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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«Методика статистичної обробки даних засобами Python»**

Виконала: здобувачка вищої освіти
за освітньою програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні системи та технології ступеня вищої освіти бакалавр
групи 126ІСТ_бз_2021
Колєснікова Ольга Юріївна
Керівник: Флегантов Леонід Олексійович
Рецензент: Брикун Олександр Миколайович

Полтава – 2025 року

ВСТУП

Актуальність теми. У сучасному світі статистична обробка даних є одним із головних напрямів розвитку інформаційних технологій, що забезпечує прийняття обґрунтованих рішень у різних сферах діяльності. Статистична обробка дозволяє проводити глибокий аналіз великих масивів інформації, виявляти закономірності та тренди, а також здійснювати прогнозування. З розвитком мов програмування та автоматизації процесів Python став одним із провідних інструментів для аналізу даних завдяки простоті, гнучкості та великій кількості спеціалізованих бібліотек.

Актуальність теми роботи зумовлена постійно зростаючою потребою у використанні ефективних методів статистичного аналізу в різних галузях науки, економіки та бізнесу. Python надає потужні можливості для обробки даних, починаючи від базових статистичних операцій і закінчуючи складними методами моделювання та машинного навчання. Проте, існує необхідність у систематизованому підході до вибору методів та інструментів Python, що дозволить покращити точність і швидкість статистичного аналізу.

Аналіз стану розробки проблєми. На сьогодні існує значна кількість досліджень, присвячених методам статистичної обробки даних. Вони охоплюють як класичні статистичні підходи, так і сучасні методи, що базуються на використанні алгоритмів машинного навчання та великих даних. Зокрема, широко досліджуються питання використання бібліотек NumPy, pandas, SciPy, statsmodels, scikit-learn для обробки статистичної інформації. Проте, відсутність узагальненої методики, що дозволяє ефективно інтегрувати ці бібліотеки для вирішення прикладних задач, потребує додаткових досліджень. Таким чином, тема розробки методики статистичної обробки даних засобами Python є актуальною, оскільки вона сприяє автоматизації процесів аналізу та забезпечує підвищення ефективності роботи з великими масивами даних.

Мета роботи: розробка та апробація методики статистичної обробки даних з використанням мови програмування Python.

Завдання роботи:

- дослідити теоретичні основи статистичного аналізу даних;
- розглянути існуючі бібліотеки Python для статистичної обробки даних;
- дослідити алгоритми статистичної обробки даних та їх реалізацію засобами Python;
- розробити методику статистичного аналізу на основі Python, яка включає автоматизовану обробку даних та їх візуалізацію;
- продемонструвати застосування розробленої методики на прикладах зі сфери ІТ;
- оцінити економічну ефективність запропонованої методики.

Об'єкт дослідження: процес статистичної обробки даних у цифрових системах.

Предмет дослідження: використання мови програмування Python для реалізації статистичних методів аналізу даних.

Методи наукових досліджень: аналіз наукових джерел – для вивчення існуючих методів статистичної обробки даних та огляду сучасних інструментів Python; контент-аналіз – для оцінки ефективності різних підходів до статистичного аналізу даних; програмування – для проведення статистичних розрахунків та оцінки точності аналізу; експериментальні дослідження – для тестування запропонованої методики; візуалізація результатів – для графічного представлення висновків аналізу.

Інформаційна база дослідження: наукові статті, підручники, офіційна документація бібліотек Python, відкриті набори даних з ресурсів Kaggle, UCI Machine Learning Repository та Google Dataset Search.

Апробація результатів дослідження. За результатами дослідження опубліковано тези доповіді: «Використання бібліотек Python для аналізу великих обсягів даних: практичний досвід», 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. Результати роботи можуть бути використані для оптимізації процесу аналізу даних у наукових дослідженнях, фінансовому секторі, медицині та інших галузях, що потребують ефективних методів статистичної обробки інформації. Запропонована методика дозволяє автоматизувати статистичний аналіз, зменшуючи витрати часу та підвищуючи точність результатів.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Основний текст викладений на 62 сторінках, містить 3 таблиці, 17 рисунків. Список використаних джерел складається з 51 найменування.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ СТАТИСТИЧНОЇ ОБРОБКИ ДАНИХ

1.1 Основні поняття статистичної обробки даних

Статистична обробка даних – це процес збирання, впорядкування, аналізу та інтерпретації числових даних за допомогою статистичних методів. Мета цього процесу – виявити закономірності, тенденції або відхилення в даних, що дозволяють зробити на цій основі обґрунтовані висновки, забезпечити прийняття обґрунтованих рішень на основі об'єктивного аналізу даних та кількісної оцінки невизначеності. Статистична обробка надає можливість приймати обґрунтовані рішення на основі емпіричних даних. Аналіз даних є головною частиною процесу статистичної обробки [1].

У контексті статистичної обробки даних засобами Python, під «даними» розуміють структуровану або неструктуровану інформацію, представлену у форматі, який може бути інтерпретований та опрацьований бібліотеками Python для статистичного аналізу. Дані можуть надходити з різних джерел і мати різні типи. Для застосування методів статистичної обробки та аналізу даних використовуються поняття змінної, основних типів змінних та шкал вимірювання. Змінною вважається будь-яка характеристика об'єкта, яку можна виміряти, та яка здатна набувати різних значень для різних об'єктів або в різні моменти часу для одного об'єкта. Різні типи змінних визначають, які статистичні методи можуть бути застосовані для їх аналізу.

Кількісні (числові) дані (Quantitative/Numerical data) представляються числами і відображають кількість або міру чогось. У Python кількісні дані найчастіше представляються типами int, float. Бібліотека NumPy використовує для зберігання та обробки числових масивів структуру ndarray, бібліотека pandas використовує для представлення кількісних даних послідовності series та таблиці DataFrame, де кожен стовпець може мати певний числовий тип даних, а саме [2]:

- дискретні дані (discrete data), які можуть приймати лише певні відокремлені значення (наприклад, кількість студентів у групі, кількість бракованих деталей). У Python можуть представляються як цілі числа (int);
- неперервні дані (continuous data), які можуть приймати будь-яке значення в заданому діапазоні (наприклад, зріст, вага, температура). У Python найчастіше бувають представлені числами з плаваючою комою (float).

Якісні (категоріальні) дані (Qualitative/Categorical data) описують деяку якість об'єкта або його належність до певної категорії. У Python представляються рядками (str) або числовими кодами. Бібліотека pandas використовує для представлення та обробки категоріальних змінних спеціальний тип даних categorical, зручний для економії пам'яті та певних статистичних операцій [3]:

- номінальні дані (nominal data) описують категорії без природного порядку (наприклад, колір, стать). У pandas можуть бути представлені як об'єкти (object dtype) або категоріальний тип без заданого порядку;
- порядкові дані (ordinal data) описують категорії з природним порядком (наприклад, рівень освіти, оцінка якості). У pandas можуть бути представлені як категоріальний тип із заданим порядком, що дозволяє виконувати операції порівняння.

Шкали вимірювання даних відображають рівень інформації, що міститься у значеннях змінних, та визначають допустимі математичні операції над цими значеннями. Основні типи шкал вимірювання змінних [4]:

- номінальна шкала (Nominal scale) визначає категорійні (якісні) змінні, значення яких є якісними відмінностями і не можуть бути природно впорядковані (наприклад, колір автомобіля (червоний, синій, зелений), стать (чоловіча, жіноча), назви міст (Київ, Харків, Львів, Полтава), тип продукту (одяг, взуття, аксесуари)). Значення змінних у номінальній шкалі є взаємовиключними категоріями. Єдиною допустимою математичною операцією над категорійними (якісними) змінними є визначення рівності або нерівності значень (наприклад, чи є колір автомобіля «червоним»?). До номінальних даних застосовуються також підрахунок частот, обчислення відсотків та використання статистичних критеріїв узгодженості

(наприклад, $\chi 2$). У Python номінальні дані можуть бути представлені в бібліотеці pandas як об'єкти (object dtype) або категоріальний тип без заданого порядку;

- порядкова шкала (ordinal scale) визначає категорійні (якісні) змінні, значення яких можуть бути впорядковані за певною ознакою, але відстань між цими значеннями не є визначеною або не має змістовного числового значення (наприклад, рівень освіти (початкова, середня, вища), оцінка якості обслуговування (дуже погано, погано, задовільно, добре, відмінно), стадія захворювання (1, 2, 3, 4)). У порядковій шкалі використовуються поняття «більше ніж» або «менше ніж», але не «на скільки більше». Допустимі математичні операції включають порівняння ($>$, $<$, $\geq$, $\leq$) та визначення медіани та квартилів. Для аналізу порядкових даних часто використовуються непараметричні статистичні методи (наприклад, критерій Манна-Уітні, критерій Краскела-Уолліса, коефіцієнт кореляції Спірмена). У Python порядкові дані можуть бути представлені як категоріальний тип із заданим порядком в бібліотеці pandas, що дозволяє виконувати операції порівняння;

- інтервальна шкала (interval scale) кількісна (дискретна, цілочислова) змінна, значення якої мають визначений порядок, а відстань між сусідніми значеннями є однаковою та має змістовне числове значення. Однак інтервальна шкала не має істинної нульової точки, тобто значення «0» не означає повної відсутності вимірюваної характеристики (наприклад, температура за шкалою Цельсія або Фаренгейта, рік народження, час доби). Оскільки нуль в інтервальній шкалі є умовним, для інтервальних даних не має сенсу відношення значень (наприклад, 20°C не є «вдвічі тепліше» за 10°C). Допустимі математичні операції включають додавання та віднімання, а також обчислення середнього значення та стандартного відхилення. Для аналізу інтервальних даних можуть застосовуватися параметричні статистичні методи (наприклад, t-тести, дисперсійний аналіз ANOVA), за умови дотримання відповідних припущень. У Python інтервальні дані зазвичай представлені числовими типами (int, float) у pandas та NumPy;

- шкала відношень (ratio scale) кількісна (неперервна, числова) змінна, яка має всі властивості інтервальної шкали, але має істинну нульову точку, що означає повну відсутність вимірюваної характеристики (наприклад, зріст, вага, вік, дохід,

кількість проданих товарів, час виконання завдання). Наявність істинного нуля дозволяє обчислювати не лише різниці, але й відношення значень (наприклад, людина зростом 180 см вдвічі вища за людину зростом 90 см). Допустимі всі арифметичні операції (додавання, віднімання, множення, ділення), а також обчислення всіх статистичних показників (середнє, медіана, мода, стандартне відхилення, коефіцієнт варіації тощо). Шкала відношень є найінформативнішою шкалою вимірювання, і для аналізу даних, виміряних у цій шкалі, можуть застосовуватися найширший спектр статистичних методів, включаючи параметричні. У Python дані шкали відношень також зазвичай представлені числовими типами (int, float) у pandas та NumPy.

Наступні категорії є основними поняттями математичної статистики [5].

Генеральна сукупність (population) – це усі можливі об'єкти або вимірювання, що становлять інтерес для дослідження. У Python може бути представлена як великий набір даних.

Вибірка (sample) – частина генеральної сукупності, відібрана для дослідження з метою отримання інформації про всю сукупність. У Python зазвичай представлена як частина більшого набору даних (наприклад, зріз масиву NumPy або підмножина DataFrame pandas). Бібліотека NumPy надає функції для випадкового вибору елементів (numpy.random.choice), а pandas має методи для випадкового відбору рядків (DataFrame.sample). Репрезентативна (представницька) вибірка (representative sample) – це вибірка, що відображає основні характеристики генеральної сукупності.

Розподіл даних (data distribution) описує частоту появи різних значень у даному наборі даних; характеризується формою розподілу (shape of distribution): симетричний, скошений (ліворуч або праворуч), унімодальний, бімодальний тощо. Бібліотеки Matplotlib та Seaborn використовуються для візуалізації розподілу даних (гістограми – plt.hist(), sns.histplot(); коробкові діаграми – sns.boxplot(); скрипкові діаграми – sns.violinplot()). Взагалі, Matplotlib та Seaborn є основними бібліотеками Python для створення різноманітних статистичних графіків. pandas також має вбудовані методи для базової візуалізації (.plot()).

Міри центральні тенденції (measures of central tendency) – статистичні характеристики набору даних, що характеризують центр розподілу даних: середнє значення (mean), медіана (median), мода (mode). Бібліотека pandas надає методи для обчислення середнього (.mean()), медіани (.median()), моди (.mode()) для стовпців DataFrame або серій. NumPy також має відповідні функції (numpy.mean, numpy.median).

Показники розсіяння даних (measures of dispersion) – значення, що характеризують ступінь розкиду даних навколо центральної тенденції: діапазон (розмах варіації) (range), дисперсія (variance), стандартне відхилення (standard deviation), міжквартильний розмах (interquartile range, IQR). pandas має методи для обчислення діапазону (через .max() та .min()), дисперсії (.var()), стандартного відхилення (.std()), міжквартильного розмаху (за допомогою .quantile()). NumPy також має відповідні функції.

Довірчий інтервал (confidence interval) – діапазон значень, в якому з певною (заданою) ймовірністю знаходиться істинний параметр генеральної сукупності. Бібліотеки Python, такі як SciPy та Statsmodels, мають функції для розрахунку довірчих інтервалів для різних параметрів.

Статистична гіпотеза (statistical hypothesis) – твердження про параметри генеральної сукупності, яке підлягає перевірці, використовує поняття нульової (основної) гіпотези (H0) та альтернативної гіпотези (H1 або Ha). Статистичний тест (statistical test) – процедура для перевірки статистичних гіпотез на основі вибіркових даних (наприклад, t-тест, $\chi 2$-тест, ANOVA). Модуль scipy.stats містить реалізації багатьох статистичних тестів (t-тести – scipy.stats.ttest_ind, $\chi 2$-тест – scipy.stats.chi2_contingency, ANOVA – scipy.stats.f_oneway та багато інших). Бібліотека Statsmodels також надає широкий набір статистичних моделей та тестів з детальними результатами. Рівень значущості (α) – ймовірність відхилити правильну нульову гіпотезу (статистична помилка першого роду). Результати статистичних тестів у Python зазвичай включають p-значення (p-value), яке використовується для прийняття рішень щодо істинності або хибності нульової гіпотези і характеризує значущість статистичних показників (satistical significance).

Кореляція (correlation) – міра лінійного статистичного зв'язку між двома змінними; описується коефіцієнтом парної лінійної кореляції r. Для обчислення коефіцієнтів кореляції між змінними використовуються метод .corr() у pandas та функція numpy.corrcoef() в NumPy. Бібліотека Seaborn надає інструменти для візуалізації кореляційних матриць (sns.heatmap()).

Регресія (regression) – статистичний метод для моделювання взаємозв'язку між однією залежною та однією або кількома незалежними змінними з метою прогнозування. Основними інструментами для побудови та аналізу регресійних моделей у Python є ібліотеки Statsmodels та Scikit-learn (лінійна регресія – statsmodels.api.OLS, sklearn.linear_model.LinearRegression; логістична регресія – sklearn.linear_model.LogisticRegression, statsmodels.api.Logit тощо). Залежна змінна (dependent variable) – це змінна, яку дослідник намагається пояснити або передбачити. Незалежна змінна (independent variable) – змінна, яка, як вважається, впливає на залежну змінну. При моделюванні в Python (наприклад, за допомогою Scikit-learn або Statsmodels) чітко визначаються залежні (цільові) та незалежні (предикторні) змінні.

Розуміння цих основних понять є необхідною умовою для статистичної обробки даних засобами Python або будь-яким іншим інструментом.

1.2 Етапи статистичного аналізу даних

Статистика, як наукова дисципліна, розділяється на два основні напрямки [6]:

- описова статистика (descriptive statistics) – займається систематизацією, узагальненням та представленням даних у зручній для розуміння формі. Включає розрахунок центральних тенденцій (середнє значення, медіана, мода), показників розсіяння (стандартне відхилення, дисперсія, діапазон, міжквартильний розмах), а також візуалізацію даних за допомогою графіків, діаграм та таблиць. У Python для цього використовуються бібліотеки pandas (методи .mean(), .median(), .mode(), .std(), .var(), .quantile()), NumPy (функції numpy.mean, numpy.median, numpy.std,

numpy.var) та Matplotlib і Seaborn для візуалізації (гістограми, коробкові діаграми, діаграми розсіювання тощо);

- вивідна статистика (inferential statistics) – дозволяє робити обґрунтовані припущення (або висновки) щодо характеристик генеральної сукупності на основі аналізу обмеженої вибірки даних. Сюди входять різні методи тестування статистичних гіпотез, оцінка параметрів популяції (точкова та інтервальна), визначення ймовірностей подій, побудова довірчих інтервалів та розрахунок p-значень. У Python інструменти для вивідної статистики надає бібліотека SciPy (модуль scipy.stats для різних тестів та розподілів) та Statsmodels (для побудови статистичних моделей та отримання детальних результатів).

Вибір відповідного статистичного методу залежить від типу змінних, розміру вибірки, припущення щодо розподілу даних у генеральній сукупності, а також конкретних дослідницьких питанньь, на які необхідно отримати відповідь.

Процес статистичного аналізу даних включає чотири послідовні етапи [7].

1. Планування дослідження (експерименту) та збір даних. Результатом етапу має бути набір даних, які характеризують досліджуваний процес або явище. Збір даних здійснюється різними методами, як метод суцільних спостережень (коли досліджуються всі елементи генеральної сукупності) або вибірковий метод (коли досліджується лише частина елементів). Статистичні методи на етапі планування можуть допомогти визначити оптимальний дизайн дослідження, типи необхідних даних, розмір вибірки та методи збору даних для забезпечення статистичної потужності та мінімізації систематичних помилок. Для того щоб висновки були достовірними, вибірка повинна бути представницькою (репрезентативною). У Python для роботи з вибірками використовуються можливості бібліотек NumPy (numpy.random.choice) та pandas (DataFrame.sample).

2. Попередня обробка та дослідження даних (Exploratory Data Analysis – EDA). Цей етап дозволяє виявити можливі проблєми з даними та отримати первинне розуміння їх структури. Він включає приведення даних до належного формату, очищення від помилок та невідповідностей. На цьому етапі виконується попереднє візуальне дослідження даних за допомогою різноманітних графіків

(гістограм, діаграм розсіювання, коробкових діаграм тощо) для виявлення трендів, сезонних ефектів, екстремальних значень (викидів), попередньої оцінки типу розподілу даних, розрахунок описових статистик. Здійснюється очищення та підготовка даних (data cleaning and preparation), що може включати обробку відсутніх значень (видалення, заповнення), кодування категоріальних змінних (наприклад, за допомогою one-hot encoding або label encoding), стандартизацію або нормалізацію числових змінних. До цього ж етапу відноситься вибір стратегій для заповнення пропущених даних (наприклад, за допомогою середнього, медіани, моди або більш складних методів моделювання) та застосування технік нормування (наприклад, Min-Max Scaling, Z-score Standardization) для приведення числових змінних до спільного масштабу. Інструменти для виконання цих операцій у Python надають бібліотеки pandas та NumPy. Для візуалізації використовуються Matplotlib та Seaborn.

3. Оцінювання параметрів статистичних і математичних моделей. На цьому етапі, на основі підготовлених даних, будуються та оцінюються відповідні статистичні та математичні моделі, які описують взаємозв'язки між змінними або виявляють закономірності в даних. Результатом цього етапу є отримання математичних і статистичних моделей, якість яких підтверджується відповідними статистичними показниками (наприклад, коефіцієнт детермінації R^2 для регресійних моделей, показники значущості для коефіцієнтів моделі тощо). У Python для цього використовуються бібліотеки Statsmodels та Scikit-learn.

4. Перевірка раніше сформульованих гіпотез, прийняття рішень на основі результатів дослідження. На цьому етапі статистичні методи використовуються для аналізу отриманих результатів моделювання та перевірки висунутих на початку дослідження статистичних гіпотез. Це дозволяє дійти обґрунтованих висновків щодо досліджуваних явищ, а також оцінити силу цих висновків та рівень їхньої невизначеності. Для перевірки гіпотез у Python використовуються функції з бібліотеки SciPy (scipy.stats) та можливості бібліотеки Statsmodels.

1.3 Основні методи статистичного аналізу

У статистичному аналізі даних застосовується широкий спектр статистичних показників та методів, вибір яких залежить від типу даних та цілей аналізу [8].

Описова статистика включає розрахунок та інтерпретацію основних статистичних характеристик набору даних, таких як:

- середнє значення (mean) – сума всіх значень, поділена на їх кількість. У Python обчислюється за допомогою методів .mean() у pandas та функції numpy.mean() у NumPy;
- медіана (median) – центральне значення у впорядкованому наборі даних. У Python обчислюється за допомогою методів .median() у pandas та функції numpy.median() у NumPy;
- мода (mode) – значення, яке найчастіше зустрічається в наборі даних. У Python обчислюється за допомогою методу .mode() у pandas;
- стандартне відхилення (standard deviation) – міра розсіювання значень навколо середнього (розрізняють вибіркове та генеральне стандартне відхилення). У Python обчислюється за допомогою методів .std() у pandas (вибіркове) та функції numpy.std() у NumPy (генеральне);
- дисперсія (variance) – квадрат стандартного відхилення. Обчислюється за допомогою методів .var() у pandas та функції numpy.var() у NumPy;
- діапазон (розмах) варіації (range) – різниця між максимальним та мінімальним значеннями. У Python може бути обчислено за допомогою методів .max() та .min() у pandas або NumPy;
- квартилі (quartiles) – значення, які ділять впорядкований набір даних на чотири рівні частини (Q1, Q2 (медіана), Q3). У Python обчислюються за допомогою методу .quantile() у pandas та функції numpy.percentile() у NumPy з відповідними відсотками (25%, 50%, 75%);
- процентилі (percentiles) – значення, які ділять впорядкований набір даних на сто рівних частин. У Python обчислюються за допомогою методу .quantile() у pandas та функції numpy.percentile() у NumPy.

Перевірка гіпотез (hypothesis testing) є головним інструментом для формального підтвердження або відхилення статистичних гіпотез про параметри генеральної сукупності на основі вибіркових даних. Процес перевірки гіпотез включає:

- формулювання нульової гіпотези (null hypothesis, H0) – твердження про відсутність ефекту або зв'язку в генеральній сукупності;
- формулювання альтернативної гіпотези (alternative hypothesis, H1 або Ha) – твердження, яке дослідник намагається довести;
- вибір рівня значущості (significance level, α) – ймовірність відхилити нульову гіпотезу, коли вона насправді є істинною (помилка першого роду), зазвичай встановлюється на рівні α=0,05;
- аналіз вибіркових даних за допомогою відповідних статистичних тестів (вибір тесту залежить від типу даних та дослідницького питання). У Python різні статистичні тести доступні у модулі scipy.stats;
- розрахунок p-значення (p-value) – ймовірність отримання спостережуваних або більш екстремальних результатів, якщо нульова гіпотеза є істинною. Значення p-value надаються як результат багатьох статистичних тестів у SciPy та Statsmodels;
- прийняття рішення щодо нульової гіпотези на основі порівняння p-значення з рівнем значущості. Якщо p-значення менше або дорівнює α, нульова гіпотеза відхиляється на користь альтернативної;
- інтерпретація результатів у контексті дослідження. Важливо розрізняти статистичну значущість (наявність статистично підтвердженого ефекту) та практичну значущість (величина ефекту є достатньо важливою для практичного застосування).

Залежно від типу даних та дослідницького питання існують різні статистичні тести для перевірки гіпотез [9], наприклад: для аналізу категоріальних даних може бути використаний критерій узгодженості χ2 (scipy.stats.chi2_contingency), для порівняння середніх – t-статистика Стьюдента (Student's t-statistic) (scipy.stats.ttest_ind, scipy.stats.ttest_rel, scipy.stats.ttest_1samp), для аналізу

дисперсій – F-статистика Фішера (Fisher's F-statistic) (scipy.stats.f_oneway), для перевірки гіпотез про розподіл – критерії Шапіро-Уілка, Колмогорова-Смірнова, Крамера-Мізеса, Андерсона-Дарлінга (scipy.stats.shapiro, scipy.stats.kstest, scipy.stats.cramervonmises, scipy.stats.anderson).

Також використовують байєсівську перевірку гіпотез (bayesian hypothesis testing) – метод, що базується на теоремі Баєса та дозволяє оцінювати ймовірність гіпотез на основі апріорних знань та спостережуваних даних [10]. У Python для байєсівського аналізу використовуються бібліотеки, такі як PyMC3 та ArviZ.

T-тест (t-test) – параметричний статистичний тест, який використовується для порівняння середніх значень двох незалежних або залежних груп. Існують різні види t-тестів (наприклад, незалежних вибірок, парний t-тест). T-тест базується на t-розподілі Стьюдента і вимагає певних припущень щодо розподілу даних (наприклад, наближена нормальність). У Python t-тести реалізовані у модулі scipy.stats бібліотеки SciPy (scipy.stats.ttest_ind для незалежних вибірок, scipy.stats.ttest_rel для залежних вибірок, scipy.stats.ttest_1samp для порівняння середнього вибірки з відомим середнім популяції).

Дисперсійний аналіз (ANOVA – Analysis of Variance) – параметричний статистичний метод, який застосовується для порівняння середніх значень трьох і більше незалежних груп. Він є розширенням t-тесту на випадок кількох груп. Тест ANOVA аналізує дисперсію між групами порівняно з дисперсією всередині груп для визначення статистично значущих відмінностей між середніми. У Python ANOVA можна виконати за допомогою функції scipy.stats.f_oneway або використовуючи моделі в бібліотеці Statsmodels (statsmodels.formula.api.ols з подальшим застосуванням statsmodels.stats.anova.anova_lm).

Кореляційний аналіз (correlation analysis) – використовується для визначення наявності, напрямку та сили лінійного зв'язку між двома кількісними змінними. Найпоширенішою мірою лінійної кореляції є вибірковий коефіцієнт кореляції Пірсона (Pearson correlation coefficient) r, який може змінюватись у діапазоні від -1 (сильний негативний лінійний зв'язок) до +1 (сильний позитивний лінійний зв'язок), де 0 вказує на відсутність лінійного зв'язку. Коефіцієнт r відображає лише

лінійну залежність, і його близькість до нуля не означає відсутності будь-якого зв'язку між змінними (може існувати нелінійна залежність). Більш глибокий аналіз може потребувати використання методів виявлення нелінійних залежностей. Існують також інші коефіцієнти кореляції для різних типів даних (наприклад, коефіцієнт Спірмена для порядкових даних). У Python кореляційний аналіз можна виконати за допомогою методу .corr() для DataFrame у pandas та функції numpy.corrcoef() у NumPy. Бібліотека Seaborn (sns.heatmap()) надає засоби для візуалізації кореляційних матриць.

Регресійний аналіз (regression analysis) – застосовується для вивчення взаємозв'язків між однією залежною змінною (відгуком) та однією або декількома незалежними змінними (предикторами). Він дозволяє оцінити силу впливу незалежних змінних на залежну та побудувати математичну модель (регресійне рівняння) для прогнозування значення залежної змінної на основі значень незалежних змінних. Результатом регресійного аналізу є побудова рівняння (функції залежності), що описує цей зв'язок. Лінійна регресія (linear regression) є одним із найчастіше використовуваних видів регресійного аналізу, що передбачає лінійну залежність між змінними. Регресія виконує дві основні функції: передбачення (prediction) майбутніх значень залежної змінної та пояснення (explanation) того, як зміни в незалежних змінних впливають на залежну. У регресійних моделях також часто використовуються індикаторні (dummy) змінні для представлення категоріальних предикторів. У Python регресійний аналіз реалізується за допомогою бібліотек Statsmodels (statsmodels.api.OLS для звичайної лінійної регресії) та Scikit-learn (sklearn.linear_model.LinearRegression). Логістична регресія (logistic regression) – використовується для аналізу даних, де залежна змінна є бінарною (категоріальною з двома рівнями, наприклад, успіх/невдача, так/ні). Логістична регресія моделює ймовірність настання певної події як функцію від однієї або кількох незалежних змінних. У Python логістична регресія реалізована через sklearn.linear_model.LogisticRegression у бібліотеці Scikit-learn та statsmodels.api.Logit у Statsmodels.

Візуалізація даних (data visualization) є не тільки важливим елементом для представлення результатів аналізу, але й основним етапом попереднього дослідження, що дозволяє швидко знаходити особливості, тренди, викиди та потенційні зв'язки між змінними.

Для візуалізації різних типів змінних застосовують різноманітні методи [11]. Наприклад, для неперервних числових змінних ефективними є діаграми розсіювання (scatter plots) з можливим накладанням підігнаних ліній регресії (regplot у бібліотеці Seaborn – sns.regplot()), гістограми (histograms) для відображення розподілу частот (plt.hist() у Matplotlib, sns.histplot() у Seaborn), графіки ймовірностей (probability plots, PP та QQ графіки) для оцінки відповідності емпіричного розподілу теоретичному (scipy.stats.probplot()), «ящик з вусами» (box-and-whiskers plot) для відображення центральної тенденції, розсіяння та наявності викидів (sns.boxplot() у Seaborn, plt.boxplot() у Matplotlib). Для категоріальних змінних використовуються стовпчасті діаграми (bar charts) (plt.bar(), sns.barplot()) та кругові діаграми (pie charts) (plt.pie()). Для візуалізації багатовимірних даних застосовуються більш складні методи, такі як теплові карти (heatmaps) (sns.heatmap()), паралельні координати (parallel coordinates) (реалізація є в деяких спеціалізованих бібліотеках або може бути створена з використанням Matplotlib та pandas) та діаграми розсіювання матриці (scatter plot matrices) (sns.pairplot()).

1.4 Програмне забезпечення для статистичного аналізу. Роль Python

Вибір програмного забезпечення для статистичного аналізу визначає доступність методів, зручність роботи з даними та можливості візуалізації. Наразі існує велика кількість потужних інструментів для статистичного аналізу, серед яких комерційні пакети SPSS, SAS, Statistica та безкоштовне програмне забезпечення з відкритим вихідним кодом R і Python. Для спеціалізованих завдань можуть використовуватися математичні програмні комплекси Matlab, MathCad, Mathematica [12].

Python – це одна найбільш популярних та універсальних сучасних мов програмування, що також є потужним інструментом для статистичного аналізу завдяки своїй розвиненій та постійно зростаючій екосистемі спеціалізованих бібліотек. До переліку найвідоміших та найбільш використовуваних бібліотек для аналітики даних та статистичного аналізу в Python належать такі бібліотеки [13]:

- pandas – використовує потужні та гнучкі структури даних для ефективної обробки та аналізу структурованих даних, зокрема табличних даних (структура DataFrame), а також має інструменти для читання та запису даних з різних форматів (CSV, Excel, SQL тощо), очищення, перетворення та агрегації даних;
- NumPy – фундаментальна бібліотека для наукових обчислень у Python, що забезпечує підтримку багатовимірних масивів та матриць, а також велику кількість математичних функцій для виконання числових операцій. NumPy є основою для багатьох інших бібліотек для аналізу даних;
- Matplotlib – базова бібліотека для створення статичних, інтерактивних та анімованих візуалізацій у Python. Надає широкий спектр графічних інструментів для побудови різноманітних діаграм, графіків та гістограм;
- seaborn – побудована на основі Matplotlib і надає високорівневий інтерфейс для створення візуально привабливих та інформативних статистичних графіків. Особливо корисна для візуалізації статистичних взаємозв'язків між змінними, таких як кореляційні матриці та розподіли. Часто використовується для візуалізації окремих моделей функцій, наприклад, ліній регресії;
- scikit-learn – одна з головних бібліотек для машинного навчання та статистичного моделювання в Python. Містить реалізації багатьох класичних алгоритмів машинного навчання (класифікація, регресія, кластеризація), а також інструменти для попередньої обробки даних, вибору моделей та оцінки їх якості. Має реалізації багатьох статистичних методів, зокрема таких, як лінійна та логістична регресія;
- SciPy – розширює функціональність NumPy, надаючи додаткові модулі для наукових та технічних обчислень, включаючи статистичні функції (тестування

гіпотез, розподіли ймовірностей), оптимізацію, інтегрування, інтерполяцію, обробку сигналів та багато іншого;

- statsmodels – спеціалізується на наданні інструментів для статистичного моделювання, включаючи лінійні та узагальнені лінійні моделі, моделі часових рядів, непараметричні методи та багато інших статистичних тестів та моделей. Statsmodels надає детальну статистичну інформацію про побудовані моделі (p-значення, довірчі інтервали тощо);
- TensorFlow та PyTorch – бібліотеки для глибокого навчання та побудови нейронних мереж, які також можуть бути застосовані для складного статистичного моделювання та аналізу великих обсягів даних;
- PyMC3 та ArviZ – бібліотеки, що використовуються для байєсівського статистичного моделювання та аналізу в Python.

Таким чином, було розглянуто теоретичну базу статистичної обробки даних:

- визначено поняття даних, як структурованої інформації та розкрито сутність змінних (кількісних та якісних), а також шкал вимірювання змінних (номінальна, порядкова, інтервальна, відношень), які визначають вибір методів аналізу. Розглянуто поняття генеральної сукупності, вибірки, розподілу даних, мір центральної тенденції та показників розсіяння;
- описано чотири послідовні етапи статистичної обробки даних: планування дослідження та збір даних, попередня обробка та дослідження даних (EDA), оцінювання параметрів моделей та перевірка гіпотез;
- розглянуто основні методи статистичного аналізу: описову статистику, перевірку статистичних гіпотез та статистичні тести, кореляційний та регресійний аналіз, візуалізацію даних як складову частину аналізу даних;
- висвітлено роль Python та його спеціалізованих бібліотек (pandas, NumPy, Matplotlib, seaborn, SciPy, statsmodels) як інструментів для аналізу даних.

РОЗДІЛ 2
ІНСТРУМЕНТИ PYTHON ДЛЯ СТАТИСТИЧНОГО АНАЛІЗУ ДАНИХ

2.1 Формати даних які підтримує Python

Python є гнучкою мовою програмування і підтримує роботу з великою кількістю форматів даних завдяки вбудованим модулям та численним стороннім бібліотекам. Представлений нижче список не є вичерпним, але охоплює більшість форматів даних, які підтримує Python [14].

Текстові формати:

- простий текст (.txt) – зчитується за допомогою вбудованої функції open();
- CSV (Comma Separated Values, .csv) – дуже поширений формат для представлення табличних даних, обробляється за допомогою вбудованого модуля csv або бібліотек, таких як pandas;
- JSON (JavaScript Object Notation, .json) – популярний формат для обміну даними у веброзробці, зчитується за допомогою вбудованого модуля json;
- XML (Extensible Markup Language, .xml) – структурований текстовий формат, бробляється за допомогою вбудованого модуля xml.etree.ElementTree або сторонніх бібліотек, як lxml;
- YAML (.yaml, .yml) – формат серіалізації даних, який вважається більш читабельним для людини, ніж JSON, потребує бібліотеки PyYAML;
- HTML (.html) – мова розмітки для вебсторінок, Python може зчитувати та парсити HTML за допомогою бібліотек BeautifulSoup, lxml, html.parser.

Бінарні формати:

- бінарні файли загального призначення зчитуються у Python за допомогою вбудованої функції open() у бінарному режимі (‘rb’);
- pickle (.pkl) – формат Python для серіалізації та десеріалізації об’єктів Python, використовується вбудований модуль pickle;
- формати електронних таблиць Excel (.xls, .xlsx) – потребують бібліотек, таких як pandas, openpyxl (для .xlsx) або xlrd (для старіших .xls);

- зображення (.jpg, .png, .gif, .bmp, .tiff тощо) обробляються за допомогою бібліотек, таких як Pillow (форк PIL) або opencv-python;

- аудіо файли (.wav, .mp3 тощо) – використовуються бібліотеки wave (вбудований для WAV), pydub, librosa;

- відео файли – бібліотека opencv-python (cv2) дозволяє працювати з відео;

- архіви (.zip, .tar, .gz, .bz2) обробляються за допомогою вбудованих модулів zipfile, tarfile, gzip, bz2.

Формати баз даних:

- реляційні бази даних (SQLite, PostgreSQL, MySQL, Oracle, SQL Server) – Python має драйвери та бібліотеки для підключення та зчитування даних з різних SQL баз даних (sqlite3, psycopg2 для PostgreSQL, mysql-connector-python для MySQL, cx_Oracle для Oracle, pyodbc або pymssql для SQL Server);

- NoSQL бази даних (MongoDB, Redis, Cassandra) – існують спеціалізовані бібліотеки для роботи з різними NoSQL базами (pymongo для MongoDB, redis-py для Redis).

Наукові та аналітичні формати:

- HDF5 (.h5, .hdf5) – формат для зберігання великих обсягів наукових даних, використовуються бібліотеки h5py, tables (PyTables) разом з pandas або xarray;

- NetCDF (.nc) – формат, поширений у кліматології, метеорології та океанографії, обробляється бібліотеками netCDF4, xarray;

- parquet (.parquet) – колонковий формат зберігання даних, популярний для Big Data, зчитується за допомогою pyarrow або fastparquet, інтегрований з pandas;

- feather (.feather) – швидкий бінарний формат для DataFrames, розроблений для взаємодії між Python (Pandas) та R, використовується бібліотекою pyarrow.

Для статистичної обробки даних у Python найчастіше використовуються такі формати [15]:

- CSV (.csv) – найпоширеніший формат для обміну табличними даними, який легко створити та прочитати. Основним інструментом для зчитування CSV-файлів у DataFrame для подальшого аналізу є бібліотека pandas (read_csv). Також є вбудований модуль csv, але pandas надає більше можливостей для аналізу;

- формат Excel (.xls, .xlsx) – популярний у бізнес-середовищі та багатьох дослідницьких сферах. Бібліотека pandas (read_excel) дозволяє зчитувати дані з одного або кількох аркушів файлу Excel. Для цього вона використовує інші бібліотеки, такі як openpyxl або xlrd;

- формати баз даних SQL (SQLite, PostgreSQL, MySQL тощо). Python дозволяє підключатися до баз даних через бібліотеки типу sqlite3, psycopg2, mysql-connector-python, виконувати SQL-запити (read_sql_query, read_sql_table) та безпосередньо завантажувати результати цих запитів у DataFrame;

- JSON (.json) часто використовується для передачі даних через API або для зберігання напівструктурованих даних. Засоби pandas (read_json) дозволяють інтерпретувати певні структури JSON як таблиці. Для загальної роботи з JSON використовується вбудований модуль json;

- Parquet (.parquet) – колонковий формат, популярний для великих наборів даних завдяки ефективності зберігання та швидкості читання/запису. pandas інтегрується з бібліотеками pyarrow або fastparquet для читання та запису цього формату (read_parquet). Рекомендований для роботи з великими обсягами даних;

- формати статистичних пакетів (Stata .dta, SPSS .sav, SAS .sas7bdat) – pandas має функції для їх прямого зчитування (read_stata, read_spss, read_sas);

- HDF5 (.h5, .hdf5) – використовується в наукових обчисленнях для зберігання великих та/або складних наборів даних. pandas може читати та записувати дані в HDF5 (read_hdf), використовуючи бібліотеки h5py або tables.

Таким чином, для більшості завдань статистичної обробки даних у Python центральну роль відіграє бібліотека pandas, яка надає зручні функції read_* для зчитування даних з цих основних форматів у структуру DataFrame, що є стандартом для аналізу даних у Python.

Найчастіше використовуються формати даних CSV, Excel, SQL та Parquet для великих даних.

Основні відомості про формати даних, що використовуються у Python з поясненнями та типами інструментів представлені у таблиці 2.1.

Таблиця 2.1 – Формати даних у Python з поясненнями та типами інструментів

Формат	Інструменти Python	Тип інструменту	Пояснення
Текстові формати			
Простий текст (.txt)	open()	Вбудована функція	Прості текстові файли для зберігання неструктурованих даних.
CSV (.csv)	csv, pandas	Стандартна + стороння бібліотека	Формат таблиць, розділених комами, зручний для імпорту/експорту.
JSON (.json)	json	Стандартна бібліотека	Формат обміну структурованими даними, особливо у веб.
XML (.xml)	xml.etree.ElementTree, lxml	Стандартна + стороння бібліотека	Формат для представлення структурованих ієрархічних даних.
YAML (.yaml, .yml)	PyYAML	Стороння бібліотека	Людиноорієнтований формат серіалізації даних.
HTML (.html)	BeautifulSoup, lxml, html.parser	Стороння + стандартна бібліотеки	Мова розмітки для створення вебсторінок.
Бінарні формати			
Бінарні файли	open('file', 'rb')	Вбудована функція	Універсальний підхід для читання будь-яких бінарних файлів.
Pickle (.pkl)	pickle	Стандартна бібліотека	Серіалізація об'єктів Python у файл і навпаки.
Excel (.xls, .xlsx)	pandas, openpyxl, xlrd	Сторонні бібліотеки	Файли електронних таблиць для обробки табличних даних.
Зображення (.jpg, .png тощо)	Pillow, opencv-python	Сторонні бібліотеки	Графічні файли, що використовуються для зображень.
Аудіо (.wav, .mp3 тощо)	wave, pydub, librosa	Стандартна + сторонні бібліотеки	Звукові файли у різних форматах для обробки аудіо.
Відео файли	opencv-python (cv2)	Стороння бібліотека	Файли відео, які можна читати та обробляти кадр за кадром.
Стиснуті архіви (.zip, .tar, .gz, .bz2)	zipfile, tarfile, gzip, bz2	Стандартна бібліотека	Формати стискання та архівації даних.
Формати баз даних			
Реляційні бази даних	sqlite3, psycopg2, mysql-connector-python, SQLAlchemy	Стандартна + сторонні бібліотеки	Бази даних зі структурованими таблицями та запитами SQL.
NoSQL бази даних	pymongo, redis-py	Сторонні бібліотеки	Бази з гнучкими структурами зберігання, без жорстких схем.

Продовження таблиці 2.1

Наукові та аналітичні формати			
HDF5 (.h5, .hdf5)	h5py, PyTables, pandas, xarray	Сторонні бібліотеки	Формат для зберігання великих обсягів наукових даних.
NetCDF (.nc)	netCDF4, xarray	Сторонні бібліотеки	Науковий формат даних для кліматології, геонаук тощо.
Parquet (.parquet)	pyarrow, fastparquet, pandas	Сторонні бібліотеки	Колонковий формат для ефективного зберігання та читання.
Feather (.feather)	pyarrow	Стороння бібліотека	Формат обміну табличними даними між Python і R.

Приклад коду Python, який використовує бібліотеку pandas для зчитування файлу data.txt наведено у додатку Б, лістинг Б.1. Результат на рисунку 2.1.

```
Файл '01_data.txt' успішно зчитано.

Перші 5 рядків даних:
   ID  Age  Gender  Score       Group     City
0   1   45  Female  78.54     Control     Lviv
1   2   23    Male  55.12  TreatmentA     Kyiv
2   3   61    Male  92.33     Control    Odesa
3   4   33  Female  68.91  TreatmentB  Kharkiv
4   5   52   Other  44.78  TreatmentA  Poltava

Інформація про DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      100 non-null    int64
 1   Age     100 non-null    int64
 2   Gender  100 non-null    object
 3   Score   100 non-null    float64
 4   Group   100 non-null    object
 5   City    100 non-null    object
dtypes: float64(1), int64(2), object(3)
memory usage: 4.8+ KB

Описова статистика:
                ID         Age       Score
count   100.000000  100.000000  100.000000
mean     50.500000   44.230000   74.470900
std      29.011492   14.899566   14.701673
min       1.000000   18.000000   44.780000
25%      25.750000   31.750000   62.027500
50%      50.500000   44.000000   75.005000
75%      75.250000   57.000000   86.975000
max     100.000000   70.000000   99.910000
```

Рисунок 2.1 – Результат завантаження даних .txt за допомогою pandas

Попередній код можна розширити, додавши етапи перевірки на поширені проблєми в даних та базові кроки їх очищення, наприклад, деякі перевірки та очищення. Зокрема, можна:

- знайти та видалити повні дублікати рядків або дублікати за унікальним ID;

- виявити стовпці з відсутніми значеннями та застосувати стратегію (наприклад, видалення рядків з NaN);

- переконатись, що числові стовпці (Age, Score) дійсно містять числові дані. Для цього, наприклад, можна спробувати конвертувати їх і обробити можливі помилки;

- виконати перевірку діапазону числових значень, чи знаходяться значення у стовпцях Age та Score в очікуваних межах;

- виконати перевірку категоріальних значень, чи значення у стовпцях Gender, Group, City відповідають очікуваним категоріям (опціонально, можна додати стандартизацію).

Код Python для зчитування файлу data.txt з перевірками на поширені проблєми в даних та базовими кроками їх очищення наведено у додатку Б, лістинг Б.2.

2.2. Бібліотеки Python для статистичної обробки даних

Найбільшою перевагою Python для науки про дані є широкий вибір бібліотек, які допомагають програмістам вирішувати більшість завдань статистичної обробки даних. Серед головних бібліотек для статистичної обробки та аналізу даних виділяють NumPy, SciPy, pandas, statsmodels та scikit-learn [16].

NumPy (Numerical Python) – масиви та математичні операції [17]. NumPy є фундаментальним пакетом для наукових обчислень у Python. Ця бібліотека Python використовує багатовимірний об’єкт масиву (ndarray) та низку функцій для швидких операцій над масивами, включаючи математичні, логічні, маніпуляції формою, сортування, вибір, введення/виведення, дискретні перетворення Фур’є, базову лінійну алгебру, базові статистичні операції, симуляцію випадкових чисел та багато іншого. NumPy може використовуватися для обробки великих багатовимірних масивів та матриць. Він використовує великий набір високорівневих математичних функцій, що робить його особливо корисним для

ефективних фундаментальних наукових обчислень. Арифметичні операції NumPy широко використовуються завдяки їх здатності виконувати прості та ефективні обчислення над масивами, включаючи поелементне додавання, віднімання, множення, ділення, піднесення до степеня та модуль. Бібліотека часто використовується для аналізу даних, створення потужних N-вимірних масивів та формування основи інших бібліотек, таких як SciPy та scikit-learn. Важливі особливості NumPy для науки про дані включають швидкі попередньо скомпільовані функції для числових процедур, підтримку об'єктно-орієнтованого підходу, орієнтацію на масиви для ефективніших обчислень, а також очищення та маніпуляції даними. Зазвичай NumPy імпортують як import numpy as np. Використання NumPy для створення масиву та виконання над ним операцій представлено у додатку Б, лістинг Б.3.

SciPy (Scientific Python) – спеціалізовані статистичні функції [18]. SciPy є ще однією бібліотекою Python для науки про дані. Це безкоштовна бібліотека Python з відкритим вихідним кодом, яка використовується для обчислень високого рівня. Вона особливо корисна для наукових та технічних обчислень та надає різні зручні та ефективні процедури для наукових обчислень. SciPy базується на NumPy та включає всі його функції, виконує наукові та технічні обчислення на великих наборах даних. Її часто застосовують для операцій з багатовимірними зображеннями, алгоритмів оптимізації та лінійної алгебри. Для простих статистичних тестів використовується підмодуль scipy.stats, що містить велику кількість розподілів ймовірностей, зведених та частотних статистик, кореляційних функцій та статистичних тестів. SciPy, як і TensorFlow, має велику та активну спільноту. Використання підмодуля scipy.stats для виконання статистичного тесту представлено у додатку Б, лістинг Б.4. Типовий вивід результату на рисунку 2.2.

```
T-статистика: -2.856
P-значення: 0.005
Є статистично значуща різниця між середніми значеннями двох наборів даних.
```

Рисунок 2.2 – Результат виконання t-тесту за допомогою scipy.stats

Pandas є однією з найбільш широко використовуваних бібліотек Python для науки про дані [19]. Вона має розвинені інструменти для обробки аналізу даних. Двома основними структурами даних бібліотеки pandas є Series та DataFrames: Series – це одновимірний мічений масив, що зберігає дані будь-якого типу; DataFrames – це двовимірна (таблична) структура даних. Обидві вони є швидкими та ефективними засобами керування та дослідження даних. До основних застосувань pandas належать загальна обробка та очищення даних, статистика, фінанси, створення діапазону дат, лінійна регресія та багато іншого. Pandas дозволяє створювати власні функції, має високорівневу абстракцію, високорівневі структури та інструменти маніпулювання, а також функції об'єднання/злиття наборів даних. Зазвичай pandas імпортують як import pandas as pd. Використання pandas для створення та маніпуляції DataFrame представлено у додатку Б, лістинг Б.5.

Statsmodels – це бібліотека Python корисна для статистики та тестування гіпотез [20]. Вона надає інструменти для підгонки різних статистичних моделей, виконання тестів та аналізу даних. Statsmodels особливо використовується для завдань у науці про дані, економіці та інших галузях, де важливе розуміння даних. Бібліотека розроблена, щоб полегшити роботу зі статистикою та надати зрозумілі та надійні результати. Statsmodels підтримує різні типи регресійних та лінійних моделей. Регресія – це статистичний метод, що використовується для розуміння зв'язків між змінними. Бібліотека надає лінійні моделі, які допомагають зрозуміти ці зв'язки та робити прогнози на основі даних. Це включає лінійну регресію (OLS – Ordinary Least Squares), яка є найбільш базовим методом лінійної регресії в statsmodels. Метою OLS є пошук найкращої прямої, що мінімізує різницю між фактичними точками даних та прогнозованими значеннями. Statsmodels також надає можливість використовувати «формули» для визначення статистичних моделей у Python, що робить її синтаксис схожим на R. Це можна побачити на прикладах підгонки моделей OLS для порівняння IQ між різними групами або типу IQ. Бібліотека також охоплює часові ряди, дискретний вибір, непараметричну статистику, узагальнені лінійні моделі, робастну регресію та багато іншого.

Використання statsmodels для лінійної регресії (OLS) представлено у додатку Б, лістинг Б.6. Результат на рисунку 2.3.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.908
Model:                            OLS   Adj. R-squared:                  0.907
Method:                 Least Squares   F-statistic:                     968.9
Date:                Fri, 23 May 2025   Prob (F-statistic):           1.31e-52
Time:                        20:14:12   Log-Likelihood:                -200.46
No. Observations:                 100   AIC:                             404.9
Df Residuals:                      98   BIC:                             410.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.4302      0.341      4.199      0.000       0.754       2.106
x1             1.9080      0.061     31.127      0.000       1.786       2.030
==============================================================================
Omnibus:                        0.900   Durbin-Watson:                   2.285
Prob(Omnibus):                  0.638   Jarque-Bera (JB):                0.808
Skew:                           0.217   Prob(JB):                        0.668
Kurtosis:                       2.929   Cond. No.                         10.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Коефіцієнт (нахил): 1.908
Перетин (константа): 1.430
```

Рисунок 2.3 – Результат застосування Statsmodels для лінійної регресії (OLS)

Scikit-learn – це потужна бібліотека Python для науки про дані, головним чином орієнтована на машинне навчання. Вона надає різноманітні корисні алгоритми, що легко інтегруються з SciPy та NumPy [21]. Завдяки таким алгоритмам, як градієнтний бустинг, DBSCAN, випадкові ліси та опорні векторні машини, бібліотека широко використовується для вирішення задач класифікації, регресії, кластеризації, зменшення розмірності та вибору моделі. Основні функції Scikit-learn включають попередню обробку даних, класифікацію та моделювання даних, вибір моделі та наскрізні алгоритми машинного навчання.

Використання scikit-learn для виконання простої лінійної регресії представлено у додатку Б, лістинг Б.7. Результат виконання коду представлено на рисунку 2.4.

```
Коефіцієнт (нахил): 1.920
Перетин (константа): 1.286
Середньоквадратична помилка (MSE): 2.615
Коефіцієнт детермінації (R-квадрат): 0.929
```

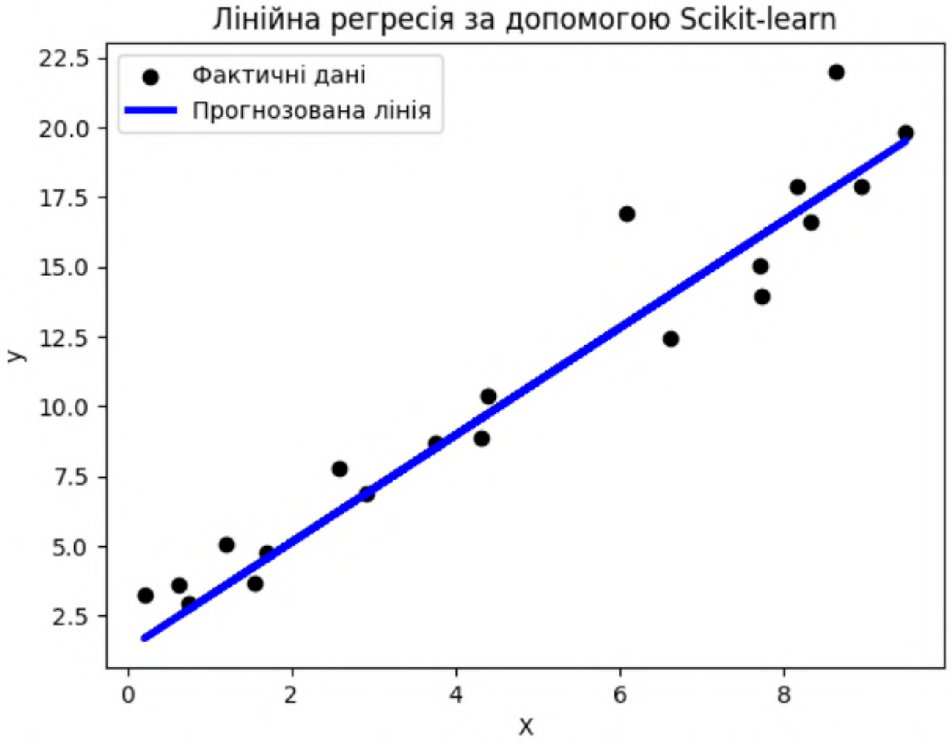

Рисунок 2.4 – Результат використання Scikit-learn для простої лінійної регресії

2.3 Реалізація основних статистичних обчислень в Python

Для статистичних обчислень у Python застосовуються основні бібліотеки NumPy, pandas та SciPy, а також бібліотеки для візуалізації Matplotlib та Seaborn.

Обчислення базових статистичних характеристик даних є важливим етапом аналізу даних. Бібліотека NumPy, яка є основою для багатьох інших бібліотек в екосистемі Python для обробки та аналізу даних, надає базові методи для маніпулювання великими масивами та матрицями, включає бібліотеку стандартних математичних функцій, які можуть бути застосовані до кожного елементу масиву, також має методи для обчислення суми (sum) та добутку (prod) елементів масиву. Для статистичних обчислень NumPy надає функції для знаходження середнього арифметичного (mean), дисперсії (var) та стандартного відхилення (std), а також методи для знаходження мінімального (min) та максимального (max) значень,

індексів мінімального (argmin) та максимального (argmax) елементів. NumPy також включає функції для обчислення медіани, коефіцієнтів кореляції та коваріаційного моменту. Використання NumPy для обчислення базових статистичних характеристик представлено у додатку В, лістинг В.1. Результат на рисунку 2.5.

```
Сума елементів: 202
Добуток елементів: 304234024550400
Середнє арифметичне: 16.83
Дисперсія: 23.14
Стандартне відхилення: 4.81
Мінімальне значення: 10
Індекс мінімального значення: 0
Максимальне значення: 23
Індекс максимального значення: 2
Медіана: 16.5
Коваріаційна матриця:
[[25.24242424 -2.54545455]
 [-2.54545455 13.        ]]
Кореляційна матриця:
[[ 1.         -0.14051676]
 [-0.14051676  1.        ]]
```

Рисунок 2.5 – Результат використання NumPy для обчислення базових статистичних характеристик

Бібліотека pandas, побудована на основі NumPy, є основним інструментом для обробки та аналізу структурованих і табличних даних [22]. Вона використовує зручні структури даних, такі як Series (одновимірний) та DataFrame (двовимірна таблиця), завдяки чому дозволяє легко й ефективно маніпулювати даними та аналізувати їх. Для обчислення підсумкової статистики, включаючи середнє значення, медіану, дисперсію, стандартне відхилення, коваріацію та кореляцію, використовуються вбудовані методи об'єктів DataFrame. Наприклад, метод describe() об'єкта DataFrame використовується для знаходження базових статистичних характеристик, таких як середнє (average), стандартне відхилення (standard deviation), мінімальне (minimum) та максимальне (maximum) значення. Коефіцієнт кореляції може бути розрахований за допомогою методу corr().

Використання pandas для обчислення базових статистичних характеристик представлено у додатку В, лістинг В.2. Результат на рисунку 2.6.

```
Базові статистичні характеристики DataFrame:
           Value1     Value2
count  12.000000  12.000000
mean   16.833333  18.083333
std     5.024184   3.629634
min    10.000000  13.000000
25%    12.000000  15.000000
50%    16.500000  18.000000
75%    21.500000  20.250000
max    23.000000  25.000000

Середнє значення 'Value1': 16.83
Медіана 'Value2': 18.00

Кореляційна матриця:
           Value1    Value2
Value1  1.000000  0.903145
Value2  0.903145  1.000000
```

Рисунок 2.6 – Результат використання pandas для обчислення базових статистичних характеристик

Вбудований модуль statistics у Python надає функції для обчислення математичної статистики числових даних [23]. Він включає функції для обчислення центрального значення, такі як середнє (mean), медіана (median), мода (mode) та квантилі (quantiles), а також міри поширення, такі як стандартне відхилення (stdev, pstdev) та дисперсія (variance, pvariance). Цей модуль не конкурує з потужними бібліотекамиPython, такими як NumPy або SciPy, і орієнтований на рівень графічних та наукових калькуляторів. Використання модуля statistics представлено у додатку В, лістинг В.3. Результат представлено на рисунку 2.7.

```
Середнє (statistics): 16.83
Медіана (statistics): 16.50
Мода (statistics): 23
Стандартне відхилення (statistics, вибіркове): 5.02
Дисперсія (statistics, вибіркова): 25.24
```

Рисунок 2.7 – Результат використання вбудованого модуля statistics

Бібліотека SciPy містить різноманітні підпакети, які дозволяють вирішувати найпоширеніші проблєми, пов'язані з науковими обчисленнями. Зокрема, підпакет scipy.stats використовується для статистики та випадкових чисел, а scipy.linalg для лінійної алгебри. SciPy надає функції для використання наукових обчислень, таких як оптимізація, апроксимація, інтегрування та інші [24]. Використання бібліотеки SciPy для статистичного аналізу представлено у додатку В, лістинг В.4. Результат на рисунку 2.8.

```
Описові статистики (scipy.stats.describe):
 DescribeResult(nobs=100, minmax=(np.float64(0.5478227059267651), np.float64(10.148303200141193)),
                mean=np.float64(4.702308336388942), variance=np.float64(3.4560048586909677),
               skewness=np.float64(0.2146153946187008), kurtosis=np.float64(0.039811494017144344))

 T-статистика (t-test): -2.847
 P-значення (t-test): 0.005
```

Рисунок 2.8 – Використання бібліотеки SciPy для статистичного аналізу даних

Очищення та обробка даних також є важливими етапами підготовки даних для аналізу. Бібліотека pandas має потужні інструменти для цих завдань. Працюючи переважно з табличними даними у вигляді DataFrame, pandas дозволяє легко читати та записувати дані з різних форматів файлів, включаючи CSV, Excel, бази даних SQL тощо.

Типові етапи підготовки даних за допомогою pandas включають [25]:

- зчитування даних. Дані можуть бути зчитані з вказаної директорії та перетворені у об'єкт типу DataFrame. Приклад зчитування даних з CSV файлу представлено у додатку В, лістинг В.5;

- видалення зайвої інформації з набору даних. Непотрібні для аналізу параметри або стовпці можуть бути видалені. Приклад видалення зайвої інформації (видаляється зайвий стовпець) представлено у додатку В, лістинг В.6;

- обробка відсутніх значень. Pandas дозволяє легко визначити відсутні значення та видалити їх або заповнити вказаним значенням, або значенням, обчисленим із сусідніх точок даних (наприклад, лінійна інтерполяція). Приклад обробки відсутніх значень представлено у додатку В, лістинг В.7;

- перетворення форматів та групування. Pandas дозволяє виконувати перетворення мітки часу у потрібний формат або групування даних за певним інтервалом (наприклад, хвилина, година, день) та обчислення агрегованих значень (наприклад, середнього) для кожного інтервалу. Приклад перетворення форматів та групування даних представлено у додатку В, лістинг В.8;

- реіндексація та фільтрація. Дані можуть бути реіндексовані для заповнення проміжків або відфільтровані за певними критеріями (наприклад, за діапазоном часу). Приклад виконання реіндексації та фільтрації даних представлено у додатку В, лістинг В.9.

Також під час обробки даних можуть застосовуватися методи згладжування даних для виявлення трендів та зменшення шуму. Поширеними методами є метод ковзного середнього, який реалізується у pandas за допомогою методів rolling() та mean(), та метод експоненціального згладжування, для реалізації якого можна використати клас ExponentialSmoothing з бібліотеки Statsmodels [26]. Приклад згладжування даних методом ковзного середнього (з використанням pandas) представлено у додатку В, лістинг В.10.

Statsmodels є бібліотекою, яка розширює можливості NumPy, SciPy та pandas і надає широкий набір інструментів для статистичного моделювання та аналізу, включаючи роботу з часовими рядами. Приклад експоненціального згладжування за допомогою Statsmodels представлено у додатку В, лістинг В.11.

NumPy також підтримує базові операції для обробки даних, такі як сортування, отримання унікальних елементів, маніпуляції з формою масивів (reshape), об'єднання (concatenate, vstack, hstack) та розділення масивів (array_split) [27]. Приклад операцій обробки масивів за допомогою NumPy представлено у додатку В, лістинг В.12.

Візуалізація даних є найбільш ефективним прийомом для розуміння розподілу даних, виявлення залежностей між змінними та представлення результатів аналізу. Найбільш уживаними інструментами для створення статичних та інтерактивних візуалізацій у Python є бібліотеки Matplotlib та Seaborn.

Matplotlib є фундаментальною бібліотекою для створення статичних, інтерактивних та анімованих візуалізацій у Python, що дозволяє будувати різноманітні типи графіків [28]:

- гістограми та графіки щільності розподілу використовуються для візуалізації розподілу однієї змінної. pandas має вбудовану підтримку для створення гістограм (додаток В, лістинг В.13).

- діаграми розсіювання (Scatter plots) показують залежність між двома числовими змінними (додаток В, лістинг В.14);

- лінійні графіки часто використовуються для візуалізації часових рядів (додаток В, лістинг В.15);

- теплові карти (Heatmaps) ефективні для візуалізації матриць, наприклад, матриці кореляції між багатьма змінними (додаток В, лістинг В.16);

Seaborn – бібліотека візуалізації даних, побудована на Matplotlib, яка надає високоякісні та естетично привабливі візуалізації, особливо корисні для статистичного дослідження даних. Seaborn спрощує створення складних візуалізацій, таких як:

- коробкові діаграми (Box plots) відображають розподіл даних та виявляють викиди (додаток В, лістинг В.17);

- скрипкові діаграми (Violin plots) комбінують ящикові діаграми з оцінкою щільності розподілу (додаток В, лістинг В.18);

- графіки парних залежностей (Pair plots) показують попарні залежності між кількома змінними (додаток В, лістинг В.19).

Загалом, бібліотека pandas інтегрується з Matplotlib та Seaborn, дозволяючи безпосередньо викликати методи візуалізації на об’єктах Series та DataFrame.

У вебдодатках для візуалізації даних можуть також використовуватися бібліотеки JavaScript, такі як ZingChart, для створення інтерактивних графіків на вебсторінці. ZingChart підтримує підключення до різних джерел даних (JSON, CSV) та створення різних типів діаграм, включаючи лінійні діаграми та теплові карти [29].

2.4 Автоматизація статистичного аналізу засобами Python

Написання скриптів Python дозволяє автоматизувати завдання аналізу даних, що є важливою навичкою для аналітика. Основні бібліотеки для аналітики даних в Python, які використовуються для написання таких скриптів, включають [30]:

- pandas – забезпечує автоматизацію завантаження даних та реалізує методи обробки, фільтрації, індексування, об'єднання та аналізу даних, а також виконання обчислень. Її оптимізовані алгоритми роблять її ефективною для роботи з великими наборами даних. Приклад використання pandas для завантаження та базової обробки даних представлено у додатку Г, лістинг Г.1;

- NumPy – використовується для втоматизації анаукових обчислень та роботи з багатовимірними масивами даних, часто виступаючи основою для pandas . Приклад використання NumPy для числових операцій представлено у додатку Г, лістинг Г.2;

- SciPy – надає широкий спектр математичних операцій та модулів для наукових обчислень, включаючи модуль scipy.stats для статистичних методів. Дозволяє автоматично виконувати різноманітні статистичні тести, розрахунки та оптимізацію. Приклад використання SciPy для статистичного тесту представлено у додатку Г, лістинг Г.3;

- Scikit-learn – автоматизує алгоритми класифікації, регресії, кластеризації та розрахунок метрик для оцінки моделей. Приклад використання Scikit-learn для простої лінійної регресії представлено у додатку Г, лістинг Г.4.

Написання скриптів дозволяє автоматизувати всі етапи аналізу даних від збору та очищення до моделювання та візуалізації. Прикладами завдань для автоматизації є:

- автоматичне завантаження та обробка даних (наприклад, з вебAPI);

- виконання ETL (Extract, Transform, Load) процесів для інтеграції даних з різних джерел;

- розрахунок описових статистик (мінімум, максимум, середнє, квартилі, мода, медіана) для великих обсягів даних;

- автоматичний моніторинг даних та сповіщення про зміни;
- автоматичне створення та оновлення звітів на основі даних.

Для інтерактивного кодування, дослідження даних та візуалізації аналітики даних активно використовують середовище Jupyter Notebook, що поєднує код Python, його вивід (включаючи візуалізації), текст та рівняння в єдиному інтерактивному документі. Використання Jupyter Notebook є особливо зручним на етапі дослідження даних, коли необхідно швидко перевіряти гіпотези, експериментувати з різними методами аналізу та візуалізувати проміжні та кінцеві результати. Ноутбуки також можуть використовуватися для обміну результатами аналізу з колегами. Jupyter Notebook дозволяє динамічно відображати результати статистичної обробки даних. Візуалізації, побудовані за допомогою бібліотек Matplotlib, Seaborn або Plotly, можуть бути безпосередньо вбудовані в ноутбук. Наприклад, після завантаження даних у pandas DataFrame, можна легко побудувати гістограму розподілу значень стовпця, виконавши команду типу df[‘стовпець’].hist() у комірці ноутбука після вказівки %matplotlib inline для відображення графіків [31]. Приклад візуалізації в Jupyter Notebook (окремі комірки) представлено у додатку Г, лістинг Г.5.

Автоматизація створення звітів є важливою сферою застосування Python в аналітиці даних. Для створення автоматизованих звітів з використанням Python застосовують шаблони – базові структури, які багаторазово використовуються для вставки нових даних. Бібліотеки Python, такі як ReportLab та PyPDF2, можуть використовуватися для програмного створення PDF-звітів та додавання даних у вже існуючі шаблони. Шаблони можуть містити текстові елементи (заголовки, підзаголовки, текст), таблиці для представлення даних, графічні елементи та зображення [32].

Крім статичних звітів, існує можливість створення інтерактивних візуалізацій, які можуть бути частиною більш складних звітів або дашбордів. Бібліотеки Python, такі як Plotly та Dash, використовуються для створення інтерактивних графіків та інформаційних панелей (дашбордів). Plotly дозволяє створювати графіки, на яких користувач може наводити курсор, щоб побачити

конкретні значення, масштабувати окремі ділянки та взаємодіяти з даними. Такі інтерактивні елементи можуть бути інтегровані для представлення результатів аналізу, підвищуючи цінність та якість звіту. Наприклад, дашборд, створений за допомогою Plotly та Dash, може відображати ключові показники, графіки трендів та розподілів, що динамічно оновлюються [33]. Приклад інтерактивного графіка, побудованого за допомогою Plotly, представлено у додатку Г, лістинг Г.6.

Розглянуті інструменти Python дозволяють автоматизувати повний цикл статистичного аналізу даних, що охоплює етапи від обробки значних обсягів інформації за допомогою скриптів на базі бібліотек pandas, NumPy, SciPy, Scikit-learn, інтерактивного аналізу та візуалізації в Jupyter Notebook, аж до формування та інтеграції інтерактивних елементів у фінальні звіти.

Таким чином, було проведено огляд найбільш важливих інструментів Python для проведення статистичної обробки та аналізу даних:

- продемонстровано гнучкість Python у роботі з різноманітними форматами даних, включаючи текстові (CSV, JSON, XML), бінарні (Excel, Pickle), бази даних (SQL, NoSQL) та спеціалізовані наукові формати (HDF5, NetCDF, Parquet);
- особливу увагу приділено форматам, які найчастіше використовуються в статистичному аналізі, таким як CSV, Excel, SQL та Parquet, та інструментам Python (переважно з бібліотеки pandas) для їх зчитування;
- розглянуто основні бібліотеки Python для статистичної обробки та аналізу даних, а саме:
  - NumPy, як фундаментальний пакет для наукових обчислень, що забезпечує ефективну роботу з багатовимірними масивами та базові математичні операції;
  - SciPy, як бібліотеку, що розширює можливості NumPy, надає для використання різноманітні спеціалізовані наукові та статистичні функції, включаючи крім описових статистик також статистичні тести та розподіли ймовірностей;

- pandas, як головний інструмент для всебічної обробки та аналізу структурованих даних, що надає структури даних DataFrame та Series, а також функції для маніпулювання даними;

- statsmodels, як бібліотеку, орієнтовану на побудову статистичних моделей, проведення тестування гіпотез та аналіз регресії; scikit-learn, як бібліотеку для машинного навчання, що також містить інструменти для кластерного аналізу, класифікації та попередньої обробки даних, які можуть бути корисними в статистичному аналізі;

- розглянуто обчислення базових статистичних характеристик (середнє, медіана, стандартне відхилення тощо) з використанням бібліотек NumPy, pandas та вбудованого модуля statistics. Також розглянуто етапи очищення та обробки даних за допомогою бібліотеки pandas, включаючи зчитування даних, видалення зайвої інформації та обробку відсутніх значень;

- представлені інструменти призначені для автоматизації всього процесу статистичного аналізу даних, що охоплює використання бібліотек pandas, NumPy, SciPy, Scikit-learn для обробки великих даних [34], Jupyter Notebook для інтерактивного дослідження та візуалізації, а також функції для автоматизованого створення та включення інтерактивних елементів у звіти.

РОЗДІЛ 3
РОЗРОБЛЕННЯ МЕТОДИКИ СТАТИСТИЧНОЇ ОБРОБКИ ДАНИХ ЗАСОБАМИ PYTHON

3.1 Методика статистичної обробки даних засобами Python

Повна статистична обробка даних включає дванадцять основних етапів, які можна виконати виключно з використанням бібліотек Python. Розглянемо зміст та послідовність виконання цих етапів.

1. Підготовка середовища та завантаження даних:

- встановлення основних бібліотек numpy, pandas, scipy, statsmodels, matplotlib, seaborn, а також інших бібліотек, які можуть знадобитися для аналізу даних (наприклад, pingouin, factor_analyzer);
- зчитування даних з різних джерел (CSV, Excel, бази даних, API тощо) у структури даних pandas (DataFrame) за допомогою функцій pd.read_csv(), pd.read_excel(), pd.read_sql() тощо.

2. Попереднє оброблення та первинний аналіз даних:

- отримання загального уявлення про структуру, типи даних та основні статистичні характеристики за допомогою методів DataFrame (.head(), .tail(), .info(), .describe(), .shape, .dtypes);
- очищення даних, що включає:
    - визначення пропущених значень (.isnull().sum()) та обрання стратегії їх обробки (видалення рядків/стовпців з пропусками .dropna(), заповнення статистичними показниками .fillna() – середнє, медіана, мода – або більш складними методами);
    - візуалізація розподілів даних (гістограми, коробкові діаграми за допомогою Matplotlib або Seaborn) для виявлення викидів та прийняття рішення щодо їх обробки (видалення, заміна на граничні значення, трансформація);
    - видалення дублікатів рядків (.duplicated().sum(), .drop_duplicates());

- перетворення типів даних стовпців (.astype()) (за потреби) (наприклад, категоріальні дані в числові для певних алгоритмів);

- трансформація даних (за потреби), що включає:

- масштабування (StandardScaler, MinMaxScaler зі Scikit-learn) для приведення числових ознак до єдиного масштабу;

- кодування категоріальних змінних – перетворення категоріальних змінних у числові (One-Hot Encoding pd.get_dummies(), Label Encoding зі Scikit-learn) для використання в статистичних моделях;

- створення нових ознак на основі існуючих, які можуть бути більш інформативними для подальшого аналізу (за потреби).

3. Описова статистика:

- обчислення основних статистичних показників: для числових даних використовуються методи pandas (.mean(), .median(), .mode(), .std(), .var(), .min(), .max(), .quantile(), .describe()) та NumPy (np.mean(), np.median(), np.std(), np.percentile()), для категоріальних даних – .value_counts() та .mode();

- візуалізація розподілу окремих змінних та зв'язків між ними – побудова гістограм (plt.hist(), sns.histplot()), коробкових діаграм (plt.boxplot(), sns.boxplot()), діаграм розсіювання (plt.scatter(), sns.scatterplot()).

4. Інференційна статистика:

- використання статистичних тестів з бібліотек SciPy (scipy.stats) та Pingouin (pingouin) для перевірки статистичних гіпотез (наприклад, t-тест для порівняння середніх, ANOVA для порівняння середніх кількох груп, критерій хі-квадрат для аналізу категоріальних даних, непараметричні тести (для даних без нормального розподілу) – критерії Манна-Уїтні, Вілкоксона);

- побудова довірчих інтервалів для оцінюваних параметрів моделей за допомогою Statsmodels.

5. Кореляційний аналіз [35]:

- розрахунок коефіцієнтів кореляції для оцінки лінійних та монотонних зв'язків між числовими змінними, використовується метод .corr() DataFrame

(Pearson, Spearman, Kendall) та функції SciPy (scipy.stats.pearsonr(), scipy.stats.spearmanr(), scipy.stats.kendalltau());

- візуалізація кореляційних матриць тепловими картами (sns.heatmap()).

6. Регресійний аналіз:

- побудова регресійних моделей. Для лінійних та логістичних регресійних моделей використовуються функції бібліотеки Statsmodels (sm.OLS(), sm.Logit()); бібліотека Scikit-learn також має інструменти для регресії (LinearRegression, LogisticRegression), що використовуються для машинного навчання; SciPy (stats.linregress()) можна використати для простої лінійної регресії;

- оцінка якості та значущості моделі – проаналізувати статистичні показники R-квадрат, p-значення коефіцієнтів, F-статистика тощо.

7. Дисперсійний аналіз (ANOVA) – для порівняння середніх значень залежної змінної між різними групами незалежної категоріальної змінної використовуються інструменти statsmodels.formula.api.ols() та statsmodels.stats.anova.anova_lm() або scipy.stats.f_oneway() та pingouin.anova().

8. Факторний аналіз та зниження розмірності [36]:

- виявлення латентних (прихованих) факторів, що можуть пояснювати кореляції між змінними, використовується sklearn.decomposition.FactorAnalysis або бібліотека factor_analyzer;

- аналіз головних компонент (PCA). Використовується для зменшення розмірності даних шляхом знаходження головних компонент, які пояснюють найбільшу дисперсію, використовується sklearn.decomposition.PCA.

9. Кластерний аналіз (за потреби). Для виявлення груп схожих об'єктів у даних використовуються алгоритми кластеризації зі Scikit-learn (KMeans, AgglomerativeClustering, DBSCAN) або ієрархічна кластеризація з scipy.cluster.hierarchy. Результати кластеризації візуалізуються за допомогою діаграм розсіювання з кольоровим кодуванням кластерів [37].

10. Аналіз часових рядів (за потреби). Для аналізу трендів, сезонності та прогнозування майбутніх значень часових рядів використовуються моделі ARIMA,

SARIMA та експоненційного згладжування зі Statsmodels (statsmodels.tsa) або бібліотека Prophet; візуалізація часових рядів та прогнозів – лінійні графіки [38].

11. Байєсівська статистика (за потреби) [39]:

- отримання апостеріорних розподілів параметрів для визначення ймовірнісних моделей, вибору апріорних розподілів та запуску алгоритмів MCMC за допомогою бібліотеки PyMC3;

- візуалізація апостеріорних розподілів, трасування MCMC та оцінка збіжності, використовується ArviZ.

12. Інтерпретація результатів та формування висновків:

- інтерпретація отриманих статистичних показників, результатів тестів та візуалізацій у контексті дослідження;

- формулювання висновків на основі проведеного аналізу;

- документування процесу обробки та аналізу даних, включаючи використані бібліотеки, код, параметри моделей та отримані результати.

Додаткові аспекти (автоматизація статистичної обробки даних):

- для повторюваних завдань розробити скрипти Python для автоматизації процесу обробки та аналізу даних;

- фінальна звітність формується з використанням бібліотек для створення звітів (наприклад, Jupyter Notebook з Markdown, ReportLab, WeasyPrint);

- для великих обсягів даних доцільне використання бібліотек для паралельних обчислень (наприклад, Dask, multiprocessing) або платформ для обробки великих даних (наприклад, Spark з PySpark).

Ця методика є загальною та може бути адаптована залежно від конкретного завдання, типу даних та цілей аналізу.

3.2 Практичне застосування запропонованої методики

Розглянуто модельні приклади зі сфери IT, що демонструють застосування запропонованої методики.

Приклад 1. Порівняти розподіл типів браузерів, які використовують користувачі для доступу до вебдодатку.

Цей приклад демонструє застосування основних кроків методики до задачі аналізу номінальних даних.

Візуалізація отриманих результатів представлена на рисунках 3.1 і 3.2.

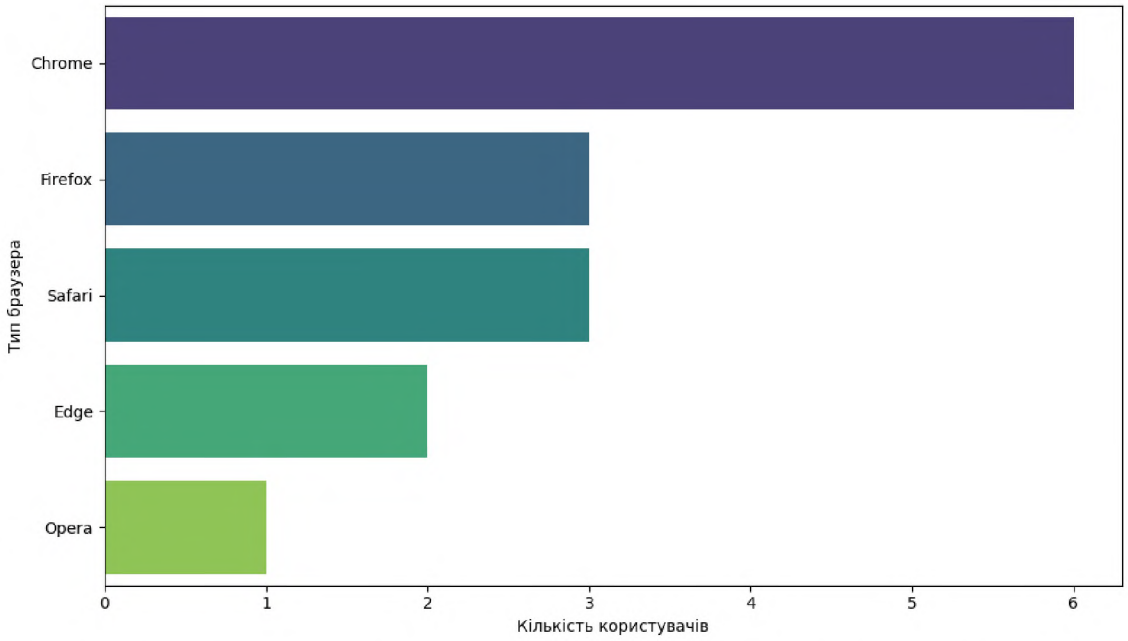

Рисунок 3.1 – Розподіл типів браузерів за кількістю користувачів

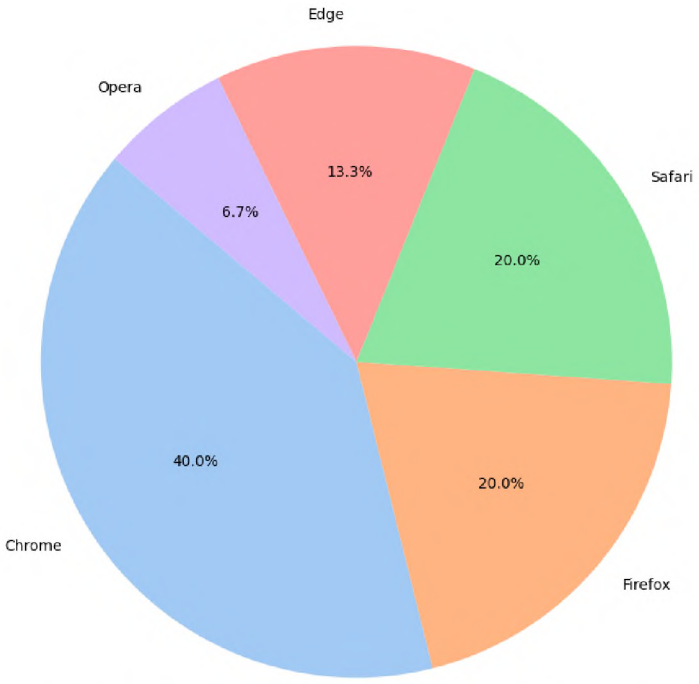

Рисунок 3.2 – Частка використання браузерів

Код реалізації (додаток Д, лістинг Д.1) об'єднує всі кроки статистичної обробки згідно методики (п.3.1) в один виконуваний блок. Спочатку він намагається завантажити дані з файлу browser_data.csv, а якщо файл не знайдено, використовує приклад даних, визначений у коді. Потім виконує первинний аналіз, описову статистику, інференційний аналіз (якщо є стовпець user_group) та візуалізацію розподілу типів браузерів. Наприкінці виводяться основні результати та підказки для інтерпретації.

Для цього конкретного завдання інференційна статистика може бути застосована для перевірки гіпотез про рівність розподілів браузерів між різними групами користувачів (якщо такі групи є у даних, наприклад, за регіоном або типом пристрою). Наприклад, може бути застосований критерій хі-квадрат для порівняння частот або непараметричні тести.

Кореляційний аналіз зазвичай застосовується до числових даних. Для номінальних даних можна використовувати спеціальні коефіцієнти кореляції, але це виходить за рамки даного прикладу.

Методи регресійного, дисперсійного, факторного, кластерного аналізу, аналіз часових рядів та байєсівська статистика зазвичай не застосовуються безпосередньо до одномірних номінальних даних, таких як типи браузерів, без додаткових контекстуальних змінних або перетворення даних.

На основі отриманих частот, моди та візуалізацій можна зробити висновки про розподіл типів браузерів серед користувачів вебдодатку. Наприклад:

- визначити найпопулярніший браузер (моду);
- порівняти частки використання різних браузерів;
- якщо проводився інференційний аналіз за групами користувачів, можна зробити висновки про наявність статистично значущих відмінностей у розподілі браузерів між цими групами.

Приклад 2. Порівняти рівень складності різних завдань у проєкті за оцінками розробників (за шкалою: дуже легко, легко, середньо, складно, дуже складно) між різними командами.

Цей приклад демонструє застосування методики (п.3.1) до порядкових даних, включаючи їх перетворення в категоріальний тип з порядком та використання відповідних статистичних тестів для порівняння груп.

Візуалізація отриманих результатів представлена на рисунку 3.3.

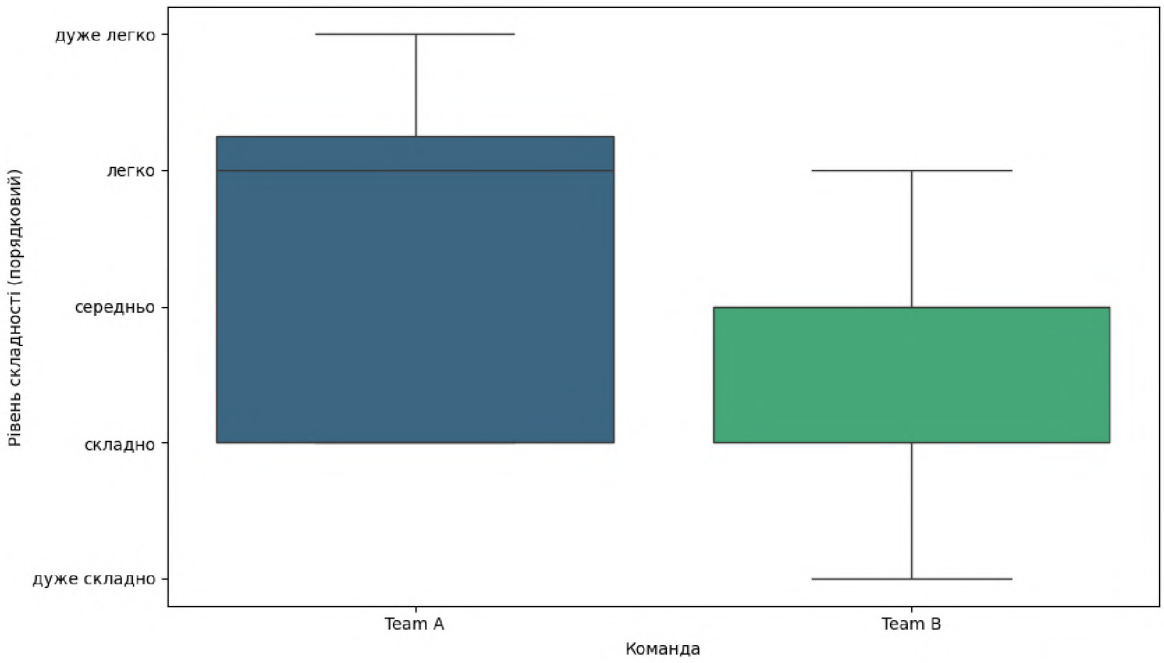

Рисунок 3.3 – Порівняння рівня складності між командами

Код прикладу (додаток Д, лістинг Д.2) об'єднує всі кроки аналізу порядкових даних в один блок, включаючи завантаження даних (з файлу або прикладу), попереднє оброблення, описову статистику (включаючи медіану), інференційну статистику (тести Манна-Уїтні або Краскела-Уолліса залежно від кількості команд) та візуалізацію розподілів. Наприкінці виводяться результати для інтерпретації.

На основі отриманих частот, медіанних рівнів складності та результатів непараметричних тестів (Манна-Уїтні або Краскела-Уолліса), а також візуалізацій, можна зробити висновки про порівняння рівня складності завдань між різними командами. Наприклад:

- визначити, чи є статистично значущі відмінності в розподілі оцінок складності між командами;
- порівняти медіанні рівні складності для кожної команди;

- візуально оцінити розподіли складності за допомогою стовпчастих діаграм та коробкових діаграм.

У даному прикладі метод .median() не може бути безпосередньо застосований до стовпця з категоріальним типом даних, навіть якщо він є впорядкованим. Для знаходження медіани для порядкових категоріальних даних, потрібно спочатку перетворити категорії на числові коди, а потім знайти медіану цих кодів. Після цього можна повернути відповідну категорію.

Приклад 3. Дослідити розподіл кількості запитів до API за секунду та порівняти його з очікуваним навантаженням.

Цей приклад демонструє застосування методики до дискретних даних, включаючи обчислення основних статистичних характеристик, візуалізацію розподілу та порівняння з очікуваним значенням за допомогою статистичного тесту.

Візуалізація отриманих результатів частково представлена на рисунку 3.4.

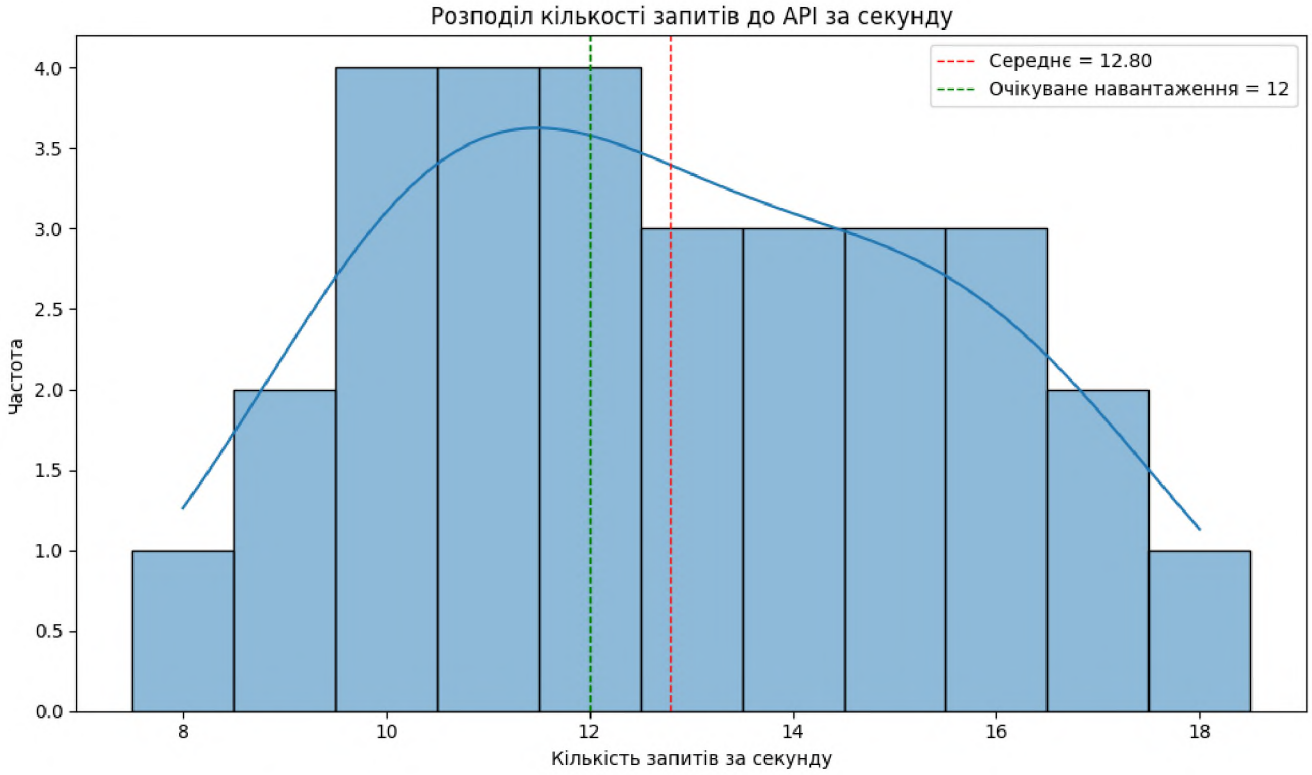

Рисунок 3.4 – Розподіл запитів до API за секунду

Код (додаток Д, лістинг Д.3) об'єднує всі кроки аналізу дискретних даних про кількість запитів до API за секунду в один виконуваний блок. Він включає завантаження даних, первинний аналіз, описову статистику (середнє, медіану, дисперсію, стандартне відхилення), візуалізацію розподілу (гістограма та boxplot), порівняння з очікуваним навантаженням та проведення одновибіркового t-тесту для формальної перевірки статистичної значущості відмінності середнього від очікуваного значення. Наприкінці виводяться основні результати для інтерпретації.

На основі отриманої описової статистики, візуалізації розподілу та порівняння з очікуваним навантаженням можна зробити висновки про характер навантаження на API:

- оцінити середню кількість запитів за секунду та порівняти її з очікуваним значенням;
- проаналізувати розкид даних (дисперсію, стандартне відхилення) для розуміння стабільності навантаження;
- за допомогою гістограми візуалізувати форму розподілу кількості запитів (наприклад, чи є піки, чи розподіл симетричний);
- виявити можливі аномалії або несподівані піки навантаження;
- на основі статистичного тесту (наприклад, одновибіркового t-тесту) формально перевірити, чи є відмінність між фактичним та очікуваним середнім навантаженням статистично значущою.

Приклад 4. Оцінити середній час завантаження вебсторінки та побудувати довірчий інтервал. Дослідити залежність між обсягом трафіку на сервері та часом відгуку. Порівняти середній час виконання певного алгоритму на різних наборах вхідних даних.

Цей приклад демонструє застосування статистичних методів для аналізу неперервних даних у різних IT-сценаріях, включаючи обчислення довірчих інтервалів, кореляційно-регресійний аналіз та порівняння середніх значень за допомогою t-тесту та ANOVA.

Код реалізації представлений у додатку Д, лістинг Д.4.

Візуалізація отриманих результатів частково представлена на рисунку 3.5.

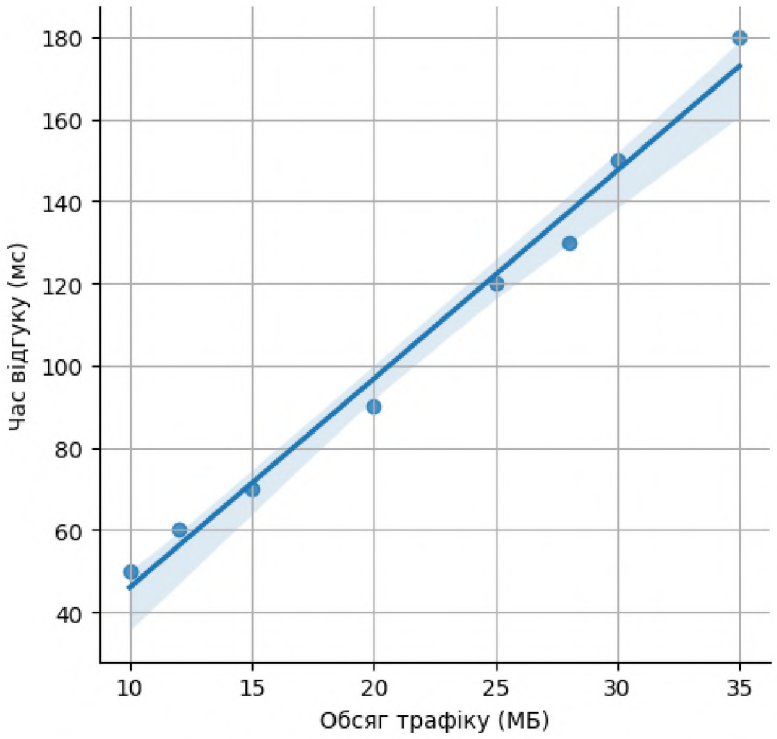

Рисунок 3.5 – Залежність між трафіком та часом відгуку сервера (лінійна регресія)

Приклад 5.Оцінити зміну середньої завантаженості центрального процесора (CPU utilization) протягом тижня.

Цей приклад демонструє застосування різних етапів методики для аналізу інтервальних даних.

Код реалізації (додаток Д, лістинг Д.5) виконує всі необхідні кроки для аналізу часового ряду завантаженості CPU, включаючи:

- завантаження даних (з альтернативними даними);
- первинний аналіз та попереднє оброблення даних (виявлення та видалення базових викидів);
- описова статистика;
- виявлення тренду (за допомогою ковзної середньої);
- виявлення періодичності (візуально на основі середніх погодинних значень);
- виявлення сезонності (за допомогою seasonal_decompose);
- виявлення аномалій (на основі стандартного відхилення);
- базова інтерпретація результатів.

Візуалізація результатів частково представлена на рисунку 3.6.

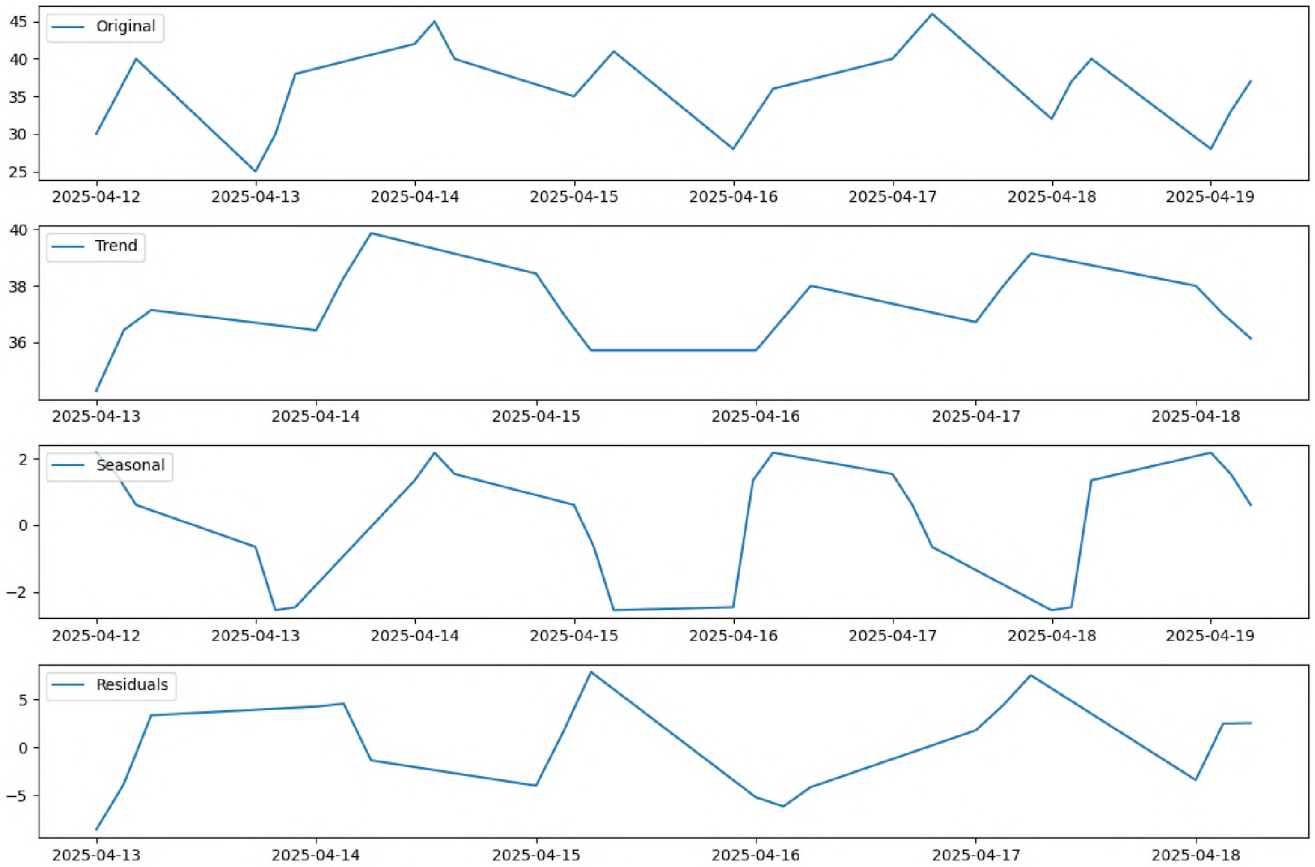

Рисунок 3.6 – Візуальний аналіз тренду та сезонних коливань завантаженості CPU

Проведений аналіз часового ряду завантаженості CPU протягом тижня дозволяє встановити наступне:

- візуалізація ковзної середньої дає уявлення про загальний тренд завантаженості;
- аналіз середньої погодинної завантаженості дозволяє виявити можливу добову періодичність;
- сезонна декомпозиція дозволяє виділити сезонну складову, тренд та випадкові коливання;
- виявлені аномалії (якщо такі є) – це точки, де завантаженість CPU значно відхиляється від середнього значення.

Подальший аналіз може бути спрямований на більш глибоке розуміння причин виявлених трендів, періодичності, сезонності та аномалій.

Попередні приклади були сфокусованими на окремих завданнях і не повністю відображали всі етапи запропонованої методики. Розглянемо комплексний приклад, що узагальнює застосування методики на одному конкретному завданні, щоб показати, як різні етапи можуть бути залучені разом.

Комплексний приклад. Проаналізувати дані про час завантаження вебсторінок для двох різних версій вебсайту (A та B) та порівняти їх ефективність.

Код реалізації (додаток Д, лістинг Д.6) виконує всі кроки аналізу, включаючи завантаження даних (з альтернативними даними у разі відсутності файлу), первинний аналіз, попереднє оброблення (виявлення та обробка викидів), описову статистику, інференційну статистику (тест на нормальність та відповідний тест для порівняння груп) та базову інтерпретацію результатів. Візуалізації включені для кращого розуміння даних на різних етапах аналізу.

Візуалізація отриманих результатів частково представлена на рисунку 3.7.

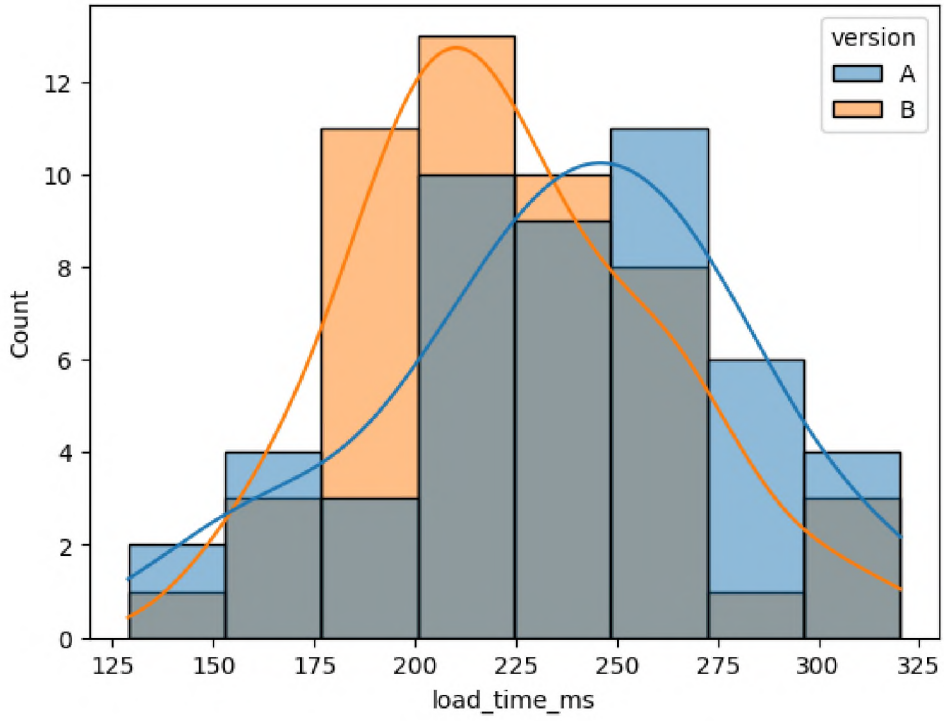

Рисунок 3.7 – Розподіл часу завантаження для версій A та B (після обробки викидів)

Цей приклад демонструє застосування кількох етапів з наданої методики: завантаження даних; первинний аналіз – огляд даних, виявлення пропусків та

унікальних значень; попереднє оброблення – виявлення та базова обробка викидів; описова статистика – розрахунок основних показників та візуалізація розподілів; інференційна статистика – перевірка гіпотез про різницю між групами (t-тест або тест Манна-Уїтні); візуалізація результатів – порівняння часу завантаження за допомогою коробкової діаграми (boxplot) після очищення даних; інтерпретація результатів – формулювання висновків на основі статистичних тестів та візуалізації. Проведений аналіз дозволив порівняти час завантаження вебсторінок для двох версій сайту (A та B). Проведена попередня обробка даних, включаючи візуальне виявлення та статистичну обробку викидів. За допомогою тесту Шапіро-Уїлка оцінено нормальність розподілів даних, і на цій основі застосовано t-тест Стьюдента (або тест Манна-Уїтні) для порівняння середніх (або розподілів) часу завантаження. Оскільки p-значення (0,090) більше за рівень значущості (0,05), не має достатньо доказів, щоб відхилити нульову гіпотезу про відсутність різниці в середньому часі завантаження між версіями A та B.

Приклад 7. Визначити середній обсяг оперативної пам'яті (RAM) використаної процесами та дослідити його залежність від кількості активних користувачів. Код реалізації: додаток Д, лістинг Д.7. Результат на рисунку 3.8.

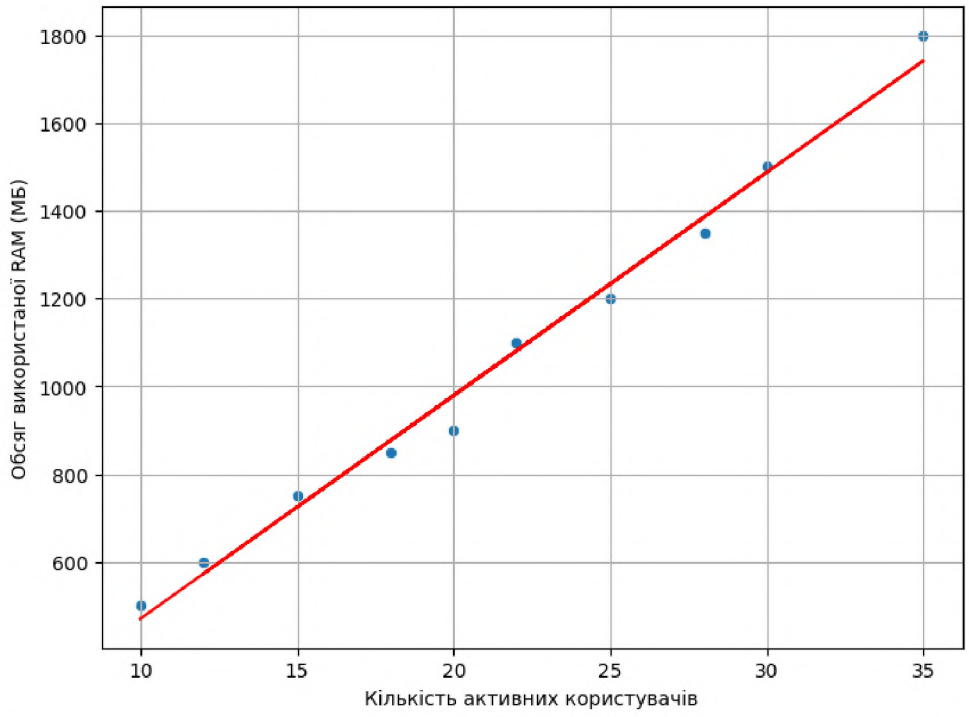

Рисунок 3.8 – Залежність використання RAM від кількості активних користувачів

Приклад 8. Порівняти середню швидкість передачі даних (пропускну здатність) між різними сегментами мережі.

Код реалізації: додаток Д, лістинг Д.8. Результат на рисунку 3.9.

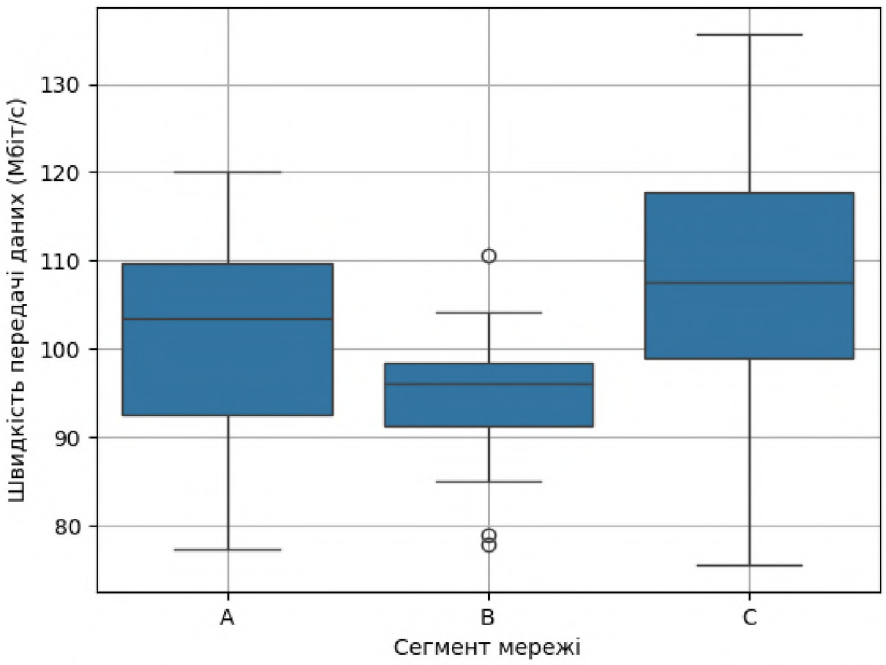

Рисунок 3.9 – Порівняння пропускної здатності мережі за сегментами

Приклади 7 і 8 продемонстрували застосування запропонованої методики статистичної обробки даних (п. 3.1) до відносних даних.

3.3 Інтеграція рішень статистичної обробки даних засобами Python

Інтегроване рішення для статистичної обробки даних за допомогою Python може охоплювати декілька етапів, зокрема це: автоматизація процесу обробки даних, побудова моделей прогнозування та генерація підсумкових документів (звітів) у різних форматах, таких як PDF та/або HTML [40].

Використання Python для автоматизації обробки даних реалізується через створення ETL-конвеєрів (Extract, Transform, Load – вилучення, перетворення, завантаження), що є процедурою, яка вилучає дані з різних джерел, перетворює їх

у відповідний формат і завантажує в цільову базу даних або сховище даних. ETL-конвеєри автоматизують цей процес, забезпечуючи ефективне переміщення та трансформацію даних для підтримки аналітики та звітності. Python є зручним для побудови ETL-конвеєрів завдяки читабельності коду, підтримці спільноти, багатій екосистемі бібліотек та гнучкості інтеграції з різними джерелами даних [41].

Приклад базової структури ETL-конвеєра в Python представлений у додатку Е (лістинг Е.1) описує модульну структуру, де кожен етап ETL представлений окремою функцією, а головна функція run_etl_pipeline координує їх виконання. Для етапів вилучення та перетворення даних типовим є використання, таких бібліотек, як pandas.

Побудова моделей прогнозування є одним з фундаментальних завдань науки про дані. Зокрема, прогнозування часових рядів – це завдання передбачення майбутніх значень на основі історичних даних.

Загальний процес побудови прогнозної моделі включає кілька кроків [42]:

- отримання даних, на основі яких буде будуватись модель;
- очищення набору даних – обробка некоректних значень, різних написань, невідповідностей та пропущених даних;
- побудова та оцінка моделі – вибір моделі, її побудова на даних та оцінка її якості.

Для побудови прогнозних моделей у Python використовуються бібліотеки NumPy, pandas (для завантаження та маніпулювання даними) та Scikit-learn (для побудови самих моделей). Для більш спеціалізованих часових рядів можуть використовуватися бібліотеки типу Statsmodels, що надає інструменти для побудови моделей ARMA, ARIMA та SARIMA, які доступні в одній бібліотеці.

Використання бібліотеки Statsmodels для побудови та прогнозування за допомогою моделі SARIMAX представлене у додатку Е, лістинг Е.2[43]. Цей код демонструє, як визначити модель SARIMAX, підігнати її до даних, отримати прогнози з довірчими інтервалами та візуалізувати результати. Оцінка моделі може проводитися за допомогою метрик, таких як коренева середньоквадратична

помилка (RMSE) або метрик, що залежать від типу завдання (наприклад, recall_score для класифікації).

При побудові прогнозних моделей, особливо моделей класифікації (наприклад, для передбачення дефолту за кредитами), використовуються алгоритми, такі як Random Forest, та інструменти для оптимізації гіперпараметрів, як GridSearchCV з Scikit-learn. Процес побудови та оцінки моделі є ітеративним і включає уточнення моделі та гіперпараметрів для досягнення найкращої продуктивності [44].

Моделювання також може застосовуватися для аналізу фінансових та економічних показників. Наприклад, регресійні моделі можуть використовуватися для аналізу залежності доходів бюджету від митних надходжень. Для такого аналізу можуть використовуватися бібліотеки NumPy, Statsmodels, Matplotlib (для візуалізації) та Xlrd (для читання даних з Excel) [45].

Однією з важливих задач при роботі з даними є звітність, представлена у читабельному та доступному форматі. Python надає можливості для генерації звітів у різних форматах, включаючи Excel, HTML та PDF.

Генерація HTML-звітів. HTML є стандартною мовою розмітки для створення вебсторінок і дозволяє легко вбудовувати звіти на вебсторінку або в тіло електронного листа. У Python можна генерувати HTML-звіти як за допомогою базового підходу (формування HTML-коду як рядка), так і за допомогою шаблонізаторів, таких як Jinja2. Jinja2 дозволяє створювати HTML-шаблони зі спеціальними змінними та керуючими структурами, подібними до синтаксису Python, а потім заповнювати їх даними для генерації кінцевого документа. Це особливо корисно для складних, повторюваних звітів [46].

Приклад використання Jinja2 для створення HTML-звіту представлений у додатку Е, лістинг Е.3 [47]. Цей шаблон показує, як використовувати змінні {{...}}, включати інші файли {% include ... %}, та використовувати керуючі структури типу {% if ... %} та {% for ... %}. Після створення шаблону, його можна відрендерити, передавши необхідні дані за допомогою бібліотеки Jinja2 в Python.

Генерація PDF-звітів. PDF є популярним форматом для обміну даними та звітами, оскільки він зберігає форматування та є загальнодоступним. Python пропонує кілька бібліотек для генерації PDF-звітів [48]:

- ReportLab – потужна бібліотека для створення складних PDF-документів з текстом, зображеннями, графіками та власними графічними елементами. Вона підходить для професійних звітів та документів з високою якістю друку. ReportLab дозволяє створювати таблиці та стилізувати їх. Приклад створення PDF та додавання таблиці за допомогою ReportLab представлений у додатку Е, лістинг Е.4. Цей приклад ілюструє створення простого документа, перетворення даних DataFrame на список списків для таблиці, застосування базового стилю та побудову PDF-файлу, що містить цю таблицю;

- FPDF – легка бібліотека для генерації PDF для простих документів з текстом та зображеннями. Вона простіша для освоєння порівняно з ReportLab. Приклад створення простого PDF за допомогою FPDF представлений у додатку Е, лістинг Е.5;

- WeasyPrint, PDFKit (використовує wkhtmltopdf) та xhtml2pdf – бібліотеки, що дозволяють конвертувати HTML та CSS у високоякісні PDF-документи. Це зручно, якщо ви вже маєте досвід роботи з HTML/CSS або якщо звіт легко представити у вебформаті. Приклад конвертації HTML у PDF за допомогою WeasyPrint представлений у додатку Е, лістинг Е.6;

- img2pdf та використання Pillow з img2pdf – бібліотеки для конвертації зображень у PDF. img2pdf підходить для простої конвертації без втрат, тоді як Pillow дозволяє попередньо обробляти зображення (зміна розміру, формату) перед їх включенням до PDF;

- pdfdocument – бібліотека для створення PDF з мінімальним налаштуванням, підходить для простих документів.

Таким чином, інтегроване рішення дозволяє об’єднати наступні етапи: дані вилучаються, обробляються та аналізуються (можливо, з побудовою прогнозних моделей) за допомогою pandas, NumPy та Scikit-learn/Statsmodels, а потім результати аналізу та прогнозування форматуються у звіти (наприклад, таблиці та

графіки) та генеруються у вигляді HTML або PDF документів для подальшого використання чи поширення. Загалом, Python надає вичерпний набір інструментів та бібліотек для створення комплексного рішення, яке автоматизує статистичну обробку даних.

3.4 Оцінка економічної ефективності запропонованої методики

Для оцінки економічної ефективності запропонованої методики статистичного аналізу даних засобами Python виконаємо модельний розрахунок економічного ефекту (щомісячна економія), сукупного економічного ефекту (річна економія), терміну окупності та рентабельності інвестицій у впровадження запропонованої методики [49].

Вихідні економічні та організаційні показники представлені у таблиці 3.1. Актуальна інформація стосовно оплати праці аналітиків даних в Україні [50].

Таблиця 3.1 – Вихідні економічні та організаційні показники

Параметр	Позначення	Значення (грн)	Примітка
Щомісячні витрати на обробку даних до впровадження	C_{tr}	42000	Людський ресурс (3 аналітики × 20 год × 700 грн/год)
Витрати на впровадження методики (разові)	C_{impl}	18000	Розробка, навчання, інтеграція
Щомісячні витрати після впровадження	C_{py}	11000	Один аналітик, підтримка
Скорочення часу на обробку даних	ΔT	45 год/міс	Зменшення з 60 до 15 год/міс
Вартість 1 години роботи	C_h	500 грн	Середня по ринку (аналітик та експерт)
Курс валют (курс НБУ станом на 02.05.2025 року)	1 USD 1 EUR	41,5945 грн 47,0808 грн	Для альтернативних оцінок

Щомісячна економія (економічний ефект) розраховується за формулою [51]:

$$E_m = \left(C_{tr} - C_{py}\right) + C_h \cdot \Delta T, \tag{3.1}$$

де: C_{tr} – витрати до впровадження методики, грн;

C_{py} – витрати після впровадження методики, грн;

ΔT – скорочення (економія) часу, год/міс;

C_h – ставка за годину, грн/год.

Термін окупності інвестицій розраховується за формулою [51]:

$$T_{pp} = \frac{C_{impl}}{E_m}, \tag{3.2}$$

де: C_{impl} – витрати на впровадження методики (разові), грн;

E_m – щомісячна економія (економічний ефект), грн/міс.

Рентабельність інвестицій (ROI) розраховується за формулою [51]:

$$ROI = \frac{E_y - C_{impl}}{C_{impl}} \cdot 100\%. \tag{3.3}$$

Розрахунок за формулою (3.1) з використанням даних таблиці 3.1 дає:

$$E_m = (42000 - 11000) + 500 \cdot 45 = 31000 + 22500 = 53500 \text{ грн/міс.}$$

Позначимо через E_y річну економію (сукупний економічний ефект), тоді:

$$E_y = E_m \cdot 12 = 53500 \cdot 12 = 642000 \text{ грн/рік.}$$

Розрахунок за формулою (3.2) з використанням даних таблиці 3.1 дає:

$$T_{pp} = \frac{18000}{53500} \approx 0{,}34 \text{ року} \approx 4 \text{ місяці.}$$

Рентабельність інвестицій (ROI) обчислимо за формулою (3.3):

$$ROI = \left(\frac{642000 - 18000}{18000}\right) \cdot 100\% \approx 346{,}6\%$$

Потенційна економія при масштабуванні (на 5 відділів) становить:

$$E_y^{(5)} = E_y \cdot 5 = 642000 \cdot 5 = 321000{,}0 \text{ грн/рік.}$$

Порівняння показників до та після впровадження розробленої методики представлене у таблиці 3.2.

Таблиця 3.2 – Порівняння показників до та після впровадження розробленої методики

Показник	До впровадження	Після впровадження	Зміна (+/-)
Витрати на місяць, грн	42000	11000	-31000
Трудові витрати, год/міс	60	15	-45
Вартість праці, грн	30000	7500	-22500
Економія на місяць, грн	–	–	53500
Річна економія, грн	–	–	642000
ROI, %	–	–	346,6 %
Термін окупності, міс	–	–	4 місяці

Таким чином, економія у 6 разів перевищує витрати на впровадження вже в перший рік. Термін окупності менше 4 місяців вказує на високий інвестиційний потенціал. На діаграмі (рисунок 3.1) представлено графік точки беззбитковості:

- суцільна похила лінія показує накопичену економію від впровадження запропонованої методики статистичної обробки даних засобами Python;
- пунктирна горизонтальна лінія – разові інвестиції (18000 грн).
- пунктирна вертикальна лінія – позначає момент, коли економія перевищує інвестиції (~4 місяці).

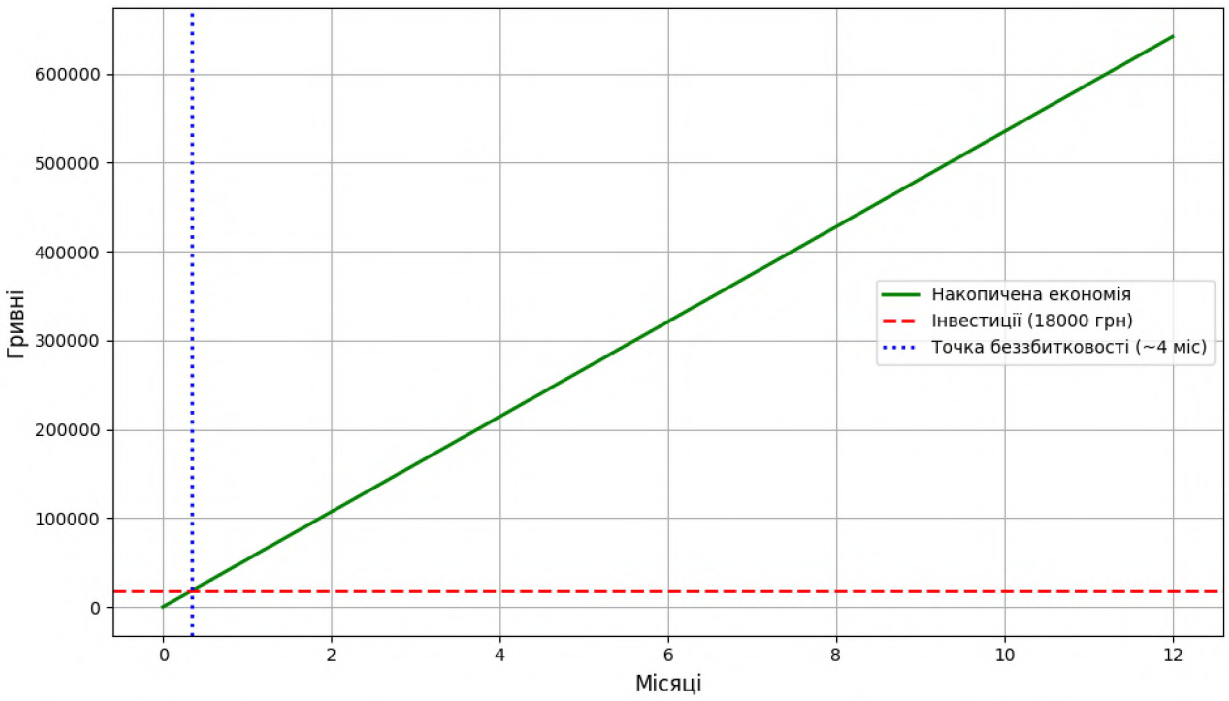

Рисунок 3.1 – Точка беззбитковості при впроваджені розробленої методики

Показник ROI на рівні 346,6% свідчить про вигідне впровадження. При масштабуванні на 5 аналогічних підрозділів економічний ефект сягне понад 3,2 млн грн/рік, що робить розроблену методику доцільною для повномасштабного розгортання. ROI-спіраль, представлена на діаграмі (рисунок 3.2), ілюструє швидке зростання віддачі на інвестиції: кожен виток показує накопичення вигоди у вигляді зростаючої спіралі, що відображає високу ефективність вкладених коштів.

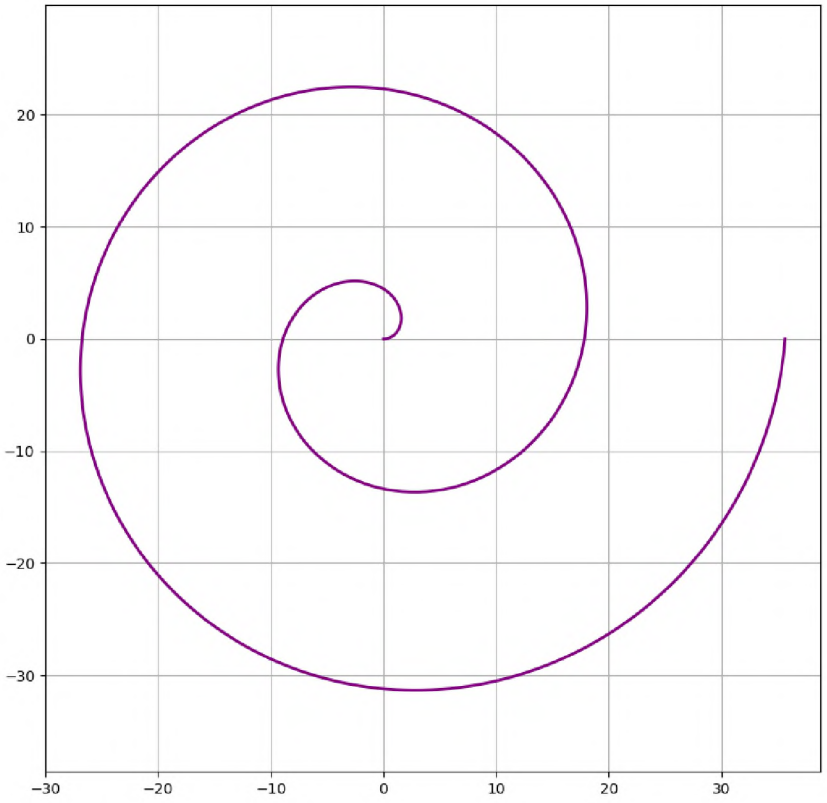

Рисунок 3.2 – ROI-спіраль при впроваджені розробленої методики

Таким чином, методика статистичної обробки на базі Python має високу ефективність, зменшуючи трудозатрати на 75% і одночасно покращуючи якість аналізу даних.

Вище було детально розглянуто методику статистичної обробки даних засобами Python, що охоплює 12 основних етапів – від підготовки середовища до інтерпретації результатів.

Практичне застосування запропонованої методики було продемонстровано на модельних прикладах з IT-сфери, що охоплюють різні типи даних: номінальні, порядкові, дискретні, неперервні та інтервальні.

Розглянуто інтегровані рішення для статистичної обробки даних, включаючи автоматизацію за допомогою ETL-конвеєрів, побудову моделей прогнозування та генерацію звітів у різних форматах.

Проведено оцінку економічної ефективності запропонованої методики, яка показала значну економію ресурсів та високий потенціал масштабування.

Таким чином, розроблена методика є ефективним інструментом для статистичної обробки даних засобами Python, що може бути застосована в різних IT-проєктах з метою підвищення ефективності аналізу даних та оптимізації витрат.

ВИСНОВКИ

У даній кваліфікаційній роботі було проведено комплексне дослідження методів статистичної обробки даних засобами Python з метою розробки та апробації відповідної методики.

В ході дослідження були виконані наступні завдання:

- розглянуто теоретичні основи статистичного аналізу даних, включаючи основні поняття, етапи та методи;
- проаналізовано існуючі бібліотеки Python для статистичної обробки даних, такі як NumPy, pandas, SciPy, statsmodels, scikit-learn, та їх можливості;
- розроблено методику статистичної обробки даних засобами Python, що включає 12 основних етапів, від підготовки середовища до інтерпретації результатів;
- продемонстровано практичне застосування розробленої методики на модельних прикладах з IT-сфери, що охоплюють різні типи даних;
- проведено оцінку економічної ефективності запропонованої методики, яка показала значну економію ресурсів та високий потенціал масштабування.

Отримані результати підтверджують досягнення поставленої мети та виконання завдань дослідження.

Основні висновки:

- розроблена методика статистичної обробки даних засобами Python є ефективним інструментом для аналізу даних в IT-сфері, що дозволяє автоматизувати процеси, підвищити точність результатів та зменшити витрати ресурсів;
- використання Python та його спеціалізованих бібліотек забезпечує гнучкість, масштабованість та можливість інтеграції з іншими інструментами та системами;
- економічна ефективність впровадження методики підтверджується значною економією витрат та швидким терміном окупності інвестицій.

Рекомендації щодо практичного використання:

- рекомендується впровадження розробленої методики в IT-компаніях для оптимізації процесів аналізу даних та підвищення ефективності прийняття рішень;
- результати дослідження можуть бути використані для подальших розробок у галузі автоматизації статистичного аналізу даних;
- доцільно продовжити дослідження з метою розширення функціональності методики, її інтеграції з хмарними обчисленнями та іншими сучасними технологіями обробки даних.