**ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ**
**Навчально-науковий інститут економіки, управління, права та інформаційних технологій**
**Кафедра інформаційних систем та технологій**

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«Методика візуального аналізу даних засобами Python»**

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні системи та технології ступеня вищої освіти бакалавр
групи 126ІСТ_бд_2021
Щербина Ілля Андрійович
Керівник: Флегантов Леонід Олексійович
Рецензент: Брикун Олександр Миколайович

**Полтава – 2025 року**

# ВСТУП

*Актуальність теми.* В умовах стрімкого зростання обсягів цифрових даних у всіх сферах людської діяльності зростає потреба у швидкому, наочному та достовірному представленні інформації. Зокрема, візуальний аналіз даних відіграє ключову роль у виявленні закономірностей, трендів, аномалій та ухваленні обґрунтованих рішень на основі великих масивів даних. Застосування мови програмування Python, що має потужний інструментарій для візуалізації, дозволяє реалізовувати гнучкі, інтерактивні та масштабовані рішення для аналізу інформації. Тема даної роботи є актуальною, оскільки відповідає сучасним тенденціям у галузі data science, цифрової трансформації бізнесу, науки та освіти. Розроблення методики візуального аналізу з використанням бібліотек Python дозволяє не лише покращити аналітичну якість представлення даних, а й сприяє ефективнішому прийняттю управлінських рішень, автоматизації звітності, підвищенню прозорості досліджень і бізнес-процесів.

Аналіз стану розробки проблєми. На сучасному ІТ-ринку існує велика кількість інструментів для візуального аналізу даних, таких як Tableau, Microsoft Power BI, Qlik Sense, які пропонують зручний графічний інтерфейс і широкі можливості для створення дашбордів. Водночас вони є комерційними, менш гнучкими для налаштування під нестандартні задачі, а також обмеженими у можливостях автоматизації. Python, як мова загального призначення з відкритим кодом, забезпечує більш високий рівень та гнучкість налаштувань, інтеграції та автоматизації, але потребує чіткої методики роботи з візуалізацією. Існують численні бібліотеки для побудови графіків (Matplotlib, Seaborn, Plotly, Bokeh, Altair), однак питання вибору оптимального інструменту для конкретного типу даних, аналітичного завдання та цільової аудиторії залишається відкритим. Наявні публікації розглядають окремі бібліотеки або технічні аспекти візуалізації, але систематизованого підходу до розроблення методики, орієнтованої на практичне застосування Python для аналізу даних, недостатньо. Тому розроблення уніфікованої методики з урахуванням теоретичних принципів візуалізації,

інструментальних можливостей та економічної доцільності є актуальним завданням.

*Мета роботи:* створення структурованого підходу, що дозволить ефективно використовувати можливості Python для візуального дослідження даних.

*Завдання роботи:*

- дослідити теоретичні основи візуального аналізу даних;
- проаналізувати можливості бібліотек Python для візуалізації даних, визначити їх переваги та недоліки;
- розробити методику вибору та застосування бібліотек Python відповідно до типу даних і завдань аналізу;
- реалізувати практичні приклади візуалізації даних з використанням різних бібліотек Python;
- оцінити економічну ефективність запропонованої методики.

*Об'єкт дослідження:* процес візуального аналізу даних.

*Предмет дослідження:* методика побудови ефективних візуалізацій даних за допомогою засобів Python.

*Методи наукових досліджень:* теоретичний аналіз літератури з тематики візуалізації даних; огляд документації бібліотек Python; експериментальне порівняння засобів візуалізації; проєктування методики; економічна оцінка ефективності розроблених рішень.

*Інформаційна база дослідження:* наукові публікації з візуалізації та аналізу даних, офіційна документація бібліотек Python, відкриті репозиторії даних (Kaggle, UCI Machine Learning Repository), матеріали фахових онлайн-ресурсів (Stack Overflow, Medium, Towards Data Science), навчально-методична література з візуального аналізу даних, наукові праці з ефективної візуалізації та застосування Python в аналізі інформації, а також відкриті набори даних, приклади візуалізацій, навчальні курси з data science, що ілюструють використання Python, професійні дискусії та аналітичні матеріали за темою дослідження.

*Апробація результатів дослідження.* За результатами дослідження опубліковано тези доповіді: «Обґрунтування вибору методів візуалізації даних за

допомогою Python», Матер. XXII щорічного міждисциплінарного семінару «Студентські роботи за науковою тематикою кафедри інформаційних систем та технологій ННІ ЕУП та ІТ ПДАУ», 16 квітня 2025 року, м. Полтава (додаток А).

*Практична значущість* роботи полягає в розробленні методики візуального аналізу даних, яка може бути застосована в усіх сферах, що потребують візуального аналізу та інтерпретації великих обсягів інформації.

*Структура роботи*: кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Основний текст викладений на 57 сторінках, містить 2 таблиці, 28 рисунків. Список використаних джерел складається з 60 найменувань.

# РОЗДІЛ 1
# ТЕОРЕТИЧНІ ОСНОВИ ВІЗУАЛЬНОГО АНАЛІЗУ ДАНИХ

## 1.1 Поняття та основні етапи візуального аналізу даних

Візуальний аналіз даних використовує наочне представлення даних для полегшення процесів їх дослідження, розуміння та представлення результатів. Головна ідея полягає у використанні можливостей зору людини для швидкого виявлення закономірностей, тенденцій, викидів (аномалій) та структур у даних, які можуть бути неочевидними при аналізі числових або текстових наборів.

Візуальний аналіз даних – це процес, що поєднує наступні аспекти:

- візуалізація даних – перетворення даних у графічне подання (графіки, діаграми, карти, мережі тощо) шляхом відображення атрибутів даних на візуальні змінні (положення, розмір, колір, форма);
- інтерактивність – надання користувачеві можливості маніпулювати візуалізацією даних (масштабування, фільтрація, панорамування, виділення зв'язаних даних) для глибшого дослідження та перевірки гіпотез в реальному часі;
- аналітичне мислення – застосування когнітивних процесів для інтерпретації візуальних патернів, формулювання гіпотез, виявлення причинно-наслідкових зв'язків та отримання значущих висновків (інсайтів) з даних.

Основна ідея, що часто асоціюється з візуальним аналізом, відома, як «мантра візуальної інформації» (Visual Information Seeking Mantra): «Спочатку огляд, масштабування та фільтрація, потім деталі за запитом» («Overview first, zoom and filter, then details-on-demand»). Цей підхід підкреслює ітеративний характер процесу візуального аналізу даних – від загального огляду до дослідження специфічних деталей [1-5].

Процес візуального аналізу даних включає такі основні етапи [6]:

- збір та підготовка даних. Завданням цього етапу є отримання даних, їх очищення від помилок та пропусків, перетворення у формат, придатний для аналізу та візуалізації (структурування, агрегація, нормалізація);

- вибір візуального кодування даних. На цьому етапі визначається, як саме атрибути даних (змінні) будуть представлені візуально. Зокрема, обирається тип візуалізації (стовпчикова діаграма, лінійний графік, розсіювання, теплова карта тощо) та відображення змінних на візуальні канали (позиція X/Y, колір, розмір, форма, текстура). Ефективність візуалізації залежить від вибору візуального кодування відповідно до типу даних (кількісні, категоріальні, часові, геопросторові) та аналітичного завдання (порівняння, дослідження розподілу, виявлення зв'язків, композиція);

- генерація та рендеринг візуального представлення. На цьому етапі застосовуються програмні засоби (ПЗ) (бібліотеки візуалізації, BI-платформи, інше спеціалізоване ПЗ) для створення статичного або інтерактивного візуального представлення на основі підготовлених даних та обраного кодування. Технічна реалізація візуалізації включає рендеринг графічних елементів;

- інтерактивне дослідження та аналіз. Це етап активної взаємодії користувача з візуалізацією для дослідження даних, на якому застосовуються, зокрема, техніки масштабування (zoom), панорамування (pan), фільтрації (filtering), виділення (brushing), зв'язування представлень (linking), сортування, запити до даних через візуальний інтерфейс. Це головний етап аналітичного процесу, на якому формулюються та перевіряються гіпотези, виявляються залежності, досліджуються різні аспекти даних, змінюючи параметри візуалізації;

- формулювання висновків та представлення результатів. На цьому етапі відбувається інтерпретація виявлених у ході дослідження залежностей та аномалій, формулювання значущих висновків (інсайтів), представлення цих результатів цільовій аудиторії у зрозумілій формі.

Ці етапи часто взаємопов'язані та можуть виконуватися ітеративно: наприклад, виявлені на етапі дослідження проблєми з даними можуть вимагати повернення до етапу підготовки, а несподівані інсайти можуть спонукати до зміни візуального кодування для кращого їх представлення.

## 1.2 Принципи візуалізації даних

Візуалізація даних, як процес перетворення абстрактної інформації у графічні зображення, керується низкою принципів, що спрямовані на забезпечення ефективності сприйняття, точності інтерпретації та уникнення спотворень чи непорозумінь при представленні даних. До таких принципів належать [7]:

- точність та вірність даних. Цей принцип означає, що візуалізація повинна точно відображати кількісні та якісні характеристики даних без спотворень. Графічні елементи візуалізації (довжина стовпчиків, площа секторів, позиція точок) мають бути пропорційними значенням, які вони представляють. Слід уникати маніпуляцій зі шкалами (наприклад, усічені осі Y, що починаються не з нуля, якщо це не обґрунтовано контекстом), використання 3D-ефектів, які ускладнюють порівняння, та вибіркового представлення даних, що може призвести до хибних висновків;
- вибір відповідного типу візуалізації. Тип графічного представлення має відповідати типу даних (кількісні, категоріальні, часові, геопросторові) та меті візуалізації (порівняння значень, показ розподілу, виявлення взаємозв'язків, відстеження змін у часі, демонстрація частин цілого). Наприклад, стовпчикові діаграми ефективні для порівняння дискретних категорій, лінійні графіки – для демонстрації трендів у часі, діаграми розсіювання – для виявлення кореляцій між двома кількісними змінними, секторні діаграми – для показу частин цілого;
- максимізація співвідношення «дані-чорнило». Цей принцип означає, що слід прагнути до того, щоб більша частина «чорнила» (пікселів на екрані) використовувалася для відображення самих даних, а не для неінформативних елементів графіки. Потрібно мінімізувати або усувати «візуальний шум» – непотрібні лінії сітки, надмірні прикраси, фонові зображення, тіні, зайві кольори, які не несуть інформаційного навантаження і відволікають від даних [8-15];
- ясність та простота. Візуалізація має бути легкою для читання та інтерпретації. Складність візуалізації повинна бути зумовлена складністю даних, а не складністю дизайну. Потрібно уникати захараченості, забезпечувати чітке

маркування осей, легенд та окремих елементів даних, використовувати зрозумілі шрифти та логічне компонування елементів;

- ефективне використання візуальних змінних. Це означає усвідомлено використовувати візуальні канали (позиція, довжина, кут, напрямок, площа, об'єм, насиченість кольору, відтінок кольору, форма, текстура) для кодування даних, оскільки людське сприйняття обробляє ці змінні з різною точністю. Для найважливіших кількісних даних слід використовувати змінні, що найбільш точно сприймаються (позиція, довжина). Використовувати колір усвідомлено: для категорій (різні відтінки), для кількісних даних (градієнти насиченості або послідовні палітри), для виділення. Враховувати особливості сприйняття кольору (наприклад, проблєми дальтонізму) [16,17];

- забезпечення контексту. Візуалізація сама по собі не завжди буває достатньою. Тому вона повинна супроводжуватись необхідною контекстною інформацією для правильного розуміння. Потрібно завжди включати інформативні заголовки, чіткі підписи осей із зазначенням одиниць виміру, легенду (якщо використовуються різні кольори, форми тощо), а також, за потреби, анотації до важливих точок даних та посилання на джерело даних;

- фокусування та акцентування. Потрібно спрямовувати увагу глядача на найважливіші аспекти даних або на ключове повідомлення, яке несе візуалізація. Наприклад, для підкреслення основних висновків або характерних особливостей, патернів потрібно використовувати відповідні візуальні засоби (контрастний колір, більший розмір, анотації, виділення);

- візуальна цілісність та чесність. Візуальне представлення даних не повинно вводити в оману, створювати хибне враження або підтримувати упереджену точку зору. Наприклад, слід послідовно використовувати шкали при порівнянні кількох графіків, уникати перебільшення відмінностей, представляти дані об'єктивно.

Дотримання цих принципів дозволяє створювати візуалізації даних, які є не тільки гармонійними та збалансованими зовні, але й інформативними, точними та ефективними інструментами для розуміння складних даних та передачі знань.

## 1.3 Сучасні інструменти для візуального аналізу даних

Сучасний IT-ринок пропонує широкий спектр інструментів для візуалізації даних, які можна поділити на кілька категорій: програмні бібліотеки, окремі програмні застосунки (BI, платформи Business Intelligence) та веборієнтовані фреймворки. У цій роботі головну увагу приділено бібліотекам мови програмування Python, до складу яких належать:

- Matplotlib – фундаментальна бібліотека для створення статичних, анімованих та інтерактивних візуалізацій, яка надає низькорівневий контроль над кожним елементом графіка (осі, мітки, кольори, стилі ліній тощо), що робить її надзвичайно гнучкою для створення візуалізацій. Бібліотека Matplotlib є основою для інших бібліотек візуалізації. Однак, її синтаксис для побудови складних графіків може бути дещо багатослівним [18].
- Seaborn – бібліотека на основі Matplotlib, надає високорівневий інтерфейс для створення привабливих та інформативних статистичних графіків. Спрощує візуалізацію складних наборів даних, має готові функції для типових статистичних діаграм (наприклад, діаграми розсіювання з регресійною прямою, коробкові та скрипкові діаграми, теплові карти). Seaborn добре інтегрується зі структурами даних Pandas [19];
- Plotly – бібліотека, що спеціалізується на створенні інтерактивних веборієнтованих візуалізацій, підтримує широкий спектр типів графіків, включаючи 3D-візуалізації. Графіки, створені за допомогою Plotly, дозволяють користувачу взаємодіяти з даними (здійснювати масштабування, панорамування, відображення значень при наведенні). Plotly може генерувати HTML файли або інтегруватись у вебдодатки, зокрема за допомогою фреймворку Dash для створення аналітичних панелей (дашбордів) [20];
- Bokeh – бібліотека, що подібно до Plotly, орієнтована на створення інтерактивних візуалізацій для веббраузерів, забезпечує можливість побудови складних графіків, дашбордів та додатків для роботи з даними. Особливістю Bokeh

є можливість роботи з потоковими даними та висока продуктивність при роботі з великими наборами даних у вебсередовищі [21];

- Altair – декларативна бібліотека статистичної візуалізації, що базується на граматиці Vega-Lite. Декларативний підхід означає, що користувач описує зв'язки між даними та візуальними елементами (осі, кольори, розміри), а Altair автоматично генерує відповідний графік. Це дозволяє створювати складні візуалізації за допомогою лаконічного коду [22];

- Pandas Visualization – вбудований функціонал бібліотеки Pandas, основний інструмент для маніпуляції даними в Python, має вбудовані методи для швидкої візуалізації даних безпосередньо з об'єктів Series та DataFrame (наприклад, df.plot()). Ці методи є оболонкою над Matplotlib і дозволяють швидко створювати базові графіки для розвідувального аналізу даних [23].

Крім Python, існують інші сучасні інструменти візуального аналізу даних:

- Microsoft Power BI, Tableau, Qlik Sense, Looker (Google Cloud), Domo, Sisense , ThoughtSpot – провідні платформи бізнес-аналітики (BI), що пропонують потужні можливості візуалізації через графічний інтерфейс користувача. Вони орієнтовані на бізнес-користувачів та аналітиків і дозволяють створювати інтерактивні дашборди та звіти без необхідності програмування [24];

- D3.js – бібліотека JavaScript, що дозволяє створювати складні інтерактивні візуалізації у вебсередовищі. Вона є стандартом для вебвізуалізації, але вимагає глибоких знань вебтехнологій (HTML, SVG, CSS, JavaScript) [25];

- R (з пакетом ggplot2) – мова програмування популярна в академічних та статистичних колах. Пакет ggplot2 реалізує концепцію «граматики графіки» (Grammar of Graphics), що дозволяє створювати складні та естетично привабливі візуалізації пошарово [26].

Таким чином, існує широкий спектр різноманітних інструментів візуалізації даних, призначених для конкретних завдань аналізу, цільової аудиторії, необхідного рівня інтерактивності та налаштувань, обсягу даних, а також навичок самого аналітика.

## 1.4 Переваги та недоліки використання Python для візуалізації даних

Використання мови програмування Python для візуалізації даних набуло значної популярності серед аналітиків даних, науковців та розробників, що зумовлено низкою переваг. Однак, як і будь-який інструмент, Python має певні обмеження та недоліки в цій сфері.

Переваги використання Python для візуалізації даних:

- Python має багато спеціалізованих бібліотек для візуалізації, що дозволяє підібрати інструмент, який найкраще відповідає конкретній задачі – від створення простих статичних графіків до складних інтерактивних вебдодатків та дашбордів [27];
- Python є фактичним стандартом для науки про дані. Бібліотеки візуалізації легко інтегруються з бібліотеками для маніпуляції даними (Pandas, NumPy), обчислень (SciPy) та машинного навчання (Scikit-learn, TensorFlow, PyTorch), що дозволяє створити єдиний, послідовний робочий процес аналізу даних – від завантаження та очищення даних до їх візуального дослідження, моделювання та представлення результатів в межах одного програмного середовища [28];
- Python є мовою загального призначення, тому фахівцям з інших галузей легше опанувати візуалізацію даних засобами Python, а також візуалізації, створені на Python, легше інтегрувати в програмні продукти та системи [29];
- Python має багато документації, навчальних матеріалів, прикладів коду, форумів та готових рішень для типових проблєм візуалізації у відкритому доступі, що спрощує процес навчання та вирішення конкретних завдань [30];
- переважна більшість бібліотек Python для аналізу та візуалізації даних є проєктами з відкритим вихідним кодом (open-source) і розповсюджуються безкоштовно, тому вони є доступними для широкого кола користувачів;
- Python використовує кодо-орієнтований підхід до візуалізації, що забезпечує відтворюваність результатів. Скрипти Python для генерації графіків можна зберігати, передавати, модифікувати та повторно виконувати, гарантуючи

ідентичність отриманих візуалізацій. Це особливо важливо для наукових досліджень та спільної роботи над даними [31].

Недоліки використання Python для візуалізації даних:

- для створення складних візуалізацій засобами Python, особливо у випадку використання Matplotlib або налаштування інтерактивності в Plotly/Bokeh, потрібен значний час на їх вивчення та більше коду порівняно з GUI-інструментами (Tableau, Power BI, Looker Studio) [32-34];
- для досягнення бажаного вигляду графіка, особливо при використанні Matplotlib, доводиться писати багато коду для налаштування деталей (міток, заголовків, кольорів, легенди тощо), що є менш інтуїтивним порівняно з декларативними підходами (як в Altair або R ggplot2) [35];
- інтерактивна візуалізація дуже великих наборів даних засобами Python безпосередньо в браузері є менш продуктивною порівняно зі спеціалізованими BI-платформами, які використовують власні оптимізовані рушії для обробки даних [36];
- створення складних інтерактивних дашбордів з багатьма пов'язаними компонентами за допомогою інструментів типу Dash (Plotly) або Panel (Bokeh) вимагає значних зусиль та знань, що виходять за рамки простої візуалізації, наближаючись до веброзробки [37];
- стандартні візуальні налаштування деяких бібліотек (наприклад, Matplotlib) є менш естетичними порівняно, наприклад, з ggplot2 в R або сучасними BI-інструментами. Це легко виправляється за допомогою налаштувань або використання додаткових бібліотек (як Seaborn), але вимагає додаткових зусиль [38].

Таким чином, мова програмування Python є потужним та гнучким інструментом для візуалізації даних, переваги якого часто переважають недоліки. Однак, вибір на користь Python має враховувати складність поставлених завдань, необхідний рівень інтерактивності, обсяги даних та кваліфікацію виконавця.

# РОЗДІЛ 2
# АНАЛІЗ ІНСТРУМЕНТІВ PYTHON ДЛЯ ВІЗУАЛІЗАЦІЇ ДАНИХ

## 2.1 Огляд основних бібліотек Python для візуалізації даних

Сьогодні існує велика кількість бібліотек Python для візуалізації даних, постійно з'являються нові бібліотеки, а деякі стають менш популярними. Нижче розглянуто основний список широко відомих та спеціалізованих бібліотек Python, що використовуються для візуального аналізу даних станом на квітень 2025 року, згрупованих за їх основними характеристиками та призначенням.

Основні загальновживані бібліотеки Python для візуального аналізу даних:

- Matplotlib – фундаментальна бібліотека для створення статичних, анімованих та інтерактивних візуалізацій у Python, багато інших бібліотек побудовані на її основі. Її характеризує високий рівень контролю над кожним елементом графіку, гнучкість, придатність для створення графіків публікаційної якості. Типи графіків: практично будь-які 2D та деякі 3D графіки (лінійні, стовпчикові, розсіювання, гістограми, секторні діаграми тощо) [39];
- Seaborn – бібліотека на основі Matplotlib, яка надає високорівневий інтерфейс для створення естетично привабливих та інформативних статистичних графіків. Її сильні сторони: спрощує створення складних статистичних візуалізацій (розподіли, залежності, категоріальні дані), має візуально привабливі стандартні стилі та палітри кольорів. Типи графіків: скрипкові діаграми (violin plots), теплокарти (heatmaps), парні графіки (pair plots), діаграми розсіювання з регресією, візуалізація категоріальних даних [40];
- Pandas Visualization – вбудований функціонал бібліотеки Pandas для роботи з даними, що надає методи для швидкої візуалізації даних, представлених одновимірними та двовимірними структурами Series та DataFrames. За замовчуванням використовує Matplotlib як бекенд. Сильні сторони: зручна для швидкого аналізу даних безпосередньо з об'єктів Pandas. Типи графіків: базові графіки (лінійні, стовпчикові, гістограми, розсіювання, щільності) [41].

Інтерактивні та веборієнтовані бібліотеки Python для візуального аналізу:

- Plotly – створює високоякісні, інтерактивні графіки, які легко вбудовуються у вебдодатки та Jupyter Notebooks. Має власну онлайн-платформу та інструменти для спільної роботи. Сильні сторони: інтерактивність візуалізацій (масштабування, панорамування графіків, відображення даних при наведенні); великий вибір типів графіків (включаючи 3D, фінансові, географічні); підтримка Dash для створення аналітичних вебдодатків Python для візуального аналізу даних. Типи графіків: широкий спектр 2D, 3D, анімованих, фінансових, наукових та статистичних графіків [42];

- Bokeh – схожа на Plotly, призначена для створення інтерактивних візуалізацій для веббраузерів, фокусується на високій продуктивності при роботі з великими або потоковими даними. Сильні сторони: інтерактивність, можливість створення складних інформаційних панелей (дашбордів) та додатків з віджетами, потокова передача даних, гнучкість налаштування. Типи графіків: широкий спектр інтерактивних 2D графіків, інструменти для зв'язування графіків та віджетів [43];

- Altair – декларативна бібліотека для статистичної візуалізації, заснована на граматиці візуалізації Vega-Lite. Користувач визначає зв'язки між даними та візуальними елементами, а Altair генерує графік. Сильні сторони: інтуїтивно зрозумілий синтаксис, легко створювати складні графіки, комбінуючи прості елементи, головне призначення у візуалізації статистичних зв'язків. Типи графіків: широкий спектр статистичних графіків [44].

Спеціалізовані бібліотеки Python для візуального аналізу даних:

- ggplot – Python-версія бібліотеки ggplot2 з мови R, базується на «граматиці графіки» (Grammar of Graphics), що дозволяє створювати графіки пошарово. Сильні сторони: гнучка граматика для створення складних візуалізацій, знайома користувачам R. Розвивається менш активно порівняно з іншими бібліотеками [45];

- Folium – дозволяє візуалізувати дані на інтерактивній карті Leaflet.js (відкрита JavaScript-бібліотека, яка використовується для створення інтерактивних карт для вебсайтів). Надає інструменти, які забезпечують взаємодію з Leaflet.js, не вимагаючи безпосереднього написання коду на JavaScript. Сильні сторони:

дозволяє легко створювати різні типи карт (маркери, теплокарти, хороплетні карти), інтеграція з Pandas. Типи графіків: картографічні візуалізації [46];

- Geopandas – розширює Pandas для роботи з геопросторовими даними. Має власні методи візуалізації, які використовують Matplotlib як бекенд. Сильні сторони: інтеграція аналізу та візуалізації геопросторових даних. Типи графіків: картографічні візуалізації (полігони, лінії, точки на карті) [47];

- NetworkX – бібліотека для створення, маніпулювання та вивчення структури, динаміки та функцій складних мереж (графів). Сильні сторони: фактично, є стандартом для роботи з графами в Python, має базові можливості для візуалізації структури графів (часто використовується разом з Matplotlib або іншими бібліотеками). Типи графіків: візуалізація графів та мереж [48];

- Missingno – має набір інструментів для візуалізації пропущених даних. Сильні сторони: корисна на етапі очищення та аналізу даних для виявлення закономірностей у відсутніх даних. Типи графіків: матриця пропусків, стовпчикова діаграма пропусків, теплокарта кореляції пропусків [49];

- Yellowbrick – набір візуальних інструментів, що розширюють API Scikit-Learn, для допомоги у виборі моделей машинного навчання, налаштуванні гіперпараметрів та діагностиці. Сильні сторони: інтеграція з процесом машинного навчання, візуалізація метрик, залишків, важливості ознак тощо. Типи графіків: криві навчання, візуалізація класифікаційних звітів, діаграми залишків, візуалізація важливості ознак [50].

Бібліотеки Python для візуального аналізу великих даних:

- Datashader – дозволяє візуалізувати великі набори даних шляхом перетворення їх у растрові зображення. Сильні сторони: продуктивність при роботі з великими даними, уникнення проблєми надмірного накладання точок (overplotting). Часто використовується разом з бібліотеками Bokeh, Plotly або HoloViews. Типи графіків: рендеринг великих наборів точок, ліній, полігонів [51];

- VisPy – високопродуктивна інтерактивна бібліотека для 2D/3D візуалізації наукових даних, що використовує потужність графічного процесора (GPU) через OpenG. Сильні сторони: швидкість, можливість роботи з дуже великими обсягами

даних у реальному часі, підходить для наукових та інженерних застосувань. Типи графіків: 2D та 3D наукові візуалізації, об'ємний рендеринг [52].

Екосистема HoloViz [53]:

- HoloViews – високорівнева бібліотека, що дозволяє створювати візуалізації незалежно від конкретного бекенду (наприклад, може використосвавати, як бекенд Matplotlib, Bokeh, Plotly), дозволяє легко додавати інтерактивність, досліджувати багатовимірні дані. Сильні сторони: гнучкість бекенду, легке створення інтерактивних панелей для дослідження параметрів;
- hvPlot – надає високорівневий API (схожий на Pandas .plot()), який генерує інтерактивні об'єкти HoloViews, що візуалізуються за допомогою Bokeh, Matplotlib або Plotly. Має простий синтаксис для створення інтерактивних графіків з об'єктів Pandas, GeoPandas, Xarray тощо.

Інші бібліотеки Python для візуального аналізу даних:

- Pygal – генерує інтерактивні SVG (Scalable Vector Graphics) графіки. Створює красиві векторні графіки, які добре масштабуються та легко вбудовуються у вебсторінки [54].

## 2.2 Порівняльна характеристика бібліотек Matplotlib, Pandas, Seaborn, Plotly

Серед усіх бібліотек Python для візуалізації даних провідні позиції займають бібліотеки Matplotlib, Pandas, Seaborn та Plotly. Кожна з цих бібліотек має свою філософію, а також як сильні сторони, так і обмеження, що робить їх більш або менш придатними для конкретних завдань.

Matplotlib є фундаментальною, низькорівневою бібліотекою для створення статичних, растрових та векторних візуалізацій у Python. Основна перевага її використання – повний контроль над кожним елементом графіка. Вона є основою (бекендом) для багатьох інших бібліотек, включаючи Pandas Visualization та Seaborn, надає переважно низькорівневий інтерфейс (хоча має модуль pyplot для

створення швидких скриптів). З Matplotlib графік будується послідовно додаючи та налаштовуючи окремі елементи (фігуру, осі, лінії, маркери, текст). Це забезпечує гнучкість та можливість унікальних налаштувань, дозволяючи створювати практично будь-які 2D (та деякі 3D) візуалізації [55].

Через свою низькорівневість та орієнтацію на детальний контроль, Matplotlib є більш складною для навчання порівняно з високорівневими бібліотеками. Створення складних, естетично привабливих графіків з Matplotlib вимагає значної кількості коду. Базова інтерактивність (панорамування, масштабування) доступна у вікнах, що генеруються певними бекендами бібліотеки Matplotlib, що не призначена для створення складних інтерактивних вебвізуалізацій. Стандартні налаштування стилю графіків Matplotlib є застарілими, хоча ця бібліотека має широкі можливості для налаштування стилів та використання тем.

Загалом, Matplotlib найкраще підходить для створення графіків для наукових публікацій, генерації статичних звітів, унікальних налаштувань візуалізацій та як основа для розробки специфічних графічних інструментів.

Pandas Visualization (.plot()) – це не окрема бібліотека, а набір методів візуалізації, вбудованих безпосередньо в об'єкти Pandas (DataFrame та Series). Основна мета – забезпечити швидкий та зручний спосіб візуалізації даних, що зберігаються в структурах Pandas, для розвідувального аналізу даних (EDA). Вона є високорівневою обгорткою, переважно над Matplotlib [56].

Синтаксис Pandas Visualization дуже лаконічний (df['column'].plot(kind='hist')). Гнучкість налаштувань візуалізації обмежена тими параметрами, які надає метод .plot(). Для більш гнучких налаштувань потрібно звертатися до об'єктів осей Matplotlib, які повертаються методом .plot(). Бібліотека дуже проста у використанні для стандартних типів графіків (лінійні, стовпчасті, гістограми, діаграми розсіювання тощо) безпосередньо з даних Pandas, і має мінімальний поріг входження для користувачів Pandas.

Загалом, бібліотека Pandas Visualization успадковує свою обмежену інтерактивність від бекенду Matplotlib, за замовчуванням використовує стиль

оформлення Matplotlib, оптимальна для швидкого розвідувального аналізу даних та перевірки гіпотез під час роботи з DataFrame/Series.

Seaborn – високорівнева бібліотека, також побудована на основі Matplotlib, яка спеціально розроблена для створення привабливих та інформативних статистичних візуалізацій. Вона спрощує візуалізацію складних взаємозв'язків у даних. Seaborn має високорівневий, декларативний інтерфейс: користувач визначає тип графіка та змінні даних, а Seaborn забезпечує статистичні агрегації та відображення. Хоча бібліотека Seaborn базується на Matplotlib і дозволяє доступ до базових об'єктів для подальших налаштувань, основна її гнучкість полягає у виборі та налаштуванні саме статистичних типів графіків. В цілому, бібліотека Seaborn значно простіша у використанні для складних статистичних графіків (наприклад, скрипкові діаграми, парні діаграми, теплові карти кореляцій, регресійні моделі), ніж якби їх довелося реалізовувати з нуля в Matplotlib. Добре інтегрується з Pandas DataFrames. Як і Pandas Visualization, успадковує обмежену інтерактивність Matplotlib. Однією з переваг Seaborn є значно покращена естетика за замовчуванням порівняно з Matplotlib, привабливі палітри кольорів та стилі графіків. Загалом, Seaborn добре підходить для статистичного аналізу та візуалізації, дослідження розподілів даних, взаємозв'язків між змінними, порівняння категорійних даних [57].

Plotly – це бібліотека, орієнтована на створення багатофункціональних, інтерактивних візуалізацій, які переважно призначені для відображення у вебсередовищі (хоча можуть працювати і офлайн). Використовує декларативний підхід і базується на бібліотеках JavaScript (Plotly.js), надає високорівневий інтерфейс (plotly.express) для швидкого створення багатьох типів інтерактивних графіків з мінімальним кодом, подібний до Seaborn. Має більш деталізований об'єктно-орієнтований інтерфейс (graph_objects), який пропонує глибоку гнучкість у налаштуванні кожного елемента інтерактивної візуалізації. Модуль plotly.express робить створення стандартних інтерактивних графіків за допомогою Plotly досить простим. Натомість, робота з graph_objects вимагає глибокого розуміння структури

фігур Plotly, що може ускладнює навчання та використання, але надає значно більший контроль [58].

Головна перевага Plotly – інтерактивність. Графіки, побудовані за допомогою Plotly, за замовчуванням є інтерактивними (дозволяють масштабування, панорамування, відображення даних при наведенні, можливість вибору/фільтрації даних). Plotly дуже добре підходить для створення інтерактивних дашбордів за допомогою фреймворку Dash, створює сучасні, привабливі та інтерактивні візуалізації зі стандартними налаштуваннями та темами, найкраще підходить для створення інтерактивних звітів та дашбордів, візуалізацій для вебсайтів та додатків, дослідження даних шляхом інтерактивної взаємодії, 3D-візуалізацій.

Таким чином, проведений аналіз показує, що кожна з розглянутих бібліотек займає свою нішу в екосистемі візуалізації даних Python: Matplotlib є потужною основою з максимальним контролем, Pandas Visualization – інструмент для швидкого розвідувального аналізу безпосередньо з даних, Seaborn – спеціалізоване рішення для привабливої статистичної візуалізації, Plotly – лідер у створенні інтерактивних веборієнтованих графіків та дашбордів. При цьому, ці бібліотеки не є взаємовиключними: Seaborn та Pandas Visualization використовують Matplotlib, і часто для налаштування графіків, створених за допомогою цих високорівневих інструментів, доводиться звертатися до об'єктів та методів Matplotlib.

## 2.3 Інструменти Python для візуалізації різних типів даних

Бібліотеки Python, що використовуються для візуалізації, є певною мірою універсальними. Однак, кожна з них має свої сильні сторони, що роблять їх оптимальними для візуалізації окремих типів даних та специфічних завдань [59].

При роботі з числовими даними важливо визначити розподіл окремих змінних. Для цього зазвичай використовують гістограми, графіки щільності (KDE), коробкові або скрипкові діаграми. Найзручнішою для такої візуалізації є бібліотека Seaborn, яка дозволяє швидко створювати інформативні й привабливі графіки,

зокрема комбінувати кілька типів відображення, наприклад, гістограму і KDE одночасно. Для швидкої оцінки розподілу даних безпосередньо у DataFrame використовують методи Pandas: .plot(kind='hist'), .plot(kind='kde') або .plot(kind='box'). Коли потрібен повний контроль над виглядом графіка для публікацій або звітів, доцільно застосовувати Matplotlib.

Для візуального вивчення взаємозв'язку між двома числовими змінними використовують діаграми розсіювання або лінійні графіки. Для їх побудови застосовуються інструменти Seaborn – графіки типу scatterplot та regplot, які дозволяють, наприклад, легко додавати лінії регресії та довірчі інтервали. Для комплексного аналізу одночасно розподілів та взаємозв'язків змінних використовується jointplot – функція з бібліотеки Seaborn, яка дозволяє візуалізувати зв'язок між двома змінними, одночасно показуючи їхні розподіли (у вигляді гістограм або оцінок щільності). Для інтерактивного масштабування або детального перегляду точок доцільно застосовувати Plotly або Bokeh. Для швидкої оцінки взаємозв'язків також використовується метод .plot.scatter() у Pandas.

Під час аналізу більше ніж двох числових змінних застосовується багатовимірний візуальний аналіз. Для цього доцільно створювати матриці діаграм розсіювання за допомогою Seaborn (pairplot), що дозволяє швидко виявити потенційні кореляції. Для оцінки сили взаємозв'язків часто будуються теплокарти кореляцій. У разі потреби створення більш оригінальних типів графіків, наприклад, з паралельними координатами або радіальних візуалізацій, використовуються специфічні інструменти Pandas або інтерактивні scatter_matrix у Plotly. Після застосування методів зниження розмірності (PCA або t-SNE) результати зазвичай візуалізуються статично за допомогою Matplotlib, Seaborn або інтерактивно з використанням Plotly.

Для роботи з категоріальними даними застосовується інший набір методів візуалізації. Для вивчення розподілу частот однієї змінної використовуються стовпчикові діаграми, що будуються через Seaborn (countplot) або Pandas (value_counts().plot(kind='bar')). Секторні діаграми (pie charts) також доступні через Pandas або Matplotlib, однак тут вони використовуються обмежено через складність

їх сприйняття при великій кількості категорій. Для вивчення залежностей між двома категоріальними змінними застосовуються груповані стовпчикові діаграми, які створюються за допомогою Seaborn catplot із параметром hue. У випадку складнішого аналізу використовуються мозаїчні графіки, що будуються за допомогою бібліотеки Statsmodels.

Для дослідження розподілу числових змінних за категоріями використовуються коробкові діаграми, скрипкові діаграми, а також точкові та стовпчикові графіки з довірчими інтервалами. Для цього найчастіше застосовується бібліотека Seaborn через функції boxplot, violinplot, stripplot, swarmplot, barplot та pointplot.

У роботі з часовими рядами вибір інструменту залежить від завдання. Для швидкої візуалізації трендів використовуються методи Pandas .plot(), які автоматично враховують часовий індекс. Для детального налаштування графіків застосовується Matplotlib. У випадку необхідності відображення кількох часових серій або довірчих інтервалів обирається Seaborn lineplot. Для інтерактивного масштабування та вибору часових діапазонів доцільно використовувати Plotly або Bokeh. Для аналізу сезонних компонентів і трендів застосовується функція seasonal_decompose із бібліотеки Statsmodels.

При роботі з геопросторовими даними для створення інтерактивних карт використовують Folium, який дозволяє легко додавати маркери, кластери та теплові карти. Для швидкої візуалізації геометрії об'єктів застосовується Geopandas у поєднанні з Matplotlib. Для побудови інтерактивних хороплетних карт або глобусів високої якості використовуються інструменти Plotly, особливо його модулі scatter_geo та choropleth. Для обробки великих обсягів геоданих ефективними є Datashader разом із HoloViews або GeoViews.

Для візуалізації мережевих даних (графів) основним інструментом є NetworkX, який дозволяє створювати базові візуалізації через Matplotlib. У разі потреби інтерактивності використовуються Pyvis або Plotly, де можна окремо налаштовувати вузли та ребра. Для створення високоякісних статичних графів або презентацій застосовуються спеціалізовані інструменти, як Graphviz або Gephi.

Для виявлення пропущених значень у даних їх структуру доцільно візуалізувати за допомогою бібліотеки Missingno, яка дозволяє створювати матриці відсутності, стовпчикові діаграми або теплокарти кореляції пропусків.

Під час роботи з надвеликими наборами даних, де звичайні інструменти Python втрачають свою ефективність, використовують бібліотеку Datashader для агрегації та растеризації великих обсягів точок перед рендерингом. Для помірно великих наборів даних ефективною є оптимізація Plotly або Bokeh із підтримкою WebGL. У випадках роботи з табличними даними, що не вміщаються в оперативну пам'ять, застосовується Vaex, а для обробки даних у реальному часі – VisPy.

Вибір інструментів візуалізації також залежить від поставлених завдань: для інтерактивного дослідження даних використовуються Plotly, Bokeh, Altair або hvPlot; для високоякісних статичних візуалізацій – Matplotlib та Seaborn; для швидких базових графіків – вбудовані засоби Pandas; для створення складних статистичних графіків – Seaborn або Altair. Важливо також враховувати інтеграцію інструментів у загальний робочий процес: наприклад, Yellowbrick добре поєднується зі Scikit-learn, Geopandas оптимально підходить для геоаналізу, а стек бібліотек HoloViz дозволяє будувати комплексні пайплайни візуалізації.

## 2.4 Інструменти Python для різних завдань аналізу даних

Оптимальний вибір інструменту візуалізації Python залежить від типу даних, мети візуалізації, цільової аудиторії, необхідного рівня інтерактивності, розміру датасету та бажаного ступеня налаштувань. Найкращі результати досягаються шляхом комбінування інструментів: швидкий аналіз за допомогою Pandas/Seaborn, створення інтерактивних дашбордів з Plotly/Bokeh, підготовка графіків для публікації за допомогою Matplotlib/Seaborn, та використання спеціалізованих інструментів (Folium, NetworkX, Missingno, Datashader) для конкретних типів даних або завдань.

Маючи детальне уявлення про характеристики бібліотек Matplotlib, Pandas Visualization, Seaborn та Plotly (як описано в п. 2.2), можна окреслити сфери застосування різних інструментів Python залежно від цілей візуалізації.

Швидкий розвідувальний аналіз даних (EDA). Основний інструмент: Pandas Visualization (.plot()). Аргументація: найшвидший та найзручніший спосіб отримати базове візуальне уявлення про дані безпосередньо з об'єктів DataFrame або Series. Синтаксис лаконічний, інтеграція безшовна. Ідеальний для первинного огляду тенденцій, розподілів та викидів. Приклади використання: df['column'].plot(kind='hist'), df.plot(kind='line'), df.plot(kind='scatter', x='col1', y='col2'). Альтернативи: Seaborn – функції типу seaborn.pairplot() або seaborn.heatmap(df.corr()) можуть швидко надати комплексне уявлення про взаємозв'язки між багатьма змінними; Plotly Express – модуль plotly.express також дозволяє швидко створювати інтерактивні графіки з мінімальним кодом (наприклад, plotly.express.scatter(df, x='col1', y='col2')), що може бути корисним для інтерактивного EDA.

Візуалізація розподілів (одномірних числових даних). Основний вибір: Seaborn. Аргументація: Seaborn пропонує спеціалізовані функції (histplot, kdeplot, ecdfplot, boxplot, violinplot), які легко створюють відповідні графіки, а також мають гарну естетику та можливість легко порівнювати розподіли за категоріями (використовуючи параметри hue, x/y). Функції часто інтегрують статистичні оцінки (напр., KDE в histplot). Приклади: seaborn.histplot(data=df, x='value', kde=True), seaborn.boxplot(data=df, y='value', x='category'). Альтернативи: Plotly – надає інтерактивні аналоги (plotly.express.histogram, plotly.express.box, plotly.express.violin), які корисні для дослідження деталей розподілу через масштабування та відображення даних при наведенні; Pandas Visualization – df['column'].plot(kind='hist') або df.plot(kind='box') для швидких базових графіків; Matplotlib: matplotlib.pyplot.hist(), matplotlib.pyplot.boxplot() – вибір, коли потрібен повний контроль над кожним елементом графіка, але вимагає більше коду.

Візуалізація взаємозв'язків (двох або більше числових змінних). Основний вибір: Seaborn (для статичних) та Plotly (для інтерактивних). Аргументація: Seaborn

– функції scatterplot та regplot дозволяють легко візуалізувати зв'язок між двома змінними, додавати кодування за кольором/розміром для третьої/четвертої змінної та вбудовувати регресійні моделі, функції pairplot та jointplot ефективно показують парні взаємозв'язки та розподіли багатьох змінних, heatmap найкраще підходить для візуалізації матриць кореляцій; Plotly – plotly.express.scatter, plotly.express.scatter_matrix створюють інтерактивні версії, де можна масштабувати, панорамувати, виділяти групи точок та бачити точні значення при наведенні, що особливо корисно при дослідженні щільних хмар точок або складних залежностей; plotly.graph_objects.Figure дозволяє створювати складні комбіновані візуалізації. Альтернативи: Pandas Visualization – df.plot(kind='scatter', x='col1', y='col2') для дуже швидкого, базового огляду; Matplotlib – matplotlib.pyplot.scatter() надає повний контроль для створення статичних діаграм розсіювання, що налаштовуються.

Візуальний аналіз часових рядів. Основний вибір: Pandas Visualization (для швидкого статичного) та Plotly (для інтерактивного). Аргументація (Pandas): якщо дані знаходяться в Pandas Series або DataFrame з часовим індексом (DatetimeIndex), метод .plot() за замовчуванням створює лінійний графік, коректно обробляючи часову вісь. Аргументація (Plotly): plotly.express.line дозволяє легко створювати інтерактивні лінійні графіки, які особливо корисні для дослідження довгих часових рядів завдяки можливостям масштабування, вибору діапазонів та відображення значень. Альтернативи: Matplotlib – matplotlib.pyplot.plot() дає повний контроль над виглядом статичного графіка часового ряду (форматування дат, міток, стилів ліній); Seaborn – seaborn.lineplot() може бути корисним при візуалізації кількох рядів з відображенням довірчих інтервалів, особливо якщо є агрегація за часовими інтервалами або категоріями.

Візуальний аналіз категоріальних даних. Основний вибір: Seaborn. Аргументація: Seaborn має найбагатший набір функцій, спеціально призначених для візуалізації категоріальних даних: countplot (підрахунок частот), barplot (відображення агрегованих значень, напр., середнього, з довірчими інтервалами), boxplot, violinplot, stripplot, swarmplot (для порівняння розподілу числової змінної

між категоріями). Ці функції легко обробляють групування за кількома категоріальними змінними (hue, col, row). Приклади: seaborn.countplot(data=df, x='category'), seaborn.barplot(data=df, x='category', y='value'). Альтернативи: Plotly – plotly.express.bar, plotly.express.box, plotly.express.violin пропонують інтерактивні аналоги для дослідження категоріальних даних; Pandas Visualization – df['category'].value_counts().plot(kind='bar') або df.groupby('category')['value'] .mean().plot(kind='bar') є швидкими способами візуалізації агрегованих даних; Matplotlib – matplotlib.pyplot.bar() вимагає попередньої агрегації даних, але надає контроль над виглядом стовпчастих діаграм.

Створення статичних графіків для публікацій. Основний вибір: Matplotlib (часто у поєднанні з Seaborn). Аргументація: Matplotlib надає низькорівневий контроль над кожним елементом візуалізації (розміри шрифтів, товщини ліній, позиції міток, кольори, роздільна здатність), що є необхідним для відповідності вимогам наукових журналів чи звітів; Seaborn можна використовувати для швидкого створення основи графіка з гарною естетикою та коректними статистичними відображеннями, а потім фінальні налаштування робити за допомогою методів Matplotlib. Альтернативи: Plotly – може експортувати статичні зображення (за допомогою додаткової бібліотеки kaleido), але його основна сила в інтерактивності.

Створення інтерактивних візуалізацій та дашбордів. Основний вибір: Plotly (часто з використанням фреймворку Dash). Аргументація: Plotly за своєю природою генерує інтерактивні HTML/JavaScript візуалізації, надає широкий спектр інтерактивних можливостей за умовчанням. У поєднанні з бібліотеками Dash, дозволяє будувати повноцінні вебдодатки та аналітичні панелі (дашборди) з взаємопов'язаними компонентами (графіками, таблицями, фільтрами). Альтернативи: Bokeh є іншою потужною бібліотекою для створення інтерактивних вебвізуалізацій та дашбордів, але вона не входила до списку для порівняння в цьому пункті; Matplotlib, Pandas та Seaborn не призначені для створення складних інтерактивних вебдодатків.

Таким чином, вибір оптимального інструменту серед Matplotlib, Pandas Visualization, Seaborn та Plotly залежить від конкретного етапу аналізу та кінцевої мети візуалізації: для швидкого первинного аналізу даних (EDA) доцільно використовувати Pandas Visualization; для глибокого статистичного аналізу та створення естетично привабливих статичних графіків розподілів та взаємозв'язків, особливо з категоріальними даними, найкраще підходить Seaborn; для інтерактивного дослідження даних, створення вебвізуалізацій та дашбордів – Plotly; для максимального контролю, гнучких налаштувань та підготовки графіків для публікацій – Matplotlib (за потреби доповнюючи можливостями Seaborn для пришвидшення роботи та покращення базової естетики).

## 2.5 Практична візуалізація наборів даних за допомогою Python

Для демонстрації застосування, порівняння можливостей та підтвердження теоретичних висновків (пп. 2.2 – 2.3) щодо бібліотек Matplotlib, Pandas Visualization, Seaborn та Plotly, було проведено експериментальне дослідження, метою якого є візуалізація набору даних для вирішення типових аналітичних завдань за допомогою кожної з цих бібліотек, аналізуючи при цьому процес написання коду, гнучкість налаштувань та характеристики отриманих візуалізацій.

Для експерименту був обраний класичний набір даних «Tips» доступний через бібліотеку Seaborn, який містить дані про чайові, залишені в ресторані, разом із супутніми характеристиками рахунків та клієнтів. Цей набір даних має як числові змінні (total_bill, tip, size), так і категоріальні змінні (sex, smoker, day, time). Тому його можна використати для демонстрації різних типів візуалізацій.

Код Python що завантажує набір даних «Tips» та виводить таблицю даних з першими 5 рядками, інформацію про стовпці total_bill, tip, sex, smoker, day, time, size, типи даних, наявність/відсутність пропущених значень, а також базові статистичні показники змінних, представлений у додатку Б (лістинг Б.1). Результат роботи коду представлений на рисунку 2.1.

```
Перші 5 рядків даних:
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4

Інформація про типи даних та відсутні значення:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    category
 3   smoker      244 non-null    category
 4   day         244 non-null    category
 5   time        244 non-null    category
 6   size        244 non-null    int64
dtypes: category(4), float64(2), int64(1)
memory usage: 7.4 KB

Описова статистика числових стовпців:
       total_bill         tip        size
count  244.000000  244.000000  244.000000
mean    19.785943    2.998279    2.569672
std      8.902412    1.383638    0.951100
min      3.070000    1.000000    1.000000
25%     13.347500    2.000000    2.000000
50%     17.795000    2.900000    2.000000
75%     24.127500    3.562500    3.000000
max     50.810000   10.000000    6.000000
```

Рисунок 2.1 – Базова інформація про набір даних «Tips»

Далі на цьому прикладі розглядаються деякі типові завдання візуального аналізу даних та їх виконання за допомогою різних бібліотек Python.

Завдання 1. Швидкий огляд розподілу кількісної змінної – огляд загальної суми рахунку (total_bill). Мета: отримати перше уявлення про те, як розподіляються суми рахунків за величиною.

Використання бібліотеки Pandas Visualization – найшвидший спосіб для виконання EDA (Express Data Analysis). Мінімальний код представлений у додатку Б (лістинг Б.2), виводить базову гістограму розподілу змінної total_bill, що має простий стандартний вигляд (рисунок 2.2).

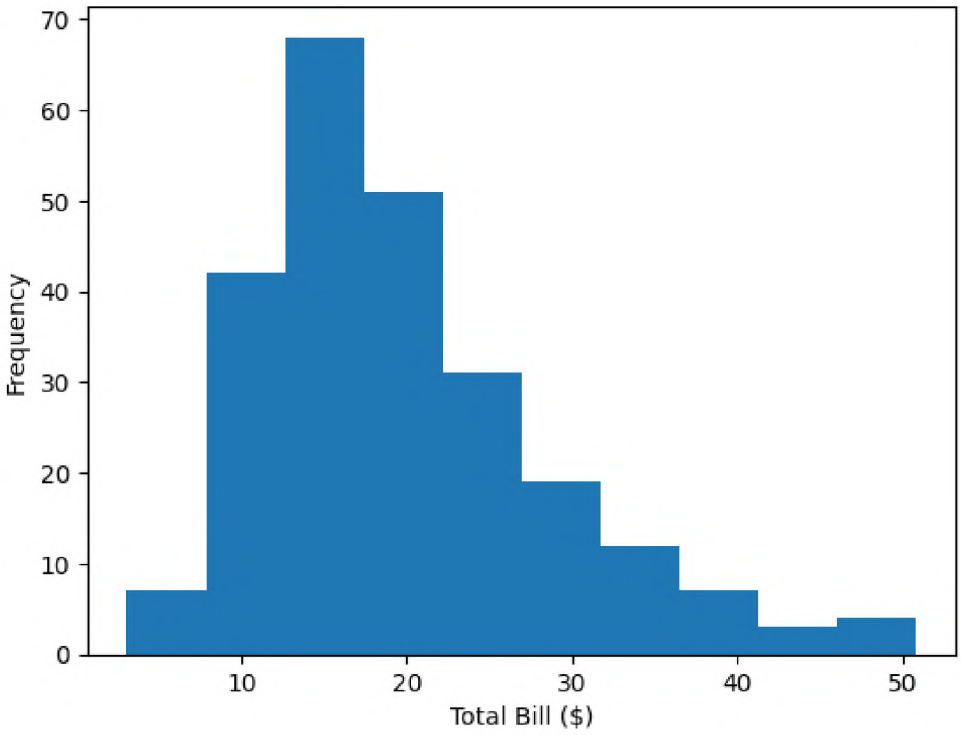


Рисунок 2.2 – Базова гістограма, побудована з використанням Pandas Visualization

Код Python, що використовує бібліотеку Seaborn (додаток Б, лістинг Б.2), виводить гістограму з кривою оцінки щільності (KDE) (рисунок 2.3).

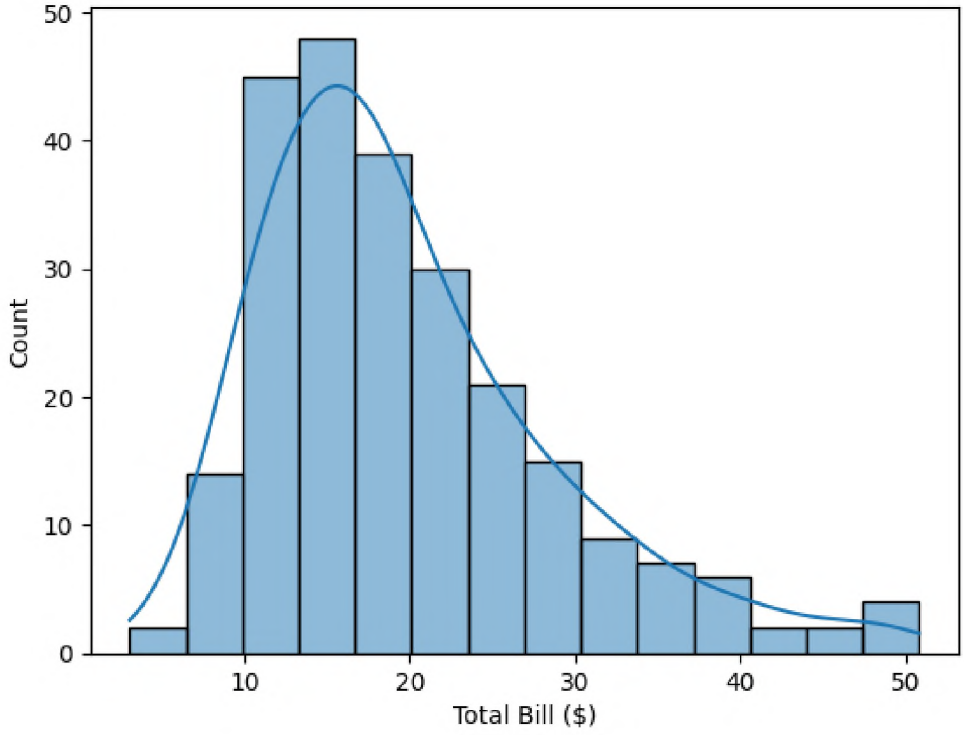


Рисунок 2.3 – Гістограма з KDE, побудована з використанням бібліотеки Seaborn

Бібліотека Plotly Express дозволяє створити інтерактивну гістограму (додаток Б, лістинг Б.2), де можна, наприклад, наводити курсор на стовпці для отримання точних значень та масштабувати графік (рисунок 2.4).

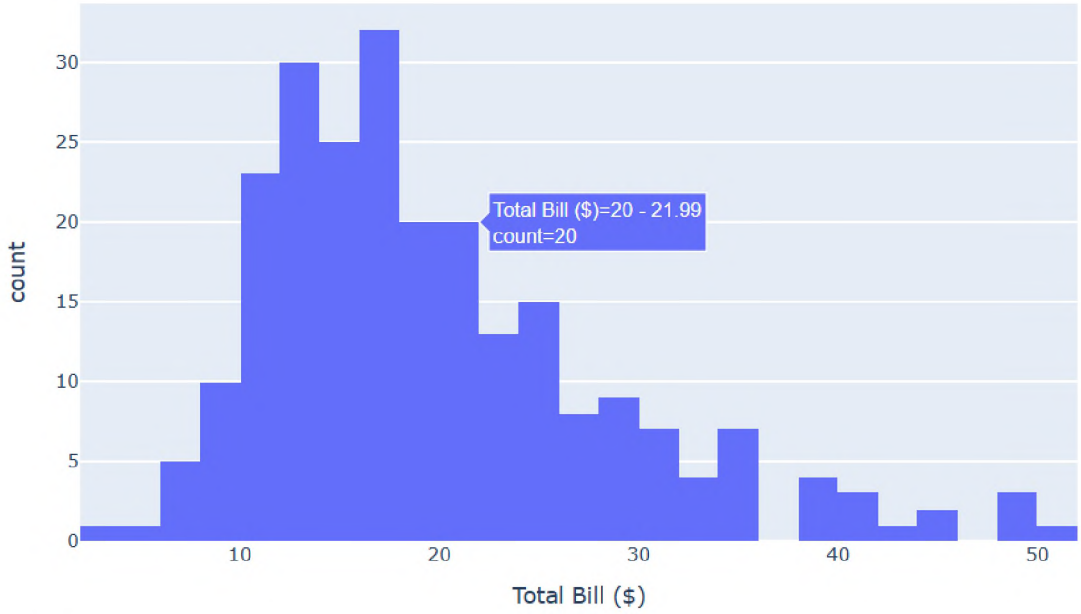


Рисунок 2.4 – Інтерактивна гістограма, побудована з використанням Plotly Express

Бібліотека Matplotlib надає повний контроль над побудовою графіку (додаток Б, лістинг Б.2), оскільки параметри графіку задаються явно (рисунок 2.5).

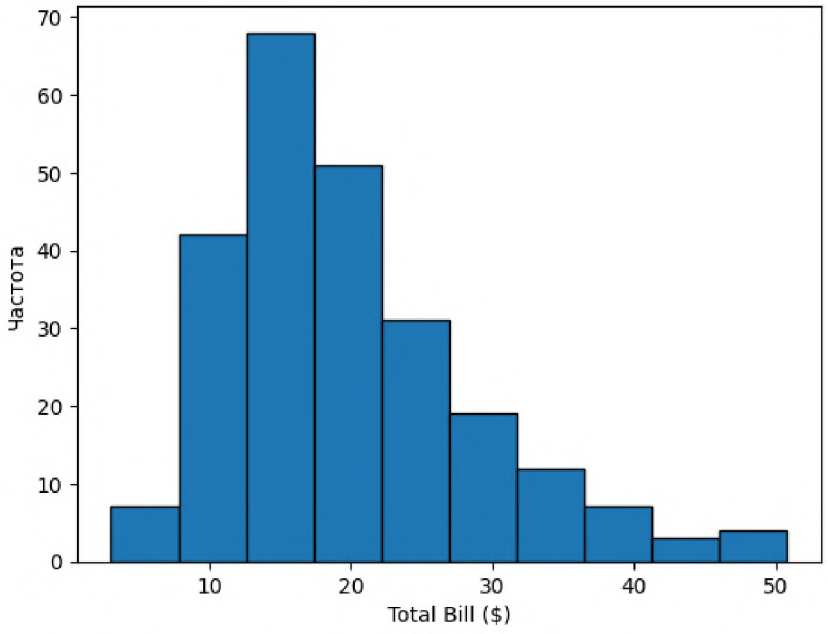


Рисунок 2.5 – Гістограма, побудована з використанням Matplotlib

Таким чином, можна зробити висновок, що бібліотека Pandas є найбільш простим інструментом Python для швидкого огляду даних (EDA), бібліотека Seaborn надає кращий зовнішній вигляд та статистичні доповнення з мінімальним кодом, бібліотека Plotly забезпечує інтерактивність, бібліотека Matplotlib вимагає найбільше коду, але дозволяє гнучке й точне налаштування параметрів графіку.

Завдання 2. Дослідити залежність між двома числовими змінними – залежність між сумою чайових (tip) та загальною сумою рахунку (total_bill). Мета: перевірити, чи існує зв'язок між сумою рахунку (total_bill) та розміром чайових (tip), і чи впливає на це статус курця (smoker).

Для візуалізації залежностей з урахуванням категорій найкраще підходить побудова діаграми розсіювання за допомогою бібліотеки Seaborn. Код (додаток Б, лістинг Б.3) виводить діаграму розсіювання, де точки на графіку розфарбовані відповідно до статусу курця (smoker). Також, Seaborn автоматично створює легенду. Отриманий графік дозволяє легко побачити загальну тенденцію та можливі відмінності між групами (рисунок 2.6).

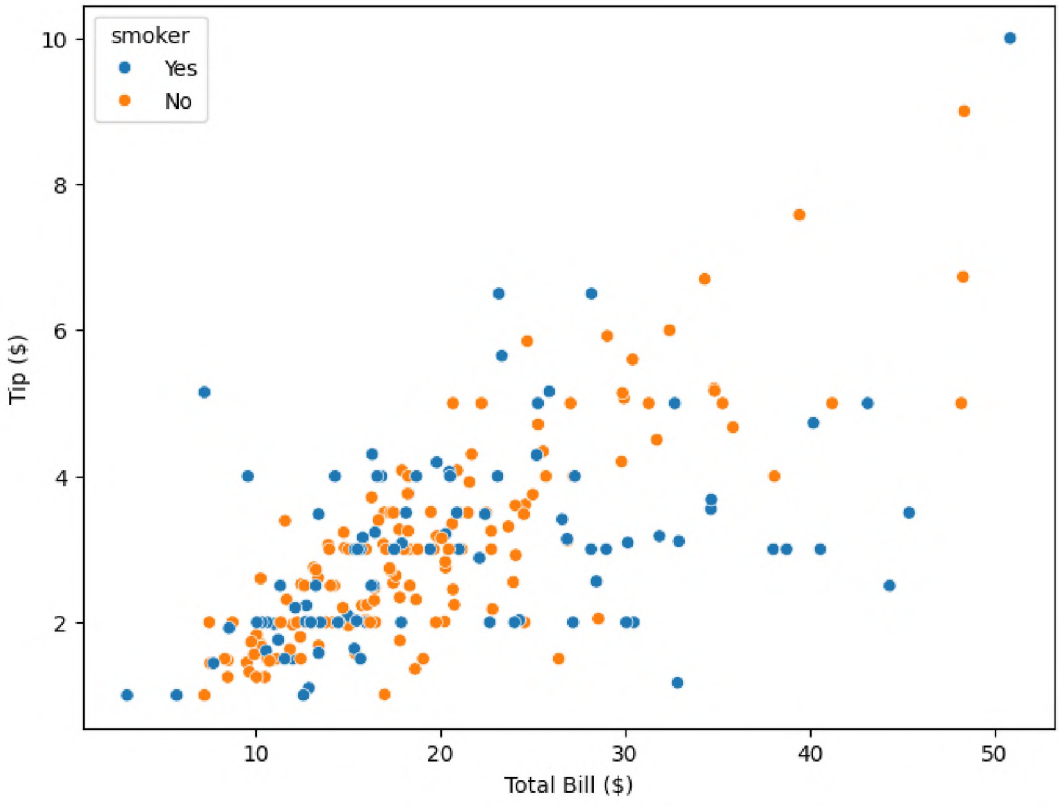


Рисунок 2.6 – Діаграма розсіювання, побудована з використанням Seaborn

Бібліотека Plotly Express дозволяє створити інтерактивну діаграму розсіювання (додаток Б, лістинг Б.3), де для аналізу даних та одержання додаткової інформації можна наводити курсор миші на точки графіка, масштабувати графік, вмикати/вимикати групи з легенди (рисунок 2.7).

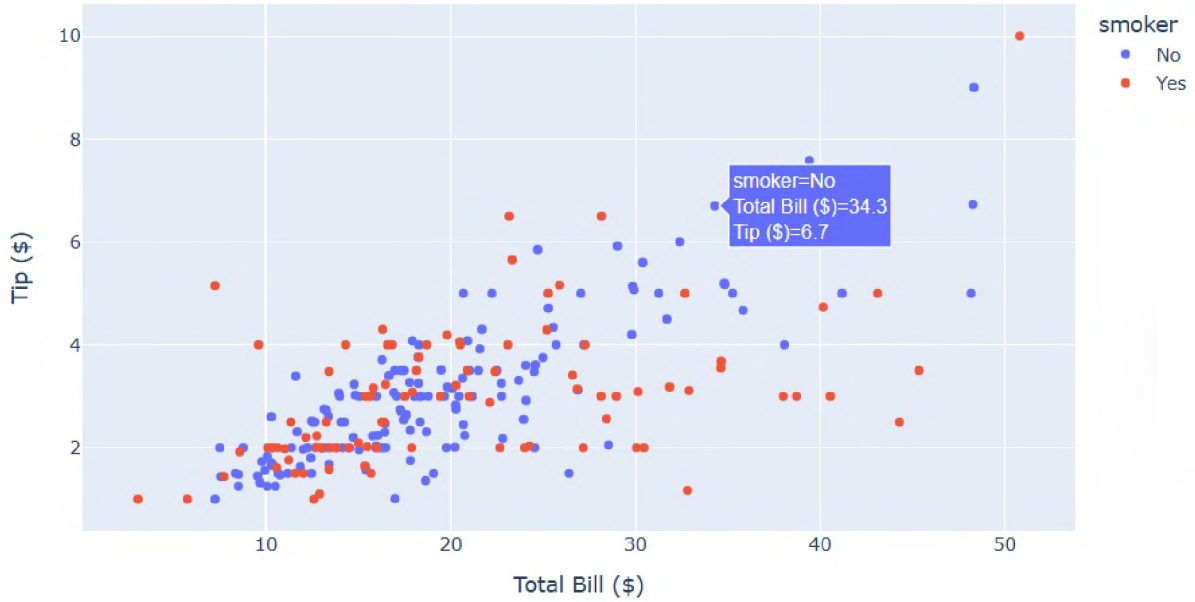


Рисунок 2.7 – Інтерактивна діаграма розсіяння з використанням Plotly Express

Використовуючи Matplotlib (додаток Б, лістинг Б.3), можна побудувати лише статичну діаграму розсіювання, аналогічну як за допомогою Seaborn (рисунок 2.8).

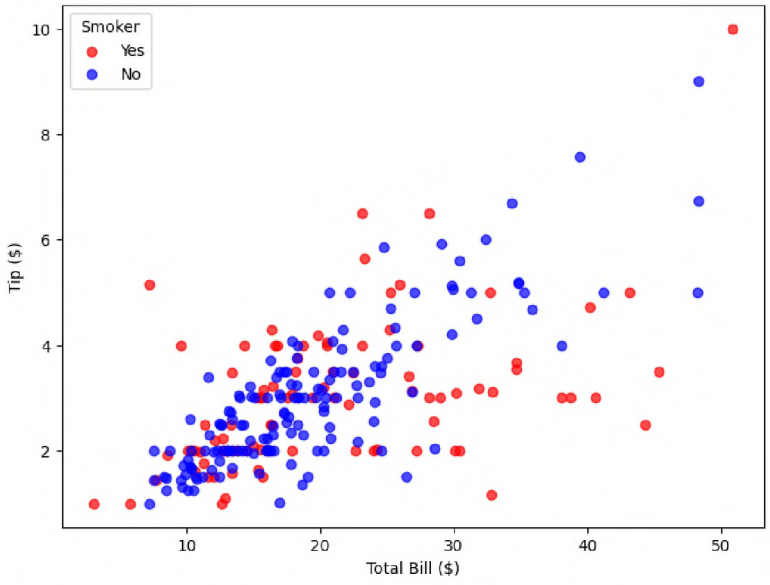


Рисунок 2.8 – Діаграма розсіяння, побудована з використанням Matplotlib

Побудова комбінованої діаграми розсіювання за допомогою Matplotlib вимагає більше коду для розфарбовування точок за категоріями та створення легенди, порівняно з Seaborn (додаток Б, лістинг Б.3).

Окремо розглянемо, як вирішити завдання 2, використовуючи інструменти візуалізації бібліотеки Pandas.

Метод 1. Побудова базової діаграми розсіювання (без поділу за категоріями) за допомогою Pandas.

Найпростіший спосіб візуалізувати залежність між двома числовими змінними за допомогою Pandas – викликати метод .plot() з параметром kind='scatter' (додаток Б, лістинг Б.3). В результаті отримуємо просту статичну діаграму розсіювання, яка показує загальну тенденцію залежності чайових від суми рахунку. Однак, ця діаграма не містить інформації про статус курця (рисунок 2.9).

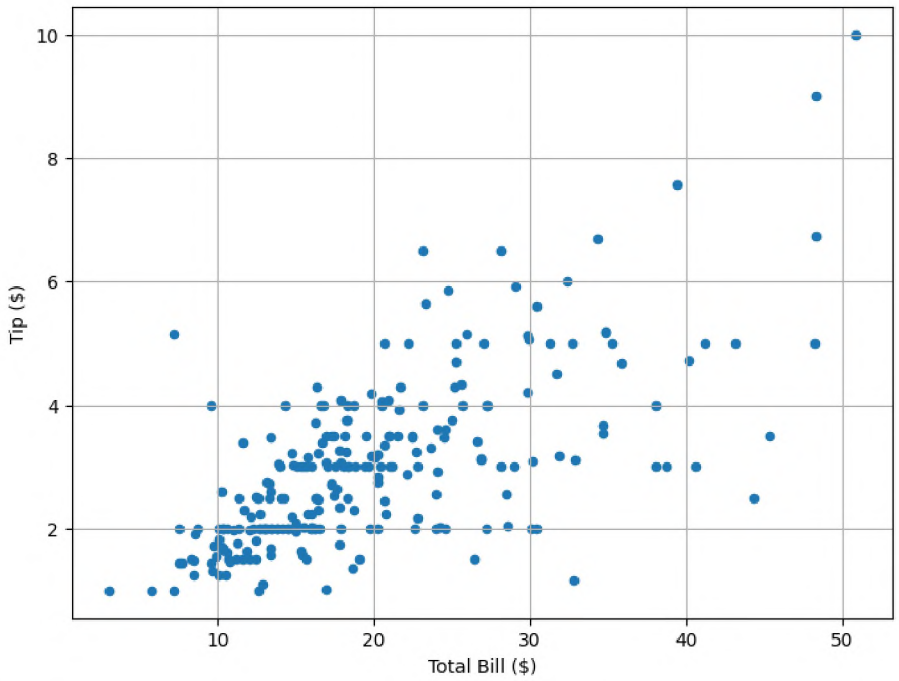


Рисунок 2.9 – Базова діаграма розсіяння (без поділу за категоріями за статусом Smoker) з використанням бібліотеки Pandas

Тут спочатку створюються об'єкти фігури та осей Matplotlib (fig, ax), потім викликається метод .plot() для DataFrame tips_df, вказується тип графіка (kind='scatter') та стовпці для осей x та y. Передача об'єкту ax в метод .plot(),

гарантує, що графік буде намальований саме на цих осях, що дозволяє потім налаштувати заголовок, мітки тощо за допомогою стандартних методів Matplotlib (ax.set_title, ax.set_xlabel).

Метод 2. Діаграма розсіювання з розфарбовуванням за категоріями (комбінований підхід) за допомогою Pandas.

Пряме автоматичне розфарбовування за категоріальною змінною з коректною легендою однією командою .plot() у Pandas є складним завданням (на відміну від seaborn.scatterplot(..., hue='smoker')). Параметр c в df.plot(kind='scatter') зазвичай очікує числові значення для відображення на колірній шкалі. Тому, щоб досягти бажаного результату, можна згрупувати дані за категоріальною змінною (smoker) і вивести кожну групу окремо на тих самих осях Matplotlib. Отриманий графік представлений на рисунку 2.11.

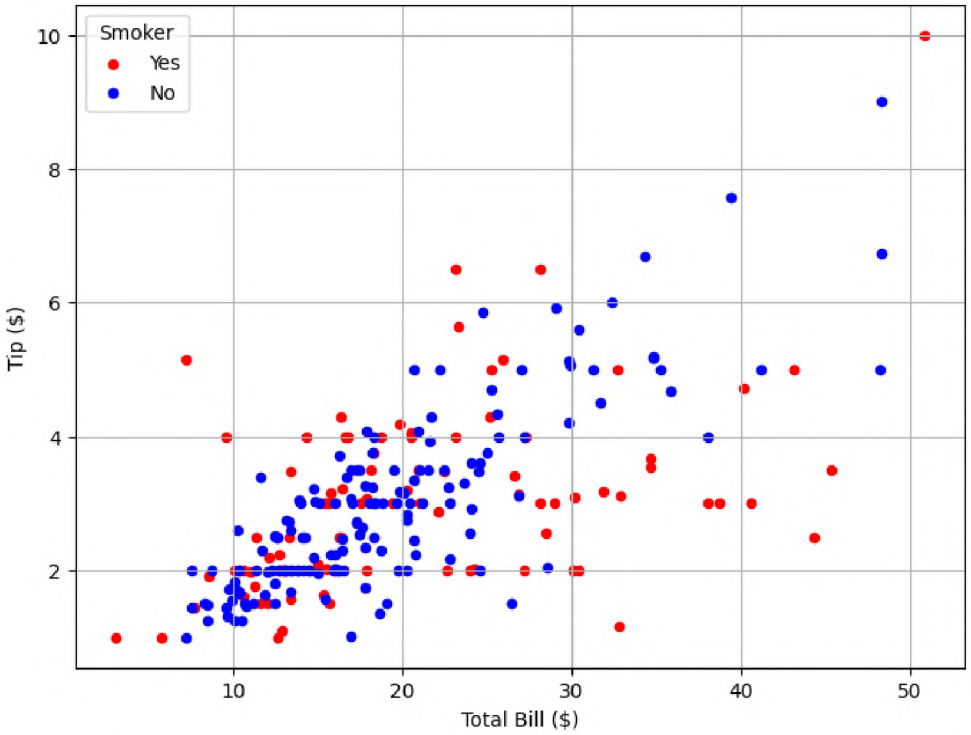


Рисунок 2.11 – Комбінована діаграма розсіювання даних з використанням Pandas

Код реалізації цієї комбінованої діаграми з поділом за категорією 'smoker' у Pandas представлений у додатку Б, лістинг Б.4. У цьому коді спочатку створюється фігура (полотно) та осі Matplotlib. Далі визначається словник colors для зіставлення

значень ‘Yes’/’No’ з конкретними кольорами. Групується DataFrame за стовпцем smoker за допомогою tips_df.groupby(‘smoker’). Виконується ітерація по отриманих групах. key містить значення групи (‘Yes’ або ‘No’), a group – це під-DataFrame, що містить лише дані для цієї групи. Для кожного group викликається .plot(kind=‘scatter’, ...), передаючи ті самі осі ax=ax_pd_scatter_color. Вказуємо label=key (щоб Matplotlib знав, як підписати елемент легенди) та color=colors[key]. Після циклу викликається ax.legend() для відображення легенди, створеної на основі label, переданих у кожному виклику .plot(). В результаті виводиться статична діаграма розсіювання, де точки розфарбовані відповідно до статусу курця, і присутня відповідна легенда. Візуально результат схожий на той, що генерується Seaborn або прямим кодом Matplotlib з циклом, але він досягається за допомогою групування Pandas та його методу .plot(), викликаного для кожної групи.

Блок-схема алгоритму побудови комбінованої діаграми розсіювання з поділом за категорією ‘smoker’ з Pandas представлена у додатку В (рисунок В.1).

Очевидно, що Pandas більш зручний для створення базової діаграми розсіювання між двома числовими змінними. Натомість, для додавання розфарбовування за категоріями з легендою Pandas вимагає ручного групування даних та ітеративного нанесення кожної групи на графік з використанням спільного об’єкта осей Matplotlib.

Таким чином, бібліотека Seaborn виявилась найбільш ефективною для створення статичних діаграм розсіювання, а Plotly – для інтерактивних. Matplotlib теж дозволяє створювати статичні діаграми, але потребує більше коду, як і Pandas, що вимагає застосування додаткових прийомів.

Завдання 3. Порівняти значення кількісної змінної за категоріями – порівняння суми рахунку (total_bill) за днями тижня (day). Мета: візуально порівняти розподіли сум рахунків у різні дні тижня.

Бібліотека Seaborn пропонує для вирішення цього завдання використання коробкових діаграм (boxplot) або скрипкових діаграм (violinplot).

Коробкова діаграма, створена з Seaborn (додаток Б, лістинг Б.5) показує медіану, квантилі та викиди за сумами рахунку (total_bill) для кожного дня тижня (day), і є інформативною для порівняння розподілів цих даних (рисунок 2.12).

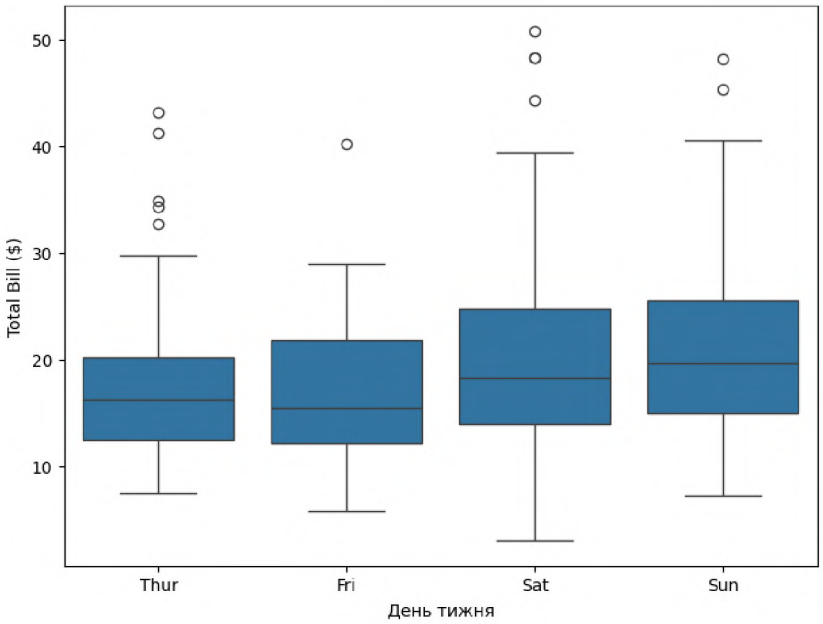


Рисунок 2.12 – Коробкова діаграма, побудована з використанням Seaborn

Код з Plotly Express дозволяє створити інтерактивну коробкову та/або скрипкову діаграму (додаток Б, лістинг Б.5), що відображає не тільки основні статистичні показники (як boxplot), але й форму розподілу (рисунок 2.13).

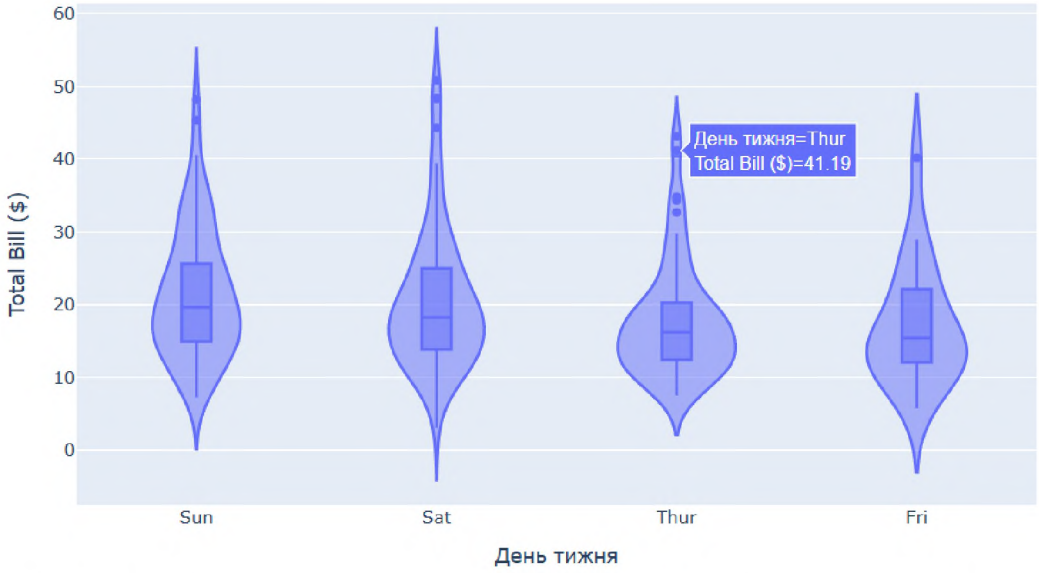


Рисунок 2.13 – Інтерактивна скрипкова діаграма з використанням Plotly Express

Побудова статичної коробкової діаграми з Matplotlib (рисунок 2.14), вимагає ручної підготовки списку масивів даних для кожної категорії (кожного дня тижня (day)) (додаток Б, лістинг Б.5).

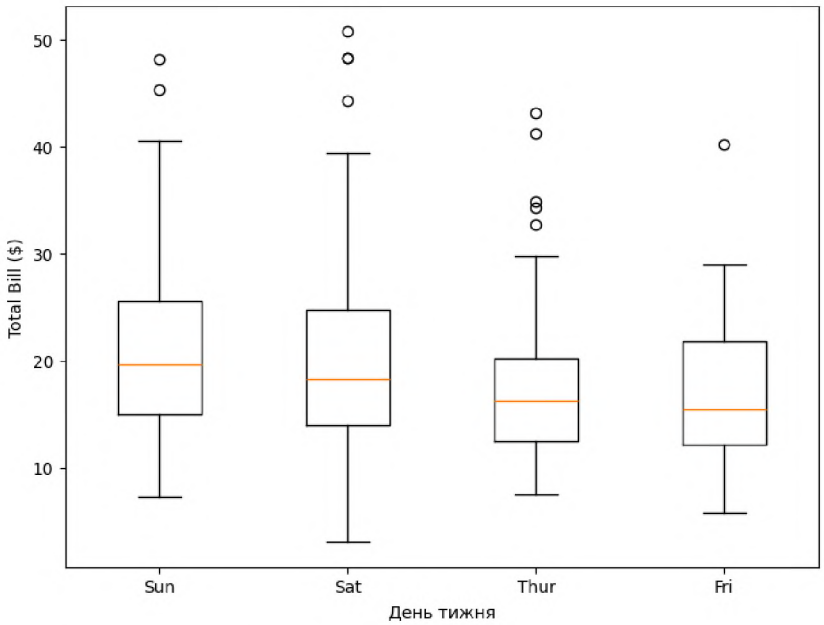


Рисунок 2.14 – Коробкова діаграма, побудована з використанням Matplotlib

Порівняння суми рахунку (total_bill) за днями тижня (day) з використанням бібліотеки Pandas представлене на рисунку 2.15.

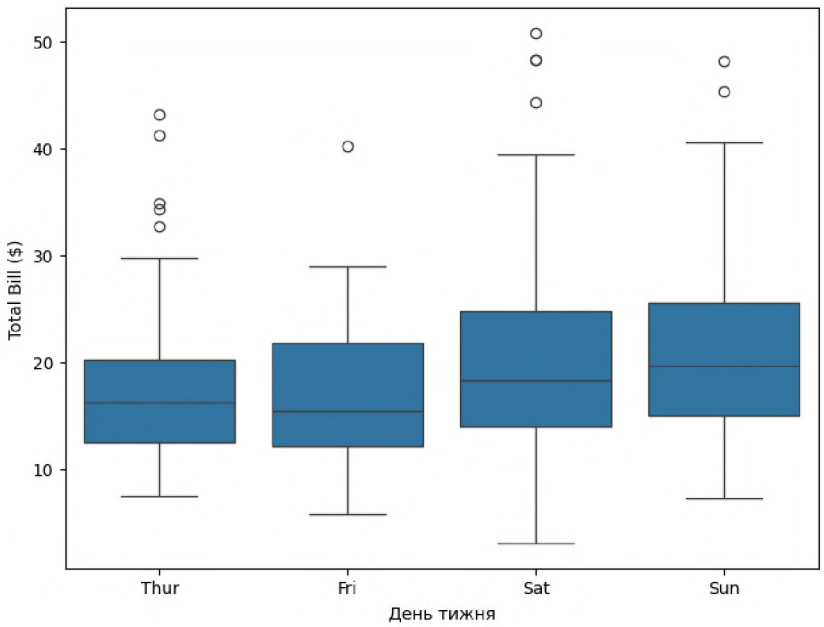


Рисунок 2.15 – Коробкова діаграма, побудована за допомогою pandas

Код реалізації коробкової діаграми (додаток Б, лістинг Б.5) спочатку імпортує необхідні бібліотеки – pandas для маніпулювання даними, seaborn для створення графіків та matplotlib.pyplot для відображення графіків. Потім завантажується набір даних tips. Далі створюється коробкова діаграма, на якій відображається розподіл total_bill для кожного значення в стовпці day. Також, цей код виводить описову статистику total_bill для кожної групи, визначеної стовпцем day (рисунок 2.16).

```
Описова статистика Total Bill за днями тижня:
      count       mean       std   min      25%    50%      75%    max
day
Thur   62.0  17.682742  7.886170  7.51  12.4425  16.20  20.1550  43.11
Fri    19.0  17.151579  8.302660  5.75  12.0950  15.38  21.7500  40.17
Sat    87.0  20.441379  9.480419  3.07  13.9050  18.24  24.7400  50.81
Sun    76.0  21.410000  8.832122  7.25  14.9875  19.63  25.5975  48.17
```

Рисунок 2.16 – Розрахунок описових статистик за допомогою pandas

Таким чином, бібліотека Seaborn значно спрощує створення порівняльних статистичних графіків. Plotly додає інтерактивність та альтернативні представлення. Matplotlib знову вимагає більше підготовчої роботи з даними. Pandas потребує низки додаткових маніпуляцій, натомість дозволяє легко маніпулювати даними та виконувати статистичні розрахунки.

Загалом, проведене експериментальне дослідження на наборі даних «Tips» підтвердило висновки, зроблені під час теоретичного порівняння бібліотек.

# РОЗДІЛ 3
# РОЗРОБЛЕННЯ МЕТОДИКИ ВІЗУАЛЬНОГО АНАЛІЗУ ДАНИХ ЗАСОБАМИ PYTHON

## 3.1 Опис методики візуального аналізу даних засобами Python

Методика візуального аналізу даних надає структурований та послідовний підхід до дослідження даних та виявлення прихованих закономірностей, залежностей і аномалій за допомогою графічного представлення інформації. Розроблена методика базується на використанні чотирьох бібліотек візуалізації: Pandas Visualization, Seaborn, Matplotlib і Plotly, кожна з яких відіграє свою роль на різних етапах аналізу.

Етап 1. Завантаження даних та швидкий первинний огляд.

Мета цього етапу – отримати перше загальне уявлення про структуру даних, типи змінних, наявність пропущених даних, загальний вигляд їх розподілів.

Основний інструмент першого етапу – бібліотека Pandas Visualization (.plot(), .hist(), .boxplot()), що дозволяє швидко генерувати базові графіки безпосередньо з об'єктів Series та DataFrame. Хоча детальна візуалізація пропусків даних вимагає інших інструментів, наприклад, Missingno, Pandas Visualization найкраще підходить для швидкої візуалізації розподілів числових змінних (df['column'].hist()) або побудови простих лінійних графіків для часових рядів (df['timeseries'].plot()).

Етап 2. Поглиблений одновимірний аналіз.

Мета етапу – детальне дослідження розподілу кожної окремої змінної (центральна тенденція, розкид, форма, модальність, викиди).

Основними інструментами другого етапу є дві бібліотеки:

- Seaborn (histplot, kdeplot, boxplot, violinplot, countplot) – основна бібліотека, що має значно більші можливості для візуалізації розподілів порівняно з Pandas. Дозволяє легко поєднувати гістограми з оцінками щільності розподілу

(KDE), створювати інформативні скрипкові діаграми, автоматично розраховувати та візуалізувати частоти категоріальних змінних;

- Matplotlib – використовується для додаткового налаштування графіків, створених Seaborn (наприклад, додавання специфічних заголовків, міток, анотацій), коли стандартних можливостей Seaborn недостатньо.

Етап 3. Двовимірний та багатовимірний аналіз даних.

Мета етапу – дослідити взаємозв'язки між парами змінних та між кількома змінними одночасно (дослідження кореляції, залежності, порівняння груп).

Інструменти третього етапу:

- Seaborn (scatterplot, lineplot, heatmap, pairplot, jointplot, catplot), як основний інструмент для дослідження взаємозв'язків. Графіки типу scatterplot та regplot дозволяють візуалізувати залежності між числовими змінними (з можливістю додавання ліній регресії), heatmap використовується для візуалізації кореляційних матриць, pairplot забезпечує комплексний огляд попарних взаємозв'язків у наборі даних, catplot є універсальним інструментом для порівняння числових змінних за категоріями;

- Pandas Visualization (.plot.scatter()) – використовується для швидкої початкової перевірки взаємозв'язків між даними;

- Plotly (px.scatter, px.scatter_matrix, px.imshow) – використовується, якщо при дослідженні взаємозв'язків потрібна інтерактивність (масштабування, панорамування, ідентифікація точок), особливо на великих наборах даних;

- Matplotlib – використовується для створення складних візуалізацій з кількома графіками (subplots) або для додавання специфічних елементів до графіків, створених за допомогою Seaborn/Plotly.

Етап 4. Цільова візуалізація.

Мета етапу – створення візуалізацій для перевірки гіпотез, що виникли на попередніх етапах, або для пошуку відповіді на конкретні дослідницькі питання.

Інструменти четвертого етапу:

- Seaborn / Matplotlib / Plotly (вибір залежить від конкретного завдання). Можлива їх комбінація для забезпечення певних аспектів: глибоке статистичне

дослідження (Seaborn), гнучкість налаштувань (Matplotlib), інтерактивність (Plotly). На цьому етапі може виникати потреба у додатковому налаштуванні візуальних елементів візуалізації (кольори, мітки, легенди, анотації), де основну роль відіграє Matplotlib, навіть якщо вихідний графік створено з Seaborn.

Етап 5. Підсумкове представлення результатів.

Мета етапу – створення підсумкових, узагальнюючих візуалізацій для включення їх у звіт, презентацію або інтерактивну інформаційну панель (дашборд).

Інструменти п'ятого етапу:

- Plotly – основний інструмент для створення інтерактивних візуалізацій, які можна легко експортувати як HTML файли або інтегрувати у вебдодатки (наприклад, за допомогою Dash);
- Matplotlib / Seaborn – використовуються для створення високоякісних статичних графіків для публікацій або звітів. Matplotlib надає максимальний контроль над фінальним виглядом візуалізації, використовується для остаточного доопрацювання графіків, створених за допомогою Seaborn.

Розроблена методика передбачає гнучкий вибір інструменту візуалізації на кожному кроці за наступними критеріями:

- швидкість або глибина аналізу (для швидкого огляду даних – Pandas; для глибокого статистичного аналізу – Seaborn);
- статика або інтерактивність (для статичних звітів/публікацій – Matplotlib/Seaborn; для інтерактивного дослідження або вебпублікацій – Plotly);
- стандартні або унікальні графіки (для стандартних візуалізацій – Seaborn; для унікальних графіків та тонкого налаштування – Matplotlib);
- тип даних (відповідно до рекомендацій, наведених у підрозділі 2.3. Наприклад, геодані – Folium/Plotly, хоча Folium виходить за рамки цих чотирьох бібліотек, але може доповнювати методику).

Загальний алгоритм застосування методики:

a) завантажити дані у Pandas DataFrame;

b) виконати швидку перевірку розподілів числових змінних, використовуючи метод df[num_cols].hist() з Pandas Visualisation;

c) виконати детальне вивчення розподілів та викидів, використовуючи метод sns.boxplot(data=df[num_cols]) з Seaborn;

d) виконати огляд попарних зв'язків, використовуючи sns.pairplot(df) з Seaborn;

e) виявити/дослідити значущі кореляції між парами змінних X і Y – Seaborn;

f) виконати інтерактивне дослідження цих зв'язків з можливістю фільтрації за категоріями, використовуючи px.scatter(df, x='X', y='Y', color='category') з Plotly;

g) створити фінальні графіки розсіювання з додаванням лінії регресії, специфічними мітками осей та заголовками: використання sns.regplot(), а потім функцій Matplotlib (plt.title, plt.xlabel, тощо) для доопрацювання.

У додатку В представлена діаграма діяльності, яка відображає етапи описаної методики візуального аналізу даних (рисунок В.2), а також діаграма послідовності, яка описує взаємодію аналітика з інструментами аналізу на кожному етапі візуального дослідження даних (рисунок В.3).

## 3.2 Практичне застосування розробленої методики

Для демонстрації застосування описаної вище методики візуального аналізу даних було обрано широко відомий набір даних «Титанік», що містить інформацію про пасажирів, включаючи демографічні дані (вік, стать), інформацію про подорож (порт посадки, клас каюти, вартість квитка, наявність родичів на борту) та змінну Survived (чи вижив пасажир).

Цей набір містить числові та категоріальні змінні, а також певну кількість пропущених значень, що дозволяє на цьому прикладі продемонструвати застосування різних технік візуального аналізу даних.

Завдання: виконати всі етапи розробленої методики, застосовуючи відповідні інструменти візуалізації для виявлення факторів, що впливали на виживання пасажирів.

Етап 1. Завантаження даних та швидкий первинний огляд.

Спочатку завантажуємо дані, оглядаємо їх базову структуру та візуально оцінюємо числові розподіли за допомогою Pandas (додаток Г, лістинг Г.1). Інформація про структуру набору даних «Титанік», типи даних та описові статистики числових даних представлена на рисунку 3.1.

```
Інформація про DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB

Описова статистика числових змінних:
         survived      pclass         age       sibsp       parch        fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```

Рисунок 3.1 – Базова інформація про структуру, типи даних та описова статистика числових змінних набору даних «Титанік» отримані засобами Pandas

Базова візуалізація розподілу числових даних з набору даних «Титанік» представлена на рисунку 3.2.

Рисунок 3.2 – Гістограми розподілу числових даних з набору даних «Титанік» за частотою, побудовані з використанням Pandas Visualization

Базові гістограми, представлені на рисунку 3.2., були отримані за допомогою pandas.DataFrame.hist(). Вони дозволяють встановити наступне:

- розподіл віку пасажирів (age) має певну схожість з нормальним розподілом, але з піком на молодших пасажирах та помітною кількістю немовлят/дітей. Видно, що є пропущені значення (кількість записів менша за загальну);
- розподіл вартості квитка (fare) сильно зміщений вліво, вказуючи на те, що більшість квитків були відносно дешевими, але була також невелика кількість дуже дорогих квитків;
- розподіли змінних sibsp та parch показують, що більшість пасажирів подорожували без братів/сестер/подружжя або батьків/дітей.

Етап 2. Поглиблений одновимірний візуальний аналіз (з використанням Seaborn).

Використовується бібліотека Seaborn (додаток Г, лістинг Г.2). Отримано деталізовані гістограми розподілу даних за віком та за вартістю квитка з кривими оцінки щільності розподілу (KDE), представлені на рисунках 3.3 і 3.4 відповідно.

Рисунок 3.3 – Деталізована гістограма розподілу змінної Age за частотою Count, отримана засобами бібліотеки Seaborn

Рисунок 3.4 – Деталізована гістограма розподілу змінної Fare за частотою Count, отримана засобами бібліотеки Seaborn

Графік Seaborn histplot з KDE на рисунку 3.3 підтверджує висновки першого етапу аналізу щодо розподілу віку, чіткіше показуючи піки та форму розподілу. Розподіл вартості квитка (fare) на рисунку 3.4. має довгий «хвіст» справа, що підтверджує наявність невеликої кількості дорогих квитків.

На рисунку 3.5 представлені графіки, що відображають кількість записів у наборі даних за категоріями survived (виживання), pclass (клас каюти) та sex (стать).



Рисунок 3.5 – Порівняння кількості значень по категоріям survived (виживання), pclass (клас каюти) та sex (стать), отримана засобами бібліотеки Seaborn

Аналіз графіків на рисунку 3.5 показує, що більшість пасажирів не вижили, найбільше пасажирів було у 3-му класі, найменше – у 2-му класі, а також, що чоловіків на борту було набагато більше, ніж жінок.

Етап 3. Двовимірний та багатовимірний аналіз (з використанням Seaborn, Plotly) – дослідження взаємозв'язів між змінними.

Код реалізації даного етапу візуального аналізу (додаток Г, лістинг Г.3) виводить одночасно п'ять дуже інформативних графіків: діаграму розсіювання, що описує зв'язок між вартістю квитка та виживанням (рисунок 3.6); теплову карту кореляції всіх числових змінних у наборі даних (рисунок 3.7); коробкову діаграму, що описує розподіл віку за класом каюти та статусом виживання (рисунок 3.8); стовпчикову діаграму, що описує частку тих, хто вижив за статтю (рисунок 3.9); підсумкову інтерактивну діаграму (рисунок 3.10).

Рисунок 3.6 – Аналіз взаємозв'язку між числовими змінними age (вік пасажира) і fare (вартість квитка) з розбивкою за ознакою survived (виживання)



Рисунок 3.7 – Теплова карта кореляції числових змінних з набору даних «Титанік»

Діаграма на рисунку 3.6 показує відсутність чіткої лінійної залежності між віком та вартістю квитка. Однак, тут легко побачити, що точки, які відповідають тим, хто вижив (помаранчеві, світлі), частіше зустрічаються серед пасажирів з вищою вартістю квитка (розташовані вище) та серед дітей (зліва на графіку).

Теплова карта на рисунку 3.7 показує слабкі лінійні зв'язки між більшістю числових змінних. Помітна негативна кореляція між класом каюти (pclass) та вартістю квитка (fare) з від'ємним коефіцієнтом парної лінійної кореляції, рівним –0,55 (нижчий клас – дешевший квиток), а також віком пасажира (age) з коефіцієнтом кореляції, рівним –0,37 (старший вік – вищий клас).

Коробкові діаграми на рисунку 3.8 чітко демонструють, що пасажири 1-го класу в середньому були старшими за віком, а пасажири 3-го класу – молодшими. Також добре видно, що у 1-му класі віковий розподіл тих, хто вижив був більш широким, а серед дітей 2-го та 3-го класів шанси вижити були вищими.



Рисунок 3.8 – Розподіл числової змінної age (дискретні числові дані) за категоріями pclass (категоріальні порядкові дані) та survived (категоріальні номінальні дані)

Наступна діаграма на рисунку 3.9 показує, що частка жінок, які вижили, значно більша, ніж частка чоловіків. Порівнюючи з рисунком 3.5, можна зробити чіткий висновок, що більшість загиблих – це чоловіки, пасажири 3 класу.



Рисунок 3.9 – Порівняння частки тих, хто вижив залежно від статі

Інтерактивні графіки дозволяють отримати більш детальну інформацію через спливаючі елементи, масштабування, фільтрування за легендою (рисунок 3.10).



Рисунок 3.10 – Аналіз взаємозв'язку між числовими змінними age (вік) і fare (вартість квитка) з розбивкою за ознакою survived (виживання)

Графік на рисунку 3.10 дозволяє глибше дослідити дані про окремі групи пасажирів. Зокрема, легко побачити, що великі бульбашки на діаграмі (3-й клас, дешевші квитки) переважно розташовані внизу – тобто відповідають загиблим.

Етап 4. Цільова візуалізація (з використанням Seaborn, Matplotlib).

Мета: визначити, як клас кають впливає на шанси виживання окремо для чоловіків та жінок. Код реалізації (додаток Г, лістинг Г.4) виводить графік візуального аналізу впливу класу кають та статі на виживання (рисунок 3.11).



Рисунок 3.11 – Аналіз впливу класу кають на статус виживання

Цей графік, демонструє, що жінки мали значно вищі шанси на виживання у всіх класах порівняно з чоловіками. Для обох статей спостерігається чітка тенденція до зниження шансів на виживання зі зниженням класу кають. Комбінація цих факторів показує, що жінки 1-го та 2-го класів мали найвищі шанси вижити, тоді як чоловіки 3-го класу – найнижчі.

Етап 5. Підсумкове (узагальнююче) представлення результатів.

Підсумковий інтерактивний графік (додаток Г, лістинг Г.5), представлений на рисунку 3.12, узагальнює головні фактори виживання для можливого включення в інтерактивний звіт або дашборд.

Рисунок 3.12 – Підсумковий інтерактивний графік демонструє комплексний візуальний аналіз факторів виживання

Інтерактивний графік на рисунку 3.12 дозволяє одночасно візуалізувати вплив всіх факторів (віку, вартості квитка, статі та класу) на виживання. Користувач може наводити курсор на точки, щоб побачити додаткову інформацію (embarked, alone), фільтрувати за статтю (колір) або класом (символ), масштабувати області, що викликають інтерес. Візуально підтверджуються попередні висновки: жінки (рожевий, світлий колір) мають більше великих маркерів (вижили), особливо ті, що відповідають 1-му та 2-му класам (ромби та чотирикутники). Пасажири з високою вартістю квитка (fare) також частіше мають великі маркери.

Таким чином, застосування розробленої методики на наборі даних «Титанік» продемонструвало її ефективність. Послідовне проходження етапів з використанням сильних сторін кожної з використаних бібліотек Python, дозволило систематично дослідити ці дані: від загального огляду розподілів до виявлення складних взаємозв'язків між факторами (клас, стать, вік, вартість квитка) та цільовою змінною (виживання). Методика забезпечила структурований підхід до генерації візуалізацій, що сприяло формулюванню та перевірці гіпотез, а також підготовці результатів для представлення у різних форматах.

## 3.3 Рекомендації щодо вибору інструментів та методів візуалізації

На основі розробленої методики візуального аналізу (п. 3.1) та її практичного застосування (п. 3.2), можна сформулювати рекомендації щодо вибору оптимальних методів візуалізації та відповідних інструментів Python (з фокусом на Matplotlib, Pandas Visualization, Seaborn та Plotly) залежно від аналітичного завдання, типу даних та цільової аудиторії.

Перед створенням візуалізації корисно дати відповідь на наступні запитання (обрати з даних варіантів), які допоможуть визначити найбільш доцільний підхід:

1. Яка мета візуалізації:

- початкове дослідження, щоб швидко зрозуміти структуру, розподіли, основні залежності;
- глибокий аналіз, щоб детально дослідити взаємозв'язки, порівняти групи, перевірити гіпотези, відповісти на конкретні питання;
- представлення результатів, щоб чітко й переконливо представити ключові висновки певній аудиторії;
- моніторинг, що відстежувати зміни показників у часі (часто в дашбордах).

2. Який тип даних візуалізується:

- розподіл однієї змінної (числова, категоріальна);
- взаємозв'язок між двома змінними (числова-числова, числова-категоріальна, категоріальна-категоріальна);
- порівняння значень між групами;
- композиція (частини цілого);
- ряди динаміки (часові ряди);
- геопросторові дані.

3. Хто є цільовою аудиторією:

- сам дослідник (швидкість і функціональність важливіші за естетику);
- технічні колеги (можуть зрозуміти складніші графіки);
- менеджмент / нетехнічні колеги (потрібна максимальна ясність, простота, можливо, інтерактивність).

4. Який рівень деталізації та налаштувань потрібен:

- достатньо стандартного вигляду;
- потрібно точно налаштувати кольори, шрифти, мітки, легенду, додати анотації.

5. Чи потрібна інтерактивність:

- чи допоможе масштабування, панорамування, фільтрація, відображення даних при наведенні краще зрозуміти дані або представити результати?

6. Який кінцевий формат візуалізації:

- графік у Jupyter Notebook;
- статичне зображення (PNG, JPG, PDF) для звіту, публікації;
- інтерактивний HTML файл;
- компонент вебдашборду.

Рекомендації щодо вибору бібліотек Python на основі мети візуалізації:

- для початкового дослідження даних – Pandas Visualization для максимально швидкого створення базових графіків (гістограм, лінійних, розсіювання) безпосередньо з DataFrame, доповнення простими графіками Seaborn (histplot, countplot) для глибшого огляду. Основні методи візуального представлення: гістограми, базові стовпчикові діаграми, прості лінійні графіки;
- для глибокого статистичного аналізу – Seaborn, як основний інструмент завдяки своєму високорівневому інтерфейсу та фокусу на статистичних зв'язках, Matplotlib – для доопрацювання та налаштування графіків Seaborn. Методи візуалізації: залежно від даних – scatterplot, regplot, boxplot, violinplot, heatmap, pairplot, jointplot, catplot та інші статистичні візуалізації Seaborn;
- для представлення результатів: статичні звіти/публікації – Seaborn для генерації основи графіка та Matplotlib для ретельного налаштування вигляду (заголовки, мітки, легенди, анотації, кольори); інтерактивні презентації/веб – Plotly (особливо Plotly Express) для створення естетично привабливих інтерактивних графіків, які легко вбудовуються та дозволяють аудиторії самостійно досліджувати дані. Методи візуального представлення: найбільш зрозумілі типи графіків (чим

простіші, ти краще) – стовпчикові діаграми, лінійні графіки, прості діаграми розсіювання;

- для побудови дашбордів – Plotly є найкращим вибором завдяки тісній інтеграції з фреймворком Dash. Бібліотека Bokeh є альтернативою для створення дашбордів (вона входить до основного списку цієї роботи). Методи візуалізації: залежно від показників, що відстежуються; часто використовуються лінійні графіки, стовпчикові діаграми, індикатори (KPI), інтерактивні таблиці та карти.

### 3.4 Оцінка економічної ефективності розробленої методики

Для оцінки економічної ефективності впровадження та використання розробленої методики візуального аналізу даних було виконано розрахунок економічних показників NPV (Net Present Value, чиста приведена вартість) та PP (Payback Period, термін окупності інвестицій) [60].

Показник NPV відображає загальну вигоду від інвестиційного проєкту, виражену в грошах, з урахуванням вартості грошей у часі. Він показує, скільки прибутку або збитку отримає компанія від проєкту після повернення вкладених коштів та дисконтування майбутніх доходів. Можливі випадки:

NPV > 0 – проєкт прибутковий і його варто реалізовувати;

NPV < 0 – проєкт збитковий, його фінансування недоцільне;

NPV = 0 – проєкт не приносить ні прибутку, ні збитків.

У розрахунках NPV враховується ставка дисконту, яка показує, як з часом зменшується цінність майбутніх доходів. Таким чином NPV дозволяє оцінити, чи поверне проєкт свої витрати і принесе додатковий дохід, з огляду на ризики та майбутню вартість грошей. Розрахунок NPV виконується за формулою:

$$NPV_T = \sum_{t=1}^{T} \frac{CF_t}{(1+r)^t} - IC \qquad (3.1)$$

де: $CF_t$ – грошовий потік (надходження, економія) у році $t$, грн;

$r$ – ставка дисконту, %;

$IC$ – інвестиційні витрати (вартість обладнання, витрати на навчання), грн;

$T$ – кількість періодів, рік.

Термін окупності визначається за формулою:

$$PP = \frac{IC}{E_p} \tag{3.2}$$

де: $E_p$ – річна економія (економія ліцензій, зменшення витрат на навчання та підтримку);

$IC$ – інвестиційні витрати.

Виконаємо оцінку економічної ефективності методики візуального аналізу даних засобами Python, використовуючи реальні ціни та актуальні дані за 2025 рік.

Вихідні дані для розрахунку представлені у таблиці 3.1.

Таблиця 3.1 – Основні статті витрат

| Стаття витрат | Сума, грн | Примітка |
|---|---|---|
| Вартість обладнання | 50000 | Середній ПК для аналітика даних |
| Вартість електроенергії на рік | 4200 | 350 кВт/рік × 12 грн/кВт*год |
| Вартість інтернету на рік | 3600 | 300 грн/міс × 12 |
| Зарплата аналітика на рік | 420000 | 35 000 грн/міс × 12 |
| Витрати на навчання | 10000 | Курс із Python (одноразово) |
| Вартість ліцензії Python | 0 | Відкритий код |
| Вартість ліцензії альтернативного інструменту Tableau Creator | 34440 | 70$/міс × 41 грн/$ × 12 |

Економія коштів формується за рахунок відсутності плати за ліцензію Python, меншої вартості навчання, більшої доступності фахівців. Витрати обмежуються обладнанням, навчанням, зарплатою, що є типовими для всіх варіантів.

Отже, грошовий потік включає економію коштів за рахунок ліцензійних витрат та додаткові надходження (відсутні), де річна економія коштів становить: $E_p = 34440 - 0 = 34440$ грн.

Розрахунок NPV на 5 років, виконаний за формулою (3.1), представлений у таблиці 3.2. У розрахунках використовується ставка дисконту 10%.

Таблиця 3.2 – Розрахунок чистої приведеної вартості (NPV) на 5 років

| Рік | Інвестиційні витрати, грн | Грошовий потік, грн | Дисконтний коефіцієнт | Дисконтований грошовий потік, грн | Чиста приведена вартість, грн |
|---|---|---|---|---|---|
| 1 | 60000 | 34 440 | 0,9091 | 31309 | -28691 |
| 2 | 0 | 34 440 | 0,8264 | 28468 | -228 |
| 3 | 0 | 34 440 | 0,7513 | 25878 | 25647 |
| 4 | 0 | 34 440 | 0,6830 | 23541 | 49170 |
| 5 | 0 | 34 440 | 0,6209 | 21389 | 70555 |

Згідно виконаних розрахунків, значення NPV є негативним у перший рік через стартові інвестиції, але загальний термін окупності становить менше 2 років, що є економічно доцільним.

Виконаємо розрахунок терміну окупності за формулою (3.2):

$$PP = \frac{50000 + 10000}{34440} \approx 1{,}74 \text{ (1 рік 9 місяців)}$$

Таким чином, запропонована методика візуального аналізу даних засобами Python виявилась економічно ефективною при середньостроковому використанні (понад 2 роки). Вона забезпечує економію за рахунок скорочення ліцензійних витрат на 34440 грн/рік і має короткий термін окупності (менше 2 років). Крім того, використання цієї методики створює додаткові нематеріальні вигоди – зростання компетентності команди аналітиків даних, незалежність від постачальників ПЗ, інтеграція з іншими Python-інструментами.

Рекомендується до впровадження в організаціях, що прагнуть скорочення витрат на програмне забезпечення та підвищення гнучкості ІТ-рішень.

# ВИСНОВКИ

Основні результати роботи:

1. Систематизовано теоретичні знання щодо принципів та етапів візуального аналізу даних;

2. Проведено детальний аналіз основних бібліотек Python для візуалізації даних (Matplotlib, Pandas Visualization, Seaborn, Plotly), виявлено їхні переваги, недоліки та оптимальне застосування;

3. Розроблено структуровану методику візуального аналізу даних засобами Python, яка поєднує теоретичні принципи з практичним використанням сильних сторін різних бібліотек Python в єдиному робочому процесі;

4. Створено готові шаблони коду на основі розробленої методики для пришвидшення типових завдань візуального аналізу даних;

5. Практично підтверджено ефективність та результативність запропонованої методики шляхом її апробації на реальних наборах даних;

6. Сформульовано конкретні рекомендації щодо вибору інструментів та типів візуалізації для вирішення типових завдань візуального аналізу даних засобами Python;

7. Наведено обґрунтування економічної доцільності впровадження та використання розробленої методики.

Перспективи подальших досліджень:

- розширення методики за рахунок інтеграції інших бібліотек Python у запропонований робочий процес;
- проведення більш формалізованого дослідження економічної ефективності методики шляхом порівняння часових витрат та якості отриманих знань при її застосуванні порівняно з альтернативними підходами в умовах конкретних бізнес-завдань;
- адаптація до предметних галузей – модифікація та деталізація методики для специфічних потреб аналізу даних у конкретних галузях.