Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних технологій
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи
другий (магістерський)
(рівень вищої освіти)

на тему: **Математичне та програмне забезпечення системи проведення гідрологічних прогнозів водних об'єктів**

Виконав: студент VI курсу, групи КН-61м
спеціальності
122 – "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)
Штипук О. В.
(прізвище та ініціали)

Керівник Пірко І. Б.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Львів – 2022 р.

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну
Кафедра інформаційних технологій
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Крошний І. М.
"____" _____2022 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Штипук Остап Васильович
(прізвище, ім'я, по батькові)

1. Тема роботи **Математичне та програмне забезпечення системи проведення гідрологічних прогнозів водних об'єктів**

керівник роботи Пірко І. Б., канд. фіз.-мат. наук, доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 13.12. 2021 року № С-617
2. Термін подання студентом роботи 12. 12. 2022 р.
3. Вихідні дані до роботи:

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
4.1. Стан проблемної області.
4.2. Інформаційне забезпечення.
4.3. Математичне забезпечення.
4.4. Програмне забезпечення.
4.5. Розроблення стартап проекту.
4.6. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація у Microsoft PowerPoint.

6. Дата видачі завдання 20 грудня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних даних та інших джерел згідно досліджуваної теми	20.12-30.01.2022	
2	Аналіз досліджуваної теми та вибір відповідних варіантів її розробки	01.02-27.02. 2022	
3	Постановка задачі та її формалізація	01.03-01.04. 2022	
4	Вибір та обґрунтування методів і засобів проведення дослідження	02.04-30.04. 2022	
5	Розроблення концептуальної схеми реалізації завдання	01.05-01.06. 2022	
6	Програмна реалізація завдання	02.06-30.10. 2022	
7	Тестування програмного продукту та отриманих результатів	01.11-15.11. 2022	
8	Розробка пояснювальної записки магістерської роботи	16.11-30.11. 2022	
9	Корегування пояснювальної записки згідно вимог, розроблення презентації	01.12-11.12. 2022	

Студент _____ Штипук О. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Пірко І. Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Пояснююча записка складається з 81 сторінок, 24 рисунків, 2 додатків та 4 таблиць, 20 літературних джерел.

В дипломній роботі проаналізовано існуючі гідрометеорологічні системи прогнозування, запропоновано підхід до прогнозування гідрологічних часових рядів. На їх основі отримано прогноз підвищеного рівня води, який можна порівняти зі значеннями історичних даних по досліджуваних водних об’єктах. Крім цього, за допомогою запропонованого методу можливе отримання прогнозу для окремих населених пунктів. Розроблений метод прогнозування може бути використаний для робіт із запобігання наслідкам затоплень населених пунктів у період повені.

Ключові слова: гідрологічний прогноз, Python, Pandas, Seaborn.

SUMMARY

The explanatory note consists of 81 pages, 24 figures, 2 attachments and 4 tables.

The thesis analyzed existing hydrometeorological forecasting systems, proposed an approach to forecasting hydrological time series. Based on them, a forecast of the increased water level was obtained, which can be compared with the values of historical data on rivers. In addition, with the help of the proposed method, it is possible to obtain an individual forecast for individual elements. The developed forecasting method can be used for works to prevent the consequences of seasonal flooding of settlements during the flood period.

Keywords: hydrological forecast, Python, Pandas, Seaborn.

ПЕРЕЛІК СКОРОЧЕНЬ І СПЕЦІАЛЬНИХ ТЕРМІНІВ

- БД – база даних;
- ООП – об’єктно-орієнтоване програмування;
- ЛКМ – ліва кнопка миші;
- ПКМ – права кнопка миші;
- ШІ – штучний інтелект;
- ML – Mashine Learning.

ЗМІСТ

ВСТУП

Актуальність дипломної роботи

В останні роки спостерігаються помітні зміни водності рік, які пов’язують із глобальним потеплінням та збільшенням на них антропогенного навантаження. Однією з найважливіших гідрологічних величин є стік водних об’єктів та його просторово-часові коливання. Він визначає розвиток та функціонування водних та навколоводних екосистем, має істотне значення при вирішенні водогосподарських завдань (об’єми водоспоживання, скидання стічних вод). Наукове та практичне значення має виявлення закономірностей просторово-часових коливань річкових стоків в сучасних умовах, адаптація існуючих та розробка нових методик їх визначення.

Предметом дослідження дипломної роботи є розробка інформаційної системи для прогнозування стану водних об’єктів.

Об’єктом дослідження є гідрологічне прогнозування стану водних об’єктів.

Мета роботи – розробка та реалізація системи для аналізу стану водних об’єктів та можливості його прогнозування.

Відповідно до мети дослідження поставлено наступні **завдання**:

- провести огляд літератури по даній тематиці, проаналізувати існуючі гідрометеорологічні системи прогнозування;
- розробити математичне забезпечення та алгоритм функціонування системи;
- реалізувати програмну модель системи аналізу стану водних об’єктів;
- провести з допомогою даного додатку дослідження динаміки підняття рівня води у водному об’єкті.

Наукова новизна одержаних результатів

Наукова обгрунтованість положень та висновків підтверджується використанням великого об’єму даних спостережень на гідрологічних постах та використанням сучасних методів аналізу гідрологічних часових рядів. Наукова новизна роботи полягає у виявленні просторово-часових закономірностей коливань мінімальних витрат води рік. Отримані результати мають практичне застосування для

раціонального використання природних ресурсів і можуть бути науковою основою під час виконання проектів у галузі гідрологічних та екологічних досліджень. Значення витрат річкового стоку можуть бути використані при проектуванні гідротехнічних споруд та меліоративних систем.

Практичне значення одержаних результатів

Розроблено інформаційну систему, яка представляє модель машинного навчання для дослідження гідрологічних явищ. Дана система і дослідження з її допомогою доможуть всім, хто цікавиться гідрологією річкових басейнів, отримати знання про базовий аналіз водозбору досліджуваної річкової системи. Результати роботи цієї моделі можна використати при дослідженні та прогнозуванні зон підтоплення прирічкових територій. Дана інформаційна система може бути впроваджена в структурні підрозділи гідрометеорологічних постів.

РОЗДІЛ І. АНАЛІЗ СТАНУ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Спостереження за станом водних об’єктів та їх використання службою прогнозів

Накопичені знання про кругообіг і перетворення води в природі, про закономірності існування річок, озер, про живлення їх атмосферними опадами в різних умовах клімату, рельєфу, рослинного та ґрунтового покриву систематизувалися та оформилися в науку гідрологію.

Гідрологія покликана вирішувати практичні завдання, що виникають у процесі розвитку та господарської діяльності суспільства. Однією з найважливіших сторін практичного застосування гідрології є гідрологічні прогнози. Слово "прогноз" складається з двох грецьких слів: "про", що означає вперед, і "гнозис", що означає знання. Отже, слово «прогноз» означає завчасне знання, тобто передбачення розвитку явищ чи подій. Під гідрологічними прогнозами розуміють науково обґрунтоване передбачення гідрологічних явищ, тобто тих природних явищ, які виникають і змінюють одне одного на річках і озерах у процесі сезонних та інших змін погоди.

Всі гідрологічні явища на річках тісно пов’язані з кліматом та умовами погоди на території їх басейнів, характером рельєфу, рослинним та ґрунтовим покривом басейнів, тобто з географічним середовищем. Складність гідрологічних явищ і складність постановки точних і детальних спостережень на великих просторах природних водозборів є головною причиною того, що гідрологія як наука сформувалася порівняно недавно і не має ще можливості розраховувати ці явища з великою точністю. Це наклало свій відбиток на сучасне розуміння терміну «гідрологічний прогноз» і на практичні прийоми прогнозів.

Гідрологічний прогноз не є точним передбаченням і містить у собі елемент науково обгрунтованої ймовірності. Тим не менш сучасна гідрологія може вже з певною завчасністю передбачати, якою буде повінь на річках наступної весни, коли відбудеться замерзання річок і ряд інших явищ.

Довгострокові гідрологічні прогнози необхідні для найбільш раціонального регулювання стоку річок, у плануванні вироблення електроенергії, для роботи водного транспорту, водопостачання. Гідрологічні прогнози мають велике значення також для боротьби з небезпечними та несприятливими явищами на річках.

Розливи річок викликають повені, при яких затоплюються міста та села, руйнуються будівлі, гинуть посіви. Повені спричиняють величезні збитки, а іноді й людські жертви. А трапляється і так, що незвичайне маловоддя на річках призводить до їх обмілення, ускладнює судноплавство, зменшує вироблення електроенергії на гідроелектростанціях, а в районах штучного зрошення спричиняє загибель посівів на великих площах.

На території нашої країни майже щорічно в якомусь із її районів спостерігаються великі розливи річок, що загрожують народному господарству, а також інші несприятливі гідрологічні явища. Своєчасне попередження народногосподарських організацій про виникнення таких явищ дозволяє вжити відповідних заходів щодо запобігання або зменшення можливої шкоди.

1.2. Основи розробки та оцінки гідрологічних прогнозів

Гідрологічний прогноз – це завчасний розрахунок елемента, який нас цікавить чи явища гідрологічного режиму в конкретних фізико-географічних умовах, які основані на знанні закономірностей природних процесів, що визначають це явище. Таким чином, кожен гідрологічний прогноз повинен мати ту чи іншу завчасність, під якою розуміється проміжок часу від моменту складання прогнозу до дати настання або закінчення передбачуваного явища.

Всі зміни, які спостерігають у стані річок та інших водних об’єктів, обумовлені кліматичними особливостями даного басейну, а в кожний конкретний момент є наслідком поточних умов погоди, тобто метеорологічних факторів.

В основу класифікації існуючих гідрологічних прогнозів можуть бути покладені три основні ознаки:

- завчасність прогнозів;
- передбачувані явища та елементи режиму;
- цільове призначення прогнозів.

Кожна з цих ознак дозволяє розділити прогнози на кілька видів. За ознакою завчасності гідрологічні прогнози поділяються на довгострокові та короткострокові. Умовно до категорії короткострокових прогнозів прийнято відносити прогнози,

завчасність яких не перевищує 10-15 діб. Завчасність довгострокових гідрологічних прогнозів зазвичай від 1-2 до 6-8 місяців.

За цільовим призначенням всі гідрологічні прогнози можна поділити на прогнози загального користування та спеціалізовані прогнози для різних галузей народного господарства.

До першої групи відносяться прогнози, що становлять спільний інтерес, наприклад прогнози максимального рівня повені та паводків та попередження про повені. Спеціалізовані прогнози враховують специфіку вимог таких галузей народного господарства, як гідроенергетика, водний транспорт.

В основі розробки методів гідрологічних прогнозів лежить фізичний аналіз природних процесів, кінцева мета якого полягає у встановленні кількісних взаємозв'язків, що дозволяють розраховувати із заданою завчасно цікавий нас елемент режиму стосовно конкретних річкових басейнів або ділянок річок.

Під методикою прогнозу розуміють розрахункові прийоми або побудови, розроблені на основі того чи іншого методу стосовно конкретних водних об'єктів або деякої території. В якості методики прогнозу може бути аналітичний розв'язок задачі, так і наближені способи розрахунку з використанням в обох випадках емпірично встановлених параметрів або суто кореляційні зв'язки.

З точки зору можливостей прогнозів всі фактори, що визначають стік та інші гідрологічні явища, можна поділити на дві категорії:

- відомі початкові фактори, які визначають умови, що вже склалися на момент випуску прогнозів, і можуть бути з тим чи іншим ступенем точності оцінені за даними гідрометеорологічних спостережень;
- невідомі фактори майбутнього, вплив яких на прогнозоване явище позначається після випуску прогнозу.

До останніх належать переважно майбутні метеорологічні умови. За наявності відповідних метеорологічних прогнозів вони можуть бути безпосередньо введені в гідрологічний прогноз. Проте сучасна метеорологія не має в своєму розпорядженні поки що досить точних методів кількісного прогнозу погоди. З цієї причини практична можливість гідрологічних прогнозів визначається тим, наскільки велика роль у

формуванні передбачуваних явищ майбутніх умов погоди та наскільки вони мінливі у часі. Чим більша роль цих умов, тим необхідніші кількісні метеорологічні прогнози.

Практично з метою гідрологічних прогнозів і попереджень у даний час можуть бути використані лише короткострокові прогнози температури повітря та опадів. Точність останніх навіть при дуже малій завчасності залишається ще низькою.

У практиці розробки методики гідрологічних прогнозів широко використовуються такі прийоми, як графічна побудова залежностей, кореляція, підбір емпіричних формул, побудова розрахункових графіків за заданими формулами.

1.3. Класифікація методів прогнозування

За характером використання вихідних даних методи короткострокових прогнозів на водних об’єктах можуть бути поділені на гідрометричні та гідрометеорологічні. Розглядаються такі методи:

- метод тенденції;
- метод відповідних рівнів;
- метод відповідних об’ємів;
- метод ізохронії.

Перші три методи базуються на використанні матеріалів гідрометричних спостережень у русловій мережі басейну, що характеризують закономірність руху паводкової хвилі. Метод ізохрон оснований на використанні не тільки гідрометричних даних, а й метеорологічних.

1.4. Гідрологічні часові ряди

При побудові моделей використовують два типи даних:

- дані, які характеризують сукупність різних об’єктів в певний момент часу;
- дані, які характеризують один об’єкт за ряд послідовних моментів часу.

Моделі, які побудовані по даних першого типу, називаються просторовими моделями. Моделі, побудовані на основі другого типу даних, називаються моделями часових рядів. Часовий ряд (ряд динаміки) – це сукупність значень якогось показника за декілька послідовних моментів чи періодів часу.

Це часовий ряд деяких показників y_1, y_2, …, y_t, де t – порядковий номер періоду, який аналізується. В залежності від того, чи відображають елементи часового ряду стан об’єкта за певний проміжок часу чи фіксують його в строго встановлені моменти, розрізняють інтервальні та моментні ряди.

Інтервальний часовий ряд – це сукупність показників, кожний з яких характеризує розвиток об’єкта дослідження за певний проміжок часу (рік, квартал, місяць, час доби і т.д.).

Моментний часовий ряд – це сукупність показників, які характеризують стан об’єкта на певну дату, наприклад на перше число кожного місяця, на 1 січня кожного року (якщо потрібно порівняти по роках, як і що змінюється).

Кожний рівень числового ряду формується під впливом великого числа факторів, які умовно можна поділити на три групи:

- фактори, які формують тенденцію ряду;
- фактори, які формують циклічні коливання ряду;
- випадкові фактори.

Реальні дані часто містять тенденцію, сезонні коливання, циклічні коливання та випадкову компоненту. Модель, в якій часовий ряд представлений як сума трендової, сезонної, циклічної та випадкової компонент, називається адитивною моделлю часового ряду.

Модель, в якій часовий ряд представлений як добуток перечислених компонент, називається мультиплікативною моделлю часового ряду.

Основне завдання дослідження часового ряду – виявлення та придання кількісного виразу кожній з перерахованих вище компонент, для того щоб використати отриману інформацію для прогнозування майбутніх значень ряду чи при побудові моделей взаємозв’язку двох чи більше часових рядів.

Мета аналізу часового ряду – опис характерних особливостей ряду, прогноз майбутніх значень на основі минулих спостережень, управління процесом, який породив часовий ряд.

При аналізі часового ряду прийнято виділяти чотири компоненти:

- тренд;
- циклічна компонента;

- сезонна компонента;
- випадкова величина.

Тренд – це компонента, яка плавно змінюється, вона описує чистий вплив тривалих факторів (ріст населення, зміна структури вікової групи і т.д.).

Циклічна компонента – це яка описує тривалі періоди відносного зростання та спаду, складається з циклів, які змінюються по амплітуді та тривалості.

Сезонна компонента складається з послідовності циклів, які майже повторюються (об’єм продаж перед новим роком, об’єм перевозок пасажирів міським транспортом).

Часовий ряд представляє собою суму цих компонент в адитивній моделі:

$$X_t = T_t + C_t + S_t + I_t. \quad (1.1)$$

або добуток в мультиплікативній моделі:

$$X_t = T_t \cdot C_t \cdot S_t \cdot I_t. \quad (1.2)$$

Для порівняння даних різних часових рядів обчислюють індекси I_t, які представляють собою виражені в процентах величини відліків. В загальному випадку вираз для обчислення індексів має вид:

$$I_t = \frac{X_t}{X_0} \cdot 100\ \%, \quad (1.3)$$

де X_0 – константа базового періоду, X_1, X_2, …, X_n – члени часового ряду.

Таким чином, індекси представляють собою значення, які виражені в процентах базового періоду.

Розглянемо автокореляцію рівнів часового ряду. Основне завдання аналізу часового ряду – виявлення та надання кількісного виразу трендовій, циклічній та випадковій компонентам з тим, щоб використати отриману інформацію для прогнозування майбутніх значень ряду чи при побудові моделей взаємозв’язку двох чи більше часових рядів. При наявності в часовому ряді тенденції та циклічних коливань значення кожного наступного рівня залежить від попередніх.

Кореляційну залежність між послідовними рівнями часового ряду називають автокореляцією рівнів ряду. Кількісно її можна виміряти з допомогою лінійного коефіцієнта кореляції між рівнями вихідного часового ряду та рівнями цього ряду, які будуть зсунуті на декілька кроків по часу. Вибіркове значення коефіцієнта кореляції випадкових величин X та Y визначається за формулою:

$$r_{xy} = \frac{\sum\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum\left(x_i - \bar{x}\right)^2 \sum\left(y_i - \bar{y}\right)^2}}. \quad (1.4)$$

Послідовність коефіцієнтів автокореляції рівнів першого, другого і т. д. порядків називають автокореляційною функцією часового ряду. Графік залежності її значень від величини порядку коефіцієнта автокореляції називають корелограмою.

Аналіз автокореляційної функції та графіку дозволяє виявити структуру часового ряду. Якщо найбільш високим виявився коефіцієнт автокореляції першого порядку, то досліджуваний ряд містить тільки тенденцію. Якщо найбільш високим виявився коефіцієнт автокореляції порядку k, то ряд містить циклічні коливання з періодичністю в k моментів часу.

ВИСНОВКИ ДО РОЗДІЛУ 1

В першому розділі проаналізовано предметну область, засоби та технології проектування інформаційної системи стану водних об’єктів. З допомогою цієї інформаційної системи можна буде моделювати стан водних обєктів та прогнозувати рівень підняття паводкових вод внаслідок несприятливих метеорологічних факторів.

РОЗДІЛ ІІ. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Проведення аналізу даних з допомогою бібліотеки Pandas

Pandas – це бібліотека з відкритим кодом на Python. Вона надає готові до використання структури даних та інструменти аналізу даних. Модуль Pandas працює поверх бібліотеки NumPy і широко використовується для обробки та аналізу даних.

NumPy – це структура даних, яка підтримує багатовимірні масиви та широкий спектр математичних операцій із масивами. Pandas має інтерфейс вищого рівня. Він також забезпечує оптимізоване узгодження табличних даних та потужну функціональність часових рядів.

DataFrame – це ключова структура даних у Pandas. Вона дозволяє зберігати та обробляти табличні дані як двовимірну структуру даних. Pandas надає багатий набір функцій для DataFrame. Наприклад, вирівнювання даних, статистика даних, об’єднання даних тощо.

Для установки бібліотеки для аналізу даних потрібно ввести:

pip install pandas

Щоб імпортувати Pandas та NumPy в скрипт Python, потрібно добавити такий фрагмент коду:

import numpy as np

import pandas as pd

Модуль Pandas надає три структури даних.

Series – це одновимірний масив постійного розміру, подібний до структури, що має однорідні дані.

DataFrames – це двомірна таблична структура зі змінним розміром та неоднорідно типізованими стовпцями.

Panel – це тривимірний масив зі змінним розміром.

DataFrame – це найважливіша структура даних, яка широко використовується, а також стандартний спосіб зберігання даних. Вона містить дані, які вирівняні по рядках і стовпцях, як у таблиці SQL або базі даних електронної таблиці. Можна зберігати дані в DataFrame або імпортувати файли CSV, файли Excel, таблиці SQL і т.д.

Для створення об’єкта DataFrame використовують фрагмент коду:

pandas.DataFrame(data, index, columns, dtype, copy)

data – створити об’єкт DataFrame із вхідних даних. Це може бути список, dict, series, Numpy arrays або будь-який інший DataFrame; index – має мітки рядків; columns – використовуються для створення підписів стовпців; dtype – використовується для вказівки типу даних кожного стовпця, необов’язковий параметр; copy – використовується для копіювання даних. Є багато способів для створення DataFrame. Можна створити об’єкт із словників або списку словників. Можна також створити його зі списку кортежів, CSV, Excel файлів і т.д.

Нижче приведено код для створення DataFrame зі списку словників:

```
import pandas as pd
import numpy as np
df = pd.DataFrame({"State": ['Andhra Pradesh', 'Maharashtra', 'Karnataka', 'Kerala', 'Tamil Nadu'],
                   "Capital": ['Hyderabad', 'Mumbai', 'Bengaluru','Trivandrum', 'Chennai'],
                   "Literacy %": [89, 77, 82, 97,85],
                   "Avg High Temp(c)": [33, 30, 29, 31, 32 ]
})
print(df)
```

Отримаємо такий результат:

	State	Capital	Literacy %	Avg High Temp(c)
0	Andhra Pradesh	Hyderabad	89	33
1	Maharashtra	Mumbai	77	30
2	Karnataka	Bengaluru	82	29
3	Kerala	Trivandrum	97	31
4	Tamil Nadu	Chennai	85	32

Перший крок – створити словник. Другий крок – передати в словник як аргумент метод DataFrame(). Останній крок – отримати DataFrame. DataFrame можна порівняти з таблицею, що має неоднорідне значення. Також можна змінити розмір. Дані були надані у вигляді карти, і ключі карти розглядаються в Pandas як мітки рядків. Індекс відображається в крайньому лівому стовпці і має позначки рядків. Заголовок стовпця та дані відображаються у вигляді таблиці. Також можна створювати індексовані DataFrames. Це можна зробити, налаштувавши параметр індексу. Можна створити DataFrame, імпортувавши файл CSV. CSV – це текстовий файл з одним записом даних у кожному рядку. Значення запису розділяються між собою символом коми. Pandas надає метод read_csv() для читання CSV-файлу. Наприклад, можна створити файл з

іменем cities.csv, який містить інформацію про міста. Файл CSV зберігається в тому ж каталозі, що і сценарії Python. Цей файл можна імпортувати за допомогою такого коду:

```
data = pd.read_csv('cities.csv')
print(data)
```

2.2. Стандартні метрики в задачах класифікації

У завданнях машинного навчання з метою оцінки якості моделей і порівняння різних алгоритмів використовуються метрики, а їх вибір і аналіз – головне завдання. Для цього потрібно розглянути деякі критерії якості у задачах класифікації, обговорити, що є важливим при виборі метрики.

Accuracy, precision, recall

Перед переходом до самих метрик необхідно ввести важливу концепцію для опису цих метрик у термінах класифікації помилок – confusion matrix (матриця помилок). Припустимо, що є два класи та алгоритм, що передбачає приналежність кожного об’єкта до одного з класів, тоді матриця помилок класифікації буде виглядати так:

	y=1	y=0
$y=1$	True Positive (TP)	False Positive (FP)
$y=0$	False Negative (FN)	True Negative (TN)

Тут y – це відповідь алгоритму на об’єкті, y – мітка класу на цьому об’єкті. Таким чином, похибки класифікації бувають двох видів: False Negative (FN) та False Positive (FP).

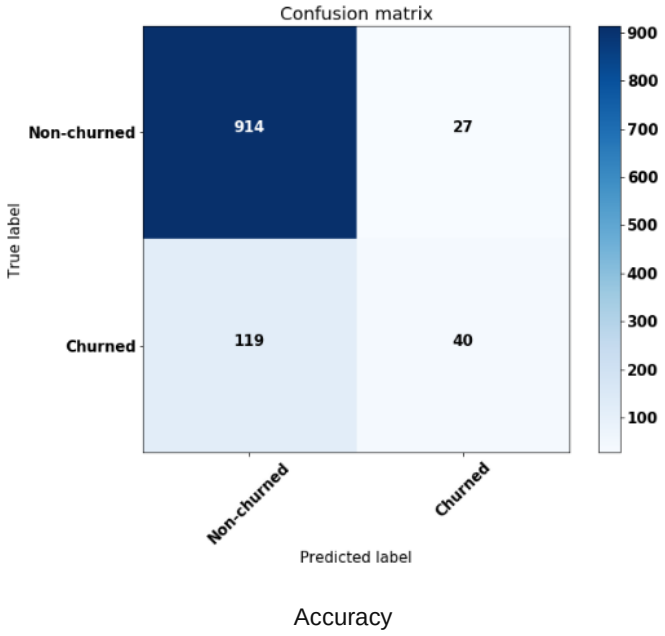


Рис. 2.1.

Інтуїтивно зрозумілою, очевидною метрикою є accuracy – частка правильних відповідей алгоритму:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{2.1}$$

Ця метрика не використовується у завданнях з нерівними класами. Нехай потрібно оцінити роботу спам-фільтра пошти. Є 100 не-спам листів, 90 з яких класифікатор визначив правильно (True Negative = 90, False Positive = 10), і 10 спам-листів, 5 з яких класифікатор також визначив правильно (True Positive = 5, False Negative = 5). Тоді accuracy:

$$\text{accuracy} = \frac{0 + 100}{0 + 100 + 0 + 10} = 90,9$$

При цьому модель абсолютно не має ніякої передбачальної силою, оскільки спочатку потрібно було визначати листи зі спамом. Подолати це допоможе перехід із загальної для всіх класів метрики до окремих показників якості класів.

Precision, recall та F-міра

Для оцінки якості роботи алгоритму кожному з класів окремо вводять метрики precision (точність) і recall (повнота).

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2.2}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2.3}$$

Precision можна інтерпретувати як частку об’єктів, названих класифікатором позитивними і при цьому вони є дійсно позитивними, а recall показує, яку частку об’єктів позитивного класу з усіх об’єктів позитивного класу знайшов алгоритм.

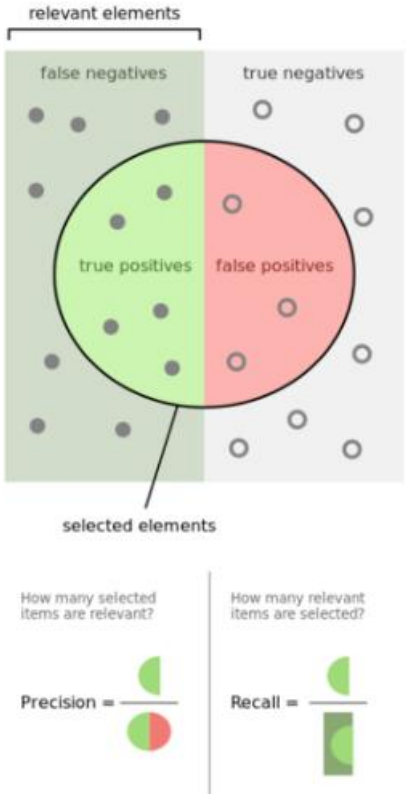


Рис. 2.2.

Саме введення precision не дозволяє записувати всі об’єкти в один клас, тому що в цьому випадку ми отримаємо зростання рівня False Positive. Recall демонструє здатність алгоритму виявляти цей клас взагалі, а precision – здатність відрізняти цей клас від інших класів.

Помилки класифікації бувають двох видів: False Positive та False Negative. У статистиці перший вид помилок називають помилкою першого роду, а другий — помилкою другого роду. У задачі з визначення відтоку абонентів помилкою першого роду буде прийняття лояльного абонента за того, що вийде, тому що нульова гіпотеза полягає в тому, що ніхто з абонентів не йде, а цю гіпотезу відкидають. Відповідно помилкою другого роду буде пропуск пропускаючого абонента і помилкове прийняття нульової гіпотези.

Precision і recall не залежить, на відміну accuracy, від співвідношення класів і тому застосовуються за умов незбалансованих вибірок. Часто у реальній практиці стоїть

завдання знайти оптимальний баланс між цими двома метриками. Класичним прикладом є завдання визначення відтоку клієнтів.

Очевидно, що ми можна знаходити всіх клієнтів, що виходять, і тільки їх. Але, визначивши стратегію та ресурс для утримання клієнтів, можна підібрати потрібні пороги по precision та recall. Наприклад, можна зосередитися на утриманні лише високоприбуткових клієнтів або тих, хто вийде з більшою ймовірністю, оскільки є обмеження в ресурсах колл-центру.

Зазвичай при оптимізації гіперпараметрів алгоритму використовується одна метрика, поліпшення якої можна побачити на тестовій вибірці. Існує кілька різних способів об'єднати precision і recall в агрегований критерій якості. F-міра (F_β) – середнє гармонійне precision і recall:

$$F_\beta = \left(1+\beta^2\right)\cdot\frac{\text{precision}\cdot\text{recall}}{\left(\beta^2\text{precision}\right)+\text{recall}} \tag{2.3}$$

AUC-ROC и AUC-PR

При конвертації дійсної відповіді алгоритму в бінарну мітку потрібно вибрати якийсь поріг, при якому 0 стає 1. Природнім і близьким є порог, рівним 0.5, але він не завжди є оптимальним.

Одним із способів оцінити модель в цілому, не привязуючись до конкретного порогу, є AUC-ROC (чи ROC AUC) – площа (Area Under Curve) під кривою похибок (Receiver Operating Characteristic curve). Дана крива представляє собою лінію від (0,0) до (1,1) в координатах True Positive Rate (TPR) та False Positive Rate (FPR):

$$\text{TPR} = \frac{\text{TP}}{\text{TP}+\text{FN}}. \tag{2.4}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP}+\text{TN}}. \tag{2.5}$$

TPR це повнота, а FPR показує, яку частину з об'єктів negative класу алгоритм передбачив невірно. В ідеальному випадку, коли класифікатор не робить помилок (FPR = 0, TPR = 1), отрмують площу під кривою, яка рівна одиниці; в іншому випадку, коли класифікатор випадково видає ймовірності класів, AUC-ROC буде прямувати к 0.5, так як класифікатор буде видавати однакову кількість TP та FP.

Кожна точка на графіку відповідає вибору деякого порогу. Площа під кривою в даному випадку показує якість алгоритму (більше – краще), крім цього, важливою характеристикою являється крутизна самої кривої – потрібно максимувати TPR, мінімізуючи FPR, а значить, крива в ідеальному випадку повинна прямувати до точки (0,1).

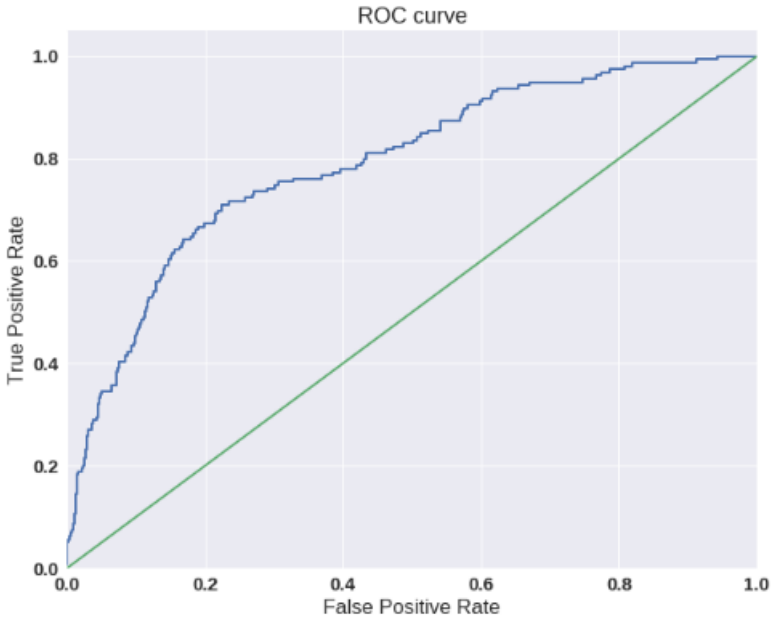


Рис. 2.3.

Критерій AUC-ROC стійкий до незбалансованих класів і може бути інтерпретований як ймовірність того, що випадково обраний positive об’єкт буде проранжований класифікатором вище (буде мати більш високу ймовірність бути positive), ніж випадково обраний negative об’єкт.

Precision та recall також використовують для побудови кривої і, аналогічно AUC-ROC, знаходять площу під нею.

Logistic Loss

Логістична функція втрат визначається як:

$$\text{logloss} = -\frac{1}{l} \cdot \left(\sum_{i=1}^{l} y_i \cdot \log y_i \right) + (1 - y_i) \cdot \log\left(1 - y_i\right)) \qquad (2.6)$$

Тут y_i – це відповідь алгоритму на і-ом обєкті, y – істинна мітка класу на і-ом об’єкті, l – розмір вибірки. Можна представити мінімізацію logloss як завдання максимізації accuracy шляхом штрафу за невірні передбачення. Однак слід зауважити, що logloss досить сильно штрафує за впевненість класифікатора в невірній відповіді.

2.3. Графічне представлення даних з допомогою бібліотеки Seaborn

Візуалізація даних – це метод, який дозволяє перетворювати дані на діаграми та графіки, які несуть цінну інформацію. Діаграми зменшують складність даних та роблять їх більш зрозумілими. Існує багато інструментів для візуалізації даних, таких як Tableau, Power BI, ChartBlocks та інших. Однак для роботи з даними Python підійде найкраще.

В даній роботі використовується бібліотека Seaborn. Вона така ж потужна, як і Matplotlib, але надає велику абстракцію для спрощення графіків та привносить деякі унікальні функції. Seaborn – це бібліотека для створення статистичних графіків на Python. Вона ґрунтується на Matplotlib і тісно взаємодіє зі структурами даних Pandas.

Архітектура Seaborn дозволяє швидко вивчити та зрозуміти дані. Seaborn захоплює цілі фрейми даних або масиви, в яких містяться всі дані, і виконує всі внутрішні функції, потрібні для мапінгу та статистичної агрегації для перетворення даних на інформативні графіки. Для установки цієї бібліотеки в Python потрібно ввести: pip install seaborn.

Перед побудовою графіків потрібно імпортувати такі бібліотеки:

```
import numpy as np
import pandas as pd
import matplotlib
import seaborn as sns
```

Перед тим як будувати графіки, будуть потрібні дані. Seaborn працює безпосередньо з об’єктами Dataframe з Pandas. Ця графічна бібліотека постачається з деякими вбудованими наборами даних, які можна використовувати прямо з коду, та не завантажувати файли вручну. Розглянемо процес побудови графіків в даній бібліотеці на наборі даних про рейси літаків.

```
flights_data = sns.load_dataset("flights")
flights_data.head()
```

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121

Це відбувається при виклику функції load_dataset, в яку потрібно ввести ім'я даних і отримують потрібний Dataframe. Розглянемо, як в бібліотеці Seaborn будується діаграма розсіювання Scatter Plot. Діаграма розсіювання – це діаграма, яка відображає точки на основі двох вимірювань набору даних.

sns.scatterplot(data=flights_data, x="year", y="passengers")

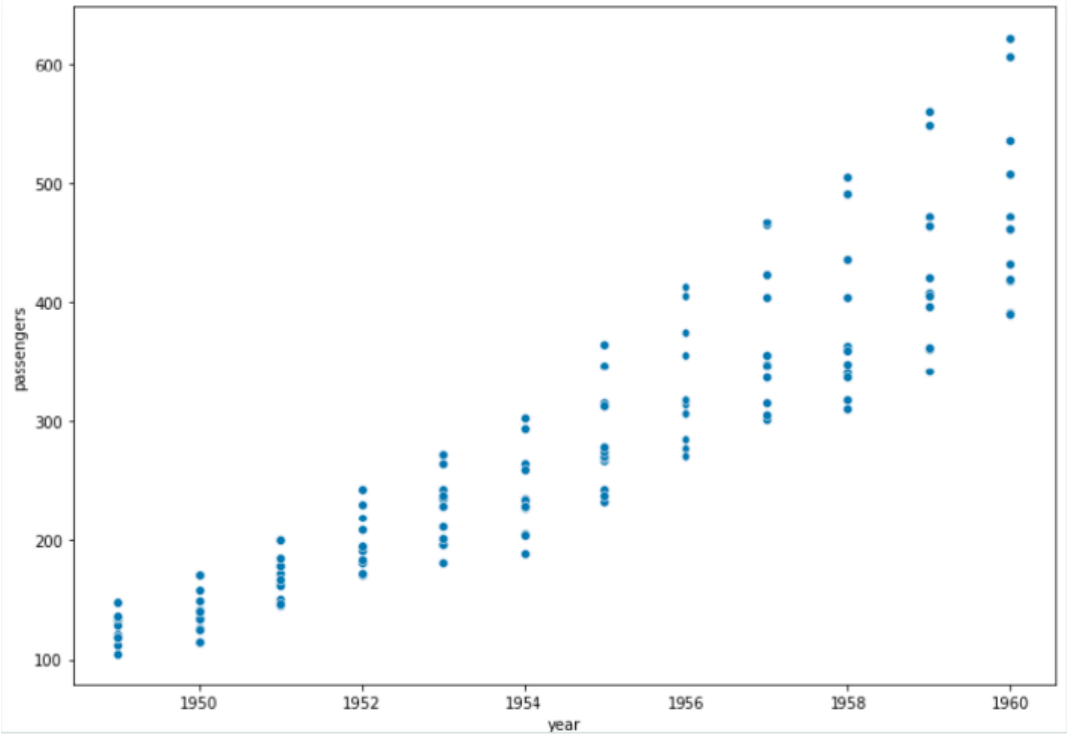


Рис. 2.4. Графік діаграми розсіювання.

Функція scatterplot приймає в себе набір даних, який потрібно візуалізувати, та стовпці, які будуть виступати як осі x та y.

Розглянемо створення лінійного графіка Line Plot за допомогою бібліотеки Seaborn. Цей графік будує лінію, який представляє собою безперервні чи категоріальні дані. Для цього використовують функцію lineplot з набором даних та стовпцями, що представляють осі x та y.

sns.lineplot(data=flights_data, x="year", y="passengers")

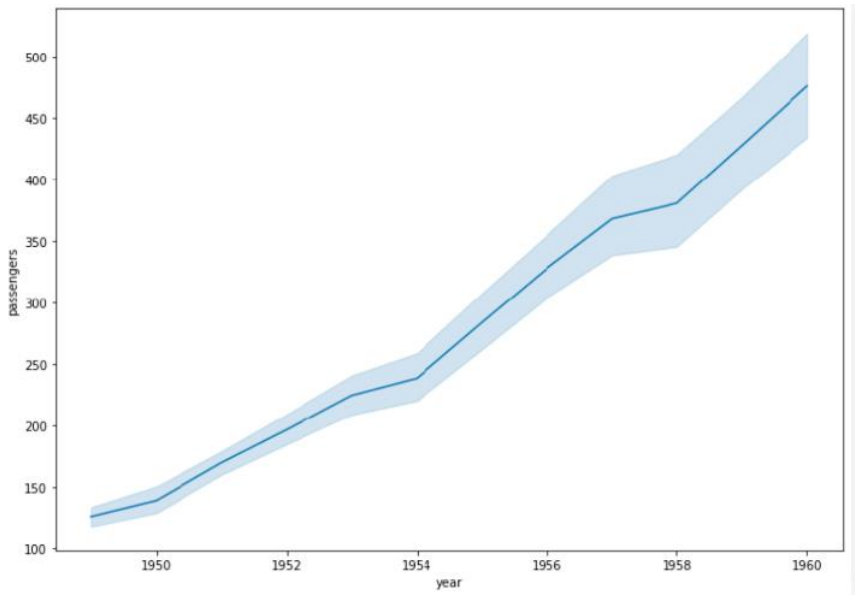


Рис. 2.5. Створення лінійного графіку.

Для створення стовпчикової діаграми в Seaborn використовують функцію barplot:
sns.barplot(data=flights_data, x="year", y="passengers")

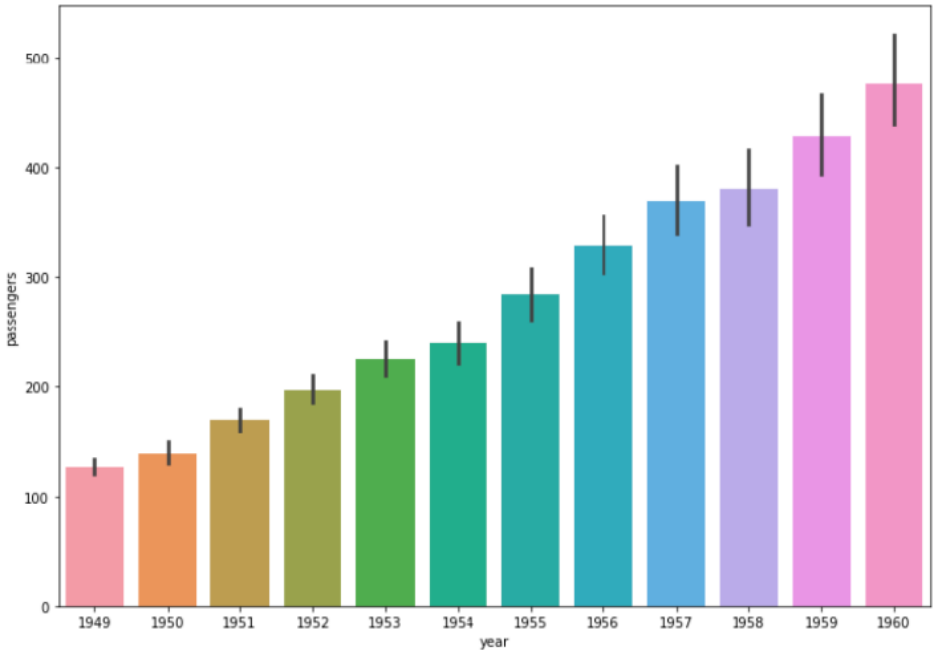


Рис. 2.6. Побудова стовпчикової діаграми.

Бібліотека Seaborn базується на Matplotlib, розширюючи її функціональні можливості. Для побудови кількох діаграм одночасно потрібно використати функцію subplot з бібліотеки Matplotlib:

```
diamonds_data = sns.load_dataset('diamonds')
plt.subplot(1, 2, 1)
sns.countplot(x='carat', data=diamonds_data)
```

```
plt.subplot(1, 2, 2)
sns.countplot(x='depth', data=diamonds_data)
```

За допомогою функції subplot на одному графіку можна побудувати кілька діаграм. Функція має три параметри: перший – кількість рядків, другий – кількість стовпців, третій – кількість діаграм.

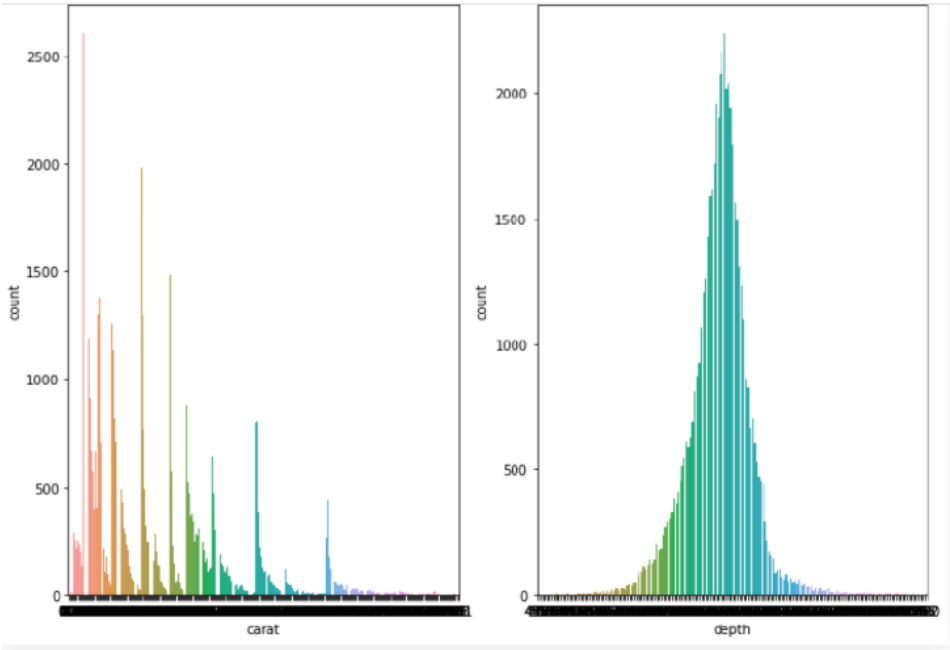


Рис. 2.7. Побудова двох графіків в одному вікні.

В бібліотеці Seaborn всі її функції будуються на структурі Dataframe. За візуалізацію даних з бібліотеки Pandas відповідає наступний фрагмент коду:

```
drinks_df = pd.read_csv("data/drinks.csv")
sns.barplot(x="country", y="beer_servings", data=drinks_df)
```

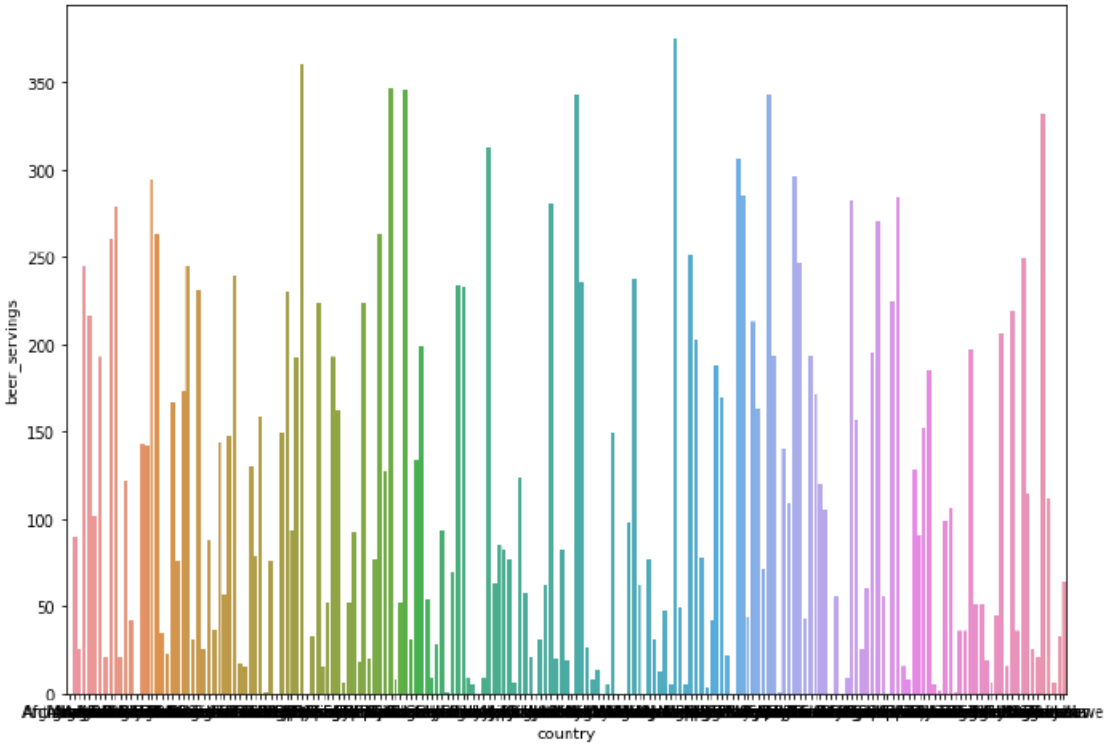


Рис. 2.8. Візуалізація даних бібліотеки Pandas.

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі приведено відомості про існуючі системи прогнозування та визначення параметрів стану водних об’єктів. Описано принципи їх функціонування, технології їх розробки. Розглянуто методи та засоби, з допомогою яких проектується інформаційна система по аналізу та прогнозуванню стану водних об’єктів. Приведено відомості про особливості проектування програмного продукту засобами Python, обробки даних з допомогою бібліотеки Pandas, візуалізації результатів з допомогою бібліотеки Seaborn.

РОЗДІЛ ІІІ. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Методика визначення рівня підйому паводкових вод

Проведено аналіз існуючих підходів до прогнозування рівня підйому паводкових вод, які викликані опадами дощу. На основі розглянених методів розроблено методику визначення рівня підйому паводкових вод та оцінки наслідків паводків. Ця методика включає в себе прогнозування безпосередньо рівня підйому паводкових вод та методику визначення наслідків підйому паводкових вод.

Загальна структурна схема представлена на рис. 3.1.

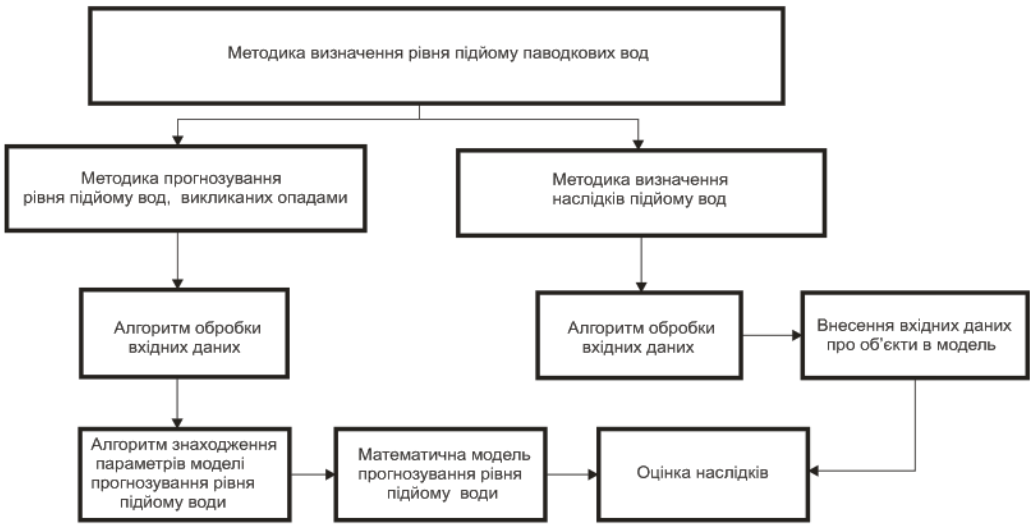


Рис. 3.1. Методика визначення рівня підйому води у водному об’єкті.

В якості алгоритмів обробки вхідних даних виступають основані на застосуванні методів кореляційного аналізу, комбінаторики, а також інформаційно-технічних методів. В результаті роботи алгоритму набори статистичних даних по зафіксованих рівнях підйому паводкових вод та погодних умов на метеостанціях проходять процедуру обробки, виділяються найбільш вагомі погодні умови, які впливають на рівень підйому паводкових вод, вхідні дані в навчальну вибірку, яка придатна для застосування методів машинного навчання.

Алгоритм знаходження параметрів моделі прогнозування рівня підйому паводкових вод дозволяє отримати математичну модель, яка базується на методах машинного навчання Дана модель буде базуватися на навчальній вибірці та відображати закономірності рівня підйому паводкових вод від зафіксованих значень погодних умов.

Виходячи із запропонованої структури, методика буде складатися з двох основних алгоритмів, які дозволять розв’язати такі задачі:

- побудова моделі прогнозу підйому рівня паводкових вод;
- проведення розрахунків по побудованій моделі.

3.2. Вимоги до вхідних даних

В якості вхідних даних виступають:

- відомості про гідрологічні характеристики розглядуваного об’єкта досліджень (ріки). Розмір вибірки для побудови моделі повинен складати не менше 10 років безперервних спостережень по кожному гідропосту:
  - рівень підйому води (мм);
  - швидкість течії (м/с);
  - сумарний водний потік ($м^3/с$).
- відомості про метеорологічні характеристики в районі, для якого здійснюється прогноз:
  - інтенсивність опадів ($мм/м^2$);
  - інтенсивність випаровування ($мм/м^2$);
  - температура (°С) та інтенсивність сонячного випромінювання ($Вт/м^2$);
  - швидкість вітру (м/с).

Проведений аналіз дозволив сформувати такі вимоги до вхідних даних. Гідрологічні та метеорологічні дані володіють деякою структурою, яку потрібно враховувати в математичній моделі чи алгоритмі. Тому важливо представити модель, яка буде описувати чи обробляти невідомі дані.

В даних (часових рядах) може бути кореляція, що потрібно враховувати при побудові моделей та відборі ознак (предикторів, незалежних змінних). Кореляція в незалежних змінних може давати негативний ефект при побудові моделі. При відсутності даних неможливо чисельно виміряти кореляцію.

Деякі явища володіють накопичуальним ефектом, деякі відбуваються раптово. Це потрібно враховувати при агрегації та відборі ознак моделі. Для досліджуваного району потрібно визначити, які явища більш характерні: явища з ефектом накопичення чи раптові.

Дані можуть бути сильно зашумлені, тому досить часто потрібна попередня обробка даних. Їх попередньо потрібно якось обробляти.

Прогнозування рівня підйому води на участку ріки з допомогою розгляненої моделі включає в себе такі етапи.

- побудова моделі прогнозування (навчання моделі) на статистичних даних;
- використання моделі прогнозування на прогнозних даних по погодніх умовах;
- уточнення моделі прогнозування (перебудова моделі).

3.3. Підготовка вхідних даних до моделювання

Для побудови навчальної вибірки було проведено аналіз доступних вхідних даних по метеорологічних показниках і по рівню підняття води в річці. Тут представлено статистичні дані по зафіксованих метеорологічних показниках на метеостанціях світу з 2005 р. Також даний ресурс має інформацію про прогноз метеоумов на метеостанції тривалістю до 6 діб. В якості вхідних даних по метеорологічних показниках взяті значення спостережень на метеостанціях по таких показниках:

- температура повітря;
- атмосферний тиск;
- відносна вологість повітря;
- швидкість вітру;
- кількість опадів;
- хмарність.

В якості статистичних даних по підйому рівня води було використано інтернет-ресурс Інформаційна система по водних ресурсах і водному господарству басейнів рік.

Тут наявні дані про вимірювання рівня води в річках подобово з 2001 р., але оскільки дані про погоду наявні з 2008 р., було використано дані з 2008 р. по теперішній час.

Після того, як дані отримані, необхідно було провести їх обробку. Від того наскільки повною буде навчальна вибірка даних, залежить якість побудованої моделі прогнозування. Без застосування процедури очистки неможливо застосувати деякі аналітичні алгоритми Data Mining. Критеріями якості даних є:

- відсутність аномальних значень;
- відсутність протирічь в даних;
- відсутність невідповідності даних по формату;
- відсутність пропусків даних;
- відсутність дублікатів в даних.

Проблема відсутності значень в таблиці даних є суттєвою проблемою при використанні моделей Data Mining. В роботі при обробці даних спостерігалася велика кількість пропущених значень як погодних даних, так і значень рівня води. Дані можуть бути відсутні за цілі періоди, такі рядки були видалені.

При використанні моделей Data Mining дані повинні подаватися на вхід в числовому форматі. Однак деякі стовпці можуть мати нечислові (категоріальні) ознаки. Наприклад, напрямок вітру – західний, хмарність 20 %. Для цього була проведена процедура кодування категоріальних значень. По кожному значенню були вибрані унікальні записи і проводилося їх кодування в числові формати даних.

Далі проводилося агрегування даних з метою приведення їх до одного формату вимірів. Представлені вхідні дані по погодним умовам мають досить велику деталізацію (виміри проводилися кожні чотири години), а дані по рівню води всього одне вимірювання на добу. Для цього проводилося усереднення даних.

Таким чином, вхідні дані після обробки представляють собою вибірку з 2010 р. по 2022 р. Незалежною (вихідною) змінною є рівень підняття води, залежними (вхідними) – спостереження метеопоказників.

3.4. Алгоритм побудови моделі прогнозування стану водних об’єктів

Нехай задано n метеорологічних факторів X (X_1, X_2, …, X_n). Нехай X_k – k-ий фактор, $k = \overline{(1,n)}$. Також нехай є m днів. Кожний фактор X_k має m спостережень X_k (X_{k1}, X_{k2}, …, X_{km}). Нехай X_{ki} – спостереження k фактора в і-ий день, $i = \overline{(1,m)}$ на конкретній метеостанції.

Нехай є також m спостережень рівня підйому води в річці на конкретному гідропості в рамках заданої території Y (Y_1, Y_2, …, Y_m). Нехай Y_i – спостереження рівня підйому води в і-день. Кожному вектору спостережень погодних умов в і-день

$W_i(X_{1i}, X_{2i}, \ldots, X_{ni})$ поставлено у відповідність значення Y_i. Нехай $W = \{W_i\}$, $i = \overline{(1,m)}$ – множина всіх векторів погодних умов, $Y = \{Y_i\}$, $i = \overline{(1,m)}$ – множина всіх спостережень підйому рівня води.

Нехай W_i – вхід, Y_i – вихід, $i = \overline{(1,m)}$. Впорядковану пару (W_i, Y_i) називають прецедентом. Множина m прецедентів називається вибіркою даних:

$$R = \{(W_i, Y_i)\}, \text{ де } i=\overline{(1,m)}, \tag{3.1}$$

Потрібно, базуючись на даних вибірки, відновити залежність між входом та виходом. Іншими словами, завдання стоїть в побудові функції

$$T(w) = y, \tag{3.2}$$

де w – прогнозні значення метеорологічних факторів; y – прогноз підйому рівня води, який видається моделлю; T – функція, яка отримує на вхід вектор w і визначає значення виходу y.

Така задача називається задачею відновлення регресії. Процес знаходження функції T називається навчанням моделі, процес визначення виходу по деякому входу з допомогою уже побудованої моделі – передбаченням (прогнозуванням).

3.5. Алгоритм ранжування сегментів річкової системи з використанням графів

Розглянемо опис алгоритму ранжування водотоків. В даний час важливим аспектом гідрологічних досліджень є не тільки натурні спостереження, але й робота з гідрологічними даними. Однак вивчення просторової структури гідрологічних систем може бути утруднене через великий об’єм даних. У таких випадках не обійтись без застосування додаткових інструментів, що дозволить фахівцеві автоматизувати процес.

Важливу роль під час роботи з гідрологічними даними відіграє візуалізація – коректне та наочне представлення результатів аналізу. Для відображення водотоків прийнято річки зображувати суцільною лінією з поступовим (залежно від кількості впадаючих у річку приток) потовщенням від витоку до гирла. Крім того, для сегментів річкової мережі часто потрібне деяке ранжування за ступенем віддаленості від витоку. Інформація такого роду важлива не тільки з погляду візуалізації, але й придатна для

повнішого сприйняття структури даних, їх просторового розподілу та коректної подальшої обробки.

Проілюструвати задачу ранжування водотоків можна таким чином.

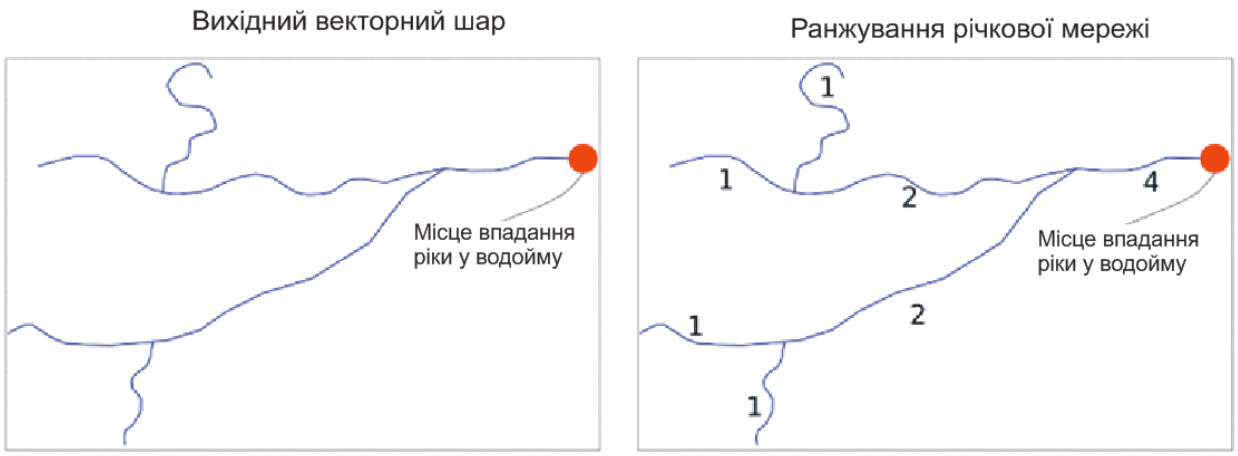


Рис. 3.2. Завдання ранжування водотоків, цифрами позначено присвоєний кожному сегменту річкової системи атрибуту сукупної кількості впадаючих приток.

Таким чином, кожному сегменту річки потрібно зіставити таке значення, яке би показувало, скільки сегментів сукупно впадає в дану ділянку. Сучасні системи мають у своєму арсеналі інструменти для роботи з річковими мережами. Проте, як правило, інструментам ранжування річок потрібна велика кількість додаткових допоміжних матеріалів та зайвих перетворень. Так рідко будь-який інструмент роботи з річковими мережами може починати свою роботу без цифрової моделі рельєфу. Істотним недоліком, крім складної та багатоступінчастої підготовки даних, є відсутність можливості використовувати вже підготовлені векторні шари з річковою мережею для аналізу, що обмежує можливість застосування цифрових даних з відкритих джерел (OpenStreetMap, Natural Earth).

Привласнити значення атрибутів сегментам можна і вручну, проте подібний підхід втрачає актуальність, якщо водозбірний басейн включає десятки тисяч сегментів. В роботі автоматизовано цю процедуру за допомогою представлення річкової мережі в математичній формі – а саме у вигляді графа та подальшого застосування алгоритмів обходу графів.

В якості вхідних даних алгоритму подається векторний шар, що складається з об’єктів з лінійним типом геометрії (Line, MultiLine). Атрибути вхідного шару зберігаються для результуючого шару. Також обов’язковим вхідним параметром є точка (Start Point Coordinates), що визначає положення гирла річкової мережі. Вона

може бути задана з карти, файлу, або з шару. Положення гирла може бути довільним, до уваги береться найближчий до точки сегмент річкової мережі (замикаючий сегмент майбутнього графа).

Описати виконувані алгоритмом завдання можна так: зіставити кожному сегменту річки значення сукупно впадаючих у нього приток, для кожного сегмента розрахувати кількість приток, а також віддаленість крайньої дальної точки сегмента від гирла.

Розглянемо опис алгоритму розв’язку даного завдання.

Дані можуть представлятися у двох основних форматах: растровому та векторному. Растр – це матриця, де у кожному пікселі зберігається певне значення параметра. У растровому форматі представляються супутникові знімки, різні вихідні шари з кліматичних моделей та інше. Кожному об’єкту векторного формату може бути співставлена деяка інформація в виді атрибутів. Всі дії будуть проводитися саме над векторним шаром річкової мережі.

Результат роботи алгоритму – векторний шар, в якому кожному об’єкту присвоєні атрибути, які визначають віддаленість сегментів від гирла ріки та сукупну кількість впадаючих в даний сегмент приток. Топологія початкового векторного шару може бути пошкоджена. Пошкоджена топологія шару може виражатися у відсутності з’єднань між об’єктами, тобто утворення різних розривів (рис. 2.7), створення додаткових замикань, перетинів тощо.

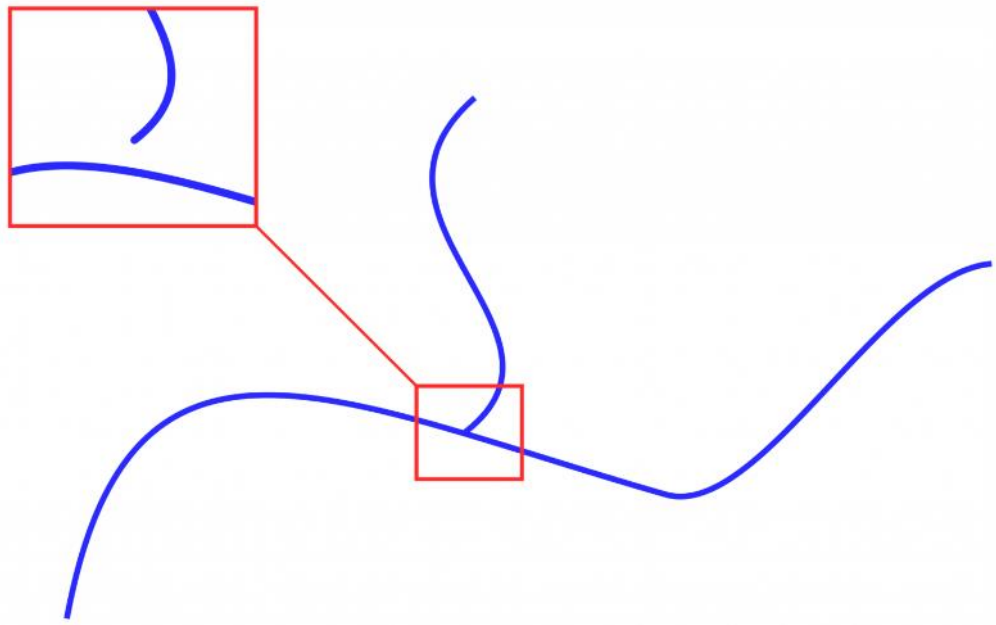


Рис. 3.3. Порушена топологія векторних об’єктів.

Тому першим етапом попередньої обробки даних є виправлення топології об’єктів: підтягнути вузли, зробити вихідний векторний шар консистентним. Після того як топологія виправлена, шар розбивається на сегменти в точках дотику та перетину ліній. Результат після розбиття проілюстровано нижче.

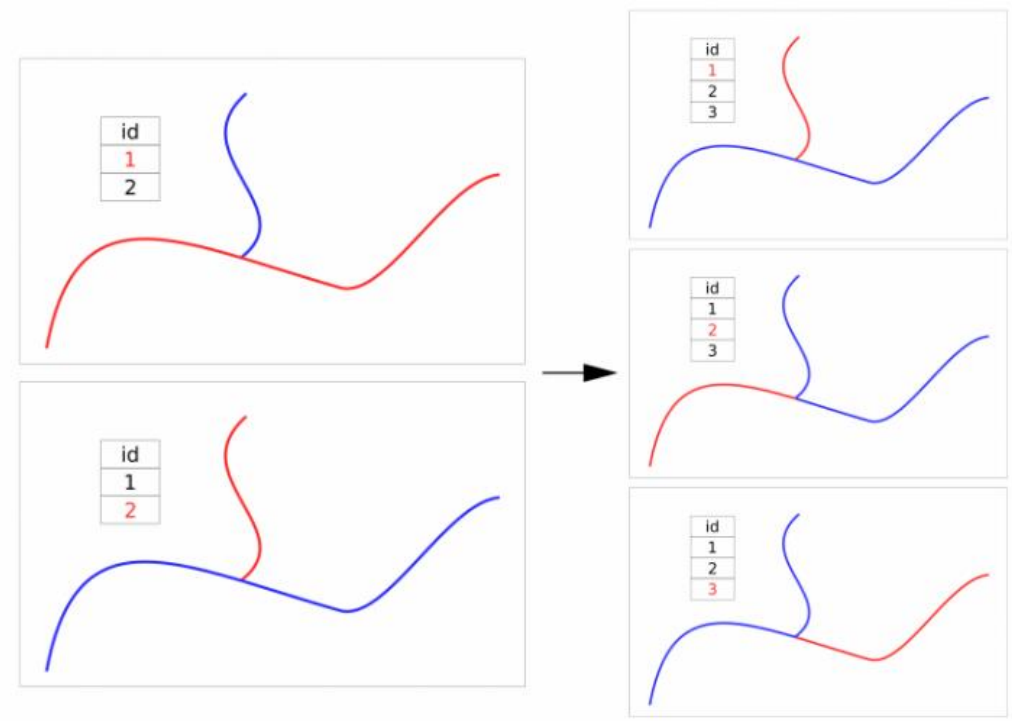


Рис. 3.4. Розбиття лінійних об’єктів на сегменти.

Для кожного сегменту розраховується протяжність і дані заносяться в атрибутивну таблицю шару. Довжина сегменту розраховується у відповідності з налаштуваннями проекту.

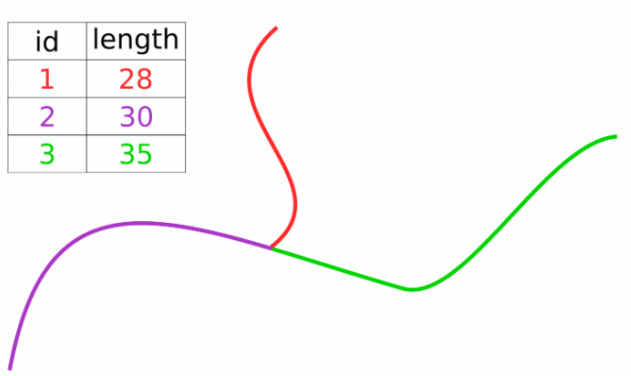


Рис. 3.5. Розрахунок протяжності сегментів.

Після цього отримують точковий векторний шар, де в таблиці атрибутів занесена інформація про перетин сегментів. Таку таблицю атрибутів можна інтерпретувати як перелік суміжності. Схематично етапи попередньої обробки представлені на наступних зображеннях.

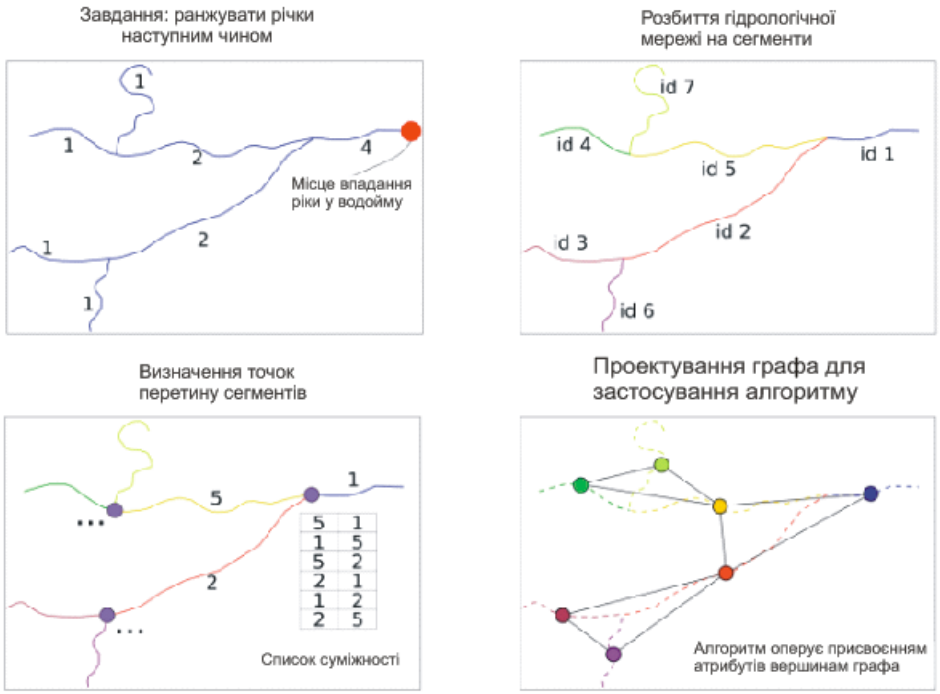


Рис. 3.6. Етапи попередньої обробки векторного лінійного шару для його представлення у вигляді графу.

В результаті попередньої обробки формується граф як математичний об’єкт Python бібліотеки Networkx. Отже, сегменти річки – це вершини графа, якщо сегменти пов’язані між собою (мають перетин), між вершинами є ребра.

Розглянемо алгоритм ранжування лінійних об’єктів. Після того, як граф сформований, відомо, з якоїсь вершини слід починати обхід (точка впадання головної річки у водойму). Надалі цю вершину називають замикаючою, тому що саме в неї впадають всі інші сегменти річкової мережі (вершини).

Алгоритм обробки складається з декількох частин.

1. Ранжування вершин графа за віддаленістю від замикаючого сегмента з привласненням атрибута міри віддаленості сегмента та атрибута кількості ділянок річкової мережі, що безпосередньо впадають в даний сегмент.

2. Обхід графа з метою привласнення атрибуту, що сукупно впадають в дану ділянку сегментів водотоків і атрибуту віддаленості крайньої точки сегмента від гирла в метрах.

Обидва етапи поділяються на кілька блоків, однак основна ідея полягає в двоетапній схемі привласнення атрибутів.

Розглянемо ранжування вершин за рівнем віддаленості від замикаючої. Починаючи від замикаючої вершини, на кожному кроці віддалялися б все далі і далі, і разом з цим, присвоювали б певний атрибут.

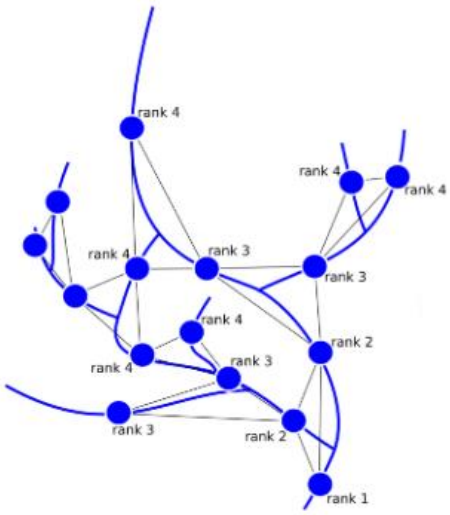


Рис. 3.7.

Можна визначити ранги для якоїсь частини річкової мережі (головної річки), і ранжувати сегменти, спираючись уже на цю інформацію. Цей підхід можна проілюструвати наступним чином.

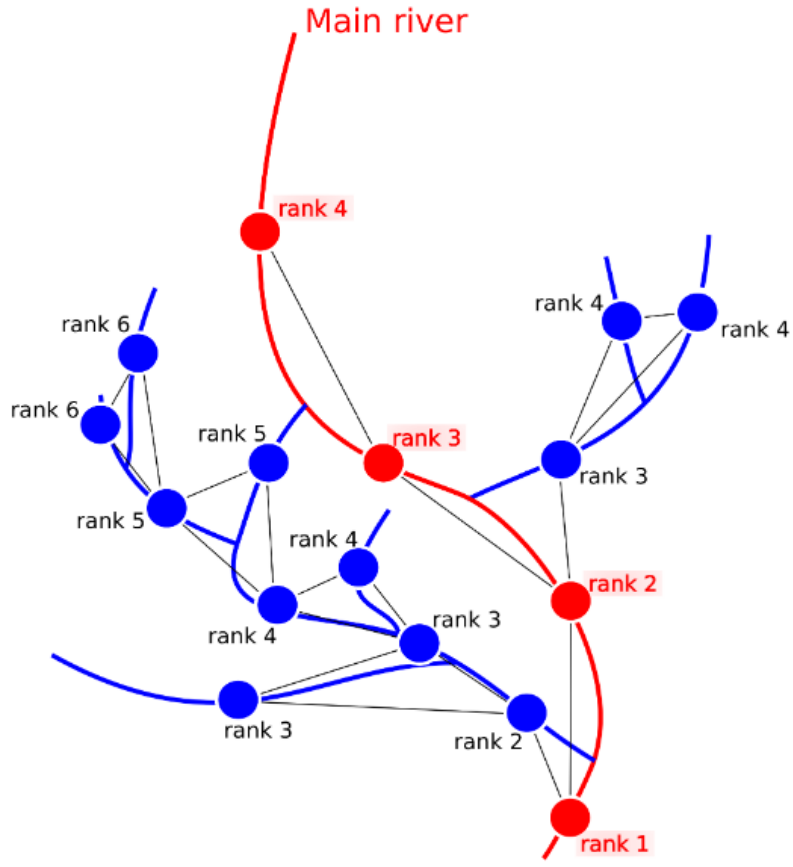


Рис. 3.8. Введення опорного маршруту в граф.

Таким чином, підграфи можуть примикати до опорного маршруту лише через одне ребро, решта ребер виключаються. Вважатимемо, що найкоротший маршрут між двома найбільш віддаленими один від одного вершинами у графі, одна з яких є замикаючою – це опорний маршрут. Такий маршрут можна отримати за допомогою алгоритму A*(A-star), але даний алгоритм працює на виваженому графі, а на ребрах даного графа поки що ніяких ваг немає. Але можна встановити ваги ребрам графа на основі довжин сегментів.

1) Привласнення ваги ребрам графа з врахуванням протяжності сегментів. Поряд із цим етапом відбувається виділення однієї компоненти зв'язаності у графі. Переміщення графом здійснюється за допомогою пошуку в ширину. Надання ваг можна продемонструвати наступним чином.

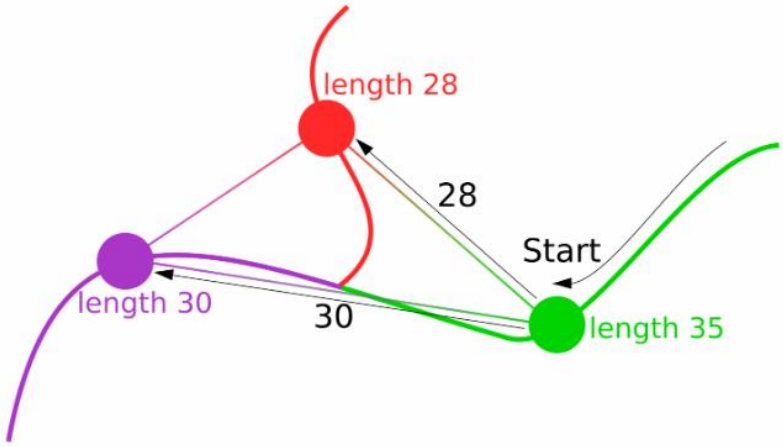


Рис. 3.9. Присвоєння ваги ребрам графу.

Таким чином, кожна вершина графа має атрибут, який відповідає за протяжність даного сегмента річки в метрах. Переносяться значення атрибутів з вершин графа на ребра ітеративно, починаючи обхід графа в ширину з замикаючої вершини.

2) A*(A-star) пошук найкоротшого шляху на зваженому графі між замикаючою та однією з найвіддаленіших вершин (сегменту в річковій мережі). Цей найкоротший маршрут між двома найвіддаленішими вершинами у графі називається опорним.

3) Ранжування за віддаленістю всіх вершин в опорному маршруті. Вершині, з якої починається обхід, надається значення рангу 1, наступній вершині – 2, далі – 3 і т.д.

4) Обхід графа з початком у вершинах опорного маршруту з ізоляцією підграфів, що розглядаються. Якщо одна з гілок графа вже має з'єднання з вершинами опорного маршруту, то ребра, що зв'язують підграфи з іншими опорними вершинами, видаляються.

Таким чином, опорний маршрут можна розглядати як головну річку в річковій мережі, коли решта сегментів є притоками для головної. Описаний вище алгоритм повною мірою відповідає цим міркуванням. Більш того, подібний підхід дозволяє досягти однозначності при ранжуванні річок у складних місцях, таких як дельта, де деякі рукави спочатку відходять від головної річки, а потім, набираючи нові притоки, знову впадають у головну річку.

Якщо вершина графа не має потомків, то це означає, що жодна притока у даний сегмент річкової мережі не впадає. Потім, для кожного вузла, у якого є потомки, необхідно додати значення у потомків (ранг потомків завжди на 1 менший, ніж ранг розглянутої вершини) і присвоїти отримане число як атрибут для вершини. Дана процедура повторюється стільки разів, скільки рангів є у графі. Таким чином, можна просуватися до початкової вершини.

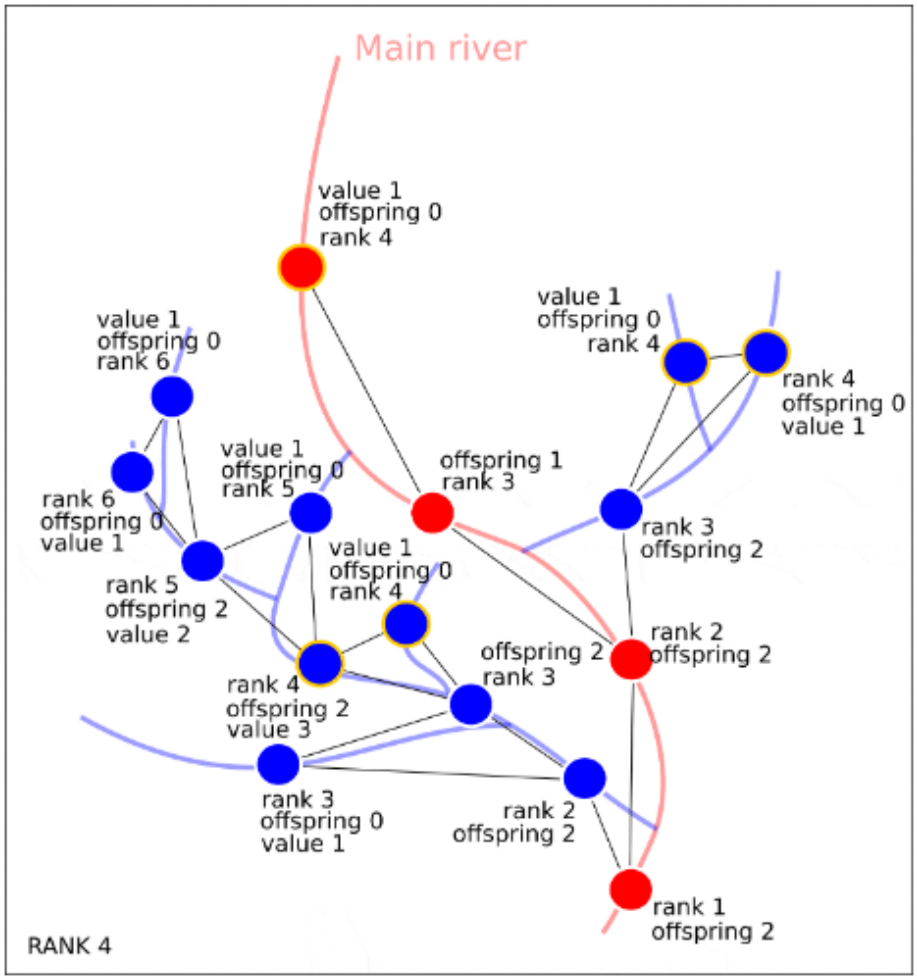


Рис. 3.10.

Одночасно з присвоєнням вершин графу атрибуту value проводиться присвоєння атрибуту, який характеризує віддаленість сегментів від гирла не кількістю сегментів,

які необхідно подолати, щоб досягти замикаючого, а відстань у метрах, яку необхідно буде подолати, щоб досягти гирла річки.

Результат застосування даного алгоритму представлено на рис. 3.11.

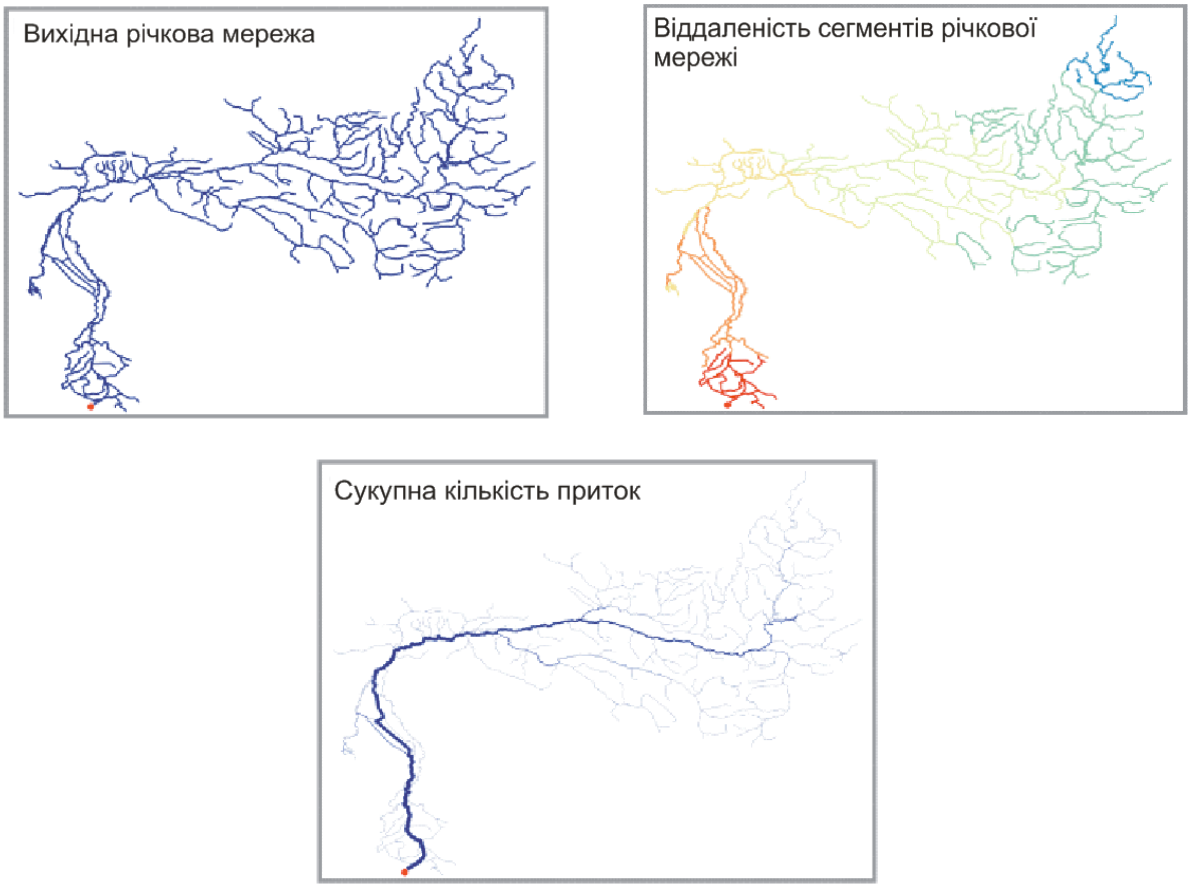


Рис. 3.11. Результати використання алгоритму.

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі приведено дані про принципи побудови алгоритму математичної моделі та підходів до прогнозування підйому рівня паводкових вод в річці, що спричинене опадами дощу. На основі приведеної математичної моделі спроектовано інформаційну систему (розділ IV). Проведені дослідження та моделювання підтверджують достовірність розробленої математичної моделі інформаційної системи та можливості по прогнозуванню стану водного об’єкта.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Розроблення інформаційної системи прогнозування стану водних об’єктів

Повені неминучі, але за умови своєчасного оповіщення їх наслідки можна мінімізувати. Через руйнівні повені щороку гине багато людей, велика кількість людей стає бездомними та багато людей помирає через відсутність належної допомоги після повені. Відсутність своєчасних сповіщень завжди була причиною питання щодо цього. Традиційні системи сповіщення дещо слабкі в прогнозуванні повеней на деякий час, щоб можна було вжити належних заходів до цієї катастрофи.

Використовуючи машинне навчання, можна передбачати повені з певною точністю. Дана робота спрямована на створення прогнозного моделювання на основі даних про погоду в окремих районах, щоб передбачити виникнення повені. Прогностична модель побудована на різних алгоритмах машинного навчання.

У цьому проекті використовуються такі алгоритми класифікації, як Decision tree, Random forest, KNN, Xgboost.

Архітектура проекту виглядає наступним чином.

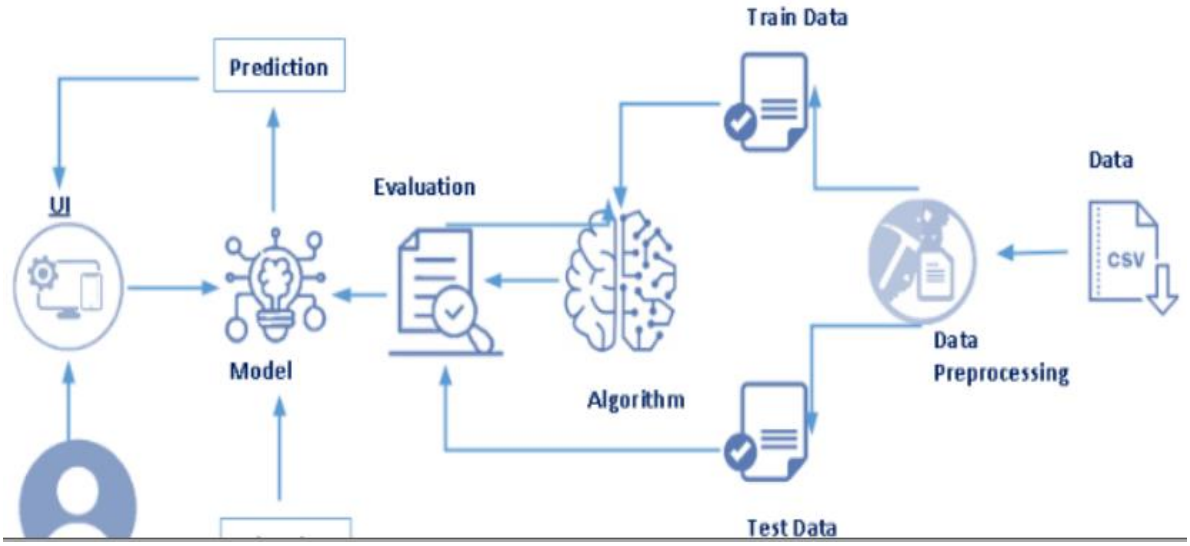


Рис. 4.1. Архітектура проекту.

Щоб досягти цього, виконуються наступні кроки:

1. встановлюються потрібні бібліотеки Python;

2. проводиться збір даних.

3. Попередня обробка даних:

- імпорт бібліотек;
- імпорт набору даних;
- розуміння типу даних і короткого опису функцій;
- обробка відсутніх дані;
- візуалізація даних;
- поділ набору даних на залежні та незалежні змінні;
- поділ даних на навчальні та тестові.

4. Побудова моделі:

- навчання та тестування моделі;
- оцінка моделі.

Читання набору даних

Формат набору даних може бути у файлах .csv, excel, .txt, .json:

```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
```

У Pandas є функція read_csv() для читання набору даних:

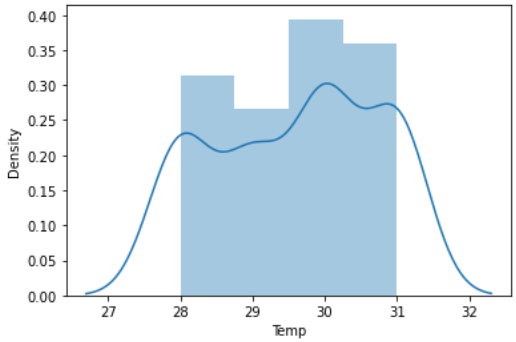
```
[ ]  dataset = pd.read_excel('flood dataset.xlsx')
```

Однофакторний аналіз

Однофакторний аналіз - це поділ даних за однією ознакою. Діаграма розподілу підходить для порівняння діапазону та розподілу для груп числові дані. Дані відображаються у вигляді значення точок вздовж осі. Використовується для визначення того, якому типу розподілу притаманні дані.

```
[ ]  print(sns.distplot(dataset["Temp"]))
```

```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be
  warnings.warn(msg, FutureWarning)
AxesSubplot(0.125,0.125;0.775x0.755)
```

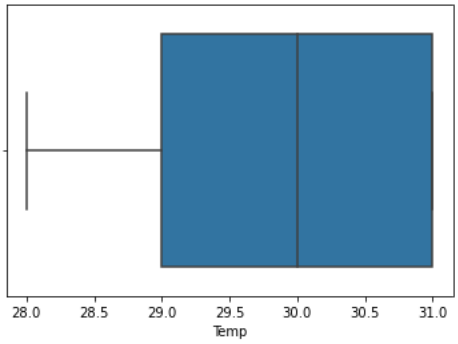


З наведеного вище графіка можна зробити висновок, що температура стовпця відповідає певній нормі розподілу, це означає, що дані підлягають нормальному розподілу.

Діаграма Boxplot:

```
[ ]  print(sns.boxplot(dataset['Temp']))
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. Fr
  warnings.warn(
AxesSubplot(0.125,0.125;0.775x0.755)
```



Багатофакторний аналіз

Багатофакторний аналіз полягає у пошуку зв’язку між декількома ознаками. Для цього будується теплова карта – це техніка візуалізації даних, яка показує величину кольору у двох вимірах. Колір може відрізнятися за відтінком або інтенсивністю, даючи очевидні візуальні підказки про те, як цеі дані скупчені або змінюються в просторі.

```
[ ]  import seaborn as sns
     fig=plt.gcf()
     fig.set_size_inches(15,15)
     fig=sns.heatmap(dataset.corr(),annot=True,cmap='summer',
                     linewidths=1,linecolor='k',square=True,
                     mask=False, vmin=-1, vmax=1,
                     cbar_kws={"orientation": "vertical"},cbar=True)
```

На графіку можна визначити менш корельовані значення та відкинути їх значення.

Описовий аналіз

Щоб перевірити перші п’ять рядків набору даних, для цього призначена функція head():

```
[ ] dataset.drop(["Oct-Dec"],axis=1,inplace=True)
```

```
[ ] dataset.head(10)
```

	Temp	Humidity	Cloud Cover	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	avgjune	sub	flood
0	29	70	30	3248.6	73.4	386.2	2122.8	274.866667	649.9	0
1	28	75	40	3326.6	9.3	275.7	2403.4	130.300000	256.4	1
2	28	75	42	3271.2	21.7	336.3	2343.0	186.200000	308.9	0
3	29	71	44	3129.7	26.7	339.4	2398.2	366.066667	862.5	0
4	31	74	40	2741.6	23.4	378.5	1881.5	283.400000	586.9	0
5	30	70	38	2708.0	34.1	230.0	1943.1	138.300000	254.1	0
6	29	74	40	3671.1	23.7	328.0	2737.8	256.966667	669.5	1
7	30	78	36	2648.3	28.8	283.7	2023.6	197.533333	450.0	0
8	30	71	40	3050.2	65.9	628.3	1940.4	234.900000	231.5	0
9	30	70	34	2848.6	28.4	296.7	1886.5	226.666667	531.2	0

Розуміння типу даних і результату функцій

● те, як інформація зберігається в об’єкті DataFrame або Python, впливає на те, чи можна її використовувати, а також результати обчислень. Є два основних типи даних: числові та текстові типи даних.

● числові типи даних включають цілі числа та числа з плаваючою крапкою.

● текстовий тип даних відомий як рядки в Python або об’єкти в Pandas. Рядки можуть містити цифри або символи.

● можна побачити, як виглядає набір даних, використовуючи метод info().

● метод info() надає інформацію про набір даних.

```
[ ]  print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115 entries, 0 to 114
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Temp         115 non-null    int64  
 1   Humidity     115 non-null    int64  
 2   Cloud Cover  115 non-null    int64  
 3   ANNUAL       115 non-null    float64
 4   Jan-Feb      115 non-null    float64
 5   Mar-May      115 non-null    float64
 6   Jun-Sep      115 non-null    float64
 7   avgjune      115 non-null    float64
 8   sub          115 non-null    float64
 9   flood        115 non-null    int64  
dtypes: float64(6), int64(4)
memory usage: 9.1 KB
None
```

У наборі даних немає текстових даних, увесь набір даних містить числа з плаваючою крапкою та цілі числа. Функція describe() використовуються для обчислення таких значень, як кількість, середнє значення, стандартне відхилення даних.

```
[ ]  dataset.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Temp	115.0	29.600000	1.122341	28.0	29.000000	30.000000	31.000000	31.000000
Humidity	115.0	73.852174	2.947623	70.0	71.000000	74.000000	76.000000	79.000000
Cloud Cover	115.0	36.286957	4.330158	30.0	32.500000	36.000000	40.000000	44.000000
ANNUAL	115.0	2925.487826	422.112193	2068.8	2627.900000	2937.500000	3164.100000	4257.800000
Jan-Feb	115.0	27.739130	22.361032	0.3	10.250000	20.500000	41.600000	98.100000
Mar-May	115.0	377.253913	151.091850	89.9	276.750000	342.000000	442.300000	915.200000
Jun-Sep	115.0	2022.840870	386.254397	1104.3	1768.850000	1948.700000	2242.900000	3451.300000
Oct-Dec	115.0	497.636522	129.860643	166.6	407.450000	501.500000	584.550000	823.300000
avgjune	115.0	218.100870	62.547597	65.6	179.666667	211.033333	263.833333	366.066667
sub	115.0	439.801739	210.438813	34.2	295.000000	430.600000	577.650000	982.700000
flood	115.0	0.139130	0.347597	0.0	0.000000	0.000000	0.000000	1.000000

Попередня обробка даних

Надалі потрібно попередньо обробити дані. Набір даних для завантаження не підходить для навчання моделі машинного навчання, тому потрібно очистити набір даних, щоб отримати хороші результати. Ця дія включає такі кроки:

- обробка відсутніх значень;
- обробка категоріальних даних;
- обробка викидів;
- поділ залежних і незалежних змінних;
- розділення набору даних на навчальний і тестовий набір;
- масштабування функцій.

Це загальні кроки попередньої обробки даних перед використанням їх для машиннного навчання.

Обробка відсутніх значень

Однією з найпоширеніших ідей для вирішення цієї проблеми є взяти середнє значення всіх значень для безперервного і для категорійних, використати значення та замінити відсутні дані.

● True означає, що певний стовпець має відсутні значення, можна побачити кількість відсутніх значень у кожному стовпці за допомогою функції isnull().sum:

```
[ ]  dataset.isnull().any()
```

```
Temp          False
Humidity      False
Cloud Cover   False
ANNUAL        False
Jan-Feb       False
Mar-May       False
Jun-Sep       False
avgjune       False
sub           False
flood         False
dtype: bool
```

Дані не містять нульових значень. Функція null.any() повертає логічні значення False.

Обробка категоріальних значень

Набір даних має категоричні дані, у які потрібно перетворити ці категоріальні дані з допомогою цілочисельного кодування або двійкового кодування. Для перетворення категоріальних ознак у числові використовуються деякі методи кодування.

Є кілька методів, але в даному проекті використовуються відображення функцій і мітки кодування. У наборі даних немає категоріального типу даних, тому можна пропустити цей крок.

Розбиття набору даних на залежні та незалежні змінні.

У машинному навчанні поняття залежної змінної (y) і незалежної змінної (x) важливо розуміти. Тут залежна змінна є нічим іншим як вихідними даними в наборі даних, а незалежна змінна - це всі вхідні дані в наборі даних.

Потрібно розділити набір даних на матрицю незалежних змінних та вектор або залежну змінну. Математично вектор визначається як матриця, яка має лише один стовпець. Створимо незалежні та залежні змінні:

```
[ ]  X = dataset.iloc[:,2:7].values
```

```
[ ]  y = dataset.iloc[:,9:].values
```

Тут створено DataFrame незалежної змінної x з вибраних стовпців і для залежної змінної y вибрано клас колонки. DataFrame використовується для представлення таблиці даних із рядками та стовпцями.

Поділ набору даних на навчальний та тестовий набори даних

Після цього потрібно розділити набір даних на навчальний набір і протестувати його за допомогою класу train_test_split. Train_test_split: він використовується для поділу масивів даних на навчальні дані та для тестування даних.

```
[ ]  from sklearn.model_selection import train_test_split
     X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=10)
```

Масштабування функцій

StandardScaler - це основний інструмент масштабування бібліотеки Sklearn, він використовує визначення standardization для стандартизації даних.

```
[ ]  from sklearn.preprocessing import StandardScaler
     sc=StandardScaler()
     X_train=sc.fit_transform(X_train)
     X_test=sc.fit_transform(X_test)
```

```
[ ]  from joblib import dump
     dump(sc,"transform.save")
```

```
['transform.save']
```

Проектування моделі

Після цього дані очищено, і настав час побудувати модель. Можна навчити дані різними алгоритми. Для цього проекту використано чотири алгоритми класифікації. найкраща модель зберігається на основі її продуктивності. Для оцінки продуктивності використовується confusion matrix та classification report.

Модель Decision Tree

Створюється функція під назвою Decision Tree, а дані навчання та тестування передаються як параметри. Усередині функції ініціалізується алгоритм DecisionTreeClassifier і навчальні дані передаються в модель за допомогою функції .fit(). Тестові дані прогнозуються за допомогою .predict() і зберігаються в новій змінній. Для оцінки моделі використовується confusion matrix та classification report.

```
: from sklearn.tree import DecisionTreeClassifier
```

```
: model = DecisionTreeClassifier()
```

```
: model.fit(X_train, y_train)
```

```
: DecisionTreeClassifier()
```

```
: from sklearn.metrics import accuracy_score,classification_report
```

```
: y_predict = model.predict(X_test)
  y_predict_train=model.predict(X_train)
```

Оцінка точності

```
: print('Test data',accuracy_score(y_test,y_predict))
  print('Train data',accuracy_score(y_train,y_predict_train))
```

```
Test data 0.9655172413793104
Train data 1.0
```

Confusion matrix

```
pd.crosstab(y_test,y_predict)
```

Classification report

```
: print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        26
           1       0.75      1.00      0.86         3

    accuracy                           0.97        29
   macro avg       0.88      0.98      0.92        29
weighted avg       0.97      0.97      0.97        29
```

Random Forest Модель

Створюється функція з назвою RandomForest, а дані навчання та тестування передаються як параметри. Усередині функції ініціалізується алгоритм RandomForestClassifier і навчальні дані передаються в модель за допомогою функції .fit(). Тестові дані прогнозуються за допомогою .predict() і зберігаються в новій змінній. Для оцінки моделі використано confusion matrix та classification report.

```
from sklearn.ensemble import RandomForestClassifier
```

```
model = RandomForestClassifier()
model.fit(X_train,y_train.ravel())
```

```
RandomForestClassifier()
```

```
from sklearn.metrics import accuracy_score,classification_report
```

```
y_predict = model.predict(X_test)
y_predict_train=model.predict(X_train)
```

Оцінка точності

```
print('Test data',accuracy_score(y_test.ravel(),y_predict))
print('Train data',accuracy_score(y_train.ravel(),y_predict_train))
```

```
Test data 0.9655172413793104
Train data 1.0
```

Confusion matrix

```
pd.crosstab(y_test.ravel(),y_predict)
```

col_0	0	1
row_0		
0	25	1
1	0	3

Classification report

```
print(classification_report(y_test.ravel(), y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        26
           1       0.75      1.00      0.86         3

    accuracy                           0.97        29
   macro avg       0.88      0.98      0.92        29
weighted avg       0.97      0.97      0.97        29
```

Модель KNN

Створюється функція під назвою KNN, а дані навчання та тестування передаються як параметри. Усередині функції ініціалізується алгоритм KNeighborsClassifier і навчальні дані передаються в модель за допомогою функції .fit(). Тестові дані прогнозуються за допомогою .predict() функції та зберігаються в новій змінній. Для оцінки моделі використано confusion matrix та classification report.

```
from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier()

knn.fit(X_train,y_train.ravel())
```

```
KNeighborsClassifier()
```

```
y_predict=knn.predict(X_test)
y_pred=knn.predict(X_train)
```

```
from sklearn.metrics import accuracy_score,classification_report

print("Test accuracy=", accuracy_score(y_test.ravel(),y_predict))
print("Train accuracy=", accuracy_score(y_train.ravel(),y_pred))
```

```
Test accuracy= 0.896551724137931
Train accuracy= 0.9418604651162791
```

```
pd.crosstab(y_test.ravel(),y_predict)
```

col_0	0	1
row_0		
0	23	3
1	0	3

```
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.88      0.94        26
           1       0.50      1.00      0.67         3

    accuracy                           0.90        29
   macro avg       0.75      0.94      0.80        29
weighted avg       0.95      0.90      0.91        29
```

Модель Xgboost

Створюється функція з іменем xgboost, а дані навчання та тестування передаються як параметри. Усередині функції ініціалізується алгоритм GradientBoostingClassifier і навчальні дані передаються в модель за допомогою функції .fit(). Тестові дані прогнозуються за допомогою .predict() і зберігаються в новій змінній. Для оцінки моделі використано confusion matrix та classification report. Підгонка моделі відбувається за допомогою даних x_train, y_train.

```
]: from xgboost import XGBClassifier
```

```
]: xgb = XGBClassifier()
```

```
]: xgb.fit(X_train,y_train)
```

```
]: XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
```

```
]: y_predict=xgb.predict(X_test)
   y_pred=xgb.predict(X_train)
```

```
]: from sklearn.metrics import accuracy_score,classification_report

   print("Test accuracy=", accuracy_score(y_test,y_predict))
   print("Train accuracy=", accuracy_score(y_train,y_pred))
```

```
Test accuracy= 0.9655172413793104
Train accuracy= 1.0
```

```
]: pd.crosstab(y_test.ravel(),y_predict)
```

```
]:
```

col_0	0	1
row_0		
0	25	1
1	0	3

```
]: print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        26
           1       0.75      1.00      0.86         3

    accuracy                           0.97        29
   macro avg       0.88      0.98      0.92        29
weighted avg       0.97      0.97      0.97        29
```

```
from sklearn import tree
from sklearn import ensemble
from sklearn import neighbors
import xgboost
```

```
dtree = tree.DecisionTreeClassifier()
Rf = ensemble.RandomForestClassifier()
knn = neighbors.KNeighborsClassifier()
xgb = xgboost.XGBClassifier()
```

```
dtree = tree.DecisionTreeClassifier()
Rf.fit(X_train,y_train.ravel())
knn.fit(X_train,y_train.ravel())
xgb.fit(X_train,y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
```

Порівняння моделей

Для порівняння наведених вище чотирьох моделей визначається функція порівняння моделі:

```
from sklearn import metrics
```

```
print(metrics.accuracy_score(y_test,y_predict))
print(metrics.accuracy_score(y_test,y_predict))
print(metrics.accuracy_score(y_test,y_predict))
print(metrics.accuracy_score(y_test,y_predict))
```

```
0.9655172413793104
0.9655172413793104
0.9655172413793104
0.9655172413793104
```

Після виклику цієї функції результати моделей відображаються як вихідні дані. Внизу видно точність моделей. Усі три моделі мають точність 96,55 %.

Оцінка ефективності моделі

Порівнявши всі три моделі з різними атрибутами, Xgboost є кращою моделлю, тому зберігаємо цю модель:

```
metrics.confusion_matrix(y_test,y_predict)
```

```
array([[25,  1],
       [ 0,  3]], dtype=int64)
```

```
print(metrics.accuracy_score(y_test,y_predict))
```

```
0.9655172413793104
```

```
print(metrics.precision_score(y_test,y_predict))
```

```
0.75
```

```
print(metrics.recall_score(y_test,y_predict))
```

```
1.0
```

4.2. Результати роботи інформаційної системи

4.2.1. Дослідження гідрометеорологічних часових рядів опадів

Для цього в роботі використовуються дані, які були отримані від метеостанцій та гідропостів. Дані про кількість опадів поступають від гідрометрологічних станцій у вигляді часових рядів, оскільки оперують періодом спостереження, найчастіше з моменту запуску станцій, а також добові кількості опадів у даному населеному пункті. Такі дані надходять у вигляді відформатованих таймсерій (файли .csv або .xlsx). В роботі використано дані часових рядів про кількість опадів та рівень стоку річок у форматі .csv.

Для роботи з експериментальними даними використано бібліотеку Pandas, для наочного графічного представлення даних використано бібліотеку Seaborn.

```
[ ]  import matplotlib.pyplot as plt
     import pandas as pd
     import seaborn as sns
     sns.set(style="whitegrid")
```

```
[ ]  d1 = pd.read_csv("d1.csv", names = ["Date", "Rainfall"])
     d1['Date'] = pd.to_datetime(d1["Date"])
     d1 = d1.set_index('Date')
     d1 = d1.resample("Y").sum()
```

```
[ ]  ax = sns.barplot(x=d1.index, y=d1["Rainfall"], color = "royalblue")
     ax.set_xlabel("Роки")
     ax.set_ylabel("Кількість опадів, мм")
     ax.set_xticklabels(labels = d1.index.year, rotation = 45)
```

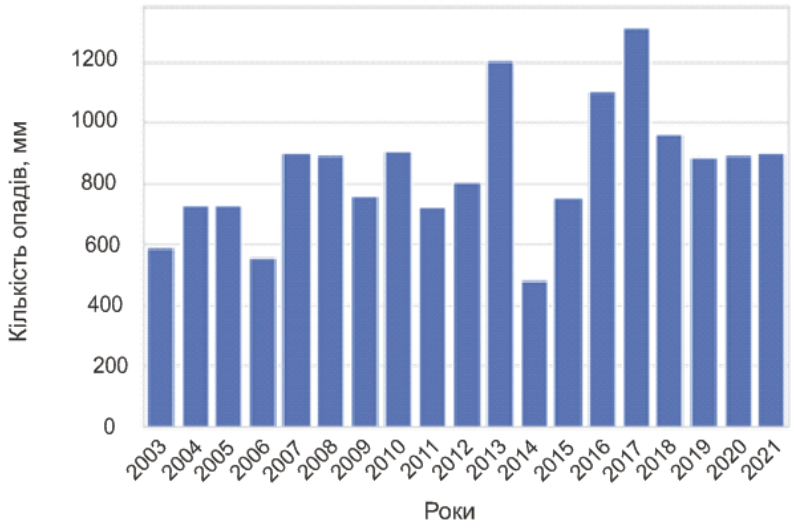


Рис. 4.2. Гістограма опадів для першої гідрометеорологічної станції.

Спочатку зчитують дані з CSV-файлу зі щоденними вимірами опадів для першої гідрометеорологічної станції у датафрейм даних Pandas, назвавши при цьому стовпці Дата та Опади. Зазвичай кількість опадів однієї станції за певний період представляється у вигляді стовпчастої діаграми у вигляді річних сум опадів у міліметрах. Таким чином, індекс datetime перераховується на річні значення шляхом підсумовування всіх денних значень кожного року. В бібліотеці Pandas це можна зробити досить просто.

Для візуалізації даних використовується бібліотека Seaborn, оскільки вона дає багато варіантів модифікацій. В ній використано опцію гістограми, коли індекс дати встановлюється як вісь x, а сумарна кількість опадів за кожен рік – як вісь y. Крім того, додані мітки осей x і y, а позначки (роки) осі x повернуті для зручності.

4.2.2. Часові ряди опадів для декількох пунктів спостережень

Часто розглядають дані з декількох гідрометеорологічних станцій для досліджуваного річкового вододілу або навколо нього. У таких випадках можна надати дані про опади у вигляді графіків з декількома річними стовпцями, де можна порівняти, наприклад, де був дуже дощовий рік або щось інше. Це часто буває, коли спостереження проводяться в гірській місцевості, що відіграє свою роль у кількості опадів. Щоб перевірити, чи був якийсь рік найбільше дощовим на певному вододілі ріки, потрібно взяти дані з декількох гідрометеорологічних станцій і спочатку представити з них дані у вигляді кількох стовпчастих графіків з відображенням медіани та середнього арифметичного значення.

```
[ ]  import matplotlib.pyplot as plt
     import pandas as pd
     import seaborn as sns
     sns.set(style="whitegrid")
```

```
[ ]  d2 = pd.read_csv("d2.csv", usecols = ["Date", "1", "2", "3", "4", "5", "6", "7", "8"])
     d2["Date"] = pd.to_datetime(d2["Date"])
     d2 = d2.set_index("Date")
     d2 = d2.resample("Y").sum()
```

```
[ ]  fig1, ax = plt.subplots(2,4, figsize = (18,8), tight_layout=True, sharex = True)

     for i, (ax, value, name) in enumerate(zip(ax.flatten(), d2.values.T, d2.columns)):
         ax.set_xticks(d2.index.year)
         ax.set_xticklabels(labels = d2.index.year.values, rotation=60)
         ax.bar(d2.index.year, d2[name], label=name, color = "royalblue")
         ax.hlines(d2[name].mean(), 1999.5, 2018.5, color="green", label="медіана",
     ls = "--")
         ax.hlines(d2[name].median(), 1999.5, 2018.5, color="red", label="ср.знач",
     ls = ":")
         if i >3:
             ax.set_xlabel ("Рік")
         ax.set_ylabel("Кількість опадів, мм")
         ax.legend(shadow=True, loc="upper left")
```

Для цього файл у форматі d2.csv з даними станцій гідрометеостанцій зчитується та конвертується у фрейм даних Pandas. В роботі проаналізовано дані з восьми станцій. Для цього створюється вісім графіків в одному вікні (2 рядки, 4 стовпці) із загальною віссю x. На осі абсцис як написи встановлені роки. Дані представлені у вигляді гістограм з додатковими горизонтальними лініями, що представляють собою середнє арифметичне та медіанне значення для кожної станції, з яких надходять дані.

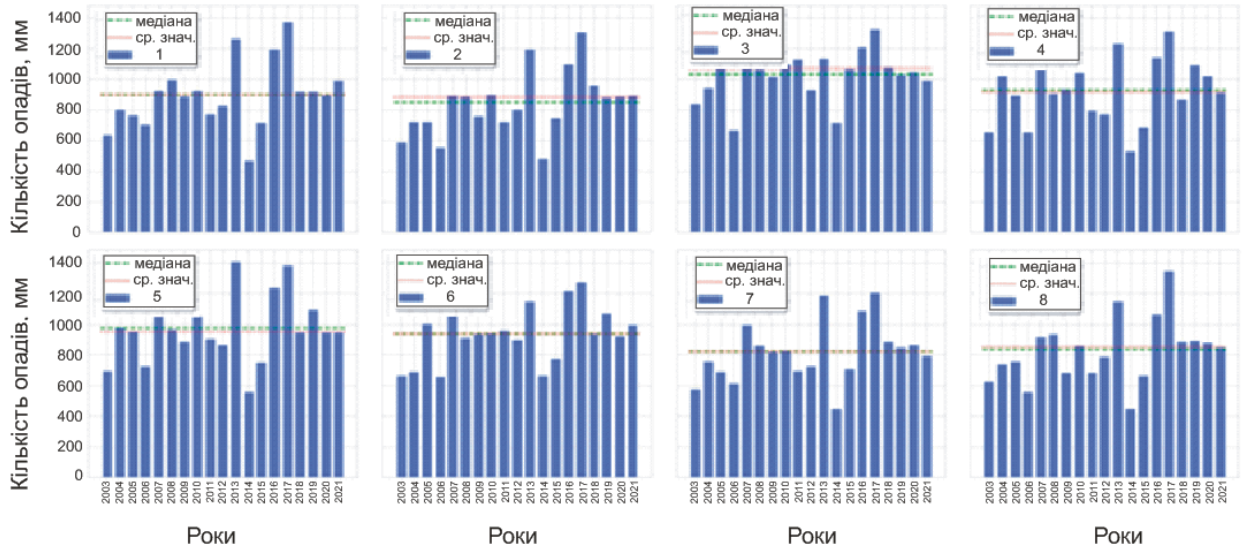


Рис. 4.3. Часові ряди кількості опадів, дані яких надходять з кількох гідрометеорологічних станцій.

4.2.3. Проведення просторового аналізу даних та кореляція отриманих результатів

Іншою можливістю представлення даних від кількох джерел даних є просторовий аналіз та кореляція результатів. Такий підхід може допомогти розрізнити можливі закономірності або гарячі точки з високими чи низькими значеннями опадів. Перед інтерполяцією потрібно підготувати дані. Це означає, що потрібно обчислити середнє

арифметичне значення за період, який цікавить, у кожній точці. Це завдання можна виконати за допомогою наступного програмного коду. Для цього треба взяти файл d2.csv з попереднього прикладу, в якому знаходяться дані про щоденні опади за дев'ятнадцять років на кількох гідропостах навколо вододілу річки.

```
[ ]  import matplotlib.pyplot as plt
     from matplotlib import cm
     import pandas as pd
     import seaborn as sns
     import numpy as np
     from scipy.interpolate import griddata
```

```
[ ]  meteo_loc = pd.read_csv("meteo_stat.csv", usecols = ["name", "real_name", "lat", "long"]).sort_values(by = ["name"])
     d2 = pd.read_csv("d2.csv")

     d2 = pd.DataFrame(d2.mean(axis = 0)).rename_axis('name').reset_index().sort_values(by=["name"]).rename(columns =
     {0: "Mean"})
```

```
[ ]  meteo_loc = pd.merge(meteo_loc, d2, on="name")
     X,Y = np.meshgrid(np.linspace(meteo_loc["long"].min()-0.05,meteo_loc["long"].max()+0.05,2000),
                       np.linspace(meteo_loc["lat"].min()-0.05,meteo_loc["lat"].max()+0.05,2000))
```

```
[ ]  grid_rain = sp.interpolate.Rbf(meteo_loc["long"],meteo_loc["lat"], meteo_loc["Mean"])
     spatial_rain = grid_rain(X,Y)
     fig, ax = plt.subplots(figsize = (16, 8))
     ax.scatter(meteo_loc["long"], meteo_loc["lat"], color = "black")
```

```
[ ]  for i, (x,y) in enumerate(zip (meteo_loc["long"], meteo_loc["lat"])):
         label = meteo_loc["real_name"][i]
         ax.annotate(label, (x,y), textcoords = "offset points", xytext = (2,3))
     fig.colorbar(gg, format = '%.1f')
```

На наступному етапі досліджень було отримано кореляційну залежність середньодобової кількості опадів над досліджуваним вододілом річки з виміряним річковим стоком. Для отримання графічних залежностей потрібно виміряти дані для річкового стоку на виході із басейну.

Для цього будують комбіновану лінійно-стовпчасту діаграму, де дані річкового стоку наносяться на першу вісь Y_1, а кількість опадів – на другу вертикальну вісь Y_2. Оскільки спостереження проводять за місячними даними, стовпчиковий графік буде хаотичним, тому кількість опадів відображається у вигляді лінійного графіка із заповненою площею під кривою для досягнення ефекту, який аналогічний для стовпчастого графіку.

```
[ ]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[ ]  sns.set_style("whitegrid")
     d2 = pd.DataFrame(pd.read_csv("d2.csv", parse_dates=['Date']).mean(axis = 1, numeric_only=True), columns = ["precip"])
     streamflow = pd.read_csv("streamflow.csv", parse_dates=['Date'])
     streamflow = pd.concat([streamflow, d2], axis = 1)
     streamflow = streamflow.resample("M", on="Date").agg({"1": np.mean, "2": np.sum})
```

```
[ ]  fig, ax1 = plt.subplots(figsize=(14,6))
     ax2 = ax1.twinx()
     sns.lineplot(x = streamflow.index, y= "precip", data = streamflow, color="blue", ax=ax2)
```

```
[ ]  ax2.fill_between(streamflow.index, 0, streamflow["precip"], alpha = 0.8)
     ax2.set(ylim=(0, 400))
     ax2.set_ylabel("Кількість опадів, мм")
     ax2.invert_yaxis()
     sns.lineplot(x = streamflow.index, y= "1", data = streamflow, color="red", ax=ax1)
     ax1.set(ylim=(0, 80))
     ax1.set_ylabel("Річковий сток, м3/с")
     ax1.set_xlabel("Роки")
```

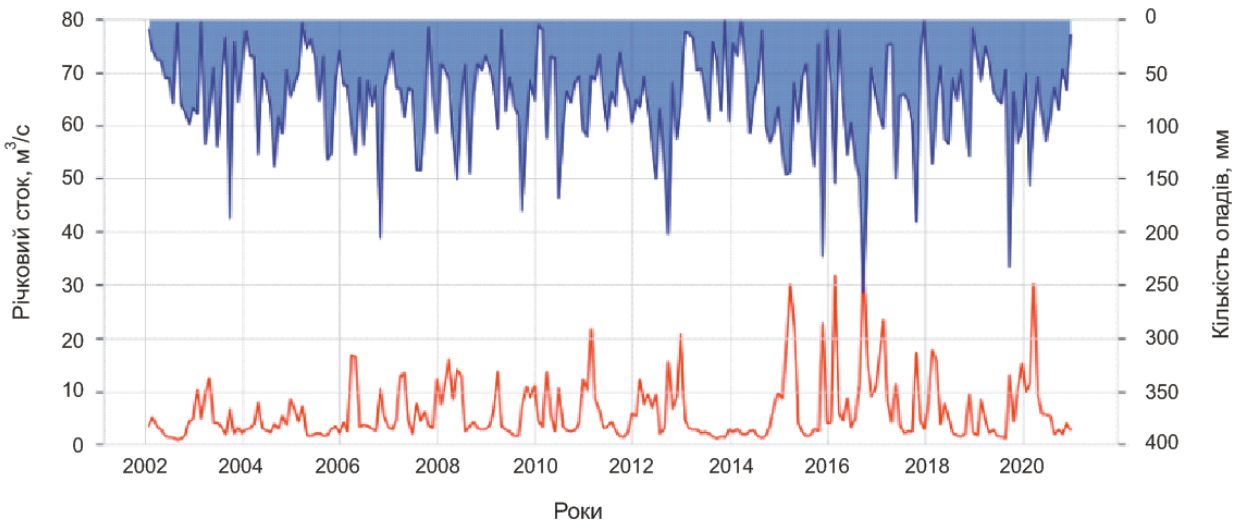


Рис. 4.4. Графік залежності річкового стоку та кількості опадів для першої гідрометеорологічної станції.

Такі графіки використовуються як первинний індикатор того, як опади впливають на річковий стік із водозбору. Крім цього, у цьому аналізі можна розглядати гідрометеостанцію, яка розташована вище за течією, для перевірки процесів стоку із субводосбору, меншої частини основного водозбору досліджуваної річки.

4.2.4. Отримання даних кількості опадів та річкового стоку як сезонного фактору

В роботі проведено аналіз даних кількості опадів та рівня води в річці на сезонній основі, щоб побачити, наскільки сильні дощі впливають на стік водозбору ріки, для якого ведуть спостереження. Для цього щокварталу проводять групування даних. Тому в Pandas встановлюють стовпець Date як індекс, щоб використовувати параметри Pandas для повторної вибірки даних і знову агрегувати річковий стік як середнє значення та кількість опадів як щомісячну суму. Дані було поділено на чотири сезони: зима (грудень-лютий), весна (березень-травень), літо (червень-серпень) та осінь (вересень-листопад).

```
[ ]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     sns.set_style("whitegrid")
```

```
[ ]  d2 = pd.DataFrame(pd.read_csv("d2.csv", parse_dates=['Date']).mean(axis = 1, numeric_only=True), columns = ["precip"])
     streamflow = pd.read_csv("streamflow.csv", parse_dates=['Date'])
     streamflow = pd.concat([streamflow, d2], axis = 1).set_index(["Date"])
     winter = streamflow[(streamflow.index.month >= 1) &  (streamflow.index.month <= 3)]
     winter = winter.reset_index().resample("M", on="Date").agg({"1": np.mean, "precip": np.sum}).dropna().reset_index()
```

```
[ ]  spring = streamflow[(streamflow.index.month >= 4) &  (streamflow.index.month <= 6)]
     spring = spring.reset_index().resample("M", on="Date").agg({"1": np.mean, "precip": np.sum}).dropna().reset_index()
     summer = streamflow[(streamflow.index.month >= 7) &  (streamflow.index.month <= 9)]
     summer = summer.reset_index().resample("M", on="Date").agg({"1": np.mean, "precip": np.sum}).dropna().reset_index()
```

```
[ ]  autumn= streamflow[(streamflow.index.month >= 10) &  (streamflow.index.month <= 12)]
     autumn = autumn.reset_index().resample("M", on="Date").agg({"1": np.mean, "precip": np.sum}).dropna().reset_index()
     fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows = 2, ncols = 2, figsize = (14,8), tight_layout=True)
     ax1a = ax1.twinx()
```

```
[ ]  ax1a = ax1.twinx()
     sns.lineplot(x = winter.index, y= winter["1"], data = winter, color="red", ax=ax1)
     ax1.set(ylim=(0, 80))
     ax1.set_ylabel("Streamflow [$m^3$/s]")
     sns.lineplot(x = winter.index, y= winter["precip"], data = winter, color="blue", ax=ax1a)
     ax1a.set(ylim=(0, 400))
     ax1a.invert_yaxis()
     ax1a.fill_between(winter.index, 0, winter["precip"], alpha = 0.8)
```

```
[ ]  ax1a.set_ylabel("Кількість опадів, мм")
     ax2a = ax2.twinx()
     sns.lineplot(x = spring.index, y= spring["1"], data = spring, color="red", ax=ax2)
     ax2.set(ylim=(0, 80))
     ax2.set_ylabel("Річковий стік, м3/с")
     sns.lineplot(x = spring.index, y= spring["precip"], data = spring, color="blue", ax=ax2a)
     ax2a.set(ylim=(0, 400))
     ax2a.invert_yaxis()
```

```
[ ]  ax2a.fill_between(spring.index, 0, spring["precip"], alpha = 0.8)
     ax2a.set_ylabel("Кількість опадів, мм")
     ax3a = ax3.twinx()
     sns.lineplot(x = summer.index, y= summer["1"], data = summer, color="red", ax=ax3)
     ax3.set(ylim=(0, 80))
     ax3.set_ylabel("Річковий сток, м3/с")
     sns.lineplot(x = summer.index, y= summer["precip"], data = summer, color="blue", ax=ax3a)
     ax3a.set(ylim=(0, 400))
     ax3a.invert_yaxis()
```

```
[ ]  ax3a.fill_between(summer.index, 0, summer["precip"], alpha = 0.8)
     ax3a.set_ylabel("Кількість опадів, мм")
     ax4a = ax4.twinx()
     sns.lineplot(x = autumn.index, y= autumn["1"], data = autumn, color="red", ax=ax4)
     ax4.set(ylim=(0, 80))
     ax4.set_ylabel("Річковий сток, м3/с")
     sns.lineplot(x = autumn.index, y= autumn["precip"], data = autumn, color="blue", ax=ax4a)
     ax4a.set(ylim=(0, 400))
     ax4a.invert_yaxis()
     ax4a.fill_between(autumn.index, 0, autumn["precip"], alpha = 0.8)
     ax4a.set_ylabel("Кількість опадів, мм")
```

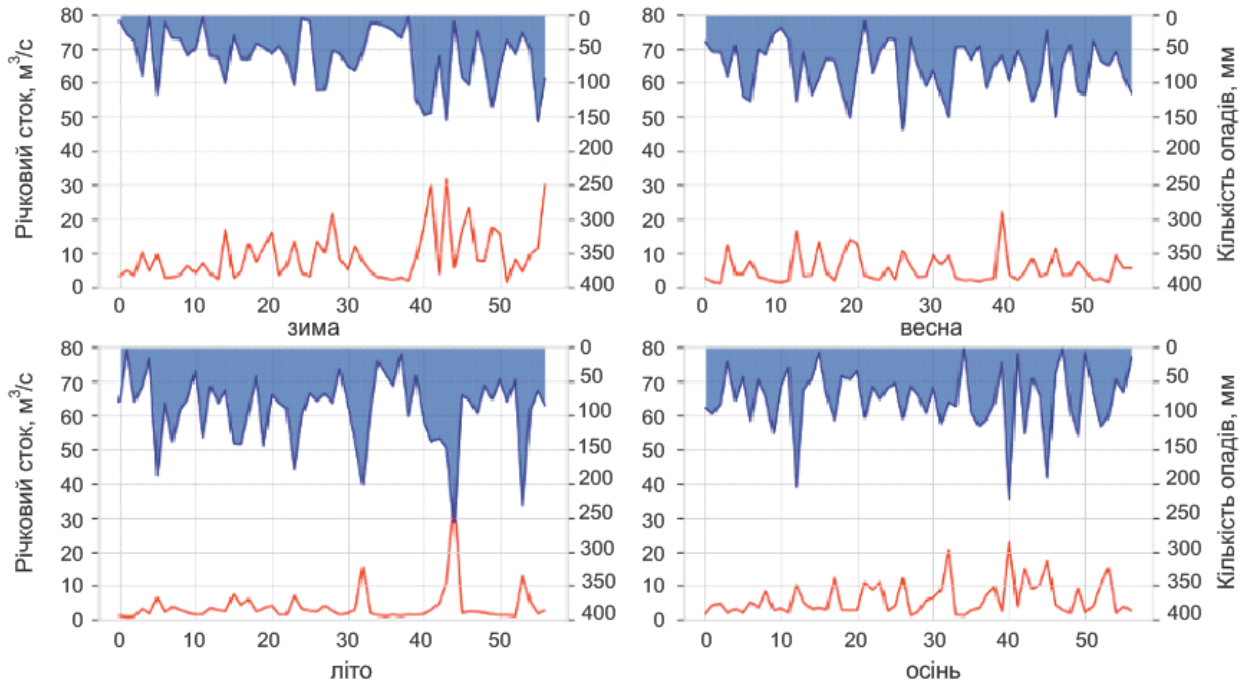


Рис. 4.5. Вплив сезонного фактору на річковий сток та кількість опадів.

Дані представлені в чотирьох підграфіках, де синя область представляє собою кількість опадів, а червона лінія представляє дані річкового стоку. На осі X показано час у місяцях, починаючи з першого місяця сезону 2000 року (три місяці з 2000 року, три місяці з 2001 року і т.д.).

З цих даних видно, що влітку та восени спостерігається найбільша кількість опадів, особливо під час літніх злив, при цьому відбувається значний підйом рівня води в річці. Восени опади випадають частіше, тому на ділянці річкового стоку також спостерігають більше пікових значень. Річковий стік зимою може мати вищі значення при невеликій кількості опадів через танення снігу протягом перших трьох місяців

року. Графіки весняного сезону чимось схожі на осінні, з меншою кількістю опадів, а значення річкових стоків досить подібні.

ВИСНОВКИ ДО РОЗДІЛУ 4

Розроблено та реалізовано інформаційно-аналітичну систему для оцінки ризику підняття рівня води в річці, в якій враховуються погодні умови, місцевість. Вона може бути реалізована на практиці. При використанні реклами та оголошень для просування програмного продукту можна досягти успіху та отримати пристойний дохід.

На їх основі отримано прогноз підвищеного рівня води, який можна порівняти зі значеннями історичних даних по річках. Крім цього, за допомогою запропонованого методу можливе отримання індивідуального прогнозу для окремих населених пунктів. Розроблений метод прогнозування може бути використаний для робіт із запобігання наслідкам сезонних затоплень населених пунктів у період повені.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

5.1. Опис проекту інформаційної системи

Розглянемо, які підготовчі дії слід провести перед тим, як почати втілювати свій проект як стартап. Для цього потрібно створити інформаційну карту для проекту, що розробляється. Це є першим кроком. Ці дані вказані в табл. 5.1, в якій приведені основні характеристики проекту та витрати на нього.

Табл. 5.1. Карта інформаційної системи

Назва номінації	Python програма
Назва проекту	Математичне та програмне забезпечення системи проведення гідрологічних прогнозів водних об’єктів
Назва ВНЗ, факультету, спеціальності	НЛТУ, кафедра інформаційних технологій, 122 «Комп’ютерні науки»
Прізвище, ім’я, по-батькові	Штипук Остап Васильович
Цілі і задачі проекту	Ціль проекту – розробка та реалізація системи для аналізу стану водних об’єктів та можливості його прогнозування. Задачі проекту: • розробити математичне забезпечення та алгоритм функціонування системи; • реалізувати програмну модель системи аналізу стану водних об’єктів; • провести дослідження динаміки підняття рівня води у водному об’єкті.

Короткий зміст проекту	Проаналізовано існуючі гідрометеорологічні системи прогнозування, запропоновано підхід до прогнозування гідрологічних часових рядів. На їх основі отримано прогноз підвищеного рівня води, який можна порівняти зі значеннями історичних даних по річках. Крім цього, за допомогою запропонованого методу можливе отримання індивідуального прогнозу для окремих населених пунктів. Розроблений метод прогнозування може бути використаний для робіт із запобігання наслідкам сезонних затоплень населених пунктів у період повені.
Терміни виконання проекту	12 місяців
Бюджет проекту	150 000 грн.

5.2. Стратегія проекту

Використання результатів фундаментальних та прикладних досліджень у різних галузях народного господарства є основою економіки та її динамічного розвитку. Таким чином, актуальність комерціалізації програмного продукту – процесу розробки та реалізації низки заходів, за допомогою яких результати наукових досліджень та дослідно-конструкторських розробок можна запропонувати на ринках товарів та послуг із комерційними цілями – очевидна. Комерціалізація програмного продукту передбачає пошук та відбір розробок для фінансування, залучення інвестицій, впровадження розробок.

Дана продукція, а точніше програмний продукт, реалізована мовою програмування Python, використовуючи методи машинного навчання. Вона дозволяє знизити витрати підприємства на закупівельну діяльність з допомогою використання прогнозування ціни.

На сьогоднішній день головною метою є отримання прибутку. Відповідно для того, щоб вибрати той чи інший спосіб комерціалізації, необхідно проаналізувати витрати та доходи, які понесе підприємство, вибравши його.

Проаналізувавши різні аспекти комерціалізації стартапів, можна дійти невтішного висновку про те, що:

- самостійне використання інновації дозволить підприємству максимізувати свій прибуток, але при цьому даний метод є і самим витратним;
- частковий продаж дозволить освоїти нові ринки за рахунок ліцензії, а також повернути частину витрачених коштів;
- при повній передачі прав дохід може бути порівняно із доходом від самостійного використання, але при цьому компанії доведеться змінювати.

5.3. Розробка програми стартап проекту

Табл. 5.2. Основні переваги та концепції інформаційної системи

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	моделювання підвищення рівня води	створення програмних модулів	вигода для кінцевого споживача
2	визначення параметрів рівня води та потужності стоку	можливість підключення датчиків оповіщення про підвищення рівня рівня води	використання безпілотних літальних апаратів
3	інтерфейс	інтерфейс, з допомогою якого можна вводити всі дані для можливості прогнозування ризику виникнення затоплення прилеглих територій	можливість прогнозування ймовірності виникнення підвищення рівня води вна досліджуваному об’єкті

Табл. 5.3. Опис трьох рівнів аналітичної системи для прогнозування стану водних об’єктів

рівні товару	**сутність та її складові**
1. Програмний продукт за задумом	моделювання стану водних об’єктів та прогнозування ймовірності виникнення повені
2. Програмний продукт, який має бути реально виконаний	інформаційно-аналітична система для моделювання рівня води рік
3. Підкріплення	служба для підтримки системи прийняття рішень в надзвичайній ситуації за допомогою даної інформаційної системи

Табл. 5.4. Визначення меж для встановлення ціни на інформаційну систему

№ п/п	рівень цін на товари замінники	рівень цін на товари-аналоги	рівень доходів цільової групи споживачів	верхня та нижня межі встановлення ціни на товар/послугу
1	наперед не задано	наперед не задано	300$+	300/150 $

ВИСНОВКИ ДО РОЗДІЛУ 5

Однією з найважливіших гідрологічних величин є стік водних об’єктів та його просторово-часові коливання. Він визначає розвиток та функціонування водних та навколоводних екосистем, має істотне значення при вирішенні водогосподарських завдань.

Розроблено та реалізовано інформаційно-аналітичну систему для оцінки ризику підняття рівня води в річці, в якій враховуються погодні умови, місцевість. Вона може бути реалізована на практиці. При використанні реклами та оголошень для просування програмного продукту можна досягти успіху та отримати пристойний дохід.

На їх основі отримано прогноз підвищеного рівня води, який можна порівняти зі значеннями історичних даних по річках. Крім цього, за допомогою запропонованого методу можливе отримання індивідуального прогнозу для окремих населених пунктів. Розроблений метод прогнозування може бути використаний для робіт із запобігання наслідкам сезонних затоплень населених пунктів у період повені.

Отримані результати мають практичне застосування для раціонального використання природних ресурсів і можуть бути науковою основою під час виконання проектів у галузі гідрологічних та екологічних досліджень. Значення витрат річкового стоку можуть бути використані при проектуванні гідротехнічних споруд та меліоративних систем. Дана інформаційна система може бути впроваджена в структурні підрозділи гідрометеорологічних постів.

СПИСОК ЛІТЕРАТУРИ

1. Нильсен Эйлин. Практический анализ временных рядов: прогнозирование со статистикой и машинное обучение. – СПб.: Диалектика, 2021. – 544 с.

2. Бэрри Пол. Изучаем программирование на Python. – М.: Эксмо, 2017. – 611 с.

3. Бизли Д. Python. Подробный справочник. СПб.: Символ-Плюс, 2010. – 864 с.

4. Абдрахманов М.И. Python. Визуализация данных: Matplotlib, Seaborn, Mayavi. – Devpractice.ru, 2020. – 413 с.

5. Васильев А. Python на примерах. Практический курс по программированию. – М.: Наука и техника, 2016. – 432 с.

6. Грас Джоэл. Data Science. Наука о данных с нуля. – СПб: БХВ-Петербург, 2021. – 418 с.

7. Дауни А.Б. Байесовские модели. Байесовская статистика на языке программирования Python. – М.: ДМК Пресс, 2018. – 182 с.

8. Иванченко В.В., Крук Ю.С., Марченко Л.Н., Летунович Ю.Е. Языки программирования. Python. – Минск: Белорусский национальный технический университет, 2021. – 91 с.

9. Копец Дэвид. Классические задачи Computer Science на языке Python. – СПб.: Питер, 2020. – 256 с.

10. Коэльо Л.П., Ричард В. Построение систем машинного обучения на языке Python. – Москва: ДМК Пресс, 2016. – 300 с.

11. Маккинни У. Python и анализ данных. – М.: ДМК Пресс, 2015. – 482 с.

12. Мюллер Джон Пол, Массарон Лука. Python и наука о данных для чайников. – СПб.: Диалектика, 2020. – 514 с.

13. Невский В.В., Копац Л.П., Смирнов Ю.С. Гидравлика. Гидрология. Гидрометрия. – М.: Транспорт. 1988. – 231 с.

14. Шелутко В.А. Численные методы прогнозов в гидрологии. – Л.: Гидрометеоиздат, 1991. – 238 с.

15. Догановский А.М., Орлов В.Г. Сборник задач по определению основных характеристик водных объектов суши (практикум по гидрологии). – Изд-во РГГМУ, Санкт-Петербург, 2011. – 315 стр.

16. Вершинин Д.А., Паромов В.В. Методы проведения гидрометрических работ на реке. – Томск: ТГУ, 2012. – 108с.

17. Волчек А.А. и др. Инженерная гидрология и регулирование стока. Общая гидрология и гидрометрия. – Горки: Белорусская государственная сельскохозяйственная академия, 2021. – 152 с.

18. Железняков Г.В., Неговская Т.А., Овчаров Е.Е. Гидрология, гидрометрия и регулирование стока. – М.: Колос, 1984. – 205 с.

19. Карасёв И.Ф. Речная гидрометрия и учет водных ресурсов. – Гидрометеоиздат, 1980г. – 310 с.

20. Клибашев К.П., Горошков И.Ф. Гидрологические расчеты. – Л.: Гидрометиздат, 1970. – 184 с.

ВИСНОВКИ

В дипломній роботі розроблено та реалізовано інформаційно-аналітичну систему для оцінки стану досліджуваного водного об’єкта, в якій враховуються погодні умови, місцевість.

В першому розділі проаналізовано предметну область, засоби та технології проектування інформаційної системи стану водних об’єктів. З її допомогою можна буде моделювати стан водних обєктів та прогнозувати рівень підняття води внаслідок несприятливих метеорологічних факторів.

У другому розділі приведено відомості про існуючі системи прогнозування та визначення параметрів стану водних об’єктів. Описано принципи їх функціонування, технології їх розробки. Розглянуто методи та засоби, з допомогою яких проектується інформаційна система по аналізу та прогнозуванню стану водних об’єктів. Приведено відомості про особливості проектування програмного продукту засобами Python, обробки даних з допомогою бібліотеки Pandas, візуалізації результатів з допомогою бібліотеки Seaborn.

В третьому розділі приведено дані про принципи побудови алгоритму математичної моделі та підходів до прогнозування підйому рівня води, що спричинене опадами дощу. На основі приведеної математичної моделі спроектовано інформаційну систему для аналізу стану водного об’єкта. Проведені дослідження та моделювання підтверджують достовірність розробленої математичної моделі інформаційної системи та можливості прогнозування стану водного об’єкта.

В четвертому розділі розроблено та реалізовано інформаційно-аналітичну систему для оцінки рівня підняття води в досліджуваному водному об’єкті, в якій враховуються погодні умови, місцевість. На їх основі отримано прогноз підвищеного рівня води, який можна порівняти зі значеннями історичних даних по даному об’єкту. Крім цього, за допомогою запропонованого методу можливе отримання прогнозу для окремих населених пунктів. Розроблений метод прогнозування може бути використаний для робіт із запобігання наслідкам сезонних затоплень населених пунктів у період повені.

Отримані результати мають практичне застосування для раціонального використання природних ресурсів і можуть бути науковою основою під час виконання проектів у галузі гідрологічних та екологічних досліджень. Значення витрат річкового стоку можуть бути використані при проектуванні гідротехнічних споруд та меліоративних систем. Дана інформаційна система може бути впроваджена в структурні підрозділи гідрометеорологічних постів.

ДОДАТКИ

ДОДАТОК А

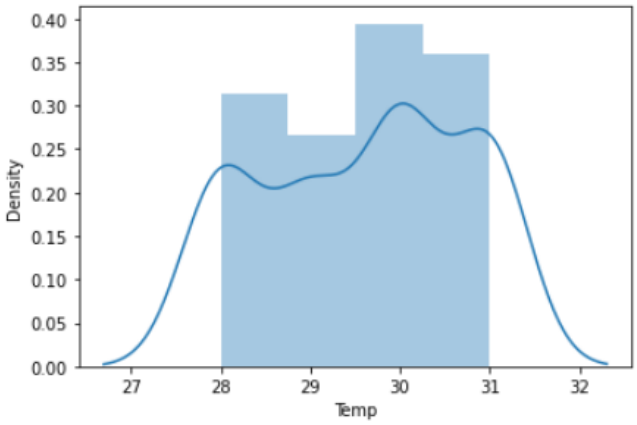
1.ipynb

```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[ ]  dataset = pd.read_excel('flood dataset.xlsx')
```

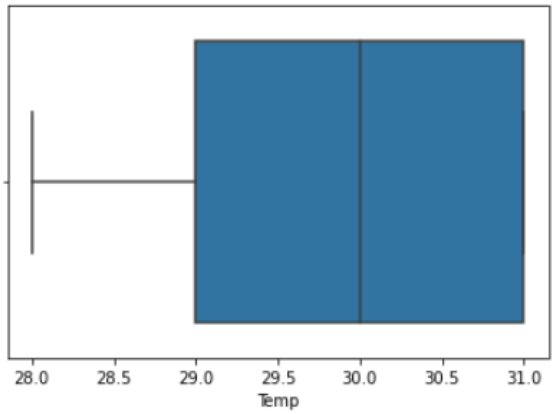
```
[ ]  print(sns.distplot(dataset["Temp"]))
```

```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated
  warnings.warn(msg, FutureWarning)
AxesSubplot(0.125,0.125;0.775x0.755)
```

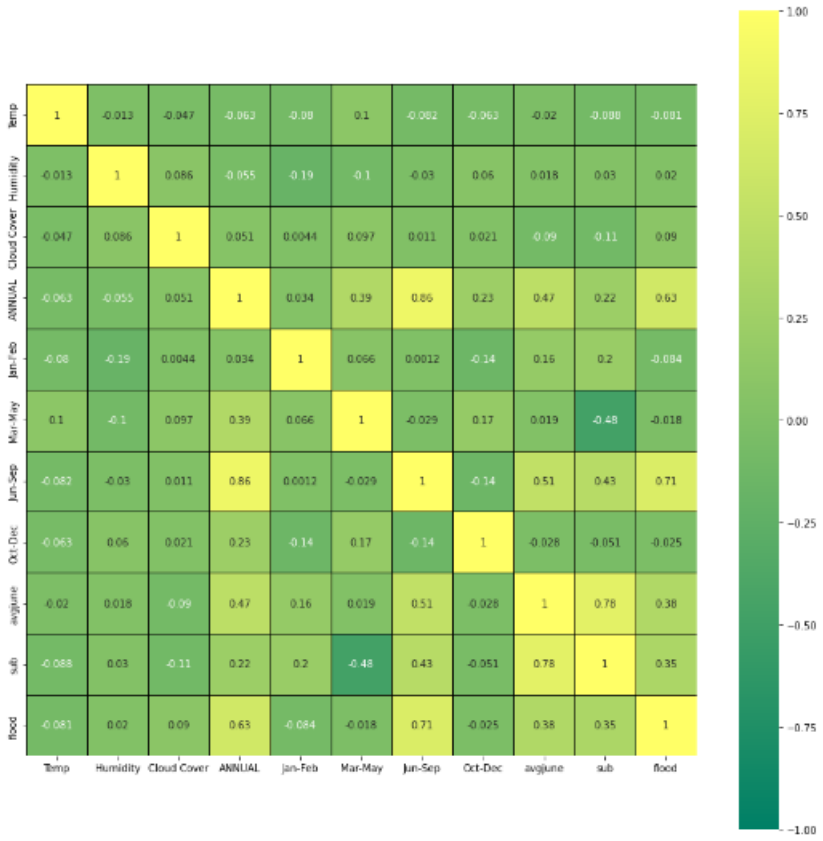



```
[ ]  print(sns.boxplot(dataset['Temp']))
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as
  warnings.warn(
AxesSubplot(0.125,0.125;0.775x0.755)
```



```
[ ]  import seaborn as sns
     fig=plt.gcf()
     fig.set_size_inches(15,15)
     fig=sns.heatmap(dataset.corr(),annot=True,cmap='summer',
                     linewidths=1,linecolor='k',square=True,
                     mask=False, vmin=-1, vmax=1,
                     cbar_kws={"orientation": "vertical"},cbar=True)
```




```
[ ]  dataset.drop(["Oct-Dec"],axis=1,inplace=True)
```

```
[ ]  dataset.head(10)
```

	Temp	Humidity	Cloud Cover	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	avgjune	sub	flood
0	29	70	30	3248.6	73.4	386.2	2122.8	274.866667	649.9	0
1	28	75	40	3326.6	9.3	275.7	2403.4	130.300000	256.4	1
2	28	75	42	3271.2	21.7	336.3	2343.0	186.200000	308.9	0
3	29	71	44	3129.7	26.7	339.4	2398.2	366.066667	862.5	0
4	31	74	40	2741.6	23.4	378.5	1881.5	283.400000	586.9	0
5	30	70	38	2708.0	34.1	230.0	1943.1	138.300000	254.1	0
6	29	74	40	3671.1	23.7	328.0	2737.8	256.966667	669.5	1
7	30	78	36	2648.3	28.8	283.7	2023.6	197.533333	450.0	0
8	30	71	40	3050.2	65.9	628.3	1940.4	234.900000	231.5	0
9	30	70	34	2848.6	28.4	296.7	1886.5	226.666667	531.2	0

```
[ ]  print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115 entries, 0 to 114
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Temp         115 non-null    int64  
 1   Humidity     115 non-null    int64  
 2   Cloud Cover  115 non-null    int64  
 3   ANNUAL       115 non-null    float64
 4   Jan-Feb      115 non-null    float64
 5   Mar-May      115 non-null    float64
 6   Jun-Sep      115 non-null    float64
 7   avgjune      115 non-null    float64
 8   sub          115 non-null    float64
 9   flood        115 non-null    int64  
dtypes: float64(6), int64(4)
memory usage: 9.1 KB
None
```

```
[ ] dataset.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Temp	115.0	29.600000	1.122341	28.0	29.000000	30.000000	31.000000	31.000000
Humidity	115.0	73.852174	2.947623	70.0	71.000000	74.000000	76.000000	79.000000
Cloud Cover	115.0	36.286957	4.330158	30.0	32.500000	36.000000	40.000000	44.000000
ANNUAL	115.0	2925.487826	422.112193	2068.8	2627.900000	2937.500000	3164.100000	4257.800000
Jan-Feb	115.0	27.739130	22.361032	0.3	10.250000	20.500000	41.600000	98.100000
Mar-May	115.0	377.253913	151.091850	89.9	276.750000	342.000000	442.300000	915.200000
Jun-Sep	115.0	2022.840870	386.254397	1104.3	1768.850000	1948.700000	2242.900000	3451.300000
avgjune	115.0	218.100870	62.547597	65.6	179.666667	211.033333	263.833333	366.066667
sub	115.0	439.801739	210.438813	34.2	295.000000	430.600000	577.650000	982.700000
flood	115.0	0.139130	0.347597	0.0	0.000000	0.000000	0.000000	1.000000

```
[ ] dataset.isnull().any()
```

```
Temp           False
Humidity       False
Cloud Cover    False
ANNUAL         False
Jan-Feb        False
Mar-May        False
Jun-Sep        False
avgjune        False
sub            False
flood          False
dtype: bool
```

```
[ ] X = dataset.iloc[:,2:7].values
```

```
[ ] y = dataset.iloc[:,9:].values
```

```
[ ] from sklearn.model_selection import train_test_split
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=10)
```

```
[ ] X_train.shape
```

```
(86, 5)
```

```
[ ] X_test.shape
```

```
(29, 5)
```

```
[ ] from sklearn.preprocessing import StandardScaler
    sc=StandardScaler()
    X_train=sc.fit_transform(X_train)
    X_test=sc.fit_transform(X_test)
```

```
[ ] from joblib import dump
    dump(sc,"transform.save")
```

```
['transform.save']
```

```
[ ] from sklearn.tree import DecisionTreeClassifier
```

```
[ ] model = DecisionTreeClassifier()
```

```
[ ] model.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```
[ ] from sklearn.metrics import accuracy_score,classification_report
```

```
[ ] y_predict = model.predict(X_test)
    y_predict_train=model.predict(X_train)
```

```
[ ] print('Test data',accuracy_score(y_test,y_predict))
    print('Train data',accuracy_score(y_train,y_predict_train))
```

```
Test data 0.9655172413793104
Train data 1.0
```

pd.crosstab(y_test,y_predict)

```
[ ] print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        26
           1       0.75      1.00      0.86         3

    accuracy                           0.97        29
   macro avg       0.88      0.98      0.92        29
weighted avg       0.97      0.97      0.97        29
```

```
[ ] from sklearn.ensemble import RandomForestClassifier
```

```
[ ] model = RandomForestClassifier()
    model.fit(X_train,y_train.ravel())
```

```
RandomForestClassifier()
```

```
[ ] from sklearn.metrics import accuracy_score,classification_report
```

```
[ ] y_predict = model.predict(X_test)
    y_predict_train=model.predict(X_train)
```

```
[ ] print('Test data',accuracy_score(y_test.ravel(),y_predict))
    print('Train data',accuracy_score(y_train.ravel(),y_predict_train))
```

```
Test data 0.9655172413793104
Train data 1.0
```

```
[ ] pd.crosstab(y_test.ravel(),y_predict)
```

col_0	0	1
row_0		
0	25	1
1	0	3

```
[ ] print(classification_report(y_test.ravel(), y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        26
           1       0.75      1.00      0.86         3

    accuracy                           0.97        29
   macro avg       0.88      0.98      0.92        29
weighted avg       0.97      0.97      0.97        29
```

```
[ ] from sklearn.neighbors import KNeighborsClassifier

    knn=KNeighborsClassifier()

    knn.fit(X_train,y_train.ravel())
```

```
KNeighborsClassifier()
```

```
[ ] y_predict=knn.predict(X_test)
    y_pred=knn.predict(X_train)
```

```
[ ] from sklearn.metrics import accuracy_score,classification_report

    print("Test accuracy=", accuracy_score(y_test.ravel(),y_predict))
    print("Train accuracy=", accuracy_score(y_train.ravel(),y_pred))
```

```
Test accuracy= 0.896551724137931
Train accuracy= 0.9418604651162791
```

```
[ ] pd.crosstab(y_test.ravel(),y_predict)
```

col_0	0	1
row_0		
0	23	3
1	0	3

```
[ ] print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.88      0.94        26
           1       0.50      1.00      0.67         3

    accuracy                           0.90        29
   macro avg       0.75      0.94      0.80        29
weighted avg       0.95      0.90      0.91        29
```

```
[ ] from xgboost import XGBClassifier
```

```
[ ] xgb = XGBClassifier()
```

```
[ ] xgb.fit(X_train,y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
```

```
[ ] y_predict=xgb.predict(X_test)
    y_pred=xgb.predict(X_train)
```

```
[ ] from sklearn.metrics import accuracy_score,classification_report

    print("Test accuracy=", accuracy_score(y_test,y_predict))
    print("Train accuracy=", accuracy_score(y_train,y_pred))
```

```
Test accuracy= 0.9655172413793104
Train accuracy= 1.0
```

```
[ ] pd.crosstab(y_test.ravel(),y_predict)
```

col_0	0	1
row_0		
0	25	1
1	0	3

```
[ ] print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        26
           1       0.75      1.00      0.86         3

    accuracy                           0.97        29
   macro avg       0.88      0.98      0.92        29
weighted avg       0.97      0.97      0.97        29
```

```
[ ] from sklearn import tree
    from sklearn import ensemble
    from sklearn import neighbors
    import xgboost
```

```
[ ] dtree = tree.DecisionTreeClassifier()
    Rf = ensemble.RandomForestClassifier()
    knn = neighbors.KNeighborsClassifier()
    xgb = xgboost.XGBClassifier()
```

```
[ ] dtree = tree.DecisionTreeClassifier()
    Rf.fit(X_train,y_train.ravel())
    knn.fit(X_train,y_train.ravel())
    xgb.fit(X_train,y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
```

```
[ ] from sklearn import metrics
```

```
[ ] print(metrics.accuracy_score(y_test,y_predict))
    print(metrics.accuracy_score(y_test,y_predict))
    print(metrics.accuracy_score(y_test,y_predict))
    print(metrics.accuracy_score(y_test,y_predict))
```

```
0.9655172413793104
0.9655172413793104
0.9655172413793104
0.9655172413793104
```

```
[ ] metrics.confusion_matrix(y_test,y_predict)
```

```
array([[25,  1],
       [ 0,  3]], dtype=int64)
```

```
[ ] print(metrics.accuracy_score(y_test,y_predict))
```

```
0.9655172413793104
```

```
[ ] print(metrics.precision_score(y_test,y_predict))
```

```
0.75
```

```
[ ] print(metrics.recall_score(y_test,y_predict))
```

```
1.0
```

ДОДАТОК Б

2.ipynb

```
[ ]  import matplotlib.pyplot as plt
     import pandas as pd
     import seaborn as sns
     sns.set(style="whitegrid")
```

```
[ ]  d1 = pd.read_csv("d1.csv", names = ["Date", "Rainfall"])
     d1['Date'] = pd.to_datetime(d1["Date"])
     d1 = d1.set_index('Date')
     d1 = d1.resample("Y").sum()
```

```
[ ]  ax = sns.barplot(x=d1.index, y=d1["Rainfall"], color = "royalblue")
     ax.set_xlabel("Роки")
     ax.set_ylabel("Кількість опадів, мм")
     ax.set_xticklabels(labels = d1.index.year, rotation = 45)
```

```
[ ]  import matplotlib.pyplot as plt
     import pandas as pd
     import seaborn as sns
     sns.set(style="whitegrid")
```

```
[ ]  d2 = pd.read_csv("d2.csv", usecols = ["Date", "1", "2", "3", "4", "5", "6", "7", "8"])
     d2["Date"] = pd.to_datetime(d2["Date"])
     d2 = d2.set_index("Date")
     d2 = d2.resample("Y").sum()
```

```
[ ]  fig1, ax = plt.subplots(2,4, figsize = (18,8), tight_layout=True, sharex = True)

     for i, (ax, value, name) in enumerate(zip(ax.flatten(), d2.values.T, d2.columns)):
         ax.set_xticks(d2.index.year)
         ax.set_xticklabels(labels = d2.index.year.values, rotation=60)
         ax.bar(d2.index.year, d2[name], label=name, color = "royalblue")
         ax.hlines(d2[name].mean(), 1999.5, 2018.5, color="green", label="медіана",
     ls = "--")
         ax.hlines(d2[name].median(), 1999.5, 2018.5, color="red", label="ср.знач",
     ls = ":")
         if i >3:
             ax.set_xlabel ("Рік")
         ax.set_ylabel("Кількість опадів, мм")
         ax.legend(shadow=True, loc="upper left")
```

```
[ ]  import matplotlib.pyplot as plt
     from matplotlib import cm
     import pandas as pd
     import seaborn as sns
     import numpy as np
     from scipy.interpolate import griddata
```

```
[ ]  meteo_loc = pd.read_csv("meteo_stat.csv", usecols = ["name", "real_name", "lat", "long"]).sort_values(by = ["name"])
     d2 = pd.read_csv("d2.csv")

     d2 = pd.DataFrame(d2.mean(axis = 0)).rename_axis('name').reset_index().sort_values(by=["name"]).rename(columns =
     {0: "Mean"})
```

```
[ ]  meteo_loc = pd.merge(meteo_loc, d2, on="name")
     X,Y = np.meshgrid(np.linspace(meteo_loc["long"].min()-0.05,meteo_loc["long"].max()+0.05,2000),
                       np.linspace(meteo_loc["lat"].min()-0.05,meteo_loc["lat"].max()+0.05,2000))
```

```
[ ]  grid_rain = sp.interpolate.Rbf(meteo_loc["long"],meteo_loc["lat"], meteo_loc["Mean"])
     spatial_rain = grid_rain(X,Y)
     fig, ax = plt.subplots(figsize = (16, 8))
     ax.scatter(meteo_loc["long"], meteo_loc["lat"], color = "black")
```

```
[ ]  for i, (x,y) in enumerate(zip (meteo_loc["long"], meteo_loc["lat"])):
         label = meteo_loc["real_name"][i]
         ax.annotate(label, (x,y), textcoords = "offset points", xytext = (2,3))
     fig.colorbar(gg, format = '%.1f')
```

```
[ ]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[ ]  sns.set_style("whitegrid")
     d2 = pd.DataFrame(pd.read_csv("d2.csv", parse_dates=['Date']).mean(axis = 1, numeric_only=True), columns = ["precip"])
     streamflow = pd.read_csv("streamflow.csv", parse_dates=['Date'])
     streamflow = pd.concat([streamflow, d2], axis = 1)
     streamflow = streamflow.resample("M", on="Date").agg({"1": np.mean, "2": np.sum})
```

```
[ ]  fig, ax1 = plt.subplots(figsize=(14,6))
     ax2 = ax1.twinx()
     sns.lineplot(x = streamflow.index, y= "precip", data = streamflow, color="blue", ax=ax2)
```

```
[ ]  ax2.fill_between(streamflow.index, 0, streamflow["precip"], alpha = 0.8)
     ax2.set(ylim=(0, 400))
     ax2.set_ylabel("Кількість опадів, мм")
     ax2.invert_yaxis()
     sns.lineplot(x = streamflow.index, y= "1", data = streamflow, color="red", ax=ax1)
     ax1.set(ylim=(0, 80))
     ax1.set_ylabel("Річковий стік, м3/с")
     ax1.set_xlabel("Роки")
```

```
[ ]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     sns.set_style("whitegrid")
```

```
[ ]  d2 = pd.DataFrame(pd.read_csv("d2.csv", parse_dates=['Date']).mean(axis = 1, numeric_only=True), columns = ["precip"])
     streamflow = pd.read_csv("streamflow.csv", parse_dates=['Date'])
     streamflow = pd.concat([streamflow, d2], axis = 1).set_index(["Date"])
     winter = streamflow[(streamflow.index.month >= 1) &  (streamflow.index.month <= 3)]
     winter = winter.reset_index().resample("M", on="Date").agg({"1": np.mean, "precip": np.sum}).dropna().reset_index()
```

```
[ ]  spring = streamflow[(streamflow.index.month >= 4) &  (streamflow.index.month <= 6)]
     spring = spring.reset_index().resample("M", on="Date").agg({"1": np.mean, "precip": np.sum}).dropna().reset_index()
     summer = streamflow[(streamflow.index.month >= 7) &  (streamflow.index.month <= 9)]
     summer = summer.reset_index().resample("M", on="Date").agg({"1": np.mean, "precip": np.sum}).dropna().reset_index()
```

```
[ ]  autumn= streamflow[(streamflow.index.month >= 10) &  (streamflow.index.month <= 12)]
     autumn = autumn.reset_index().resample("M", on="Date").agg({"1": np.mean, "precip": np.sum}).dropna().reset_index()
     fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows = 2, ncols = 2, figsize = (14,8), tight_layout=True)
     ax1a = ax1.twinx()
```

```
[ ]  ax1a = ax1.twinx()
     sns.lineplot(x = winter.index, y= winter["1"], data = winter, color="red", ax=ax1)
     ax1.set(ylim=(0, 80))
     ax1.set_ylabel("Streamflow [$m^3$/s]")
     sns.lineplot(x = winter.index, y= winter["precip"], data = winter, color="blue", ax=ax1a)
     ax1a.set(ylim=(0, 400))
     ax1a.invert_yaxis()
     ax1a.fill_between(winter.index, 0, winter["precip"], alpha = 0.8)
```

```
[ ]  ax1a.set_ylabel("Кількість опадів, мм")
     ax2a = ax2.twinx()
     sns.lineplot(x = spring.index, y= spring["1"], data = spring, color="red", ax=ax2)
     ax2.set(ylim=(0, 80))
     ax2.set_ylabel("Річковий сток, м3/с")
     sns.lineplot(x = spring.index, y= spring["precip"], data = spring, color="blue", ax=ax2a)
     ax2a.set(ylim=(0, 400))
     ax2a.invert_yaxis()
```

```
[ ]  ax2a.fill_between(spring.index, 0, spring["precip"], alpha = 0.8)
     ax2a.set_ylabel("Кількість опадів, мм")
     ax3a = ax3.twinx()
     sns.lineplot(x = summer.index, y= summer["1"], data = summer, color="red", ax=ax3)
     ax3.set(ylim=(0, 80))
     ax3.set_ylabel("Річковий сток, м3/с")
     sns.lineplot(x = summer.index, y= summer["precip"], data = summer, color="blue", ax=ax3a)
     ax3a.set(ylim=(0, 400))
     ax3a.invert_yaxis()
```

```
[ ]  ax3a.fill_between(summer.index, 0, summer["precip"], alpha = 0.8)
     ax3a.set_ylabel("Кількість опадів, мм")
     ax4a = ax4.twinx()
     sns.lineplot(x = autumn.index, y= autumn["1"], data = autumn, color="red", ax=ax4)
     ax4.set(ylim=(0, 80))
     ax4.set_ylabel("Річковий сток, м3/с")
     sns.lineplot(x = autumn.index, y= autumn["precip"], data = autumn, color="blue", ax=ax4a)
     ax4a.set(ylim=(0, 400))
     ax4a.invert_yaxis()
     ax4a.fill_between(autumn.index, 0, autumn["precip"], alpha = 0.8)
     ax4a.set_ylabel("Кількість опадів, мм")
```