МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розроблення комп'ютерної програми для аналізу двофакторних експериментів з повторенням спостережень

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

Виконав: студент групи МгІТ-1-22

Бут.Є.О

Науковий керівник:
д.ф.-м..н., проф. Краснитський С.М
Рецензент к.т.н., доц. Яхно В.М

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ
ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерні науки

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

_____________ Володимир ЩЕРБАНЬ

«___»________________2023___року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Бут Євгеній Олегович

1.Тема роботи: Розроблення комп'ютерної програми для аналізу двофакторних експериментів з повторенням спостережень
Науковий керівник роботи Сергій Краснитський, д.ф.-м..н., проф.
затверджені наказом закладу вищої освіти від _12 . 09.2023 року , № 210-уч.
2. Вихідні дані до кваліфікаційної роботи:
3. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)
Вступ. РОЗДІЛ 1. Теоретичні аспекти аналізу двофакторних експериментів; РОЗДІЛ 2. Вибір мови програмування та технологій; РОЗДІЛ 3. Реалізація алгоритмів аналізу. Висновки.
4. Дата видачі завдання 08.2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної магістерської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	30.08.2023	
2	Розділ 1 Теоретичні аспекти аналізу двофакторних експериментів	06.09.2023	
3	Розділ 2. Вибір мови програмування та технологій	28.09.2023	
4	Розділ 3. Реалізація алгоритмів аналізу	21.10.2023	
5	Висновки	29.10.2023	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	06.11.2023	
7	Подача кваліфікаційної роботи (проєкту) науковому керівнику для відгуку (за 14 днів до захисту)		
8	Подача кваліфікаційної роботи (проєкту) для рецензування		
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату		
10	Подання кваліфікаційної роботи на затвердження завідувачу кафедри		

З завданням ознайомлений:

Студент ___________ Евгеній БУТ

Науковий керівник ___________ Сергій КРАСНИТСЬКИЙ

АНОТАЦІЯ

Бут Є.О Розроблення комп'ютерної програми для аналізу двофакторних експериментів з повторенням спостережень

Дипломна кваліфікаційна робота за спеціальністю 122 - «Компю'терні науки». – Київський національний університет технологій та дизайну, Київ, 2023 рік

Магістерська робота присвячена розробленню та впровадженню комп'ютерної програми для аналізу двофакторних експериментів з повторенням спостережень. Програма реалізована з використанням сучасних технологій програмування та включає в себе інтуїтивний інтерфейс користувача, який спрощує введення та обробку даних.У роботі представлено теоретичні аспекти аналізу двофакторних експериментів та вивчено особливості повторення спостережень, а також розглянуто статистичні методи, використані в розробленій програмі. Програма була протестована нареальних наборах даних для підтвердження її ефективності та правильності результатів.Розроблена програма може стати корисним інструментом для дослідників, які проводять подібні експерименти та потребують надійного засобу для аналізу та визначення статистичної значущості факторів та їх взаємодії.

Ключові слова: PYTHON, ANOVA, двофакторний експеримент, matplotlib, статистичні методи,SQLite , pyplot.

ANNOTATION

BUT E.O Development of a computer programme for analysing two-factor experiments with repeated measures

Master's diploma qualification work in speciality 122 - "Computer Science." - Kyiv National University of Technology and Design, Kyiv, 2023

The master's thesis is devoted to the development and implementation of a computer program for analysing two-factor experiments with repetition of observations. The program is implemented using modern programming technologies and includes an intuitive user interface that simplifies data entry and processing. The paper presents theoretical aspects of the analysis of two-factor experiments and studies the features of repeated measures, as well as the statistical methods used in the developed program. The program has been tested on real data sets to confirm its effectiveness and correctness of the results. The developed program can be a useful tool for researchers conducting similar experiments and needing a reliable means to analyse and determine the statistical significance of factors and their interaction.

Key words: PYTHON, ANOVA, two-factor experiment, matplotlib, statistical methods, SQLite, pyplot.

ЗМІСТ

ВСТУП

Актуальність. Актуальність розробки комп'ютерної програми для аналізу двофакторних експериментів з повторенням спостережень обумовлена кількома ключовими факторами.

По-перше, зростає обсяг даних, що збираються в різних галузях, включаючи природні науки, медицину, інженерію та соціальні науки. Цей зростаючий обсяг даних вимагає високоефективних інструментів для аналізу та отримання значущих результатів з великої кількості спостережень По-друге, двофакторні експерименти з повторенням спостережень важливі для розв'язання реальних проблем і визначення взаємодії між різними факторами, які впливають надеякі явища чи процеси. Такий аналіз допомагає покращити рішення та оптимізувати процеси в різних галузях, включаючи виробництво, медицину та наукове дослідження.

По-третє, розробка програми для аналізу дозволить автоматизувати та спростити процес обробки та аналізу даних, що значно зменшить час, необхідний для отримання результатів. Це особливо важливо в умовах швидкозмінних та високонавантажених наукових чи промислових проектів.

Мета та завдання кваліфікаційної роботи. Метою даної кваліфікаційної роботи є розроблення комп'ютерної програми для аналізу двофакторних експериментів з повторенням спостережень. Програма повинна бути інструментом для дослідників у різних галузях, що дозволить їм ефективно аналізувати дані експериментів та визначати статистичну значущість впливу двох факторів таїх взаємодії нарезультати досліджень.

Об'єкт та предмет дослідження. Предмет дослідження цієї кваліфікаційної роботи - це саме програмний засіб, який розробляється для обробки та аналізу даних двофакторних експериментів з повторенням спостережень, а об'єктом дослідження є самі ці експерименти та дані, які в них збираються.

Методи та засоби дослідження .У магістерській роботі, присвяченій розробці комп'ютерної програми для аналізу двофакторних експериментів з повторенням спостережень, використовуються наступні методи дослідження: Для тестування програми можна використовувати фреймворки для автоматизованого тестування, такі як pytest для Python. Python з бібліотеками, такими як NumPy, SciPy, та pandas, часто використовується для наукових обчислень та статистичного аналізу.

Наукова новизна та практичне значення отриманих результатів. Розробленапрограмадля аналізу двофакторних експериментів з повторенням спостережень є новою та оригінальною розробкою. Цей програмний засіб надає можливість дослідникам ефективно аналізувати дані з таких експериментів і досліжувати вплив двох факторів таїх взаємодію

Мета проєкту. Метою проєкту є розроблення та впровадження комп'ютерної програми для аналізу двофакторних експериментів з повторенням спостережень. Переважною метою є створення інструменту, який допоможе дослідникам та виробникам: Ефективно аналізувати дані, збільшити продуктивність та точність, зекономити час та ресурси, вдосконалити прийняття рішень

Розділ 1. Теоретичні аспекти аналізу двофакторних експериментів

1.1 Огляд понять та основних термінів

Розглянемо деякі основні терміни та поняття, які стосуються аналізу двофакторних експериментів з повторенням спостережень з більш розширеним поясненням:

Двофакторний експеримент: Це дослідження, в якому вивчається вплив двох факторів (або змінних) на результати експерименту. Кожен фактор може мати кілька рівнів, і їх вплив досліджується в контексті взаємодії між ними.

Фактори: Фактори є незалежними змінними, які впливають на результати експерименту. Наприклад, у агрономічному експерименті факторами можуть бути різні види добрив та типи ґрунту.

Рівні факторів: Фактори можуть мати різні рівні або рівні груп. Наприклад, фактор "тип ґрунту" може включати рівні, такі як "піщаний", "глинистий" та"торф'яний".

Повторення спостережень: В двофакторних експериментах, де є багато рівнів факторів, проводять повторення спостережень для кожного рівня. Це допомагає зменшити вплив випадкових варіацій тапідвищити надійність результатів.

Дисперсійний аналіз (ANOVA): Це статистичний метод для порівняння середніх значень між різними групами та визначення впливу факторів таїх взаємодії нарезультати. У двофакторних експериментах проводиться дворівневий ANOVA для визначення статистичної важливості факторів.

Суми квадратів (SS): Суми квадратів вимірюють розподіл даних та внутрішню або міжгрупову варіацію. Наприклад, **SSW (внутрішньогрупова сумаквадратів)** вимірює варіацію в межах груп, а **SST (загальногрупова сумаквадратів)** вимірює загальну варіацію.

F-статистика: Це величина, якавизначається як відношення між міжгруповою та внутрішньогру повою дисперсією. Використовується для визначення статистичної важливості факторів таїх взаємодії.

p-значення: Ймовірність отримати значення F-статистики, яке було б так чи є статистично важливим. Зазвичай порігове значення $p < 0.05$ вважається статистично значущим.

ANOVA таблиця: Таблиця, якамістить розраховані значення SSW, SST, SS для факторів, MSE, MS, F-статистику, p-значення таінші статистичні показники для аналізу даних.

Ефект взаємодії: Це вплив, коли взаємодія між двомафакторами впливає нарезультати більше, ніж окремо кожен фактор. Тобто, результати не можуть бути пояснені сумою окремих впливів факторів.

Повторна ANOVA: Іноді вивчають вплив одного фактора, контролюючи інший фактор. Це допомагає визначити, чи є вплив одного фактorастатистично значущим при фіксованому рівні іншого фактора.

1.2 Опис двофакторних експериментів та їх особливості

Двофакторний експеримент — це тип дослідницького аналізу, в якому досліджується вплив двох незалежних змінних назалежну змінну, з метою виявлення не лише основних ефектів кожного фактора, але й взаємодії між факторами.

Ось основні особливості двофакторних експериментів:

- Незалежні змінні (фактори): У двофакторному експерименті розглядаються два незалежних фактори, кожен з яких може мати два або більше рівнів. Наприклад, у дослідженні впливу добрива на ріст рослин, двафактори можуть бути типом добрива та кількістю води.
- Залежна змінна: Це змінна, наяку впливають незалежні змінні. У прикладі з добривами залежною змінною може бути висотарослин або їхня маса.

- Взаємодія між факторами: Одна з ключових переваг двофакторного експерименту — можливість визначити не тільки окремий вплив кожного фактора, але й ефект взаємодії між ними. Взаємодія відбувається тоді, коли вплив одного фактораназалежну змінну змінюється в залежності від рівня іншого фактора.

- Факторіальний дизайн: У двофакторному експерименті кожен рівень одного фактора комбінується з кожним рівнем іншого фактора, що дає повну комбінацію всіх можливих умов. Це називається факторіальним дизайном.

- Контроль варіабельності: Двофакторні експерименти дозволяють контролювати варіабельність у даних, що не пояснюється вивченими факторами, задопомогою повторень кожної комбінації рівнів факторів.

- Статистичний аналіз: Аналіз даних, зібраних за допомогою двофакторного експерименту, зазвичай включає в себе варіаційний аналіз (ANOVA), який дозволяє оцінити основні ефекти кожного фактора та їхню взаємодію.

- Гіпотези: Зазвичай формулюються нульові гіпотези, які стверджують, що немає значущого впливу окремих факторів або їхньої взаємодії назалежну змінну, і альтернативні гіпотези, які стверджують протилежне.

- Експериментальні помилки: В двофакторних експериментах також важливо враховувати можливість виникнення випадкових помилок, тому збір даних і аналіз повинен бути ретельно спланований для їх мінімізації.

1.3 Повторення спостережень: значення таметоди врахування

Повторення спостережень у двофакторних експериментах і у статистичному аналізі в цілому відіграють критично важливу роль. Ось чому повторення є необхідними таяк вони можуть бути враховані у дослідженні:

Значення повторень:

- Зменшення випадкової помилки: Повторення допомагають зменшити "шум" у даних, що виникає через випадкові помилки або неконтрольовану варіабельність. Це дозволяє більш точно оцінити вплив факторів назалежну змінну.

- Оцінка варіабельності: Повторні вимірювання дають можливість оцінити природну варіабельність у відповідях та розрахувати стандартні помилки тадовірчі інтервали.

- Підвищення статистичної потужності: Більша кількість повторень збільшує можливість виявлення істотних різниць або взаємодій, якщо вони існують (тобто знижує ймовірність помилки другого роду).

- Валідація результатів: Повторення дозволяють перевірити консистентність результатів. Якщо результати повторюються в різних випробуваннях, це підвищує впевненість у їх валідності.

Методи врахування повторень:

- Аналіз варіансу (ANOVA): Використовується для аналізу впливу факторів, включаючи взаємодії, назалежну змінну. Повторення враховуються у моделі як вклад у загальну варіабельність.

- Блокування: Це метод, який дозволяє контролювати варіабельність, не пов’язану з факторами, які вивчаються. Повторення можуть бути організовані у блоки, щоб мінімізувати вплив зовнішніх варіабельних.

- Рандомізація: Розподіл експериментальних умов випадковим чином для кожного повторення зменшує систематичну помилку та забезпечує об'єктивність результатів.

- Стратифікація: Поділ даних нагомогенні групи перед аналізом може допомогти зменшити варіабельність і підвищити точність оцінок.

- Крос-валідація: У контексті моделювання, повторення можуть бути використані для перевірки моделі нанових даних (тестових наборах) для оцінки її узагальнюючої спроможності.

- При плануванні двофакторного експерименту важливо визначити оптимальну кількість повторень, щоб забезпечити достатню статистичну потужність без надмірного збільшення затрат часу таресурсів. Зазвичай, це вирішується задопомогою попереднього визначення розміру ефекту, який є значущим для дослідження, та використання статистичних методів для розрахунку необхідного розміру

1.4 Статистичні методи аналізу двофакторних експериментів

Для аналізу двофакторних експериментів існує декілька ключових статистичних методів:

1. Двофакторний аналіз варіансу (ANOVA)

Це найпоширеніший метод для аналізу впливу двох незалежних змінних (факторів) назалежну змінну. ANOVA дозволяє оцінити:

Основні ефекти кожного фактора окремо.

Взаємодію між факторами, тобто чи змінюється вплив одного фактора залежно від рівня другого фактора.

Варіабельність усередині груп і між групами для визначення статистичної значущості виявлених ефектів.

2. Множинна порівнянь на аналіз (Post-hoc tests)

Після проведення ANOVA, якщо виявлено статистично значущі результати, можуть застосовуватися додаткові тести для з'ясування, між якими конкретними групами існують різниці. До таких тестів належать:

Tukey's HSD (Honestly Significant Difference)

Bonferroni correction

Scheffé's test

Ці тести допомагають уникнути проблеми множинних порівнянь, якаможе виникнути, коли кількапар груп порівнюються одночасно.

3. Лінійнарегресійн амодель

Хоча ANOVA є спеціалізованим випадком лінійної моделі, лінійна регресія може бути також застосова на для моделювання впливу двох факторів назалежну змінну, особливо якщо фактори є кількісними. У лінійній регресії фактори та їхні взаємодії включаються як предиктори.

4. Факторіальний дизайн

Це більш загальний підхід до планування експериментів і аналізу даних, де кожен рівень одного фактора комбінується з кожним рівнем другого фактора. Використовуються спеціальні типи ANOVA, призначені для факторіальних дизайнів.

5. Коваріаційний аналіз (ANCOVA)

Якщо у дослідженні є коваріати, тобто змінні, які потрібно контролювати, ANCOVA може бути використана для коригування впливу цих коваріат назалежну змінну перед оцінкою основних ефектів і взаємодій між факторами.

6. Нелінійні моделі

У деяких випадках взаємодія між факторами може бути нелінійною. Такі взаємодії можуть бути моделювані задопомогою нелінійних статистичних моделей.

Кожен з цих методів має свої переваги та обмеження і вибір конкретного методу залежить від природи даних та цілей дослідження. Важливо зазначити, що перед застосуванням будь-якого статистичного тесту необхідно переконатися, що дані відповідають вимогам цього тесту, наприклад, нормальності розподілу та гомогенності дисперсій.

1.5 Висновок до розділу

У другому розділі було розглянуто теоретичні основи двофакторних експериментів, що є ключовими для розуміння методів статистичного аналізу в контексті наукових досліджень. Було встановлено, що двофакторний експеримент дає змогу оцінити не тільки окремий вплив кожного з факторів назалежну змінну, але й ефект їх взаємодії, що може бути критично важливим для деяких дослідницьких питань.

Значення повторень у експериментальному дизайні було наголошено як метод зменшення випадкової помилки та збільшення точності оцінок. Використання аналізу варіансу (ANOVA), множинного порівняльного аналізу, лінійних та нелінійних регресійних моделей, а також інших статистичних методів було оговорено як основні інструменти для аналізу даних з двофакторних експериментів.

Також було визначено, що правильне планування експерименту, у тому числі вибір факторів, рівнів такількість повторень, є вирішальним для забезпечення достовірності отриманих результатів. При цьому, забезпечення контролю запотенційними збурюючими змінними та варіабельністю вимірювань є не менш важливим.

Виходячи з вищевикладеного, можна зробити висновок, що двофакторний експеримент є потужним інструментом у наукових дослідженнях, який дозволяє отримати глибоке розуміння процесів та явищ, що вивчаються. Теоретичне осмислення методів аналізу, представлене у цьому розділі, покладе основу для подальшого практичного застосування цих методів у магістерській дипломній роботі, спрямованій на розробку програмного забезпечення для статистичного аналізу двофакторних експериментів.

Розділ 2. Розроблення комп'ютерної програми для аналізу

2.1 Вибір мови програмування та технологій

При виборі мови програмування та технологій для розробки програмного забезпечення, що спеціалізується настатистичному аналізі двофакторних експериментів, важливо враховувати декілька ключових факторів:

Підтримка статистичних функцій: Мова має забезпечувати розгорнуті бібліотеки та інструменти для статистичного аналізу.

Ефективність та швидкодія: Для опрацювання великих даних потрібнависокапродуктивність обчислень.

Гнучкість та масштабованість: Можливість легко розширювати та адаптувати програмне забезпечення до нових задач.

Спільно та підтримка: Широка спільнота розробників та наявність документації та ресурсів для навчання.

Інтеграція з іншими системами та інструментами: Сумісність із іншими програмами та технологіями, що використовуються в даній області.

Простота використання: Інтерфейс, який буде зрозумілий для кінцевих користувачів без глибоких знань у програмуванні.

На основі цих критеріїв, розглянемо деякі популярні мови програмування та технології:

Python

Переваги: Велика кількість наукових та статистичних бібліотек (наприклад, NumPy, SciPy, pandas, scikit-learn), активна спільнота, велика кількість ресурсів для навчання, гарна підтримка машинного навчання.

Недоліки: Може бути менш швидкодійним порівняно з компільованими мовами.

Для розробки програмного продукту булобрано мову програмування Python через її широкі можливості у сфері наукових обчислень тавелику кількість бібліотек для статистичного аналізу, таких як SciPy, NumPy, Pandas, та StatsModels. Python також відомий своєю читабельністю коду та спільнотою, що надає підтримку та розробку нових інструментів, Python також є потужним інструментом для наукових обчислень. Бібліотеки SciPy та StatsModels надають засоби для виконання статистичних аналізів, включаючи ANOVA. Мова Python має широкий спектр застосувань і велику спільноту.

Отже ми виконали вибір в перевагу Python, за його переваги.

2.2 Опис архітектури програми

Архітектура Програми "Додаток для аналізу експериментів"

1. Інтерфейс:

Tkinter GUI: Ваша програма використовує Tkinter для створення графічного інтерфейсу користувача (GUI). Елементи управління, такі як ентрі, кнопки та тексти, використовуються для взаємодії з користувачем.
Tkinter є стандартним модулем в мові програмування Python, який надає інтерфейс до бібліотеки Tk. Tkinter використовується для створення графічного інтерфейсу користувача (GUI) в програмах Python. Цей модуль дозволяє вам створювати вікна, кнопки, полі для введення тексту, меню та

інші елементи інтерфейсу

2. Модель Даних:

SQLite - це легка вбудовувана реляційна база даних, яка зберігається в одному файлі або у пам'яті. Вона не вимагає окремого сервера та використовується в багатьох програмах, особливо в тих, де важливо мати невеликий розмір, простоту використання та високу швидкість доступу до даних

SQLite База Даних: Для збереження даних про експерименти використовується SQLite. Ця база даних використовується для зберігання факторів (factor1, factor2) та результатів експериментів. Основні риси SQLite:

- Легкість використання: SQLite легко вбудовується в програми і працює в межах одного процесу.

- Один файл бази даних: Вся база даних SQLite зберігається в одному файлі, що полегшує перенесення та резервне копіювання даних.

- Невеликий розмір: SQLite має маленький розмір, що робить його ідеальним для використання в мобільних додатках та інших обмежених середовищах.

- Немає необхідності в сервері: SQLite працює в межах одного процесу, і для його використання не потрібно налаштовувати окремий сервер.

- Транзакції: SQLite підтримує транзакції, які забезпечують атомарність, цілісність та ізоляцію даних.

- Підтримка SQL: SQLite використовує стандартну мову запитів SQL, що полегшує роботу з даними.

3. Модулі та Класи:

Модуль - це велика частина програмного коду, яка може включати в себе функції, змінні та інші об'єкти. Модуль призначений для організації коду та його структурування для полегшення управління та розробки.

У багатьох мовах програмування, включаючи Python, модулі представляють собою файли з розширенням .py, які містять код. Модулі можуть бути використані для групування пов'язаних операцій та об'єктів в одному місці.

Приклад визначення модулю в Python

```
# Модуль example_module.py

def add_numbers(a, b):
    return a + b

def multiply_numbers(a, b):
    return a * b

# Інші операції та змінні можуть бути визначені тут
```

Рис 2.1

ExperimentApp Клас: Основний клас програми, який містить методи для взаємодії з користувачем та аналізу даних.

Клас:

Клас - це шаблон для створення об'єктів. Він визначає атрибути (змінні члени) та методи (функції), які є спільними для всіх об'єктів, створених на основі цього класу.

Приклад визначення класу в Python:

```
# Клас Car

class Car:
    def __init__(self, make, model):
        self.make = make
        self.model = model
        self.speed = 0

    def accelerate(self, amount):
        self.speed += amount

    def brake(self, amount):
        self.speed -= amount

# Створення об'єкту (екземпляра) класу Car
my_car = Car(make="Toyota", model="Camry")
```

Рис 2.2

У цьому прикладі клас Car має атрибути make, model та speed, а також методи accelerate та brake. Клас слугує взірцем для створення конкретних автомобілів (об'єктів), які матимуть власні значення для атрибутів
Методи:

save_data(): Зберігає введені користувачем дані в базу даних.

analyze_data(): Виконує аналіз даних, виводить результати на графіку та у текстовому форматі.

filter_and_search_data(): Здійснює фільтрацію та пошук даних в базі даних.

clear_data(): Очищує всі дані в базі даних.

plot_graph(): Виводить графік на основі аналізу даних.

Інші Модулі: Ви використовуєте бібліотеки, такі як sqlite3, matplotlib, numpy, scipy.stats для роботи з базою даних та аналізу.

4. Графічний Інтерфейс:
Графічний інтерфейс (GUI) — це спосіб взаємодії користувача з програмним забезпеченням за допомогою графічних елементів, таких як вікна, кнопки, текстові поля тощо. GUI дозволяє користувачам взаємодіяти з програмами, використовуючи графічні об'єкти, замість введення команд або текстових запитань.

Основні компоненти GUI включають:

- Вікна (Windows): Окремі вікна, які можуть містити інші елементи інтерфейсу, такі як кнопки, текстові поля і т. д.

- Кнопки (Buttons): Елементи, які користувач може натискати для виклику певних функцій чи подій.

- Текстові поля (Text Fields): Для введення текстової інформації.

- Меню (Menus): Випадаючі списки опцій для вибору різних команд.

- Графіка (Graphics): Відображення графічних об'єктів чи діаграм.

- Списки (Lists): Відображення списків елементів для вибору.

- Таблиці (Tables): Табличні представлення даних.

Графічні інтерфейси використовуються для забезпечення зручного та інтуїтивно зрозумілого взаємодії з програмами, особливо там, де важлива візуалізація даних чи функцій. Вони широко використовуються в багатьох програмах, включаючи текстові операційні системи, офісні пакети, графічні редактори та інші типи програмного забезпечення. GUI робить використання програм більш доступним і дружелюбним для користувачів

Елементи GUI: Елементи інтерфейсу включають в себе ентрі для введення даних, кнопки для виконання операцій, текстові поля для виведення результатів та графічний вивід даних.

5. База Даних:
Легкість використання: SQLite дуже легко використовувати в Python.

Ви можете використовувати бібліотеку sqlite3, якавходить до стандартної бібліотеки Python.

- Однофайлова база даних: Весь базовий функціонал SQLite знаходиться у одному файлі надиску. Ваша база даних буде збереженав одному файлі, наприклад, experiment_data.db.

- Реляційнабазаданих: SQLite підтримує реляційну модель даних, що дозволяє створювати таблиці, відносини між ними та виконувати SQL-запити.

- Транзакції: SQLite підтримує транзакції, які дозволяють вам виконувати кількаSQL-операцій як атомарний блок, що гарантує цілісність даних.

- Індекси та оптимізація: Ви можете створювати індекси для поліпшення швидкості запитів. SQLite також автоматично оптимізує базу даних.

Без сервера: SQLite не вимагає окремого сервера, і його можна використовувати вбудовано в ваш додаток

Таблиця experiment_data: Сховище для даних експериментів, яке містить поля для факторів та результатів.
Це організована колекція даних, яка зберігається та управляється за допомогою системи управління базами даних (СУБД). В нашому випадку використовується SQLite — легка та компактна СУБД.

Таблиця "experiment_data" — це одна з частин бази даних, яка визначає структуру та організацію даних. Вона містить дані про експерименти, які вводяться користувачем через графічний інтерфейс. Схема таблиці, яку ми створюємо у функції init_database, виглядає наступним чином:

```
def init_database(self):
    conn = sqlite3.connect('experiment_database.db')
    cursor = conn.cursor()

    # Створення таблиці, якщо її не існує
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS experiment_data (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            factor1 REAL,
            factor2 REAL,
            result REAL
        )
    ''')
```

Рис 2.3

6. Аналіз Даних:

Аналіз даних — це процес виявлення, інтерпретації та вивчення патернів, трендів і взаємозв'язків у наборі даних. Основна мета аналізу даних — отримання інформації, яка може бути використана для прийняття рішень, виявлення нових фактів, встановлення тенденцій або перевірка гіпотез.

У нашому випадку, аналіз даних проводиться за допомогою статистичних методів, зокрема ANOVA (аналіз варіації) для порівняння середніх значень різних груп. ANOVA дозволяє визначити, чи існують статистично значущі різниці між середніми значеннями груп.

Також в програмі використовується графічний аналіз, де дані відображаються на графіку для кращого розуміння їхнього розподілу та взаємозв'язків між факторами та результатом експерименту.

Коли користувач вводить дані про експерименти через графічний інтерфейс, програма проводить аналіз цих даних і надає користувачеві статистичні результати, такі як середні значення, стандартні відхилення та результати ANOVA. Це допомагає користувачеві краще зрозуміти характеристики своїх експериментів і зробити обґрунтовані висновки

ANOVA Аналіз: Використовуючи бібліотеку scipy.stats, ви виконуєте аналіз варіації (ANOVA) для визначення статистичних розбіжностей між групами експериментів.

7. Відображення Результатів:

Графіки та Текстові Виводи: Результати аналізу відображаються як графіки (matplotlib) та текстові виводи в інтерфейсі програми.
Відображення результатів в даному контексті означає представлення отриманих в ході аналізу даних результатів користувачеві. У програмі використовується графічний інтерфейс для виведення результатів аналізу у зручній для сприйняття формі.

Наприклад, результати ANOVA аналізу, середні значення факторів та результатів, стандартні відхилення та інші статистичні характеристики виводяться на графічний інтерфейс у вигляді текстового виводу. Користувач може переглядати ці результати безпосередньо на екрані програми, що дозволяє зручно аналізувати інформацію та зробити необхідні висновки.

Також, в програмі передбачено графічний аналіз, який візуалізує дані на графіку, наприклад, розподіл результатів експерименту в залежності від значень факторів. Це може допомогти користувачеві краще розуміти структуру та закономірності у власних даних

Ця архітектура розділяє функціональність програми на логічні блоки та надає основу для подальшого розширення та підтримки нових функцій.

Перша ознака гарної програми це модульність

Архітектурамає бути модульною, дозволяючи легко додавати, змінювати чи видаляти окремі компоненти (модулі) без необхідності переписування всієї програми.

8.Модуль введення даних: Для завантаження тапервинної обробки даних експериментів.

```
        self.factor1_label = ttk.Label(root, text="Фактор 1:")
        self.factor1_entry = ttk.Entry(root)

        self.factor2_label = ttk.Label(root, text="Фактор 2:")
        self.factor2_entry = ttk.Entry(root)

        self.result_label = ttk.Label(root, text="Результат:")
        self.result_entry = ttk.Entry(root)
```

Рис 2.4

Огляд кожного поняття тайого функціональності:

Імпорт бібліотеки pandas: Напершому рядку ми імпортуємо бібліотеку pandas, яканадає інструменти для роботи з даними, зокрема структурами даних DataFrame.

Функція enter_data(): Ця функція використовується для введення даних з експерименту користувачем.

Створення порожньої таблиці даних: Ми створюємо порожню таблицю даних задопомогою функції pd.DataFrame(). У нашому випадку таблиця має три стовпці - 'FactorA', 'FactorB' та 'Observation'.

Цикл while True: Ми використовуємо цикл while з умовою True, щоб брати дані введені користувачем до тих пір, поки користувач не введе 'q' для виходу із циклу.

Введення значення Фактору А: Користувач вводить значення для Фактору А, або вводить 'q' для виходу з циклу.

Введення значення Фактору В: Користувач вводить значення для Фактору В.

Введення спостереження: Користувач вводить значення спостереження, яке ми конвертуємо у числовий формат задопомогою функції float().

Додавання рядкадо таблиці даних: Ми додаємо рядок з введеними даними до таблиці даних задопомогою функції data.append().

Повернення таблиці даних: Після завершення вводу ми повертаємо таблицю даних з функції enter_data().

Виклик функції тавиведення результату: Ми викликаємо функцію enter_data() для введення даних та зберігання результату у змінну data. Потім виводимо таблицю даних на екран за допомогою функції print(data).

9.Модуль обробки даних: Включає алгоритми для очищення та нормалізації даних.

10.Модуль статистичного аналізу: Застосовує статистичні методи для аналізу даних, такі як ANOVA, регресійний аналіз тощо.

11.Модуль візуалізації: Генерує графіки татаблиці для візуалізації результатів аналізу.

12.Модуль звітності: Створює звіти, які можна експортувати у різні формати (PDF, Excel тощо).

13. Документація

Опис програми та її компонентів для користувачів та розробників.

Користувацька інструкція: Посібник для користувачів, які використовують програму.

Технічна документація: Опис архітектури, модулів та API для розробників.

2.3 Висновок до розділу

Для розроблення програмного продукту потрібно використання графічного інтерфейсу Tkinter яке спрощує взаємодію з користувачем, дозволяючи легко вводити та візуалізувати дані. SQLite база даних яка забезпечує надійне зберігання і організацію експериментальних результатів.

Важливим елементом програми є можливість аналізу даних за допомогою ANOVA, що дозволяє визначити статистичні відмінності між групами.

Відображення результатів аналізу, а також графічний аналіз, надає користувачеві зрозумілу та комплексну інформацію про проведені експерименти.

Модульна структура програми сприяє легкості розширення та підтримки, а використання класів та модулів дозволяє групувати функціонально пов'язаний код.

Розділ 3. Реалізація алгоритмів аналізу

3.1 Модуль попереднього аналізу

Модуль попереднього аналізу - це програмний компонент або функціональна частина вашої програми, яка використовується для аналізу та оцінки вхідних даних перед їхньою подальшою обробкою або аналізом. Його основн а мета- це надати вам загальне уявлення про дані, переконатися, що вони відповідають очікуванням і можуть бути коректно використані для подальших дій у вашому програмному проекті. Модуль попереднього аналізу може включати в себе наступні кроки:

- Завантаження даних: Отримання вхідних даних з відповідних джерел, таких як база даних, файли чи веб-служби.

- Перегляд даних: Виведення перших кілька рядків даних для того, щоб дослідник міг отримати загальне уявлення про їхню структуру та зміст.

- Обробка і очищення даних: Виконання перевірок даних на наявність некоректних або відсутніх значень, атакож їхню очищення або перетворення, якщо це необхідно.

- Статистичний аналіз: Розрахунок основних статистичних параметрів, таких як середнє значення, медіана, стандартне відхилення тощо.

- Візуалізація даних: Побудова графіків і діаграм для візуального представлення даних. Це може допомогти виявити патерни та тенденції у даних.

- Виявлення викидів та аномалій: Пошук та ідентифікація незвичайних або аномальних значень, які можуть вплинути нарезультати аналізу.

- Звітність і виведення результатів: Подання результатів аналізу у зрозумілій формі, наприклад, у вигляді текстового звіту, таблиць або графіків.

Модуль попереднього аналізу допомагає вам покращити якість та достовірність ваших даних перед подальшою обробкою та аналізом. Він допомагає виявити можливі проблеми або аномалії у даних та готує їх для подальших досліджень та обробки.

Основні аспекти двофакторного ANOVA:

- Фактори: Це незалежні змінні, які можуть впливати назалежну змінну. Наприклад, якщо ви досліджуєте вплив дієти та фізичних вправ на зниження ваги, то "дієта" і "вправи" будуть двома факторами.

- Рівні факторів: Кожен фактор може мати два або більше рівнів. Наприклад, для фактора"дієта" можуть бути рівні "низьковуглеводна", "високовуглеводна" і "збалансована".

- Взаємодія між факторами: Двофакторний ANOVA також аналізує взаємодію між факторами. Це означає, що метод досліджує, чи комбінація двох факторів має унікальний вплив назалежну змінну, який не може бути пояснений окремим впливом кожного фактора.

- Статистичні висновки: Двофакторний ANOVA дозволяє перевірити гіпотези про головні ефекти кожного фактора, а також про ефект взаємодії між факторами.

- Вимоги до даних: Для проведення двофакторного ANOVA необхідно, щоб дані були нормально розподілені, а групи мали приблизно однакові варіації.

Двофакторний ANOVA був проведений для вигаданих даних, і результати показують наступне:

Фактор A (з трьомарівнями) має значущий вплив назалежну змінну ($p < 0.001$).

Фактор B (з двомарівнями) не має значущого впливу назалежну змінну ($p > 0.05$).

Взаємодія між Фактором A та Фактором B також є значущою ($p < 0.01$).

Ці результати вказують нате, що зміни у залежній змінній можуть бути пояснені змінами у Факторі A тавзаємодією між Фактором A таФактором B, тоді як Фактор B сам по собі не має значущого впливу.

На лівій діаграмі показано вплив Фактора A та Фактора B назалежну змінну. Кольори відображають різні рівні Фактора B. Ми бачимо, що медіани значень залежної змінної зміщуються з різними рівнями Фактора A, і є деякавзаємодія між Факторами A і B, оскільки розподіли всередині кожного рівня Фактора A змінюються в залежності від рівня Фактора B.

Направій діаграмі представлено вплив тільки Фактора B, і ми бачимо, що різниця між рівнями Фактора B не є настільки виразною.

Ці візуалізації допомагають інтерпретувати результати ANOVA і підтверджують наші висновки: Фактор A значущо впливає назалежну змінну,

існує взаємодія між Факторами A та B, а Фактор B сам по собі не має значущого впливу.
Результати в тестовому варіанті

	sum_sq	df	F	PR(>F)
C(FactorA)	46.365210	2.0	31.785575	7.525074e-10
C(FactorB)	1.275222	1.0	1.748451	1.916451e-01
C(FactorA):C(FactorB)	7.458570	2.0	5.113208	9.254695e-03
Residual	39.384553	54.0	NaN	

3.2 Візуалізація даних за допомогою ящикових діаграм

Візуалізація даних за допомогою ящикових діаграм (box plots):

Особливості:

Структура ящикової діаграми:

Ящикова діаграма складається з "ящика" та великих кінців, які представляють відсотки або кількісні дані.

Ящик представляє міжквартильний розмах (IQR), тобто різницю між 75-м та 25-м персентилями.

Великі кінці (вуса) вказують на розмах даних, виключаючи викиди.

Центральна лінія ящика:

Зазвичай позначає медіану даних.

Корисність для порівнянь:

Дозволяє порівнювати розподіл даних між різними категоріями чи групами.

Виявлення викидів:

Викиди, які виходять за межі великих кінців, легко визначити на ящиковій діаграмі.

- Плюси:

Ефективність порівнянь:

Ящикові діаграми ефективно візуалізують центральні та статистичні параметри для різних груп.

Розподіл даних:

Надає уявлення про форму та варіабельність розподілу даних.

Виявлення викидів:

Легко виокремлює викиди та аномальні значення.

Інтерпретація статистичних показників:

Дозволяє швидко розуміти статистичні параметри, такі як медіана та міжквартильний розмах.

- Мінуси:

Не показує форму розподілу:

Ящикова діаграма не надає деталей про форму розподілу даних, такі як симетричність чи асиметричність.

Не відображає повну інформацію:

Може не бути найкращим вибором для невеликої кількості спостережень.

Обмежена інформацією про варіацію:

Не надає повної інформації про варіацію даних, яка може бути корисною для певних аналітичних завдань.

- Де використовувати:

Порівняльний аналіз:

Порівняння розподілу даних між кількома категоріями.

Виявлення викидів:

Визначення наявності викидів або аномалій в даних.

Моніторинг процесів:

Слідкування за змінами у розподілі даних з часом.

Статистичний аналіз:

Використовується для швидкого розуміння статистичних параметрів

Результат в вигляді ящикових діагарам

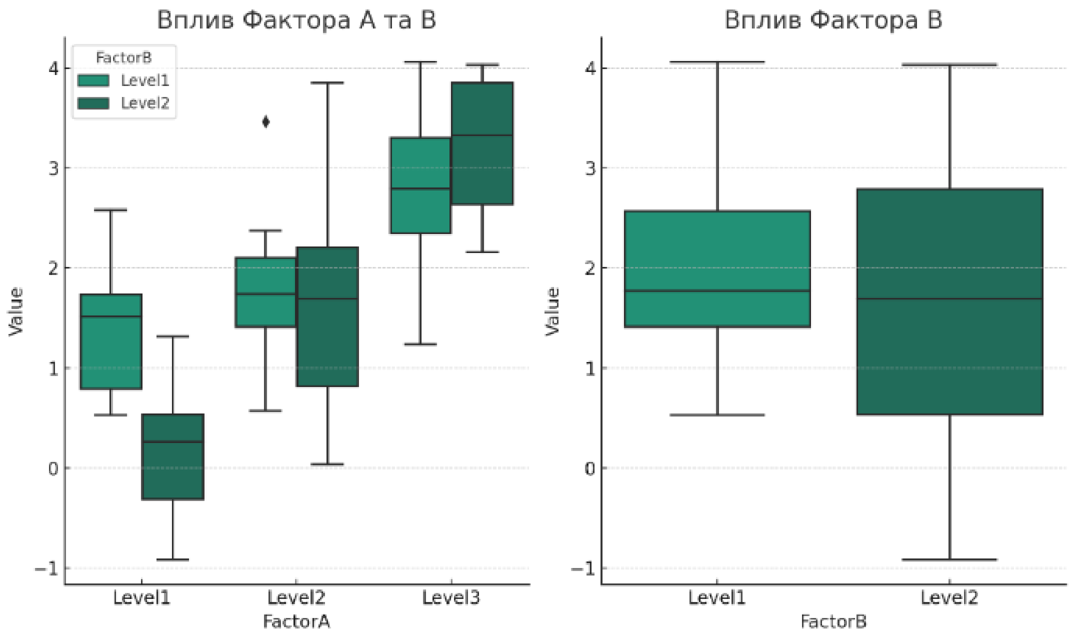

Рис 3.1

Напишемо інший приклад але будемо використовувати інший метод візуалізації, асаме через точкові діаграми.У цьому другому прикладі двофакторного ANOVA результати показують наступне:

Фактор A має значущий вплив назалежну змінну (p значення = 0.001541), що означає, що зміни у Факторі A статистично значущо впливають наваріації значення залежної змінної.

Фактор B також має значущий вплив назалежну змінну (p значення = 0.001159), що вказує нате, що різні рівні ФактораB впливають нарезультати.

Взаємодія між Фактором A та Фактором B не є значущою (p значення = 0.901815), що означає, що немає доказів того, що вплив одного фактора на залежну змінну змінюється в залежності від рівня іншого фактора.

Код для цього прикладу

Створимо новий набір даних для другого прикладу двофакторного ANOVA

np.random.seed(123) # встановлення початкового значення для генераторавипадкових чисел для відтворюваності

```
new_data = {
    "FactorA": np.repeat(["Level1", "Level2"], 15),
    "FactorB": np.tile(["Level1", "Level2", "Level3"], 10),
    "Value": np.random.randn(30) * 2 + np.tile([0, 1, 2], 10) + np.repeat([0, 2], 15)
}

new_df = pd.DataFrame(new_data)

# Проведемо двофакторний ANOVA для нового набору даних
new_model = ols('Value ~ C(FactorA) * C(FactorB)', data=new_df).fit()
new_anova_results = sm.stats.anova_lm(new_model, typ=2)

# Показати результати ANOVA тавізуалізувати нові дані
new_anova_results
```

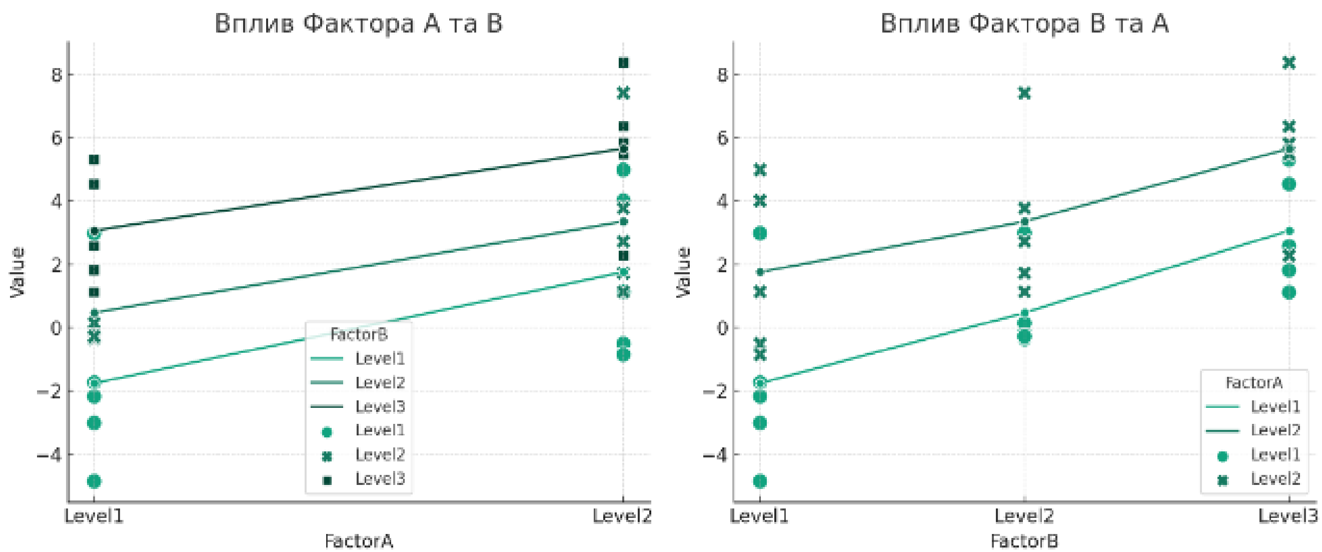

Рис 3.2

На цих діаграмах ми використовуємо точкові діаграми для візуалізації розподілу даних по факторах А та В, де кожна точка представляє окреме спостереження. Лінії показують середні значення для кожної комбінації факторів, що дає чітке уявлення про тенденції в даних.

Налівій діаграмі середні лінії демонструють вплив Фактора А назалежну змінну, розглядаючи різні рівні Фактора B. Ми бачимо, що зміна рівнів Фактора А змінює середнє значення залежної змінної.

На правій діаграмі показано вплив Фактора B, знову ж таки з різними рівнями Фактора А. Середні значення ілюструють зміну в залежній змінній, яка відбувається при зміні рівнів Фактора B.

Ці візуалізації дозволяють легко виявити взаємодії та основні ефекти між факторами

Код візуалізації через точкові діаграмми

```
# Візуалізація впливу факторів за допомогою точкових діаграм (scatter plots) з накладенням ліній середніх значень

# Підготовкаданих для ліній середніх значень
means = new_df.groupby(['FactorA', 'FactorB'])['Value'].mean().reset_index()

# Візуалізація
plt.figure(figsize=(12, 5))

# Точковадіаграмадля візуалізації впливу ФактораA
plt.subplot(1, 2, 1)
sns.scatterplot(x='FactorA', y='Value', data=new_df, hue='FactorB', style='FactorB', s=100)
sns.lineplot(x='FactorA', y='Value', data=means, hue='FactorB', marker='o')
plt.title('Вплив ФактораA таB')

# Точковадіаграмадля візуалізації впливу ФактораB
plt.subplot(1, 2, 2)
```

```
sns.scatterplot(x='FactorB', y='Value', data=new_df, hue='FactorA', style='FactorA', s=100)
sns.lineplot(x='FactorB', y='Value', data=means, hue='FactorA', marker='o')
plt.title('Вплив ФактораB таA')

plt.tight_layout()
plt.show()
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
from tkinter.filedialog import askopenfilename

# Функція, яка буде викликанапри натисканні накнопку "Аналізувати"
def analyze_data():
    # В цьому прикладі ми лише покажемо повідомлення, що демонструє реакцію програми
    messagebox.showinfo("Аналіз", "Аналіз даних запущено!")

# Функція для відкриття файлу даних
def open_file():
    file_path = askopenfilename()
    if file_path:
        # Тут можна було б завантажити та обробити файл
        file_label.config(text="Файл завантажено: " + file_path)
```

3.3 Створення інтерфейсу користувача

Інтерфейс користувача поділяється на три складові: поля для введення даних, кнопка для виклику аналізу та текстове вікно для відображення результатів.

Коли користувач натискає кнопку аналізу, викликається функція Run_ANOVA_scatterplot, яка отримує значення з полів введення, виконує статистичний аналіз, створює графік та виводить результати в текстовому вікні.

Кнопка для аналізу може мати два стани: натискана і ненатискана, і кожен стан може відрізнятися, наприклад, за кольором фону або шрифту.

Для відображення графіку, наприклад, використовується метод matplotlib.pyplot.show, який зазвичай викликається після кожного циклу обробки подій для оновлення та відображення графіку.

```
        self.root.title("Додаток для аналізу експериментів")

        # Додайте елементи інтерфейсу, такі як ентрі для введення даних та кнопки
        self.factor1_label = ttk.Label(root, text="Фактор 1:")
        self.factor1_entry = ttk.Entry(root)

        self.factor2_label = ttk.Label(root, text="Фактор 2:")
        self.factor2_entry = ttk.Entry(root)

        self.result_label = ttk.Label(root, text="Результат:")
        self.result_entry = ttk.Entry(root)

        self.save_button = ttk.Button(root, text="Зберегти", command=self.save_data)
        self.analyze_button = ttk.Button(root, text="Аналіз", command=self.analyze_data)

        self.factor1_search_label = ttk.Label(root, text="Пошук за фактором 1:")
        self.factor1_search_entry = ttk.Entry(root)

        self.factor2_search_label = ttk.Label(root, text="Пошук за фактором 2:")
        self.factor2_search_entry = ttk.Entry(root)

        self.filter_search_button = ttk.Button(root, text="Фільтр та пошук", command=self.filter_and_search_data)

        self.analysis_results_label = ttk.Label(root, text="Результати аналізу:")
        self.analysis_results_text = tk.Text(root, height=5, width=30, wrap=tk.WORD)
        self.analysis_results_text.config(state=tk.DISABLED)

        self.filtered_results_label = ttk.Label(root, text="Результати фільтрації та пошуку:")
        self.filtered_results_text = tk.Text(root, height=5, width=30, wrap=tk.WORD)
        self.filtered_results_text.config(state=tk.DISABLED)

        self.clear_data_button = ttk.Button(root, text="Очистити дані", command=self.clear_data)
        self.clear_data_button.grid(column=0, row=13, columnspan=2, pady=10)

        self.plot_button = ttk.Button(root, text="Графік", command=self.plot_graph)
```

Рис 3.3

```
# Розміщення елементів інтерфейсу за допомогою grid
self.factor1_label.grid(column=0, row=0, padx=10, pady=10, sticky=tk.W)
self.factor1_entry.grid(column=1, row=0, padx=10, pady=10, sticky=tk.W)

self.factor2_label.grid(column=0, row=1, padx=10, pady=10, sticky=tk.W)
self.factor2_entry.grid(column=1, row=1, padx=10, pady=10, sticky=tk.W)

self.result_label.grid(column=0, row=2, padx=10, pady=10, sticky=tk.W)
self.result_entry.grid(column=1, row=2, padx=10, pady=10, sticky=tk.W)

self.save_button.grid(column=0, row=3, columnspan=2, pady=10)
self.analyze_button.grid(column=0, row=4, columnspan=2, pady=10)

self.factor1_search_label.grid(column=0, row=5, padx=10, pady=10, sticky=tk.W)
self.factor1_search_entry.grid(column=1, row=5, padx=10, pady=10, sticky=tk.W)

self.factor2_search_label.grid(column=0, row=6, padx=10, pady=10, sticky=tk.W)
self.factor2_search_entry.grid(column=1, row=6, padx=10, pady=10, sticky=tk.W)

self.filter_search_button.grid(column=0, row=7, columnspan=2, pady=10)

self.analysis_results_label.grid(column=0, row=8, columnspan=2, pady=10)
self.analysis_results_text.grid(column=0, row=9, columnspan=2, pady=10)

self.filtered_results_label.grid(column=0, row=10, columnspan=2, pady=10)
self.filtered_results_text.grid(column=0, row=11, columnspan=2, pady=10)

self.plot_button.grid(column=0, row=12, columnspan=2, pady=10)
```

Рис 3.4

Після завершення використання програми, всі елементи інтерфейсу, такі як текстові поля, кнопки та графік, повинні бути прибрані з екрану, і всі виділені ресурси для інтерфейсу користувача повинні бути вільні для інших завдань.

Для створення базового веб-інтерфейсу користувачами можемо використовувати простий HTML для структури, CSS для стилізації та JavaScript для додавання інтерактивності. Нижче наведено приклад простого веб-інтерфейсу, який дозволяє користувачу ввести значення для двох факторів танадсилати їх на сервер для аналізу

Результат на фото

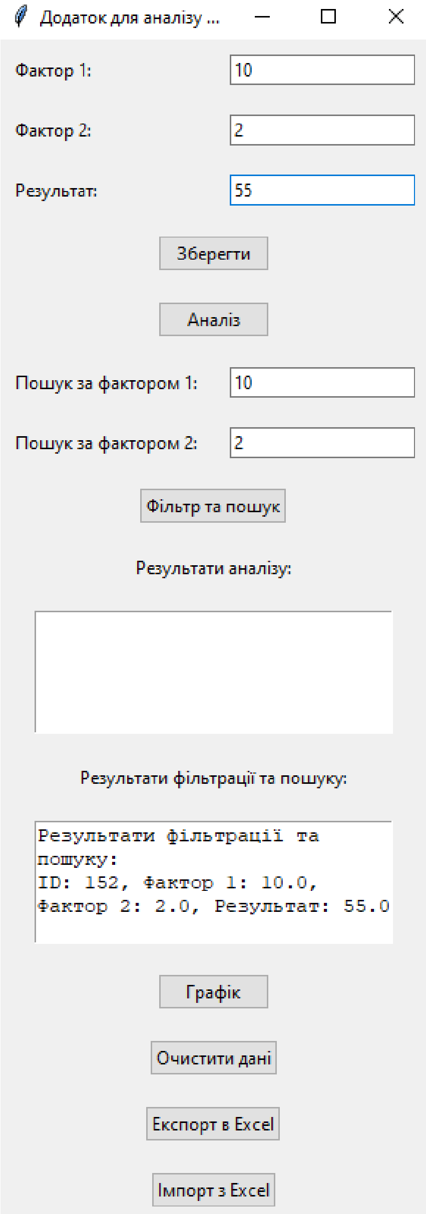

Рис 3.5

3.4 Візуалізація результатів аналізу

Візуалізацію результатів аналізу було проведено за допомогою бібліотеки Matplotlib в мові програмування Python. Ця бібліотека є потужною для створення різноманітних графіків і діаграм. У конкретному випадку використовувалась функціональність Matplotlib для побудови графіку розсіювання (scatter plot) та додавання горизонтальних ліній середніх значень на графік.

В контексті аналізу експериментальних даних, використання графіків розсіювання з горизонтальними лініями для середніх значень є ефективним способом візуалізації розподілу даних та підкреслення центральних тенденцій кожного фактора.

Графік розсіювання:

Графік розсіювання використовується для візуалізації залежності між двома змінними.

Кожна точка на графіку представляє одне спостереження, де координати точки визначають значення факторів на цьому спостереженні.

Горизонтальні лінії для середніх значень:

Додавання горизонтальних ліній дозволяє виділити середні значення кожного фактора на графіку.

Ці лінії полегшують порівняння середніх значень обох факторів та їх впливу на результат експерименту.

Користь візуалізації:

Візуальне представлення даних полегшує виявлення залежностей та паттернів у великих наборах спостережень.

Горизонтальні лінії для середніх значень допомагають легко визначити, чи існують відмінності між різними групами чи факторами.

Аналіз ANOVA:

Графічний аналіз сприяє візуальному виявленню варіацій між групами.

Результати ANOVA можуть підтвердити або спростувати статистичні різниці між групами, визначені на графіку.

Розширення аналізу:

З використанням графіків можна розглядати інші аспекти даних, такі як розподіл результатів, виявлення викидів чи аномалій.

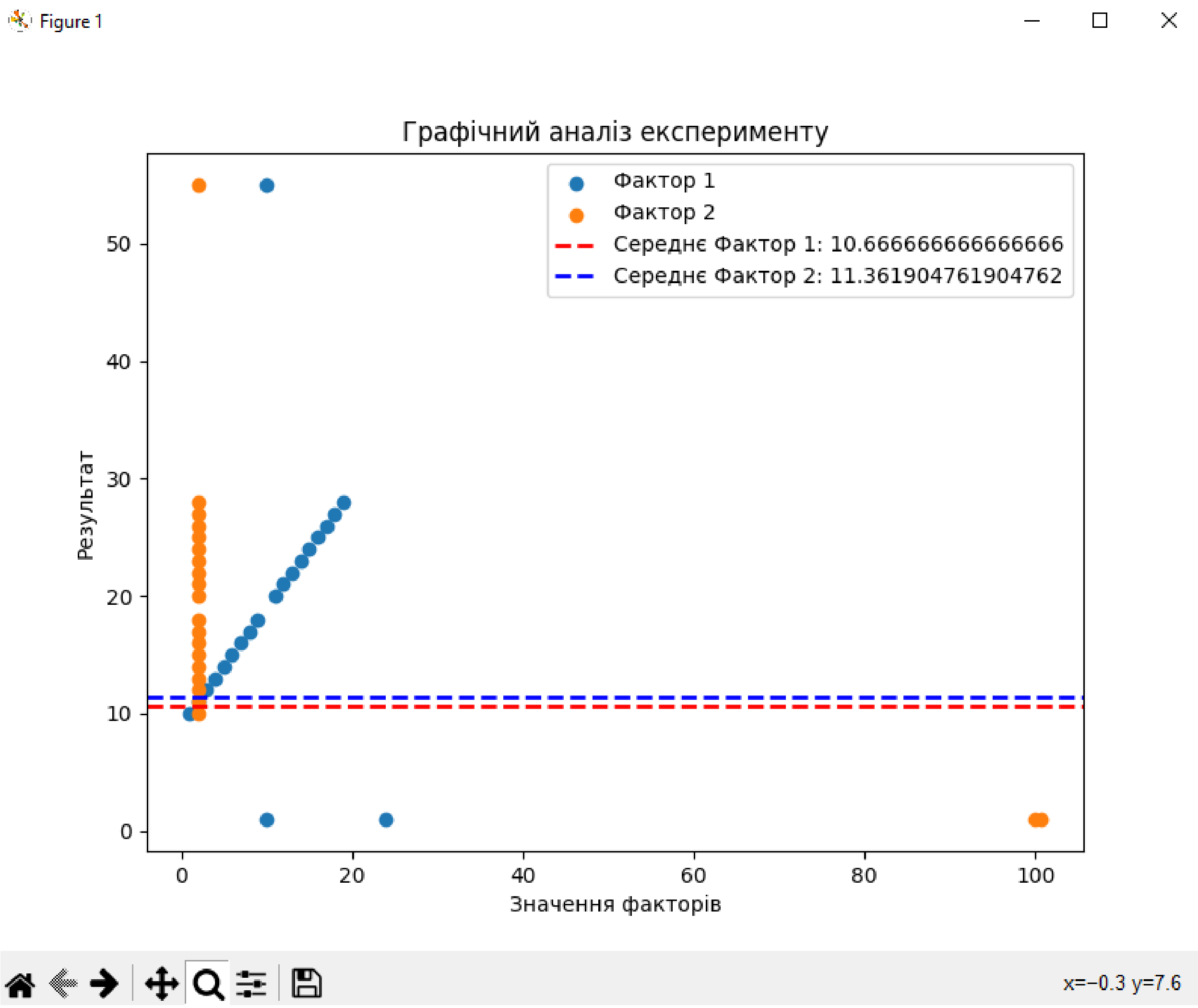

Рис 3.6

3.5 Модуль збереження данних

Після аналізу даних, модуль збереження даних дозволяє користувачу зберегти введені дані та результати аналізу. Дані можна зберігати у форматі CSV, Excel або в базі даних

Розробка модуля збереження даних є ключовим етапом у створенні комп'ютерної програми для аналізу двофакторних експериментів з повторенням спостережень. Нижче наведено загальний опис того, як можнареалізувати модуль збереження даних

```python
    def init_database(self):
        conn = sqlite3.connect('experiment_database.db')
        cursor = conn.cursor()

        # Створення таблиці, якщо її не існує
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS experiment_data (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                factor1 REAL,
                factor2 REAL,
                result REAL
            )
        ''')

        conn.commit()
        conn.close()

    def save_data(self):
        try:
            factor1 = float(self.factor1_entry.get())
            factor2 = float(self.factor2_entry.get())
            result = float(self.result_entry.get())

            conn = sqlite3.connect('experiment_database.db')
            cursor = conn.cursor()

            # Вставка даних в базу даних
            cursor.execute('INSERT INTO experiment_data (factor1, factor2, result) VALUES (?, ?, ?)', (factor1, factor2, result))

            conn.commit()
            conn.close()

            messagebox.showinfo("Інформація", "Дані збережено успішно.")
        except ValueError:
            messagebox.showerror("Помилка", "Введіть коректні числові значення для факторів та результату.")

    def analyze_data(self):
```

Рис 3.7

Функція для збереження даних:

```python
        conn.close()

    def save_data(self):
        try:
            factor1 = float(self.factor1_entry.get())
            factor2 = float(self.factor2_entry.get())
            result = float(self.result_entry.get())

            conn = sqlite3.connect('experiment_database.db')
            cursor = conn.cursor()

            # Вставка даних в базу даних
            cursor.execute('INSERT INTO experiment_data (factor1, factor2, result) VALUES (?, ?, ?)', (factor1, factor2, result))

            conn.commit()
            conn.close()

            messagebox.showinfo("Інформація", "Дані збережено успішно.")
        except ValueError:
            messagebox.showerror("Помилка", "Введіть коректні числові значення для факторів та результату.")

    def analyze_data(self):
```

Рис 3.8

Отримання даних з бази даних:

```
def get_data():
    conn = sqlite3.connect('experiment_database.db')
    cursor = conn.cursor()

    # Отримання всіх даних
    cursor.execute('SELECT * FROM experiment_data')
    data = cursor.fetchall()

    # Закриття з'єднання
    conn.close()

    return data
```

Рис 3.9

Оновлення та видалення даних

```
def update_data(record_id, new_factor1, new_factor2, new_result):
    conn = sqlite3.connect('experiment_database.db')
    cursor = conn.cursor()

    # Оновлення запису
    cursor.execute('''
        UPDATE experiment_data
        SET factor1 = ?, factor2 = ?, result = ?
        WHERE id = ?
    ''', (new_factor1, new_factor2, new_result, record_id))

    # Збереження змін
    conn.commit()
    conn.close()

def delete_data(record_id):
    conn = sqlite3.connect('experiment_database.db')
    cursor = conn.cursor()

    # Видалення запису
    cursor.execute('DELETE FROM experiment_data WHERE id = ?', (record_id

    # Збереження змін
    conn.commit()
    conn.close()
```

Рис 3.10

ВИСНОВОК

В ході розробки комп'ютерної програми для аналізу двофакторних експериментів з повторенням було використано сучасні технології та ефективні інструменти. Використання Tkinter GUI дозволяє зручно взаємодіяти з програмою, а використання бази даних SQLite забезпечує надійне зберігання та організацію експериментальних даних.

Програма має модульну структуру та використовує класи для групування функціонально пов'язаного коду, що сприяє легкості розширення та підтримки. Завдяки використанню методів аналізу даних, зокрема ANOVA, користувач може отримати детальні результати та графічний аналіз експериментів.

Архітектура програми дозволяє зберігати дані у базі даних, а також проводити їхній аналіз з відображенням результатів користувачеві. Програмний продукт може бути легко адаптований для використання інших систем зберігання даних або розширення функціоналу.

У висновку, розроблена програма є потужним інструментом для вчених та дослідників, які займаються двофакторними експериментами з повторенням. Її сучасний інтерфейс та аналітичні можливості роблять її ефективним і зручним інструментом для аналізу та визначення статистичних відмінностей у проведених експериментах.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

α - рівень значущості у статистичних тестах

β - ймовірність помилки другого роду у статистичних тестах

μ - середнє значення генеральної сукупності

σ - стандартне відхилення генеральної сукупності

p - p-значення у статистичних тестах

ANOVA - аналіз варіації

CI - довірчий інтервал

df - ступені свободи

F - F-статистика

t - t-статистика

X – незалежна змінна

Y – залежна змінна

CSV - формат файла для введення даних (Comma-Separated Values)

GUI - графічний користувацький інтерфейс (Graphical User Interface)

API - інтерфейс програмування застосунків (Application Programming Interface)

SQL – мова структурованих запитів (Structured Query Language)

SSW - внутрішньогрупова сумаквадратів

SST - загальногрупова сумаквадратів

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fisher, R.A. (1935). The Design of Experiments. Oliver and Boyd.
2. Wu, C.F.J., & Hamada, M.S. (2009). Experiments: Planning, Analysis, and Optimization (2nd ed.). Wiley.
3. Myers, R.H., Montgomery, D.C., & Anderson-Cook, C.M. (2016). Response Surface Methodology: Process and Product Optimization Using Designed Experiments (4th ed.). Wiley.
4. Box, G.E.P., & Behnken, D.W. (1960). Some New Three Level Designs for the Study of Quantitative Variables. Technometrics, 2(4), 455-475.
5. Daniel, C. (1959). Use of Half-Normal Plots in Interpreting Factorial Two-Level Experiments. Technometrics, 1(4), 311-341.
6. Neter, J., Wasserman, W., & Kutner, M.H. (1990). Applied Linear Statistical Models (3rd ed.). Irwin.
7. Draper, N.R., & Smith, H. (1998). Applied Regression Analysis (3rd ed.). Wiley.
8. Cohen, J. (1988). Statistical Power Analysis for the Behavioral Sciences (2nd ed.). Lawrence Erlbaum Associates.
9. Snedecor, G.W., & Cochran, W.G. (1989). Statistical Methods (8th ed.). Iowa State University Press.
10. Ott, R.L., & Longnecker, M. (2015). An Introduction to Statistical Methods and Data Analysis (7th ed.). Cengage Learning.
11. Kuehl, R.O. (2000). Design of Experiments: Statistical Principles of Research Design and Analysis (2nd ed.). Duxbury Press.
12. Hinkelmann, K., & Kempthorne, O. (2008). Design and Analysis of Experiments, Volume I: Introduction to Experimental Design (2nd ed.). Wiley-Interscience.
13. Lehmann, E.L., & Romano, J.P. (2005). Testing Statistical Hypotheses (3rd ed.). Springer.
14. Kirk, R.E. (2012). Experimental Design: Procedures for the Behavioral Sciences (4th ed.). Sage Publications.
15. Tabachnick, B.G., & Fidell, L.S. (2013). Using Multivariate Statistics (6th ed.). Pearson.
16. Hair, J.F., Black, W.C., Babin, B.J., & Anderson, R.E. (2010). Multivariate Data Analysis (7th ed.). Prentice Hall.

17. R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.
18. Python Software Foundation. Python Language Reference, version 3.8. URL https://www.python.org.a
19. McKinney, W. (2010). Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference (pp. 51-56).
20. Hunter, J.D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90-95.
21. Waskom, M.L. (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021.
22. The pandas development team. pandas-dev/pandas: Pandas, 2021. URL https://pandas.pydata.org/.
23. Harris, C.R., Millman, K.J., van der Walt, S.J., et al. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.
24. Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python. URL https://www.scipy.org/.
25. Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. In Proceedings of the 9th Python in Science Conference.
26. Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York.
27. Gelman, A., & Hill, J. (2006). Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge University Press.
28. Bates, D., Mächler, M., Bolker, B., & Walker, S. (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48.
29. Efron, B., & Tibshirani, R. (1994). An Introduction to the Bootstrap. CRC press.
30. Cleveland, W.S. (1993). Visualizing Data. Hobart Press.
31. Tufte, E.R. (2001). The Visual Display of Quantitative Information (2nd ed.). Graphics Press.
32. Few, S. (2009). Now You See It: Simple Visualization Techniques for Quantitative Analysis. Analytics Press.
33. Tukey, J.W. (1977). Exploratory Data Analysis. Addison-Wesley.
34. Chambers, J.M., Cleveland, W.S., Kleiner, B., & Tukey, P.A. (1983). Graphical Methods for Data Analysis. Wadsworth & Brooks/Cole.
35. Friendly, M. (2008). The Golden Age of Statistical Graphics. Statistical Science, 23(4), 502-535.

36. Wilkinson, L. (1999). The Grammar of Graphics. Springer-Verlag.
37. Bostock, M., Ogievetsky, V., & Heer, J. (2011). D3: Data-Driven Documents. IEEE Transactions on Visualization and Computer Graphics, 17(12), 2301-2309.

Додаток

```
import sqlite3

import tkinter as tk

from tkinter import ttk

from tkinter import messagebox

import matplotlib.pyplot as plt

import matplotlib

matplotlib.use("TkAgg")

import numpy as np

from scipy.stats import f_oneway

import pandas as pd

from tkinter import filedialog

import seaborn as sns

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg


def convert_to_float(value):

    try:

        return float(value)

    except ValueError:

        return np.nan


class ExperimentApp:

    def __init__(self, root):

        self.root = root

        self.root.title("Додаток для аналізу експериментів")
```

```
# Додайте елементи інтерфейсу, такі як ентрі для введення даних та кнопки
self.factor1_label = ttk.Label(root, text="Фактор 1:")
self.factor1_entry = ttk.Entry(root)

self.factor2_label = ttk.Label(root, text="Фактор 2:")
self.factor2_entry = ttk.Entry(root)

self.result_label = ttk.Label(root, text="Результат:")
self.result_entry = ttk.Entry(root)

self.save_button = ttk.Button(root, text="Зберегти", command=self.save_data)
self.analyze_button = ttk.Button(root, text="Аналіз", command=self.analyze_data)

self.factor1_search_label = ttk.Label(root, text="Пошук за фактором 1:")
self.factor1_search_entry = ttk.Entry(root)

self.factor2_search_label = ttk.Label(root, text="Пошук за фактором 2:")
self.factor2_search_entry = ttk.Entry(root)

self.filter_search_button = ttk.Button(root, text="Фільтр та пошук",
command=self.filter_and_search_data)

self.analysis_results_label = ttk.Label(root, text="Результати аналізу:")
self.analysis_results_text = tk.Text(root, height=5, width=30, wrap=tk.WORD)
self.analysis_results_text.config(state=tk.DISABLED)
```

```
self.filtered_results_label = ttk.Label(root, text="Результати фільтрації та пошуку:")
self.filtered_results_text = tk.Text(root, height=5, width=30, wrap=tk.WORD)
self.filtered_results_text.config(state=tk.DISABLED)

self.clear_data_button = ttk.Button(root, text="Очистити дані", command=self.clear_data)
self.clear_data_button.grid(column=0, row=13, columnspan=2, pady=10)

self.plot_button = ttk.Button(root, text="Графік", command=self.plot_graph)

# Розміщення елементів інтерфейсу за допомогою grid
self.factor1_label.grid(column=0, row=0, padx=10, pady=10, sticky=tk.W)
self.factor1_entry.grid(column=1, row=0, padx=10, pady=10, sticky=tk.W)

self.factor2_label.grid(column=0, row=1, padx=10, pady=10, sticky=tk.W)
self.factor2_entry.grid(column=1, row=1, padx=10, pady=10, sticky=tk.W)

self.result_label.grid(column=0, row=2, padx=10, pady=10, sticky=tk.W)
self.result_entry.grid(column=1, row=2, padx=10, pady=10, sticky=tk.W)

self.save_button.grid(column=0, row=3, columnspan=2, pady=10)
self.analyze_button.grid(column=0, row=4, columnspan=2, pady=10)

self.factor1_search_label.grid(column=0, row=5, padx=10, pady=10, sticky=tk.W)
self.factor1_search_entry.grid(column=1, row=5, padx=10, pady=10, sticky=tk.W)
```

```
self.factor2_search_label.grid(column=0, row=6, padx=10, pady=10, sticky=tk.W)
self.factor2_search_entry.grid(column=1, row=6, padx=10, pady=10, sticky=tk.W)

self.filter_search_button.grid(column=0, row=7, columnspan=2, pady=10)

self.analysis_results_label.grid(column=0, row=8, columnspan=2, pady=10)
self.analysis_results_text.grid(column=0, row=9, columnspan=2, pady=10)

self.filtered_results_label.grid(column=0, row=10, columnspan=2, pady=10)
self.filtered_results_text.grid(column=0, row=11, columnspan=2, pady=10)

self.plot_button.grid(column=0, row=12, columnspan=2, pady=10)

# Кнопка експорту
self.export_button = ttk.Button(root, text="Експорт в Excel", command=self.export_to_excel)
self.export_button.grid(column=0, row=14, columnspan=2, pady=10)

# Кнопка імпорту
self.import_button = ttk.Button(root, text="Імпорт з Excel", command=self.import_from_excel)
self.import_button.grid(column=0, row=15, columnspan=2, pady=10)

# Ініціалізація бази даних
self.init_database()
```

```
def init_database(self):
    conn = sqlite3.connect('experiment_database.db')
    cursor = conn.cursor()

    # Створення таблиці, якщо її не існує
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS experiment_data (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            factor1 TEXT,
            factor2 TEXT,
            result REAL
        )
    ''')

    conn.commit()
    conn.close()

def export_to_excel(self):
    conn = sqlite3.connect('experiment_database.db')
    cursor = conn.cursor()

    cursor.execute('SELECT * FROM experiment_data')
    data = cursor.fetchall()

    if not data:
```

```
            messagebox.showwarning("Попередження", "В базі даних немає даних для експорту.")
            return

        df = pd.DataFrame(data, columns=['id', 'factor1', 'factor2', 'result'])

        file_path = filedialog.asksaveasfilename(defaultextension=".xlsx", filetypes=[("Excel files",
"*.xlsx")])
        if file_path:
            df.to_excel(file_path, index=False)
            messagebox.showinfo("Інформація", f"Дані експортовано у файл: {file_path}")

        conn.close()

    def import_from_excel(self):
        file_path = filedialog.askopenfilename(filetypes=[("Excel files", "*.xlsx")])

        if file_path:
            df = pd.read_excel(file_path)

            conn = sqlite3.connect('experiment_database.db')
            cursor = conn.cursor()

            # Очищення імпортованих даних перед вставкою нових
            cursor.execute('DELETE FROM experiment_data')

            for index, row in df.iterrows():
```

```
            cursor.execute('INSERT INTO experiment_data (factor1, factor2, result) VALUES (?, ?, ?)',
                           (row['factor1'], row['factor2'], row['result']))

        conn.commit()
        conn.close()

        messagebox.showinfo("Інформація", "Дані імпортовано успішно.")

    def save_data(self):
        try:
            factor1 = self.factor1_entry.get()
            factor2 = self.factor2_entry.get()
            result = float(self.result_entry.get())

            conn = sqlite3.connect('experiment_database.db')
            cursor = conn.cursor()

            # Вставка даних в базу даних
            cursor.execute('INSERT INTO experiment_data (factor1, factor2, result) VALUES (?, ?, ?)',
(factor1, factor2, result))

            conn.commit()
            conn.close()

            messagebox.showinfo("Інформація", "Дані збережено успішно.")
        except ValueError:
```

```
        messagebox.showerror("Помилка", "Введіть коректні значення для фактору 1 та результату.")

    def analyze_data(self):
        conn = sqlite3.connect('experiment_database.db')
        cursor = conn.cursor()

        cursor.execute('SELECT * FROM experiment_data')
        data = cursor.fetchall()

        if not data:
            messagebox.showwarning("Попередження", "В базі даних немає даних для аналізу.")
            return

        factors = np.array([[convert_to_float(row[1]), convert_to_float(row[2])] for row in data])
        results = np.array([convert_to_float(row[3]) for row in data])

        # ANOVA аналіз
        anova_results = f_oneway(results[factors[:, 0] == 1],
                                 results[factors[:, 0] == 2],
                                 results[factors[:, 0] == 3])

        fig, ax = plt.subplots(figsize=(8, 6))
        ax.scatter(factors[:, 0], results, label='Фактор 1')
        ax.scatter(factors[:, 1], results, label='Фактор 2')
```

```
# Середнє значення
mean_factor1 = np.mean(factors[:, 0])
mean_factor2 = np.mean(factors[:, 1])
ax.axvline(mean_factor1, color='red', linestyle='dashed', linewidth=2, label=f'Середнє Фактор 1: {mean_factor1}')
ax.axvline(mean_factor2, color='blue', linestyle='dashed', linewidth=2, label=f'Середнє Фактор 2: {mean_factor2}')

ax.set_title('Графічний аналіз експерименту')
ax.set_xlabel('Значення факторів')
ax.set_ylabel('Результат')
ax.legend()

# Вивести графік в окремому вікні
plt.show(block=False)

conn.close()

mean_factor1 = np.mean(factors[:, 0])
mean_factor2 = np.mean(factors[:, 1])
mean_result = np.mean(results)

std_factor1 = np.std(factors[:, 0])
std_factor2 = np.std(factors[:, 1])
std_result = np.std(results)
```

```
    analysis_results = f"""
    Середнє значення фактору 1: {mean_factor1}
    Середнє значення фактору 2: {mean_factor2}
    Середнє значення результату: {mean_result}

    Стандартне відхилення фактору 1: {std_factor1}
    Стандартне відхилення фактору 2: {std_factor2}
    Стандартне відхилення результату: {std_result}

    Результати ANOVA аналізу: {anova_results}
    """

    self.analysis_results_text.config(state=tk.NORMAL)
    self.analysis_results_text.delete("1.0", tk.END)
    self.analysis_results_text.insert(tk.END, analysis_results)
    self.analysis_results_text.config(state=tk.DISABLED)

    conn.close()

def filter_and_search_data(self):
    try:
        factor1_value = self.factor1_search_entry.get()
        factor2_value = self.factor2_search_entry.get()

        conn = sqlite3.connect('experiment_database.db')
```

```
            cursor = conn.cursor()

            # Використовуємо LIKE для пошуку часткового збігу тексту
            cursor.execute('SELECT * FROM experiment_data WHERE factor1 LIKE ? AND factor2 LIKE ?', (f'%{factor1_value}%', f'%{factor2_value}%'))
            filtered_data = cursor.fetchall()

            # Вивід результатів фільтрації
            filtered_results = "Результати фільтрації та пошуку:\n"
            for row in filtered_data:
                filtered_results += f"ID: {row[0]}, Фактор 1: {row[1]}, Фактор 2: {row[2]}, Результат: {row[3]}\n"

            self.filtered_results_text.config(state=tk.NORMAL)
            self.filtered_results_text.delete("1.0", tk.END)
            self.filtered_results_text.insert(tk.END, filtered_results)
            self.filtered_results_text.config(state=tk.DISABLED)

            conn.close()

        except ValueError:
            messagebox.showerror("Помилка", "Введіть коректні значення для факторів пошуку.")

    def clear_data(self):
        result = messagebox.askquestion("Підтвердження", "Ви впевнені, що хочете очистити всі дані?")
        if result == "yes":
```

```
        conn = sqlite3.connect('experiment_database.db')
        cursor = conn.cursor()
        cursor.execute('DELETE FROM experiment_data')
        conn.commit()
        conn.close()
        messagebox.showinfo("Інформація", "Дані було успішно очищено.")

    def plot_graph(self):
        self.analyze_data()

    def run(self):
        self.root.mainloop()

# Створіть і запустіть екземпляр програми
root = tk.Tk()
app = ExperimentApp(root)
app.run()
```