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 СУББАСЕЙНУ ПРИП'ЯТІ»

Виконала: студентка 4 курсу, групи СА-21б
спеціальності 124 – «Системний аналіз»
_____ Марина СТРУТИНСЬКА
Керівник: к.т.н., доц. каф. САІТ
_____ Євгеній КРИЖАНОВСЬКИЙ
«08» 06 2025 р.
Рецензент: к.т.н., доц. кафедри КН
_____ Володимир ОЗЕРАНСЬКИЙ
«08» 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" 03 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ
Струтинській Марині Василівні

1. Тема роботи: «Системний аналіз водогосподарських балансів ділянок суббасейну Прип'яті»
керівник роботи: Крижановський Є. М., к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «28» 02 2025 року № 97
2. Строк подання студентом роботи «31» 05 2025 року
3. Вихідні дані до роботи:
1) Дані про водогосподарський баланс для району суббасейну річки Прип'ять;
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 водогосподарських ділянок;
3) Графік витрат води;
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» 03 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	01.04	20.04	вик
2	Порівняльний аналіз існуючих аналітичних досліджень	20.04	30.04	вик
3	Вибір оптимальних інформаційних технологій	01.05	10.05	вик
4	Системний аналіз водогосподарський балансів ділянок суббасейну Прип'яті для підтримки прийняття рішень щодо управління резервом води	10.05	20.05	вик
5	Оформлення матеріалів до захисту БКР	20.05	30.05	вик

Студент _____ Марина СТРУТИНСЬКА

Керівник роботи _____ Євгеній КРИЖАНОВСЬКИЙ

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 77 сторінок формату А4, на яких є 41 рисунків, список використаних джерел містить 30 найменування.

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

У роботі проведено системний аналіз водогосподарських балансів ділянок суббасейну Прип’яті. 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 суббасейну Прип’яті 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, суббасейн Прип’яті.

ABSTRACT

The bachelor’s thesis consists of 77 A4 pages, which contain 41 figures, the list of sources used contains 30 titles.

The aim of the work is to conduct a systems analysis of water management balances in the Prypyat River subbasin 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 Prypyat River subbasin 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, Prypyat subbasin.

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. Суббасейн річки Прип’ять має велике екологічне та соціально-економічне значення для західних і північних регіонів України, забезпечуючи водопостачання для населення, сільського господарства та промисловості. У той же час, у межах субббасейну спостерігаються періодичні досить невеликі резерви 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 проведення системного аналізу.

Мета і задачі дослідження. Метою дослідження є системний аналіз водогосподарських балансів ділянок суббасейну Прип’яті, 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 витрат 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 суббасейну річки Прип’ять 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 «Картографічна візуалізація результатів розрахунку балансів по водогосподарським ділянкам суббасейну Прип’яті» 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 виступає водогосподарська система ділянок суббасейну річки Прип'ять (рис. 1.1), що є однією з ключових складових гідрографічної мережі басейну Дніпра.

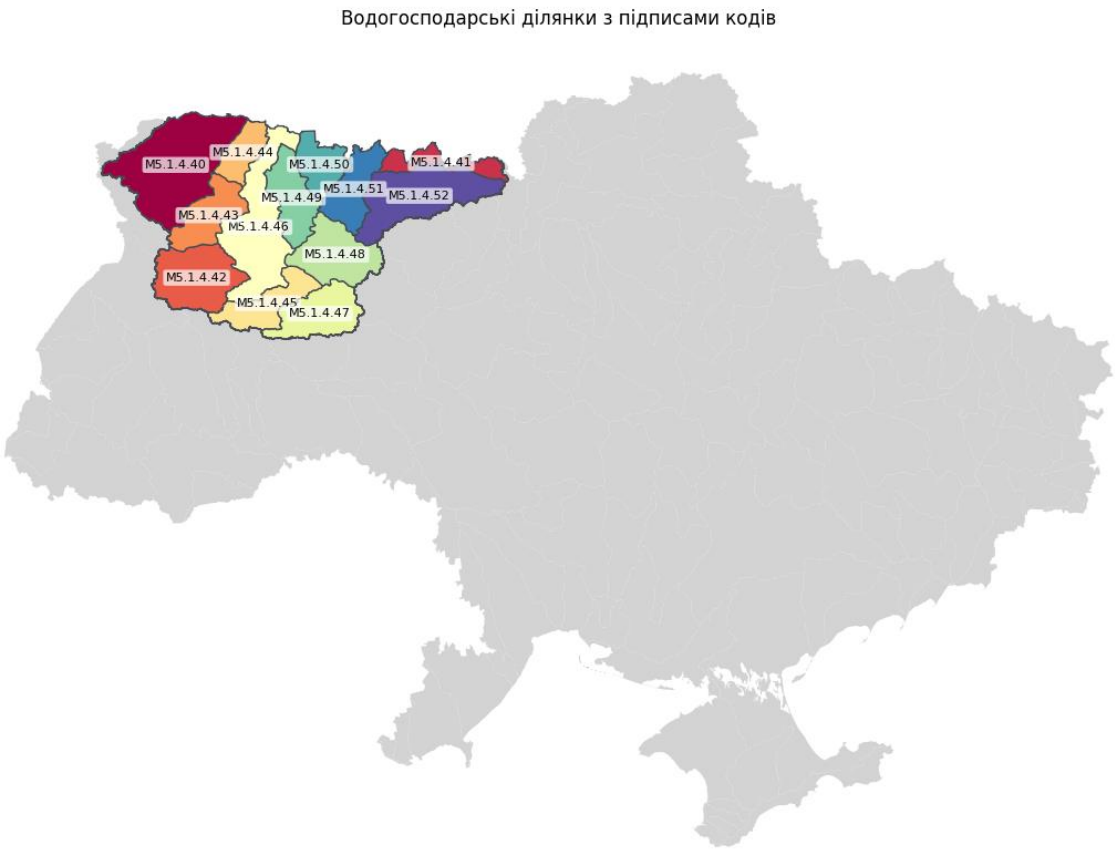


Рисунок 1.1 – Водогосподарські ділянки суубасейну Прип'яті

Дніпро – одна з найбільших річок Європи. Його довжина – 2 201 км (в межах України 1121 км), загальна площа басейну – 504 тис. км2 . Басейн річки Дніпро є транскордонним: 20% його площі знаходиться в Російській Федерації, 23% – Республіці Білорусь та 57% – у межах України. За площею цей басейн охоплює майже половину території України (48%). Район басейну Дніпра охоплює територію 19 областей України та повністю розташований у межах 6 областей

України – Житомирської, Чернігівської, Полтавської, Дніпропетровської, Рівненської та Сумської (рис. 1.2)

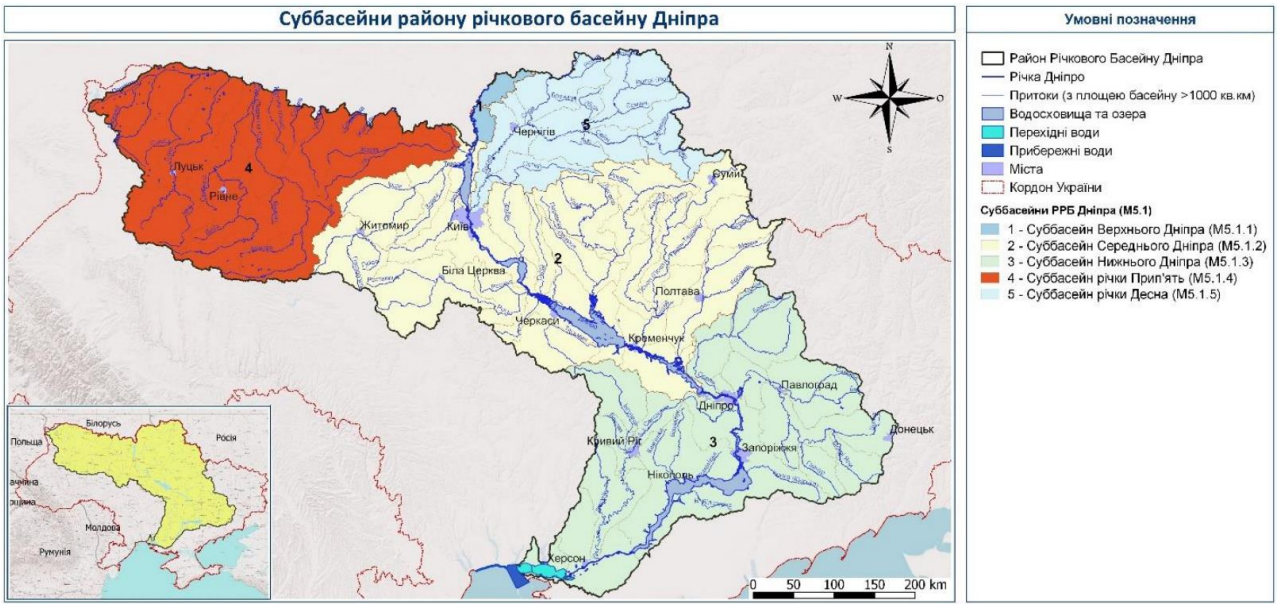


Рисунок 1.2 – Суббасейни району річкового басейну Дніпра [2, рис. 1]

Враховуючи значні розміри басейну Дніпра, управління басейном здійснюється за виділеними суббасейнами. Так, у межах району басейну річки Дніпро виділено 5 суббасейнів: Верхнього, Середнього та Нижнього Дніпра, а також Прип’яті та Десни (рис 1.3). Загальна водозбірна площа суббасейну Прип’яті складає 114 300 км2 , а площа в межах України – 68 400 км2 . Прип'ять – найбільша за площею басейну, довжиною і водністю права притока Дніпра. Довжина Прип'яті становить 775 км, в тому числі 254 км в межах України.

Клімат на території суббасейну – помірно-континентальний з теплим і вологим літом та достатньо м'якою зимою. Весна затяжна та нестійка, з частою змінною холодних та теплих періодів, літо тепле та дощове. Річна сума опадів по території суббасейну змінюється від 550 до 600 мм. Водний режим характеризується тривалою весняною повінню, короткочасною літньою меженню, що порушується дощовими паводками та майже щорічними осінніми підняттями рівня води. Вода підіймається на 1-4 м, на ділянках із звуженою заплавою – на 7 м. На весну припадає 65 % річного стоку. Більша частина суббасейну розташована в

межах Поліської низовини, південно-західна частина суббасейну знаходиться на Волинській височині. Рельєф представлений переважно плоскими та похилохвилястими низинами та рівнинами. Широко поширені денудаційні форми рельєфу, які утворенні на кристалічних породах.

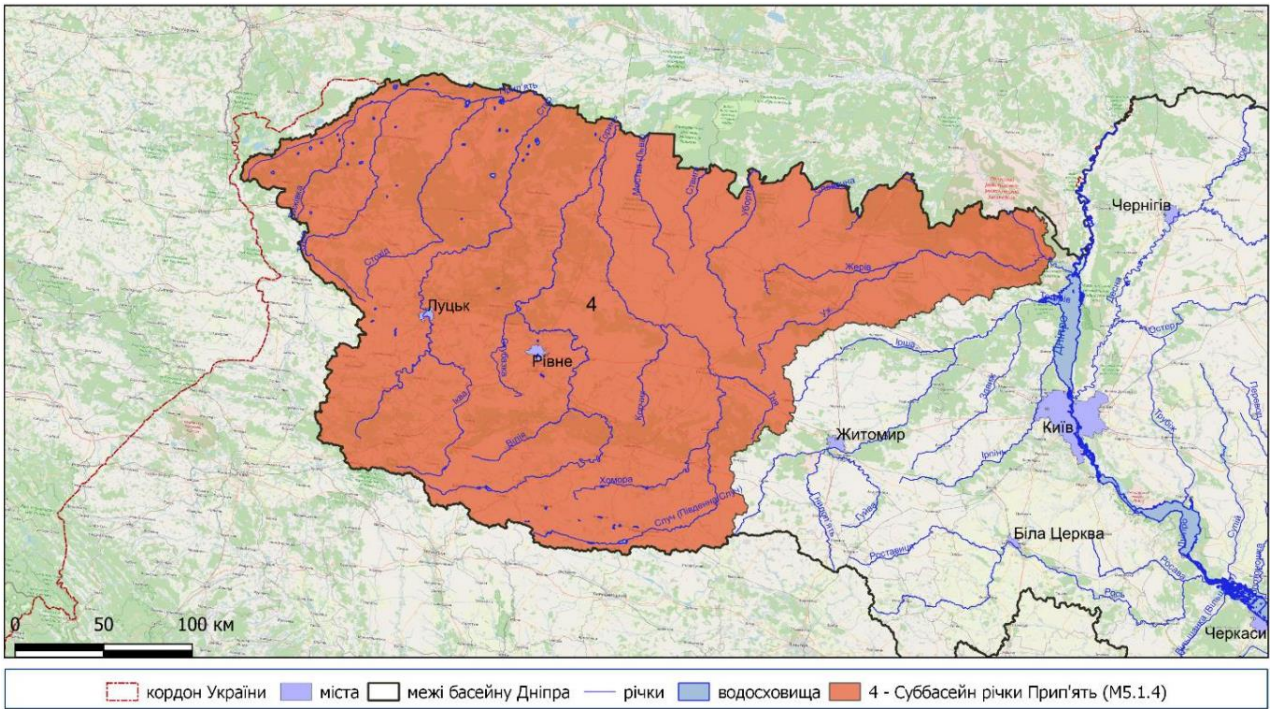


Рисунок 1.3 – Суббасейн Прип’яті [2]

Територія суббасейну річки Прип’ять вирізняється складною гідрогеологічною структурою, оскільки охоплює три гідрогеологічні регіони: Волинсько-Подільський артезіанський басейн, гідрогеологічну область Українського щита та Дніпровсько-Донецький артезіанський басейн. Це зумовлює відмінності у формуванні підземних вод – сприятливі умови характерні для артезіанських басейнів, тоді як у межах Українського щита вони менш вигідні для накопичення та використання підземних вод.

Основними об’єктами спостереження й оцінки стану підземних вод виступають масиви підземних вод (МПзВ), для яких формулюються відповідні екологічні цілі. В межах суббасейну, залежно від геологічних і гідрогеологічних особливостей, виділяються 4 масиви безнапірного типу та 8 – напірного.

Безнапірні масиви пов’язані з четвертинними відкладами, тоді як напірні залягають на різних глибинах у породах різного віку – від найновіших до архейських кристалічних утворень. Останні добре захищені від поверхневих джерел забруднення завдяки наявності товстого шару водотривких порід.

Обсяги підземних водних ресурсів значною мірою залежать від належності території до певного гідрогеологічного регіону. Найбільш водозабезпеченими є райони в межах артезіанських басейнів, тоді як найменші запаси зафіксовано в зоні Українського щита. У середньому, рівень використання підземних водних ресурсів у межах усього суббасейну становить близько 5% від розрахункових прогнозних запасів.

Під час дослідження антропогенного впливу та рівня навантаження на басейн Дніпра та його суббасейни було окреслено ключові водно-екологічні проблеми разом із чинниками, що їх зумовлюють:

1. органічне забруднення, спричинене неефективною або повною відсутністю очищення стічних вод;
2. надлишкове надходження біогенних речовин, джерелами яких є як недостатньо очищені стоки, так і змив добрив з аграрних територій;
3. наявність у водному середовищі небезпечних забруднювачів, що надходять зі стоками промисловості та житлово-комунального господарства, зокрема пестициди та інші агрохімікати, а також внаслідок просочування з полігонів і аварійних ситуацій;
4. гідроморфологічні трансформації, пов’язані з впровадженням заходів протипаводкового захисту, будівництвом гідроелектростанцій, створенням водосховищ і ставків, а також спрямленням річкових русел.

Окрім перелічених, до важливих екологічних загроз також належать забруднення побутовим сміттям (зокрема пластиковими відходами) і зміни клімату, що проявляються як у вигляді частіших паводків, так і тривалих посух.

Як уже згадувалося раніше, майже половина виділених масивів поверхневих вод (МПВ), а саме 516 одиниць, класифікуються як істотно змінені. Серед них:

– 119 масивів підпадають під категорію зарегульованих через наявність водосховищ і ставків;
– 93 масиви зазнали одночасно спрямлення русел та зарегульованості;
– 304 масиви мають змінене русло внаслідок спрямлення (див. рис. 4.1).

Найбільш значні гідроморфологічні трансформації спостерігаються в басейнах окремих річок суббасейну:

- Турія: істотні зміни фіксуються у 64% МПВ (25 з 39), з яких 23 змінені через спрямлення, а 2 — через зарегулювання;
- Стир: 71% масивів (106 із 149) визнано істотно зміненими, серед них 66 — внаслідок спрямлення, 13 — через зарегульованість, і ще 27 — у результаті поєднання обох впливів;
- Горинь: істотні зміни виявлено у 191 із 486 масивів (41%), включаючи 77 масивів зі зміненим руслом, 78 — із зарегулюванням і 36 — із комбінацією цих факторів.

Загалом, із 418 річок, розташованих у межах суббасейну, лише 104 (близько 25%) зберегли свою природну гідроморфологічну структуру без втручань чи змін.

Рисунок 1.4 – Категорії істотно змінених масивів поверхневих вод

Ознайомившись із різними джерелами інформації, можна виділити низку ключових екологічних викликів, характерних для басейну річки Прип’ять (рис. 5.1):

- Повеневі явища та паводки, що охоплюють великі площі населених пунктів і сільськогосподарських угідь у рівнинній місцевості. Такі явища спричиняють змив у річкову систему значної кількості агрохімікатів — добрив і пестицидів;
- Радіаційне забруднення, яке залишилося у спадок після аварії на Чорнобильській АЕС. Особливо високий рівень вмісту радіонуклідів спостерігається в північних і північно-східних частинах басейну – до 2 кюрі/км², особливо на низинних торфовищах, що призвело до включення цих земель до зони з підвищеним радіоактивним фоном;
- Процеси замулення, пов’язані з активною ерозією ґрунтів у водозбірній площі;
- Хімічне забруднення водотоків унаслідок скидів стічних вод різного походження;
- Гідроморфологічні зміни – зарегулювання водотоків, створення водосховищ і спрямлення русел, що змінює природний характер річкових систем;
- Зниження здатності водойм до самоочищення. Якщо раніше для Полісся були характерні природні процеси самоочищення, то нині ці механізми або суттєво ослаблені, або повністю порушені, що сприяє накопиченню забруднень;
- Зменшення біорізноманіття – як флори, так і фауни, що свідчить про деградацію природних екосистем;
- Меліоративна діяльність, яка змінює гідрологічний режим територій і може призводити до втрати природних водно-болотних екосистем.

Ці екологічні загрози є комплексними й взаємопов’язаними, тому їх подолання потребує системного підходу до управління водними ресурсами та впровадження природоорієнтованих заходів.

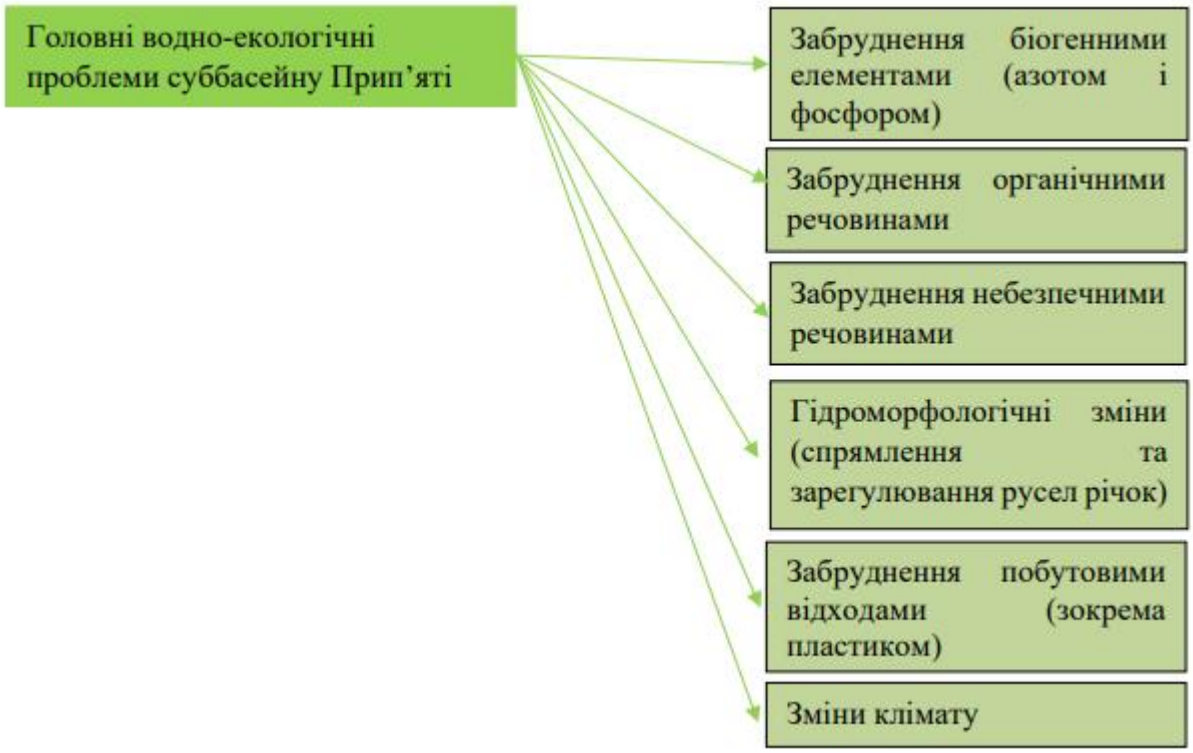


Рисунок 1.5 – Головні водно-екологічні проблеми суббасейну Прип'яті

Система розглядається як просторово розподілена динамічна система з великою кількістю взаємопов'язаних елементів, характеризується складною внутрішньою структурою, наявністю ієрархії (від локальних ділянок до загального басейнового рівня), а також сильними міжділянковими зв'язками – зміна водного режиму на одній ділянці може суттєво вплинути на наступні. У системі діють як керовані елементи (ліміти на забір і скидання води), так і некеровані (опади, природне випаровування), що обумовлює складність процесу управління.

Прийняття рішень у такій системі вимагає врахування сезонної мінливості факторів, багаторічної гідрологічної інформації, просторової взаємозалежності ділянок та допустимих технічних і екологічних обмежень. Основне завдання – оптимальне регулювання керованих впливів метою зменшення дефіциту води, особливо у критичних періодах (літо, осінь) та на вразливих ділянках. Таким чином, водогосподарська система досліджуваного суббасейну є типовим прикладом складного об'єкта дослідження в системному аналізі, де потрібна

інтеграція математичного моделювання, аналізу даних та інформаційних технологій для підтримки прийняття рішень.

1.2 Вибір мови програмування та середовища розробки

У процесі розробки аналітичної системи для дослідження водного балансу суббасейну Прип’яті постало питання вибору мови програмування, яка б забезпечувала ефективну обробку великих обсягів даних, гнучкість побудови моделей прогнозування та простоту візуалізації результатів. Вибір мови програмування є важливим етапом, оскільки він впливає на зручність реалізації проєкту, масштабованість, підтримку сторонніх бібліотек та можливості інтеграції з іншими інструментами.

Основні критерії, що були враховані при виборі мови програмування:

- широка підтримка бібліотек для аналізу даних і машинного навчання;
- активна спільнота розробників і наявність документації;
- простота синтаксису і швидкість розробки;
- можливість обробки часових рядів і побудови статистичних моделей;
- інтеграція з візуалізаційними інструментами (графіки, дашборди);
- хороша підтримка як локального, так і хмарного середовища виконання.

Було проаналізовано кілька найбільш популярних мов, що використовуються в галузі аналізу даних та наукових досліджень. Результат аналізу в розрізі сильних та слабких сторін приведено в таблиці 1.1.

Таблиця 1.1 – Найбільш популярні мови для аналізу даних

Мова програмування	Сильні сторони	Слабкі сторони
Python	Простота, гнучкість, величезна кількість бібліотек (pandas, scikit-learn, xgboost, prophet), активна спільнота	Повільніший за компільовані мови при великих обсягах обчислень
R	Потужні статистичні пакети, спеціалізація на аналізі даних	Більш складний синтаксис, обмеженість у загальному програмуванні
MATLAB	Зручний для чисельних обчислень та інженерних задач	Платна ліцензія, менш популярна у сфері машинного навчання
Java / C++	Висока продуктивність, контроль над пам’яттю	Складніші в розробці, менше аналітичних бібліотек
Julia	Сучасна, оптимізована для чисельних обчислень	Менш розвинена екосистема, порівняно невелика спільнота

Перевагами мови програмування Python для реалізації програми аналізу та прогнозування даних є:

- Масова підтримка бібліотек і фреймворків для статистики, обробки часових рядів, машинного навчання та глибокого навчання: pandas, numpy, statsmodels, scikit-learn, xgboost, prophet, keras, tensorflow тощо.
- Високий рівень читабельності коду, що дозволяє зосередитися на логіці дослідження, а не на синтаксичних деталях.

- Інтерактивність та візуалізація — у поєднанні з Jupyter Notebook або Google Colab Python дозволяє інтерактивно запускати код, будувати графіки (matplotlib, seaborn, plotly) та коментувати результати.
- Кросплатформеність і зручність розгортання як локально, так і в хмарних середовищах.
- Наявність великої спільноти та безлічі навчальних ресурсів, що полегшує вирішення технічних труднощів.

Варто відзначити, що Python також чудово інтегрується з іншими інструментами (наприклад, Excel, SQL, API-сервісами) і дозволяє будувати масштабовані аналітичні рішення в реальному середовищі.

Для реалізації системного аналізу водогосподарських балансів у межах суббасейну річки Прип’ять було використано мову програмування Python [3]. Ця мова користується широким попитом серед наукових дослідників, інженерів і фахівців з аналітики завдяки простому синтаксису та багатому функціоналу. Python широко застосовується у таких напрямках, як обробка даних, математичне моделювання, геоінформаційні технології, екологічні дослідження, природокористування і побудова прогнозних систем. Її відкритість, активна підтримка професійної спільноти та наявність численних бібліотек роблять її ефективним інструментом для міждисциплінарних досліджень, у тому числі й у сфері водного менеджменту.

У ході аналітичної роботи застосовувалися сучасні інструменти Python для обробки гідрологічної інформації, просторового аналізу, побудови візуалізацій і створення прогнозних моделей. Для роботи з табличними наборами даних основним інструментом стала бібліотека pandas [4], яка забезпечує зручні можливості для фільтрації, трансформацій, агрегування, заповнення пропусків і об’єднання інформації з різних джерел. Побудову графіків сезонної динаміки та аналітику показників виконували з використанням matplotlib [5] і seaborn [6], які дозволили візуалізувати залежності, тренди та кореляції.

Просторовий аспект аналізу реалізовано за допомогою бібліотеки geopandas [6], яка дала змогу створювати мапи із нанесенням меж водогосподарських ділянок

суббасейну Прип’яті, прив’язувати до них значення водного балансу та будувати тематичні картограми. Це дозволило не лише працювати з числовими даними, а й аналізувати географічне розташування критичних зон, визначати зони ризику та оцінювати гідрологічну ситуацію на рівні всього басейну.

Для прогнозування рівня води були застосовані як класичні методи статистики – Prophet та ARIMA_auto, так і сучасні алгоритми машинного навчання, зокрема: Support Vector Machines (SVM), Linear SVR, XGBoost Regressor, Linear Regression, Random Forest Regressor, Bagging Regressor, KNeighbors Regressor та MLP Regressor [8–18]. Усі моделі були натреновані на актуальних гідрологічних даних. Для оцінювання точності прогнозів використовувалися метрики RMSE (середньоквадратична похибка) та MAPE (середня абсолютна відносна похибка) [19], що дало змогу визначити найефективніші моделі для короткотермінового прогнозування.

Упроцесі виконання завдань, пов’язаних із аналізом гідрологічних даних і побудовою моделей прогнозування, критично важливим є вибір оптимального середовища розробки. Правильно обране середовище дозволяє ефективно працювати з великими обсягами даних, забезпечує зручність при візуалізації результатів та інтеграцію сучасних бібліотек для машинного навчання.

Під час вибору програмного середовища було враховано низку вимог, зокрема:

- підтримка мови Python, яка є провідною у сфері аналітики та наукових обчислень;
- інтеграція з бібліотеками для роботи з даними (pandas, numpy, matplotlib, seaborn, scikit-learn, xgboost, statsmodels, prophet тощо);
- зручність інтерактивної роботи, що важливо для дослідження часових рядів, перевірки гіпотез та швидкого тестування моделей;
- можливість роботи з візуалізацією, включаючи побудову графіків та інтерактивних діаграм;
- доступність, кросплатформеність та підтримка хмарних сервісів;
- підтримка командної роботи та реплікації досліджень.

На основі вказаних критеріїв було проаналізовано низку інструментів та середовищ, які часто використовуються в науковій спільноті та в прикладній роботі з даними.

Найбільш поширеними середовищами для аналізу даних на Python є такі середосища та платформи (табл. 1.2).

Таблиця 1.2 – Найбільш поширені середовища для аналізу даних на Python

Середовище	Тип	Основні переваги	Недоліки
Jupyter Notebook	Веб/локальне	Інтерактивність, візуалізація, підтримка Markdown, доступ до ядра Python в реальному часі	Менш зручне для великих проєктів
Google Colab	Хмарне	Безкоштовний доступ до GPU, зручний обмін, інтеграція з Google Drive	Потребує інтернету, обмеження ресурсів
VS Code + Jupyter	Локальне	Потужний редактор, інтеграція з git, підтримка розширень	Необхідність налаштування середовища
PyCharm Professional	Локальне	Потужне IDE, зручна робота з великими проєктами, вбудований відлагоджувач	Платна ліцензія, менш зручна інтерактивність
Anaconda Navigator	Локальне	Інтуїтивна робота з середовищами, попередньо встановлені бібліотеки	Великий обсяг дискового простору

У цій роботі для аналізу даних та моделювання було обрано Google Colab для аналізу даних та як основне середовище, а також Kaggle Notebook.

Це зумовлено такими перевагами:

- висока доступність: не потребує встановлення програмного забезпечення на локальному комп’ютері, що спрощує початок роботи;
- інтеграція з зовнішніми джерелами даних: дозволяє зберігати результати, графіки, моделі у хмарі;
- нтерактивність: зручна робота з графіками, таблицями, текстовими описами прямо в середовищі;
- підтримка GPU та TPU: за потреби можна прискорити обчислення безкоштовно;
- підтримка бібліотек для обробки часових рядів: можливість імпортувати та використовувати сучасні інструменти (prophet, xgboost, scikit-learn, tsfresh тощо) без додаткових налаштувань.

Приклад використання Google Colab представлений на рисунку X.1:

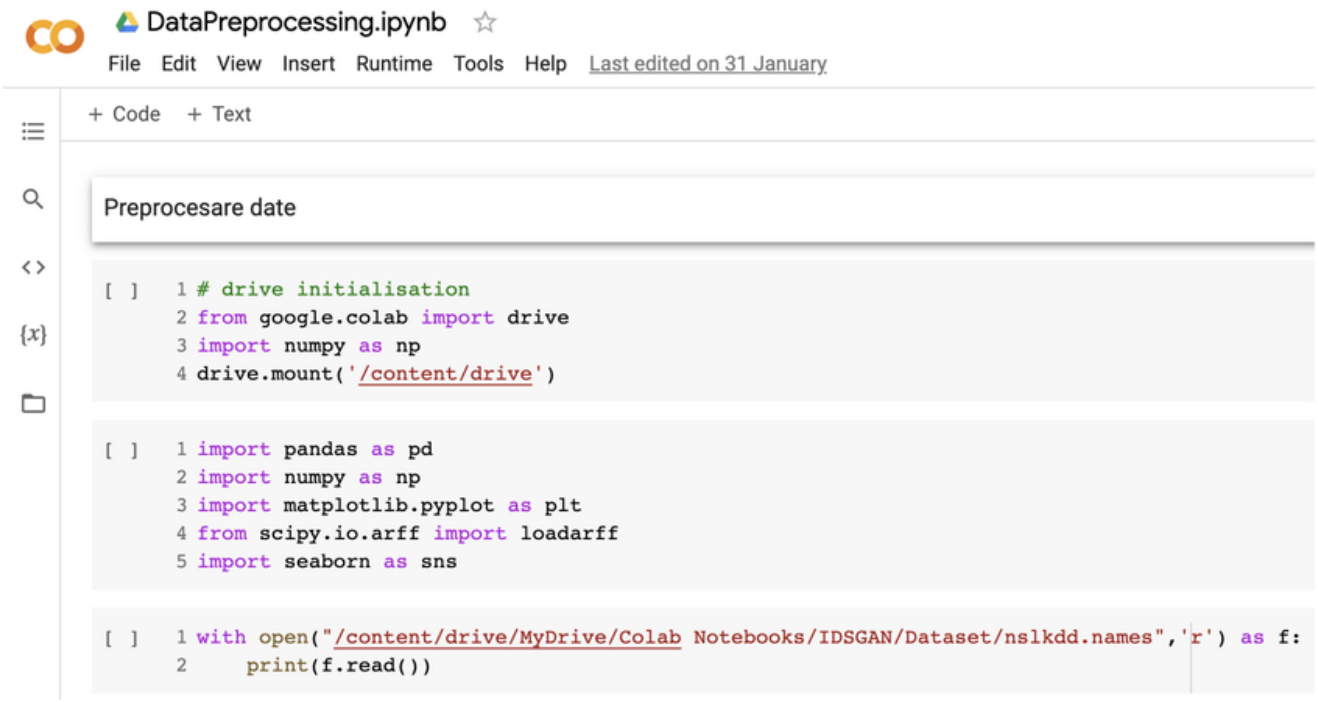


Рисунок 1.5 – Інтерфейс середовища Google Colab

В якості середовища розробки обрано платформу Kaggle Notebook [20] (рис. 1.6), яка об'єднує функції Jupyter Notebook, інтерфейс для програмування на Python і доступ до відкритих датасетів. Цей інструмент дозволяє запускати проєкти у хмарі без потреби локального встановлення ПЗ, що особливо зручно для спільної роботи,

публікацій та гарантування відтворюваності результатів. Завдяки вбудованим бібліотекам, підтримці GPU-прискорення та інтерактивному середовищу, Kaggle підходить навіть для складних аналітичних задач.

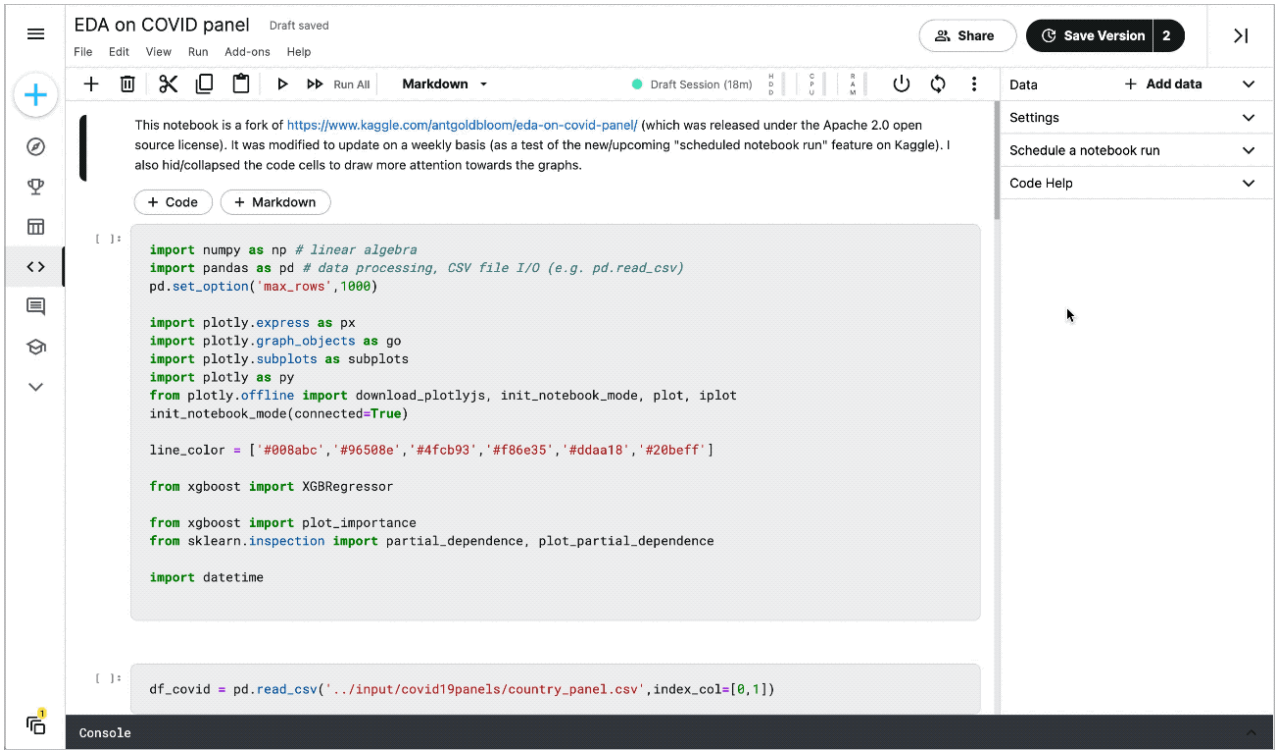


Рисунок 1.6 – Інтерфейс середовища Kaggle

Комплексне використання зазначених інструментів дозволило реалізувати системний підхід до аналізу водного балансу ділянок суббасейну Прип’яті. Усі ключові етапи – від попередньої обробки даних до їх візуалізації, просторового представлення, прогнозування і оцінювання моделей – були охоплені в межах одного дослідження. Такий підхід не лише виявив динамічні закономірності у зміні водогосподарських параметрів, а й створив основу для подальшого масштабування та включення розробленого інструментарію в системи підтримки прийняття рішень у сфері водокористування.

1.3 Висновки

У першому розділі проведено характеристику водогосподарської системи окремих ділянок суббасейну річки Прип’ять, яку розглянуто як складну динамічну

структуру з вираженою просторовою та сезонною мінливістю. Описано ключові елементи водного балансу, зокрема: поверхневий стік, бічний приплив, водозабір, повернені води, екологічний стік тощо. Окрему увагу приділено чинникам, що спричиняють виникнення водного дефіциту.

У цьому ж розділі обґрунтовано доцільність застосування мови програмування Python для реалізації основних завдань аналізу, а також вибір хмарного середовища Kaggle Notebooks, яке було використано як технічну платформу для виконання просторової аналітики, побудови моделей та візуалізації результатів.

2. ПІДГОТОВКА ДАНИХ, РОЗВІДУВАЛЬНИЙ АНАЛІЗ ТА ІНЖЕНЕРІЯ ОЗНАК

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

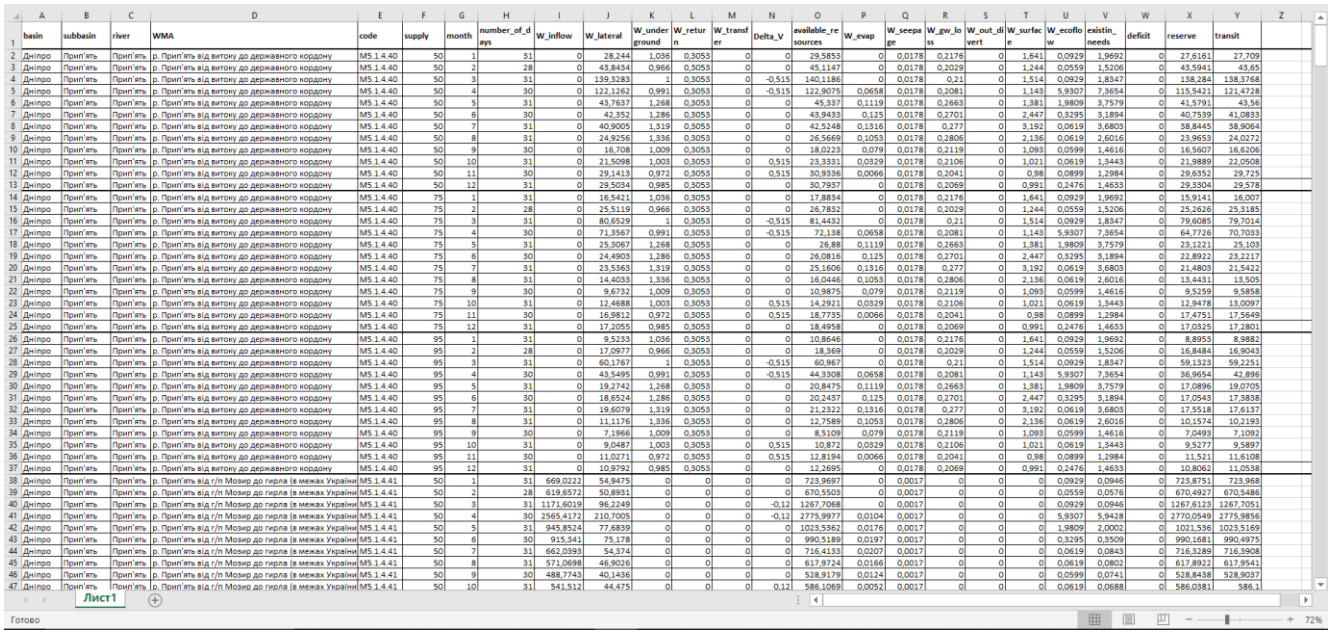
Для виконання системного аналізу водогосподарських балансів суббасейну Прип'яті були використані офіційні дані, опубліковані Державним агентством водних ресурсів України у вигляді щомісячних водогосподарських балансів у форматі PDF [21]. З метою подальшої обробки ці дані були перетворені у формат XLSX, а після цього збережені у вигляді CSV-файлів.

На етапі попередньої обробки виконано стандартизацію числових значень, замінивши роздільник десяткових частин з коми на крапку. Назви стовпців були змінені з довгих українських варіантів на скорочений англомовний формат, що сприяло зручнішій обробці в програмних середовищах та відповідало вимогам до даних для машинного навчання. Також була перевірена коректність структури таблиці, наявність даних для всіх місяців та ідентифікація водогосподарських ділянок (ВГД). Нижче на рисунках 2.1 – 2.3 представлено вигляд даних до та після обробки.

Таблиця 1. Водогосподарський баланс для
р. Прип'ять від витоку до державного кордону (код М5.1.4.40)
(назва водогосподарської ділянки)
при 50% забезпеченості стоку

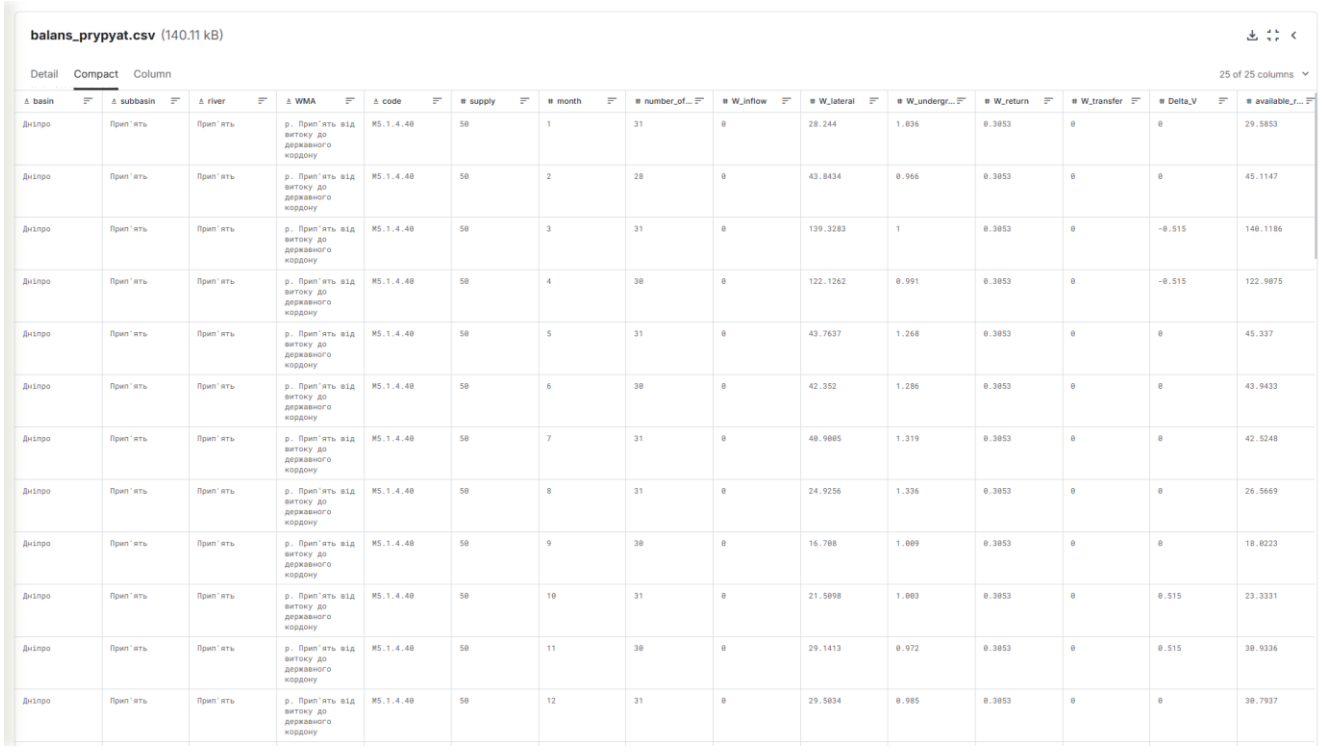
Складові водогосподарського балансу	Розрахункові інтервали часу водогосподарського року (місяці/дні)											
	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII
	31	28	31	30	31	30	31	31	30	31	30	31
I. Прибуткова частина												
1. Об'єм стоку, що надходить за розрахунковий період з розташованих вище ВГД, Wвх, млн куб. м	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
2. Об'єм стоку, що формується на розрахунковій ВГД (бічний приплив), Wбіч, млн куб. м	28,2440	43,8434	139,3283	122,1262	43,7637	42,3520	40,9005	24,9256	16,7080	21,5098	29,1413	29,5034
3. Об'єм водозабору із підземних водних об'єктів, Wпзв, млн куб. м	1,0360	0,9660	1,0000	0,9910	1,2680	1,2860	1,3190	1,3360	1,0090	1,0030	0,9720	0,9850
4. Об'єм зворотних вод на розрахунковій ВГД, Wзв, млн куб. м	0,3053	0,3053	0,3053	0,3053	0,3053	0,3053	0,3053	0,3053	0,3053	0,3053	0,3053	0,3053
5. Дотаційний об'єм води на ВГД (зовнішні та внутрішньобасейнові перекидання), Wдот, млн куб. м	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
6. Спрацювання (+), наповнення (-) ставків та водосховищ, ±ΔV, млн куб. м	0,0000	0,0000	-0,5150	-0,5150	0,0000	0,0000	0,0000	0,0000	0,0000	0,5150	0,5150	0,0000
7. Усього по прибутковій частині (наявні ресурси), млн куб. м	29,5853	45,1147	140,1186	122,9075	45,3370	43,9433	42,5248	26,5669	18,0223	23,3331	30,9336	30,7937

Рисунок 2.1 – Приклад вихідних даних балансу у форматі PDF



basin	subbasin	river	WMA	code	supply	month	number_of_days	W_inflow	W_lateral	W_underground	W_return	W_transfer	Delta_V	available_resources	W_evap	W_seepage	W_gw_loss	W_out_divert	W_surface	W_ecoflow	existin_needs	deficit	reserve	transit
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	1	31	0	28,244	1,036	0,3053	0	0	29,5853	0	0,0178	0,2176	0	1,641	0,0929	1,9692	0	27,6161	27,709
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	2	28	0	43,8434	0,966	0,3053	0	0	45,1147	0	0,0178	0,2029	0	1,244	0,0559	1,5206	0	43,5941	43,65
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	3	31	0	139,3283	1	0,3053	0	-0,515	140,1186	0	0,0178	0,21	0	1,514	0,0929	1,8347	0	138,284	138,3768
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	4	30	0	122,1262	0,991	0,3053	0	-0,515	122,9075	0,0658	0,0178	0,2081	0	1,143	5,9307	7,3654	0	115,5421	121,4728
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	5	31	0	43,7637	1,268	0,3053	0	0	45,337	0,1119	0,0178	0,2663	0	1,381	1,9809	3,7579	0	41,5791	43,56
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	6	30	0	42,352	1,286	0,3053	0	0	43,9433	0,125	0,0178	0,2701	0	2,447	0,3295	3,1894	0	40,7539	41,0833
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	7	31	0	40,9005	1,319	0,3053	0	0	42,5248	0,1316	0,0178	0,277	0	3,192	0,0619	3,6803	0	38,8445	38,9064
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	8	31	0	24,9256	1,336	0,3053	0	0	26,5669	0,1053	0,0178	0,2806	0	2,136	0,0619	2,6016	0	23,9653	24,0272
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	9	30	0	16,708	1,009	0,3053	0	0	18,0223	0,079	0,0178	0,2119	0	1,093	0,0599	1,4616	0	16,5607	16,6206
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	10	31	0	21,5098	1,003	0,3053	0	0,515	23,3331	0,0329	0,0178	0,2106	0	1,021	0,0619	1,3443	0	21,9889	22,0508
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	11	30	0	29,1413	0,972	0,3053	0	0,515	30,9336	0,0066	0,0178	0,2041	0	0,98	0,0899	1,2984	0	29,6352	29,725
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	12	31	0	29,5034	0,985	0,3053	0	0	30,7937	0	0,0178	0,2069	0	0,991	0,2476	1,4633	0	29,3304	29,578
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	75	1	31	0	16,5421	1,036	0,3053	0	0	17,8834	0	0,0178	0,2176	0	1,641	0,0929	1,9692	0	15,9141	16,007
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	75	2	28	0	25,5119	0,966	0,3053	0	0	26,7832	0	0,0178	0,2029	0	1,244	0,0559	1,5206	0	25,2626	25,3185
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	75	3	31	0	80,6529	1	0,3053	0	-0,515	81,4432	0	0,0178	0,21	0	1,514	0,0929	1,8347	0	79,6085	79,7014
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	75	4	30	0	71,3567	0,991	0,3053	0	-0,515	72,138	0,0658	0,0178	0,2081	0	1,143	5,9307	7,3654	0	64,7726	70,7033
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	75	5	31	0	25,3067	1,268	0,3053	0	0	26,88	0,1119	0,0178	0,2663	0	1,381	1,9809	3,7579	0	23,1221	25,103
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	75	6	30	0	24,4903	1,286	0,3053	0	0	26,0816	0,125	0,0178	0,2701	0	2,447	0,3295	3,1894	0	22,8922	23,2217
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	75	7	31	0	23,5363	1,319	0,3053	0	0	25,1606	0,1316	0,0178	0,277	0	3,192	0,0619	3,6803	0	21,4803	21,5422
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	75	8	31	0	14,4033	1,336	0,3053	0	0	16,0446	0,1053	0,0178	0,2806	0	2,136	0,0619	2,6016	0	13,4431	13,505
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	75	9	30	0	9,6732	1,009	0,3053	0	0	10,9875	0,079	0,0178	0,2119	0	1,093	0,0599	1,4616	0	9,5259	9,5858
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	75	10	31	0	12,4688	1,003	0,3053	0	0,515	14,2921	0,0329	0,0178	0,2106	0	1,021	0,0619	1,3443	0	12,9478	13,0097
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	75	11	30	0	16,9812	0,972	0,3053	0	0,515	18,7735	0,0066	0,0178	0,2041	0	0,98	0,0899	1,2984	0	17,4751	17,5649
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	75	12	31	0	17,2055	0,985	0,3053	0	0	18,4958	0	0,0178	0,2069	0	0,991	0,2476	1,4633	0	17,0325	17,2801
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	95	1	31	0	9,5233	1,036	0,3053	0	0	10,8646	0	0,0178	0,2176	0	1,641	0,0929	1,9692	0	8,8953	8,9882
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	95	2	28	0	17,0977	0,966	0,3053	0	0	18,369	0	0,0178	0,2029	0	1,244	0,0559	1,5206	0	16,8484	16,9043
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	95	3	31	0	60,1767	1	0,3053	0	-0,515	60,967	0	0,0178	0,21	0	1,514	0,0929	1,8347	0	59,1323	59,2251
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	95	4	30	0	43,5495	0,991	0,3053	0	-0,515	44,3308	0,0658	0,0178	0,2081	0	1,143	5,9307	7,3654	0	36,9654	42,896
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	95	5	31	0	19,2742	1,268	0,3053	0	0	20,8475	0,1119	0,0178	0,2663	0	1,381	1,9809	3,7579	0	17,0896	19,0705
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	95	6	30	0	18,6524	1,286	0,3053	0	0	20,2437	0,125	0,0178	0,2701	0	2,447	0,3295	3,1894	0	17,0543	17,3838
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	95	7	31	0	19,6079	1,319	0,3053	0	0	21,2322	0,1316	0,0178	0,277	0	3,192	0,0619	3,6803	0	17,5518	17,6137
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	95	8	31	0	11,1176	1,336	0,3053	0	0	12,7589	0,1053	0,0178	0,2806	0	2,136	0,0619	2,6016	0	10,1574	10,2193
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	95	9	30	0	7,1966	1,009	0,3053	0	0	8,5109	0,079	0,0178	0,2119	0	1,093	0,0599	1,4616	0	7,0493	7,1092
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	95	10	31	0	9,0487	1,003	0,3053	0	0,515	10,872	0,0329	0,0178	0,2106	0	1,021	0,0619	1,3443	0	9,5277	9,5897
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	95	11	30	0	11,0271	0,972	0,3053	0	0,515	12,8194	0,0066	0,0178	0,2041	0	0,98	0,0899	1,2984	0	11,521	11,6108
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	95	12	31	0	10,9792	0,985	0,3053	0	0	12,2695	0	0,0178	0,2069	0	0,991	0,2476	1,4633	0	10,8062	11,0538
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від г/п Мозир до гирла (в межах України	M5.1.4.41	50	1	31	669,0222	54,9475	0	0	0	0	723,9697	0	0,0017	0	0	0	0,0929	0,0946	0	723,8751	723,968
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від г/п Мозир до гирла (в межах України	M5.1.4.41	50	2	28	619,6572	50,8931	0	0	0	0	670,5503	0	0,0017	0	0	0	0,0559	0,0576	0	670,4927	670,5486
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від г/п Мозир до гирла (в межах України	M5.1.4.41	50	3	31	1171,6019	96,2249	0	0	0	-0,12	1267,7068	0	0,0017	0	0	0	0,0929	0,0946	0	1267,6123	1267,7051
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від г/п Мозир до гирла (в межах України	M5.1.4.41	50	4	30	2565,4172	210,7005	0	0	0	-0,12	2775,9977	0,0104	0,0017	0	0	5,9307	5,9428	0	2770,0549	2775,9856
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від г/п Мозир до гирла (в межах України	M5.1.4.41	50	5	31	945,8524	77,6839	0	0	0	0	1023,5362	0,0176	0,0017	0	0	0	1,9809	2,0002	0	1021,536	1023,5169
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від г/п Мозир до гирла (в межах України	M5.1.4.41	50	6	30	915,341	75,178	0	0	0	0	990,5189	0,0197	0,0017	0	0	0	0,3295	0,3509	0	990,1681	990,4975
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від г/п Мозир до гирла (в межах України	M5.1.4.41	50	7	31	662,0393	54,374	0	0	0	0	716,4133	0,0207	0,0017	0	0	0	0,0619	0,0843	0	716,3289	716,3908
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від г/п Мозир до гирла (в межах України	M5.1.4.41	50	8	31	571,0698	46,9026	0	0	0	0	617,9724	0,0166	0,0017	0	0	0	0,0619	0,0802	0	617,8922	617,9541
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від г/п Мозир до гирла (в межах України	M5.1.4.41	50	9	30	488,7743	40,1436	0	0	0	0	528,9179	0,0124	0,0017	0	0	0	0,0599	0,0741	0	528,8438	528,9037
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від г/п Мозир до гирла (в межах України	M5.1.4.41	50	10	31	541,512	44,475	0	0	0	0,12	586,1069	0,0052	0,0017	0	0	0	0,0619	0,0688	0	586,0381	586,1

Рисунок 2.2 – Вигляд даних балансу після обробки у форматі XLSX



balans_prypyat.csv (140.11 kB)

Detail Compact Column — 25 of 25 columns

basin	subbasin	river	WMA	code	supply	month	number_of...	W_inflow	W_lateral	W_undergr...	W_return	W_transfer	Delta_V	available_r...
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	1	31	0	28.244	1.036	0.3053	0	0	29.5853
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	2	28	0	43.8434	0.966	0.3053	0	0	45.1147
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	3	31	0	139.3283	1	0.3053	0	-0.515	140.1186
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	4	30	0	122.1262	0.991	0.3053	0	-0.515	122.9075
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	5	31	0	43.7637	1.268	0.3053	0	0	45.337
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	6	30	0	42.352	1.286	0.3053	0	0	43.9433
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	7	31	0	40.9005	1.319	0.3053	0	0	42.5248
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	8	31	0	24.9256	1.336	0.3053	0	0	26.5669
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	9	30	0	16.708	1.009	0.3053	0	0	18.0223
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	10	31	0	21.5098	1.003	0.3053	0	0.515	23.3331
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	11	30	0	29.1413	0.972	0.3053	0	0.515	30.9336
Дніпро	Прип'ять	Прип'ять	р. Прип'ять від витоку до державного кордону	M5.1.4.40	50	12	31	0	29.5034	0.985	0.3053	0	0	30.7937

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 [22] по автоматичному гідропосту «Горинь» за період з 1946 по 2023 роки (рис. 2.4 – 2.5)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**Витрати води за даними гідропоста р. Горинь – м. Нетішин**												
2	Рік	Середня місячна витрата води, м^3/с											
3		1	2	3	4	5	6	7	8	9	10	11	12
4	1946	13,6					4,7	4,5	4,6	5,1	6,1	9,6	9,8
5	1947	3,6	3,8	87,6	22,7	11,2	8,6	6,6	4,5	9,5	9,8	19	19
6	1948	37,4	32,9	35,8	16,9	7,7	21,9	61,5	11,5	7,2	9	9,8	7,7
7	1949	7,6	25,3	34,5	24,1	8,9	7,5	23,2	17,3	9,1	7,7	9,3	12,5
8	1950	7,6	33,2	16	10,5	6,8	4,8	3,7	4,8	5	5,5	13,8	11,2
9	1951	8,2	6,6	40,9	17,1	14,9	6,5	5,8	5,7	6,1	7,4	7,4	9,1
10	1952	8,2	8,4	7,6	39,3	10,4	7,3	5,3	5,4	7,2	8,4	10,3	9,2
11	1953	11,5	24,5	61,5	13,7	9,6	8,6	5,6	5,9	6,9	6,8	6,8	6,3
12	1954	3,9	3,6	27,7	16,8	14,5	13,6	8,4	10	8,3	8,1	7,9	9,8
13	1955	6,3	7,9	32,5	20,3	8,5	11,7	16,4	23,3	9,3	9,5	10,1	16,2
14	1956	12,6	6,5	8,5	118,7	12,6	8	6,9	5,6	9,5	8,9	9,3	14,1
15	1957	8,9	17,8	15	8,2	7,4	6,7	4,3	4,8	6,1	5,7	5,1	6
16	1958	5,5	31,3	17	44	9,5	11,1	13,2	6,3	9,2	13,3	11	9,6
17	1959	13,8	8,6	22	11	7,6	6,1	4,1	8,6	5,7	5,9	12,8	6,5
18	1960	12,9	19,5	41,1	12,9	8,5	7,1	7,5	6,2	6,1	9,6	26,1	18,8
19	1961	10,9	15,8	14,7	8,8	6,3	5,1	4,3	4,3	4,9	5,2	5,8	5,3
20	1962	5,6	6,2	15,1	71,7	19,5	12,1	18,8	16,8	10,4	10,4	12,4	8,9
21	1963	6	8,7	36,9	52,5	9,3	7,4	6,1	6,2	6,1	6,9	7,1	4,8
22	1964	4,7	5,2	9,1	41	10	5,2	8,5	9,8	9,5	7,9	7,4	11,4
23	1965	4,1	7,6	30,5	32,9	13,6	23,6	20,8	12	8,9	9	9,7	13,1
24	1966	8,9	37,2	33,9	16,6	9,1	8,8	9,4	10,3	10	9,1	10,8	7,7
25	1967	6,6	8,8	59,5	18,7	13,8	10,5	8,4	6,9	7,7	7,7	7,7	8

Рисунок 2.4 – Вихідні дані витрат води за даними посту «Горинь»

Date	Average
1946-01-16 2023-12-16	3.6 121
1/16/1946	13.6
2/14/1946	
3/16/1946	
4/15/1946	
5/16/1946	
6/15/1946	4.7
7/16/1946	4.5
8/16/1946	4.6
9/15/1946	5.1
10/16/1946	6.1
11/15/1946	9.6
12/16/1946	9.8

Рисунок 2.5 – Датасет з витрат води за даними посту «Горинь»

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 [23]. 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 суббасейну Прип’яті

У процесі системного аналізу даних, реалізованого в середовищі Kaggle Notebooks, було побудовано прототип системи оцінювання гідрологічного стану водогосподарських ділянок суббасейну Прип’яті. 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 W_underground, W_return, reserve, 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). Дефіцити по досліджуваним ділянкам відсутні. Натомість резерв води (reserve) 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.

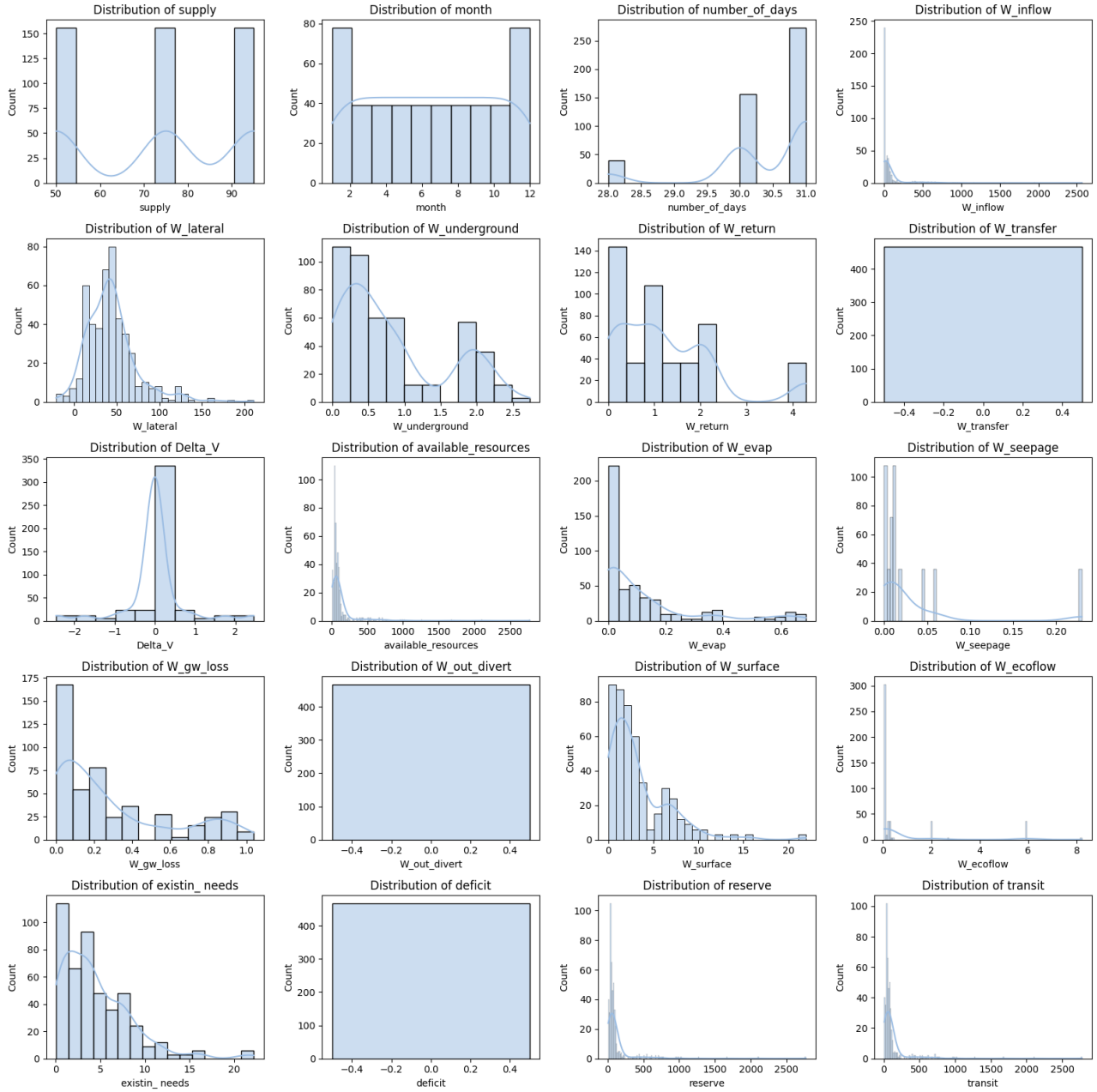


Рисунок 2.7 – Гістограми розподілу числових характеристик

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 [24]. 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.

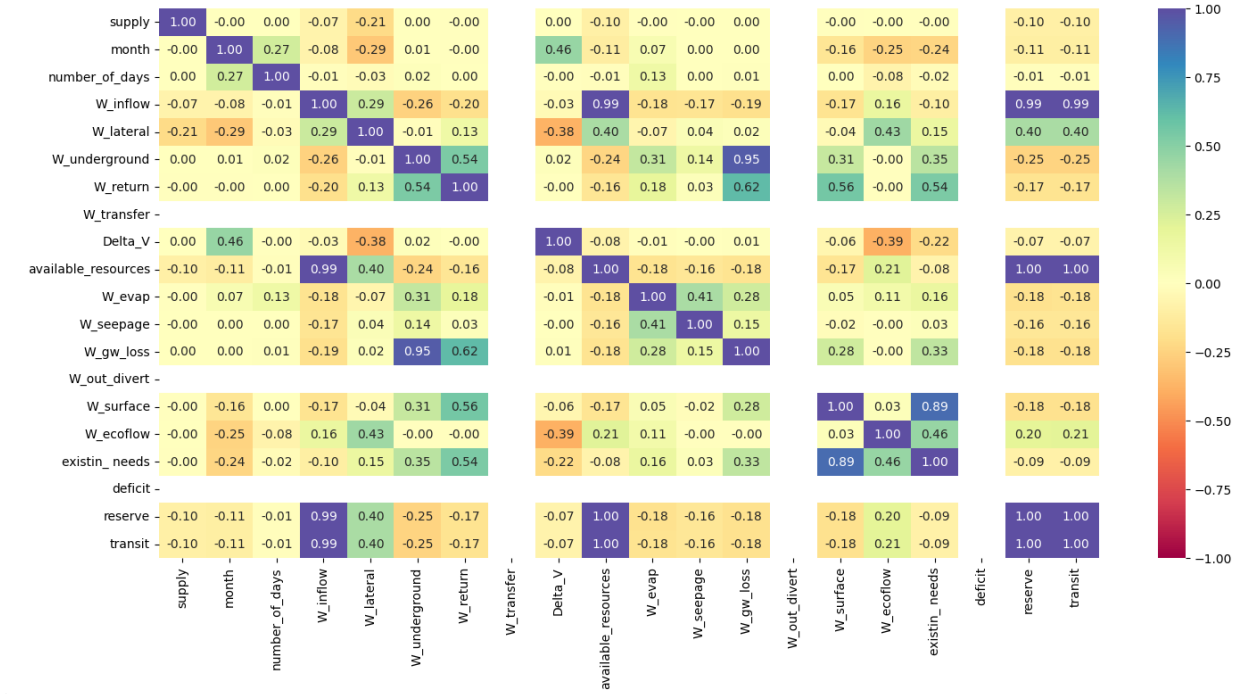


Рисунок 2.8 – Кореляційна матриця

Серед цікавих зв’язків слід відзначити сильні позитивні кореляції:

- W_inflow (Об’єм стоку, що надходить за розрахунковий період з розташованих вище ВГД) має дуже високу кореляцію з:
  - available_resources (Усього по прибутковій частині (наявні ресурси)) (0.99)
  - Reserve (Резерв водних ресурсів , transit (Транзит стоку на розташовану нижче ВГД з урахуванням мінімального екологічного стоку у замикаючому створі) (0.99)
- W_gw_loss (Зменшення стоку річки, викликане забором гідравлічно зв'язаних з нею підземних вод) сильно пов’язаний з:
  - W_underground (Об'єм водозабору із підземних водних об’єктів) (0.95)
  - W_return (Об'єм зворотних вод на розрахунковій ВГД) (0.62)
- W_surface (Забір поверхневих вод) корелює з:
  - W_out_divert (Перекидання частини стоку за межі розрахункової) (0.89)

– existin_needs (Усього по витратній частині (наявні потреби), (0.54)

Побудовано графіки динаміки водного резерву (рис. 2.9). 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.

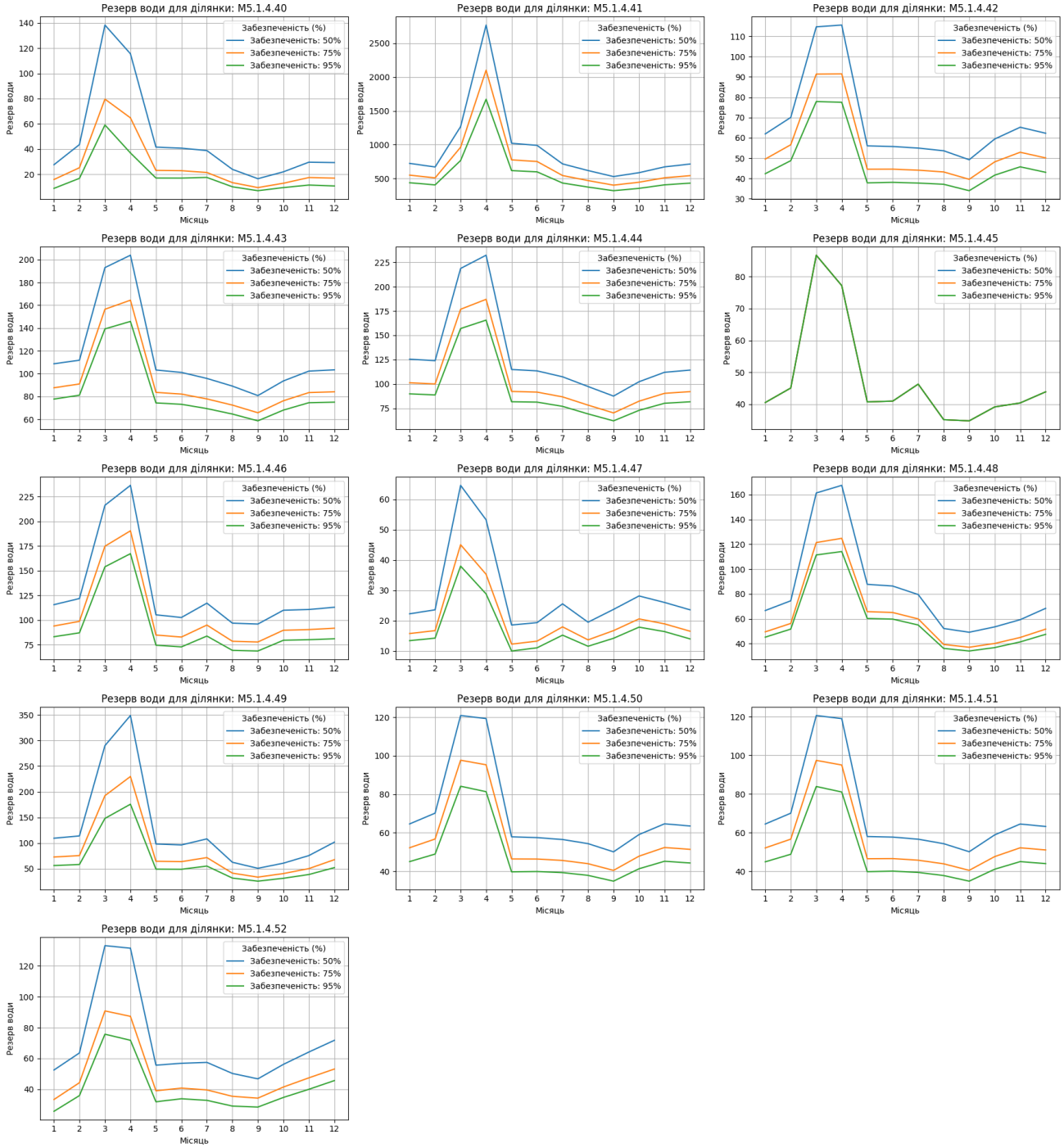


Рисунок 2.9 – Резерв води для ділянок суббасейну Прип’яті

Здійснено побудову тематичної карти для резервів водогосподарських ділянок суббасейну (рис. 2.10).

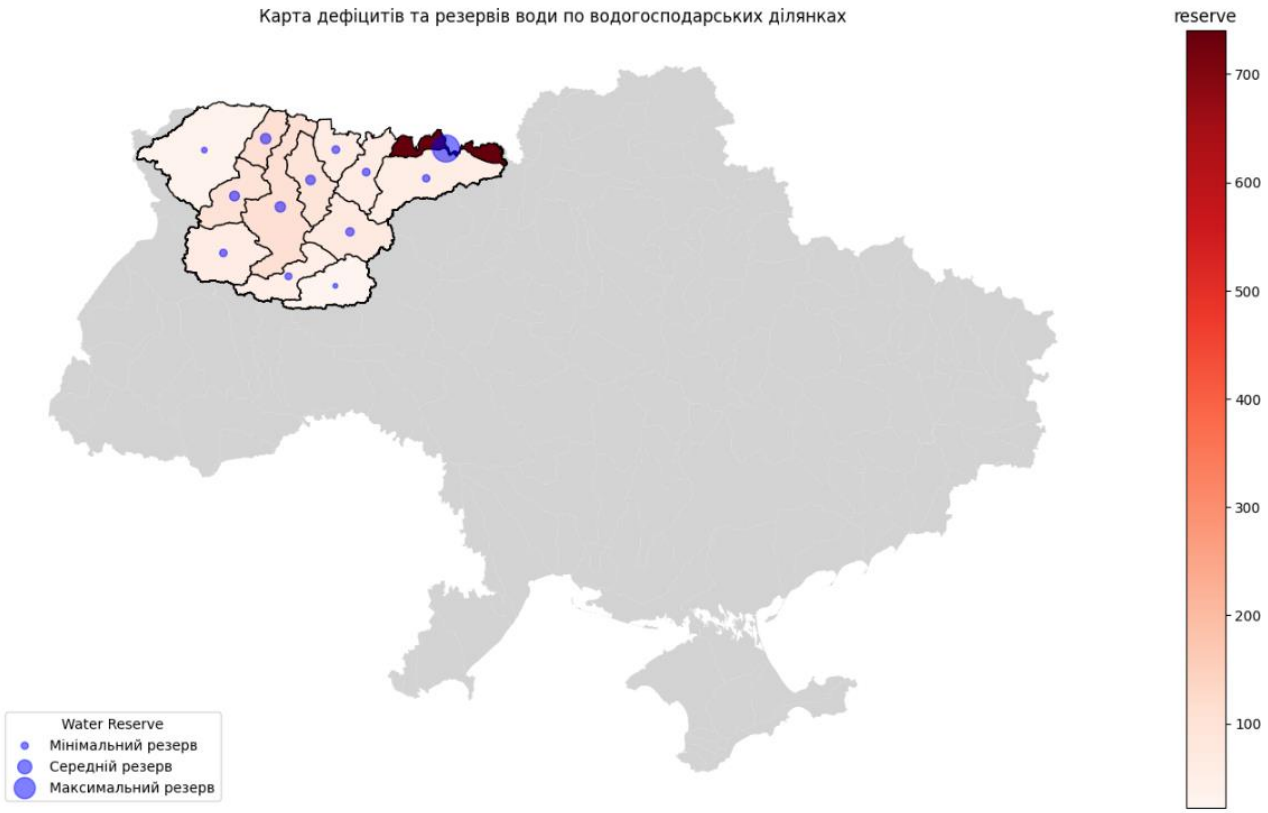


Рисунок 2.10 – Тематична карта для резервів водогосподарських ділянок суббасейну

Визначено водогосподарські ділянки з найменшими резервами води (рис. 2.11, 2.12).

code	reserve	SEM116
M5.1.4.47	22.107194	р. Случ від витоку до гирла р. Хомора (включаючи р. Хомора)
M5.1.4.40	30.938050	р. Прип'ять від витоку до державного кордону
M5.1.4.45	47.539550	р. Горинь від витоку до кордону Хмельницької та Рівненської областей

Рисунок 2.11 – Водогосподарські ділянки з найменшими резервами води

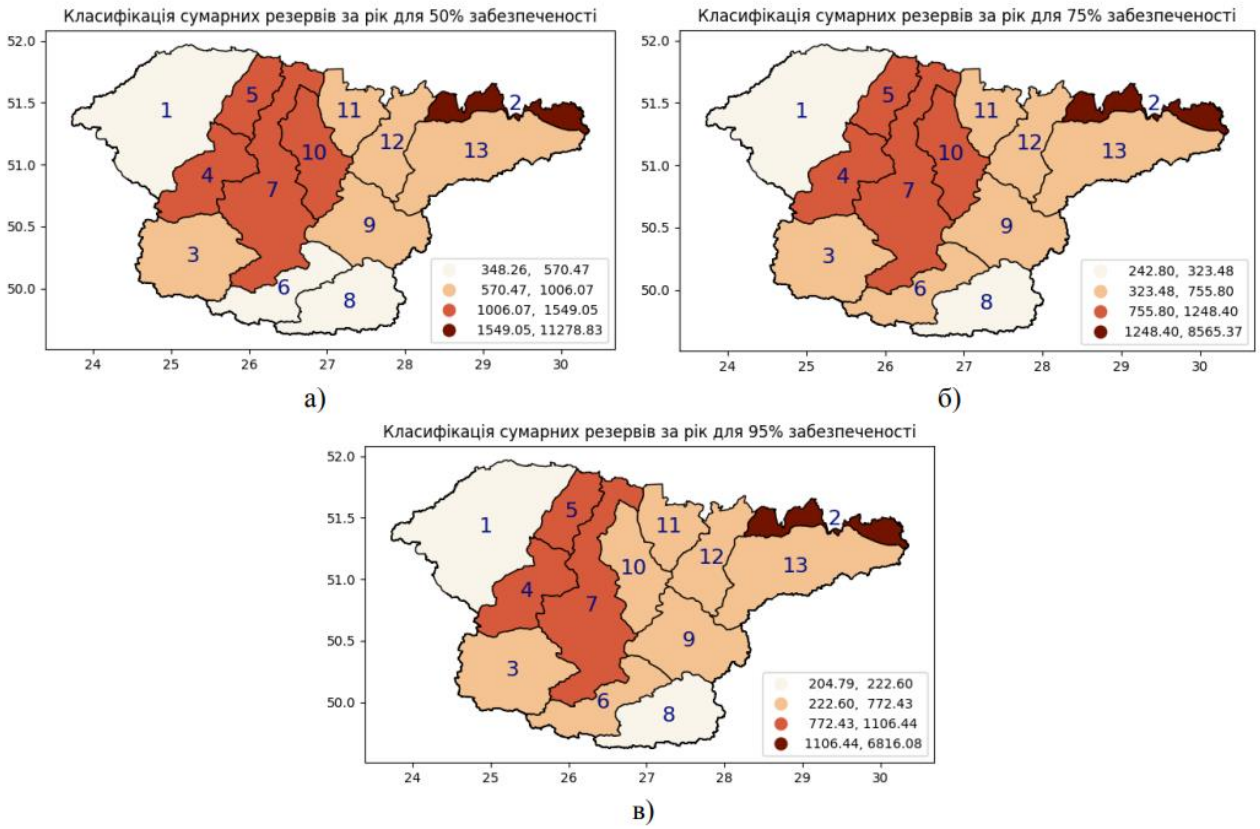


Рисунок 2.12 – Результати виконання класифікації сумарних резервів за рік: а) для забезпеченості 50%; б) для забезпеченості 75%; в) для забезпеченості 95%

2.3 Виявлення аномальних значень

Для подальшого прогнозування витрат води в суббасейні При було проведено розвідувальний аналіз даних (EDA) [25] на основі спостережень гідропосту на річці Горинь біля міста Нетішин. Дослідження охоплює середньомісячні спостереження витрат води за 1946-2023 роки.

Першим кроком аналізу стало дослідження середньомісячних значень витрат води. На рисунку 2.13 наведено графік, що ілюструє зміну середнього рівня води упродовж усього періоду спостережень.

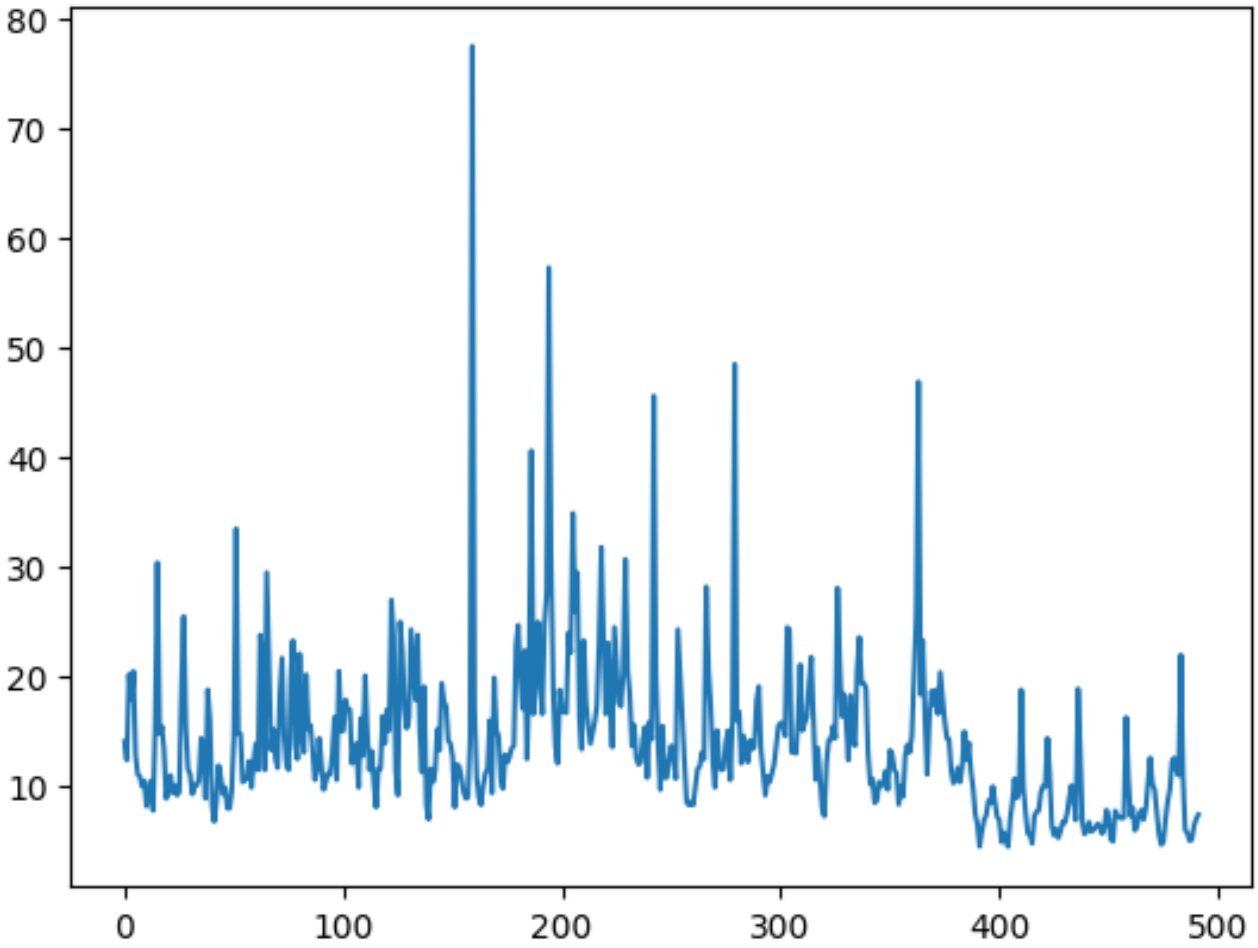


Рисунок 2.13 – Графік середньомісячних витрат води

У задачах аналізу часових рядів, зокрема у сфері прогнозування гідрологічних показників, важливим кроком є виявлення та обробка аномальних значень (аномалій) (рис. 2.14). Аномальні значення — це такі спостереження, які суттєво відрізняються від загальної закономірності або поведінки даних. Вони можуть виникати з різних причин і мати різний вплив на результати моделювання.

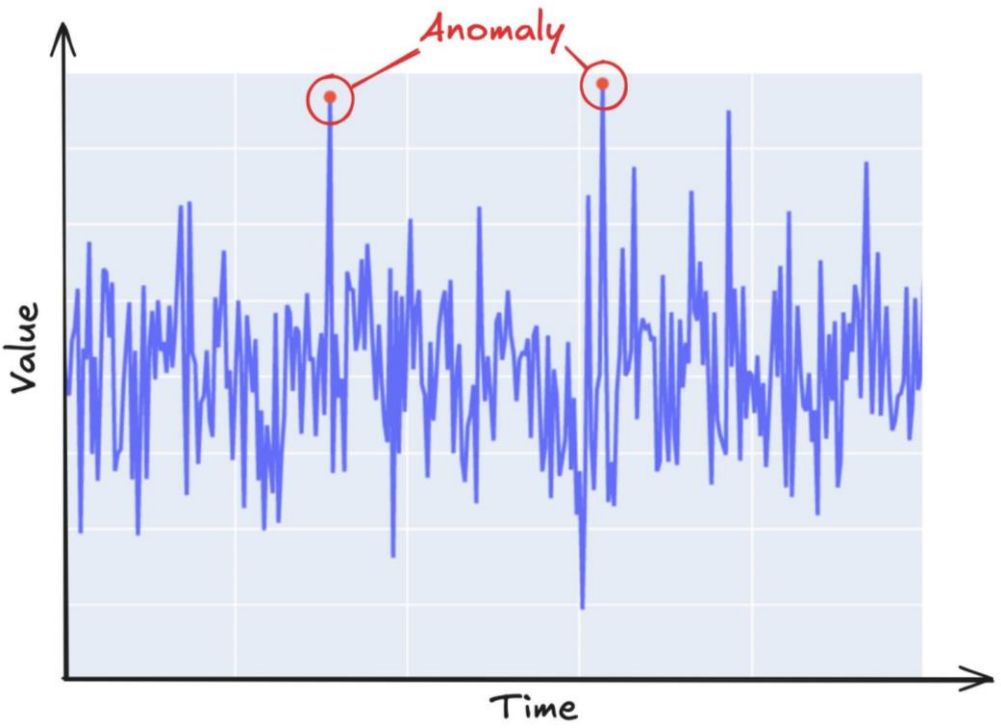


Рисунок 2.14 – Приклад аномалій

У контексті гідрологічних даних (витрати води, рівні, опади тощо) до потенційних джерел аномальних значень можна віднести:

- Похибки при вимірюваннях — викликані несправністю приладів, людським фактором або збоями у зборі даних;
- Природні екстремальні події — паводки, посухи, танення снігу, раптові опади;
- Антропогенні впливи — аварійні скиди, зміна режиму регулювання водосховищ, вплив меліорації;
- Зміна методології спостережень або формату представлення даних;
- Помилки при обробці чи трансформації даних, наприклад, дублікати, зміщення, пропущені значення, неправильне масштабування.

Аномальні значення можуть істотно спотворювати як статистичні розрахунки, так і якість прогнозів:

- Зсув середніх значень та дисперсії, що веде до хибної інтерпретації сезонності або тренду;

- Перенавчання моделі — алгоритм намагається підлаштуватися під аномальні точки, втрачаючи узагальнювальну здатність;
- Зниження точності прогнозу на тестових даних;
- Поява шумів у візуалізації, що ускладнює аналітичне сприйняття.

Саме тому очищення даних від аномалій або їхня коректна обробка є необхідною умовою для побудови адекватних прогнозних моделей.

Існує декілька підходів до виявлення аномальних значень у часових рядах:

1. Статистичні методи:
    - Відхилення від середнього значення на понад 2–3 стандартних відхилення (z-score);
    - Перевищення міжквартильного діапазону (IQR);
    - Побудова контрольних меж у ковзному вікні.
2. Машинне навчання:
    - Використання методів кластеризації (наприклад, DBSCAN) для виявлення аномальних точок;
    - Autoencoder-мережі або ізоляційні дерева (Isolation Forest);
    - Побудова моделей передбачення і оцінка залишків (наприклад, за допомогою ARIMA або Prophet).
3. Візуальні методи:
    - Аналіз графіків (line plot, boxplot, scatter) для виявлення вибухів чи спадів;
    - Візуалізація ковзних середніх, залишків моделей, сезонних компонентів.

Після ідентифікації аномальних значень можливі кілька підходів до їх обробки:

1. Ігнорування (залишити без змін) — якщо вони відображають реальні події;
2. Вилучення — для зменшення впливу на модель;
3. Заміна — наприклад, середнім або інтерпольованим значенням;
4. Коригування — у випадку відомих джерел похибки (наприклад, нульове значення замість відсутнього).

Виявлення та коректна обробка аномальних значень є важливим етапом перед моделюванням, оскільки без цього жодна модель — незалежно від її складності — не здатна забезпечити стабільний і точний прогноз. Для задач гідрологічного прогнозування це особливо критично, адже аномальні пікові значення можуть суттєво вплинути на водогосподарські рішення в контексті екологічної безпеки, управління паводками та планування водокористування.

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 2.15.

	ds	lower_window	upper_window	prior_scale	holiday
0	1996-04-15	0	0	10	anomalous_dates
1	1998-07-16	0	0	10	anomalous_dates
2	1999-03-16	0	0	10	anomalous_dates
3	2003-03-16	0	0	10	anomalous_dates
4	2006-04-15	0	0	10	anomalous_dates
5	2013-04-15	0	0	10	anomalous_dates
6	2023-04-15	0	0	10	anomalous_dates

Рисунок 2.15 – 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.

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 Прип’ять. 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.

2.4 Висновки

У другому розділі було реалізовано повний процес підготовки даних: від первинного зчитування інформації з різнорідних джерел до побудови структурованого та інформативного набору характеристик, придатного для моделювання прогнозів. Було виконано уніфікацію та трансформацію таблиць водогосподарських балансів, середньомісячних даних про витрати води на гідропості в місті Нетішин, а також просторових шейп-файлів. Усі ці дані були розміщені на платформі Kaggle, що дозволяє забезпечити повторюваність аналізу та подальше розширення даних.

Під час дослідження балансових показників було виявлено значну просторову нерівномірність у розподілі дефіциту та запасів води, що дало змогу окреслити найбільш критичні зони з мінімальними водними резервами.

Загалом ці підготовчі дії сформували міцне підґрунтя для наступного етапу – оцінювання та вибору алгоритмів машинного навчання з метою оптимізації системи управління водними ресурсами в межах суббасейну річки Прип’ять.

3 ПРОГНОЗУВАННЯ РІВНЯ ВОДИ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

3.1 Опис моделей машинного навчання для прогнозування рівня води

Для здійснення прогнозу витрат води у річці Горинь було залучено широкий спектр моделей, зокрема: Facebook Prophet, AutoARIMA, Linear Regression, KNeighbors Regressor, Support Vector Machines, Linear SVR, Random Forest Regressor, Bagging Regressor, XGB Regressor та MLP Regressor. Основною метою такого різноманіття підходів було забезпечення можливості порівняльного аналізу їх продуктивності для вибору найбільш ефективної архітектури прогнозної системи.

Facebook Prophet – інструмент для моделювання часових рядів, який базується на адитивній структурі. Він дозволяє враховувати нелінійні тренди, сезонні коливання (річні, тижневі, денні) та святкові дні [8]. Метод розділяє ряд на кілька складових – тренд, сезонність і вплив подій – та об'єднує їх у цілісну модель. Prophet вирізняється автоматизованістю, зрозумілим розділенням впливів та стійкістю до пропущених значень і аномалій. Недоліком є відсутність авторегресійної частини, що ускладнює моделювання складних часових залежностей, а також зниження точності при довгострокових прогнозах та обмеження через припущення про адитивність компонентів.

ARIMA (autoregressive integrated moving average) – класична модель для роботи з нестаціонарними часовими рядами. Її автоматизований варіант, Auto-ARIMA, самостійно підбирає найкращі параметри моделі, використовуючи статистичні тести, зокрема тест Дікі-Фуллера, Квятковського-Філіпса-Шмідта-Шина та тест Перрона, для визначення порядку інтегрування [9-10]. У разі сезонності модель додатково виконує тест Канова-Гансена. Хоча цей підхід дозволяє уникнути ручного підбору параметрів, він вимагає значних обчислювальних ресурсів і може зупинятися на локальних, а не глобальних, оптимумах.

Linear Regression – один із базових методів, який знаходить оптимальні

коефіцієнти моделі, що мінімізують відхилення прогнозів від фактичних значень [11].

KNeighbors Regressor передбачає значення цільової змінної на основі k найближчих до точки прогнозу прикладів, здійснюючи локальну інтерполяцію [12].

Support Vector Machines (SVM) – методи, які підходять як для класифікаційних, так і для регресійних задач. У варіанті SVR (Support Vector Regression) передбачення здійснюється в межах допустимого рівня похибки [13]. Linear SVR реалізує аналогічний принцип, проте використовує інший тип функції втрат і базується на бібліотеці liblinear [14].

Random Forest Regressor є ансамблевим методом, що поєднує результати багатьох дерев рішень, побудованих на різних підвибірках даних, задля підвищення точності та зниження ризику перенавчання [15]. Bagging Regressor працює за схожим принципом, навчаючи окремі моделі на випадкових підмножинах і агрегуючи їх прогнози в єдиний результат [16].

XGB Regressor реалізує ефективний алгоритм градієнтного бустингу, оптимізований для високопродуктивної обробки великих обсягів даних і зручної інтеграції з інструментами scikit-learn [17].

MLP Regressor – це багатошарова нейромережа, яка навчається через алгоритм зворотного поширення помилки. Вона не містить активаційної функції на вихідному шарі, що робить її придатною для регресійних задач [18].

Оцінювані алгоритми мають різні характеристики, що впливають на їхню ефективність у контексті прогнозування часових рядів. Методи, як-от Linear Regression, є швидкими, простими в реалізації та інтерпретованими, однак мають обмеження через припущення про лінійність зв’язків.

KNeighbors Regressor ефективний у задачах із нерегулярними розподілами, проте не враховує часову структуру, що є критичним для динамічних процесів. SVM та Linear SVR можуть будувати як лінійні, так і нелінійні моделі залежно від ядра, показують хорошу узагальнюваність при невеликих даних, але мають високу обчислювальну складність і чутливість до параметрів. Linear SVR, хоча і менш гнучкий, краще масштабується.

Ансамблеві методи, як Random Forest та Bagging, добре справляються зі складними патернами, нечутливі до шуму, але без спеціальної обробки не враховують часову послідовність, що знижує їхню точність у задачах прогнозування.

XGBoost забезпечує гнучкість та високу точність, але потребує належної настройки та достатніх ресурсів. MLP Regressor здатен навчатися на великих обсягах даних та моделювати складні нелінійні взаємозв'язки, однак є менш інтерпретованим, чутливим до перенавчання та вимагає уважного вибору архітектури.

Попри обмеження, пов'язані з відсутністю вбудованої підтримки часових залежностей у більшості моделей, Facebook Prophet і AutoARIMA виділяються як спеціалізовані інструменти для роботи з часовими рядами. Вони автоматично враховують сезонність і тренди, що особливо цінне в контексті гідрологічного моделювання, де часові закономірності відіграють ключову роль.

3.2 Тренування моделей машинного навчання

Модель Facebook Prophet була використана як одна з базових інструментів для прогнозування середньодобового рівня води на гідропості Нетішин. Prophet є адитивною моделлю часового ряду, яка враховує тренд, сезонність і вплив зовнішніх подій (наприклад, аномальних дат). У розробленому ноутбуці «Water discharge – EDA & forecasting» [26] 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. 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 витрат 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.

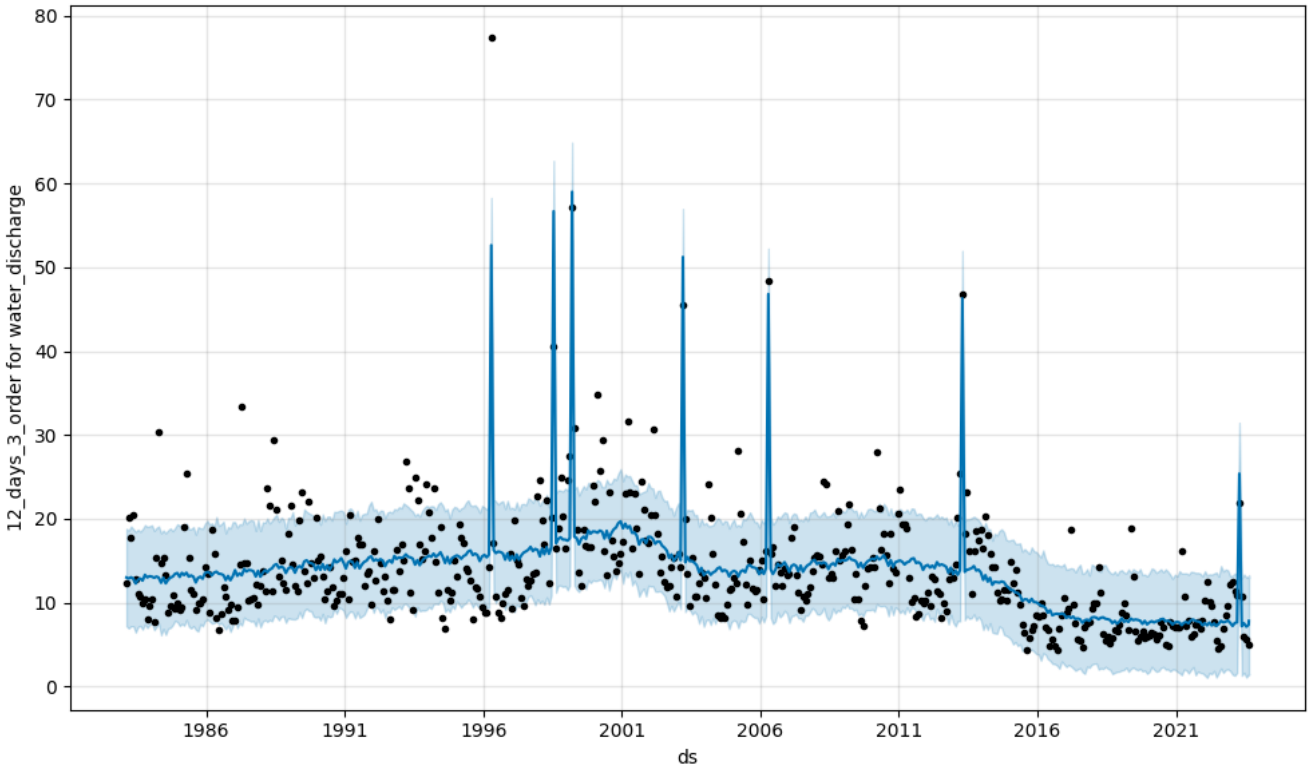


Рисунок 3.2 – Результат тренування моделі Prophet_12_days_3_order

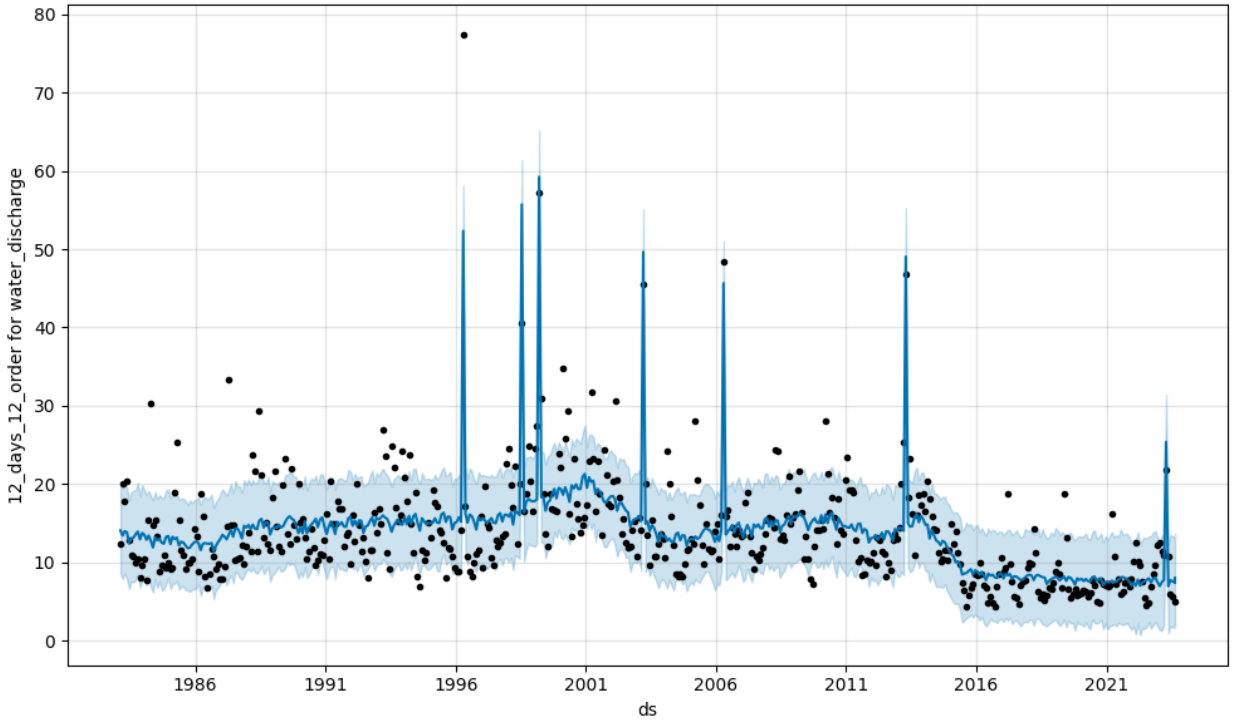


Рисунок 3.3 – Результат тренування моделі Prophet_12_days_12_order

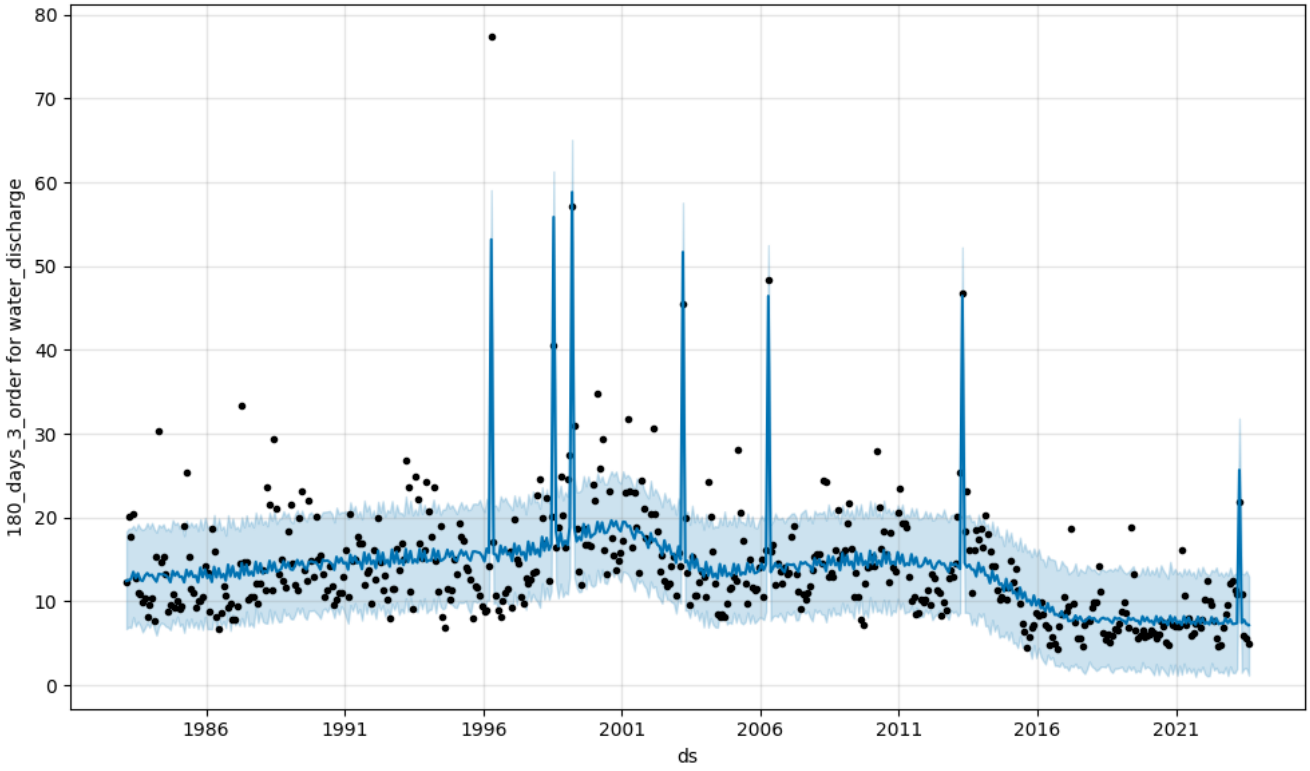


Рисунок 3.4 – Результат тренування моделі Prophet_180_days_3_order

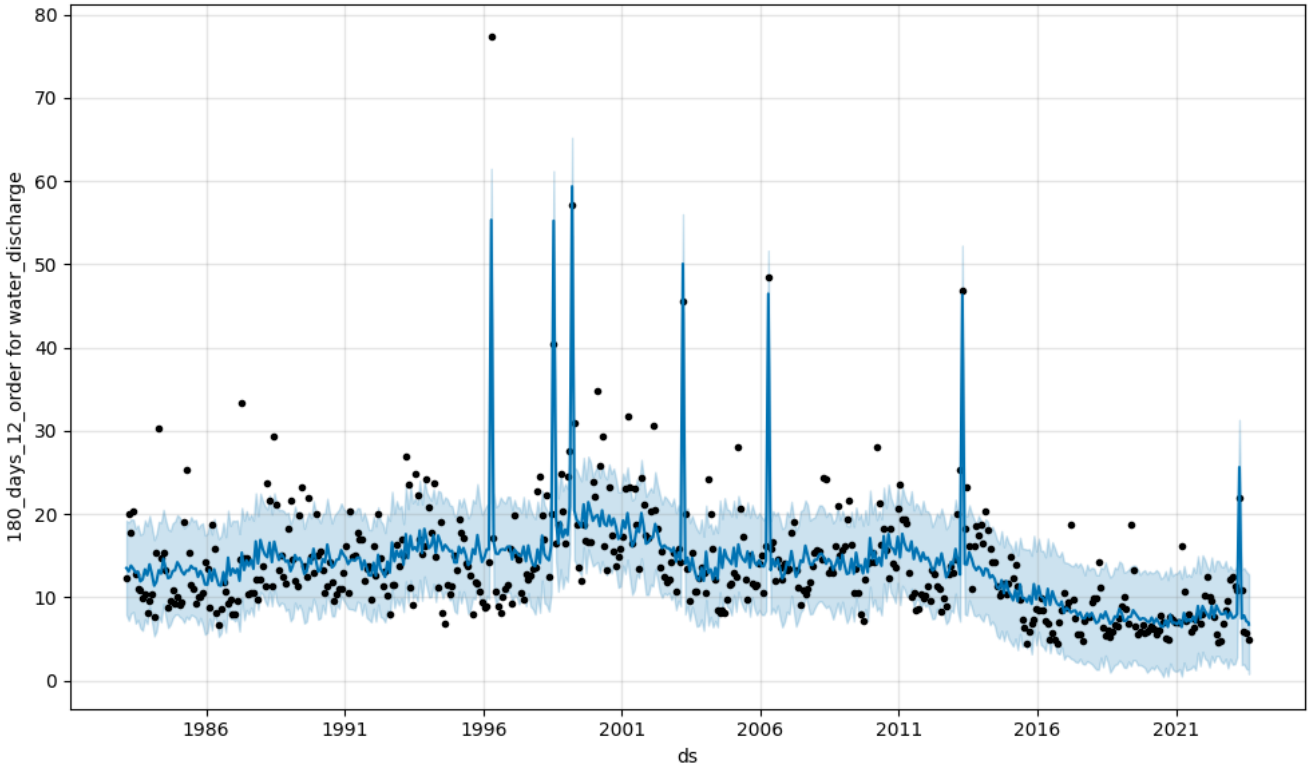


Рисунок 3.5 – Результат тренування моделі Prophet_180_days_12_order

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.

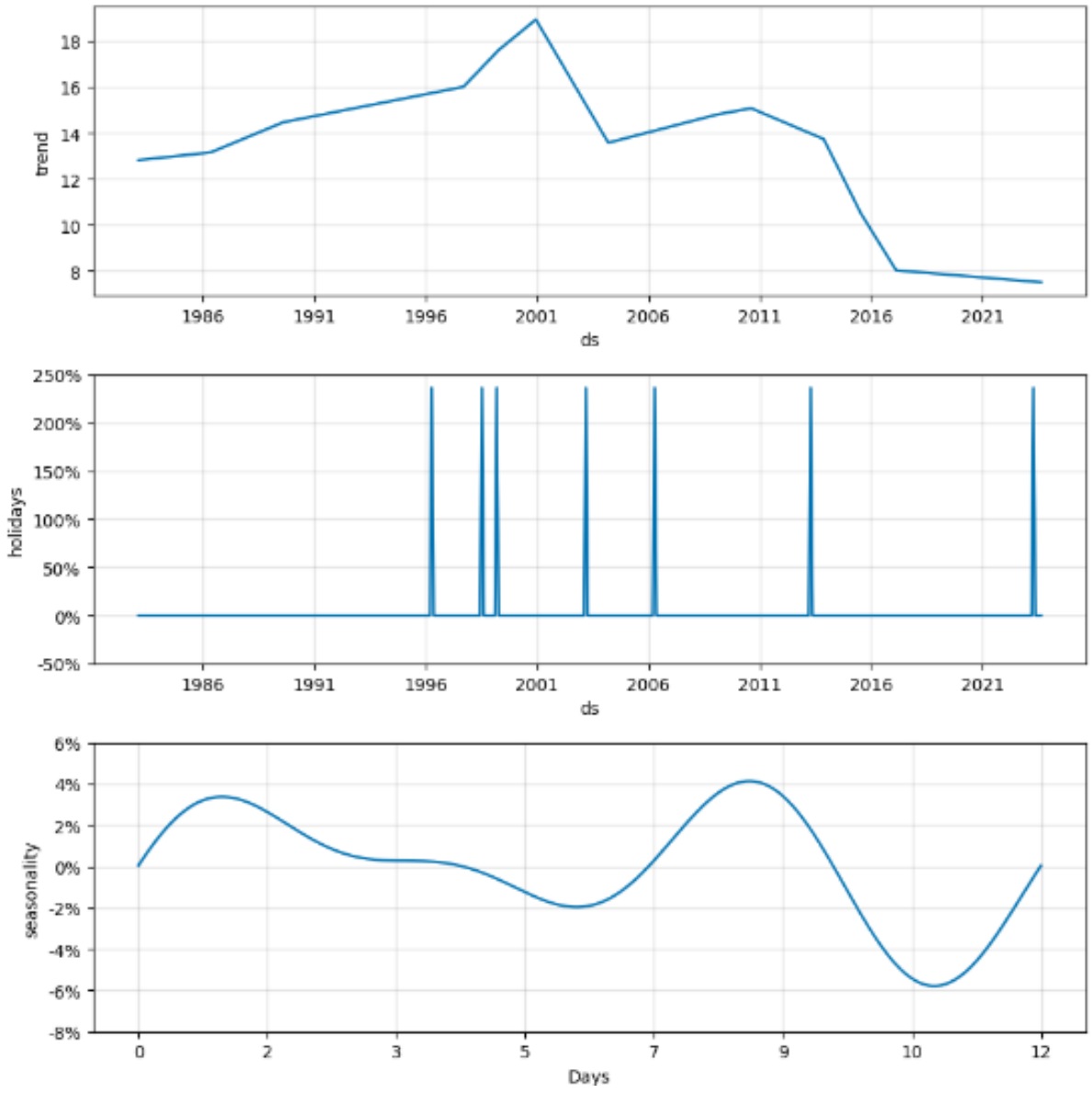


Рисунок 3.6 – Компоненти моделі Prophet_12_days_3_order

На рисунку 3.7 приведено компоненти моделі Prophet_12_days_12_order, за якими можна оцінити вплив довжини сезонного періоду та кількості гармонік (порядку Фур’є-розкладу) на точність і стабільність передбачення.

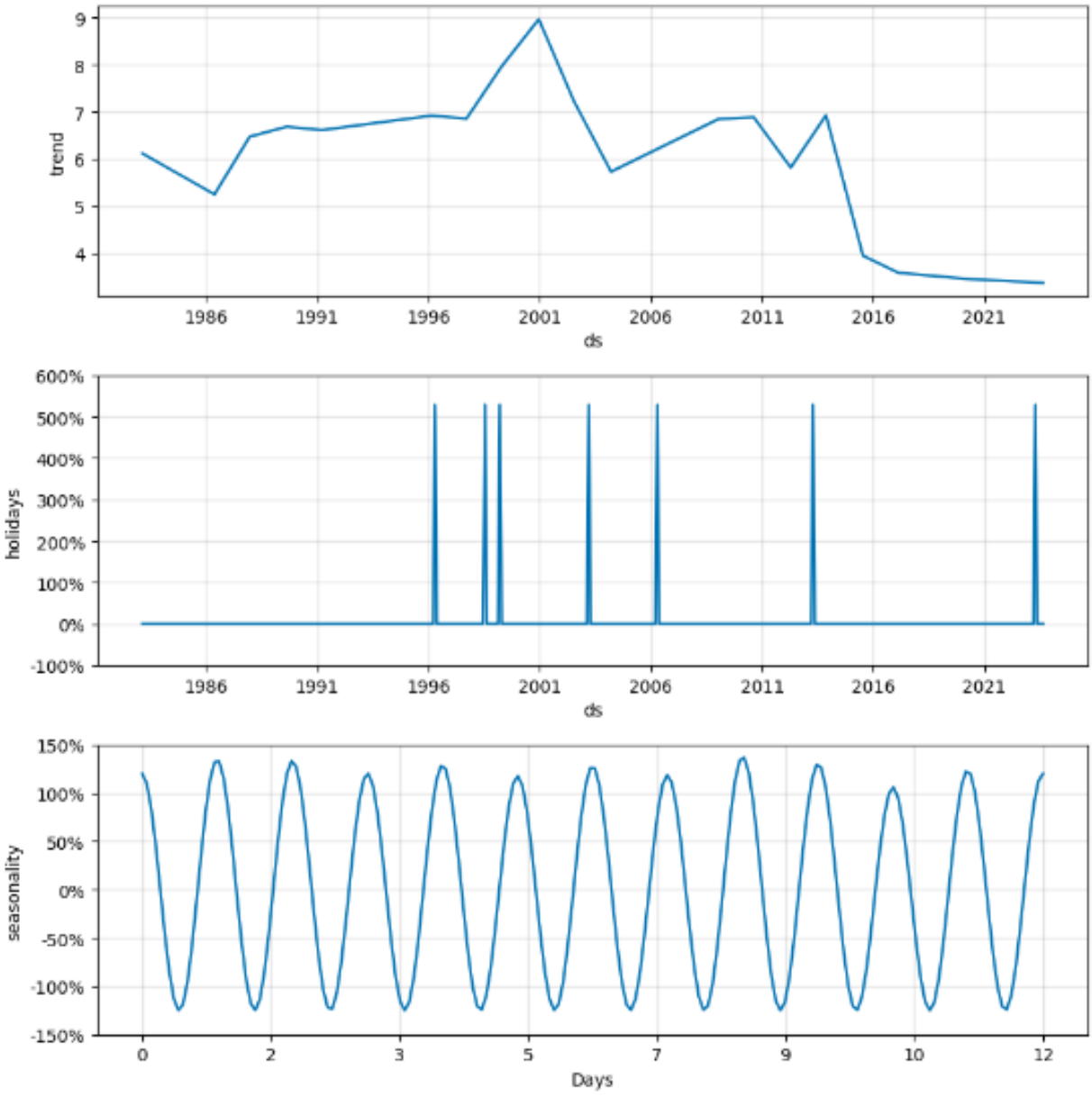


Рисунок 3.7 – Компоненти моделі Prophet_12_days_12_order

На рисунку 3.8 приведено компоненти моделі Prophet_180_days_3_order, за якими можна оцінити вплив довжини сезонного періоду та кількості гармонік (порядку Фур’є-розкладу) на точність і стабільність передбачення.

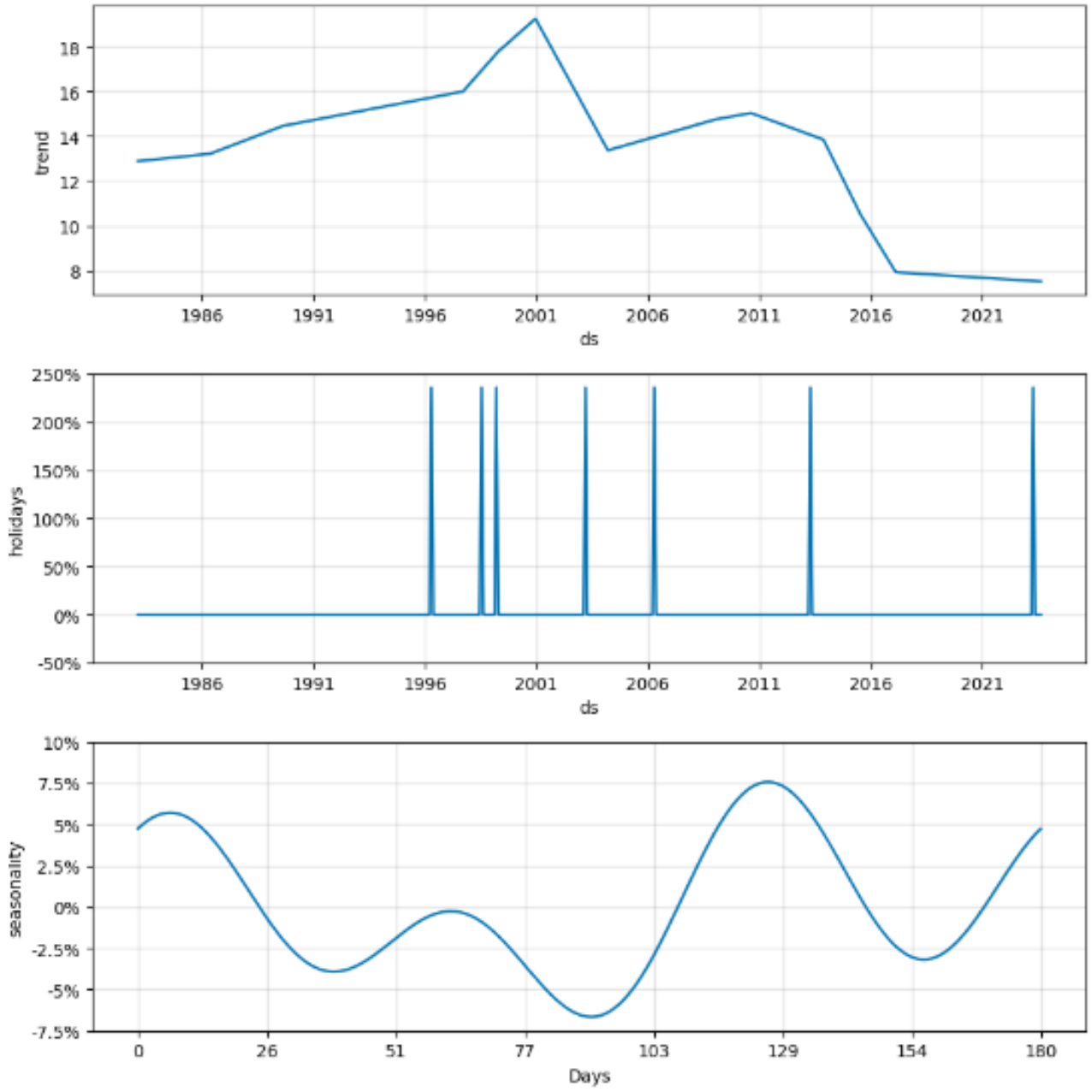


Рисунок 3.8 – Компоненти моделі Prophet_180_days_3_order

На рисунку 3.9 приведено компоненти моделі Prophet_180_days_12_order, за якими можна оцінити вплив довжини сезонного періоду та кількості гармонік (порядку Фур'є-розкладу) на точність і стабільність передбачення.

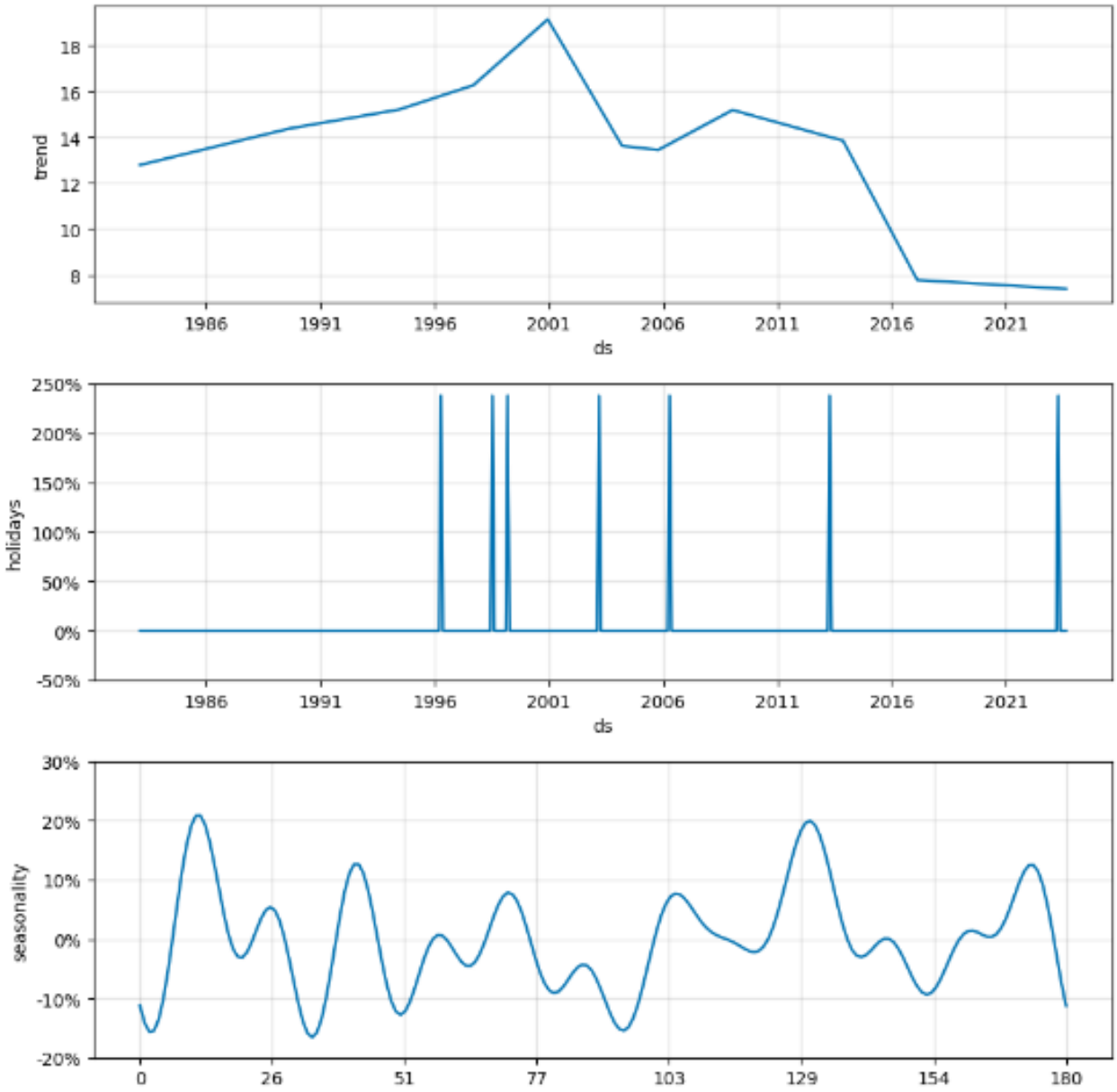


Рисунок 3.9 – Компоненти моделі Prophet_180_days_12_order

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.

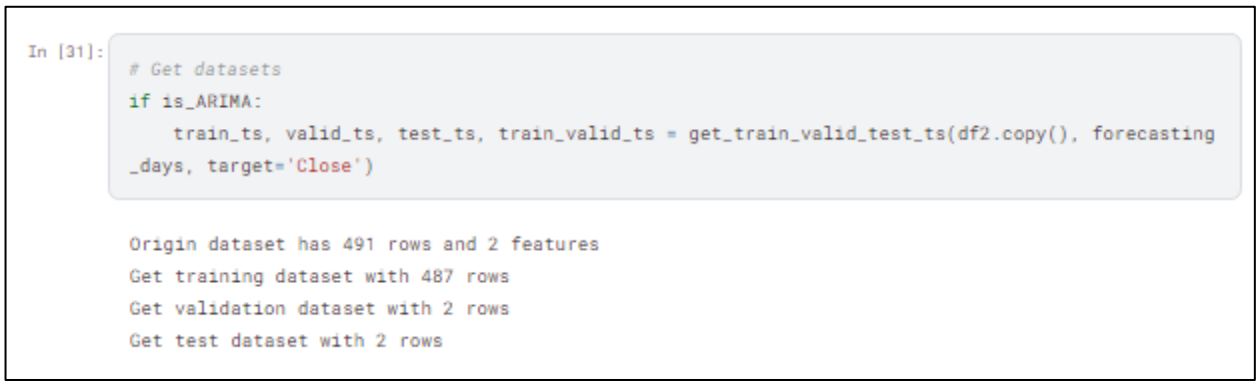


Рисунок 3.10 – Розбиття даних на тренувальний, валідаційний і тестовий набори для побудови моделі ARIMA

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).

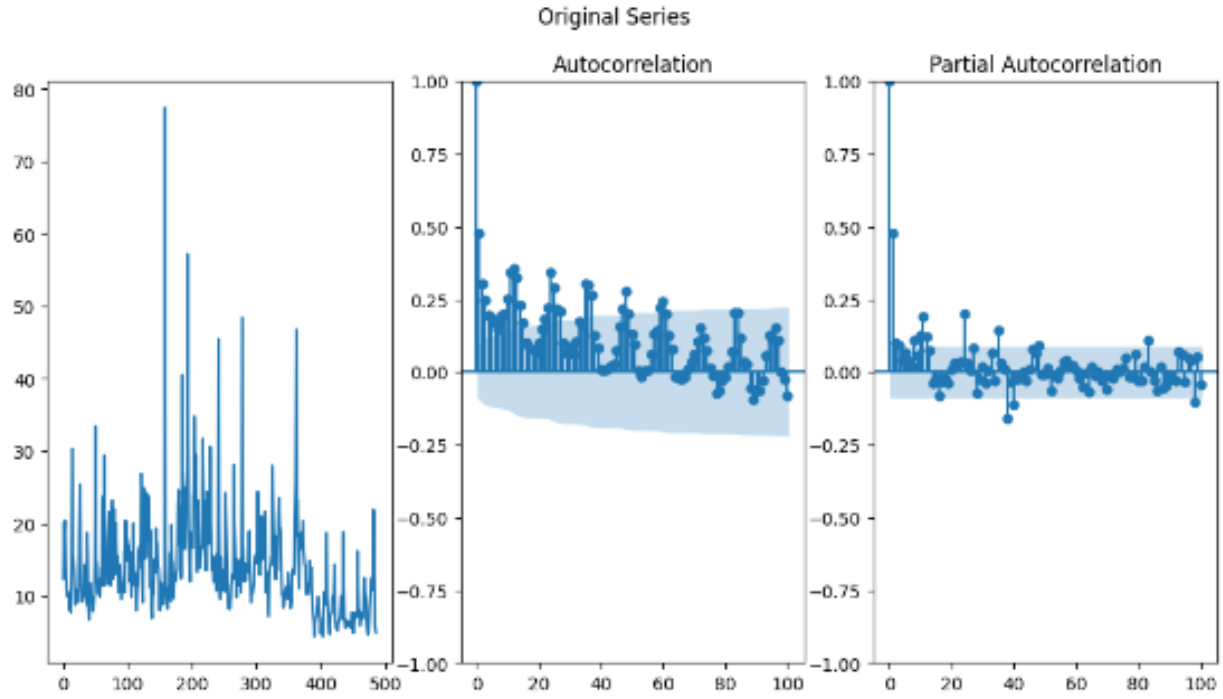


Рисунок 3.11 – ACF та PACF для вихідного часового ряду (Original Series)

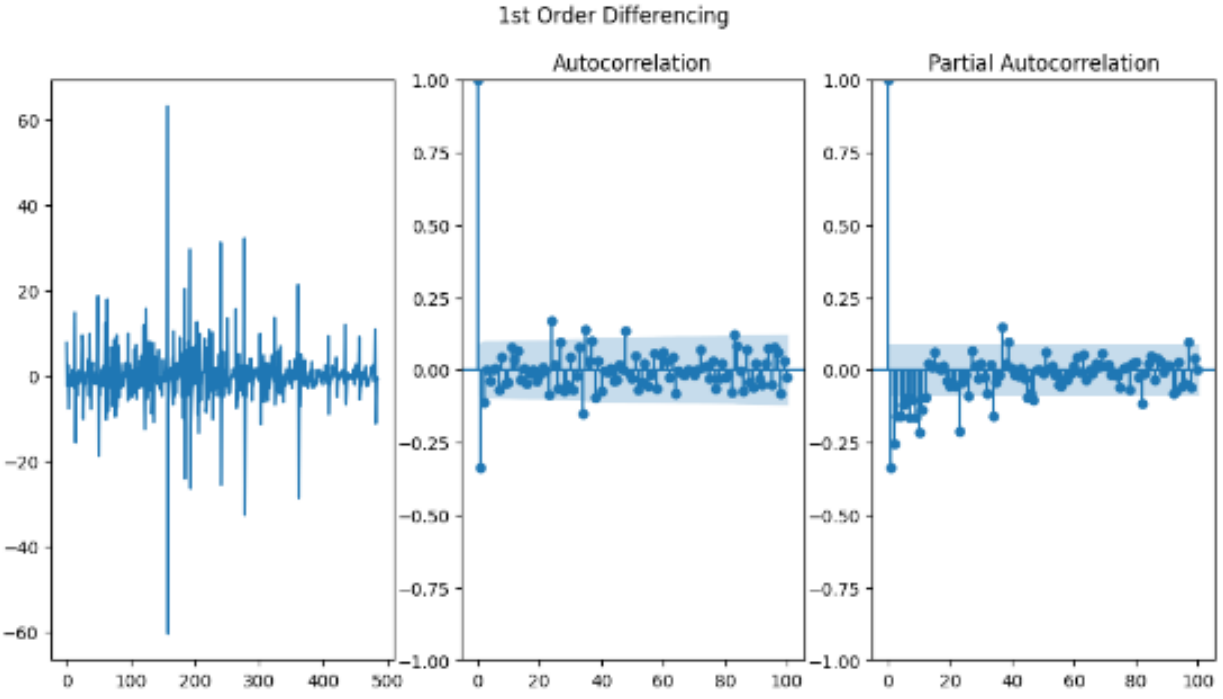


Рисунок 3.12 – ACF та PACF для ряду після першого диференціювання (1st Order Differencing)

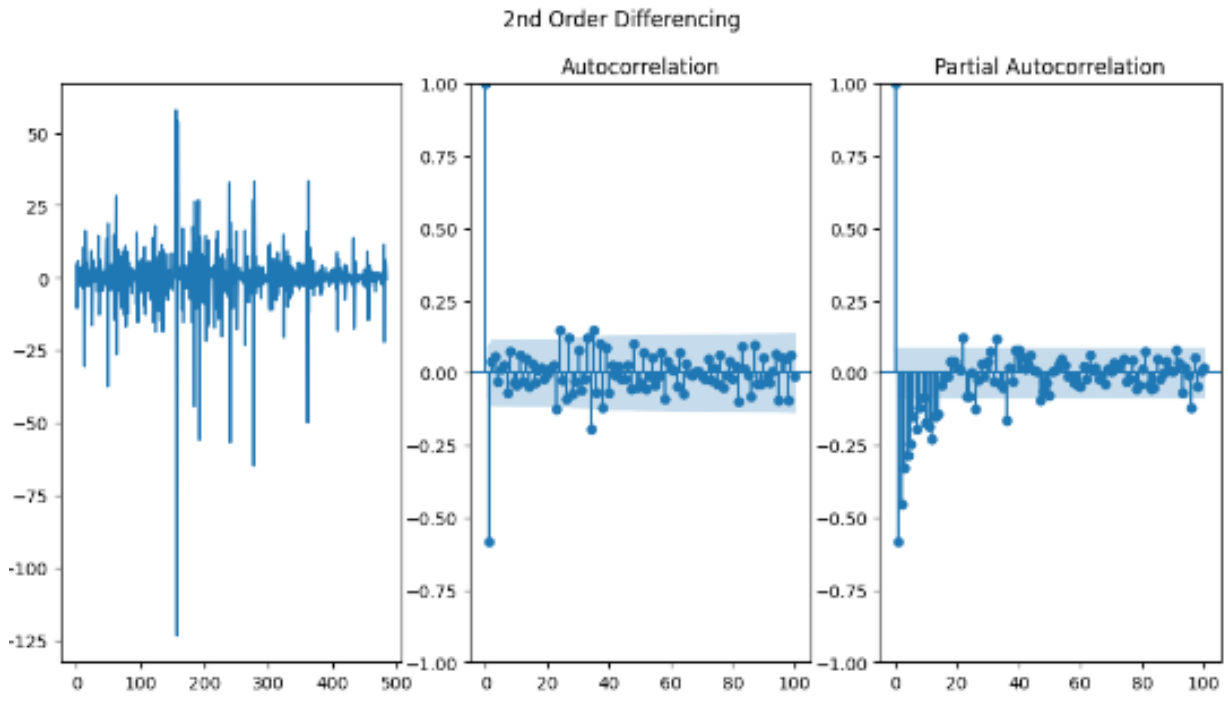


Рисунок 3.13 – ACF та PACF для ряду після другого диференціювання (2nd Order Differencing)

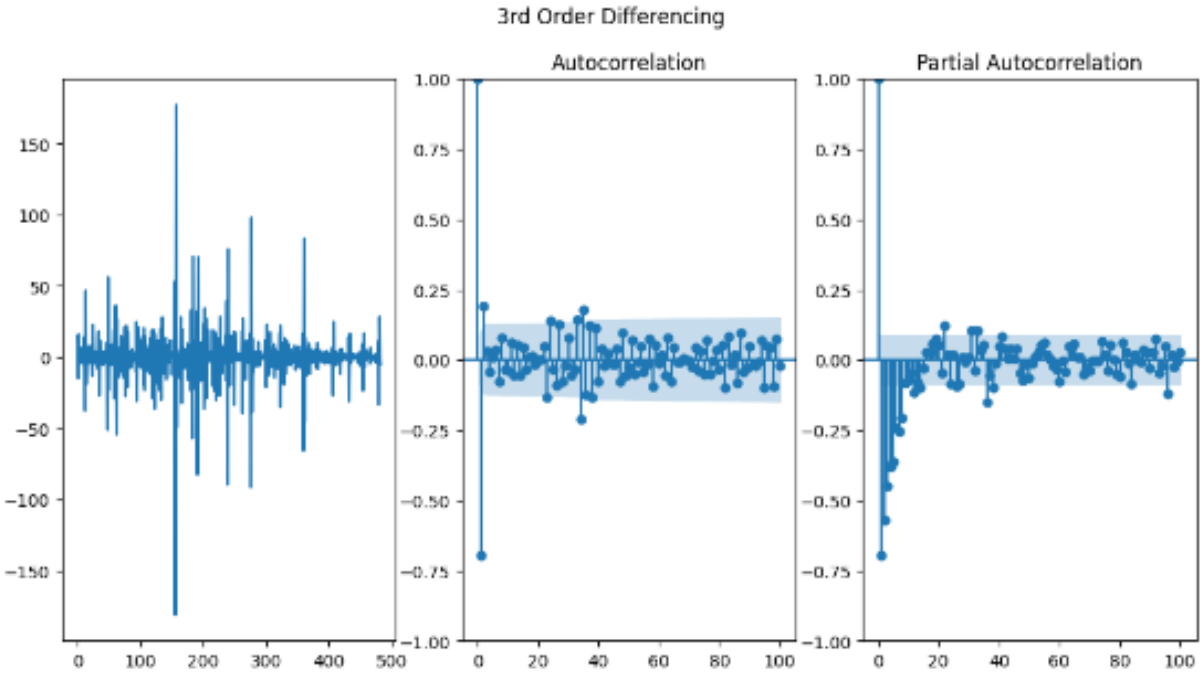


Рисунок 3.14 – ACF та PACF для ряду після третього диференціювання (3rd Order Differencing)

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=3145.490, Time=1.44 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=4061.728, Time=0.02 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=3288.482, Time=0.03 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=3710.497, Time=0.06 sec
 ARIMA(3,0,4)(0,0,0)[0]             : AIC=3152.429, Time=1.07 sec
 ARIMA(4,0,3)(0,0,0)[0]             : AIC=inf, Time=3.61 sec
 ARIMA(5,0,4)(0,0,0)[0]             : AIC=inf, Time=1.62 sec
 ARIMA(4,0,5)(0,0,0)[0]             : AIC=3151.240, Time=1.38 sec
 ARIMA(3,0,3)(0,0,0)[0]             : AIC=3148.288, Time=0.94 sec
 ARIMA(3,0,5)(0,0,0)[0]             : AIC=3150.363, Time=1.27 sec
 ARIMA(5,0,3)(0,0,0)[0]             : AIC=inf, Time=1.78 sec
 ARIMA(5,0,5)(0,0,0)[0]             : AIC=3148.563, Time=1.81 sec
 ARIMA(4,0,4)(0,0,0)[0] intercept   : AIC=3148.123, Time=1.71 sec

Best model:  ARIMA(4,0,4)(0,0,0)[0]
Total fit time: 16.888 seconds
                               SARIMAX Results
==========================================================================
Dep. Variable:                      y   No. Observations:                  487
Model:               SARIMAX(4, 0, 4)   Log Likelihood               -1563.745
Date:                Mon, 26 May 2025   AIC                          3145.490
Time:                        17:42:57   BIC                          3183.184
Sample:                             0   HQIC                         3160.297
                                  - 487
Covariance Type:                  opg
==========================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------
ar.L1          0.4208      0.132      3.176      0.001       0.161       0.680
ar.L2         -0.0033      0.021     -0.161      0.872      -0.044       0.037
ar.L3          0.9838      0.021     46.310      0.000       0.942       1.025
ar.L4         -0.4015      0.138     -3.886      0.002      -0.656      -0.147
ma.L1         -0.0443      0.136     -0.326      0.745      -0.311       0.222
ma.L2          0.0501      0.026      1.900      0.057      -0.002       0.102
ma.L3         -0.9413      0.027    -34.319      0.000      -0.995      -0.887
ma.L4          0.0460      0.125      0.370      0.712      -0.198       0.290
sigma2        35.3600      0.732     48.337      0.000      33.926      36.794
==========================================================================
Ljung-Box (L1) (Q):                   0.04   Jarque-Bera (JB):             23244.30
Prob(Q):                              0.84   Prob(JB):                         0.00
Heteroskedasticity (H):               0.34   Skew:                             4.15
Prob(H) (two-sided):                  0.00   Kurtosis:                        35.81
==========================================================================
```

Рисунок 3.15 – Автоматичний вибір параметрів ARIMA та діагностика моделі

Статистика Ljung-Box (p = 0.84) 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.

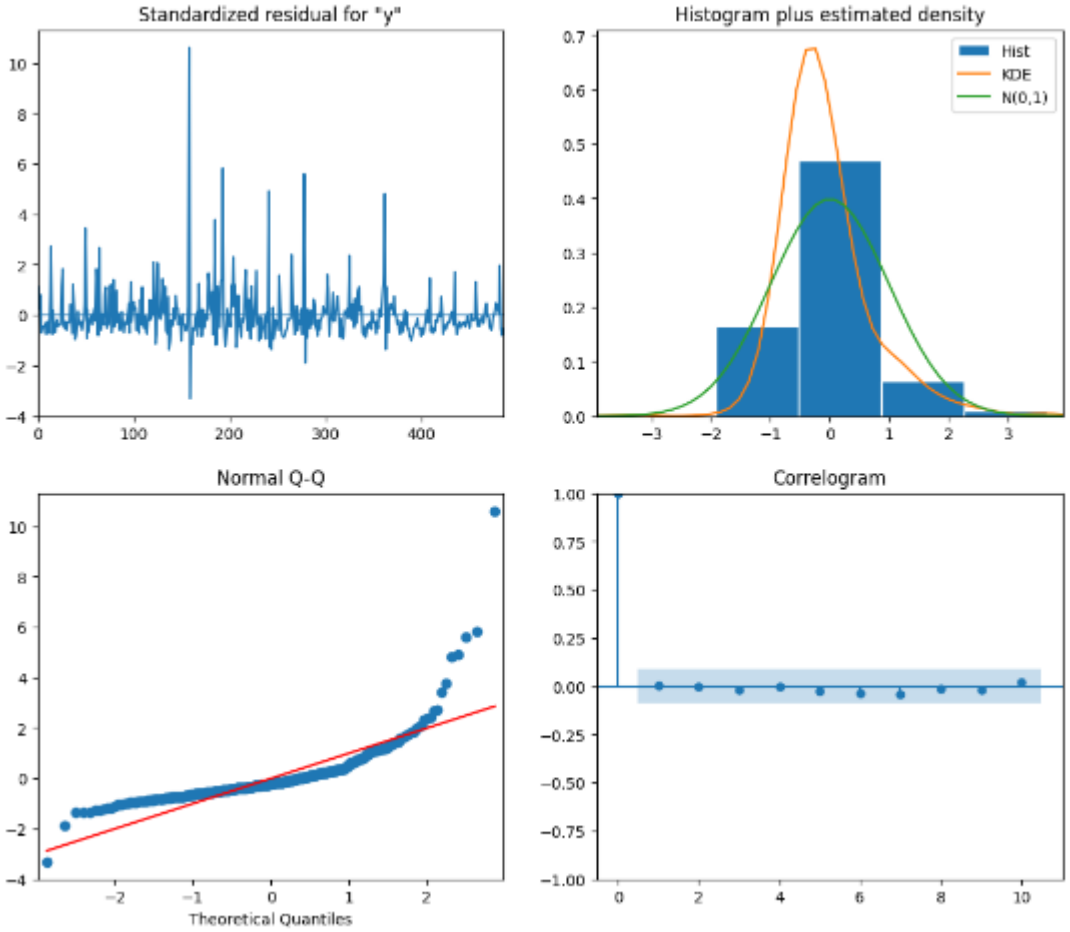


Рисунок 3.16 – Діагностика залишків моделі ARIMA

Решта (залишки) коливаються поблизу нульової осі, що свідчить про відсутність систематичних відхилень у прогнозах. Аналіз графіка розподілу густини виявляє асиметричну форму, що вказує на відхилення від нормального розподілу. При цьому автокореляційний аналіз залишків не виявив статистично значущих зв’язків, що підтверджує їхню часову незалежність та свідчить про відповідність моделі.

Таким чином, побудована модель ARIMA продемонструвала прийнятні

властивості залишків, статистичну значущість і здатність якісно прогнозувати на основі валідаційних даних. Вона була обрана як одна з еталонних моделей для подальшого зіставлення з іншими підходами, зокрема моделлю Prophet та низкою методів машинного навчання.

Після створення моделей ARIMA та Facebook Prophet було реалізовано набір моделей машинного навчання, які працюють з мультифакторними вхідними даними. Для цього на етапі формування ознак за допомогою бібліотеки TSFRESH було згенеровано розширений датасет, який згодом поділили на навчальну, валідаційну та тестову частини (рис. 3.17).

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.

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.

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

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 [19].

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

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

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 [19].

На рисунку 3.19 наведено таблиця зі значеннями RMSE та MAPE для всіх

побудованих моделей на валідаційних даних:

	name_model	type_data	rmse	mape
12	MLP Regressor	valid	0.284465	4.979669
9	Random Forest Regressor	valid	0.379591	6.790386
7	Support Vector Machines	valid	0.511406	7.280615
6	KNeighbors Regressor	valid	0.434058	7.750227
11	XGB Regressor	valid	0.663275	8.452855
5	Linear Regression	valid	1.043145	14.015372
3	Prophet_180_days_12_order	valid	1.285184	21.0554
2	Prophet_180_days_3_order	valid	1.650448	28.496155
1	Prophet_12_days_12_order	valid	2.068762	37.192197
0	Prophet_12_days_3_order	valid	2.100708	37.465432
4	ARIMA_auto	valid	3.109804	55.804166
8	Linear SVR	valid	4.329728	68.213462
10	Bagging Regressor	valid	6.939311	112.036826

Рисунок 3.19 – Порівняльна таблиця точності моделей

Проаналізувавши отриманий результат видно, MLP Regressor показує найкращу якість на валідації: RMSE ≈ 0.284 та MAPE ≈ 4.98%. Це свідчить про дуже точний прогноз серед усіх протестованих моделей.

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 прогнозування витрат води на тестовому наборі даних за допомогою моделі MLP Regressor

Зелені точки на графіках, зображених на рисунку 3.20, відповідають реальним значенням ви вратоди, зібраним у тестовому наборі, тоді як червоні точки позначають прогноз, згенерований навченими моделями.

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 витрат води виявилася модель MLP Regressor (RMSE ≈ 0.284 та MAPE ≈ 4.98%.). Її застосування дозволяє досягнути збалансованого поєднання абсолютної точності та відносної надійності прогнозів.

3.4 Висновки

У третьому розділі представлено опис, реалізацію та порівняння низки

моделей для короткострокового прогнозування витрат води в суббасейні Прип’яті. Для цього були задіяні як алгоритми, спеціально розроблені для часових рядів (Facebook Prophet, AutoARIMA), так і загальні регресійні методи класичного машинного навчання (Linear Regression, KNeighbors Regressor, Support Vector Machines, Linear SVR, Random Forest Regressor, Bagging Regressor, XGB Regressor і MLP Regressor).

Під час валідації виявилося, що лідерські позиції займають підходи, заточені під часові ряди, зокрема MLP Regressor, який досяг найменшої середньої відносної помилки (MAPE ≈ 4.98%) і показав послідовно високі результати на тестовій вибірці. У свою чергу класичні ML-моделі продемонстрували прийнятну, але менш точну продуктивність, що підкреслює їхні обмеження в задачах, де важливим є збереження часової структури даних.

Отже, на підставі аналізу показників точності прогнозів, графічного порівняння результатів та оцінки стабільності моделювання можна зробити висновок: для короткострокового прогнозування рівня води найбільш доцільно використовувати MLP Regressor. Ця модель забезпечує високу абсолютну й відносну точність, стабільно працює за умов сезонних коливань і водночас залишається прозорою з точки зору інтерпретації результатів, що є важливим для практичних гідрологічних застосувань.

ВИСНОВКИ

У рамках виконання бакалаврської кваліфікаційної роботи водогосподарські ділянки суббасейну річки Прип’ять були розглянуті як складна просторово-часова система з багатьма взаємопов’язаними параметрами, серед яких обсяги водозабору, природний стік, втрати, екологічний стік, дефіцит і запаси. Такий підхід дозволяє сприймати водогосподарський баланс не як фіксований набір показників, а як динамічну модель, котра змінюється у просторі та часі і вимагає багатофакторного аналізу для виявлення ключових закономірностей і критичних зон.

Було проведено дослідження питань системного управління водними ресурсами в суббасейні Прип’яті, зокрема підкреслено складність урахування просторово-часової мінливості дефіциту води та гідрологічної забезпеченості. Проаналізовано існуючі методи вивчення водогосподарських балансів, визначено їхні обмеження щодо інтегрованої візуалізації, динамічного аналізу та автоматизованого прогнозування на основі реальних гідрологічних спостережень.

Для реалізації поставлених завдань здійснено збір і підготовку гідрологічних даних: приведено їх до єдиного формату та структурували показники водогосподарського балансу за місяцями й окремими водогосподарськими ділянками. На базі обробленого масиву даних створено інструменти для відображення дефіциту та запасів води з урахуванням показника гідрологічної забезпеченості. Проведений системний аналіз дав змогу виявити закономірності водного дефіциту та відокремити найбільш критичні ділянки з недостатніми запасами води, що сприяє прийняттю обґрунтованих управлінських рішень на рівні конкретних водогосподарських ділянок у межах суббасейну.

Для прогнозування рівня води протестовано кілька моделей машинного навчання, серед яких Prophet, ARIMA, Random Forest, XGB та MLP Regressor. Найвищу точність показала модель MLP Regressor, яка досягла RMSE приблизно 0,284 і MAPE близько 4,98%, що свідчить про її здатність коректно враховувати сезонні коливання рівня води.

Узагальнення отриманих результатів дозволило виокремити просторові та

часові закономірності дефіциту водних ресурсів у суббасейні Прип’яті, зумовлені нерівномірністю надходження води протягом року, інтенсивністю заборів і варіаціями гідрологічної забезпеченості. Запропонований підхід може стати основою для розробки системи підтримки прийняття рішень у сфері управління водними ресурсами, зокрема для балансування дефіциту, виявлення зон підвищеного ризику та прогнозного планування у водогосподарській практиці.

За результатами даної роботи була зроблена доповідь на тему «Картографічна візуалізація результатів розрахунку балансів по водогосподарським ділянкам суббасейну Прип’яті» на Міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» з публікацією тез [1].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Наюк А. О., Крижановський Є. М., Струтинська М. В. 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/25649/21176

2. План управління річковим басейном Дніпра. Суббасейн річки Прип’ять. URL: https://davr.gov.ua/fls18/prypiat_summary_23072020.pdf

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

21. Водогосподарський баланс для суббасейну річки Прип'ять району басейну річки Дніпро. URL: https://www.davr.gov.ua/fls18/prypjat.pdf.

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

23. Prypyat. URL: https://www.kaggle.com/datasets/sttrutynska/prypyat.

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

25. Kaggle Notebook «Water discharge – EDA». URL: https://www.kaggle.com/code/vbmokin/water-discharge-eda

26. Kaggle Notebook «Water discharge – EDA & forecasting». URL: https://www.kaggle.com/code/vbmokin/water-discharge-eda-forecasting

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

28. 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.

29. 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.

30. 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 Технічне завдання

(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 СУББАСЕЙНУ ПРИП’ЯТІ

08-34.БКР.009.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Євгеній КРИЖАНОВСЬКИЙ
«__» _____ 2025 р.
Розробила студентка гр. СА-21б
_____ Марина СТРУТИНСЬКА
«__» _____ 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 суббасейну Прип'яті, 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 суббасейну річки Прип'ять, 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 суббасейну Прип'яті 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 суббасейну Прип'яті для виявлення та візуалізації ділянок із дефіцитом водних ресурсів та прогнозування рівня води в річці Прип'ять 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 _____ Марина СТРУТИНСЬКА

Додаток Б
Протокол перевірки кваліфікаційної роботи
(обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Системний аналіз водогосподарських балансів ділянок суббасейну Прип'яті»

Тип роботи: бакалаврська кваліфікаційна робота

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

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 5,18%

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 _____ Марина СТРУТИНСЬКА
(підпис)

Додаток В

Фрагмент лістингу програми

(довідниковий)

```
# 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

Ілюстративна частина

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

СИСТЕМНИЙ АНАЛІЗ ВОДОГОСПОДАРСЬКИХ БАЛАНСІВ ДІЛЯНОК СУББАСЕЙНУ ПРИП’ЯТІ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

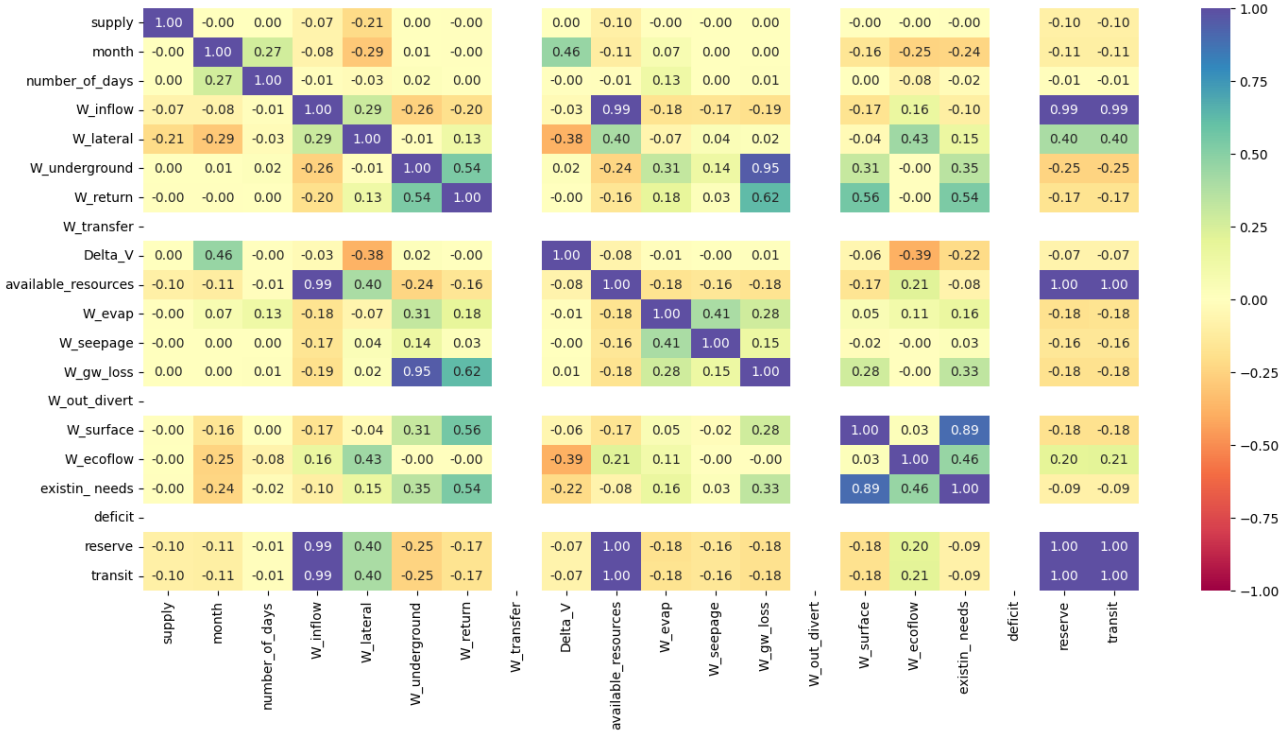


Рисунок Г.1 – Кореляційна матриця

Рисунок Г.2 – Тематична карта резервів води по водогосподарських ділянках

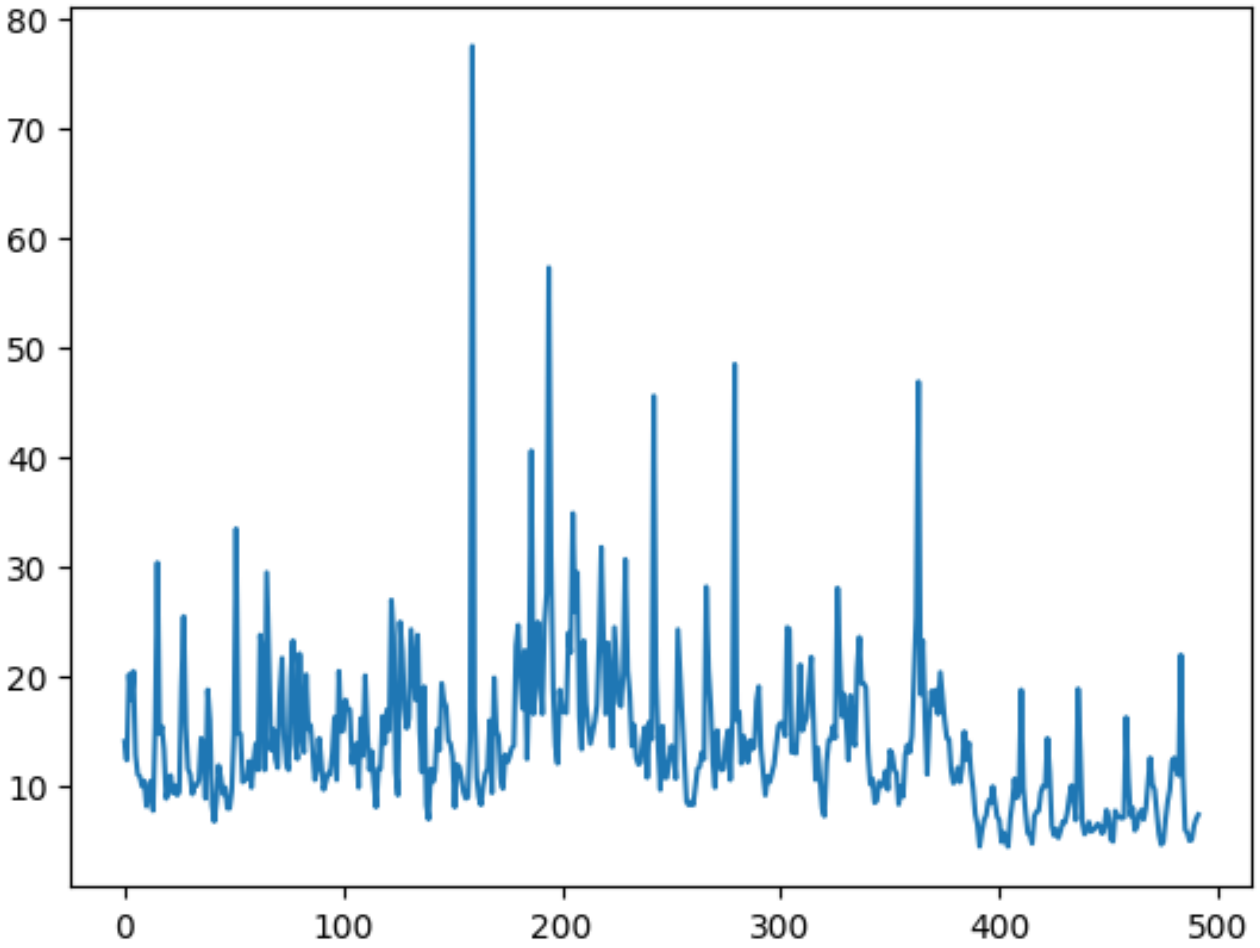


Рисунок Г.3 – Графік середніх добових рівнів води на гідропості Нетішин

	name_model	type_data	rmse	mape
12	MLP Regressor	valid	0.284465	4.979669
9	Random Forest Regressor	valid	0.379591	6.790386
7	Support Vector Machines	valid	0.511406	7.280615
6	KNeighbors Regressor	valid	0.434058	7.750227
11	XGB Regressor	valid	0.663275	8.452855
5	Linear Regression	valid	1.043145	14.015372
3	Prophet_180_days_12_order	valid	1.285184	21.0554
2	Prophet_180_days_3_order	valid	1.650448	28.496155
1	Prophet_12_days_12_order	valid	2.068762	37.192197
0	Prophet_12_days_3_order	valid	2.100708	37.465432
4	ARIMA_auto	valid	3.109804	55.804166
8	Linear SVR	valid	4.329728	68.213462
10	Bagging Regressor	valid	6.939311	112.036826

Рисунок Г.4 – Порівняльна таблиця точності моделей

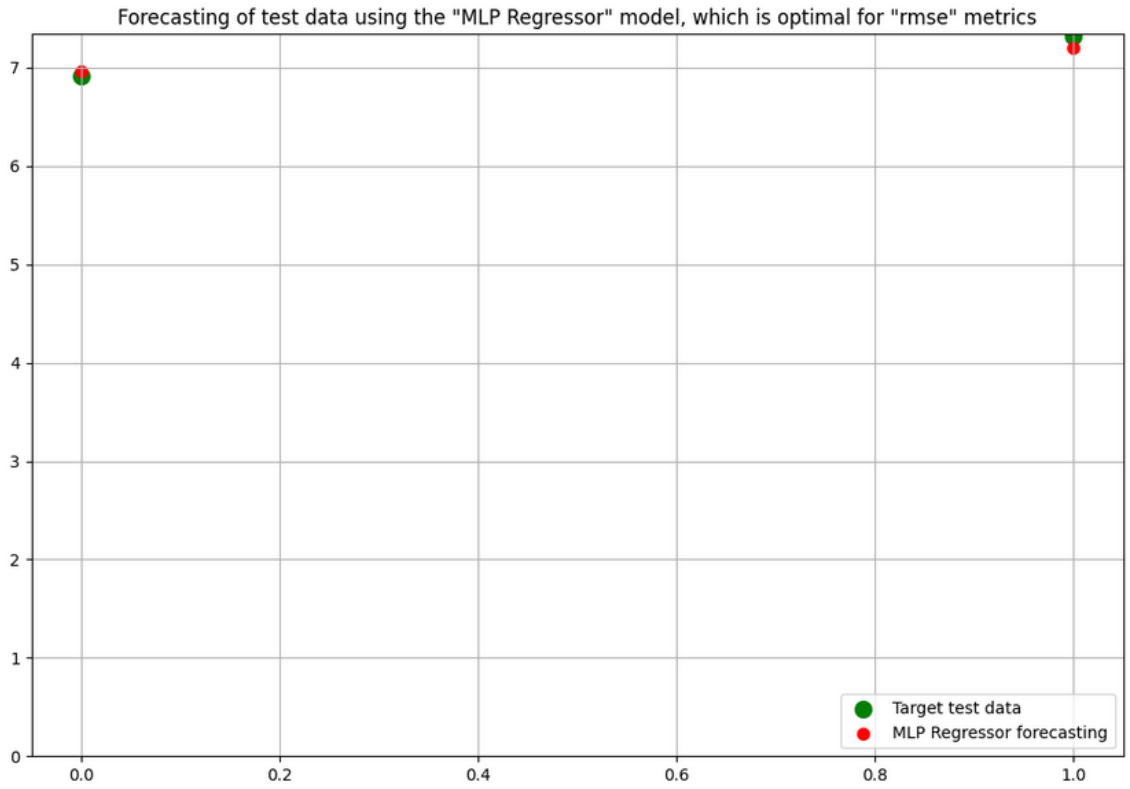


Рисунок Г.5 – Результати прогнозування рівня води на тестовому наборі даних за допомогою моделі MLP Regressor