index(drop=False).reset_index(drop=False)

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

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

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

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

    return extracted_features_clean

def plot_with_anomalies(df, cols_y_list, cols_y_list_name, dates_x, anomalous_dates, log_y=False):
    # Thanks to https://www.kaggle.com/vbmokin/covid-in-ua-prophet-with-4-nd-seasonality
    # Draws a plot with title - the features cols_y_list (y) and dates_x (x) from the dataframe df
    # and with vertical lines in the dates from the list anomalous_dates
    # with the length between the minimum and maximum of feature cols_y_list[0]
    # with log_y = False or True
    # cols_y_list - dictionary of the names of cols from cols_y_list (keys - name of feature, value - it's name for the plot
legend),
```

```
    # name of cols_y_list[0] is the title of the all plot

    fig = px.line(df, x=dates_x, y=cols_y_list[0], title=cols_y_list_name[cols_y_list[0]], log_y=log_y,
template='gridon',width=800, height=600)
    y_max = df[cols_y_list[0]].max()
    for i in range(len(cols_y_list)-1):
        fig.add_trace(go.Scatter(x=df[dates_x], y=df[cols_y_list[i+1]], mode='lines',
name=cols_y_list_name[cols_y_list[i+1]]))
        max_i = df[cols_y_list[i+1]].max()
        y_max = max_i if max_i > y_max else y_max

    y_min = min(df[cols_y_list[0]].min(),0)
    for i in range(len(anomalous_dates)):
        anomal_date = anomalous_dates[i]
        #print(anomal_date, y_min, y_max)
        fig.add_shape(dict(type="line", x0=anomal_date, y0=y_min, x1=anomal_date, y1=y_max, line=dict(color="red",
width=1)))
    fig.show()

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

def get_target_mf(df, forecasting_days, col='y'):
    # Get target as difference of the df[col]
    # Returns target which is shifted for forecasting_days days in the dataframe df
    # "Close" -> "Close_diff" -> "Target"
    col_diff = f"{col}_diff"
    df[col_diff] = df['y'].diff()
    df['target'] = df[col_diff].shift(-forecasting_days)
    df = df.drop(columns=[col_diff]).dropna()

    return df

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

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

    # Data smoothing
#     df.index = df.ds
#     df = df.drop(columns=['ds'])
#     df['y'] = df['y'].rolling(7).mean()
#     df = df.dropna().reset_index(drop=False)

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

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

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

```
    return train, valid, test, train_valid

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

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

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

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

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


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


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

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

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

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

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

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

    return result

def prophet_modeling(result,
                     series_name,
                     train,
                     test,
                     holidays_df,
                     period_days,
```

```
                fourier_order_seasonality,
                forecasting_period,
                name_model,
                type_data):
    # Performs FB Prophet model training for given train dataset, holidays_df and seasonality_mode
    # Performs forecasting with period by this model, visualization and error estimation
    # df - dataframe with real data in the forecasting_period
    # can be such combinations of parameters: train=train, test=valid or train=train_valid, test=test
    # Save results into dataframe result

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

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

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

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

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

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

    return result, ypred

def acf_pacf_draw(df, lag_num=40, acf=True, pacf=True, title="", ylim=1):
    # Draw plots named title with ACF and PACF for dataframe df

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

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

```
        ax[1].set(ylim=(-ylim, ylim))

        if pacf:
            # PACF drawing
            plot_pacf(df.values.squeeze(), lags=lag_num, ax=ax[2])
            ax[2].set(ylim=(-ylim, ylim))

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

    fig.suptitle(title)
    plt.show()

def arima_fit(df, col, order=(1,1,1)):
    # ARIMA model fitting for series df[col]

    model = sm.tsa.arima.ARIMA(df[col].values.squeeze(), order=order)
    model = model.fit()
    return model

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

def arima_forecasting(result, model, params, name_model, df, type_data):
    # Data df (validation or test) forecasting on the num days by the model
    # with params and save metrics to result

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

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

    return result

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

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

    # Results
    model_all = []

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

```
            model_all.append(model)
            print(f"Best parameters: {model.best_params_}\n")

            # Prediction
            ypred = model.predict(valid_features)

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

    return result, model_all

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

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

def result_recover_and_metrics(result, df_ts, type_data, start_points):
    # Recovering prediction: from shifted_Close_diff to Close
    # Calculation metrics for recovering ypred forecasting for all models in result
    # ypred real is from df_ts['y']
    # start points value for the recovering is from dictionary start_points
    # type_data = 'valid' or 'test'

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

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

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

    return result

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

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

    elif name_model=='KNeighbors Regressor':
        model = KNeighborsRegressor(**params)

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

    elif name_model=='Linear SVR':
        model = LinearSVR(**params)
```

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

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

    elif name_model=='MLP Regressor':
        model = MLPRegressor(**params)

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

    else: model = None

    return model

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

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

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

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

    return opt_name_model, opt_type_model, opt_params_model

def model_training_forecasting(result, df, y, test, ytest,
                               name_model, type_model, params, type_test='1'):
    # Model training for df and y
    # Forecasting ypred
    # type_model = 'Prophet' or "ARIMA" or 'Other ML'
    # type_test = '1' (with find optimal parameters by GridSearchCV)
    # type_test = '2' (with optimal parameters - without GridSearchCV)
    # return params and metrics in the dataframe result

    if type_model=='Prophet':
        season_days_optimal = params[0]
        fourier_order_seasonality_optimal = params[1]
        model_opt = None
        _, ypred = prophet_modeling(result,
                                    series_name,
                                    df,
                                    test,
                                    holidays_df,
                                    season_days_optimal,
                                    fourier_order_seasonality_optimal,
                                    forecasting_days,
```

```
                        f'{type_model}_optimal',
                        'test')
    elif type_model=='ARIMA':
        season_days_optimal = params[0]
        fourier_order_seasonality_optimal = params[1]
        model_opt = None

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

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

        # Plot residual errors
        get_residual_errors(model_opt)

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

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

        # Forecasting
        ypred = model_opt.predict(test)


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

    return result, model_opt, ypred

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


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

```
            y_train_valid = train_valid_ts['y'].copy()
            test = test_ts.copy()
            ytest = test_ts['y'].copy()

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

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

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

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

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

        return opt_name_model
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

СИСТЕМНИЙ АНАЛІЗ ВОДОГОСПОДАРСЬКИХ БАЛАНСІВ ДІЛЯНОК БАСЕЙНУ ДНІСТРА

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

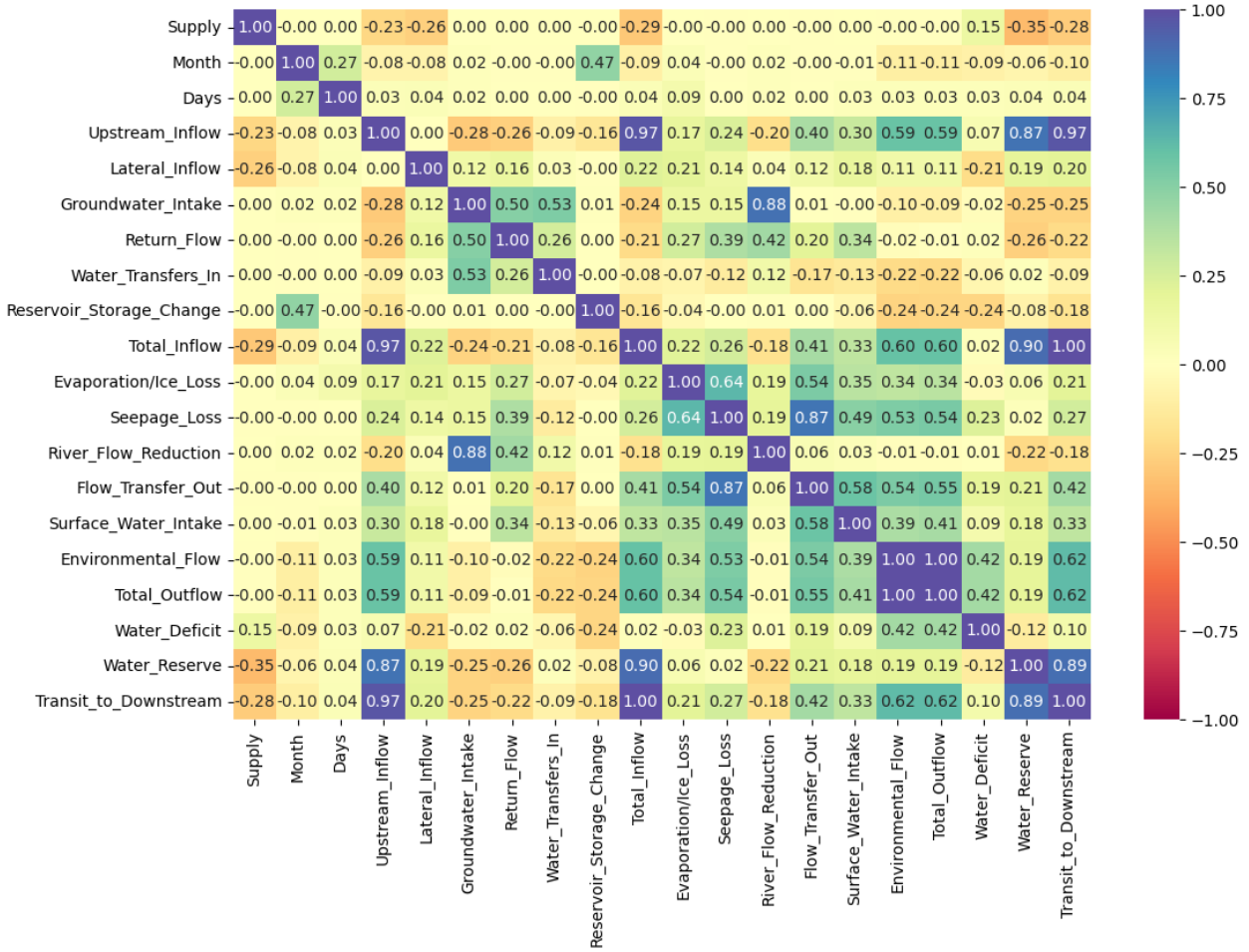


Рисунок Г.1 – Кореляційна матриця

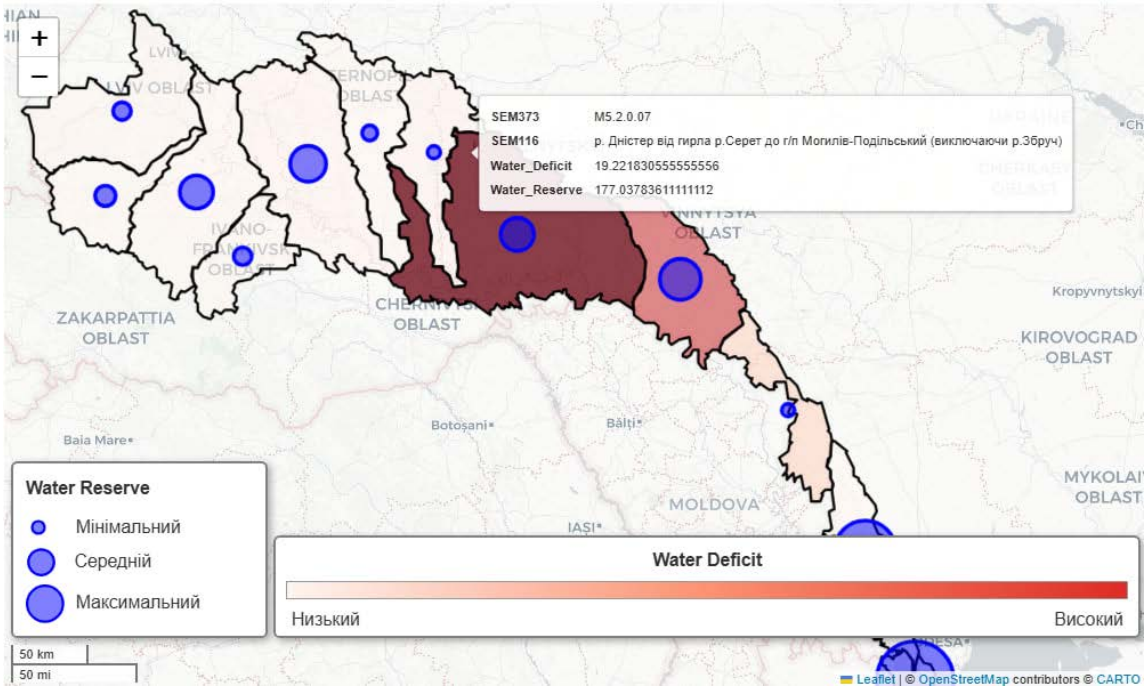


Рисунок Г.2 – Інтерактивна карта дефіцитів та резервів води по водогосподарських ділянках

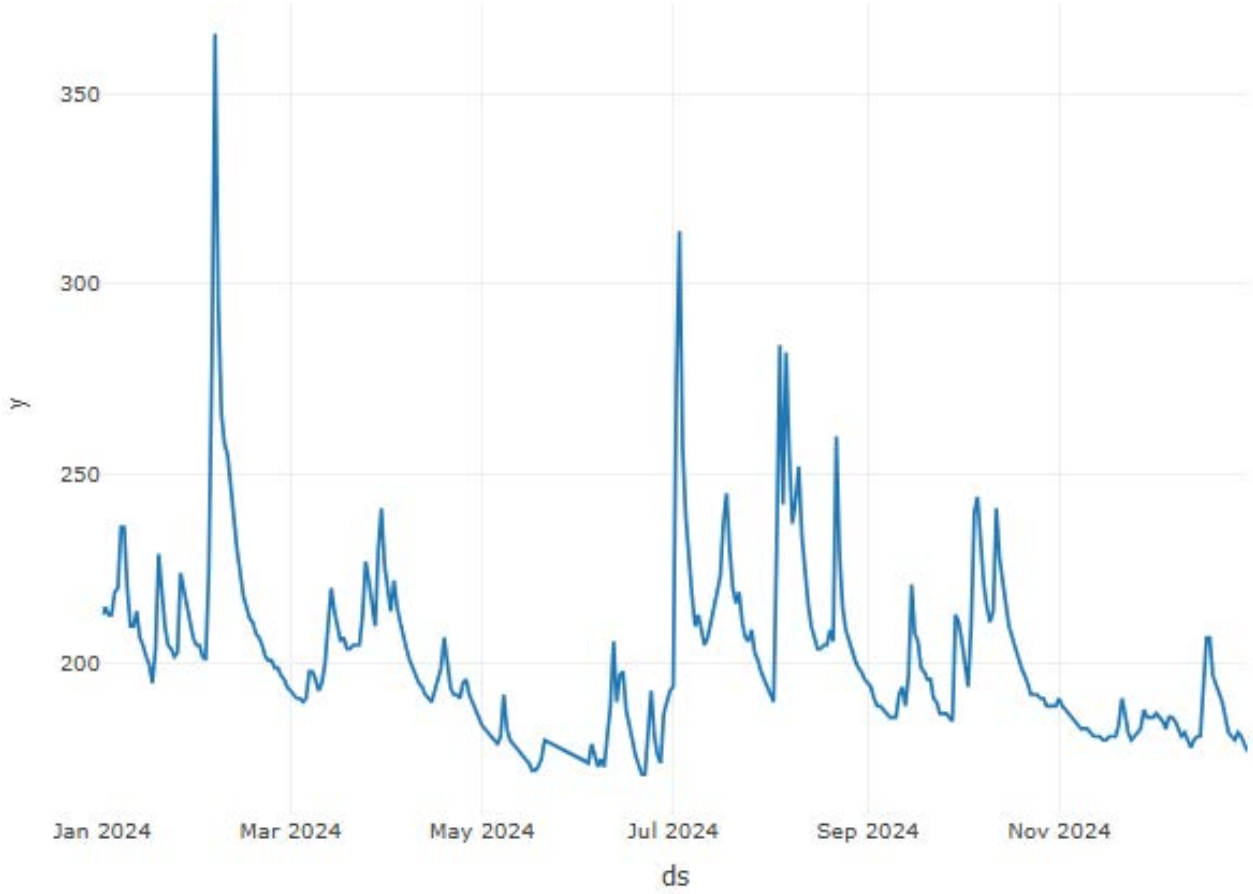


Рисунок Г.3 – Графік середніх добових рівнів води на гідропості Самбір

	name_model	type_data	rmse	mape
3	Prophet_120_days_12_order	valid	5.774616	1.990526
1	Prophet_90_days_12_order	valid	4.958168	2.507532
0	Prophet_90_days_3_order	valid	10.004556	4.696355
2	Prophet_120_days_3_order	valid	10.193718	4.752827
4	ARIMA_auto	valid	18.796533	9.748766
7	Support Vector Machines	valid	19.604184	10.238392
8	Linear SVR	valid	20.571144	10.715677
11	XGB Regressor	valid	21.790427	11.294953
5	Linear Regression	valid	22.396888	11.642071
9	Random Forest Regressor	valid	23.039506	11.963893
10	Bagging Regressor	valid	23.459753	12.274336
6	KNeighbors Regressor	valid	23.741567	12.398002
12	MLP Regressor	valid	24.24602	12.571904

Рисунок Г.4 – Порівняльна таблиця точності моделей

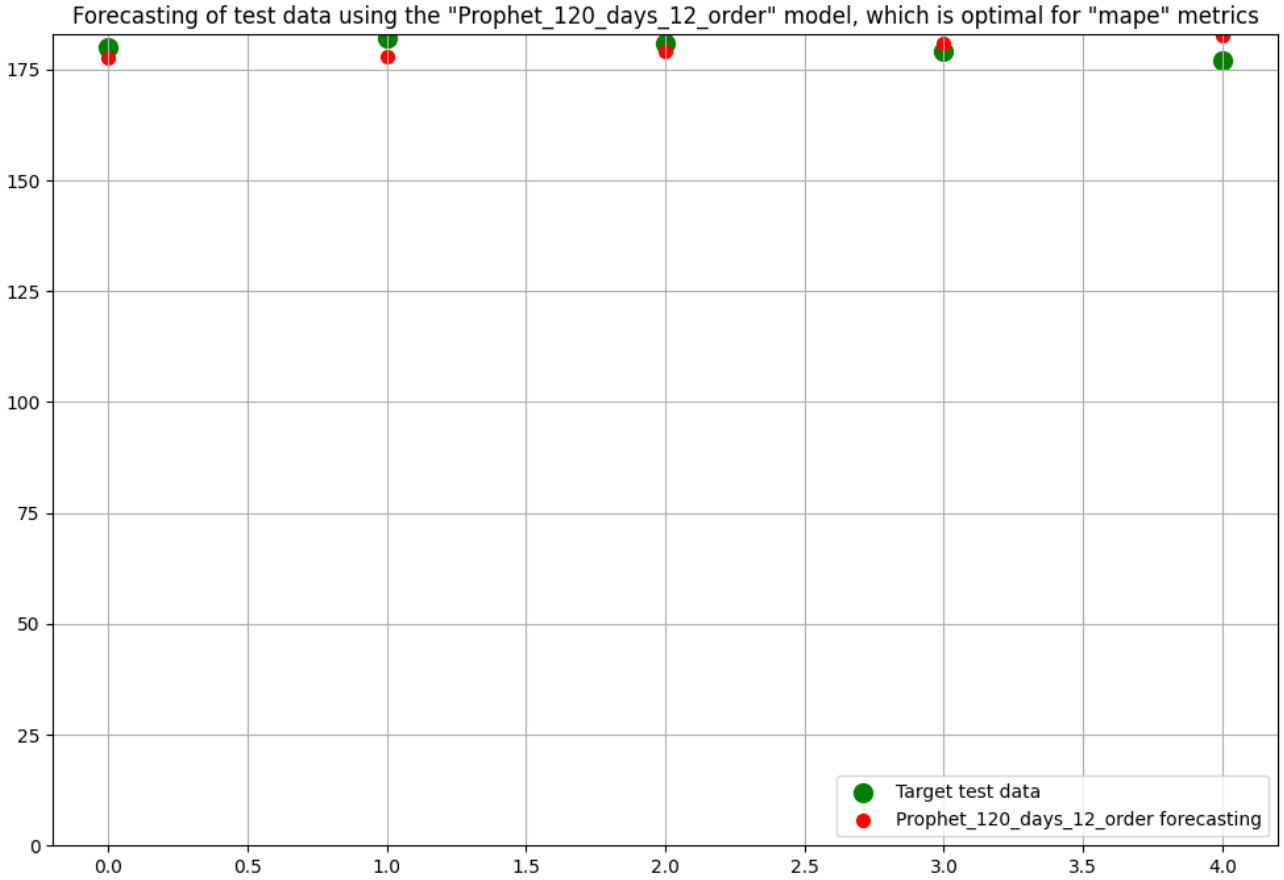


Рисунок Г.5 – Результати 5-денного прогнозування рівня води на тестовому наборі даних за допомогою моделі Prophet_120_days_12_order