МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

та завдання до лабораторних робіт з курсу
«Методи та інструменти системного аналізу»
для студентів денної та заочної форм навчання спеціальності
F4 – «Системний аналіз та наука про дані»

2025

Методичні вказівки та завдання до лабораторних робіт з курсу «Методи та інструменти системного аналізу» для студентів денної та заочної форм навчання спеціальності F4 – «Системний аналіз та наука про дані» / Укл.: Д.В. Широкорад., Е.В. Терещенко - Запоріжжя: НУ «Запорізька політехніка», 2025. - 41 с.

Укладачі**:** Д.В. Широкорад, доцент, к.ф.-м..н.,
Е.В. Терещенко, доц., д.ф.-м.н.

Рецензент: А.Є. Рябенко, доцент, к.ф.-м.н.

Відповідальний
за випуск Е.В. Терещенко, доцент, к.ф.-м.н.

Затверджено на засіданні кафедри
“Системний аналіз та
обчислювальна математика»
Протокол № 9 від 23.01.2025

Рекомендовано до видання
НМК ФКНТ
Протокол № 6 від 13.02.2025

ЗМІСТ

ВСТУП

Курс «Методи та інструменти системного аналізу» є ключовим для майбутніх спеціалістів, які працюватимуть у сферах інженерії, інформаційних технологій, бізнес-аналітики та інших галузей, де застосовується системний підхід. Сучасна наука та практика вимагають від фахівців не лише засвоєння теоретичних концепцій, але й набуття практичних навичок у розробці моделей, аналізі даних та впровадженні інформаційних технологій. Лабораторні роботи, що входять до курсу, спрямовані на інтеграцію теоретичних знань із практичними завданнями, дозволяючи студентам систематизувати матеріал та застосовувати його у вирішенні реальних проблем..

Метою цих методичних вказівок є надання студентам чіткої структури та покрокових інструкцій для виконання лабораторних робіт. Кожна лабораторна робота побудована таким чином, щоб допомогти студентам:

– засвоїти принципи планування експериментів та побудови регресійних моделей;

– ознайомитися з методами статистичного аналізу даних, зокрема, методом Монте-Карло;

– отримати практичні навички інтеграції систем через використання API;

– вивчити основи хмарних технологій для зберігання та обробки даних;

– розібратися з принципами роботи з Big Data, що є актуальними в сучасному аналізі інформації;

– ознайомитися з підходами автоматизованого машинного навчання (AutoML) як інструменту оптимізації моделей.

1 ЛАБОРАТОРНА РОБОТА №1. ТЕОРІЯ ПЛАНУВАННЯ ЕКСПЕРИМЕНТУ

1.1 Загальні теоретичні відомості

Основні поняття ТПЕ

- Генеральна сукупність і вибірка. Генеральна сукупність – це повний набір можливих результатів, а вибірка – це кінцевий набір спостережень, отриманих під час експерименту.

- Фактори та рівні. Фактори – незалежні змінні, які можуть впливати на вихідну (відгукову) змінну. Для кожного фактора задають певні рівні (наприклад, мінімальний і максимальний).

- Матриця планування експерименту. За допомогою кодування (часто використовується нормоване кодування з рівнями –1 та +1) формується матриця, де кожен рядок відповідає конкретному експериментальному досліду.

Види експериментальних планів

- Повний факторний експеримент. При n факторах та двох рівнях кожного, кількість дослідів дорівнює 2^n. Такий план дозволяє оцінити як головні ефекти, так і взаємодії.

- Дробовий факторний експеримент. Застосовується для зменшення кількості дослідів при збереженні інформації про головні ефекти, хоча деякі взаємодії можуть бути «змішаними».

- Ортогональні плани. Забезпечують незалежність оцінок коефіцієнтів моделі, що полегшує статистичний аналіз.

1. Оцінка коефіцієнтів регресії. За допомогою методу найменших квадратів (МНК) обчислюються коефіцієнти лінійної моделі, яка може мати вигляд:

$$y = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \ldots + b_n \cdot x_n + \varepsilon$$

2. Перевірка значущості коефіцієнтів. За допомогою t-критерію Стьюдента перевіряється нульова гіпотеза про те, що коефіцієнт рівний нулю.

3. Перевірка адекватності моделі. Застосовується F-критерій Фішера для порівняння залишкової дисперсії з дисперсією відтворюваності.

1.2 Завдання до лабораторної роботи

Вам необхідно спроектувати та проаналізувати експеримент, метою якого є вивчення впливу температури (фактор 1) та вологості

(фактор 2) на швидкість хімічної реакції (відгукова змінна). Завдання передбачає побудову повного факторного плану з двома рівнями для кожного фактора, збір (або симуляцію) даних, побудову регресійної моделі методом найменших квадратів, перевірку значущості отриманих коефіцієнтів та оцінку адекватності моделі.

1.3 Вказівки до виконання лабораторної роботи

Вибір факторів та їх рівнів

1. **Фактор 1 – Температура (T):**
   - Низький рівень (кодове значення –1): 20 °C
   - Високий рівень (кодове значення +1): 40 °C
2. **Фактор 2 – Відносна вологість (H):**
   - Низький рівень (кодове значення –1): 30 %
   - Високий рівень (кодове значення +1): 70 %

Побудова експериментального плану

1. Формування матриці планування

а) **Тип плану:**

Для двох факторів із двома рівнями кожного використовується повний факторний експеримент. Кількість базових дослідів = $2^2 = 4$.

б) **Нормоване кодування:**

Використовується стандартне кодування: –1 для низького рівня, +1 для високого рівня.

в) **Матриця планування:**

Додайте стовпець для фіктивного фактора x_0 (всі значення +1) для оцінки вільного члена.

Приклад матриці:

Дослід	x_0	x_1 (T)	x_2 (H)
1	+1	–1	–1
2	+1	–1	+1
3	+1	+1	–1
4	+1	+1	+1

2. Організація повторів та рандомізація

а) **Повтори:**

Для кожної комбінації факторних значень проведіть не менше 3 повторів, що дозволить оцінити випадковість та визначити дисперсію вимірювань.

Загальна кількість дослідів = 4 комбінації × 3 повтори = 12 дослідів.

б) **Рандомізація:**

Для зменшення впливу неконтрольованих факторів, сформуйте випадковий порядок проведення дослідів. Запишіть номер кожного досліду в окремій колонці, перемішайте рядки матриці планування за допомогою генератора випадкових чисел або вручну.

Проведення експерименту (або симуляція даних)

1. Збір даних

Симуляція даних (за потреби):**

Якщо експеримент проводити неможливо, згенеруйте дані за апроксимуючою моделлю, наприклад:

$$y = 5 + 2 \cdot x_1 + 3 \cdot x_2 + 1.5 \cdot (x_1 \cdot x_2) + \varepsilon$$

де x_1 та x_2 – нормовані значення (–1 або +1), а ε – випадкова похибка, розподілена нормально (наприклад, N(0, 0.5)). Створіть таблицю з симульованими значеннями для 12 дослідів.

2. Фіксація даних

Сформуйте таблицю з наступними колонками:

- Номер досліду.
- Значення факторів (x_1 та x_2) та x_0.
- Отримане значення y (швидкість реакції).

VI. Аналіз даних та побудова регресійної моделі

1. Розрахунок коефіцієнтів регресії

а) **Форма моделі:**

Побудуйте модель з взаємодією:

$$y = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + b_{12} \cdot (x_1 \cdot x_2) + \varepsilon$$

б) **Розрахунок коефіцієнтів методом найменших квадратів:**

Для кожного коефіцієнта використовуйте формулу:

$$b_j = (1/N) \cdot \sum (x_{ij} \cdot y_i)$$

де:

- $N = 12$ – загальна кількість дослідів,
- x_{ij} – значення j-го фактору (або добуток для взаємодії) у і-му досліді,

- y_i – отримане значення швидкості реакції.

Покажіть кожен крок обчислення (якщо використовуєте Excel, побудуйте таблиці з формулами, або використовуйте MathCad/MatLab для обчислень).

а) **Перевірка значущості коефіцієнтів (t-тест):**

Для кожного коефіцієнта обчисліть t-статистику:

$t = b_j / SE(b_j)$

де $SE(b_j)$ – стандартна помилка коефіцієнта, яку можна визначити з дисперсії залишків. Порівняйте отримане значення t з критичним t-значенням (наприклад, для 95% довірчого інтервалу).

б) **Аналіз залишків:**

Розрахуйте залишки:

$\varepsilon_i = y_i$ (виміряне) – y_i (розрахункове за моделлю)

Побудуйте графік залишків для перевірки нормальності їх розподілу (гістограма або Q-Q графік).

в) **Перевірка адекватності моделі (F-тест):**

Обчисліть F-статистику як співвідношення дисперсії моделі до залишкової дисперсії. Порівняйте з критичним значенням F для відповідних ступенів свободи.

3. Графічне представлення даних

а) **Графік регресійної моделі:**

Побудуйте графік, на якому зобразіть:

- Експериментальні точки (значення y для кожного досліду).
- Криву регресійної моделі, отриману з розрахованими коефіцієнтами.

б) **Графік залишків:**

Побудуйте діаграму розкиду залишків, щоб перевірити їх випадковий характер.

2 ЛАБОРАТОРНА РОБОТА №2. МЕТОД МОНТЕ-КАРЛО

2.1 Загальні теоретичні відомості

Метод Монте-Карло – це клас стохастичних чисельних методів, який базується на використанні випадкових чисел для апроксимації математичних обчислень. Назва методу походить від казино Монте-Карло, оскільки основна ідея полягає у використанні випадковості, що нагадує азартні ігри. Цей метод знаходить застосування в широкому спектрі наукових дисциплін – від фізики та інженерії до фінансів та статистики. Його особливістю є здатність вирішувати складні інтегральні задачі, моделювати випадкові процеси та аналізувати системи, де аналітичне розв'язання є неможливим або надто складним.

1. Суть методу Монте-Карло:

- Метод Монте-Карло ґрунтується на статистичному аналізі результатів чисельних експериментів. Він полягає у генерації великої кількості випадкових чисел (або випадкових подій) для апроксимації певного математичного значення, наприклад, інтегралу або розподілу ймовірностей.

- Завдяки принципу великої кількості (закон великих чисел), середнє значення результатів наближається до точного математичного значення при збільшенні кількості симуляцій.

Принципи методу Монте-Карло

1. Випадкова вибірка:

- Основою методу є генерація випадкових чисел. У сучасних комп’ютерних системах використовується псевдовипадкова генерація чисел, яка забезпечує достатню якість випадковості для більшості задач.

- Генерація чисел повинна задовольняти вимоги рівномірного розподілу, а при моделюванні нормальних процесів – нормального розподілу.

2. Апроксимація інтегралів:

- Один із класичних прикладів застосування методу Монте-Карло – чисельне обчислення визначених інтегралів. Наприклад, інтеграл може бути апроксимований як середнє значення функції, обчислене по випадково вибраних точках області інтегрування.

- Формула апроксимації інтегралу I має вигляд:

$$I \approx V \cdot (1/N) \cdot \sum(f(x_i)),$$

де V – об'єм області інтегрування, N – кількість випадкових точок, а $f(x_i)$ – значення функції в і-й точці.

3. Закон великих чисел:

- Ефективність методу Монте-Карло базується на законі великих чисел, згідно з яким середнє арифметичне великої кількості незалежних спостережень сходиться до математичного сподівання.

- Чим більше спостережень, тим точніше апроксимація.

4. Застосування і варіанти:

- Метод Монте-Карло застосовується для розв'язання різноманітних задач: оптимізаційних, інтегральних, симуляційних, аналізу систем з невизначеністю, фінансового моделювання та інших.

- Серед популярних варіантів – метод Монте-Карло для розв’язання диференціальних рівнянь, алгоритми оптимізації, оцінка статистичних характеристик та аналіз ймовірнісних розподілів.

Переваги:

- Простота реалізації алгоритму.

- Гнучкість у застосуванні до задач з високою вимірністю.

- Можливість вирішення задач, для яких традиційні аналітичні методи не застосовні.

- Легкість паралельних обчислень, що дозволяє значно зменшити час виконання симуляцій.

Недоліки:

- Висока обчислювальна складність для досягнення високої точності (вимагається велика кількість ітерацій).

- Залежність точності від якості генератора випадкових чисел.

- Можливість виникнення статистичних похибок, які потребують ретельного аналізу.

Метод Монте-Карло широко застосовується у таких галузях:

- Фізика: моделювання ядерних реакцій, квантова хімія, статистична механіка.

- Фінанси: оцінка ризиків, розрахунок вартості опціонів, аналіз портфеля.

- Інженерія: аналіз надійності систем, оптимізація конструкцій, розрахунок інтегралів.

- Статистика: оцінка розподілів ймовірностей, бутстреп-перевірки, симуляції випадкових процесів.

- Комп'ютерні науки: генерація випадкових тестів, оптимізаційні алгоритми, алгоритми машинного навчання.

2.2 Завдання до лабораторної роботи

1. Реалізувати метод Монте-Карло для обчислення числа **π**.
2. Дослідити вплив кількості випробувань на точність.
3. Порівняти результати з аналітичним значенням **π**.

2.3 Вказівки до виконання лабораторної роботи

Крок 1. Підготовка середовища

Встановіть Python 3.8+ та бібліотеки:

bash

Copy

```
pip install numpy matplotlib
```

Крок 2. Реалізація алгоритму

Створіть файл monte_carlo.py та додайте код:

```
import numpy as np
import matplotlib.pyplot as plt

def calculate_pi(num_points):
    # Генерація випадкових точок у квадраті [-1, 1] x [-1, 1]
    x = np.random.uniform(-1, 1, num_points)
    y = np.random.uniform(-1, 1, num_points)

    # Визначення точок всередині кола (x² + y² <= 1)
    inside_circle = (x**2 + y**2) <= 1
    points_inside = np.sum(inside_circle)

    # Обчислення π
    pi_estimate = 4 * points_inside / num_points
    return pi_estimate, x, y, inside_circle

# Параметри
```

```
    num_points = 10_000
    pi, x, y, inside = calculate_pi(num_points)

    # Вивід результату
    print(f"Оцінка π: {pi}")
    print(f"Точність:  {abs(pi  -  np.pi)/np.pi  *
100:.2f}%")

    # Візуалізація
    plt.figure(figsize=(8, 8))
    plt.scatter(x[inside],              y[inside],
color='blue', s=1, label='Всередині кола')
    plt.scatter(x[~inside],            y[~inside],
color='red', s=1, label='Зовні кола')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title('Метод Монте-Карло для обчислення π')
    plt.legend()
    plt.axis('equal')
    plt.show()
```

Крок 3. Запуск програми

Виконайте скрипт:

```
python monte_carlo.py
```

Результат виведеться у консолі, а графік з точками з’явиться у новому вікні.

Аналіз результатів

Для 1 000 точок:

- Оцінка π буде грубою (наприклад, 3.12 або 3.18).
- Похибка: 1-5%.

Для 100 000 точок:

- Оцінка π стане точнішою (наприклад, 3.1412).
- Похибка: 0.1-0.5%.

Для 1 000 000 точок:

- Оцінка π наблизиться до 3.14159.
- Похибка: <0.01%.

3 ЛАБОРАТОРНА РОБОТА №3. ІНТЕГРАЦІЯ СИСТЕМ ЗА ДОПОМОГОЮ API

3.1 Теоретичні відомості

API (Application Programming Interface) — це набір правил, протоколів та інструментів, які дозволяють різним програмним системам взаємодіяти між собою. API визначає:

- Які запити може робити клієнт.
- Які дані повертаються.
- Як відбувається обмін даними.

Інтеграція систем через API — це процес об'єднання різних програмних компонентів або сервісів через їхні інтерфейси. Наприклад, мобільний додаток може отримувати дані про погоду з зовнішнього сервісу через його API.

REST API

Визначення та принципи

REST (Representational State Transfer) — це архітектурний стиль, заснований на шести принципах:

1. **Клієнт-серверна архітектура**: Розділення відповідальності між клієнтом і сервером.
2. **Безстанність (Stateless)**: Кожен запит містить всю інформацію, необхідну для його обробки.
3. **Кешування**: Відповіді можуть кешуватися для покращення продуктивності.
4. **Єдині інтерфейси**: Стандартизовані методи взаємодії (наприклад, HTTP-методи).
5. **Шаруватість**: Система може мати кілька шарів (наприклад, балансування навантаження).
6. **Код за вимогою (необов'язково)**: Сервер може надавати клієнту виконуваний код (наприклад, JavaScript).

Особливості REST

- **Формат даних**: JSON (легкий), XML, текст.
- **Методи HTTP**:
  - GET — отримання даних.
  - POST — створення нового ресурсу.
  - PUT — оновлення ресурсу.
  - DELETE — видалення ресурсу.
- **URL-структура**: Ресурси ідентифікуються через URI (наприклад, /api/users/1).

Приклад REST-запиту:

- GET https://api.example.com/users/1

Відповідь (JSON):

- {
- "id": 1,
- "name": "Іван Петренко",
- "email": "ivan@example.com"
- }

SOAP API

Визначення та структура

SOAP (Simple Object Access Protocol) — це протокол для обміну структурованими даними у розподілених системах. Основні компоненти:

- **XML-конверт (Envelope)**: Визначає початок і кінець повідомлення.
- **Заголовок (Header)**: Містить метадані (наприклад, токени безпеки).
- **Тіло (Body)**: Основні дані запиту або відповіді.
- **Fault (необов'язково)**: Опис помилок.

Особливості SOAP

- **Стандартизація**: Використовує WSDL (Web Services Description Language) для опису сервісів.
- **Протоколи**: Працює поверх HTTP, SMTP, TCP.
- **Безпека**: Підтримує WS-Security для шифрування, цифрових підписів.
- **Транзакції**: Забезпечує ACID-властивості через WS-AtomicTransaction.

Порівняння REST та SOAP

Критерій	REST	SOAP
Архітектура	Набір принципів (стиль)	Протокол з жорсткими стандартами
Дані	JSON, XML, текст	Тільки XML
Швидкість	Швидше через легкий формат	Повільніший через XML
Складність реалізації	Простота інтеграції	Вимагає знання WSDL і XML
Безпека	HTTPS + OAuth/JWT	WS-Security (шифрування, підписи)
Використання	Веб-додатки, мобільні сервіси	Банки, корпоративні системи

Коли використовувати REST/SOAP?

- **RART**:
  - Публічні API (наприклад, соцмережі, погодні сервіси).
  - Мобільні додатки.
  - Проекти, де важлива швидкість і простота.
- **SOAP**:
  - Системи з високими вимогами до безпеки (банки, медицина).
  - Складні транзакції з підтримкою ACID.
  - Інтеграція зі старими системами (наприклад, корпоративний софт).

Інструменти для роботи з API

1. **Postman**: Тестування REST/SOAP-запитів, створення документації.
2. **SoapUI**: Спеціалізований інструмент для тестування SOAP-сервісів.
3. **Swagger/OpenAPI**: Документування REST-API.
4. **Zeep (Python)**: Бібліотека для роботи з SOAP.
5. **cURL**: Командний рядок для відправки HTTP-запитів.

Безпека в API

- **REST**:

- **HTTPS**: Шифрування даних.
- **OAuth 2.0 / JWT**: Авторизація через токени.
- **CORS**: Контроль доступу до ресурсів.

• **SOAP**:

- **WS-Security**: Шифрування, цифрові підписи.
- **SSL/TLS**: Захист каналу передачі.

Реальні кейси використання

• **REST**:

- **Twitter API**: Отримання твітів, публікація постів.
- **Google Maps API**: Вбудовування мап на веб-сайти.

• **SOAP**:

- **Платіжні системи**: Банківські транзакції з підтримкою WS-Security.
- **ERP-системи**: Інтеграція між корпоративними базами даних.

3.2 Завдання до лабораторної роботи

1. Робота з REST API:

Налаштувати запити до публічного REST API (наприклад, погода, курси валют).

Отримати дані та зберегти їх у файл (JSON/CSV).

2. Робота з SOAP API:

Виконати запит до SOAP-сервісу за допомогою XML.

Проаналізувати відповідь.

3. Порівняння REST та SOAP:

Скласти таблицю з основними відмінностями на основі досвіду.

3.3 Вказівки до виконання лабораторної роботи

Завдання 1: Робота з REST API (OpenWeatherMap)

Крок 1. Отримання API-ключа

Перейдіть на OpenWeatherMap.

Натисніть "Sign Up" та створіть обліковий запис.

Після входу перейдіть у розділ "API Keys" та скопіюйте ключ (виглядає як a1b2c3d4e5f6g7h8i9j0).

Крок 2. Виконання GET-запиту через Postman

Відкрийте Postman → New Request.

Введіть URL:

https://api.openweathermap.org/data/2.5/weather?q=Київ&appid=ВАШ_КЛЮЧ&units=metric&lang=ua

Замініть ВАШ_КЛЮЧ на отриманий API-ключ.

Виберіть метод GET → натисніть Send.

Перевірте відповідь (має містити JSON з даними про погоду, наприклад):

```
{
  "weather": [{"description": "хмарно"}],
  "main": {"temp": 18.5, "humidity": 65},
  "name": "Київ"
}
```

Крок 3. Автоматизація запиту на Python

Створіть файл rest_api.py і додайте код:

```
import requests
import json

API_KEY = "ВАШ_КЛЮЧ"
CITY = "Київ"
URL = f"https://api.openweathermap.org/data/2.5/weather?q={CITY}&appid={API_KEY}&units=metric&lang=ua"

# Виконання GET-запиту
response = requests.get(URL)
if response.status_code == 200:
    data = response.json()
    # Збереження у JSON
    with open("weather.json", "w", encoding="utf-8") as f:
        json.dump(data, f, ensure_ascii=False, indent=4)
    # Вивід даних
    print(f"Погода в {CITY}: {data['weather'][0]['description']}")
```

```
        print(f"Температура:
{data['main']['temp']}°C")
    else:
        print(f"Помилка: {response.status_code}")
```

Запустіть скрипт:

python rest_api.py

Переконайтеся, що файл weather.json створено.

Завдання 2: Робота з SOAP API (Currency Converter)

Крок 1. Підготовка SOAP-запиту

Відкрийте Postman → New Request.

Введіть URL SOAP-сервісу:

http://currencyconverter.kowabunga.net/converter.asmx

Оберіть метод POST.

Крок 2. Налаштування заголовків

У розділі Headers додайте:

Key: Content-Type, Value: text/xml; charset=utf-8

Key: SOAPAction, Value: http://www.webserviceX.NET/GetConversionRate

Крок 3. Формування XML-тіла

У розділі Body оберіть raw → XML і вставте:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetConversionRate
xmlns="http://www.webserviceX.NET/">
      <FromCurrency>USD</FromCurrency>
      <ToCurrency>UAH</ToCurrency>
    </GetConversionRate>
  </soap:Body>
</soap:Envelope>
```

Run HTML

Крок 4. Відправка запиту

Натисніть Send. У відповідь має прийти XML з курсом USD до UAH:

```
<GetConversionRateResponse>
```

```
<GetConversionRateResult>37.5</GetConversionRateResult>
    </GetConversionRateResponse>
```

Run HTML

Скріншот:

Крок 5. Автоматизація на Python (бібліотека zeep)

Створіть файл soap_api.py і додайте код:

```
from zeep import Client
from zeep.cache import SqliteCache
from zeep.transports import Transport

# Налаштування клієнта з кешуванням
cache = SqliteCache()
transport = Transport(cache=cache)
wsdl = "http://currencyconverter.kowabunga.net/converter.asmx?WSDL"
client = Client(wsdl, transport=transport)

# Виклик SOAP-методу
try:
    rate = client.service.GetConversionRate("USD", "UAH")
    print(f"Курс USD до UAH: {rate}")
except Exception as e:
    print(f"Помилка: {e}")
```

Запустіть скрипт:

python soap_api.py

Завдання 3: Порівняння REST та SOAP

Складіть таблицю, заповнивши її на основі вашого досвіду:

Критерій	REST	SOAP
Простота використання	Легко інтегрується з JSON, мінімум налаштувань.	Вимагає знання XML і WSDL, складніший синтаксис.

Критерій	REST	SOAP
Швидкість	Швидкий через легкі JSON-дані.	Повільніший через обробку XML.
Безпека	Використовує HTTPS та OAuth.	Підтримує WS-Security для шифрування.
Підтримка мов	Працює з будь-якою мовою.	Часто вимагає спеціальних бібліотек.

Додаткові завдання

1. **Збереження даних у CSV**:
   - Додайте код до rest_api.py для експорту даних у CSV:

```
import pandas as pd
df = pd.DataFrame({
    "Місто": [data["name"]],
    "Температура": [data["main"]["temp"]],
    "Опис": [data["weather"][0]["description"]]
})
df.to_csv("weather.csv", index=False)
```

2. **Обробка помилок**:
   - Додайте перевірку на помилковий статус код (наприклад, 404) у REST-запиті.
   - Обробіть винятки для SOAP-запиту (наприклад, недоступність сервісу).

Звітність

1. **Скріншоти**:
   - REST-запит у Postman з відповіддю.
   - SOAP-запит у Postman з відповіддю.
2. **Файли**:
   - weather.json (результат REST).
   - weather.csv (додаткове завдання).
3. **Код**:
   - rest_api.py та soap_api.py.
4. **Висновки**:
   - Порівняйте REST і SOAP за 3-ма параметрами на ваш вибір.

4 ЛАБОРАТОРНА РОБОТА №4. ВСТУП ДО ХМАРНИХ ТЕХНОЛОГІЙ

4.1 Теоретичні відомості

Хмарні обчислення — модель надання обчислювальних ресурсів (сервери, сховища, мережі, ПЗ) через інтернет з оплатою за фактичне використання.

Ключові характеристики (за NIST):

1. **Самообслуговування** (On-demand self-service): Користувач самостійно керує ресурсами через веб-інтерфейс.
2. **Універсальний доступ** (Broad network access): Доступ з будь-якого пристрою (ПК, смартфон, планшет).
3. **Об'єднання ресурсів** (Resource pooling): Багатокористувацька модель з динамічним розподілом ресурсів.
4. **Еластичність** (Rapid elasticity): Миттєве масштабування під навантаження.
5. **Вимірюваність** (Measured service): Автоматичний моніторинг та тарифікація.

Моделі обслуговування

Модель	Опис	Приклади
IaaS (Infrastructure as a Service)	Надання віртуальної інфраструктури: сервери, мережі, ОС.	AWS EC2, Google Compute Engine
PaaS (Platform as a Service)	Середовище для розробки та запуску додатків (без керування ОС).	Heroku, Google App Engine
SaaS (Software as a Service)	Готові програмні рішення через браузер.	Microsoft 365, Salesforce

Моделі розгортання

1. **Публічна хмара**:
   - Ресурси належать провайдеру (AWS, Azure).
   - Переваги: низька вартість, масштабованість.
   - Недоліки: обмежений контроль безпеки.
2. **Приватна хмара**:

- Інфраструктура для однієї організації (наприклад, OpenStack).
- Переваги: повний контроль, відповідність стандартам.
- Недоліки: висока вартість розгортання.

3. **Гібридна хмара**:
- Поєднання публічної та приватної хмар.
- Приклад: зберігання конфіденційних даних локально, а обчислення — у публічній хмарі.

4. **Спільнотна хмара**:
- Спільне використання ресурсів організаціями зі схожими цілями (наприклад, медичні установи).

Архітектура хмарних систем

1. **Frontend**:
- Клієнтський інтерфейс (браузер, мобільний додаток).

2. **Backend**:
- Хмарна інфраструктура: сервери, сховища, віртуалізація.

3. **Платформа керування**:
- Інструменти для автоматизації (наприклад, Kubernetes для оркестрації контейнерів).

Безпека в хмарі

1. **Спільна відповідальність** (Shared Responsibility Model):
- **Провайдер**: Захист фізичної інфраструктури.
- **Користувач**: Керування даними, доступом, шифруванням.

2. **Основні загрози**:
- Втрата даних через помилки конфігурації.
- DDoS-атаки.
- Витоки конфіденційної інформації.

Переваги та недоліки

Переваги	**Недоліки**
Зменшення капітальних витрат (не потрібно купувати сервери).	Залежність від інтернет-з’єднання.
Гнучкість (масштабування за хвилини).	Ризик втрати контролю над даними.
Доступність з будь-якої точки світу.	Можливі непередбачені витрати.

.

4.2 Завдання до лабораторної роботи

1. Реєстрація на хмарній платформі

Зареєструйтеся на безкоштовному tier AWS, Google Cloud або Microsoft Azure.

2. Робота з IaaS

Створіть віртуальну машину (VM) на базі ОС Linux/Windows.

Підключіться до VM через SSH/RDP.

3. Робота з хмарним сховищем

Завантажте файл у хмарне сховище (наприклад, Amazon S3 або Google Cloud Storage).

Налаштуйте публічний доступ до файлу.

4. Розгортання додатку на PaaS

Розгорніть простий веб-додаток (наприклад, "Hello World") на платформі Heroku або Google App Engine.

5. Аналіз витрат

Проаналізуйте вартість використаних ресурсів за допомогою калькулятора хмарних послуг.

4.3 Вказівки до виконання лабораторної роботи

Завдання 1: Реєстрація на хмарній платформі (на прикладі AWS)

Крок 1: Створення облікового запису

Перейдіть на сторінку реєстрації AWS.

Введіть email, пароль та назву облікового запису (наприклад, my-cloud-lab).

Вкажіть тип облікового запису – "Особистий".

Заповніть контактні дані (країна, ПІБ, адреса).

Крок 2: Введення платіжних даних

Введіть дані банківської картки (AWS стягне $1 для верифікації, потім поверне).

Підтвердьте номер телефону через SMS або дзвінок.

Крок 3: Вибір підписки

Оберіть "Безкоштовний базовий план" (Free Tier).

Перевірте email для підтвердження облікового запису.

Крок 4: Налаштування безпеки

Увімкніть Multi-Factor Authentication (MFA) для додаткової захисту:

Відкрийте IAM (Identity and Access Management) → Users → Ваш обліковий запис → Security credentials.

Оберіть "Assign MFA device" → дотримуйтесь інструкцій (наприклад, використовуйте Google Authenticator).

Завдання 2: Створення віртуальної машини (EC2 на AWS)

Крок 1: Запуск інстансу

У консолі AWS знайдіть сервіс EC2 → натисніть "Launch Instance".

Оберіть Amazon Machine Image (AMI):

Для Linux: "Ubuntu Server 22.04 LTS".

Для Windows: "Microsoft Windows Server 2022 Base".

Оберіть тип інстансу: t2.micro (позначено як Free Tier eligible).

Крок 2: Налаштування ключа доступу

У розділі "Key Pair" натисніть "Create new key pair".

Введіть ім’я ключа (наприклад, ec2-key) → оберіть формат .pem → завантажте файл.

Зберігайте ключ у безпеці! Втрата ключа = втрата доступу до VM.

Крок 3: Налаштування мережі

У розділі "Network Settings":

Дозвольте SSH (для Linux) або RDP (для Windows).

Для публічного доступу оберіть "Allow SSH traffic from Anywhere" (0.0.0.0/0).

Крок 4: Підключення до VM

Після запуску інстансу знайдіть його Public IPv4 address у консолі EC2.

Для Linux (через термінал):

```
chmod 400 ec2-key.pem  # Зміна прав доступу до ключа
ssh -i "ec2-key.pem" ubuntu@public-ip
```

Для Windows:

Використовуйте Remote Desktop Protocol (RDP).

Завантажте файл пароля через консоль EC2 → "Get Password" → використовуйте .pem ключ для розшифрування.

Крок 5: Тестування VM

Виконайте команду в терміналі:

```
sudo apt update && sudo apt upgrade -y  # Оновлення пакетів (для Linux)
```

Переконайтеся, що інстанс відповідає на запити.

Завдання 3: Робота з хмарним сховищем (Amazon S3)

Крок 1: Створення Bucket

У консолі AWS перейдіть до S3 → "Create bucket".

Введіть унікальне ім’я (наприклад, my-lab-bucket-2024).

Оберіть регіон (наприклад, us-east-1).

Налаштування блокування:

"Block all public access" → зніміть галочку для публічного доступу.

Підтвердьте "I acknowledge that...".

Крок 2: Завантаження файлу

Відкрийте створений bucket → "Upload".

Перетягніть файл (наприклад, report.pdf) або оберіть його через діалогове вікно.

У властивостях файлу оберіть "Make public using ACL".

Крок 3: Генерація публічного URL

Після завантаження клацніть правою кнопкою на файл → "Open" → скопіюйте URL з адресного рядка браузера.

URL має виглядати так:

https://my-lab-bucket-2024.s3.amazonaws.com/report.pdf

Крок 4: Перевірка доступу

Відкрийте URL у іншому браузері або приватному вікні.

Якщо файл не відкривається:

Перевірте права доступу в Permissions → Bucket Policy.

Додайте політику:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": "*",
        "Action": "s3:GetObject",
        "Resource":      "arn:aws:s3:::my-lab-
bucket-2024/*"
    }]
}
```

Завдання 4: Розгортання додатку на Heroku

Крок 1: Підготовка коду

Створіть папку проекту з файлами:

app.py (основний додаток):

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def home():
    return "Лабораторна робота з хмарних обчислень!"
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

requirements.txt (залежності):

flask==2.0.2

gunicorn==20.1.0

Procfile (інструкції для Heroku):

web: gunicorn app:app

Крок 2: Встановлення Heroku CLI

Завантажте інструменти з офіційного сайту.

У терміналі виконайте:

```
heroku login  # Авторизація
heroku create my-cloud-lab-app  # Створення додатку
git init
git add .
git commit -m "Initial commit"
git push heroku master  # Deploy коду
```

Крок 3: Перевірка роботи

Після успішного деплою отримайте URL додатку:

heroku open

Якщо виникає помилка:

Перегляньте логи: heroku logs --tail.

Переконайтеся, що всі файли (зокрема Procfile) додані у репозиторій.

Завдання 5: Аналіз витрат

Крок 1: Використання калькулятора AWS

Відкрийте AWS Pricing Calculator.

Додайте сервіси: EC2, S3, Data Transfer.

Вкажіть параметри:

EC2: 1 інстанс t2.micro, 30 годин на місяць.

S3: 5 ГБ сховища, 1000 запитів GET.

Оцініть вартість (для Free Tier більшість послуг безкоштовні перші 12 місяців).

Крок 2: Моніторинг у консолі

У AWS перейдіть до Cost Explorer → "Forecast".

Перегляньте витрати за останні 7 днів.

Додаткові поради

Зупинка ресурсів: Після завершення роботи зупиніть EC2-інстанс і видаліть S3 bucket, щоб уникнути списання коштів.

Термінал для Windows: Використовуйте Git Bash або PuTTY для SSH.

Помилка "Permission denied": Виправте права доступу до .pem-файлу:

chmod 400 ec2-key.pem

Звіт повинен містити:

Скріншоти:

1. EC2-інстанс у стані "Running".
2. Публічний URL файлу в S3.
3. Веб-сторінка додатку на Heroku.

5 ЛАБОРАТОРНА РОБОТА №5 ОСНОВИ РОБОТИ З BIG DATA

5.1 Теоретичні відомості

Big Data — це дисципліна, що охоплює методи роботи з даними, які неможливо обробити класичними інструментами через їхній **обсяг**, **швидкість генерації**, **різноманітність форматів** та **складність аналітики**.

Додаткові характеристики (7V):

- **Variability** (змінність): Дані можуть змінювати структуру з часом (наприклад, сезонні тренди).
- **Visualization** (візуалізація): Необхідність інтерактивного представлення результатів.
- **Validity** (валідність): Коректність даних для поставлених завдань.

Приклади застосування:

- **Медицина**: Аналіз геномних даних для персоналізованого лікування.
- **Фінанси**: Виявлення шахрайства через аномалії в транзакціях.
- **Роздрібна торгівля**: Прогнозування попиту на основі історії покупок.

Архітектура Hadoop

Hadoop — фреймворк для розподіленої обробки даних. Складається з:

1. **HDFS (Hadoop Distributed File System)**:
    - **NameNode**: Керує метаданими (назви файлів, розміри, місцезнаходження).
    - **DataNode**: Зберігає блоки даних (за замовчуванням 128 МБ).
2. **MapReduce**:
    - **Map**: Розбиває дані на частини та обробляє паралельно.
    - **Reduce**: Агрегує проміжні результати.
3. **YARN (Yet Another Resource Negotiator)**: Керує ресурсами кластера.

Переваги Hadoop:

- Масштабування до тисяч вузлів.
- Відмовостійкість (дані реплікуються між вузлами).
- Обробка неструктурованих даних (тексти, логи).

Інші технології Big Data

Інструмент	Призначення
Apache Spark	Швидка обробка даних у пам’яті (в 100x швидше за MapReduce).
Apache Kafka	Потокова обробка даних у реальному часі.
Elasticsearch	Пошук та аналіз текстових даних.
Apache Flink	Обробка потокових даних з підтримкою транзакцій.

5.2 Завдання до лабораторної роботи

1. Встановити локальний Hadoop-кластер (або використати хмарний, напр. AWS EMR).
2. Обробити набір даних за допомогою MapReduce/Spark.
3. Провести аналіз даних з використанням Hive або Pig.
4. Створити інтерактивну візуалізацію в Tableau.

5.3 Вказівки до виконання лабораторної роботи

Завдання 1: Встановлення Hadoop локально

Крок 1. Встановіть Java

Hadoop потребує Java 8 або 11.

```
sudo apt update
sudo apt install openjdk-11-jdk -y
java -version   # Перевірка версії (має бути 11.x)
```

Крок 2. Налаштуйте SSH без пароля

```
sudo apt install openssh-server -y
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys
ssh localhost  # Тест підключення (не повинен питати пароль)
```

Крок 3. Завантажте та розпакуйте Hadoop

```
    wget
https://archive.apache.org/dist/hadoop/core/hadoop
-3.3.6/hadoop-3.3.6.tar.gz
    tar -xzvf hadoop-3.3.6.tar.gz
    sudo mv hadoop-3.3.6 /usr/local/hadoop
```

Крок 4. Налаштуйте змінні середовища

Відредагуйте ~/.bashrc:

```
    export HADOOP_HOME=/usr/local/hadoop
    export
PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
    export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-
amd64
```

Застосуйте зміни:

source ~/.bashrc

Крок 5. Налаштуйте конфігураційні файли Hadoop

```
    core-site.xml ($HADOOP_HOME/etc/hadoop/core-
site.xml):
    xml
    Copy
    <configuration>
        <property>
            <name>fs.defaultFS</name>
            <value>hdfs://localhost:9000</value>
        </property>
    </configuration>
    Run HTML
    hdfs-site.xml ($HADOOP_HOME/etc/hadoop/hdfs-
site.xml):
    xml
    Copy
    <configuration>
        <property>
            <name>dfs.replication</name>
            <value>1</value>
        </property>
        <property>
            <name>dfs.namenode.name.dir</name>
```

```
<value>/usr/local/hadoop/data/namenode</value>
        </property>
        <property>
            <name>dfs.datanode.data.dir</name>

<value>/usr/local/hadoop/data/datanode</value>
        </property>
    </configuration>
```

Run HTML

mapred-site.xml:

```
    <configuration>
        <property>
            <name>mapreduce.framework.name</name>
            <value>yarn</value>
        </property>
    </configuration>
```

Run HTML

yarn-site.xml:

```
    <configuration>
        <property>
            <name>yarn.nodemanager.aux-
services</name>
            <value>mapreduce_shuffle</value>
        </property>
    </configuration>
```

Run HTML

Крок 6. Форматування HDFS та запуск

```
    hdfs namenode -format  # Форматування (1 раз!)
    start-dfs.sh           # Запуск HDFS
    start-yarn.sh          # Запуск YARN
    jps                       # Перевірка процесів
(мають бути: NameNode, DataNode, ResourceManager)
```

Завдання 2: Обробка даних з MapReduce (Python)

Крок 1. Завантажте тестові дані

```
    wget  https://norvig.com/big.txt    #  Великий
текстовий файл (6.5 МБ)
```

```
hdfs dfs -mkdir /input
hdfs dfs -put big.txt /input
```

Крок 2. Напишіть маппер та редюсер

mapper.py:

```
#!/usr/bin/env python3
import sys
import re

for line in sys.stdin:
    line = re.sub(r'\W+', ' ', line.strip()).lower()
    words = line.split()
    for word in words:
        print(f"{word}\t1")
```

reducer.py:

```
#!/usr/bin/env python3
import sys

current_word = None
count = 0

for line in sys.stdin:
    word, value = line.strip().split('\t')
    if word == current_word:
        count += int(value)
    else:
        if current_word:
            print(f"{current_word}\t{count}")
        current_word = word
        count = 1
if current_word:
    print(f"{current_word}\t{count}")
```

Крок 3. Запустіть MapReduce-завдання

```
# Дозволити виконання скриптів
chmod +x mapper.py reducer.py

# Запуск через Hadoop Streaming
```

```
    hadoop                                          jar
$HADOOP_HOME/share/hadoop/tools/lib/hadoop-
streaming-3.3.6.jar \
    -input /input/big.txt \
    -output /output \
    -mapper ./mapper.py \
    -reducer ./reducer.py \
    -file mapper.py \
    -file reducer.py
```

Крок 4. Перегляньте результат

```
    hdfs dfs -cat /output/part-00000 | sort -nrk2 |
head -n10 # Топ-10 слів
```

Завдання 3: Візуалізація результатів

Варіант 1: Python + Matplotlib

Встановіть бібліотеки:

```
    pip install matplotlib pandas
    Створіть скрипт visualize.py:
    python
    Copy
    import pandas as pd
    import matplotlib.pyplot as plt

    # Завантажте дані з HDFS (або локального файлу)
    df   =   pd.read_csv('results.csv',   sep='\t',
header=None, names=['Word', 'Count'])
    top10       =       df.sort_values(by='Count',
ascending=False).head(10)

    # Побудуйте графік
    plt.barh(top10['Word'],         top10['Count'],
color='skyblue')
    plt.xlabel('Частота')
    plt.title('Топ-10 слів')
    plt.gca().invert_yaxis()
    plt.savefig('word_count.png')
    plt.show()
```

Запустіть:

```
hdfs dfs -get /output/part-00000 results.csv
python3 visualize.py
```

Варіант 2: Google Data Studio
Експортуйте дані у Google Sheets.
Перейдіть на Google Data Studio.
Створіть звіт з стовпчиковою діаграмою.

Завдання 4: Прибирання
Зупиніть Hadoop:

```
stop-yarn.sh
stop-dfs.sh
```

Видаліть тимчасові дані:

```
rm -rf /usr/local/hadoop/data/*
```

6 ЛАБОРАТОРНА РОБОТА №6 ВИКОРИСТАННЯ AUTOML

6.1 Теоретичні відомості

AutoML (Automated Machine Learning) – це галузь штучного інтелекту, яка автоматизує процес створення, налаштування та оптимізації моделей машинного навчання. Вона покликана усунути необхідність ручного втручання на таких етапах:

– Попередня обробка даних (наприклад, нормалізація, кодування категоріальних змінних).

– Вибір ознак (feature selection) та інженерія ознак (feature engineering).

– Вибір алгоритму (наприклад, дерева рішень, градієнтний бустинг).

– Оптимізація гіперпараметрів (hyperparameter tuning).

– Оцінка моделі (метрики точності, AUC-ROC тощо).

2. Основні компоненти AutoML

Automated Data Preprocessing:

– Обробка пропущених значень: автозаповнення медіаною, середнім або режимом.

– Кодування категоріальних змінних: one-hot encoding, label encoding.

– Скаліювання даних: StandardScaler, MinMaxScaler.

Automated Feature Engineering:

– Генерація нових ознак (наприклад, взаємодія між змінними).

– Вибір найважливіших ознак за допомогою методів: ANOVA, LASSO, SHAP-значень.

Model Selection & Hyperparameter Tuning:

– Гіперпараметри: параметри моделі, які не навчаються (наприклад, глибина дерева, кількість нейронів у шарі).

Методи пошуку:

– Grid Search: повний перебір комбінацій.

– Random Search: випадковий пошук.

– Bayesian Optimization: прогнозування найкращих гіперпараметрів на основі попередніх спроб.

– Genetic Algorithms: еволюційний підхід, що імітує природний відбір.

Ensemble Learning:

– Комбінування прогнозів кількох моделей для підвищення точності (наприклад, VotingClassifier, Stacking).

3. Підходи в AutoML

Повністю автоматизований підхід:

– Інструменти: H2O AutoML, Google AutoML Tables, Auto-sklearn.

– Користувач лише вказує дані та цільову змінну.

Напівавтоматизований підхід:

– Інструменти: TPOT, MLJAR.

– Користувач може втручатися в окремі етапи (наприклад, додати власні ознаки).

Нейронні архітектури (AutoDL):

– Автоматичний пошук архітектури нейронної мережі (наприклад, AutoKeras).

4. Переваги та обмеження AutoML

Переваги	Обмеження
Економить час – автоматизація рутинних задач.	Обчислювальна складність – пошук гіперпараметрів може бути ресурсомістким.
Доступність – дозволяє неекспертам будувати моделі.	Ризик перетренування – деякі інструменти "підганяють" модель під конкретний набір даних.
Висока точність – використання ансамблів та оптимізації.	"Чорний ящик" – складність інтерпретації результатів.

5. Використання AutoML у реальних проектах

Фінанси:

– Прогнозування кредитних ризиків.

– Виявлення шахрайських транзакцій.

Маркетинг:

– Сегментація клієнтів.

– Прогнозування відтоку (churn prediction).

Медицина:

– Діагностика захворювань на основі медичних зображень.

– Персоналізоване лікування.

6. Порівняння AutoML з традиційним ML

Критерій	AutoML	Традиційний ML
Час розробки	Години/хвилини.	Дні/тижні.
Рівень експертності	Мінімальний.	Високий (потрібне глибоке розуміння алгоритмів).
Гнучкість	Обмежена можливостями інструменту.	Повний контроль на всіх етапах.
Точність	Часто конкурує з ручними моделями.	Залежить від досвіду інженера.

7. Вибір інструментів AutoML

Інструмент	Тип	Переваги
H2O AutoML	Open-source	Підтримка ансамблів, інтеграція з Python/R.
Google AutoML	Хмарний сервіс	Висока точність, графічний інтерфейс.
TPOT	Бібліотека Python	Генерує код для подальшого використання.
Auto-sklearn	Розширення scikit-learn	Використовує мета-навчання для вибору моделей.

8. Етика та майбутнє AutoML

Етичні аспекти:

– AutoML може посилювати зміщення (bias) у даних, якщо навчальний набір нерепрезентативний.

– Важливість інтерпретованості моделей (XAI – Explainable AI).

Майбутнє:

– Інтеграція з MLOps для автоматизації всього життєвого циклу моделі.

– AutoML для часових рядів та NLP.

6.2 Завдання до лабораторної роботи

1. Завантажити набір даних (наприклад, Titanic).
2. Використати AutoML-інструмент (H2O або TPOT) для автоматичного навчання моделі.
3. Оцінити точність моделі.
4. Порівняти з моделлю, побудованою вручну (наприклад, Random Forest).

6.3 Вказівки до виконання лабораторної роботи

Завдання 1: Підготовка середовища

Встановіть Python 3.8+ та бібліотеки:

```
pip install h2o tpot pandas scikit-learn
```

Завантажте дані Titanic:

```
import pandas as pd
train_data = pd.read_csv("train.csv")
test_data = pd.read_csv("test.csv")
```

Завдання 2: Обробка даних

Видаліть непотрібні стовпці:

```
train_data = train_data.drop(["PassengerId", "Name", "Ticket", "Cabin"], axis=1)
```

Заповніть пропущені значення:

```
train_data["Age"].fillna(train_data["Age"].median(), inplace=True)
train_data["Embarked"].fillna(train_data["Embarked"].mode()[0], inplace=True)
```

Закодуйте категоріальні змінні:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
train_data["Sex"] = le.fit_transform(train_data["Sex"])
train_data["Embarked"] = le.fit_transform(train_data["Embarked"])
```

Завдання 3: AutoML з H2O

Ініціалізуйте H2O:

```
import h2o
```

```
h2o.init()  # Локальний сервер H2O
```

Перетворіть дані у формат H2O:

```
h2o_train = h2o.H2OFrame(train_data)
```

Запустіть AutoML:

```
from h2o.automl import H2OAutoML
aml = H2OAutoML(max_models=10, seed=42, max_runtime_secs=300)
aml.train(y="Survived", training_frame=h2o_train)
```

Перегляньте результати:

```
print(aml.leaderboard)  # Топ-моделі
```

Прогнозування:

```
preds = aml.leader.predict(h2o_test)
```

Завдання 4: Порівняння з ручним підходом

Створіть модель Random Forest:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Оцініть точність:

```
from sklearn.metrics import accuracy_score
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Звітність

Код та результати:

- Топ-3 моделі з AutoML (скріншот leaderboard).
- Точність AutoML vs Random Forest.

Додаткові завдання (опціонально)

Використайте інший AutoML-інструмент (наприклад, TPOT).

Налаштуйте гіперпараметри моделі вручну та порівняйте з AutoML.

Приклад коду для TPOT

```
from tpot import TPOTClassifier
```

```
    pipeline    =    TPOTClassifier(generations=5,
population_size=50, random_state=42, verbosity=2)
    pipeline.fit(X_train, y_train)
    print("Accuracy:",      pipeline.score(X_test,
y_test))
```

ЛІТЕРАТУРА

1. Montgomery D. C. *Design and Analysis of Experiments*. – New York: Wiley, 2017. – 672 с.

2. Kuehl R. O. *Design of Experiments: Statistical Principles of Research Design and Analysis*. – Belmont, CA: Brooks/Cole, 2000. – 448 с.

3. Назаренко Л. А. *Основи планування експерименту*. – Харків: ХНУ імені О. М. Бекетова, 2005. – 320 с.

4. Glasserman P. *Monte Carlo Methods in Financial Engineering*. – New York: Springer, 2003. – 496 с.

5. Robert C. P., Casella G. *Monte Carlo Statistical Methods*. – New York: Springer, 2004. – 618 с.

6. Ross S. M. *Simulation*. – San Francisco: Academic Press, 2013. – 560 с.

7. Richardson L., Amundsen M., Ruby S. *RESTful Web APIs: Services for a Changing World*. – Sebastopol: O'Reilly Media, 2013. – 328 с.

8. Newman S. *Building Microservices: Designing Fine-Grained Systems*. – Sebastopol: O'Reilly Media, 2015. – 280 с.

9. Fielding R. T. *Architectural Styles and the Design of Network-based Software Architectures* [Dissertation]. – Irvine: University of California, 2000.

10. Erl T., Mahmood Z., Puttini R. *Cloud Computing: Concepts, Technology & Architecture*. – Upper Saddle River: Prentice Hall, 2013. – 328 с.

11. Marz N., Warren J. *Big Data: Principles and Best Practices of Scalable Real-Time Data Systems*. – Shelter Island: Manning Publications, 2015. – 352 с.

12. Hutter F., Kotthoff L., Vanschoren J. *Automated Machine Learning: Methods, Systems, Challenges*. – Cham: Springer, 2019. – 360 с.