МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Створення універсальної веб-платформи для прикладного статистичного аналізу даних**

Виконав: студент 4 курсу, групи 6.1210-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
Д.Є.Болейко
(ініціали та прізвище)
Керівник доцент, к.т.н., доцент
О.М.Міхайлуца
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалавський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Програмне забезпечення систем_____
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Тетяна КРИТСЬКА
"_01_" __березня_____2024_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

_____Болейку Данилу Євгеновичу_____
(прізвище, ім'я, по батькові)

1.Тема роботи: **Створення універсальної веб-платформи для прикладного статистичного аналізу даних**

керівник роботи ___Міхайлуца Олена Миколаївна, к.т.н, доцент_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від ___26.12.2023 № 2215-с_____

2. Строк подання студентом кваліфікаційної роботи_____13.06.2024_____
3. Вихідні дані кваліфікаційної роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми аналізу даних;
- створення програмного продукту та його опис;
- тестування програмної системи та розробка висновків.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	01.03-08.03.24	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	09.03-10.03.24	виконано
3	Аналіз існуючих методів рішення	11.03-13.03.24	виконано
4	Дослідження засобів реалізації серверної частини	14.03-18.03.24	виконано
5	Дослідження засобів реалізації клієнтської частини	19.03-20.03.24	виконано
6	Узгодження подальших дій з науковим керівником	21.03-22.03.24	виконано
7	Програмна реалізація серверної частини	23.03-02.04.24	виконано
8	Програмна реалізація алгоритмів аналізу даних	03.04-12.04.24	виконано
9	Представлення отриманих результатів науковому керівнику та узгодження плану подальшого дослідження	13.04-15.04.24	виконано
10	Реалізація користувацького інтерфейсу	16.04-18.04.24	виконано
11	Написання тестів	19.04-20.04.24	виконано
12	Перевірка працездатності проєкту	21.04-01.06.24	виконано
13	Оформлення звіту	02.06-13.06.24	виконано

Студент _____ Д.Є. Болейко
(підпис) (прізвище та ініціали)

Керівник роботи _____ О.М. Міхайлуца
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок: 86

Рисунків: 21

Джерел: 20

Болейко Д.Є. Створення універсальної веб-платформи для прикладного статистичного аналізу даних: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник О.М. Міхайлуца. Запоріжжя: ЗНУ, 2024.

Мета і завдання роботи полягають у розробці універсальної веб-платформи для прикладного статистичного аналізу даних. Для цього необхідно було вивчити сучасні методи і засоби аналізу даних, розробити архітектуру платформи, що забезпечує ефективну обробку великих обсягів даних, інтегрувати алгоритми прогнозування на основі вхідних даних, а також створити зручний та інтуїтивний користувацький інтерфейс. Крім того, було необхідно протестувати платформу на реальних наборах даних для верифікації її функціональних можливостей та точності результатів.

У ході роботи була створена універсальна веб-платформа, яка дозволяє користувачам здійснювати прикладний статистичний аналіз даних. Платформа надає інструменти для зчитування, обробки, аналізу, та візуалізації даних. Основними функціональними можливостями розробленого застосунку є використання алгоритмів прогнозування, таких як лінійна та поліноміальна регресії, відображення результатів регресії у вигляді графіку та відображення завантажених датасетів у вигляді таблиці та графіків. Результати тестування показали високу точність і ефективність розробленої платформи.

Ключові слова: *веб-платформа, статистичний аналіз, обробка даних, прогнозування, користувацький інтерфейс*.

SUMMARY

Pages: 86

Drawings: 21

Source: 20

Boleyko D.E. Creation of Universal Web Platform for Applied Statistical Data Analysis: qualification work of the bachelor of specialty 121 "Software engineering" / Science. Head O.M. Mihailutsa Zaporizhzhia: ZNU, 2023.

The purpose of the work is to develop a universal web platform for applied statistical data analysis. To achieve this, it was necessary to study modern methods and tools for data analysis, design a platform architecture that ensures efficient processing of large data volumes, integrate forecasting algorithms based on input data, and create a user-friendly and intuitive interface. Additionally, it was essential to test the platform on real datasets to verify its functional capabilities and accuracy of results.

During the work, a universal web platform was created that allows users to perform applied statistical data analysis. The platform provides tools for reading, processing, analyzing, and visualizing data. The main functional features of the developed application include the use of forecasting algorithms such as linear and polynomial regression, displaying regression results in the form of charts, and displaying uploaded datasets in tables and charts. The test results demonstrated the high accuracy and efficiency of the developed platform.

Keywords: *web platform, statistical analysis, data processing, forecasting, user interface*.

ЗМІСТ

ВСТУП

Актуальність теми

Безперечно, аналіз даних є актуальною темою у сучасному світі, оскільки дані стали фундаментом для прийняття рішень у бізнесі, науці, урядових інституціях, повсякденному житті тощо. Ми живемо в епоху, коли кількість та доступність даних ростуть експоненціально, що відкриває нові можливості для розуміння складних процесів, прогнозування майбутніх трендів та оптимізації рішень.

Прогрес у сфері технологій розширив рамки аналізу даних, надаючи можливість для глибшого занурення в складність та різноманітність інформації навколо нас. Використання даних для розробки персоналізованих медичних лікувань, створення ефективніших міських планувань, оптимізація ланцюгів поставок, та підвищення рівня освіти та безпеки в суспільстві є лише декількома прикладами, як аналіз даних, підсилюваний останніми технологічними досягненнями, відкриває нові горизонти можливостей.

Вплив великих даних (Big Data) на сферу аналізу даних є на рівні значним, вносячи революційні зміни в обсяги доступної інформації, а також у методи та інструменти її обробки. Цей напрямок змінив традиційні підходи до аналізу, надаючи можливість здійснювати глибокий аналіз величезних масивів даних. Це відкриває двері для прогресивного використання даних у прийнятті рішень, дозволяючи організаціям і установам не тільки виявляти актуальні тренди та закономірності але й прогнозувати майбутні явища з високою точністю.

Аналіз даних не тільки сприяє економічному зростанню та ефективності в бізнес-середовищі, але й має значний соціальний вплив, покращуючи якість життя людей через розвиток сфер охорони здоров’я, освіти, міського планування та багатьох інших. Навіть у побуті, аналіз даних стає невід’ємною частиною нашого життя. Від персоналізованих рекомендацій у соціальних

мережах та платформах стрімінгу до інтелектуальних домашніх систем, які оптимізують споживання енергії та покращують безпеку.

Враховуючи всі ці аспекти, актуальність розробки інструментів для аналізу даних стає не тільки актуальною, але й необхідною задачею, спрямованою на задовільнення зростаючих потреб у обробці та аналізі даних.

Апробація результатів

Результати роботи було представлено на XVII університетській науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених «Молода наука-2024» [21].

Мета і завдання дослідження

Метою даного дослідження є розробка веб-додатку для статистичного аналізу часових рядів, що дозволить користувачам ефективно обробляти та аналізувати датасети з часовими відмітками. Завдання включають інтеграцію алгоритмів прогнозування для виявлення тенденцій та закономірностей у даних, а також розробку інтуїтивно зрозумілого інтерфейсу, який спрощує взаємодію з комплексними статистичними інструментами. Кінцевою метою є надання користувачам потужного інструменту для підтримки прийняття обґрунтованих рішень на основі аналізу часових рядів.

Глосарій

Інтерпретація даних — це пояснення отриманих даних після або під час аналізу, враховуючи контекст, в якому ці дані були зібрані.

Big Data — це напрямок аналізу, зберігання та обробки великих об’ємів даних, з урахуванням оптимальної швидкості і безперервної роботи.

Інсайти — це висновки, отримані з аналізу даних.

Регресійний аналіз — це аналіз, який дозволяє оцінити взаємозв’язок між залежною та однією або кількома незалежними змінними, прогнозуючи значення одних змінних на основі інших.

1 ДОСЛІДЖЕННЯ ПРОБЛЕМИ АНАЛІЗУ ДАНИХ

1.1 Огляд проблеми аналізу даних

Незважаючи на значення аналізу даних, як такого, в сучасному світі, сам процес аналізу супроводжується цілим рядом викликів та проблем, які потребують детального розгляду та вирішення. Ці проблеми варіюються від збору і зберігання даних до їх детального аналізу та інтерпретації. Детальний огляд цих проблем допоможе краще зрозуміти стратегію їх вирішення.

У сучасному світі кожну секунду генерується величезний обсяг даних. Це роблять різні боти, електроні пристрої, соціальні мережі та багато інших джерел. Вони збирають та накопичують всю зібрану інформацію для подальшого її аналізу. Основна проблема тут полягає не лише у фізичному зберіганні цієї інформації, а й у швидкій обробці гігантських об’ємів цих даних. У цьому контексті слова Джона Нейсбітта з його книги 1982 року «Megatrends» [1], «Drowning in Information But Starved for Knowledge», набувають особливої актуальності, відображаючи глибину проблеми, з якою ми стикаємося.

Наступна проблема полягає в різноманітному форматі даних. Від текстових файлів і таблиць до відео і зображень — всі ці формати вимагають окремих, спеціалізованих підходів до обробки та аналізу. Врахування цього аспекту є одним з основних викликів у сфері аналізу даних. Розділяють три категорії форматів даних:

- *Структуровані дані* — це дані які мають чітку структуру та зберігаються у вигляді таблиць. Гарним прикладом є реляційні бази даних. Коли дані організовані у рядки та стовпці ними дуже легко оперувати, тому вони гарно підходять для аналізу.
- *Неструктуровані дані* — це дані які не мають чіткої структури і можуть включати різноманітні текстові документи, відео, зображення, аудіозаписи тощо. Такі дані складно аналізувати, оскільки для їх

обробки необхідні спеціальні інструменти зчитування, розпізнавання, класифікації та аналізу інформації різних типів.

- *Напівструктуровані дані* — це дані які не мають чіткої структури але мають якісь метадані для організації та ідентифікації контенту. Прикладами можуть бути XML та JSON файли, де дані представлені у ієрархічній структурі або пар ключ-значення.

Інформація знаходиться повсюди і немає універсального обладнання для збирання інформації будь-якого типу та походження. Це створює необхідність у розробці та використанні різноманітних джерел для збору даних. Розглянемо детальніше основні категорії джерел даних:

- *Електронні пристрої та сенсори*. Сучасні технології, зокрема інтернет речей (IoT), збирають дані в реальному часі, використовуючи різноманітні сенсори. Вони можуть вимірювати температуру, тиск, вологість, рух та багато інших параметрів.
- *Соціальні медіа*. Платформи соціальних медіа є величезним джерелом неструктурованих даних, які включають в себе тексти, зображення, відео та інші. Ці дані відображають людські думки, тренди, переваги та багато іншої цінної інформації, яка може бути використана для аналізу поведінки та вподобань. Саме на цьому базуються сучасні системи рекомендацій.
- *Публічні бази даних та відкриті дані*. Різні організації та інституції часто надають доступ до величезної кількості даних через публічні бази даних. Такі дані можуть включати наукові дослідження, економічні звіти та інше.
- *Внутрішні системи організацій*. Компанії збирають великі обсяги даних у своїх внутрішніх системах інформацію про продажі, клієнтів, операцій, фінанси та багато іншого, що може бути аналізовано для підвищення ефективності бізнесу та прибутку компанії.

Кожне з цих джерел даних вимагає специфічних методів збору та аналізу. Розуміння особливостей кожного джерела та вміння їх використовувати є ключовим для успішного аналізу даних.

Складність інструментів для аналізу даних також може стати перепоною. Ця проблема торкається не тільки початківців в галузі аналізу даних, а й звичайних людей, які мають власний бізнес, керівників проєктів, та інших професіоналів, що прагнуть використовувати дані та їх аналіз для підвищення ефективності своєї діяльності. Сучасні інструменти аналізу даних відкривають широкі можливості для збирання, аналізу та візуалізації даних, але складність опанувати такі інструменти може бути великим бар'єром для тих, хто не має спеціалізованої підготовки.

Останньою проблемою є забезпечення безпеки та конфіденційності даних. У світі, де інформація має величезну цінність, захист даних від несанкціонованого доступу та їх неправомірне використання є критично важливим.

Разом, ці виклики формують складне поле діяльності для фахівців у сфері аналізу даних. Всі описані проблеми вимагають комплексного підходу до їх вирішення. Це стосується розробки нових технологій, методів аналізу, а також підготовки фахівців, які зможуть ефективно працювати з даними. Таким чином, успіх в аналізі даних залежить не тільки від технічних засобів, але й від розуміння контексту даних, вміння правильно інтерпретувати результати та приймати на їх основі обґрунтовані рішення.

1.2 Огляд літературних джерел

На сьогоднішній час написано чимало робіт, досліджень, книжок та статей на тему аналізу даних. Всі вони висвітлюють проблеми та інструменти або шляхи для їх вирішення, пропонуючи різні підходи до ефективного аналізу та обробки даних. Серед множини інструментів для аналізу даних, Python виокремлюється як один з найбільш популярних та ефективних мов

програмування, завдяки своїй гнучкості, масштабованості та великій кількості доступних бібліотек [2-5].

В останні роки Python зміцнив свої позиції як лідер серед мов програмування в сфері аналізу даних, чому не в останню чергу сприяла підтримка широкого спектру спеціалізованих бібліотек. Pandas, NumPy, Matplotlib та Scikit-learn стали основними інструментами, що підкреслюють потужність Python у роботі з великими обсягами даних, візуалізацією результатів, та розвитку складних моделей машинного навчання. Ці бібліотеки, як вказано в книзі Веса Мак Кінні «Python for Data Analysis, 3E» [2], надають не лише готові до використання рішення для широкого спектру задач, але й значно спрощують процес розробки проєктів у цій галузі.

В книзі «Data Science at the Command Line» автора Єроена Янссенса [4] викладено практичний підхід до науки про дані, описуючи процес через модель OSEMN, що вимовляється як «awesome». Ця модель включає п’ять основних етапів: отримання даних, очищення даних, дослідження даних, моделювання даних та інтерпретація даних:

- *Отримання Даних*. Першим кроком є отримання даних. Якщо у вас ще немає даних, вам доведеться або завантажити їх з іншого місця, запитати з бази даних або API, екстрагувати з іншого файлу, або згенерувати самостійно через сенсори, опитування тощо.
- *Очищення Даних*. Дані часто містять пропуски, помилки, неконсистентні значення або непотрібні колонки. Очищення даних включає фільтрацію, вибірку певних колонок, заміну значень, обробку відсутніх значень та дублікатів, конвертацію форматів даних тощо. Цей етап є критично важливим і часто забирає значну частину часу проєкту.
- *Дослідження Даних*. Після очищення даних можна приступати до їх дослідження. На цьому етапі аналітики ознайомлюються з даними, визначають статистику та створюють візуалізації для кращого розуміння ключових ідей та важливих відкриттів, які можуть бути приховані в наборі даних.

- *Моделювання Даних*. Для пояснення даних або прогнозування майбутніх тенденцій розробляють статистичну модель. Це може включати кластеризацію, класифікацію, регресію, зниження розмірності. Автор описує використання командного рядка та різних інструментів для побудови моделей.
- *Інтерпретація Даних*. Останній етап включає аналіз результатів, формулювання висновків на основі отриманих даних, оцінку їх значущості та ефективне представлення цієї інформації. На цьому етапі від аналітика потрібне глибоке розуміння контексту дослідження, здатність критично мислити та вміння чітко аргументувати свої позиції, а також навички у викладі складних даних у зрозумілій формі.

Автор підкреслює, що процес аналізу даних є ітеративним і не обов'язково лінійним. Робота з даними часто вимагає повернення до попередніх етапів для коригування або додаткового аналізу. Цей підхід дозволяє глибше зрозуміти дані та досягнути більш точних висновків.

1.3 Приклади інструментів аналізу даних

Інструменти та технології, які застосовуються в аналізі даних, є різноманітними і підходять для широкого спектру задач. Наведемо приклади найбільш популярних інструментів і технологій в цій галузі.

SPSS (*Statistical Package for the Social Sciences*) — це програмне забезпечення, розроблене IBM для статистичного аналізу даних. SPSS забезпечує широкий спектр статистичних тестів, графічних засобів та моделювання даних, що робить його ідеальним інструментом для дослідників, які потребують глибокого аналізу даних.

Statistica — це комплексне програмне забезпечення для статистичного аналізу, розроблене компанією StatSoft, а потім перейшло до TIBCO Software. Програма забезпечує широкий спектр аналітичних інструментів, включаючи дескриптивну (описову) статистику, регресійний аналіз, аналіз часових рядів,

кластеризацію, класифікацію та багато інших методів для обробки та інтерпретації даних.

Datapine — це потужний інструмент бізнес-аналітики, який дозволяє компаніям перетворювати дані в значущі бізнес-інсайти за допомогою інтуїтивно зрозумілого інтерфейсу. З його допомогою користувачі можуть легко створювати складні запити, візуалізувати дані через інтерактивні дашборди та звіти, а також автоматизувати процеси аналізу для ефективного прийняття рішень.

Datatab.net — це інтерактивний онлайн-інструмент для аналізу даних, який дозволяє користувачам легко виконувати складні статистичні аналізи, візуалізувати дані та ділитися результатами. Ця платформа підходить для аналітиків, студентів, викладачів, які прагнуть швидко отримати інсайти зі своїх даних без необхідності глибоких знань у програмуванні.

Окремими інструментами для аналізу даних можна виділити мову програмування Python із її бібліотеками, а також мову програмування R. Обидва інструменти мають унікальні особливості та переваги для аналізу даних, пропонуючи набір інструментів для аналітиків та розробників.

Завдяки своїй читабельності, простоті у використанні та великій спільноті, Python став одним із провідних виборів серед дослідників даних та аналітиків по всьому світу. Найпопулярніші бібліотеки Python для аналізу даних включають:

Pandas: Надає швидкі, гнучкі та виразні структури даних, призначені для роботи з реляційними даними, що робить Python потужним інструментом для аналізу та очищення даних.

NumPy: Дозволяє виконувати математичні та логічні операції над масивами, є основою для багатьох інших бібліотек аналізу даних.

SciPy: Використовується для наукових та технічних обчислень як розширення NumPy, включаючи модулі для оптимізації, лінійної алгебри, інтеграції, інтерполяції та інші задачі.

Matplotlib: Потужна бібліотека для створення статичних, анімованих та інтерактивних візуалізацій в Python.

Seaborn: Надбудова над Matplotlib, що спрощує створення красивих візуалізацій для статистичних моделей.

Scikit-learn: Одна з найпопулярніших бібліотек машинного навчання, що надає прості та ефективні інструменти для аналізу даних та моделювання.

В свою чергу мова програмування R відома своєю потужною системою для створення графіків, можливістю легко обробляти дані та проводити статистичний аналіз. Основні особливості мови R включають:

Великий набір пакетів: *CRAN (Comprehensive R Archive Network)* надає тисячі пакетів для різноманітних статистичних аналізів та методів, включаючи лінійну та нелінійну регресію, статистичні тести, аналіз часових рядів, класифікацію, кластеризацію тощо.

Потужні засоби для візуалізації даних: R містить розширені можливості для створення високоякісних графіків і візуалізацій, такі як *ggplot2*, *lattice* та інші.

1.4 Аналіз застосунків-аналогів

З аналогів розглянемо *SPSS*, Statistica, *Datapine* та *Datatab.net*, тому що ці платформи представляють різні підходи до аналізу даних, кожен з яких має свої унікальні особливості, що можуть бути корисними при розробці власного застосунку. Детальний аналіз цих платформ дозволить визначити ключові функціональні можливості, які слід інтегрувати в наш проєкт, та виявити потенційні можливості для інновацій та диференціації.

SPSS

SPSS пропонує користувачам гнучкий і інтуїтивно зрозумілий інтерфейс з широким спектром опцій для аналізу, включаючи дескриптивну статистику, кореляційний аналіз, регресійний аналіз, ANOVA (аналіз варіації) та багато іншого (див. Рисунок *1*). Також програма дозволяє користувачам легко

візуалізувати дані за допомогою графіків та діаграм, що сприяє кращому розумінню та інтерпретації результатів аналізу (див. Рисунок *2*).

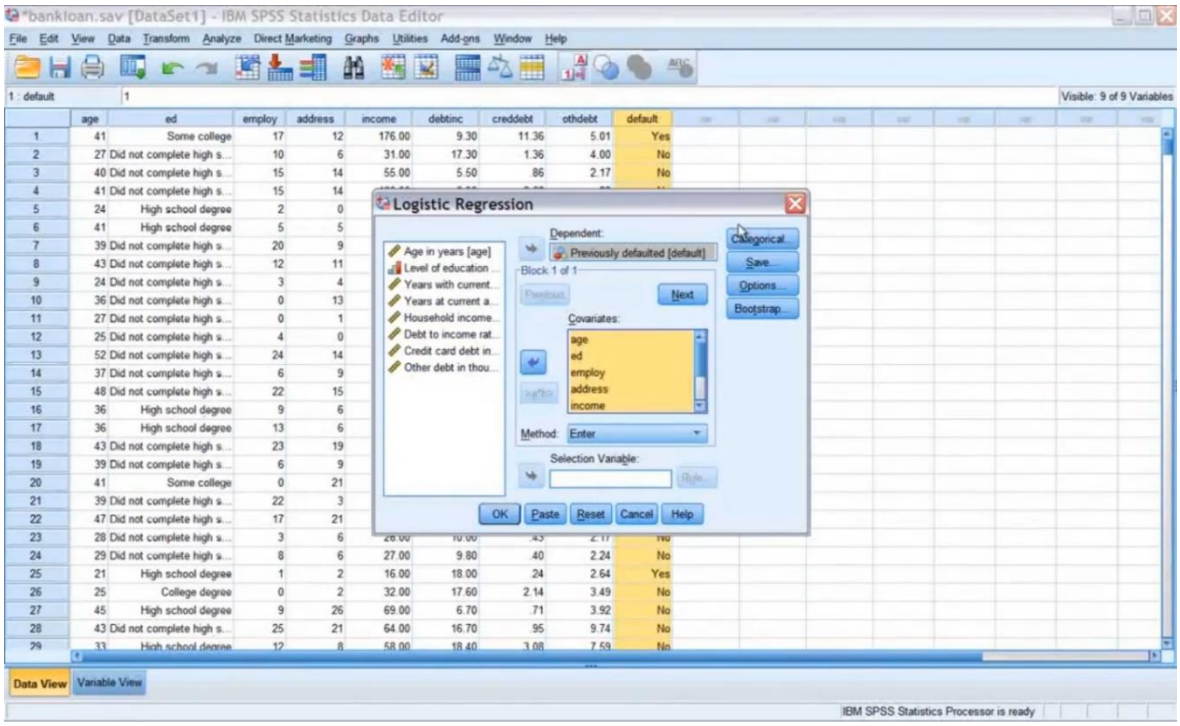

Рисунок *1 — Інтерфейс SPSS, вікно логістичної регресії [6]*

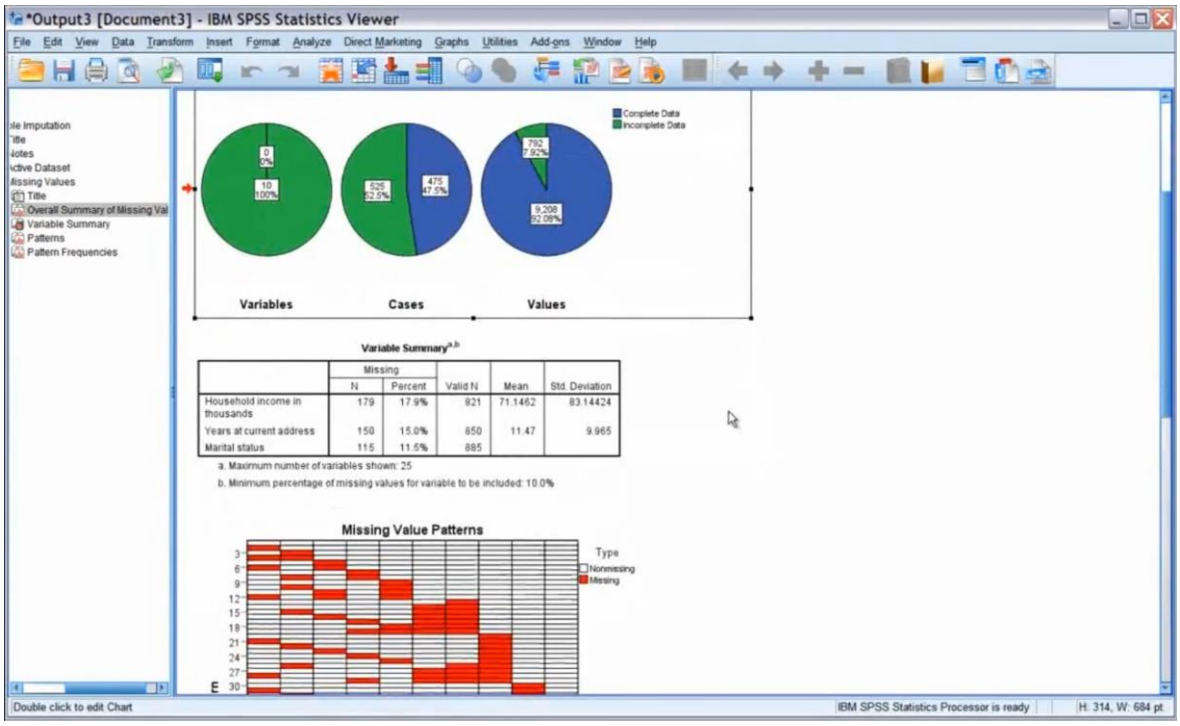

Рисунок *2 — Інтерфейс SPSS, формування звіту [6]*

Переваги SPSS:

- Широкий спектр статистичних функцій: SPSS пропонує великий набір статистичних інструментів для роботи з різноманітними типами даних і дослідницькими задачами.
- Модульність: Програма пропонує додаткові модулі для спеціалізованих аналізів, що дозволяє користувачам налаштовувати функціонал згідно зі своїми потребами.
- Підтримка різноманітних форматів даних: SPSS може імпортувати та експортувати дані в багатьох форматах, що забезпечує високу сумісність з іншими програмами та даними.
- Надійність та визнання в академічних колах: SPSS має довгу історію використання в наукових дослідженнях, що робить його надійним інструментом, визнаним спільнотою.

Недоліки SPSS:

- Висока вартість: Одним з головних недоліків SPSS є його ціна, що може бути значною, особливо для індивідуальних користувачів або малого та середнього бізнесу.
- Обмежена гнучкість у скриптінгу та автоматизації: Попри те, що SPSS має власну мову скриптів (Syntax), вона може здатися менш гнучкою порівняно з мовами програмування, такими як R або Python.

Statistica

Statistica відома своєю здатністю обробляти великі обсяги даних, високою швидкістю виконання аналізу та гнучкістю у використанні, пропонуючи користувачам як графічний інтерфейс, так і мову програмування для складніших аналітичних задач (див. Рисунок *3*).

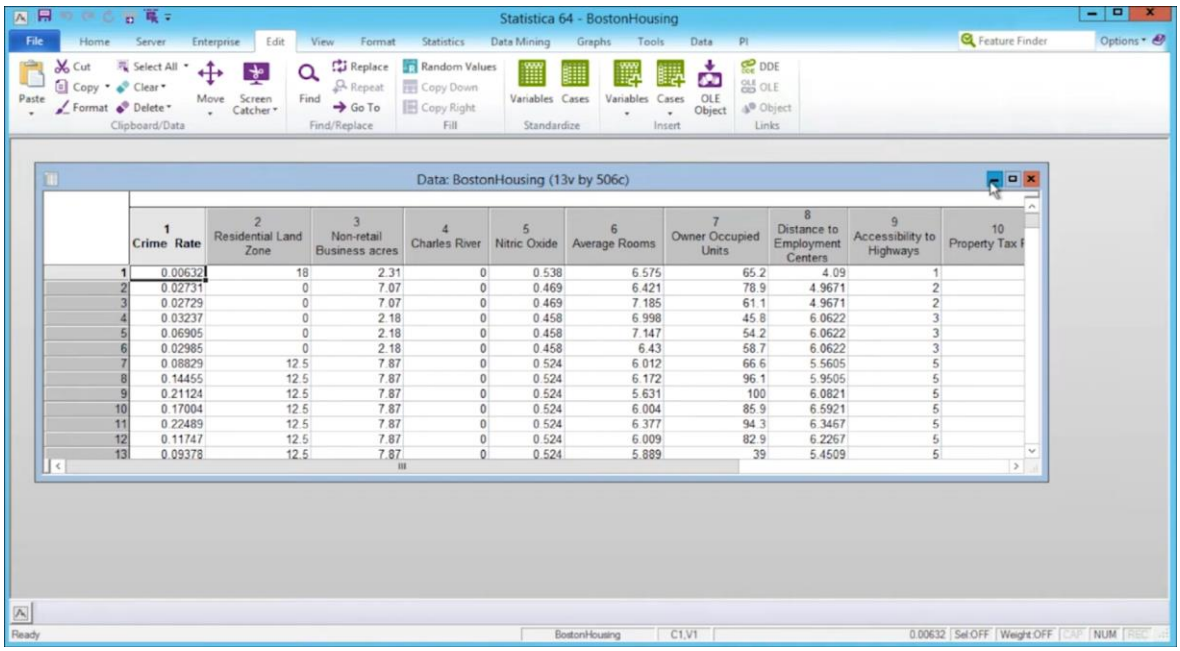

Рисунок *3* — *Інтерфейс Statistica, візуалізація даних у вигляді таблиці [7]*

Datapine Analyzer

Datapine Analyzer — це інструмент для побудови графіків різних типів та їх аналізу, вбудований в сервіс Datapine (див. Рисунок *4*). Він забезпечує користувачам можливість швидко створювати візуалізації, використовуючи різноманітні типи діаграм, включаючи лінійні графіки, гістограми, кругові діаграми, точкові діаграми та багато інших.

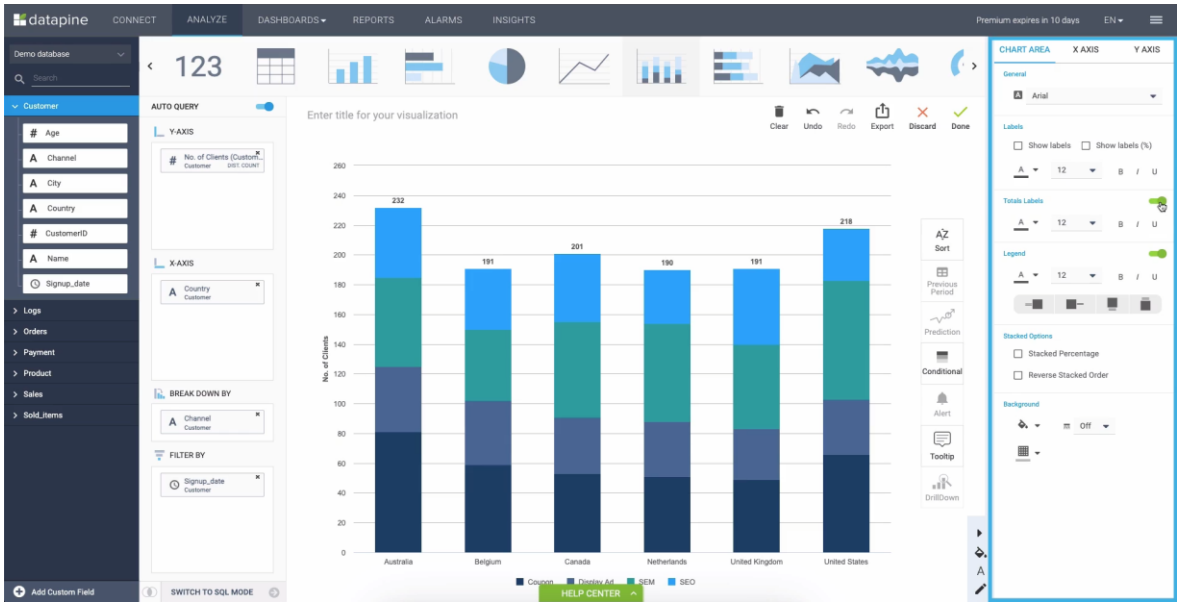

Рисунок *4* — *Інтерфейс Datapine Analyzer [8]*

Переваги Datapine Analyzer:

- Інтуїтивно зрозумілий інтерфейс: Datapine Analyzer пропонує дружній до користувача інтерфейс, що робить процес створення візуалізацій простим і зручним навіть для тих, хто не має технічного фону в аналітиці даних.
- Гнучкість у візуалізації даних: Користувачі можуть легко налаштовувати свої візуалізації, вибираючи з різноманітних типів графіків та діаграм для найкращого представлення своїх даних.
- Інтеграція з різними джерелами даних: Datapine дозволяє імпортувати дані з багатьох джерел, включаючи бази даних, хмарні сховища, а також пряме підключення до різних бізнес-додатків і платформ.

Недоліки Datapine Analyzer:

- Ціна: Послуги Datapine, включаючи Analyzer, можуть бути відносно дорогими, особливо для малого та середнього бізнесу або індивідуальних користувачів з обмеженим бюджетом.

Datatab.net

Datatab.net надає користувачам інструменти для проведення статистичного аналізу (див. Рисунок *6*) та візуалізації даних (див. Рисунок *5*) без необхідності володіння глибокими знаннями у програмуванні або статистиці.

Ця платформа зосереджена на спрощенні процесу аналізу даних, роблячи його доступним для широкої аудиторії, включаючи студентів, викладачів, маркетологів, малий бізнес та інших користувачів, які потребують аналізу даних у своїй роботі або дослідженнях.

Крім того, платформа надає детальні інсайти, допомагаючи користувачам краще розуміти результати аналізу та приймати обґрунтовані рішення на основі отриманих даних.

DATAtab Statistics Calculator Survey Tutorials Books Pricing Login EN Install

Online Statistics Calculator 100% data security

Clear Table Export / Import Transform data Settings

	nominal	metric	metric	nominal	metric	nominal	ordinal			
Cases	Gender	Salary	Age	Place	Weight	Company	Academic degree			
1	Female	1500	33	Chicago	80	BMW	Bachelor			
2	Female	1200	33	Chicago	82.5	Ford	No			
3	Male	2200	34	New York	100.8	BMW	Bachelor			
4	Male	2100	42	New York	90	BMW	Master			
5	Female	1500	29	Chicago	67	Ford	Master			
6	Female	1700	19	Washington	60	Ford	Master			
7	Male	3000	50	Washington	77	Ford	No			
8	Male	3000	55	Washington	77	Ford	Bachelor			
9	Female	2800	31	New York	87	Ford	Bachelor			
10	Male	2900	46	New York	70	GM	Master			
11	Female	2780	36	Washington	57	BMW	No			
12	Male	2550	48	New York	64	GM	Master			
13										
14										

Descriptive Charts Hypothesis tests Correlation Regression Mediation/Moderation PCA Reliability Cluster

Dependent Variable:

○ Gender ○ Salary ◉ Age ○ Place ○ Weight ○ Company ○ Academic degree

Independent Variable:

☐ Gender ☑ Salary ☐ Age ☐ Place ☑ Weight ☐ Company ☐ Academic degree

Linear Regression

Рисунок 5 — *Інтерфейс Datatab.net*

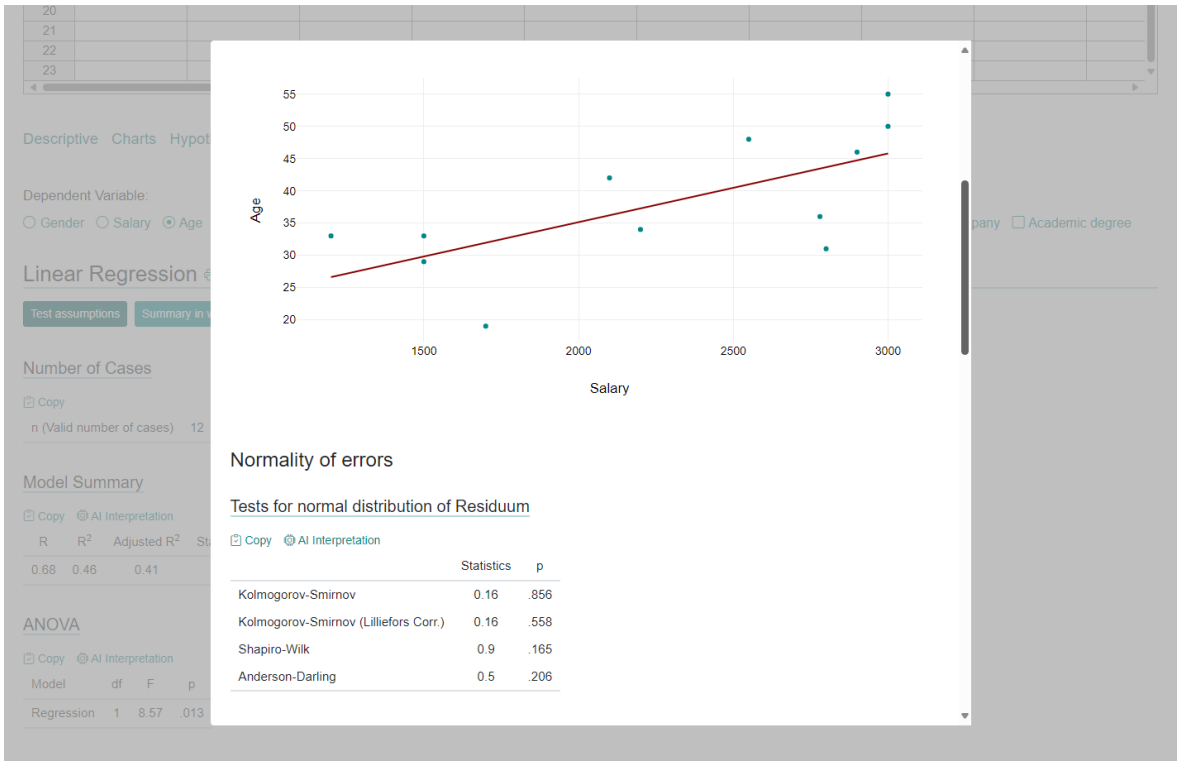

Рисунок 6 — *Регресійний аналіз в Datatab.net*

Переваги Datatab.net:

- Простота використання: Datatab.net вирізняється своїм інтуїтивно зрозумілим інтерфейсом, що дозволяє користувачам легко завантажувати дані, виконувати аналіз та створювати візуалізації без складних налаштувань.
- Безкоштовний доступ: Однією з ключових переваг є безкоштовний доступ до основних функцій платформи, що робить Datatab.net привабливим вибором для користувачів з обмеженим бюджетом.
- Підтримка різноманітних форматів даних: Платформа підтримує імпорт даних у різних форматах, що забезпечує гнучкість при роботі з даними з різних джерел.
- Штучний інтелект: Datatab.net може видати інформацію та аналітичні висновки в формі, зрозумілій для людини, завдяки використанню алгоритмів штучного інтелекту (ШІ). Ця можливість включає генерацію зрозумілих описів тенденцій, закономірностей та аномалій у даних, що значно спрощує інтерпретацію результатів аналізу для користувачів без глибоких знань у статистиці або аналізі даних.
- Безпека даних: Всі дані та розрахунки виконуються на клієнтській частині без відправки жодної інформації на сервер.

Недоліки Datatab.net:

- Обмежені аналітичні можливості: Порівняно з більш розвиненими аналітичними платформами, Datatab.net може мати обмеження в плані глибини та складності доступних аналітичних методів.

Таким чином, розгляд аналогів підкреслює важливість і потребу в інструментах, які дозволяють ефективно обробляти, аналізувати та візуалізувати великі обсяги даних. Кожен з цих інструментів вносить свій унікальний вклад у розвиток аналітичних можливостей, пропонуючи рішення для різноманітних завдань та потреб користувачів.

Однак, не дивлячись на широкий спектр доступних аналітичних інструментів, постійно зростаючий обсяг даних та зміна потреб бізнесу та

науки вимагають розробки нових продуктів, що володіють вищою швидкістю обробки даних, гнучкістю у візуалізації, інтеграції з іншими сервісами та платформами, а також можливістю використання новітніх технологій, таких як штучний інтелект і машинне навчання.

Розробка нового продукту в цій області повинна враховувати як позитивний досвід існуючих рішень, так і їх недоліки, а також акцентувати увагу на підвищенні доступності аналітичних інструментів для широкого кола користувачів. Це включає забезпечення інтуїтивно зрозумілого інтерфейсу, ефективних механізмів для роботи з великими даними, високу продуктивність аналітичних алгоритмів, та можливість глибокої кастомізації аналізу та візуалізацій.

Враховуючи ці фактори, актуальність створення нових або удосконалення існуючих продуктів у сфері аналізу даних не може бути переоцінена. Інновації в цій галузі відкривають нові можливості для наукових досліджень, розвитку бізнесу, оптимізації процесів та прийняття обґрунтованих рішень на основі даних, що робить аналітичні продукти ключовим елементом в структурі сучасного інформаційного простору.

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Аналіз сучасних фреймворків

Фреймворк, у загальному сенсі, це набір інструментів та бібліотек, який спрощує та прискорює розробку програмного забезпечення, надаючи готовий до використання каркас. Він визначає структуру програми, пропонуючи стандартизовані способи вирішення загальних програмних завдань, що дозволяє розробникам зосередитися на унікальних аспектах своїх проєктів, а не на повторному винаходженні базової інфраструктури. Фреймворки можуть бути спрямовані на різні аспекти розробки, включаючи графічний інтерфейс користувача, доступ до баз даних, управління мережевими запитами тощо.

Фреймворки, як важливий компонент в розробці програмного забезпечення, надають ряд критично важливих функцій, які допомагають розробникам створювати ефективні та масштабовані додатки. Ось декілька ключових функцій фреймворків:

Стандартизація коду: Фреймворки пропонують набір стандартів і практик для написання коду, що сприяє узгодженості, читабельності та підтримуваності проєктів. Це спрощує співпрацю в команді, оскільки всі розробники слідують одній і тій же структурі і набору правил.

Безпека: Багато фреймворків включають вбудовані механізми безпеки для захисту від загальних загроз, таких як SQL-ін’єкції, cross-site scripting (XSS), і cross-site request forgery (CSRF). Використання таких фреймворків допомагає розробникам застосовувати найкращі практики безпеки без необхідності вручну реалізовувати захист на кожному етапі розробки.

Швидкість розробки: Завдяки використанню готових компонентів і бібліотек, фреймворки значно знижують час, необхідний для розробки проєктів. Розробники можуть зосередитися на реалізації специфічної логіки та функціональності своїх додатків, не витрачаючи час на створення базової інфраструктури.

Масштабованість: Фреймворки часто розроблені з урахуванням потреб масштабування додатків, надаючи розробникам інструменти та конвенції для побудови ефективних систем, які можуть обробляти зростаючий об’єм даних та трафіку.

Тестування: Велика кількість фреймворків включають інструменти для автоматизації тестування, що полегшує перевірку коду на відповідність вимогам і виявлення помилок на ранніх стадіях розробки.

Веб-фреймворк — це спеціалізований тип фреймворку, який призначений для розробки веб-додатків. Він надає набір інструментів та бібліотек, спрямованих на спрощення розробки та управління веб-додатками, включаючи обробку HTTP-запитів, шаблонізацію веб-сторінок, управління сесіями, взаємодію з базами даних та інші веб-специфічні функції. Веб-фреймворки дозволяють розробникам швидко створювати веб-додатки, забезпечуючи їм стандартизовану архітектуру та набір рекомендацій для реалізації типових веб-завдань, таких як автентифікація користувачів, управління даними та рендеринг веб-сторінок.

У процесі розробки веб-додатків, вибір відповідного веб-фреймворку є критичним кроком, який може мати далекосяжні наслідки. Цей вибір прямо впливає на швидкість розробки, оскільки деякі фреймворки пропонують широкий спектр готових до використання компонентів та інструментів, що значно пришвидшує процес розробки. Водночас, гнучкість фреймворку визначає, наскільки легко можна буде адаптувати додаток до змінюваних вимог та інтегрувати нові функції в майбутньому. Масштабованість також є критичною характеристикою, яка вимагає уваги. Вибраний фреймворк повинен бути здатним ефективно справлятися з зростаючим обсягом даних та користувацьких запитів, забезпечуючи стабільну та швидку роботу додатку.

Сучасний ринок пропонує широкий вибір веб-фреймворків, кожен з яких пропонує унікальний набір функціональностей, переваг та обмежень. Вибір конкретного фреймворку вимагає глибокого розуміння як технічних деталей, так і бізнес-вимог проєкту. Тому, в цьому розділі ми зосередимося на

детальному аналізі лідируючих веб-фреймворків, їх порівняльному розгляді, з акцентом на особливості, які роблять їх особливо придатними для розробки додатку аналізу даних.

2.1.1 Огляд веб-фреймворків для фронтенду

Django

Django є високорівневим веб-фреймворком, написаним на Python, який надає широкий спектр функціональностей, що дозволяє розробникам зосередитися на створенні веб-додатку, а не на вирішенні стандартних задач, таких як аутентифікація користувачів, управління сесіями, робота з формами та інтеграція з базами даних [9].

Основні характеристики Django:

- *Повнофункціональний*: Django включає в себе все необхідне для розробки веб-додатку: адміністративний інтерфейс, системи аутентифікації, шаблони, обробку форм, інтернаціоналізацію, та багато іншого.
- *Масштабованість*: Django підходить для проєктів будь-якого розміру, від невеликих додатків до великих порталів з високим навантаженням, і може бути масштабований для задоволення зростаючих потреб проєкту.
- *Гнучка архітектура*: Заснований на архітектурному шаблоні Model-View-Template (MVT), Django дозволяє легко модифікувати кожну частину веб-додатку незалежно.

Django підходить для розробки великих веб-додатків із складною бізнес-логікою, таких як соціальні мережі, платформи електронної комерції та інші великі портали.

Django є міцним фундаментом для будь-якого веб-проєкту, що вимагає швидкої розробки, високого рівня безпеки та здатності до масштабування. Він дозволяє розробникам відмовитися від повторного винаходження колеса для стандартних завдань і зосередитись на унікальних аспектах свого проєкту,

прискорюючи процес розробки та знижуючи загальні витрати на реалізацію проєкту.

Angular

Angular — це відома платформа та фреймворк для розробки односторінкових веб-додатків (SPA), побудована на TypeScript компанією Google. Вона включає компонентний фреймворк для створення масштабованих веб-додатків та набір інтегрованих бібліотек, які покривають широкий спектр функціональностей: від маршрутизації до комунікації клієнт-сервер. Angular спрощує процес розробки за рахунок стандартизації рутинних завдань і надає інструменти для розробки, тестування та збірки коду [10].

Основні характеристики Angular:

- *Компонентний підхід*: Angular використовує компоненти як основні будівельні блоки додатків, що спрощує розробку та тестування. Компоненти в Angular є самодостатніми елементами, що включають HTML-шаблони та логіку обробки даних, які вони представляють.
- *TypeScript*: Angular розробляється з використанням TypeScript, що додає строгу типізацію та об’єктно-орієнтовані можливості. Це сприяє кращій читабельності коду, полегшує його рефакторинг та допомагає уникати помилок на ранніх етапах розробки.
- *Декларативність*: Angular підтримує декларативний підхід до опису користувацьких інтерфейсів, дозволяючи розробникам концентруватися на тому, що вони хочуть досягти, а не на тому, як саме це реалізувати. Це робить код більш читабельним, спрощує процес розробки і сприяє легкості підтримки додатків.
- *Масштабованість*: Має здатність ефективно управляти станом при значному збільшенні обсягів даних і кількості користувачів, а також модульна структура, сприяють простоті розширення проєктів.
- *Двостороннє зв’язування даних*: Angular надає механізм двостороннього зв’язування даних, що автоматично синхронізує дані між моделлю компоненту та його представленням.

- *Масштабованість*: Завдяки компонентному підходу, розширеним можливостям та гнучкій архітектурі, Angular дозволяє створювати веб-додатки, які можуть ефективно рости та адаптуватися до змінних вимог.

Angular є потужним інструментом у сучасній веб-розробці, що надає широкі можливості для створення інтерактивних та динамічних веб-додатків. Завдяки своїй архітектурі, орієнтованій на компоненти, підтримці TypeScript та іншим передовим функціям, Angular допомагає розробникам ефективно вирішувати складні завдання, підвищуючи якість та швидкість розробки веб-додатків.

Vue

Vue або Vue.js, є прогресивним JavaScript фреймворком, створеним Еваном Ю. [11]. Завдяки своїй простоті та гнучкості, Vue швидко здобув популярність серед розробників, ставши одним з провідних інструментів для створення інтерфейсів користувача та односторінкових додатків (SPA). Vue забезпечує реактивне та компонентне програмування даних, дозволяючи розробникам легко створювати швидкі та гнучкі веб-додатки.

Основні характеристики Vue:

- *Прогресивність*: Vue розроблений як прогресивний фреймворк. Це означає, що ви можете використовувати Vue для будь-якої частини вашого проєкту, від невеликих компонентів до повноцінних SPA, інтегруючи його поступово без необхідності переписувати існуючий код.
- *Реактивність*: Використання системи реактивності дозволяє автоматично оновлювати UI (користувацький інтерфейс) відповідно до змін у стані додатку, забезпечуючи високий рівень інтерактивності та користувацького досвіду.
- *Декларативність*: Vue дозволяє розробникам використовувати декларативний підхід для створення інтерфейсів, описуючи що має бути зроблено через HTML-шаблони.

- *Компонентний* підхід: Vue сприяє розробці додатків через повторно використовувані компоненти, що дозволяє краще організувати код та полегшити його підтримку.
- *Деталізована документація*: Vue має одну з найкращих документацій серед веб-фреймворків, що робить його доступним для новачків та забезпечує швидке введення в роботу для досвідчених розробників.
- *Гнучкість та легкість інтеграції*: Vue може легко інтегруватися з іншими бібліотеками або існуючими проєктами, а також може функціонувати як частина складнішої екосистеми веб-розробки.

Vue використовується для розробки динамічних веб-інтерфейсів та односторінкових додатків. Його легкість, гнучкість і простота вивчення роблять його ідеальним вибором для проєктів різного масштабу, від маленьких до великих корпоративних рішень. Vue дозволяє розробникам швидко прототипувати ідеї, ефективно управляти станом додатків та реалізовувати складні користувацькі інтерфейси з мінімальними зусиллями.

React

React є декларативною, ефективною та гнучкою JavaScript бібліотекою для створення користувацьких інтерфейсів, розробленою командою Facebook [12]. Завдяки своїй гнучкості та інтегрованості з іншими бібліотеками та фреймворками, React можна використовувати як основу в масштабних веб-додатках, ефективно вписуючись в ширший екосистему сучасної веб-розробки.

Основні характеристики React:

- *Компонентний підхід*: React спрощує розробку за допомогою повторно використовуваних компонентів, які можуть управляти своїм станом та логікою, забезпечуючи чистоту та організованість коду.
- *Декларативність*: Робить код більш зрозумілим та легшим для налагодження, оскільки React автоматично оновлює інтерфейс користувача відповідно до змін у даних.

- *Реактивність*: Забезпечує реактивне оновлення інтерфейсу через свою систему станів та пропсів (англ. props), автоматично реагуючи на зміни.
- *Односторонній потік даних*: Спрощує управління даними та сприяє створенню більш передбачуваних та легко керованих додатків.

Хоча React сам по собі не є веб-фреймворком, він часто розглядається як основний інструмент при розробці сучасних веб-додатків, особливо односторінкових аплікацій (SPA), де потрібна швидка взаємодія з користувачем та динамічне оновлення контенту без перезавантаження сторінки. Завдяки можливості інтеграції з іншими бібліотеками та сервісами, React стає важливою частиною екосистеми розробки, надаючи розробникам гнучкість у виборі інструментів для вирішення конкретних задач.

ASP.NET

ASP.NET є відкритим та безкоштовним веб-фреймворком для створення веб-додатків, що розробляється компанією Microsoft [13]. Це частина платформи .NET, що дозволяє розробникам створювати веб-додатки, сервіси та динамічні веб-сайти. ASP.NET підтримує різноманітні мови програмування, включаючи C#, VB.NET, та F#, надаючи розробникам гнучкість у виборі інструментів та підходів до розробки.

Основні характеристики ASP.NET:

- *Модель програмування подій*: ASP.NET використовує модель програмування на основі подій, схожу на ту, що використовується в розробці десктопних додатків (додатків для настільних комп’ютерів). Це дозволяє легко управляти подіями користувацького інтерфейсу, такими як натискання кнопок або події форм.
- *Підтримка MVC та Web API*: ASP.NET включає підтримку патерну Model-View-Controller (MVC), що дозволяє розробникам створювати веб-додатки, організовані за чіткою архітектурою, та Web API для створення RESTful веб-сервісів.

ASP.NET широко використовується для розробки корпоративних веб-додатків, веб-сайтів з великим обсягом даних, веб-сервісів, а також для

створення інтернет-магазинів, форумів, порталів та інших складних веб-проєктів. Його продуктивність, масштабованість та безпека роблять ASP.NET відмінним вибором для бізнесу будь-якого розміру.

Отже, для розробки веб-додатку було обрано фреймворк Vue. Вибір Vue базується на ряді ключових факторів, які роблять цей фреймворк особливо привабливим для проєктів, що цінують швидкість розробки, легкість вивчення та гнучкість використання. Порівняння Vue з іншими популярними веб-фреймворками і бібліотеками висвітлює його переваги та обґрунтовує вибір на його користь.

Django орієнтований на швидкість розробки та чистоту дизайну. Проте, в контексті розробки додатків, що вимагають високого рівня інтерактивності та динамічного оновлення інтерфейсу без перезавантаження сторінки, Django може виявитися менш зручним. Відсутність вбудованої підтримки реактивності або двостороннього зв’язування даних може ускладнити розробку додатків, які зосереджені на користувацькому досвіді та інтерактивності.

ASP.NET ідеально підходить для створення високопродуктивних веб-додатків з суворою типізацією, безпекою та масштабованістю. Однак ASP.NET може вимагати більше часу порівняно з використанням Vue, особливо коли мова йде про проєкти, які потребують швидкого прототипування або легкої інтеграції з іншими технологіями на стороні клієнта. Це пов’язано з тим, що ASP.NET спрямований на розробку більш комплексних серверних рішень, які можуть включати ретельну настройку конфігурації, розгортання та взаємодії з іншими серверними сервісами, що з часом може збільшувати загальний час розробки.

У порівнянні Vue з Angular та React, вибір найбільш підходящого фреймворка може виявитися складним завданням, оскільки кожен з них пропонує унікальний набір можливостей та переваг, які можуть бути цінними в залежності від потреб проєкту. Всі три фреймворки більш менш схожі, адже з часом переймали різні функції один від одного.

Вибір Vue ж виявився найбільш оптимальним для розробки цього застосунку для аналізу даних, особливо через його легкість у навчанні та гарно прописану документацію. Vue поєднує в собі гнучкість та ефективність, що робить його ідеальним для швидкого прототипування та розробки, що є критично важливим в контексті обмеженого часу на реалізацію проєкту.

Також, у порівнянні з Angular та React, Vue виділяється своєю простотою та високою продуктивністю, що робить його особливо привабливим для розробки нескладних, але водночас інтерактивних веб-додатків. Це ідеально підходить для цього проєкту, який не вимагає складної бізнес-логіки або великої масштабованості на початковому етапі. Vue дозволяє швидко створити прототип і поступово додавати необхідну функціональність.

2.1.2 Огляд фреймворків для бекенду

Flask

Flask є легким WSGI (Web Server Gateway Interface) веб-фреймворком. Він написаний на мові програмування Python [14]. Особливість Flask полягає в його простоті та гнучкості, дозволяючи розробникам швидко створювати веб-додатки, використовуючи мінімалістичний підхід.

Основні характеристики Flask:

- *Мінімалістичний та легкий*: Flask надає базові інструменти для розробки веб-додатків, не перевантажуючи розробників великою кількістю непотрібних можливостей. Це робить його ідеальним вибором для розробки простих веб-додатків або прототипів.
- *Гнучкість та розширення*: Завдяки системі розширень (extensions), Flask може бути легко адаптований для виконання складніших завдань. Розробники можуть додавати функціональність, таку як обробка форм, автентифікація користувачів, бази даних і багато іншого, використовуючи доступні розширення.

- *Використання шаблонів*: Flask підтримує систему шаблонів Jinja2, яка дозволяє легко створювати динамічний контент веб-сайтів, забезпечуючи швидке та ефективне відтворення веб-сторінок.
- *Підтримка розробки RESTful API*: Flask відмінно підходить для створення легких RESTful веб-сервісів завдяки своїм легким запитам та простоті обробки вхідних та вихідних даних.
- *Тестування*: Flask забезпечує підтримку тестування веб-додатків з можливістю легко інтегрувати одиниці тестів для забезпечення якості та надійності програмного забезпечення.

Flask широко використовується для розробки як малих, так і середніх веб-додатків, веб-сайтів, веб-сервісів та API. Його легкість, простота в навчанні та велика екосистема роблять Flask привабливим вибором для стартапів та індивідуальних розробників.

FastAPI

FastAPI є сучасним, швидким (як випливає з назви) веб-фреймворком для створення API з Python з підтримкою асинхронного програмування [15]. FastAPI автоматично генерує документацію для API за допомогою Swagger UI та ReDoc, що робить його ідеальним вибором для створення як простих, так і складних веб-сервісів. Розроблений на основі Starlette (для веб-частини) та Pydantic (для даних), він забезпечує високу продуктивність та ефективність, використовуючи асинхронні запити та валідацію даних.

Основні характеристики FastAPI:

- *Швидкість та продуктивність*: FastAPI було спроєктовано для оптимізації швидкості та продуктивності, що робить його одним з найшвидших веб-фреймворків для Python.
- *Автоматична документація*: Завдяки використанню стандартних анотацій Python для типів, FastAPI автоматично генерує документацію для API, що робить розробку та тестування API зручнішими.

- *Підтримка асинхронного програмування*: Асинхронність вбудована в ядро FastAPI, дозволяючи розробникам використовувати асинхронні функції Python для покращення продуктивності веб-додатків.
- *Легкість використання*: FastAPI легко вчити та використовувати, дозволяючи розробникам швидко створювати веб-додатки та API. Його дизайн зосереджений на розробці з мінімальними вимогами до бойлерплейта і високою читабельністю коду.
- *Сильна підтримка типізації та валідації даних*: Використовуючи Pydantic, FastAPI забезпечує сильну підтримку типізації та валідацію даних, що допомагає зменшити кількість помилок і підвищити якість коду.

FastAPI широко використовується для створення високопродуктивних API, що вимагають великої швидкості відповіді та ефективного оброблення запитів. Його легкість у навчанні та гнучкість роблять його відмінним вибором для стартапів, великих компаній та індивідуальних розробників, які прагнуть створити швидкі, надійні та легко масштабовані веб-сервіси.

Отже, в порівнянні між Flask та FastAPI вибір перепадає на більш швидкий та новий фреймворк — FastAPI, з огляду на його вражаючу продуктивність, вбудовану підтримку асинхронного програмування, та автоматичну генерацію документації. FastAPI пропонує сучасні можливості для розробників, спрямовані на швидку розробку високопродуктивних API з чіткою типізацією та валідацією даних. Тим не менш, Flask залишається дуже популярним вибором завдяки своїй простоті, гнучкості та широкій підтримці в розробницькій спільноті.

Flask ідеально підходить для проєктів, які вимагають мінімалістичного підходу або коли розробники віддають перевагу більш традиційному синхронному стилю програмування. В свою чергу FastAPI підходить для проєктів, що вимагають максимальної продуктивності, використання асинхронності та сучасних підходів до створення API, це саме те, що робить його ідеальним для веб-застосунків аналізу даних. Ці аспекти важливі для

додатку з аналізу даних, оскільки вони дозволяють ефективно обробляти великі обсяги даних, забезпечують швидку чуйність системи та високу доступність сервісу, що є критично важливим для роботи з динамічними та великими наборами даних. Асинхронність зокрема дає змогу оптимізувати процеси обробки запитів до сервера, не блокуючи основний потік виконання програми, що є ключовим для підтримки високої продуктивності в режимі реального часу.

2.1.3 Огляд баз даних і технологій зберігання даних

SQLite

SQLite є компактною бібліотекою, яка реалізує вбудований, самодостатній, безсерверний і безконфігураційний SQL-двигун бази даних [16]. Відрізняючись від більшості SQL-баз даних, SQLite не використовує окремий серверний процес. Вона читає та записує дані безпосередньо у звичайні дискові файли. SQLite забезпечує швидкодію, надійність та стабільність, та є найбільш широко розповсюдженою базою даних у світі.

Основні характеристики SQLite:

- *Самодостатність*: SQLite є вбудованим SQL-двигуном бази даних, який не вимагає окремого серверного процесу, роблячи його самодостатнім рішенням для управління даними.
- *Безсерверність*: Відсутність необхідності в окремому сервері робить SQLite ідеальним для застосунків, які потребують легковажної та ефективної системи управління базами даних.
- *Швидкодія*: SQLite може працювати швидше за прямий доступ до файлової системи, особливо при використанні оптимального обсягу пам’яті.
- *Надійність та стабільність*: SQLite пройшов ретельне тестування, маючи репутацію надійної системи управління базами даних з високим рівнем покриття коду тестами.

- *Тривала підтримка*: Розробники SQLite планують підтримувати проєкт до 2050 року, забезпечуючи довготривалу підтримку та розвиток.

Ці характеристики роблять SQLite відмінним вибором для розробників, які шукають надійну, ефективну та легко інтегровану систему управління базами даних для своїх застосунків.

PostgreSQL

PostgreSQL — це потужна об’єктно-реляційна система управління базами даних [17]. Ця система надає розширені можливості SQL, що дозволяє ефективно зберігати та обробляти навіть найскладніші набори даних. Завдяки понад 35 рокам активної розробки, PostgreSQL зарекомендувала себе як надійна платформа з високою продуктивністю, гнучкістю та широким набором функцій.

Основні характеристики PostgreSQL:

- *Гнучкість та розширення*: PostgreSQL дозволяє визначати власні типи даних, створювати користувацькі функції та інтегрувати код на різних мовах програмування, не перекомпілюючи базу даних.
- *Висока продуктивність та оптимізація*: Включає індексацію, паралелізацію запитів, розширене планування запитів, та JIT-компіляцію виразів для забезпечення високої продуктивності.
- *Забезпечення безпеки*: Містить розширені засоби аутентифікації, контролю доступу на рівні стовпців та рядків, та підтримку багатофакторної аутентифікації.
- *Підтримка стандартів SQL*: PostgreSQL підтримує значну кількість обов'язкових функцій стандарту SQL:2023, забезпечуючи високий рівень сумісності з іншими системами.
- *Розширені можливості зберігання даних*: Підтримує широкий спектр типів даних включно з примітивними, структурованими, документальними та геометричними типами даних.

PostgreSQL ідеально підходить для проєктів, що вимагають високої надійності, продуктивності та гнучкості в роботі з даними. Це можуть бути

веб-додатки, які обробляють великі обсяги транзакцій, системи для аналізу даних, що потребують складних запитів та агрегації. Крім того, PostgreSQL чудово підходить для проєктів з високими вимогами до безпеки даних та для компаній, які планують масштабування своїх систем.

MongoDB

MongoDB є NoSQL системою управління базами даних розробленою для сучасних застосунків, які потребують швидкої обробки великих обсягів неструктурованих або напівструктурованих даних. Вона використовує гнучку схему документів у форматі JSON-подібний (BSON, тобто бінарний JSON), що дозволяє розробникам легко і швидко модифікувати та розширювати свої бази даних [18-19]. Розподілена архітектура MongoDB забезпечує високу доступність, горизонтальне масштабування та географічне розподілення даних.

Основні характеристики MongoDB:

- *Гнучкість схеми*: MongoDB дозволяє зберігати документи в форматі BSON, які можуть містити різноманітні типи даних та структури без потреби в єдиній схемі для всієї бази даних.
- *Масштабованість*: Підтримує горизонтальне масштабування за допомогою сегментування (sharding), дозволяючи розподіляти дані по кластерах серверів для забезпечення високої продуктивності обробки великих обсягів даних.
- *Великий набір операцій з даними*: Підтримує різноманітні операції з даними, включаючи агрегацію, текстовий пошук, геопросторові запити, що робить її потужним інструментом для розробки складних застосунків.

MongoDB ідеально підходить для проєктів, що вимагають високої гнучкості у роботі з даними. Її можливість швидко адаптуватися до змін у структурі даних робить MongoDB відмінним вибором для стартапів та інноваційних проєктів, які еволюціонують разом з потребами користувачів. Також вона ефективна для проєктів, де потрібно зберігати велику кількість

неструктурованих або напівструктурованих даних, наприклад, логів (журналів) дій користувачів, соціальних мереж, контенту, генерованого користувачами та інше.

Враховуючи специфіку проєкту, де основна увага приділяється обробці структурованих даних було вирішено, що NoSQL бази даних, такі як MongoDB, не відповідають потребам проєкту через їхню орієнтацію на неструктуровані або напівструктуровані дані. Так, MongoDB може також справлятися з задачами, представленими цим застосунком, проте її використання могло б призвести до зайвої складності при роботі з даними, які краще піддаються стандартному реляційному зберіганню та аналізу. Додатково, реляційна модель, яку пропонує PostgreSQL, сприяє більш строгій організації даних і підтримці цілісності на рівні схеми бази даних, що є критично важливим для аналітичних застосунків, які залежать від точності та надійності даних.

SQLite, хоча і є дуже надійною та легко інтегрованою системою для управління базами даних, може зіткнутися з обмеженнями при масштабуванні та обробці великих обсягів даних, що є критичним для аналітичного застосунку, особливо коли кількість користувачів різко зростає.

У результаті, вибір був зроблений на користь PostgreSQL — потужної об'єктно-реляційної системи управління базами даних, яка ідеально підходить для проєктів, що вимагають високої продуктивності, надійності та гнучкості в роботі з даними. PostgreSQL вирізняється високою масштабованістю та здатністю ефективно справлятися з великими обсягами даних, що робить її оптимальним вибором для застосунку аналізу даних, який планується розширювати та розвивати з часом.

2.2 Аналіз алгоритмів аналізу даних

Алгоритми, які було розглянуто, включають лінійну регресію, поліноміальну регресію, градієнтний спуск та стохастичний градієнтний

спуск. Кожен з цих алгоритмів має унікальні властивості та призначення, які роблять їх незамінними для різноманітних аналітичних завдань.

Лінійна регресія

Лінійна регресія є одним з базових алгоритмів статистичного аналізу і машинного навчання, використовуваним для прогнозування числових значень [3]. Основна концепція лінійної регресії полягає у встановленні лінійного зв'язку між залежною змінною Y і однією або декількома незалежними змінними X_i. Цей зв'язок можна виразити у формі рівняння:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon, \quad (1)$$

де Y — залежна пояснювана змінна;

X_i — незалежні пояснювальні змінні;

β_0 — вільний член;

β_i — коефіцієнти регресії, які вказують вплив відповідних незалежних змінних на залежну змінну;

ε — випадкова помилка, що враховує випадкові відхилення.

Середньоквадратична похибка

Середньоквадратична помилка (англ. Mean Squared Error, MSE) визначає середнє значення квадратів різниць між фактичними (реальними) значеннями і передбаченими значеннями [3]. Чим менше значення MSE, тим краще модель відповідає даним. Формула середньоквадратичної помилки виглядає так:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \quad (2)$$

де Y_i — спостережувана змінна;

$\hat{Y}_i$ — передбачене значення;

n — довжина вибірки;

Поліноміальна регресія

Поліноміальна регресія є розширенням лінійної регресії, що дозволяє моделювати більш складні відносини між залежною змінною і незалежними змінними через введення вищих ступенів незалежних змінних [3]. Цей метод є особливо корисним, коли відносини між змінними мають нелінійний характер. Рівняння поліноміальної регресії може бути представлене у наступній формулі:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_n X^n + \varepsilon, \quad (3)$$

де Y — залежна пояснювана змінна;

X — незалежна пояснювальна змінна;

β_0 — коефіцієнт поліноміальної моделі, який визначає вплив кожного ступеню X;

ε — випадкова помилка, яка припускає нормальний розподіл із середнім значенням нуль.

Аналогічно лінійній регресії, коефіцієнти поліноміальної регресії зазвичай оцінюються за допомогою методу найменших квадратів. Процес мінімізації суми квадратів різниць між спостережуваними та прогнозованими значеннями є ідентичним.

Градієнтний спуск

Градієнтний спуск є фундаментальним алгоритмом в оптимізації для мінімізації функції втрат. Цей метод дозволяє знаходити мінімальне значення функції, регулюючи параметри в напрямку найсильнішого спаду її градієнта.

Основна ідея градієнтного спуску полягає в ітеративному оновленні параметрів моделі, щоб зменшити функцію втрат [3]. На кожному кроці параметри моделі оновлюються в напрямку, протилежному градієнту (або похідній) функції втрат по цим параметрам. Величина змін визначається швидкістю навчання, яка контролює, наскільки великі зміни вносяться до

параметрів. Занадто велика швидкість може призвести до нестабільності, а занадто мала — до повільного зближення. Ось загальна формула градієнтного спуску [20]:

$$x_{i+1} = x_i - \alpha \nabla f(x_0), \quad (4)$$

де x_{i+1} — оновлені значення параметрів;

x_i — поточне значення параметрів;

α — швидкість навчання (англ. learning rate);

$\nabla f(x_0)$ — градієнт функції f в точці x_0;

i — індекс кроку.

Стохастичний градієнтний спуск

Стохастичний градієнтний спуск (SGD) є варіантом звичайного градієнтного спуску, який використовує менші вибірки з випадкових даних датасету для обчислення градієнта функції втрат [3]. На відміну від класичного градієнтного спуску, який використовує весь датасет для обчислення градієнта на кожному кроці, SGD обирає випадковий піднабір даних (часто один або кілька прикладів) для кожного кроку. Це дозволяє значно прискорити процес навчання та може допомогти уникнути локальних мінімумів за рахунок додаткової стохастичної природи оновлень параметрів.

3 РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ АНАЛІЗУ ДАНИХ

3.1 Опис предметної області

У контексті постійного розвитку сфери аналізу даних, ця галузь стає дедалі важливішою як для великих компаній, так і для індивідуальних осіб. Сьогодні не тільки великі корпорації, але й малі підприємства, стартапи та навіть окремі спеціалісти стикаються з необхідністю робити прикладний аналіз даних. Причиною цьому є потреба приймати обґрунтовані рішення на основі даних для підвищення ефективності бізнес-процесів, виявлення тенденцій та прогнозування майбутніх результатів.

Однак, багато хто з них не бажає або не має можливості опановувати складні спеціалізовані програми для аналізу даних. Вони прагнуть до простих у використанні інструментів, які дозволяють швидко та ефективно обробляти дані без глибоких технічних знань. Це створює попит на веб-застосунки, які є інтуїтивно зрозумілими, доступними з будь-якого пристрою та забезпечують потужний функціонал для аналізу даних.

Різні компанії та розробники програмного забезпечення вже зробили значні кроки у створенні інструментів для аналізу даних, які є доступними та зручними для користувачів різного рівня підготовки. Наприклад, SPSS, Statistica, Datapine та Datatab.net пропонують прості у використанні рішення, які поєднують потужний функціонал та інтерактивність.

Одним із ключових рішень є розробка інтуїтивно зрозумілих інтерфейсів. Це дозволяє користувачам проводити складний аналіз даних без необхідності вивчення програмування або глибоких технічних знань.

Крім того, ці інструменти пропонують широкий спектр статистичних методів та інструментів для візуалізації даних. Це включає можливості для проведення описової та інферентної статистики, регресійного аналізу, кластеризації та багато іншого. Візуалізація даних дозволяє користувачам

швидко отримувати уявлення про основні тенденції та закономірності у даних, що сприяє прийняттю обґрунтованих рішень.

Важливим аспектом є інтеграція з різними джерелами даних. Ці платформи дозволяють легко підключати та обробляти дані з різних джерел, таких як бази даних, файли CSV, Excel та інші. Це забезпечує гнучкість та зручність у роботі з даними з різних систем.

Останній аспект, на який варто звернути увагу, це можливість автоматизації звітності та створення інтерактивних дашбордів. Користувачі можуть генерувати звіти, які автоматично оновлюються при зміні даних, а також створювати дашборди для моніторингу ключових показників у режимі реального часу. Це робить процес аналізу даних більш ефективним та дозволяє швидко реагувати на зміни.

Таким чином, SPSS, Statistica, Datapine та Datatab.net зосереджені на наданні зручних, потужних та інтерактивних інструментів для аналізу даних, що дозволяє користувачам легко виконувати аналітичні задачі та приймати обґрунтовані рішення на основі даних незалежно від їхнього рівня підготовки.

3.2 Визначення вимог до веб-застосунку

Для створення ефективного та зручного інструменту для аналізу даних необхідно визначити чіткі вимоги до веб-застосунку. Ці вимоги поділяються на бізнес вимоги, які відображають загальні цілі та очікування користувачів, та функціональні вимоги, що визначають конкретні можливості і функції, які повинен забезпечувати застосунок.

Веб-застосунок повинен мати наступні функціональні можливості:

- Забезпечити можливість завантаження датасетів у форматах CSV, XSLX, PARQUET.
- Реалізувати функціонал для зберігання завантажених датасетів у базі даних.

- Інтегрувати базу даних для надійного зберігання завантажених датасетів.
- Забезпечити механізми для швидкого доступу до даних у базі даних.
- Реалізувати інтерфейс для детального перегляду датасетів у форматі таблиці.
- Забезпечити можливість фільтрування, сортування та пошуку даних у таблиці.
- Інтегрувати функціонал для перегляду даних у вигляді графіків.
- Підтримати проведення лінійної та поліноміальної регресії.
- Реалізувати інтерфейс для вибору змінних та проведення регресійного аналізу.
- Надати результати аналізу у вигляді графіків та інформацію з основними статистичними показниками.

Ці вимоги були сформульовані на основі висновків, отриманих від аналізу ринку та вивчення програм-аналогів. Такий підхід дозволив визначити ключові потреби користувачів та забезпечити простий й зручний функціонал для ефективного аналізу даних.

Для кращого розуміння бізнес та функціональних вимог, будь ласка, дивіться діаграму прецедентів (див. Рисунок *7*).

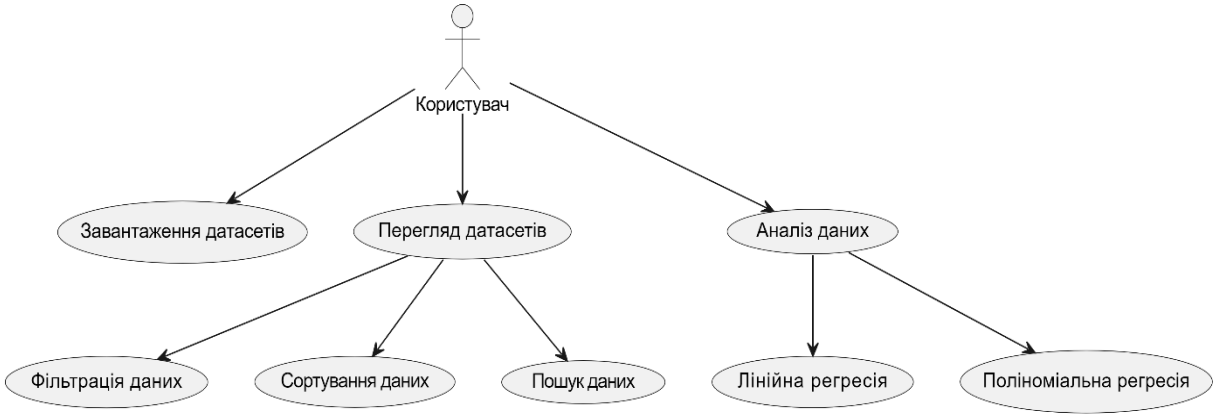

Рисунок *7 — Діаграма прецедентів*

3.3 Проєктування архітектури веб-застосунку

Застосунок було вирішено розбити на дві частини: серверну (бекенд) та клієнтську (фронтенд). Такий підхід забезпечує низку переваг як для розробників, так і для користувачів.

Для розробників розподіл на бекенд та фронтенд дозволяє працювати над окремими частини застосунків незалежно, що значно спрощує підтримку та подальше розширення функціоналу. Незважаючи на те, що і серверна, і клієнтська частини знаходяться в одному моноліті, такий підхід все ж дозволяє розробникам спеціалізуватися на своїх конкретних областях, покращуючи ефективність і якість розробки. Крім того, це дає можливість більш легко масштабувати застосунок і інтегрувати нові функції, зберігаючи при цьому організовану структуру коду.

Для користувачів такий підхід забезпечує кілька ключових переваг. По-перше, клієнтська частина застосунку, реалізована на фронтенді, надає більш інтерактивний і швидкий відгук на дії користувача, що покращує загальний досвід взаємодії з застосунком. По-друге, можливість оптимізації інтерфейсу для різних пристроїв (мобільних, десктопних) дозволяє користувачам комфортно працювати з застосунком з будь-якої платформи. Нарешті, розподіл на серверну та клієнтську частини сприяє безпеці даних, оскільки критичні операції та обробка даних виконуються на сервері, що мінімізує ризики витоку інформації.

Для бекенду було обрано багатошарову архітектуру (див. Рисунок *8*), яка передбачає розбиття функціоналу по папках відповідно до основних компонентів системи. Це дозволить забезпечити модульність, легкість у підтримці та розширюваність системи.

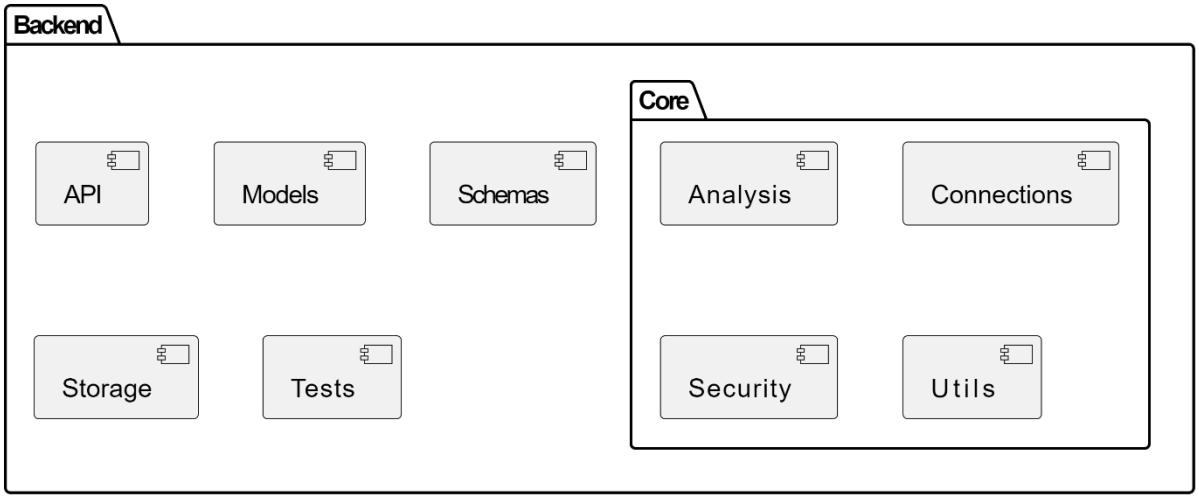

Рисунок *8 — Схема серверної частини*

Опис шарів архітектури:

- *API*: Обробляє запити від користувачів.
- *Models*: Визначає моделі таблиць бази даних.
- *Schemas*: Відповідає за валідацію та серіалізацію даних, визначає структуру і типи даних для запитів та відповідей.
- *Storage*: Сховище даних.
- *Tests:* Все, що пов’язане з тестуванням.
- *Core*:
    - *Analysis*: Обробка та аналіз даних.
    - *Connections*: Взаємодія з зовнішніми сервісами та ресурсами.
    - *Security*: Аутентифікація, авторизація та захист від загроз.
    - *Utils*: Допоміжні функції та інструменти.

Для фронтенду було вирішено використати архітектуру FSD (Feature Sliced Design) (див. Рисунок *9*), яка базується на бізнес-сутностях. Ця архітектура дозволяє чітко розділити функціонал на окремі модулі, що полегшує розробку, тестування та масштабування фронтенд-частини застосунку. Кожна бізнес-сутність має свої власні компоненти, що дозволяє забезпечити високу організованість та зручність у розробці інтерфейсу користувача.

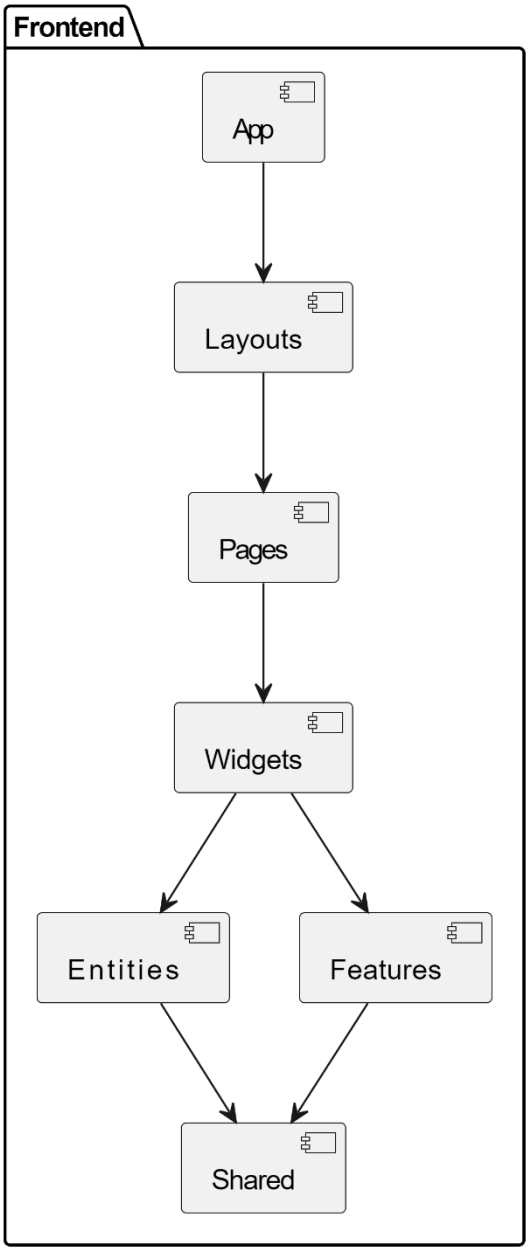

Рисунок *9* — *Схема клієнтської частини*

Опис шарів архітектури:

- *App*: Головний компонент застосунку, відповідає за ініціалізацію та налаштування загальних аспектів застосунку, таких як плагіни, глобальні стилі та контексти.
- *Layouts*: Шар відповідає за розміщення основних структурних компонентів сторінок, таких як заголовки, навігаційні панелі та футери.
- *Pages*: Сторінки, які складаються з різних віджетів та компонентів. Кожна сторінка відповідає за відображення певного контенту або

функціональності і об'єднує віджети для представлення цілісного вигляду.

- *Widgets*: Компоненти, що використовуються на сторінках для відображення конкретної інформації чи взаємодії з користувачем.
- *Entities*: Шар, що містить бізнес-логіку та моделі, які представляють основні сутності застосунку. Він забезпечує взаємодію з даними та їх обробку.
- *Features*: Функціональні можливості, які забезпечують конкретні дії або процеси в застосунку. Вони можуть використовувати сутності та віджети для реалізації своєї функціональності.
- *Shared*: Шар спільних ресурсів, який включає загальні компоненти, утиліти, стилі та константи, що використовуються в інших частинах застосунку.

3.4 Розробка серверної частини

Основні компоненти серверної частини включають API, моделі, схеми та зберігання даних, а також ядро системи, яке складається з аналізу, з'єднань, безпеки та утиліт.

API

Шар API (Application Programming Interface) є важливою частиною серверної архітектури, забезпечуючи взаємодію між фронтендом та серверною частиною застосунку. API цього веб-застосунку побудовано з використанням FastAPI, що забезпечує високу продуктивність та зручність розробки.

Авторизація в застосунку здійснюється за допомогою JWT (JSON Web Tokens). Користувачі проходять аутентифікацію через API, надаючи свої пошту та пароль. Після успішної аутентифікації сервер генерує два токени: токен доступу (access token) та токен для оновлення (refresh token). Токен доступу використовується для підтвердження подальших запитів і має обмежений час дії, забезпечуючи додатковий рівень безпеки. Токен для

оновлення має триваліший термін дії та використовується для отримання нового токену доступу без необхідності повторної аутентифікації, що покращує зручність для користувачів.

API дозволяє користувачам додавати, видаляти та змінювати свої датасети. Процес роботи з датасетами включає кілька ключових етапів:

- *Додавання датасетів*: Користувачі можуть завантажувати свої датасети через API. При завантаженні, датасет шифрується за допомогою бібліотеки *Cryptography*. Для шифрування використовується ключ, який зберігається на сервері у вигляді змінної оточення в *.env* файлі. Після шифрування датасет зберігається у директорію *storage* за унікальним ID. Цей ID також зберігається в базі даних в таблиці Dataset, що дозволяє легко ідентифікувати та керувати датасетами.
- *Видалення датасетів*: Користувачі можуть видаляти свої датасети через API. Після отримання відповідного запиту сервер знаходить датасет за унікальним ID, видаляє його з директорії *storage* та видаляє відповідний запис з бази даних.
- *Зміна датасетів*: Користувачі можуть оновлювати свої датасети через API. Цей процес включає завантаження нового датасету, який також шифрується та зберігається за новим унікальним ID, а старий датасет видаляється з системи.

В API також є інші можливості, які забезпечують додатковий функціонал для користувачів. Наприклад, можна отримати результати аналізу датасету, такі як лінійна та поліноміальна регресія. Ці запити дозволяють користувачам виконувати аналіз даних і отримувати результати у зручному форматі. Використовуючи ці аналітичні інструменти, користувачі можуть отримувати цінну інформацію та робити прогнози на основі своїх даних, що значно розширює можливості використання веб-застосунку для прийняття обґрунтованих рішень.

Інтерфейс з базою даних реалізовано за допомогою SQLAlchemy та SQLModel. SQLAlchemy використовується як ORM (Object-Relational

Mapping), що забезпечує зручну взаємодію з базою даних, виконання запитів та управління транзакціями. SQLModel використовується для кращої інтеграції з Pydantic моделями, що дозволяє легко валідувати та серіалізувати дані.

Ця структура API дозволяє забезпечити надійну і безпечну роботу з даними, а також забезпечує гнучкість і масштабованість системи.

Моделі

Шар Models (Моделі) містить в собі моделі сутностей бази даних, які ініційовані за допомогою SQLModel та Pydantic класів.

Кожна з моделей має такі поля:

- id — унікальний цілочисельний ідентифікатор.
- created_at — дата та час створення запису.
- updated_at — дата та час оновлення запису.

Також кожна модель має помічні методи для швидкого збереження, редагування та видалення запису, що прискорює написання логіки роботи з базою даних та уникає повторення коду.

Кожна модель розроблена з урахуванням потреб у зберіганні та швидкому доступі до даних, що забезпечує ефективну роботу з великими обсягами інформації та швидкий відгук системи.

Ця структура моделей бази даних дозволяє ефективно управляти даними, підтримувати їхню цілісність та забезпечувати високий рівень продуктивності та масштабованості системи.

Схеми

Шар Schemas (Схеми) у серверній частині веб-застосунку відповідає за валідацію та серіалізацію даних, які проходять через API. Тут зберігаються загальні схеми у вигляді Pydantic моделей, що забезпечує коректність і безпеку обміну даними між клієнтом і сервером.

Основна функція цього шару полягає у визначенні структури і типів даних, які очікуються в запитах та відповідях API. Pydantic моделі використовуються для перевірки вхідних даних, що надходять від

користувачів або інших сервісів, а також для серіалізації вихідних даних, що відправляються клієнту. Це дозволяє уникнути помилок і забезпечити, щоб всі дані, що обробляються застосунком, відповідали очікуваній структурі та типам.

Ядро

Шар Core (Ядро) є центральною частиною серверної архітектури, де знаходиться вся основна логіка бекенду. Він забезпечує виконання ключових функцій застосунку та складається з декількох підшарів, кожен з яких відповідає за специфічні аспекти роботи системи:

- *Analysis*: У цьому підшарі розміщуються класи і функції для роботи з аналізом даних. Тут реалізовані алгоритми лінійної та поліноміальної регресії, а також інші помічні функції, необхідні для виконання аналітичних завдань.
- *Connections*: Підшар, відповідальний за підключення до бази даних PostgreSQL. Він містить логіку для встановлення та управління з'єднаннями з базою даних.
- *Security*: У цьому підшарі реалізована логіка роботи з шифруванням даних та виконання доступу до API. Він включає механізми аутентифікації та авторизації користувачів, шифрування і дешифрування даних, а також захист від загроз і атак. Це забезпечує безпечну роботу застосунку і захист даних користувачів.
- *Utils*: Підшар, що містить різноманітні допоміжні функції та утиліти, які можуть використовуватися будь-де у бекенді. Це можуть бути функції для логування, обробки помилок, роботи з файлами, форматування даних та інші загальні утиліти, які полегшують розробку і підтримку коду.

Сховище

Шар Storage (Сховище) містить в собі зашифровані файли, які були завантажені користувачами. Файли датасетів зберігаються у форматі CSV з унікальною назвою, яка використовується у базі даних для зв’язування

реального файлу з записом у PostgreSQL. Такий підхід допомагає швидко ідентифікувати та отримувати потрібні файли для обробки або аналізу.

Зашифроване зберігання файлів забезпечує безпеку даних, захищаючи їх від несанкціонованого доступу. Завдяки цьому підходу користувачі можуть бути впевнені в безпеці та цілісності своїх даних, а розробники отримують інструменти для швидкого доступу до необхідної інформації та її подальшої обробки.

Тести

Шар Tests (Тести) містить всі необхідні компоненти для забезпечення якісного та ефективного тестування веб-застосунку. Він включає тести, фікстури та тестові публічні датасети, на яких проводиться тестування.

Тести покривають регресійні алгоритми. Вони перевіряють коректність і надійність роботи лінійної та поліноміальної регресії. Фікстури забезпечують підготовку тестових даних та об'єктів перед виконанням тестів, що дозволяє уникнути дублювання коду і забезпечити чистоту та ізольованість тестового середовища.

Тестові публічні датасети зберігаються у форматі CSV і містять реальні або згенеровані дані. Вони використовуються для перевірки коректності роботи регресійних алгоритмів. Такий підхід дозволяє ретельно перевірити роботу цих алгоритмів, забезпечуючи високу якість та надійність прогнозів та аналізу даних у веб-застосунку.

3.6 Реалізація механізмів аналізу даних

Реалізація механізмів аналізу даних є ключовим етапом у розробці веб-додатку, оскільки саме на цьому етапі забезпечується можливість обробки та інтерпретації даних. У цьому розділі розглядаються методи та алгоритми, які використовуються для аналізу даних, а також способи візуалізації отриманих результатів.

Для реалізації функцій аналізу даних у нашому додатку застосовуються такі алгоритми: лінійна регресія, поліноміальна регресія та градієнтний спуск. Лінійна регресія дозволяє моделювати прості залежності між змінними, тоді як поліноміальна регресія використовується для більш складних, нелінійних моделей. Градієнтний спуск застосовується як метод оптимізації для навчання моделей, забезпечуючи мінімізацію функції помилки.

Алгоритми аналізу даних

Алгоритми аналізу даних є основою для отримання інформативних результатів та прогнозів на основі вхідних даних.

Реалізація алгоритмів написана на мові Python, тому для швидкої обробки рядків даних було вирішено використовувати бібліотеку Numpy.

Для стандартизації коду створено базовий інтерфейс для класів регресійного аналізу під назвою «BaseRegression», код якого наведено у Лістинг *1*.

Лістинг *1 Базовий клас для регресійних моделей*

```
from abc import abstractmethod, ABC
from typing import Optional

import numpy as np

from core.analysis.utils import LearningRateTooHigh

DEFAULT_ALPHA = 0.01  # learning rate
DEFAULT_ITERATIONS = 100_000  # maximum number of iterations
DEFAULT_THRESHOLD = 1e-10  # minimum step size
MAX_DATASET_SIZE = 100_000  # maximum number of data points
to not use batch gradient descent

class BaseRegression(ABC):
```

```
    def __init__(self, x_values: np.array, y_values: np.array):
        self._x_values = x_values
        self._y_values = y_values

    def _get_data(self, batch_size: Optional[int] = None) -> tuple[np.array, np.array]:
        if len(self._x_values) <= MAX_DATASET_SIZE or batch_size is None:
            return self._x_values, self._y_values
        indices = np.random.choice(len(self._x_values), batch_size, replace=False)
        return self._x_values[indices], self._y_values[indices]

    @abstractmethod
    def _fit(self, alpha: float, iterations: int, threshold: float):
        raise NotImplementedError

    @abstractmethod
    def _predict(self, x: np.array) -> np.array:
        raise NotImplementedError

    @abstractmethod
    def _mean_squared_error(self) -> float:
        raise NotImplementedError

    def fit(self, alpha: Optional[float] = None, iterations: int = DEFAULT_ITERATIONS, threshold: float = DEFAULT_THRESHOLD):
        def attempt_fit(_alpha: float) -> bool:
            try:
```

```
                self._fit(_alpha, iterations, threshold)
                return True
            except LearningRateTooHigh:
                return False

        if alpha is None:
            _alpha = DEFAULT_ALPHA
            while _alpha >= threshold:
                if attempt_fit(_alpha):
                    return
                _alpha /= 10
            raise LearningRateTooHigh
        else:
            self._fit(alpha, iterations, threshold)

    def predict(self, x: np.array) -> np.array:
        return self._predict(x)

    def error(self) -> float:
        return self._mean_squared_error()
```

Перед ініціалізацією класу прописані константи:

- *DEFAULT_ALPHA* — визначає початкове значення alpha, якщо воно не було передано користувачем.
- *DEFAULT_ITERATIONS* — максимальна кількість ітерацій проходження алгоритмом оптимізації.
- *DEFAULT_THRESHOLD* — мінімальний крок алгоритму оптимізації.
- *MAX_DATASET_SIZE* — максимальний розмір датасету для опрацювання градієнтним спуском, якщо набір даних перевищує це число буде використовуватися стохастичний градієнтний спуск.

Методи базового класу:

1. Конструктор складається з двох масивів, які є залежними стовпцями даних датасету.
2. Метод *_get_data* — повертає набір даних, необхідних на кожному кроці градієнтного спуску, повертає випадкову підмножину даних у випадку використання стохастичного градієнтного спуску.
3. Метод *_fit* — захищений абстрактний метод, який реалізує знаходження лінії регресії.
4. Метод *_predict* — захищений абстрактний метод, який приймає масив вхідних даних та повертає масив вихідних, вже прогнозованих даних, основаних на даних, що були отримані під час виконання методу fit.
5. Метод *_mean_squared_error* — захищений абстрактний метод, який повертає середньоквадратичну похибку (MSE).
6. Метод *fit* — публічна обгортка методу _fit, яка реалізує автоматичний підбір змінної alpha, яка вказує на швидкість навчання.
7. Метод *predict* — публічна обгортка методу *_predict*.
8. Метод *error* — викликає метод *_mean_squared_error*.

Цей клас задає інтерфейс для основних методів, щоб обидва класи регресії були зручними у використанні та не повторювали фрагменти коду одне одного.

Лінійна регресія

Реалізація класу лінійної регресії під назвою «LinearRegression», що успадковує базовий клас «BaseRegression», наведено у Лістинг *2*.

Лістинг *2 Реалізація класу лінійної регресії*

```
import numpy as np

from core.analysis.regressions.base_regression import
  BaseRegression
from core.analysis.utils import LearningRateTooHigh
```

```
class LinearRegression(BaseRegression):
    """
    A class used to perform linear regression using gradient
  descent.
    """

    def __init__(self, x_values: np.array, y_values:
  np.array):
        """
        Initializes the LinearRegression with x and y values.
        :param x_values: The input x values.
        :param y_values: The input y values.
        """
        super().__init__(x_values, y_values)
        self._intercept = 0  # initial y-intercept of the
  fitting line
        self._theta = 1  # initial slope of the fitting line
        # Normalize x and y values
        self._x_mean = np.mean(self._x_values)
        self._x_std = np.std(self._x_values)
        self._y_mean = np.mean(self._y_values)
        self._y_std = np.std(self._y_values)

        self._x_values = (self._x_values - self._x_mean) /
  self._x_std
        self._y_values = (self._y_values - self._y_mean) /
  self._y_std

    def _fit(self, alpha: float, iterations: int, threshold:
  float):
        """
        Fits the linear regression model to the data using
  gradient descent.
```

```
        :param alpha: The learning rate, defaults to
DEFAULT_ALPHA.
        :param iterations: The maximum number of iterations,
defaults to DEFAULT_ITERATIONS.
        :param threshold: The minimum step size, defaults to
DEFAULT_THRESHOLD.
        """
        self._intercept = 0
        self._theta = 1

        for _ in range(iterations):
            x_values, y_values = self._get_data(int(0.1 *
len(self._x_values)))

            position_gradient = -2 * np.sum(y_values -
(self._intercept + self._theta * x_values))
            slope_gradient = -2 * np.sum(x_values * (y_values
- (self._intercept + self._theta * x_values)))

            new_intercept = self._intercept - alpha *
position_gradient
            new_theta = self._theta - alpha * slope_gradient

            if np.isnan(new_intercept) or np.isnan(new_theta)
or np.isinf(new_intercept) or np.isinf(new_theta):
                raise LearningRateTooHigh

            # Check if the update is below the threshold
            if abs(new_intercept - self._intercept) <
threshold and abs(new_theta - self._theta) < threshold:
                break
            self._intercept = new_intercept
            self._theta = new_theta
```

```
  def _predict(self, x_values: np.array) -> np.array:
      """
      Predicts the y values for a given array of x values.
      :param x_values: An array of input x values.
      :return: An array of predicted y values.
      """
      x_values_standardized = (x_values - self._x_mean) /
self._x_std
      y_values_norm = self._theta * x_values_standardized +
self._intercept
      y_values = y_values_norm * self._y_std + self._y_mean
      return y_values

  def _mean_squared_error(self) -> float:
      """
      Mean squared error of the model.
      :return: The mean squared error.
      """
      y_pred_norm = self._theta * self._x_values +
self._intercept
      y_pred = y_pred_norm * self._y_std + self._y_mean
      mse = np.mean((self._y_values * self._y_std +
self._y_mean - y_pred) ** 2)
      return mse
  @property
  def theta(self):
      return self._theta * self._y_std / self._x_std

  @property
  def intercept(self):
      return self._intercept
```

Методи класу:

1. Конструктор реалізує ініціалізацію змінних та стандартизацію (нормалізацію) вхідних даних.
2. Метод *_fit* виконує пошук лінії, що найкраще відповідає вхідним даним, за допомогою двох змінних: *theta* (нахил) та *intercept* (положення по осі Y). Ці дві змінні калібруються за допомогою градієнтного спуску та методу найменших квадратів.
3. Метод *_predict* повертає прогнозовані дані в початковому форматі (виконується зворотне масштабування) за відкаліброваними значеннями змінних *theta* та *intercept*.
4. Метод *_mean_squared_error* повертає середньоквадратичну похибку.

Приклад виконання лінійної регресії на відкритому датасеті годин навчання студентом та бали які він отримував (див. Рисунок *10*).

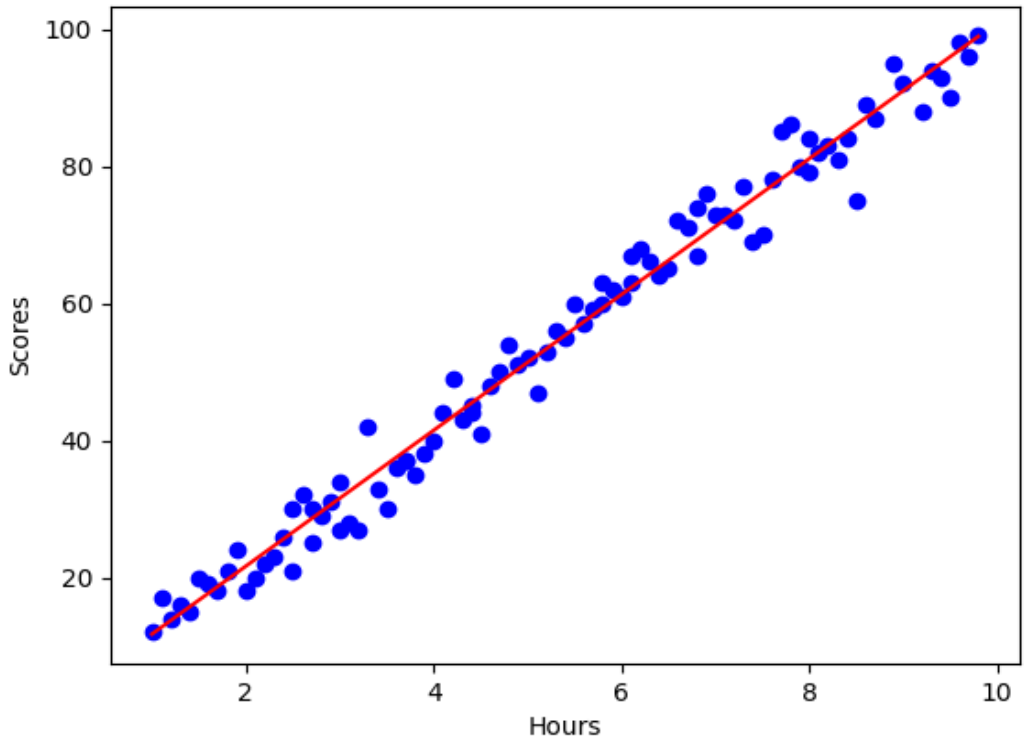

Рисунок *10 — Візуалізація лінійної регресії з використанням Matplotlib*

Поліноміальна регресія

Реалізація класу поліноміальної регресії схожа на реалізацію лінійної регресії, оскільки обидві моделі успадковують базовий клас «BaseRegression». Основною відмінністю є формування вихідних даних: у випадку поліноміальної регресії до кожної змінної додаються додаткові члени, підняті до різних ступенів. Реалізація класу поліноміальної регресії наведена у Лістинг *3*.

Лістинг *3 Реалізація класу поліноміальної регресії*

```
import numpy as np

from core.analysis.regressions.base_regression import
BaseRegression
from core.analysis.utils import LearningRateTooHigh

class PolynomialRegression(BaseRegression):
    """
    A class used to perform polynomial regression using
gradient descent.
    """

    def __init__(self, x_values: np.array, y_values:
np.array, *, degree: int = 2):
        """
        Initializes the PolynomialRegression with x and y
values and the polynomial degree.
        :param x_values: The input x values.
        :param y_values: The input y values.
        :param degree: The degree of the polynomial,
defaults to 2.
        """
        super().__init__(x_values, y_values)
```

```
        self._degree = degree
        self._coefficients = np.zeros(degree + 1)  #
initial coefficients for the polynomial

        # Normalize x and y values
        self._x_mean = np.mean(self._x_values)
        self._x_std = np.std(self._x_values)
        self._y_mean = np.mean(self._y_values)
        self._y_std = np.std(self._y_values)

        self._x_values = (self._x_values - self._x_mean) /
self._x_std
        self._y_values = (self._y_values - self._y_mean) /
self._y_std

    def _polynomial_features(self, x: np.array) ->
np.array:
        """
        Generates polynomial features up to the specified
degree.
        :param x: The input x values.
        :return: The polynomial features of the input x
values.
        """
        return np.vstack([x ** i for i in
range(self._degree + 1)]).T

    def _fit(self, alpha: float, iterations: int,
threshold: float):
        """
        Fits the polynomial to the data using gradient
descent.
```

```
        :param alpha: The learning rate, defaults to
DEFAULT_ALPHA.
        :param iterations: The number of iterations for
gradient descent, defaults to 1000.
        :param threshold: The minimum step size, defaults
to DEFAULT_THRESHOLD.
        """
        self._coefficients = np.zeros(self._degree + 1)
        old_coefficients = self._coefficients.copy()

        for _ in range(iterations):
            x_values, y_values = self._get_data(int(0.1 *
len(self._x_values)))
            x_poly = self._polynomial_features(x_values)

            predictions = x_poly.dot(self._coefficients)
            errors = y_values - predictions
            gradients = -2 * x_poly.T.dot(errors)
            self._coefficients -= alpha * gradients

            if np.isnan(self._coefficients).any() or
np.isinf(self._coefficients).any():
                raise LearningRateTooHigh

            # Check if the update is below the threshold
            if np.all(np.abs(old_coefficients -
self._coefficients) < threshold):
                break

            old_coefficients = self._coefficients.copy()

    def _predict(self, x: np.array) -> np.array:
        """
```

```
        Predicts the y values for the given x values.
        :param x: The input x values.
        :return: The predicted y values.
        """
        x = (x - self._x_mean) / self._x_std
        x_poly = self._polynomial_features(x)
        y_norm = x_poly.dot(self._coefficients)
        y = y_norm * self._y_std + self._y_mean
        return y

    def _mean_squared_error(self) -> float:
        """
        Mean squared error of the model.
        :return: The mean squared error.
        """
        x_poly = self._polynomial_features(self._x_values)
        y_pred_norm = x_poly.dot(self._coefficients)
        y_pred = y_pred_norm * self._y_std + self._y_mean
        y_true = self._y_values * self._y_std +
self._y_mean
        return np.mean((y_true - y_pred) ** 2)

    @property
    def coefficients(self) -> np.array:
        coefficients = self._coefficients * self._y_std /
self._x_std ** np.arange(self._degree + 1)
        return coefficients
```

Методи класу:

1. Конструктор реалізує ініціалізацію змінних та стандартизацію (нормалізацію) вхідних даних.

2. Метод *_fit* виконує пошук лінії, що найкраще відповідає вхідним даним, за допомогою градієнтного спуску, калібруючи коефіцієнти.
3. Метод *_predict* повертає прогнозовані дані в початковому форматі (виконується зворотне масштабування) за відкаліброваними коефіцієнтами.
4. Метод *_mean_squared_error* повертає середньоквадратичну похибку (MSE).

Приклад виконання поліноміальної регресії другого порядку на відкритому датасеті продажів морозива та температури середи (див. Рисунок *11*).

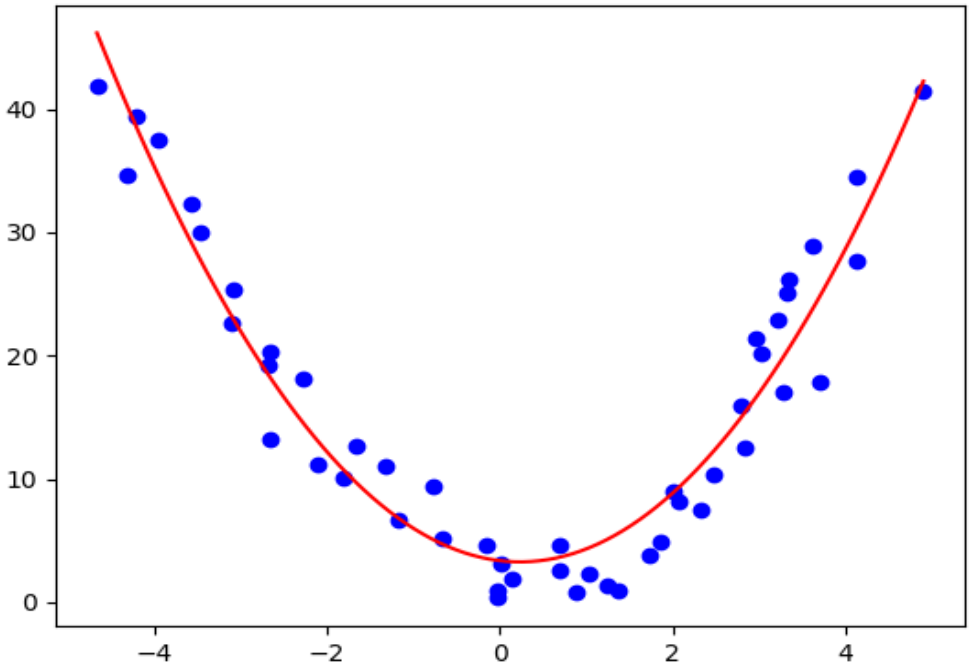

Рисунок *11 — Візуалізація поліноміальної регресії другого порядку*

Візуалізація результатів аналізу

Візуалізація результатів аналізу є ключовим елементом у процесі обробки даних, оскільки вона дозволяє легко і зрозуміло інтерпретувати складні числові результати. Графічне представлення даних надає можливість виявляти закономірності, тенденції та аномалії, які можуть бути приховані у числових таблицях. Особливо важливо відображати результати регресійного

аналізу у вигляді графіків, щоб наочно побачити залежності між змінними та точність моделей.

Для лінійної та поліноміальної регресії графік зазвичай представляється у вигляді точкової діаграми (англ. scatter plot), де точки відповідають фактичним даним, а пряма лінія відображає передбачувані значення, що отримуються за допомогою моделі регресії. Це дозволяє легко оцінити, наскільки добре модель відповідає даним, та ідентифікувати будь-які відхилення або аномалії.

3.5 Розробка інтерфейсу користувача

У цьому розділі ми розглянемо дизайн та функціональні можливості кожної сторінки користувацького інтерфейсу веб-застосунку, який розроблений з використанням фреймворку Vue.js та UI Framework Vuetify. Vue.js забезпечує гнучкість і управління станом додатку, тоді як Vuetify додає набір готових компонентів та функціональних елементів інтерфейсу.

Для зручного використання запитів до API застосовується Vue Query, яка дозволяє ефективно працювати з асинхронними даними, кешуванням та синхронізацією стану між компонентами. Це забезпечує плавний досвід для користувачів, оскільки дані оновлюються в режимі реального часу.

Для відображення інтерактивних графіків використовується ChartJs, що інтегрується з Vue.js і дозволяє створювати динамічні графічні представлення даних. Це корисно для візуалізації результатів регресійного аналізу та інших даних.

Таким чином, комбінація Vue.js, Vuetify, Vue Query та ChartJs забезпечує створення багатофункціонального, інтуїтивно зрозумілого та естетичного користувацького інтерфейсу.

Сторінка авторизації

Сторінка авторизації призначена для того, щоб користувачі могли увійти до свого облікового запису. На сторінці знаходиться форма з полями вводу

електронної пошти та паролю (див. Рисунок *12*). Користувачі вводять свої данні та натискають кнопку «Sign in», щоб отримати доступ до облікового запису.

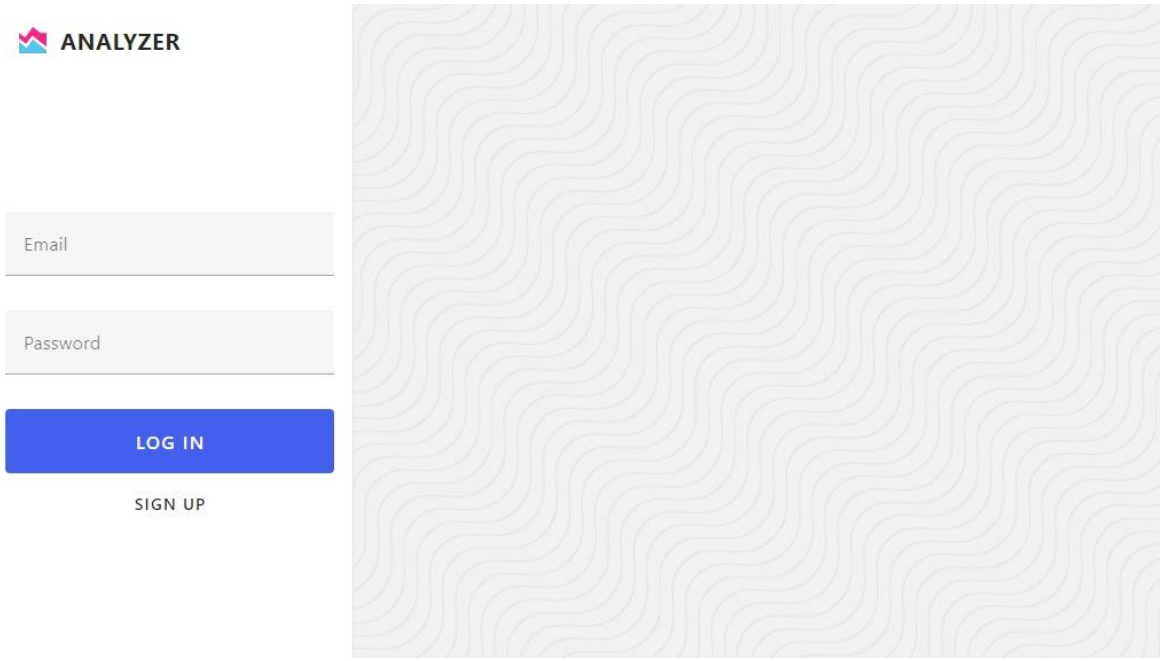

Рисунок *12* — *Сторінка авторизації*

Сторінка реєстрації

Сторінка реєстрації призначена для створення нового облікового запису користувачем. Вона містить форму, яка дозволяє ввести основні дані для реєстрації: електронну пошту, ім’я, пароль та підтвердження пароля (див. Рисунок *13*). Після заповнення всіх полів користувач натискає кнопку «Sign Up», щоб зареєструватися та отримати доступ до свого нового облікового запису. Окрім цього, на сторінці є кнопка для переходу на сторінку авторизації, якщо користувач вже має обліковий запис.

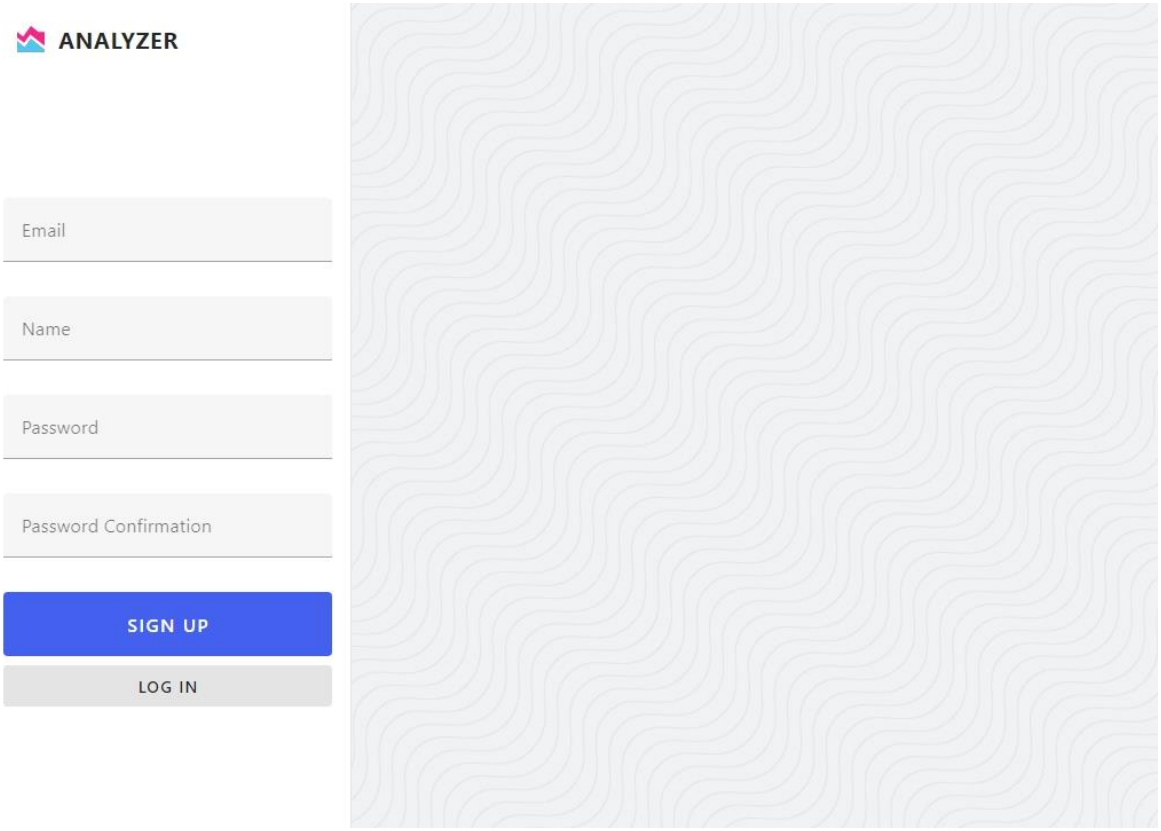

Рисунок *13 — Сторінка реєстрації*

Домашня сторінка

Домашня сторінка є головною сторінкою веб-застосунку. Вона вітає користувачів і надає їм початкову інформацію про застосунок. Вміст цієї сторінки динамічно змінюється залежно від того, чи авторизований користувач. Якщо користувач не авторизований буде повідомлення з пропозицією зареєструватися або увійти до системи, щоб почати використовувати застосунок (див. Рисунок *14*). В іншому випадку буде кнопка з пропозицією переглянути власні набори даних, яка веде користувача до відповідного розділу застосунку (див. Рисунок *15*).

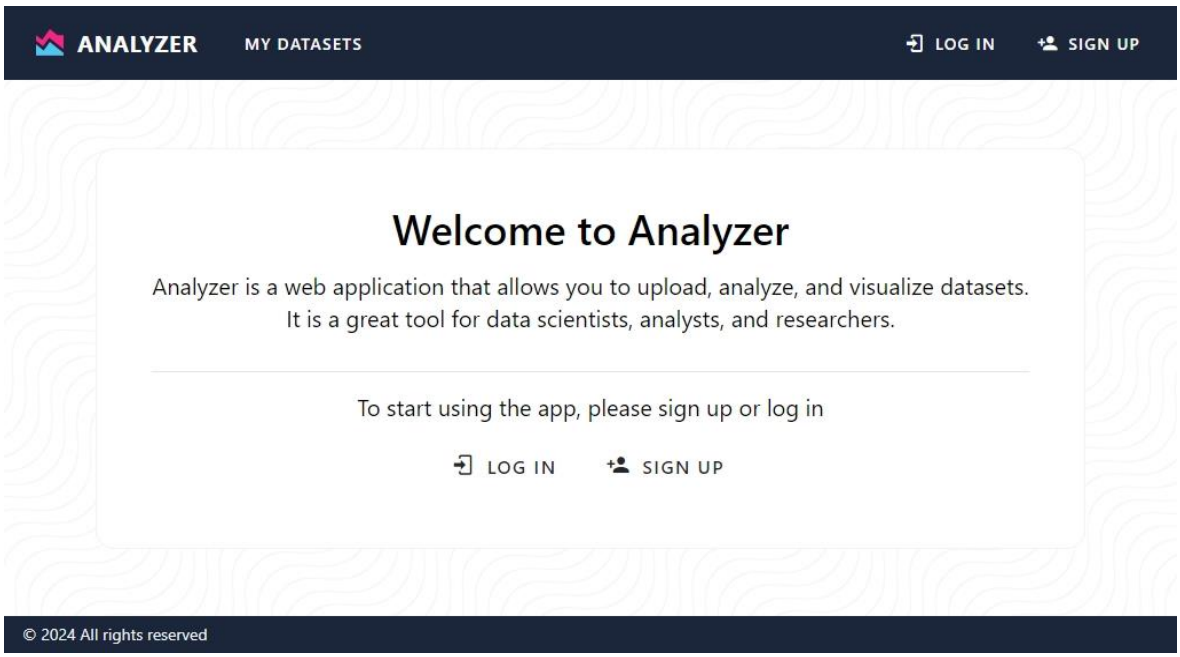

Рисунок *14* — *Домашня сторінка для неавторизованих користувачів*

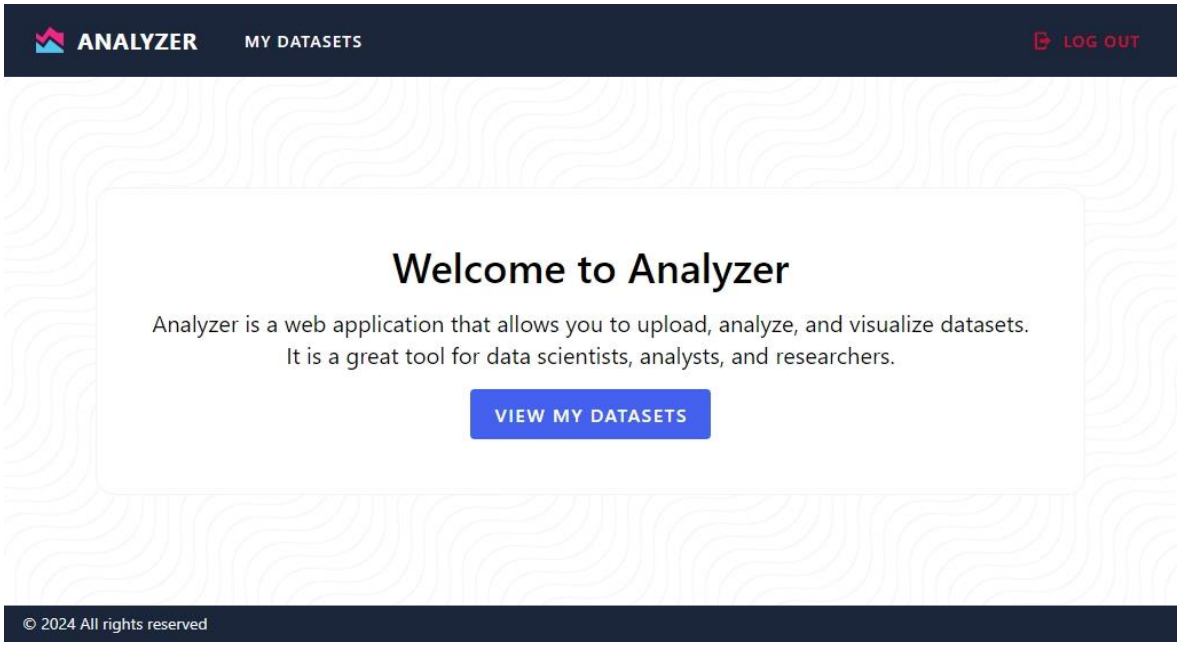

Рисунок *15* — *Домашня сторінка для авторизованих користувачів*

Сторінка датасетів

Сторінка «My Datasets» призначена для керування наборами даних користувача і включає зручний інтерфейс для перегляду, додавання та видалення наборів даних (див. Рисунок *16*). Вона забезпечує користувача всіма необхідними інструментами для роботи з даними в одному місці.

Біля заголовку сторінки розташована панель інструментів, яка містить дві основні кнопки: «Delete» та «Add dataset». Кнопка для видалення обраних наборів даних неактивна, поки користувач не вибере хоча б один набір для видалення, тоді як кнопка для додавання нового набору даних активна завжди і натискання на неї відкриває діалогове вікно з формою для завантаження нового набору.

Основну частину сторінки займає список наборів даних. Кожен набір даних представлений у вигляді окремого елемента списку, який включає назву набору, дату останнього оновлення, чекбокс (англ. checkbox) для вибору та кнопку для видалення. Назва набору даних є посиланням на детальну інформацію про цей набір. У нижній частині сторінки розміщена панель пагінації, яка забезпечує навігацію між сторінками наборів даних, якщо їх кількість перевищує одну сторінку.

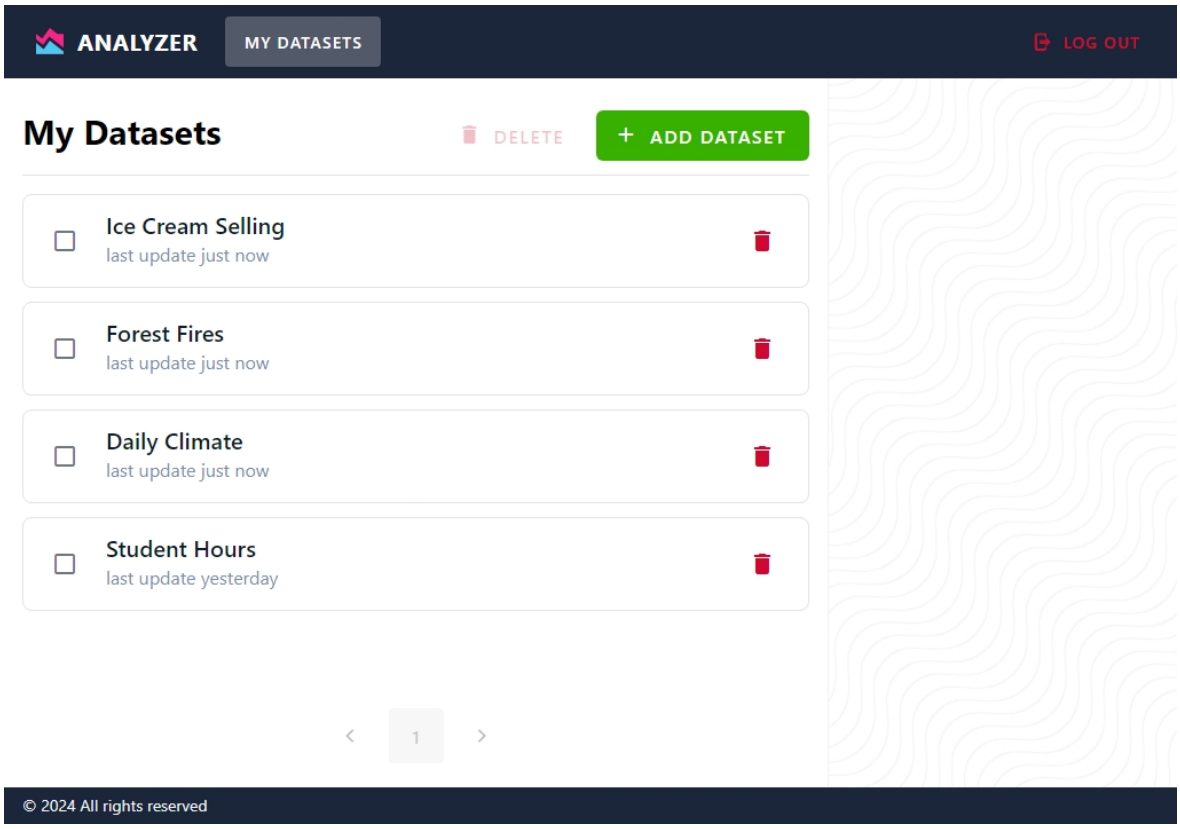

Рисунок *16 — Сторінка «My Datasets»*

Діалогове вікно для додавання нового набору даних містить форму, яка складається з кількох полів для введення (див. Рисунок *17*). Ця форма включає обов'язкове поле для введення назви, поле для опису, яке можна залишити порожнім, та поле для завантаження файлу з даними у форматах CSV, XLSX та інших.

У нижній частині діалогового вікна розташовані кнопки «Cancel» і «Create». «Cancel» скасовує дію та закриває вікно без збереження, а «Create» підтверджує додавання набору даних, відправляє запит до серверу і після обробки запиту закриває діалогове вікно, відразу відображаючи створений датасет у списку.

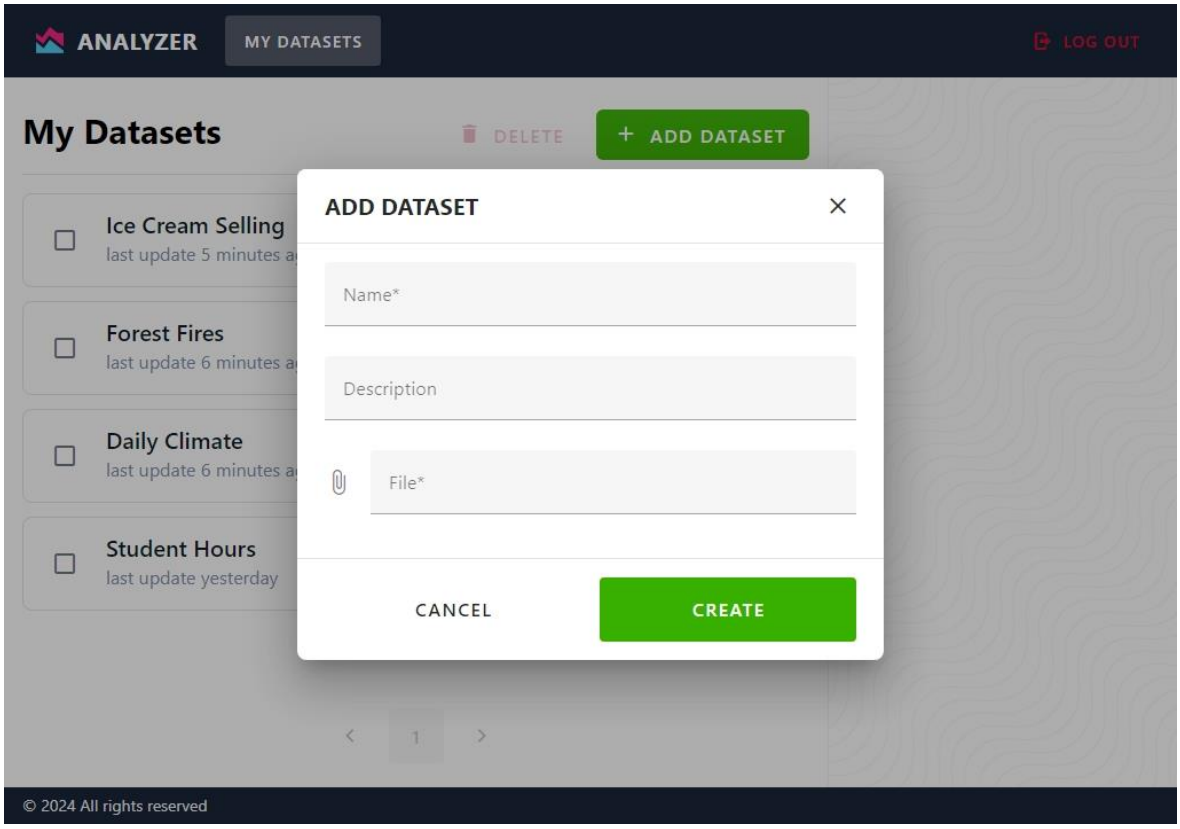

Рисунок *17 — Діалогове вікно додавання нового набору даних*

Сторінка детального перегляду набору даних

Сторінка детального перегляду набору даних в застосунку надає користувачу повний доступ до інформації про обраний набір даних і дозволяє

проводити різноманітні дії з цими даними (див. Рисунок *18*). Вона включає кілька ключових секцій для максимальної зручності користувача.

Основну частину сторінки займає таблиця з даними, де відображаються всі наявні дані набору у зручному для перегляду форматі. Користувач може змінювати кількість рядків на одній сторінці та користуватися пагінацією для навігації між сторінками даних. Кожна колонка в таблиці має кнопки сортування для впорядкування даних за зростанням або спаданням.

Нижче таблиці розташований графік, який відображає дані у візуальному форматі, допомагаючи користувачеві швидко оцінити тенденції та взаємозв'язки. Графік автоматично оновлюється відповідно до обраних даних і налаштувань.

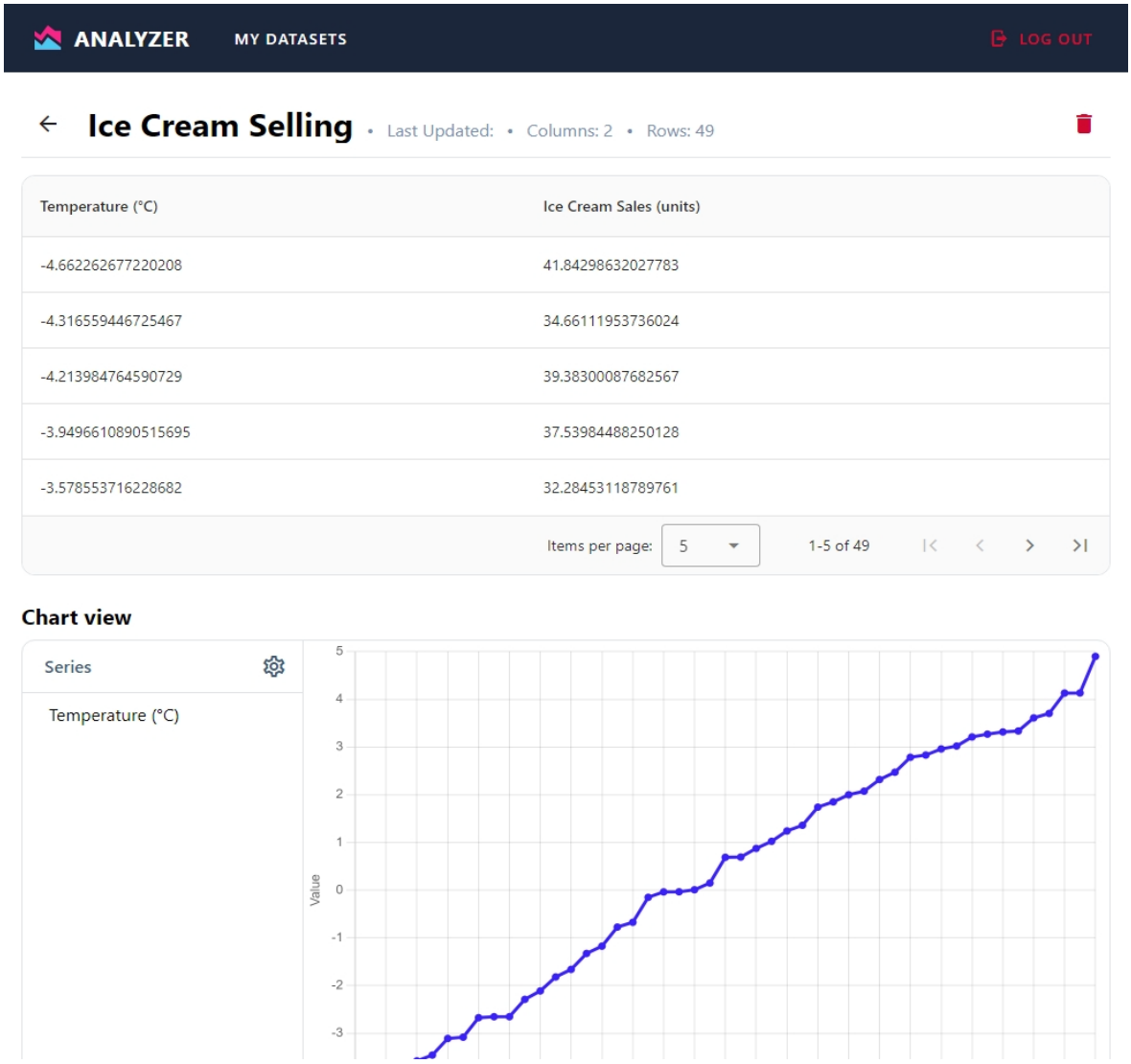

Рисунок *18 — Сторінка детального перегляду завантаженого набору даних*

Графік можна змінювати, натиснувши кнопку з шестернею. Після цього відкривається діалогове вікно з налаштуваннями графіка (див. Рисунок *19*).

Основний вміст складається з таблиці, де можна вибрати колонки, тип даних та стиль відображення графіка. Користувач може додавати нові колонки, змінювати тип даних на числовий або інший, а також налаштовувати стиль графіка, наприклад, вибирати лінійний графік.

У нижній частині діалогового вікна знаходяться дві кнопки: «Cancel» для скасування змін і закриття вікна без збереження, і «Save» для збереження змін і оновлення графіка відповідно до нових налаштувань. При натисканні «Save» відправляється запит до серверу з оновленими налаштуваннями графіка, після чого діалогове вікно закривається.

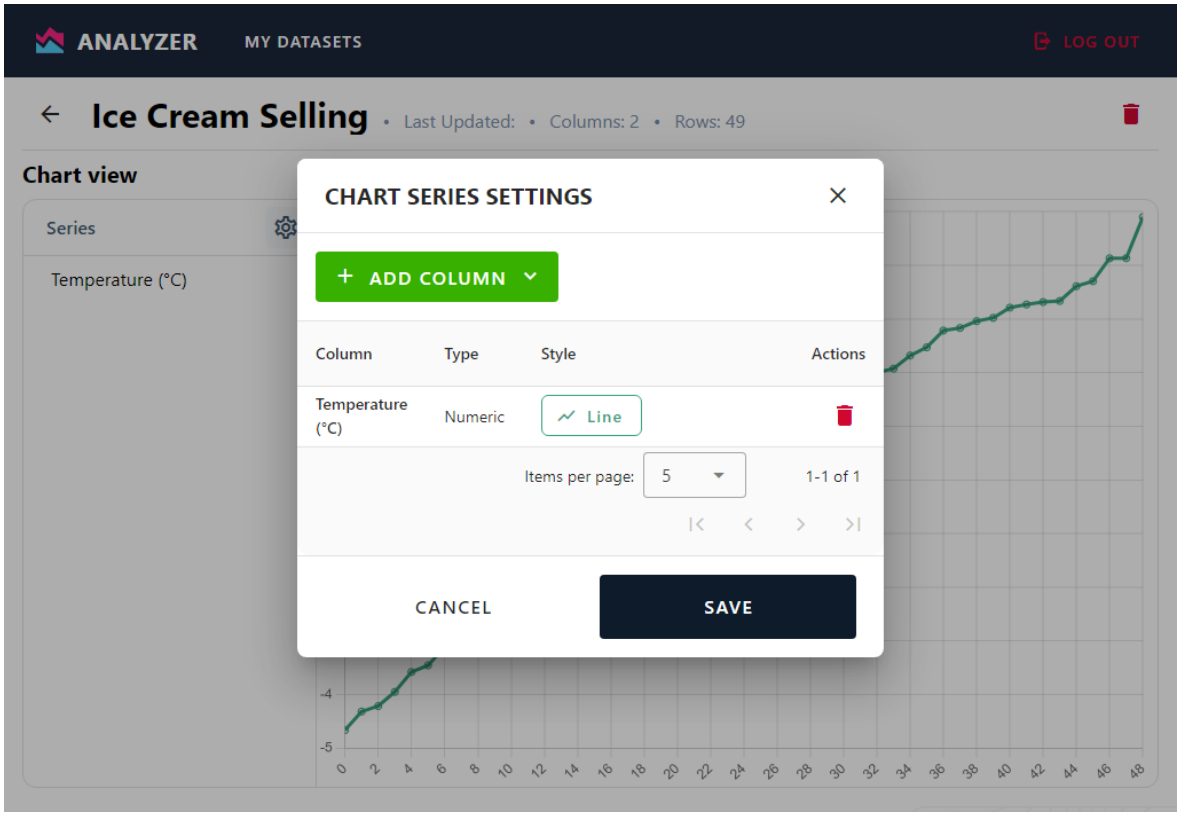

Рисунок *19* — *Діалогове вікно налаштування графіку перегляду*

Після графіку відображення набору даних знаходиться блок регресійного аналізу, в якому користувач може зробити регресійний аналіз за вибраними параметрами (див. Рисунок *20*).

Форма для введення параметрів складається з кількох ключових елементів. По-перше, поле для вибору режиму регресії, представлене у вигляді випадаючого списку, де користувач може обрати між «лінійною» та «поліноміальною» регресією. Лінійна регресія є вибраною за замовчуванням.

У випадку якщо вибрано

Далі є поле для вибору колонки X, яка буде використовуватися як незалежна змінна в аналізі. Це поле також реалізовано у вигляді випадаючого списку, де користувач може обрати потрібну колонку з наявного набору даних. Аналогічно, поле для вибору колонки Y дозволяє обрати колонку, яка буде використовуватися як залежна змінна в аналізі.

Після заповнення всіх необхідних полів користувач може натиснути кнопку «Calculate», яка запускає обчислення регресійного аналізу. Якщо деякі поля залишилися порожніми або не були обрані, на екрані з'являється повідомлення з помилкою, яке вказує, які поля необхідно заповнити для успішного виконання аналізу. Це робить процес інтуїтивно зрозумілим і допомагає користувачам уникнути помилок.

В результаті аналізу повертаються розраховані дані регресії, які можна інтерпретувати у лінію або криву, в залежності від обраного режиму регресії. Графік побудовано за допомогою бібліотеки Chart.js, що забезпечує високу якість візуалізації та інтерактивність. Точки, з яких побудовано графік, обчислюються на сервері і повертаються у вигляді масиву чисел. Цей масив включає значення залежної змінної (Y) для кожного значення незалежної змінної (X), що дозволяє точно відобразити регресійну модель на графіку. Користувач може побачити як вихідні дані, так і лінію або криву регресії, що допомагає наочно оцінити точність моделі.

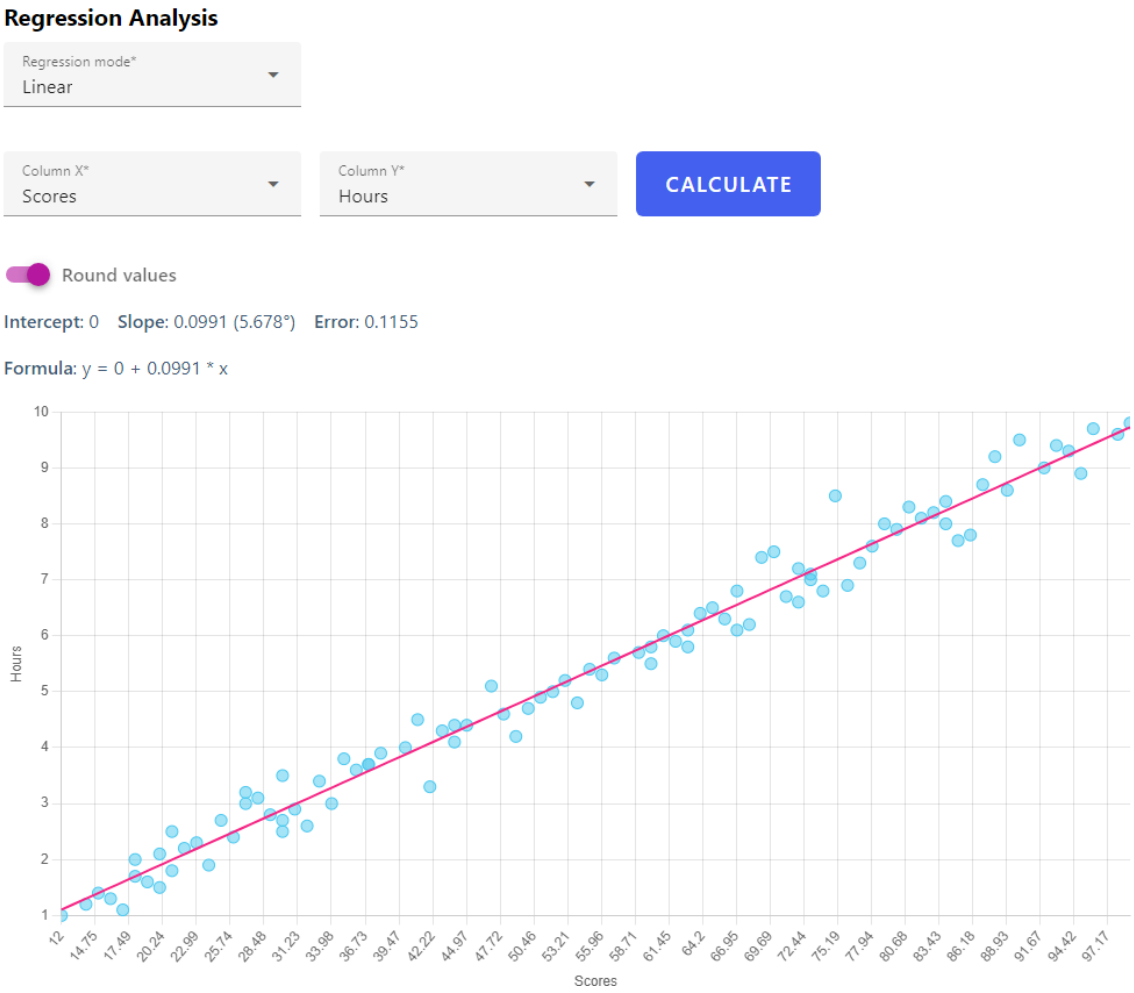

Рисунок *20* — *Результати лінійної регресії*

Розраховані дані регресії включають середньоквадратичну помилку та формулу лінії регресії, а також додаткові параметри, які залежать від вибраного типу регресії. У випадку лінійної регресії додатково повертаються значення intercept (перетин з віссю y) та slope (нахил). У випадку поліноміальної регресії відображається додатково лише ступінь регресії (див. Рисунок *21*). Всі повернені дані можна округлити до чотирьох знаків після коми, що спрощує перегляд та дає можливість краще інтерпретувати результати.

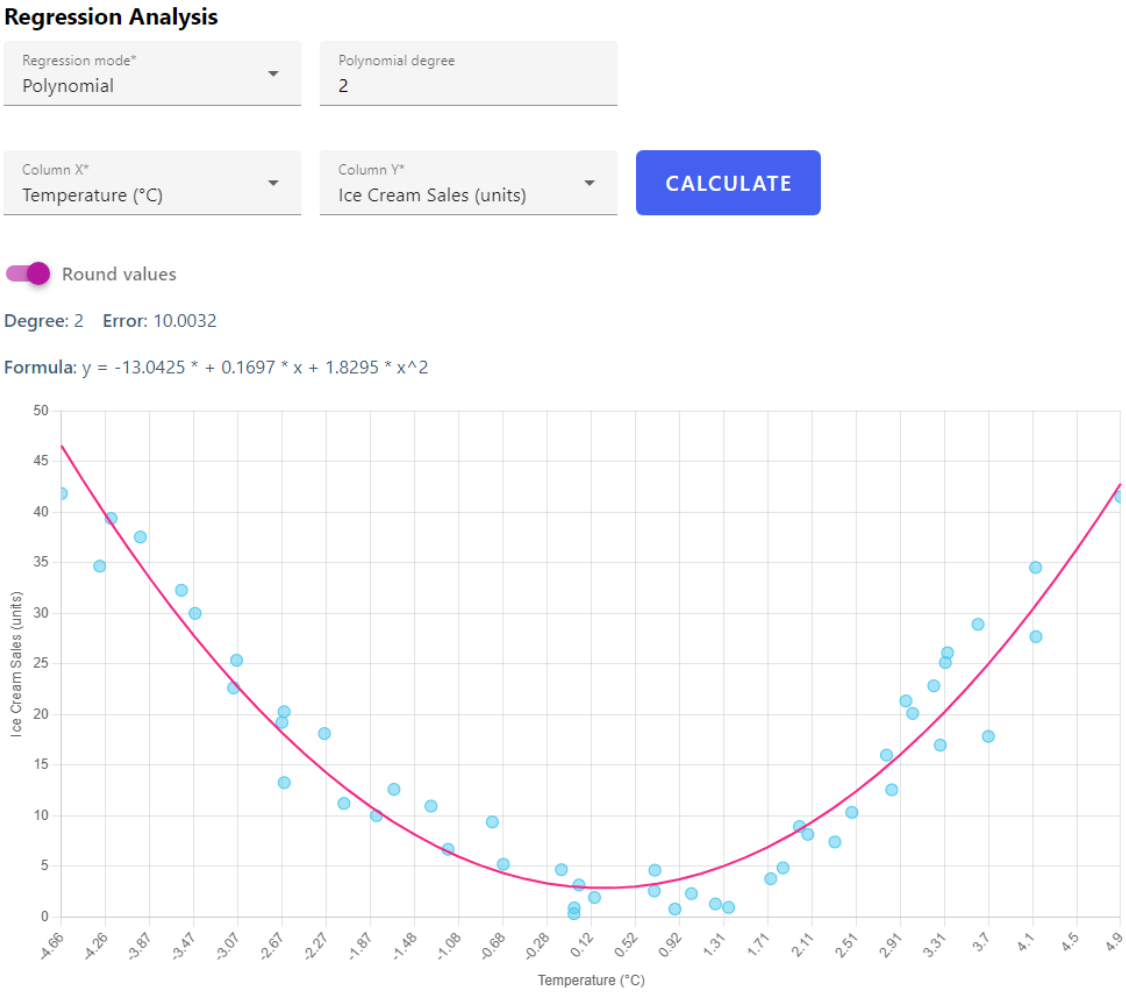

Рисунок *21 — Результати поліноміальної регресії другого порядку*

3.7 Тестування веб-застосунку

Тестування веб-застосунку є критично важливим етапом розробки, оскільки воно дозволяє забезпечити його надійність, ефективність та коректність роботи. Особливо це стосується алгоритмів регресійного аналізу, такі як лінійна та поліноміальна регресії, тому що вони відіграють ключову роль у застосунку. Тестування допомагає верифікувати коректність результатів, що є ключовим аспектом при використанні функцій регресійного аналізу для прогнозування та моделювання залежностей між змінними. Функції регресійного аналізу використовуються для прогнозування та моделювання залежностей між змінними, тому будь-яка помилка в реалізації

алгоритмів може призвести до некоректних прогнозів, що, в свою чергу, може мати значний негативний вплив на прийняття рішень на основі цих прогнозів.

В цьому проекті тестування виконується шляхом порівняння результатів власних реалізацій алгоритмів з результатами, отриманими за допомогою класів з бібліотеки scikit-learn. Цей підхід був обраний через популярність scikit-learn та доведену коректність її роботи. Такий метод дозволяє впевнитися в коректності реалізації алгоритмів: якщо результати збігаються, тест вважається пройденим. Це забезпечує надійність та точність розрахунків застосунку, гарантуючи, що реалізовані функції працюють відповідно до очікувань і не містять критичних помилок.

Перевагою тестування є також підтримка якості коду. Написання тестів сприяє покращенню структури та якості коду, оскільки в процесі тестування часто вимагається рефакторинг, щоб зробити код більш модульним та легким для тестування. Це спрощує майбутню підтримку та розвиток застосунку.

Ініціалізація тестів

Для забезпечення якості та коректності роботи веб-застосунку було вирішено використовувати бібліотеку *pytest*, яка є популярною і зручною у використанні для написання та виконання тестів в Python. Вона надає широкий набір інструментів для створення тестів, їхнього запуску та аналізу результатів.

Всі тести знаходяться в окремому шарі в бекенді, який включає необхідні дані та фікстури для тестування. У цьому шарі також є окрема директорія з датасетами, на яких саме і тестуються регресійні алгоритми. Така організація тестового середовища забезпечує легкість в управлінні тестовими даними та їхньою ізоляцією, що сприяє більш точному і надійному тестуванню.

Для прогонки тестів через датасети реалізовано фікстури. Вони дозволяють підготувати необхідні дані та об'єкти перед виконанням тестів. Це допомагає забезпечити повторюваність та ізольованість тестів, що робить

результати тестування більш надійними та точними. Реалізацію фікстур наведено у Лістинг *4*.

Лістинг *4 Реалізація фікстур для тестування*

```
import pandas as pd
import numpy as np

simple_dataset = (np.array([0, 1, 2, 3, 4]), np.array([0,
2, 4, 6, 8]))

ice_cream_selling =
pd.read_csv('datasets/ice_cream_selling.csv')
ice_cream_selling_dataset = (ice_cream_selling['Temperature
(°C)'].values, ice_cream_selling['Ice Cream Sales
(units)'].values)

student_hours_scores =
pd.read_csv('datasets/score_updated.csv')
student_hours_scores_dataset =
(student_hours_scores['Hours'].values,
student_hours_scores['Scores'].values)
```

Тестування лінійної регресії

Для перевірки правильності роботи власної реалізації алгоритму лінійної регресії результати порівнюються з результатами, отриманими за допомогою класу LinearRegression з бібліотеки scikit-learn, у Лістинг *5*.

Лістинг *5 Реалізація тестування лінійної регресії*

```
import numpy as np
import pytest
from sklearn.linear_model import LinearRegression as
SkLinearRegression
```

```
from backend.core.analysis.regressions.linear_regression
import LinearRegression
from fixtures import simple_dataset,
student_hours_scores_dataset

@pytest.fixture(params=[simple_dataset,
student_hours_scores_dataset])
def linear_regression_fixture(request):
    x_values, y_values = request.param
    regression = LinearRegression(x_values, y_values)
    regression.fit()

    sk_regression = SkLinearRegression()
    sk_regression.fit(x_values.reshape(-1, 1), y_values)

    return regression, sk_regression, x_values, y_values

def test_fit(linear_regression_fixture):
    regression, sk_regression, x_values, y_values =
linear_regression_fixture
    np.testing.assert_allclose(regression.intercept,
sk_regression.intercept_, rtol=1, atol=1)
    np.testing.assert_allclose(regression.theta,
sk_regression.coef_[0], rtol=1e-05, atol=1e-02)

def test_error(linear_regression_fixture):
    regression, sk_regression, x_values, y_values =
linear_regression_fixture
    error = regression.error()
    sk_error =
np.mean((sk_regression.predict(x_values.reshape(-1, 1)) -
y_values) ** 2)
```

```
    np.testing.assert_allclose(error, sk_error, rtol=1e-05,
atol=1e-02)

def test_predict(linear_regression_fixture):
    regression, sk_regression, x_values, y_values =
linear_regression_fixture
    predicted = regression.predict(x_values)
    sk_predicted = sk_regression.predict(x_values.reshape(-
1, 1))
    np.testing.assert_allclose(predicted, sk_predicted,
rtol=1e-05, atol=1e-02)
```

Опис функцій:

1. Тестування методу fit — перевіряє правильність обчислення коефіцієнтів регресії (нахил та перетин з віссю y). Порівнюються коефіцієнти власної реалізації з тими, що розраховані scikit-learn.
2. Тестування обчислення помилки — перевіряє правильність обчислення середньоквадратичної помилки (MSE). Результати власної реалізації порівнюються з результатами scikit-learn.
3. Тестування прогнозування — перевіряє правильність прогнозування значень на основі вхідних даних. Порівнюються результати прогнозування власної реалізації з результатами scikit-learn.

Тестування поліноміальної регресії

Тестування поліноміальної регресії здійснюється шляхом порівняння результатів власної реалізації алгоритму з результатами SkLinearRegression з бібліотеки scikit-learn, що наведено у .

Лістинг *6*.

Лістинг *6 Реалізація тестування поліноміальної регресії*

```
import numpy as np
```

```
import pytest
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression as
SkLinearRegression
from
backend.core.analysis.regressions.polynomial_regression
import PolynomialRegression
from fixtures import simple_dataset,
student_hours_scores_dataset, ice_cream_selling_dataset

@pytest.fixture(params=[simple_dataset,
student_hours_scores_dataset, ice_cream_selling_dataset])
def polynomial_regression_fixture(request):
    x_values, y_values = request.param
    degree = 1
    regression = PolynomialRegression(x_values, y_values,
degree=degree)
    regression.fit()

    poly_features = PolynomialFeatures(degree=degree)
    x_poly = poly_features.fit_transform(x_values.reshape(-
1, 1))

    sk_regression = SkLinearRegression()
    sk_regression.fit(x_poly, y_values)

    return regression, sk_regression, x_values, y_values,
x_poly

def test_fit(polynomial_regression_fixture):
    regression, sk_regression, x_values, y_values, x_poly =
polynomial_regression_fixture
```

```
    np.testing.assert_allclose(regression.coefficients,
sk_regression.coef_, rtol=1e-05, atol=1e-02)

def test_error(polynomial_regression_fixture):
    regression, sk_regression, x_values, y_values, x_poly =
polynomial_regression_fixture
    error = regression.error()
    sk_error = np.mean((sk_regression.predict(x_poly) -
y_values) ** 2)
    np.testing.assert_allclose(error, sk_error, rtol=1e-05,
atol=1e-02)

def test_predict(polynomial_regression_fixture):
    regression, sk_regression, x_values, y_values, x_poly =
polynomial_regression_fixture
    predicted = regression.predict(x_values)
    sk_predicted = sk_regression.predict(x_poly)
    np.testing.assert_allclose(predicted, sk_predicted,
rtol=1e-05, atol=1e-02)

def test_coefficients(polynomial_regression_fixture):
    regression, sk_regression, x_values, y_values, x_poly =
polynomial_regression_fixture
    coefficients = regression.coefficients
    expected_coefficients = sk_regression.coef_
    np.testing.assert_allclose(coefficients,
expected_coefficients, rtol=1e-05, atol=1e-01)
```

Перевірка результатів тестування

Для перевірки результатів тестування треба запустити виконання pytest тестів, виконавши команду «pytest tests/». Після виконання команди отримуємо результати тестування, які представлені у Лістинг *7*.

Лістинг *7 Результати тестування за допомогою pytest*

```
============================= test session starts
==============================
collecting ... collected 18 items

test_linear_regression.py::test_fit[linear_regression_fixtu
re0]
test_linear_regression.py::test_fit[linear_regression_fixtu
re1]
test_linear_regression.py::test_error[linear_regression_fix
ture0]
test_linear_regression.py::test_error[linear_regression_fix
ture1]
test_linear_regression.py::test_predict[linear_regression_f
ixture0]
test_linear_regression.py::test_predict[linear_regression_f
ixture1]
test_polynomial_regression.py::test_fit[polynomial_regressi
on_fixture0]
test_polynomial_regression.py::test_fit[polynomial_regressi
on_fixture1]
test_polynomial_regression.py::test_fit[polynomial_regressi
on_fixture2]
test_polynomial_regression.py::test_error[polynomial_regres
sion_fixture0]
test_polynomial_regression.py::test_error[polynomial_regres
sion_fixture1]
test_polynomial_regression.py::test_error[polynomial_regres
sion_fixture2]
test_polynomial_regression.py::test_predict[polynomial_regr
ession_fixture0]
```

```
test_polynomial_regression.py::test_predict[polynomial_regr
ession_fixture1]
test_polynomial_regression.py::test_predict[polynomial_regr
ession_fixture2]
test_polynomial_regression.py::test_coefficients[polynomial
_regression_fixture0]
test_polynomial_regression.py::test_coefficients[polynomial
_regression_fixture1]
test_polynomial_regression.py::test_coefficients[polynomial
_regression_fixture2]

======================== 18 passed, 1 warning in 2.82s
========================
```

В результаті проведеного тестування за допомогою бібліотеки pytest, було успішно виконано 18 тестів, що охоплюють різні аспекти роботи лінійної та поліноміальної регресії. Усі тести пройшли без помилок, що свідчить про коректне функціонування основних методів прогнозування, обчислення помилок та отримання коефіцієнтів моделей регресії. Однак, було зафіксовано одне попередження, яке не вплинуло на загальний результат тестування. Це свідчить про надійність і стабільність розробленої системи для аналізу даних та прогнозування.

ВИСНОВКИ

1. Розробка універсальної веб-платформи для прикладного статистичного аналізу даних є актуальною та важливою задачею в умовах стрімкого зростання обсягів даних та потреби в їх аналізі. Під час виконання кваліфікаційної роботи були досліджені сучасні інструменти та технології, що дозволяють ефективно обробляти та аналізувати великі масиви даних. Проведено огляд і аналіз існуючих рішень та аналогів, які підкреслюють необхідність створення нових продуктів, що володіють вищою швидкістю обробки даних та гнучкістю у візуалізації.
2. На основі проведених досліджень було обрано веб-фреймворк Vue для клієнтської частини застосунку завдяки його простоті у вивченні, гнучкості та швидкості розробки. Для серверної частини обрано Python разом із FastAPI, який забезпечує високу продуктивність та підтримує асинхронне програмування, що є критичним для обробки великих обсягів даних в реальному часі. Базу даних вибрано PostgreSQL, яка забезпечує надійне та ефективне зберігання структурованих даних, підтримує високий рівень безпеки та продуктивності.
3. Створений веб-застосунок дозволяє користувачам виконувати статистичний аналіз, інтегруючи алгоритми прогнозування для виявлення тенденцій та закономірностей у даних. Інтерфейс платформи розроблено з урахуванням інтуїтивності та зручності, що спрощує взаємодію з комплексними статистичними інструментами. Проведене тестування веб-застосунку підтвердило його ефективність та надійність.
4. Таким чином, створена платформа є потужним, але досить простим інструментом для підтримки прийняття обґрунтованих рішень на основі аналізу даних, що відкриває нові можливості для різних сфер діяльності. Вона особливо корисна для індивідуальних аналітиків та звичайних користувачів, яким потрібен ефективний інструмент для аналізу даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Джон Нейсбітт. Megatrends: Ten New Directions Transforming Our Lives. Warner Books, 1982 p.
2. Вес Мак Кінні. Python for Data Analysis: Edition 3. O’Reilly Media, 2022 p.
3. Гарет М. Джеймс, Даніела Віттен, Тревор Гасті, Роберт Тібширані. An Introduction to Statistical Learning: With Applications in Python. Springer, 2023 p.
4. Єроен Янссенс. Data Science at the Command Line. O’Reilly Media, 2021 p.
5. Джейк Вандер Плас. Python Data Science Handbook. O’Reilly Media, 2016 p.
6. IBM SPSS Statistics. URL: https://www.ibm.com/products/spss-statistics (дата звернення: 24.02.2024).
7. Getting Started: Navigating the Statistica User Interface. URL: https://www.youtube.com/watch?v=W7w2yFz-JwU&list=PLnmoGGHHJldizem1Mzu4AJi7qSR9zdV3o&index=2 (дата звернення: 24.02.2024).
8. A software for everyone from managers to data scientists. URL: https://www.datapine.com/self-service-analytics (дата звернення: 24.02.2024).
9. Why Django? URL: https://www.djangoproject.com/start/overview/ (дата звернення: 09.03.2024).
10. What is Angular? URL: https://angular.io/guide/what-is-angular (дата звернення: 10.03.2024).
11. Vue.js. URL: https://vuejs.org/ (дата звернення: 10.03.2024)
12. Learn React. URL: https://react.dev/learn/describing-the-ui (дата звернення: 10.03.2024).

13. Overview of ASP.NET Core. URL: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0 (дата звернення: 12.03.2024).
14. Welcome to Flask. URL: https://flask.palletsprojects.com/en/3.0.x/ (дата звернення: 23.03.2024).
15. FastAPI. URL: https://fastapi.tiangolo.com/ (дата звернення: 23.03.2024)
16. About SQLite. URL: https://www.sqlite.org/about.html (дата звернення: 04.04.2024).
17. About PostgreSQL. URL: https://www.postgresql.org/about/ (дата звернення: 04.04.2024).
18. What is MongoDB. URL: https://www.mongodb.com/company/what-is-mongodb (дата звернення: 04.04.2024).
19. MongoDB Docs. URL: https://www.mongodb.com/docs (дата звернення: 04.04.2024).
20. Gradient Descent. URL: https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/what-is-gradient-descent (дата звернення: 20.04.2024).
21. Болейко Д.Є., Міхайлуца О.М. доцент, канд. технічних наук — науковий керівник. Переваги feature-sliced design архітектури над component-based на прикладі веб застосунку аналізу даних. Збірник наукових праць студентів, аспірантів, докторантів і молодих вчених «Молода наука-2024» / Запорізький національний університет. — Запоріжжя : ЗНУ, 2024. — Т.5

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Болейко Данило Євгенович, студент 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти ipz20bd-201@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Створення універсальної веб-платформи для прикладного статистичного аналізу даних**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 25.05.2024 Підпис _____ Болейко Данило Євгенович

(студент)

Дата 25.05.2024 Підпис _____ Міхайлуца Олена Миколаївна

(науковий керівник)