СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь бакалавр
спеціальність 126 « Інформаційні системи та технології»
(шифр і назва спеціальності)

на тему «Веб-додаток для автоматизованої обробки та статистичного аналізу даних»

Виконав: студент групи ІСТ-21бд _____ Н.П. Пікун
(підпис) (ініціали і прізвище)

Керівник _____ В.Г. Іванов
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ О.І. Захожай
(підпис) (ініціали і прізвище)

Рецензент _____ В.О. Лифар_____

Київ 2025

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування
Освітній ступінь бакалавр
спеціальність 126 „ Інформаційні системи та технології ”
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Захожай О.І.
“___” _____ 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Пікун Наталія Петрівна
(прізвище, ім’я, по батькові)

1. Тема роботи: Веб-додаток для автоматизованої обробки та статистичного аналізу даних
керівник роботи Віталій Геннадійович Іванов доцент, к.т.н.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від “_27_”_травня_2025 року №67/15.15-С

2. Строк подання студентом роботи 16.06.2025р.
3. Вихідні дані до роботи: матеріали переддипломної практики
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1 Вступ
4.2 Аналітичний огляд сучасних платформ для аналізу даних та потреб користувачів.
4.3 Основна частина, де досліджено методи статистичного аналізу,. алгоритми регресії та розроблено веб-платформу для їх реалізації
4.4 Практична частина, в якій розглянуто технології (FastAPI, Vue.js, PostgreSQL, NumPy, Chart.js) та проведено тестування платформи.
4.5 Висновки
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників) Електронні плакати

Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 01.04.2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Отримання завдання та збір матеріалів	01.04.25	виконано
2	Огляд предметної області	18.04.25	виконано
3	Формулювання вимог до системи та розробка основних алгоритмів	04.05.25	виконано
4	Розробка практичної частини	01.06.25	виконано
5	Оформлення пояснювальної записки	12.06.25	виконано
6	Підготовка та подання роботи до захисту	16.06.25	виконано

Студент _____ Н.П. Пікун
підпис (ініціали і прізвище)

Керівник роботи _____ В.Г. Іванов
підпис (ініціали і прізвище)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту бакалавра: 66 сторінок, 14 рис., 2 табл., 20 джерела посилань, 7 додатків.

У дипломному проекті розроблено універсальну веб-платформу для прикладного статистичного аналізу даних, яка забезпечує зручну обробку, аналіз і візуалізацію даних для користувачів різного рівня підготовки. Проведено аналіз сучасних аналітичних платформ (SPSS, Statistica, Datapine, Datatab.net), виявлено їхні обмеження та сформовано вимоги до нового рішення. Особливу увагу приділено реалізації алгоритмів лінійної та поліноміальної регресії, оптимізованих градієнтним спуском, з використанням бібліотеки NumPy, а також інтеграції з реляційною базою даних PostgreSQL для надійного зберігання даних.

Розроблена платформа включає серверну частину на основі FastAPI з підтримкою асинхронної обробки та шифрування даних, а також клієнтську частину, створену за допомогою Vue.js, Vuetify і Chart.js для інтерактивної візуалізації. Здійснено тестування функціональності за допомогою pytest, підтвердивши відповідність результатів бібліотеці scikit-learn. Платформа підтримує формати CSV, XLSX, PARQUET, забезпечує фільтрацію, сортування та автоматизовану звітність, що робить її ефективним інструментом для бізнес-аналітики, освіти та наукових досліджень.

У проекті враховано нормативні вимоги з охорони праці, забезпечено відповідність умов праці розробників сучасним стандартам. Розглянуто аспекти ергономіки робочого місця, освітлення та мікроклімату, що сприяють створенню комфортного середовища для роботи над платформою.

Дипломний проект демонструє впровадження інноваційних веб-технологій для статистичного аналізу даних, забезпечуючи доступність, надійність і високу продуктивність для широкого кола користувачів.

ВЕБ-ПЛАТФОРМА, СТАТИСТИЧНИЙ АНАЛІЗ, РЕГРЕСІЙНИЙ АНАЛІЗ, ВІЗУАЛІЗАЦІЯ ДАНИХ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ЗМІСТ

ВСТУП

У сучасному світі дані стали основою для прийняття рішень у бізнесі, науці, освіті та інших сферах. Стрімке зростання обсягів інформації, спричинене цифровізацією, вимагає ефективних інструментів для її обробки та аналізу. Складність традиційних аналітичних платформ, таких як SPSS чи Statistica, часто створює бар'єри для користувачів без глибоких технічних знань, тоді як сучасні хмарні рішення, як Datapine чи Datatab.net, мають обмеження у функціональності чи доступності. Це підкреслює необхідність розробки універсальних веб-застосунків, які поєднують простоту використання з потужними аналітичними можливостями.

Актуальність дослідження полягає у створенні доступного веб-застосунку для статистичного аналізу даних, який відповідає потребам широкого кола користувачів — від аналітиків і менеджерів до студентів і дослідників. Такий інструмент дозволяє автоматизувати обробку даних, проводити регресійний аналіз і надавати наочні результати, що сприяє швидкому прийняттю обґрунтованих рішень у динамічних умовах.

Об'єктом дослідження є процеси розробки веб-застосунків для аналізу даних, що включають створення серверної та клієнтської частин, інтеграцію алгоритмів прогнозування та забезпечення зручного інтерфейсу.

Предметом дослідження є методи, технології та архітектурні рішення для створення веб-платформи, яка підтримує обробку структурованих даних, регресійний аналіз (лінійний і поліноміальний) і їхню візуалізацію.

Метою дослідження є розробка веб-застосунку для прикладного статистичного аналізу даних, який забезпечує інтуїтивну взаємодію, високу продуктивність і надійність, дозволяючи користувачам ефективно обробляти та аналізувати дані.

.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Виклики сучасного аналізу даних

Сьогодні аналіз даних є основою для прийняття рішень у бізнесі, науці, державному управлінні та повсякденному житті. Зростання обсягів інформації, спричинене цифровізацією, відкриває можливості для прогнозування трендів, оптимізації процесів і персоналізації рішень. Проте обробка великих масивів даних супроводжується низкою викликів, які потребують нових технологій і підходів.

Основні труднощі включають:

1. Обсяг і швидкість даних: Щосекунди генеруються величезні обсяги даних із соціальних мереж, сенсорів і бізнес-систем. За словами Джона Нейсбітта у книзі «Megatrends» [1], ми «потопаємо в інформації, але відчуваємо нестачу знань», що підкреслює потребу в швидкій обробці та інтерпретації..
2. Різноманітність форматів: Дані бувають структурованими (таблиці, бази даних), неструктурованими (текст, відео, зображення) та напівструктурованими (JSON, XML), що ускладнює їх аналіз.
3. Безпека: Захист даних від несанкціонованого доступу є критично важливим.
4. Доступність інструментів: Складність сучасних аналітичних платформ часто створює бар’єри для користувачів без технічної підготовки.

1.2 Джерела та формати 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.

Сучасні дані надходять із різних джерел, кожне з яких має унікальні

особливості. Це створює основу для подальшого аналізу інструментами, описаними нижче Таблиця 1.1.

Таблиця 1.1 — Сучасні дані

Джерело	Опис
Електронні пристрої та IoT	Сенсори фіксують параметри в реальному часі (температура, рух).
|Соціальні медіа	Пости та відео відображають поведінку користувачів.
Публічні бази даних	Наукові звіти та статистика доступні для аналізу
Внутрішні системи	Дані про продажі та клієнтів для бізнес-аналітики

Формати даних поділяються на три категорії:

1. Структуровані: Таблиці в реляційних базах даних, зручні для аналізу.
2. Неструктуровані: Тексти, медіафайли, що потребують спеціалізованих методів обробки.
3. Напівструктуровані: Дані у форматах JSON або XML із метаданими для організації.

Різноманітність джерел і форматів ускладнює створення універсальних інструментів аналізу, що вимагає гнучких рішень для збору та обробки інформації.

1.3 Інструменти аналізу даних

Для обробки даних використовуються спеціалізовані інструменти, які забезпечують аналіз, моделювання та візуалізацію.

Мови програмування: Python (бібліотеки Pandas, NumPy, Matplotlib, Scikit-learn [2-5]) і R (ggplot2, CRAN) популярні завдяки гнучкості.

. Програмне забезпечення, таке як SPSS і Statistica, пропонує графічні інтерфейси для дослідників, тоді як Datapine і Datatab.net забезпечують онлайн-аналітику для бізнесу та освіти.

1.4 Методи прогнозування в аналізі даних

Прогнозування є ключовим аспектом аналізу даних, дозволяючи передбачати майбутні тенденції на основі історичних даних. Одними з найпоширеніших методів є лінійна та поліноміальна регресія, які широко застосовуються в бізнесі, науці та техніці завдяки простоті й ефективності [2, 5].

Оптимізація параметрів виконується за допомогою градієнтного спуску, який мінімізує середньоквадратичну помилку (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overrightarrow{y_i})^2$$

Цей метод ефективний для лінійних залежностей, наприклад, прогнозування продажів залежно від витрат на рекламу.

Поліноміальна регресія розширює лінійну модель, дозволяючи моделювати нелінійні залежності через поліном виду

$$y = \theta_0 + \theta_1 x + \theta_1 x^2 + \cdots + \theta_n x^n$$

Вона корисна для складніших даних, таких як залежність споживання енергії від температури.

Обидва методи реалізуються за допомогою бібліотек, таких як Scikit-learn, і потребують автоматизації для обробки великих масивів даних. Обмеження включають чутливість до аномалій і потребу в якісних даних. Автоматизовані платформи, які інтегрують ці методи, значно спрощують аналіз і підвищують точність прогнозів.

1.5 Роль штучного інтелекту в аналізі даних

Штучний інтелект (ШІ) і машинне навчання (ML) трансформують аналіз даних, забезпечуючи автоматизацію обробки, прогнозування та інтерпретації великих масивів інформації [2, 8]. ШІ дозволяє створювати системи, які самостійно знаходять закономірності, оптимізують процеси та надають користувачам цінні інсайти.

Основні напрями застосування ШІ в аналізі даних включають:

1. Класифікація та кластеризація: Автоматизоване групування даних, наприклад, сегментація клієнтів у маркетингу.
2. Прогнозування: Моделі ML, такі як регресія чи нейронні мережі, підвищують точність прогнозів порівняно з традиційними методами.
3. Інтерпретація даних: Генерація текстових звітів або рекомендацій на основі аналізу.
4. Візуалізація: ШІ спрощує створення інтерактивних графіків, адаптованих до потреб користувача.

Платформи, такі як Datatab.net, уже використовують ШІ для автоматизації статистичного аналізу, що знижує бар’єри для користувачів без технічної підготовки [8]. Наприклад, алгоритми ШІ можуть автоматично підбирати оптимальні моделі регресії або виявляти аномалії в даних.

Проте впровадження ШІ супроводжується викликами:

1. Обчислювальні ресурси: Моделі ML потребують значної потужності для обробки великих даних.
2. Якість даних: ШІ залежить від точності та повноти вхідної інформації.
3. Етичні питання: Використання персональних даних вимагає дотримання конфіденційності та прозорості.

Інтеграція ШІ у веб-додатки для аналізу даних відкриває перспективи для створення універсальних платформ, які поєднують простоту використання з потужними аналітичними можливостями. Це підкреслює актуальність розробки нових рішень, які враховують сучасні технологічні тренди.

1.6 Порівняння аналітичних платформ

Розглянемо ключові платформи для аналізу даних, які є аналогами розробленого веб-додатку: SPSS, Statistica, Datapine і Datatab.net. Кожна платформа має унікальні переваги та обмеження, що впливають на вибір інструменту.

SPSS (IBM) пропонує широкий набір методів (регресія, ANOVA) з модульністю, але має високу вартість і обмежену гнучкість у скриптах [6]. Statistica (TIBCO) забезпечує високу продуктивність, але складна для новачків [7]. Datapine орієнтований на бізнес-аналітику з інтуїтивним інтерфейсом, але дорогой для малого бізнесу [8]. Datatab.net дає безкоштовний доступ до базового аналізу через ШІ, але з обмеженнями для складних задач.

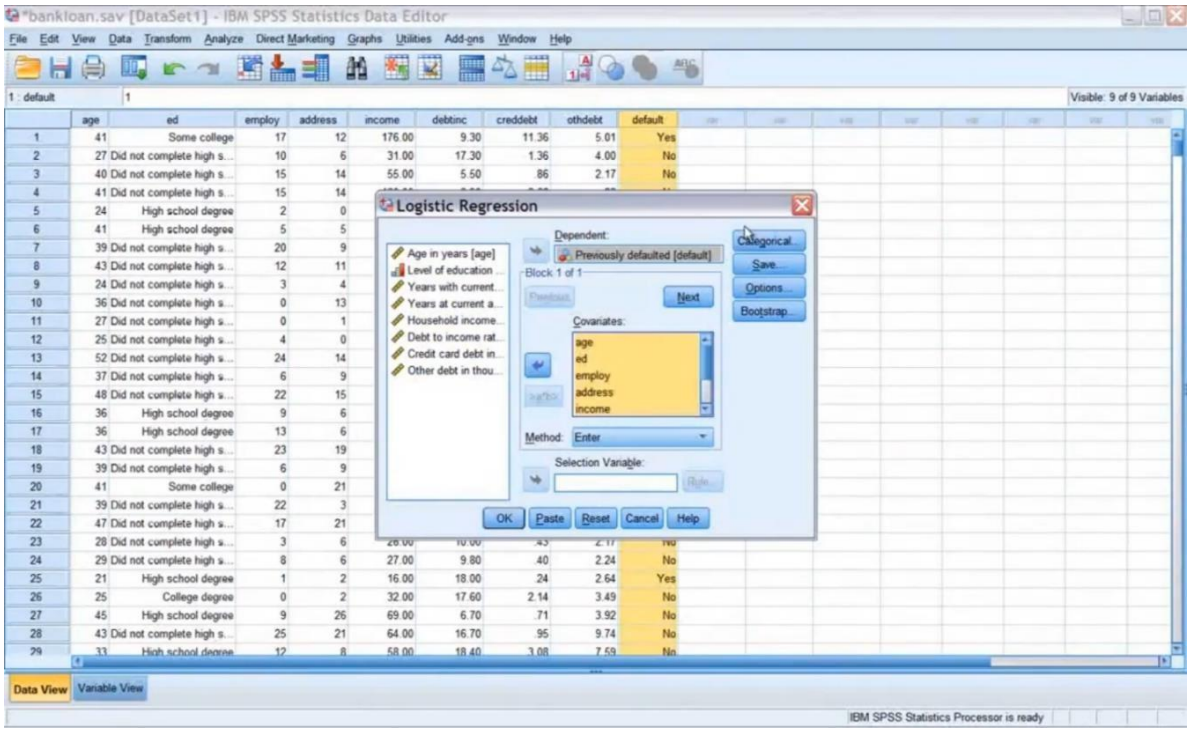


Рисунок 1.1 — Інтерфейс SPSS, вікно логістичної регресії [6]

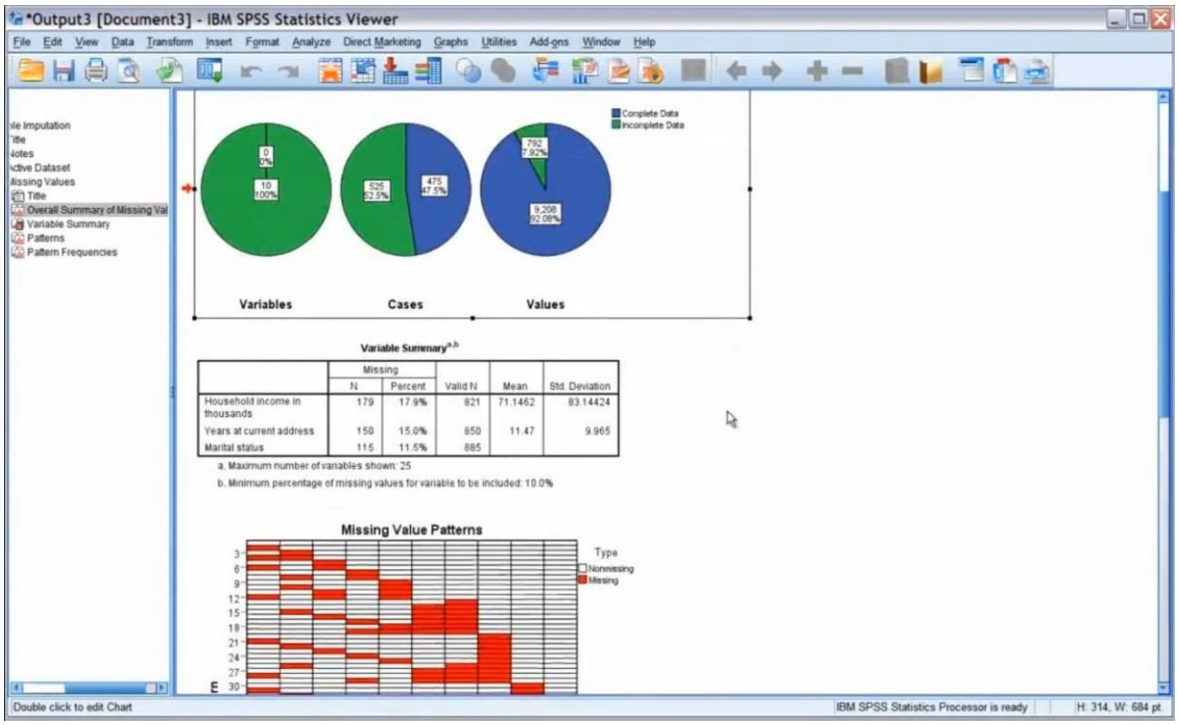


Рисунок 1.2 — Інтерфейс SPSS, формування звіту [6]

Переваги:

1. Великий набір інструментів.
2. Модульність для спеціалізованих аналізів.
3. Сумісність із різними форматами даних.
4. Визнання в академічному середовищі.

Недоліки:

1. Висока вартість.
2. Обмежена гнучкість у скриптах порівняно з Python чи R

Statistica (TIBCO) підтримує швидку обробку великих даних і пропонує як графічний інтерфейс, так і програмування [7].

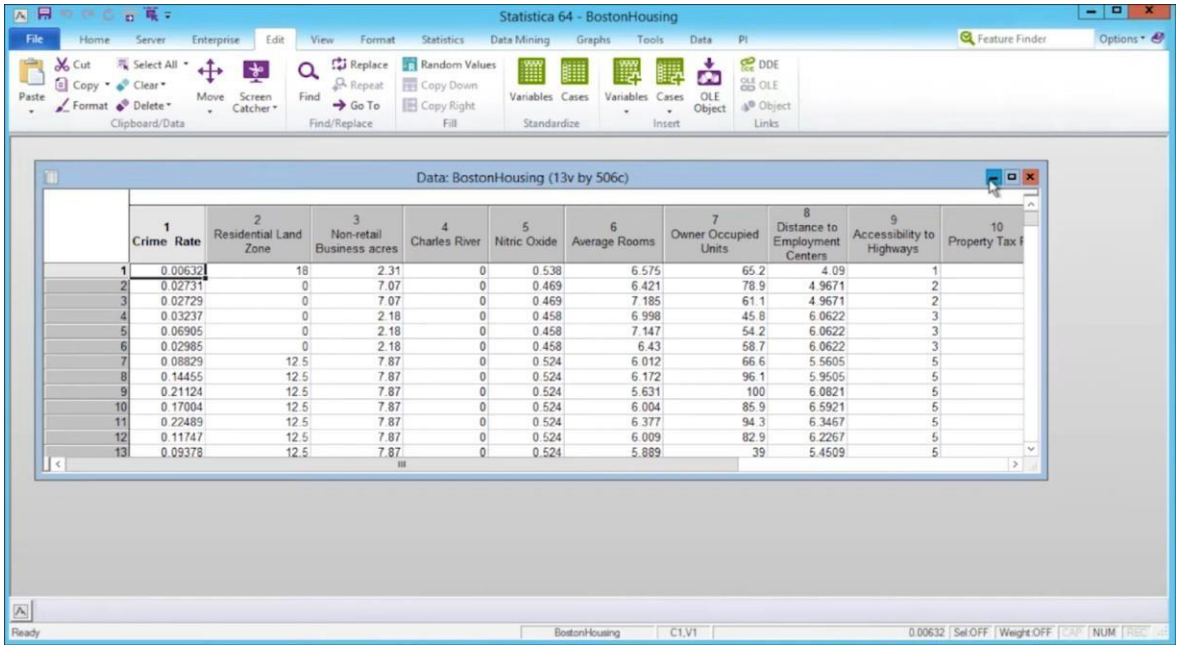


Рисунок 1.3 — Інтерфейс Statistica, візуалізація даних у вигляді таблиці [7]

Переваги:

1. Висока продуктивність.
2. Гнучкість у налаштуваннях.

Недоліки:

1. Складність для новачків.
2. Вартість ліцензії.

Datapine Analyzer орієнтований на бізнес-аналітику, дозволяючи створювати інтерактивні дашборди та звіти [8]

Рисунок 1.4 — Інтерфейс Datapine Analyzer [8]

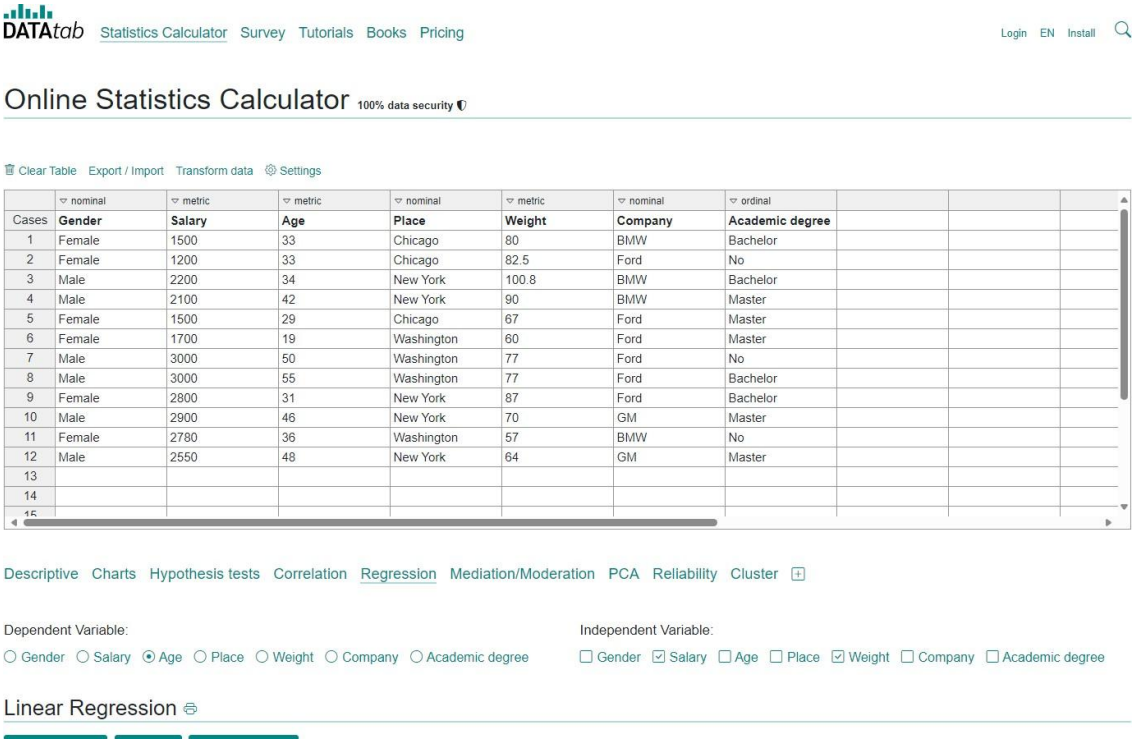
Переваги:

1. Інтуїтивний інтерфейс.
2. Гнучка візуалізація.
3. Інтеграція з різними джерелами.

Недоліки:

1. Висока ціна для малого бізнесу.

Datatab.net — онлайн-платформа для простого аналізу та візуалізації, доступна широкій аудиторії.



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							

Рисунок 1.5 — Інтерфейс Datatab.net

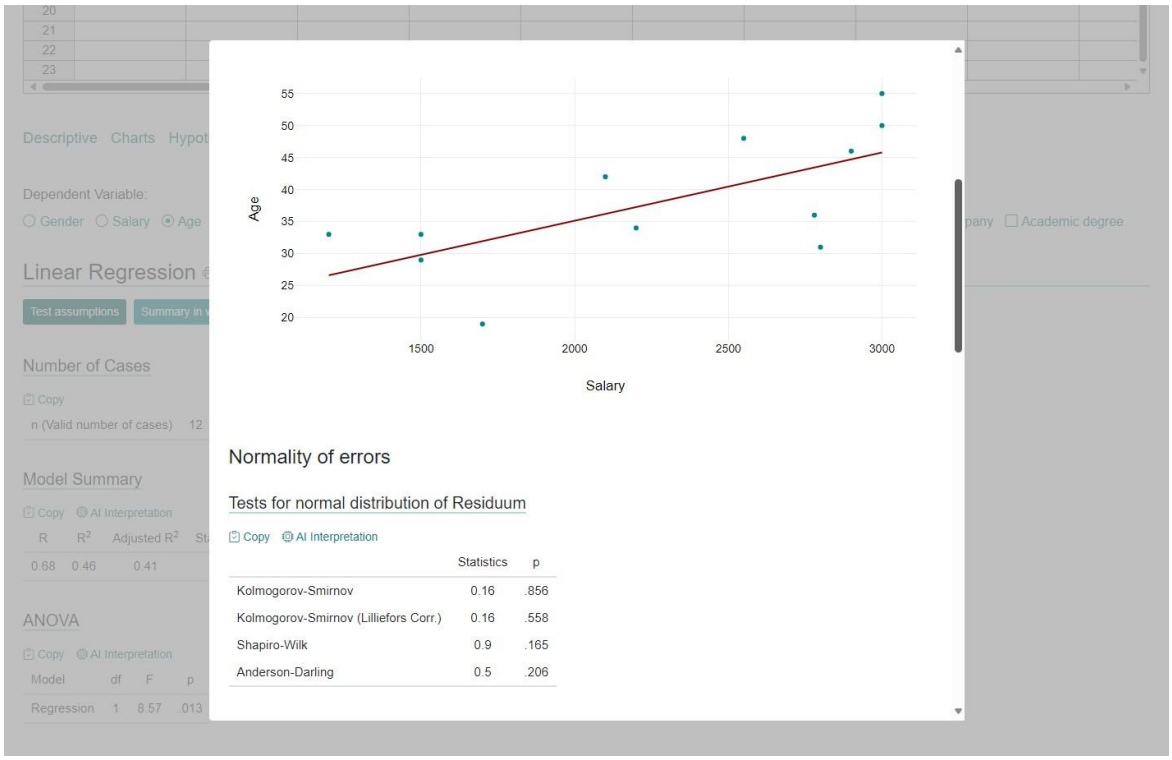


Рисунок 1.6 — Регресійний аналіз в Datatab.net

Переваги:

1. Безкоштовний доступ до базових функцій.
2. Простота використання.
3. Аналітика через ШІ.

4. Безпека даних (обробка на клієнтській стороні).

Недоліки:

1. Обмежені можливості для складного аналізу.

Аналіз аналогів показує, що сучасні платформи поєднують потужні алгоритми з інтуїтивними інтерфейсами, але часто мають високу вартість або обмежену гнучкість. Це підкреслює потребу в нових рішеннях, які забезпечують швидкість, доступність і підтримку новітніх технологій, таких як машинне навчання.

.

2 ІНФОРМАЦІЙНА МОДЕЛЬ

На основі проведеного аналізу існуючих платформ для аналізу даних (таких як SPSS, Statistica та Datatab.net) було розроблено інформаційну модель веб-платформи, яка враховує їхні сильні сторони та усуває обмеження. Модель забезпечує гнучку структуру для зберігання та обробки даних, інтеграцію алгоритмів прогнозування та зручний доступ через інтуїтивний інтерфейс, що детально описано в наступних підрозділах.

Інформаційна модель є основою для організації, зберігання та обробки даних у веб-платформі для прикладного статистичного аналізу. Вона визначає структуру даних, алгоритми їх обробки та взаємодію між компонентами системи (фронтенд, бекенд, база даних). У контексті платформи інформаційна модель забезпечує ефективне зчитування датасетів, їх аналіз за допомогою алгоритмів прогнозування (лінійна та поліноміальна регресія), а також візуалізацію результатів через інтерактивні графіки. Цей розділ описує сутності даних, їх зв’язки, алгоритми обробки та принципи взаємодії компонентів, що дозволяє користувачам отримувати точні прогнози та інсайти

2.1 Структура даних

2.1.1 Сутності та зв’язки

Інформаційна модель веб-платформи базується на реляційній структурі даних, реалізованій у системі управління базами даних (СУБД) PostgreSQL, яка забезпечує цілісність, масштабованість та ефективну обробку структурованих даних [16]. Основними сутностями моделі є:

Користувач (User): Зберігає інформацію про зареєстрованих користувачів. Атрибути: id (первинний ключ, ціле число), name (рядок, не null), email (рядок, унікальний, не null), password_hash (рядок, не null), created_at (часова мітка, за замовчуванням поточний час).

Датасет (Dataset): Містить метадані завантажених наборів даних.

Атрибути: id (первинний ключ), user_id (зовнішній ключ, посилання на users.id), name (рядок, не null), description (текст, опціонально), created_at, updated_at (часові мітки).

Файл датасету (DatasetFile): Зберігає шлях до зашифрованого CSV-файлу. Атрибути: id (первинний ключ), dataset_id (зовнішній ключ, унікальний, посилання на datasets.id), file_path (рядок, не null).

Результати аналізу (AnalysisResult): Зберігає результати регресійного аналізу. Атрибути: id (первинний ключ), dataset_id (зовнішній ключ, посилання на datasets.id), regression_type (рядок, не null), parameters (JSONB, не null), mse (дійсне число, не null), created_at (часова мітка).

Зв’язки між сутностями забезпечують цілісність даних і підтримують нормалізацію до третьої нормальної форми (3NF), що усуває надлишковість даних і підвищує ефективність їх обробки [3]:

Користувач – Датасет: Один користувач може мати багато датасетів (зв’язок 1:N). Зовнішній ключ user_id у таблиці datasets посилається на id таблиці users. Реалізовано каскадне видалення (ON DELETE CASCADE), що забезпечує автоматичне видалення всіх датасетів користувача при видаленні його облікового запису.

Датасет – Файл датасету: Один датасет відповідає одному файлу (зв’язок 1:1). Зовнішній ключ dataset_id у таблиці dataset_files є унікальним і посилається на id таблиці datasets. Каскадне видалення забезпечує видалення файлу при видаленні відповідного датасету.

Датасет – Результати аналізу: Один датасет може мати багато результатів аналізу (зв’язок 1:N). Зовнішній ключ dataset_id у таблиці analysis_results посилається на id таблиці datasets. Каскадне видалення автоматично видаляє результати аналізу при видаленні датасету.

Вибір PostgreSQL як СУБД обґрунтований її підтримкою складних типів даних, зокрема JSONB для зберігання параметрів регресійних моделей, високою продуктивністю та можливістю масштабування [16]. Реляційна структура даних оптимізована для швидкого виконання запитів і обробки великих обсягів інформації, що є критично важливим для реалізації функціоналу статистичного аналізу та прогнозування.

Для наочності структури даних розроблено UML-діаграму класів, яка відображає сутності (User, Dataset, DatasetFile, AnalysisResult) та їхні зв’язки (див. рисунок 2.1). Діаграма ілюструє атрибути кожної сутності, включаючи первинні та зовнішні ключі, а також асоціації між класами з урахуванням каскадного видалення (ON DELETE CASCADE). Використання UML забезпечує чітке представлення інформаційної моделі, що відповідає стандартам моделювання програмного забезпечення [3]

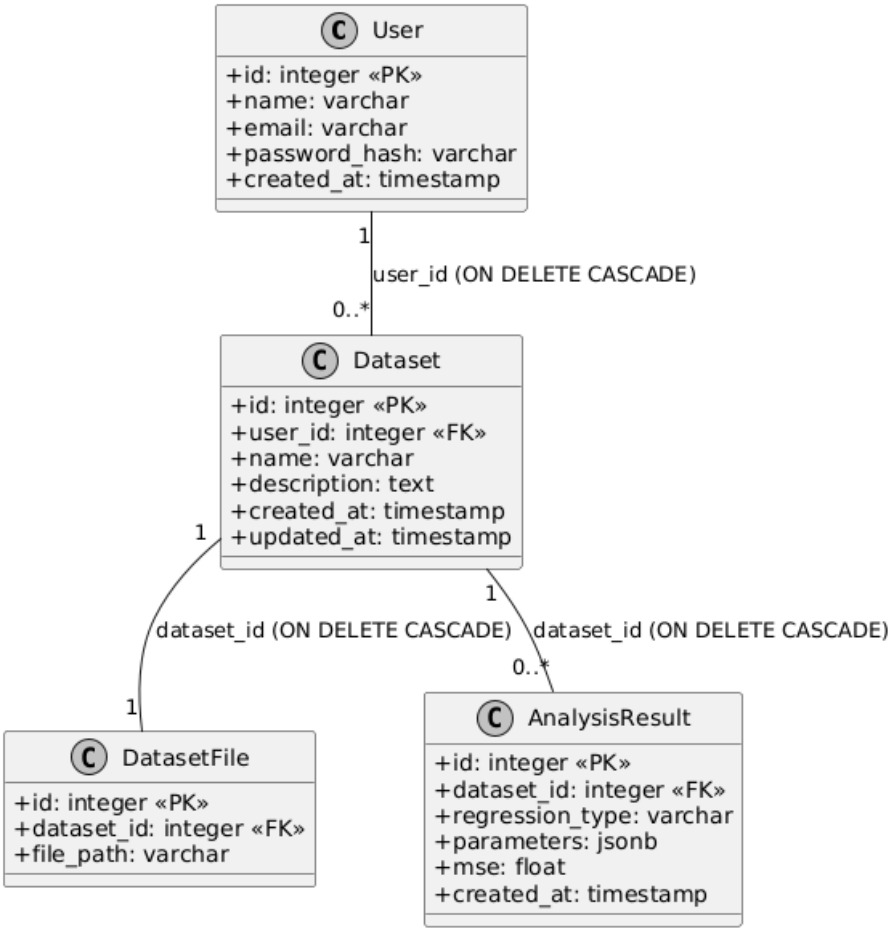


Рисунок 2.1 — UML-діаграма

2.1.2 Формати даних

Платформа підтримує структуровані дані у форматі, які містять числові та текстові колонки. Наприклад, датасет може мати структуру:

Таблиця 2.1 — Структура датасет

Параметр	Значення
Hours	2
Scores	50
Hours	4
Scores	70
Hours	6
Scores	90

Ці дані нормалізуються перед аналізом (наприклад, масштабування до [0, 1]) для підвищення точності алгоритмів. Результати аналізу (прогнози, параметри регресії) зберігаються у JSON для гнучкості

2.2 Алгоритми обробки даних даних

Алгоритми аналізу даних є ядром платформи, дозволяючи прогнозувати тенденції та виявляти залежності. Основними алгоритмами є лінійна та поліноміальна регресія, оптимізовані через градієнтний спуск

Лінійна регресія

Лінійна регресія моделює лінійну залежність між незалежною змінною X (наприклад, години навчання) та залежною Y (бали) [3]. Цей зв'язок можна виразити у формі рівняння:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon, \qquad (2.1)$$

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:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{h}\left(Y_i - \hat{Y}i_i\right)^2 \qquad (2.2)$$

де Y_i — спостережувана змінна;

$\hat{Y}_i$ — передбачене значення;

n — довжина вибірки;

Лістинг 2.1 — Лінійна регресія:

```
def LinearRegression(X, Y, alpha, iterations, threshold):
    beta_0 = 0
    beta_1 = 0
    normalize(X, Y)
    for i in range(iterations):
        prediction = beta_0 + beta_1 * X
        error = Y - prediction
        gradient_beta_0 = -2 * sum(error) / len(X)
        gradient_beta_1 = -2 * sum(error * X) / len(X)
        beta_0 = beta_0 - alpha * gradient_beta_0
        beta_1 = beta_1 - alpha * gradient_beta_1
        if abs(beta_0_change) < threshold and abs(beta_1_change) < threshold:
            break
    return beta_0, beta_1, MSE(prediction, Y)
```

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:

$$Y = \beta_0 + \beta X + \beta X^2 + \cdots + \beta X^n + \varepsilon \qquad (2.3)$$

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.

Лістинг 2.2 — Поліноміальна регресія:

```
def PolynomialRegression(X, Y, degree, alpha, iterations, threshold):
  coefficients = [0] * (degree + 1)
  X_poly = [X**i for i in range(degree + 1)]
  normalize(X_poly, Y)
  for i in range(iterations):
    prediction = sum(coefficients[j] * X_poly[j] for j in range(degree + 1))
    error = Y - prediction
    for j in range(degree + 1):
      gradient = -2 * sum(error * X_poly[j]) / len(X)
      coefficients[j] = coefficients[j] - alpha * gradient
    if abs(coefficients_change) < threshold:
      break
        return coefficients, MSE(prediction, Y)
```

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]:

$$x_{i+1} = x_i - \alpha \nabla f(x_0), \qquad (2.4)$$

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.

Регуляризація

Для запобігання перенавчанню використовується L2-регуляризація (Ridge):

$$J(\beta) = MSE + \lambda \sum_{i=1}^{n} \beta_i^2 \qquad (2.5)$$

де λ — коефіцієнт регуляризації.

Лістинг 2.3 — Ridge-регресія:

```
def RidgeRegression(X, Y, alpha, lambda_, iterations, threshold):
    beta_0 = 0
    beta_1 = 0
    normalize(X, Y)
    for i in range(iterations):
        prediction = beta_0 + beta_1 * X
        error = Y - prediction
        gradient_beta_0 = -2 * sum(error) / len(X) + 2 * lambda_ * beta_0
        gradient_beta_1 = -2 * sum(error * X) / len(X) + 2 * lambda_ * beta_1
        beta_0 = beta_0 - alpha * gradient_beta_0
        beta_1 = beta_1 - alpha * gradient_beta_1
        if abs(beta_0_change) < threshold and abs(beta_1_change) < threshold:
            break
    return beta_0, beta_1, MSE(prediction, Y)
```

2.3 Взаємодія компонентів

Модель забезпечує зв’язок між Vue.js (фронтенд), FastAPI (бекенд) і PostgreSQL. Етапи:

1. *Завантаження датасету*: CSV-файл шифрується і зберігається у storage, метадані записуються в datasets.2. *Аналіз даних*: Користувач обирає регресію, FastAPI викликає алгоритми (NumPy), результати зберігаються в analysis_results.

3. *Візуалізація*: Фронтенд отримує дані через API, Chart.js будує

графіки.UML-діаграма (Рисунок 2.1) відображає зв’язки, забезпечуючи інтуїтивну взаємодію.

Лістинг 2.4 — Аналіз датасету:

```
def AnalyzeDataset(dataset_id, regression_type, X_col, Y_col):
   dataset = fetch_from_PostgreSQL(dataset_id)
   data = read_CSV_from_storage(dataset)
   X = data[X_col]
   Y = data[Y_col]
   if regression_type == "linear":
      result = LinearRegression(X, Y)
   elif regression_type == "polynomial":
      result = PolynomialRegression(X, Y, degree=2)
   save_results(parameters, mse, analysis_results_table)
   return JSON_result
```

Інформаційна модель платформи забезпечує ефективне зберігання, обробку та аналіз структурованих даних. Реляційна структура на основі PostgreSQL підтримує цілісність і масштабованість даних. Алгоритми лінійної та поліноміальної регресії, оптимізовані градієнтним спуском, дозволяють точно прогнозувати тенденції. Додавання регуляризації підвищує стійкість моделей до перенавчання. Псевдокод і формули ілюструють логіку обробки даних, а UML-діаграма (див. Рисунок 2.1) чітко відображає зв’язки між сутностями. Модель забезпечує інтуїтивну взаємодію користувача з даними через фронтенд, роблячи платформу потужним інструментом для статистичного аналізу.

3 СТВОРЕННЯ ІНТЕРАКТИВНОГО ІНСТРУМЕНТУ ДЛЯ АНАЛІЗУ ДАНИХ

3.1 Аналіз контексту та потреб користувачів

3.1 Аналіз контексту та потреб користувачів

Сучасний світ переживає справжній вибух обсягів даних, що накопичуються внаслідок цифровізації всіх сфер життя. Ці дані, отримані з різноманітних джерел — від соціальних мереж і сенсорів Інтернету речей до внутрішніх корпоративних систем, — потребують ефективної обробки та аналізу для прийняття обґрунтованих і своєчасних рішень. Аналіз даних перетворився на незамінний інструмент для організацій різного масштабу: від транснаціональних корпорацій, що оптимізують глобальні ланцюги поставок, до невеликих стартапів, які шукають конкурентні переваги, а також для окремих фахівців, таких як аналітики, вчителі чи дослідники, які прагнуть підвищити ефективність своєї роботи. Водночас значна частина цих користувачів стикається з бар’єром: брак технічних навичок чи часу для освоєння складних спеціалізованих програм, таких як SPSS чи R. У зв’язку з цим зростає попит на прості, інтуїтивно зрозумілі інструменти, які дозволяють швидко завантажувати дані, проводити базовий або розширений аналіз і отримувати наочні результати без необхідності занурюватися в програмування чи складні інтерфейси.

Розробка веб-застосунків для аналізу даних стала логічною відповіддю на ці виклики, пропонуючи гнучкі та доступні рішення, які працюють на будь-якому пристрої з доступом до інтернету — від настільних комп’ютерів до смартфонів. Такі платформи усувають потребу в локальній установці програмного забезпечення, надаючи користувачам зручний доступ до потужних аналітичних інструментів через браузер. Вони дозволяють зосередитися на інтерпретації результатів, а не на технічних деталях, таких як налаштування серверів чи обробка помилок. Наприклад, відомі платформи, як-от SPSS, Statistica, Datapine та Datatab.net, демонструють різні

підходи до вирішення цих потреб: SPSS і Statistica пропонують розширені функції для професійних аналітиків із акцентом на локальну роботу [10], тоді як Datapine і Datatab.net роблять ставку на хмарні рішення, орієнтовані на бізнес-аналітику та освіту [11, 12]. Однак кожна з цих платформ має свої обмеження: висока вартість ліцензій, складність інтерфейсу чи недостатня гнучкість для специфічних завдань.

Розроблений веб-застосунок вирізняється унікальним поєднанням простоти використання та широких функціональних можливостей, що робить його конкурентоспроможним серед аналогів. На відміну від SPSS і Statistica, які вимагають встановлення на локальні пристрої та орієнтовані переважно на досвідчених користувачів, цей інструмент доступний через веб-інтерфейс, що значно знижує вхідний бар’єр. Порівняно з Datapine, який зосереджений на бізнес-дашбордах, чи Datatab.net, що обмежується базовими функціями, новий застосунок підтримує широкий спектр форматів даних — CSV, XLSX, PARQUET — і пропонує інтерактивні інструменти для аналізу, такі як регресійний аналіз (лінійний і поліноміальний), динамічні графіки та інтерактивні дашборди. Це дозволяє користувачам із різним рівнем підготовки — від студентів до менеджерів середнього рівня — швидко адаптуватися до платформи та ефективно працювати з даними, не витрачаючи час на освоєння складних систем.

Ключовими потребами сучасних користувачів є не лише зручність, але й швидкість роботи з даними. Вони очікують можливості легко завантажувати файли з різних джерел — від локальних комп’ютерів до хмарних сховищ, таких як Google Drive чи Dropbox, — а також проводити їх обробку з мінімальними зусиллями. Крім того, важливим є представлення результатів у зрозумілій і візуально привабливій формі, наприклад, через таблиці, гістограми чи інтерактивні графіки, які дозволяють миттєво оцінити тенденції. Розроблений застосунок відповідає цим вимогам завдяки інтеграції з реляційними базами даних, що забезпечують надійне зберігання даних, а також підтримці таких функцій, як фільтрація, сортування та пошук у таблицях. Автоматизація створення звітів дозволяє користувачам генерувати документи з результатами аналізу за лічені хвилини, тоді як інтерактивні

дашборди надають можливість моніторингу ключових показників у реальному часі — наприклад, продажів чи споживання ресурсів. Такий підхід робить веб-застосунок не лише зручним, але й стратегічно важливим інструментом для оперативного прийняття рішень у різних сферах: від маркетингу до наукових досліджень. Це особливо цінно для користувачів, які працюють у динамічних умовах, де час і доступність даних є критичними факторами успіху..

3.2 Визначення технічних вимог і специфікацій

Розробка веб-застосунку для аналізу даних потребує чіткого визначення вимог, які забезпечують його функціональність, зручність та відповідність потребам користувачів. Вимоги поділяються на бізнес-вимоги, що відображають цілі застосунку, та функціональні вимоги, які деталізують необхідні можливості. Окремо розглядаються основні сценарії використання, щоб забезпечити відповідність застосунку реальним потребам користувачів.

Бізнес-вимоги

Бізнес-вимоги визначають загальні цілі веб-застосунку, спрямовані на забезпечення ефективного аналізу даних для користувачів різного рівня підготовки. Застосунок має бути доступним через веб-інтерфейс, що дозволяє працювати з ним із будь-якого пристрою з доступом до інтернету. Важливим є забезпечення інтуїтивного інтерфейсу, який мінімізує час на освоєння інструменту. Крім того, застосунок повинен підтримувати обробку даних у реальному часі та надавати користувачам можливість швидкого отримання результатів аналізу. Технічні обмеження включають необхідність оптимізації продуктивності для обробки великих наборів даних (до 100 000 записів) без значних затримок, а також забезпечення безпеки даних шляхом шифрування під час зберігання та передачі.

Функціональні вимоги

Функціональні вимоги деталізують можливості, які має забезпечувати веб-застосунок, щоб відповідати очікуванням користувачів:

Підтримка завантаження наборів даних у форматах CSV, XLSX,

PARQUET.

Зберігання завантажених даних у базі даних із забезпеченням швидкого доступу.

Інтеграція з базою даних для надійного управління наборами даних.

Реалізація інтерфейсу для перегляду даних у табличному вигляді з можливостями фільтрування, сортування та пошуку.

Відображення даних у вигляді інтерактивних графіків для наочного аналізу.

Підтримка лінійної та поліноміальної регресії з можливістю вибору змінних.

Надання результатів регресійного аналізу у вигляді графіків та основних статистичних показників.

Ці вимоги сформовані на основі аналізу потреб користувачів та вивчення функціоналу аналогічних систем, що забезпечує їх актуальність і практичність.

Сценарії використання

Основні сценарії використання застосунку включають завантаження даних користувачем, їх обробку та аналіз. Наприклад, користувач може завантажити CSV-файл із фінансовими даними, переглянути їх у таблиці, відфільтрувати за певними параметрами, побудувати графік для виявлення тенденцій і провести регресійний аналіз для прогнозування. Діаграма прецедентів (див. Рисунок 3.1) ілюструє ключові взаємодії користувача із системою, такі як авторизація, керування наборами даних, налаштування графіків і виконання аналізу. На діаграмі показано акторів (користувачів) і їхні дії, що допомагає зрозуміти послідовність операцій і взаємозв’язок між функціями.

Технічні обмеження

Для забезпечення стабільної роботи застосунок має враховувати технічні обмеження, такі як обмеження обсягу оперативної пам’яті на сервері (рекомендується щонайменше 8 ГБ для обробки великих наборів даних), підтримка браузерів (сумісність із Chrome, Firefox, Safari останніх версій) і вимоги до швидкості обробки запитів (відповідь API не більше 2 секунд для

стандартних операцій). Безпека даних забезпечується шифруванням за стандартом AES-256, а для масштабованості передбачено можливість розгортання на хмарних платформах.

Такий підхід до формування вимог забезпечує створення ефективного та зручного інструменту для аналізу даних, який відповідає сучасним потребам користувачів.

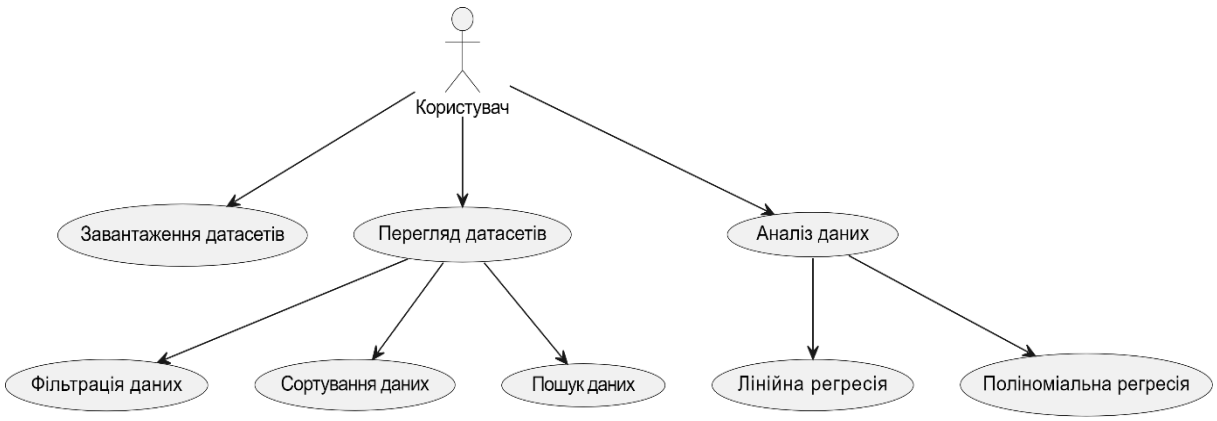


Рисунок 3.1 — Діаграма прецедентів

3.3 Проєктування інфраструктури системи

Проєктування інфраструктури веб-застосунку для аналізу даних передбачає чітке розділення системи на серверну (бекенд) і клієнтську (фронтенд) частини. Такий підхід забезпечує гнучкість у розробці, спрощує підтримку та підвищує зручність для користувачів. Бекенд відповідає за обробку даних і логіку системи, тоді як фронтенд забезпечує інтерактивний інтерфейс для взаємодії з користувачем. Нижче розглянуто особливості архітектури кожної частини з відповідними схемами.

Архітектура серверної частини

Серверна частина застосунку реалізована з використанням багатошарової архітектури, що сприяє модульності, легкості у підтримці та масштабуванні. Кожен шар відповідає за окремі функції, що дозволяє розробникам працювати над різними компонентами незалежно. Схема серверної частини представлена на рисунку 3.2

.

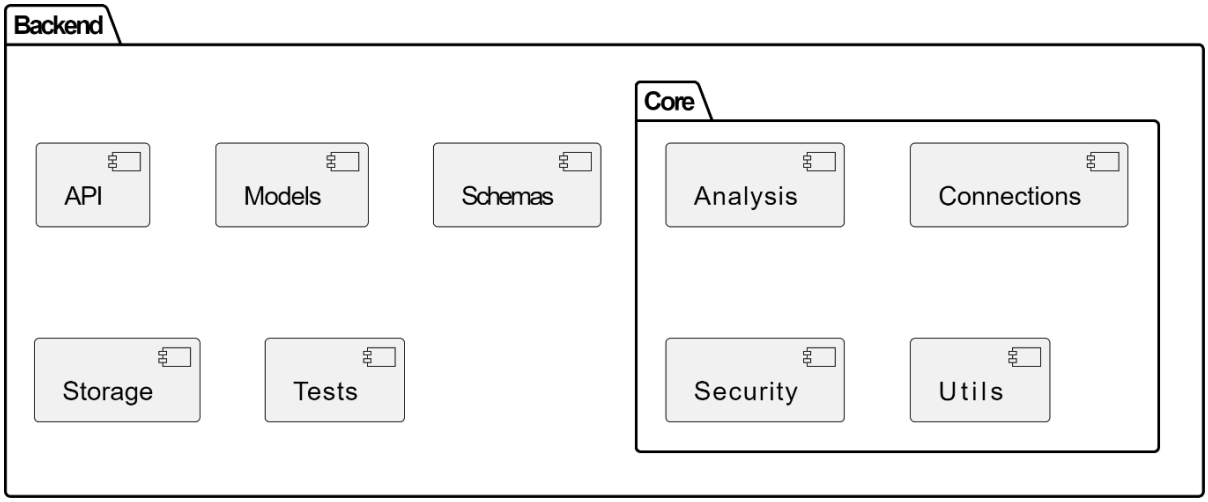


Рисунок 3.2 — Схема серверної частини

Опис шарів архітектури:

API: обробляє запити від клієнтської частини, забезпечуючи взаємодію між фронтендом і сервером.

Models: містить визначення моделей бази даних, що описують структуру даних.

Schemas: відповідає за валідацію та серіалізацію даних, гарантуючи їхню коректність.

Storage: забезпечує зберігання зашифрованих файлів даних із унікальними ідентифікаторами.

Tests: включає інструменти для тестування функціоналу системи.

Core: об’єднує ключові модулі, такі як:

Analysis: реалізує алгоритми аналізу даних, зокрема регресійний аналіз.

Connections: керує взаємодією з зовнішніми сервісами та базами даних.

Security: забезпечує аутентифікацію, авторизацію та захист даних.

Utils: містить допоміжні функції для загального використання.

Ця структура дозволяє ефективно організувати код, спрощує додавання нових функцій і забезпечує надійність системи.

Архітектура клієнтської частини

Клієнтська частина розроблена за архітектурою Feature Sliced Design (FSD), яка базується на розділенні функціоналу на модулі, пов’язані з бізнес-сутностями. Такий підхід полегшує розробку, тестування та масштабування інтерфейсу. Схема клієнтської частини представлена на рисунку 3.3.

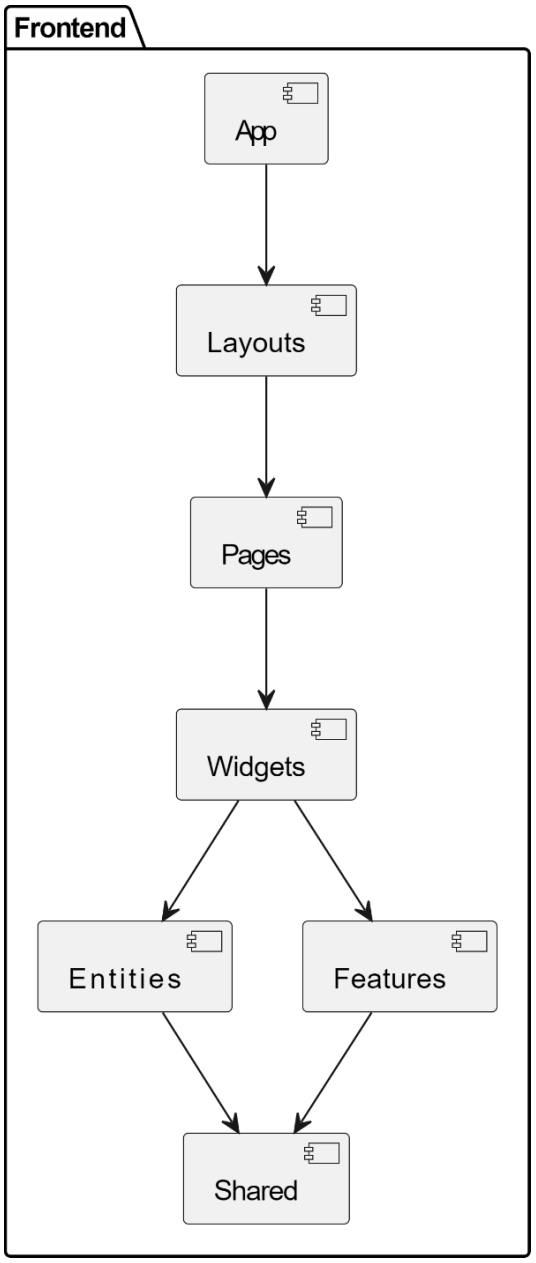


Рисунок 3.3 — Схема клієнтської частини

Опис шарів архітектури:

App: ініціалізує застосунок, налаштовує глобальні стилі та плагіни.

Layouts: визначає основні структурні компоненти сторінок, такі як навігація чи футер.

Pages: об’єднує віджети та компоненти для формування цілісних сторінок.

Widgets: містить окремі компоненти для відображення даних або взаємодії з користувачем.

Entities: відповідає за бізнес-логіку та моделі даних, що використовуються в застосунку.

Features: реалізує конкретні функціональні можливості, такі як фільтрація чи регресійний аналіз.

Shared: включає спільні ресурси, такі як стилі, утиліти та константи.

FSD-архітектура забезпечує чітке розділення функціоналу, що дозволяє легко оновлювати окремі компоненти без впливу на решту системи.

Переваги архітектурного підходу

Розділення на бекенд і фронтенд надає низку переваг. Для розробників це спрощує паралельну роботу над різними частинами системи, підвищує читабельність коду та полегшує інтеграцію нових функцій. Для користувачів такий підхід забезпечує швидку взаємодію з інтерфейсом, адаптивність до різних пристроїв і безпечну обробку даних на сервері. Використання багатошарової архітектури для бекенду та FSD для фронтенду створює надійну основу для створення ефективного веб-застосунку.

3.4 Розробка серверного модуля

Серверна частина веб-застосунку є основою для обробки даних, забезпечення безпеки та взаємодії з клієнтською частиною. Вона реалізована з використанням сучасних технологій і складається з кількох ключових компонентів: API, моделей, схем, ядра, сховища та тестів. Кожен компонент виконує специфічні функції, забезпечуючи надійність, продуктивність і зручність роботи системи.

API

Шар API відповідає за обробку запитів від клієнтської частини та забезпечує зв’язок між фронтендом і сервером. Для реалізації API використано фреймворк FastAPI, який відзначається високою швидкістю та підтримкою асинхронного програмування. API підтримує ключові операції з наборами даних: додавання, видалення та оновлення.

Авторизація користувачів здійснюється за допомогою JSON Web Tokens (JWT). При вході користувач вводить електронну пошту та пароль, після чого сервер генерує токен доступу з обмеженим терміном дії та токен оновлення для автоматичного продовження сесії. Це забезпечує безпеку та

зручність. Завантажені набори даних шифруються за допомогою бібліотеки Cryptography з використанням ключа, що зберігається в змінних оточення. Кожен датасет отримує унікальний ідентифікатор, який записується до бази даних, що спрощує їх керування.

API також дозволяє отримувати результати аналізу, такі як лінійна чи поліноміальна регресія, у зручному форматі, що розширює можливості користувачів для прогнозування та прийняття рішень.

Моделі

Шар моделей визначає структуру даних, які зберігаються в базі даних. Моделі створено за допомогою SQLModel і Pydantic, що забезпечує зручну роботу з даними та їх валідацію. Кожна модель містить обов’язкові поля: унікальний ідентифікатор (id), дати створення (created_at) та оновлення (updated_at). Додаткові методи дозволяють швидко зберігати, редагувати та видаляти записи, зменшуючи дублювання коду та підвищуючи ефективність розробки. Такий підхід гарантує цілісність даних і швидкий доступ до них.

Схеми

Шар схем відповідає за валідацію та серіалізацію даних, що проходять через API. Використання Pydantic-моделей дозволяє визначити структуру даних для запитів і відповідей, забезпечуючи їх коректність. Схеми перевіряють типи даних і формат, що надходять від користувачів, та форматують вихідні дані для клієнтської частини. Це зменшує ймовірність помилок і підвищує надійність обміну даними.

Ядро

Ядро серверної частини об’єднує основну логіку застосунку та складається з кількох підшарів:

Analysis: містить алгоритми для аналізу даних, зокрема лінійної та поліноміальної регресії, що забезпечують прогнозування та обробку даних.

Connections: керує підключенням до бази даних PostgreSQL, забезпечуючи стабільну взаємодію.

Security: реалізує механізми шифрування, аутентифікації та захисту від зовнішніх загроз.

Utils: включає допоміжні функції для логування, обробки помилок і

форматування даних.

Ця структура ядра забезпечує гнучкість і дозволяє легко розширювати функціонал системи.

Сховище

Шар сховища відповідає за збереження зашифрованих файлів даних, завантажених користувачами. Кожен файл, наприклад, у форматі CSV, отримує унікальну назву, яка пов’язана з записом у базі даних PostgreSQL. Шифрування даних захищає їх від несанкціонованого доступу, а унікальні ідентифікатори спрощують швидкий доступ і обробку.

Тести

Шар тестів забезпечує перевірку функціональності серверної частини. Він включає набір тестів, фікстур і тестових датасетів, які використовуються для перевірки регресійних алгоритмів. Тести покривають ключові сценарії, такі як коректність прогнозів і обробки даних, що гарантує надійність системи. Використання фікстур дозволяє ізолювати тестові дані, підвищуючи точність результатів.

Такий підхід до розробки серверного модуля забезпечує створення надійної, безпечної та масштабованої системи, яка ефективно обробляє дані та підтримує потреби користувачів.

3.5 Реалізація механізмів аналізу даних

Реалізація механізмів аналізу даних є центральним елементом веб-застосунку, оскільки забезпечує обробку та інтерпретацію даних для отримання цінних результатів. У цьому розділі розглянуто алгоритми лінійної та поліноміальної регресії, метод градієнтного спуску для їх оптимізації, а також підходи до візуалізації результатів аналізу. Використання бібліотек NumPy, Matplotlib і Chart.js дозволяє ефективно обробляти дані та представляти їх у зрозумілому вигляді.

Алгоритми аналізу даних

Алгоритми аналізу даних є основою для прогнозування та виявлення закономірностей у наборах даних. Для їх реалізації використано мову Python

і бібліотеку NumPy, яка забезпечує швидку обробку числових масивів. Базовий інтерфейс для регресійних моделей реалізовано через клас BaseRegression, який визначає ключові методи та константи для всіх типів регресії (див. Додаток А).

Опис базового класу BaseRegression:

Константи:

DEFAULT_ALPHA — початкове значення швидкості навчання (learning rate), яке використовується за відсутності заданого значення.

DEFAULT_ITERATIONS — максимальна кількість ітерацій для алгоритму градієнтного спуску.

DEFAULT_THRESHOLD — мінімальний розмір кроку для зупинки оптимізації.

MAX_DATASET_SIZE — максимальний розмір набору даних, для якого застосовується звичайний градієнтний спуск; при перевищенні використовується стохастичний градієнтний спуск.

Методи:

Конструктор ініціалізує залежні та незалежні змінні (масиви x і y).

Метод _get_data повертає набір даних для ітерації градієнтного спуску, включаючи випадкову підмножину для стохастичного режиму.

Метод _fit (абстрактний) реалізує навчання моделі регресії.

Метод _predict (абстрактний) прогнозує вихідні дані на основі навченої моделі.

Метод _mean_squared_error (абстрактний) обчислює середньоквадратичну похибку (MSE).

Метод fit є обгорткою для _fit, автоматично підбираючи швидкість навчання.

Метод predict викликає _predict для прогнозування.

Метод error повертає значення MSE через _mean_squared_error.

Цей клас забезпечує уніфікований інтерфейс, що унеможливлює дублювання коду в класах регресії.

Лінійна регресія

Клас LinearRegression успадковує BaseRegression і реалізує алгоритм

лінійної регресії з використанням градієнтного спуску (див. Додаток В). Він призначений для моделювання лінійних залежностей між змінними.

Опис класу LinearRegression:

Конструктор: ініціалізує модель, нормалізує вхідні дані (x і y) для підвищення точності обчислень, задає початкові значення для нахилу (theta) та перетину з віссю Y (intercept).

Метод _fit: використовує градієнтний спуск для калібрування параметрів theta та intercept за методом найменших квадратів.

Метод _predict: повертає прогнозовані значення, виконуючи зворотне масштабування для відповідності вихідному формату даних.

Метод _mean_squared_error: обчислює середньоквадратичну похибку моделі.

Властивості theta **та** intercept: повертають відкалібровані значення параметрів.

Приклад результатів лінійної регресії на датасеті, що містить години навчання студентів та їхні оцінки, представлено на рисунку 3.3.

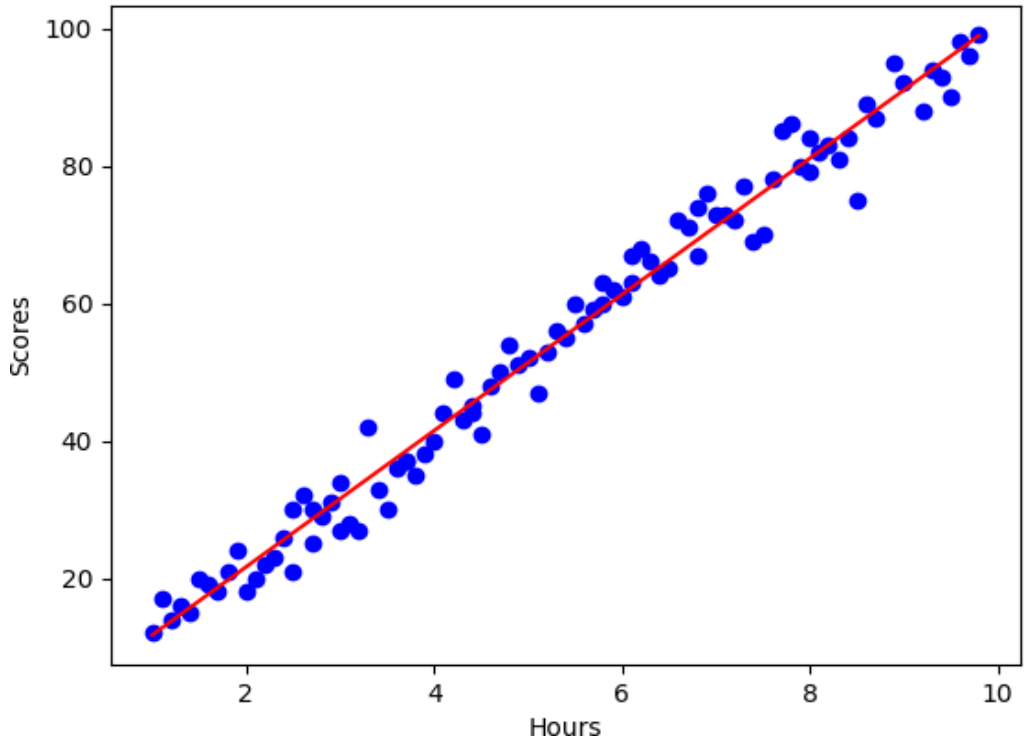


Рисунок 3.3 — Візуалізація лінійної регресії (показує точкову діаграму з даними та лінію регресії, що ілюструє залежність між змінними, дозволяючи оцінити точність моделі.)

Поліноміальна регресія

Клас PolynomialRegression, який також успадковує BaseRegression, реалізує поліноміальну регресію для моделювання нелінійних залежностей (див. Додаток С). Основна відмінність полягає у формуванні поліноміальних ознак шляхом піднесення змінних до різних ступенів.

Опис класу PolynomialRegression:

Конструктор: ініціалізує модель, задає ступінь полінома (за замовчуванням 2), нормалізує дані та створює масив коефіцієнтів.

Метод _polynomial_features: генерує поліноміальні ознаки для вхідних даних.

Метод _fit: калібрує коефіцієнти полінома за допомогою градієнтного спуску.

Метод _predict: прогнозує значення, використовуючи поліноміальні ознаки та зворотне масштабування.

Метод _mean_squared_error: обчислює MSE для оцінки точності моделі.

Властивість coefficients: повертає відкалібровані коефіцієнти полінома.

Приклад виконання поліноміальної регресії другого порядку на датасеті продажів морозива залежно від температури представлено на рисунку 3.4

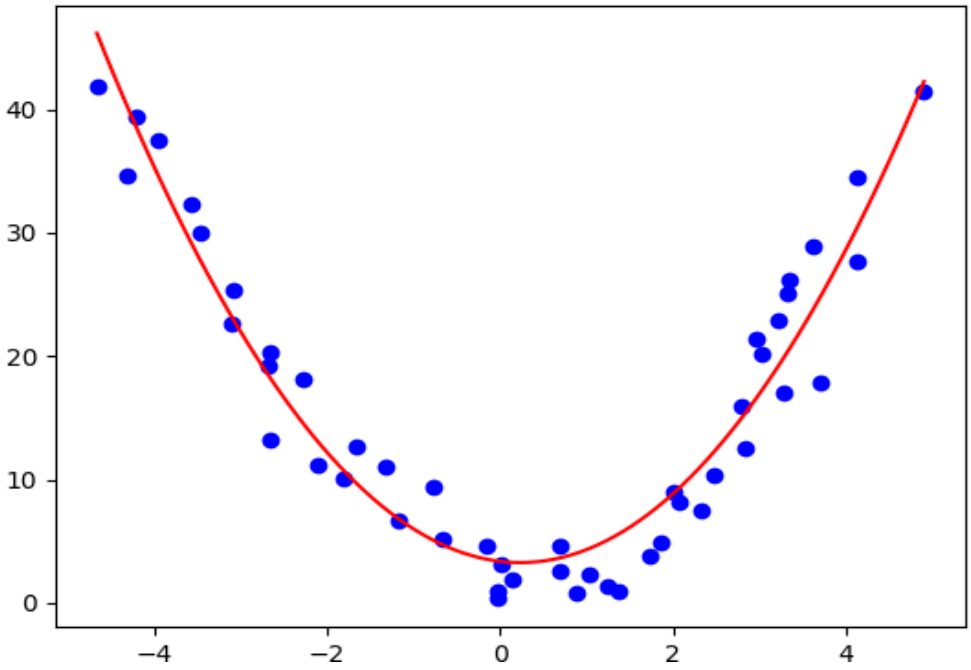


Рисунок 3.4 — Візуалізація поліноміальної регресії другого порядку (демонструє точкову діаграму та криву регресії, що відображає нелінійну залежність між змінними).

Візуалізація результатів аналізу

Візуалізація є важливим інструментом для інтерпретації результатів аналізу, оскільки дозволяє виявляти закономірності та оцінювати точність моделей. У веб-застосунку для цього використовується бібліотека Chart.js, яка інтегрується з фронтендом для створення інтерактивних графіків. Результати лінійної та поліноміальної регресії відображаються у вигляді точкових діаграм, де точки представляють фактичні дані, а лінія чи крива — прогнозовані значення. Це дає змогу користувачам швидко оцінити відповідність моделі даним і виявити аномалії.

Використання бібліотеки Matplotlib на сервері дозволяє створювати статичні візуалізації для тестування та перевірки алгоритмів, як показано на рисунках 3.3 та 3.4. Chart.js на клієнтській частині забезпечує динамічну взаємодію, дозволяючи користувачам налаштовувати графіки для аналізу.

Такий підхід до реалізації механізмів аналізу даних забезпечує ефективну обробку, точне прогнозування та наочне представлення результатів, що відповідає потребам користувачів.

3.6 Розробка інтерфейсу користувача

Користувацький інтерфейс веб-застосунку розроблено з урахуванням зручності та інтуїтивності, щоб забезпечити легку взаємодію для користувачів різного рівня підготовки. Для створення інтерфейсу використано сучасний стек технологій: фреймворк Vue.js для управління структурою та станом, бібліотека Vuetify для готових компонентів інтерфейсу, Vue Query для ефективної роботи з асинхронними запитами та Chart.js для створення інтерактивних графіків. Такий підхід забезпечує швидку реакцію інтерфейсу, адаптивність та естетичний вигляд. Нижче розглянуто основні сторінки застосунку та їх функціонал.

Технології інтерфейсу

Використання Vue.js дозволяє створювати динамічний та реактивний інтерфейс, де компоненти автоматично оновлюються при зміні даних. Vuetify надає набір стильних і готових до використання компонентів, таких як кнопки, таблиці та форми, що спрощує розробку та забезпечує однаковий вигляд на різних пристроях. Vue Query оптимізує обробку запитів до API, забезпечуючи кешування даних і синхронізацію стану, що зменшує час очікування для користувача. Chart.js використовується для створення інтерактивних графіків, які дозволяють користувачам візуально аналізувати дані та результати регресійного аналізу.

Сторінка авторизації

Сторінка авторизації забезпечує безпечний вхід до системи. Вона містить форму з полями для введення електронної пошти та пароля, а також кнопку «Sign in» для підтвердження (див. Рисунок 3.5).

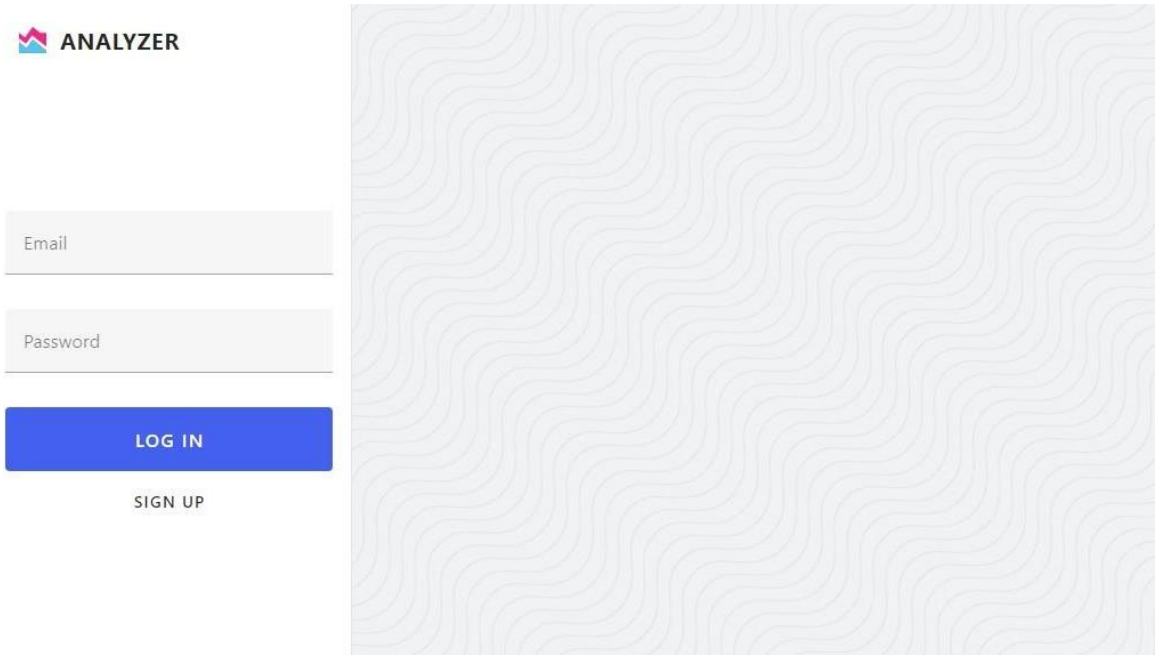


Рисунок 3.5 — Сторінка авторизації (демонструє простий і зрозумілий дизайн, де користувач може швидко авторизуватися, ввівши свої дані. У разі помилки введення з’являється повідомлення, що допомагає уникнути неправильних спроб входу).

Сторінка реєстрації

Сторінка реєстрації дозволяє створювати нові облікові записи. Форма включає поля для введення електронної пошти, імені, пароля та підтвердження пароля, а також кнопку «Sign Up» для завершення реєстрації (див. Рисунок 3.6)

.

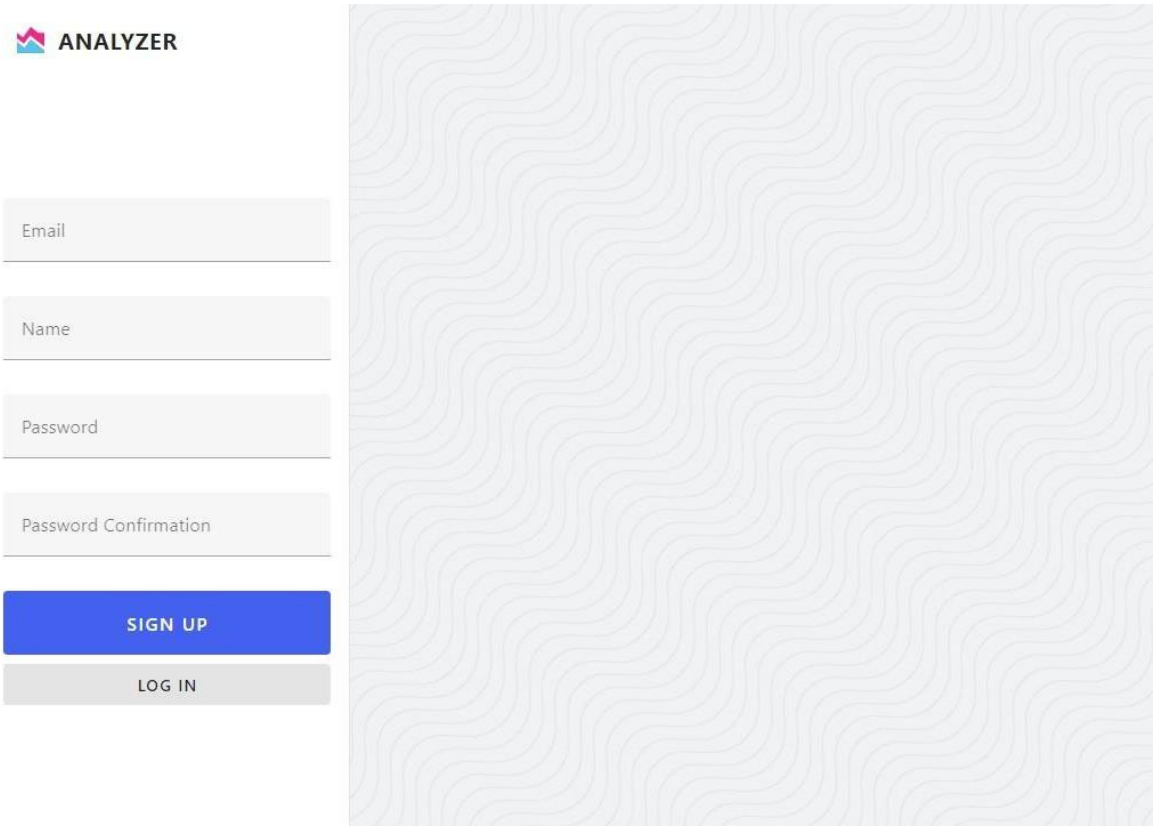


Рисунок 3.6 — Сторінка реєстрації (показує зручний інтерфейс із можливістю переходу до сторінки авторизації для користувачів, які вже мають обліковий запис. Дизайн форми інтуїтивний, із чіткими підказками для заповнення).

Домашня сторінка

Домашня сторінка є центральною точкою входу до застосунку. Для неавторизованих користувачів вона містить привітальне повідомлення та пропозицію зареєструватися чи увійти (див. Рисунок 3.7).

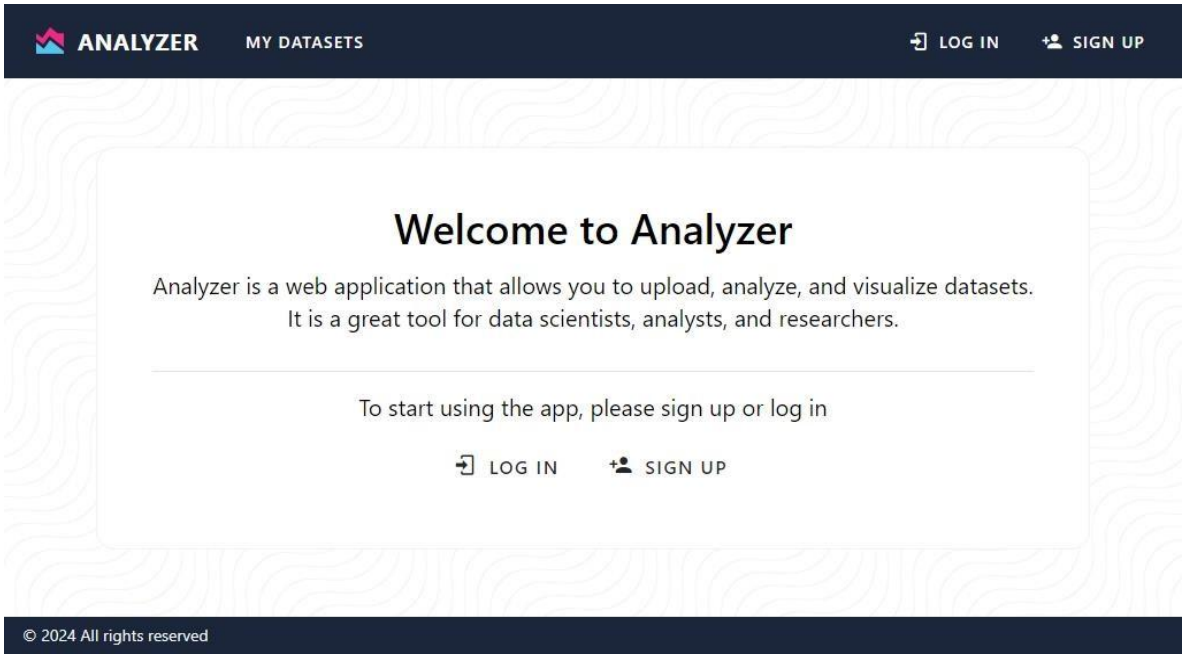


Рисунок 3.7 — Домашня сторінка для неавторизованих користувачів (відображає заклик до дії для початку роботи)

Для авторизованих користувачів сторінка пропонує кнопку для переходу до розділу з наборами даних (див. Рисунок 3.8)

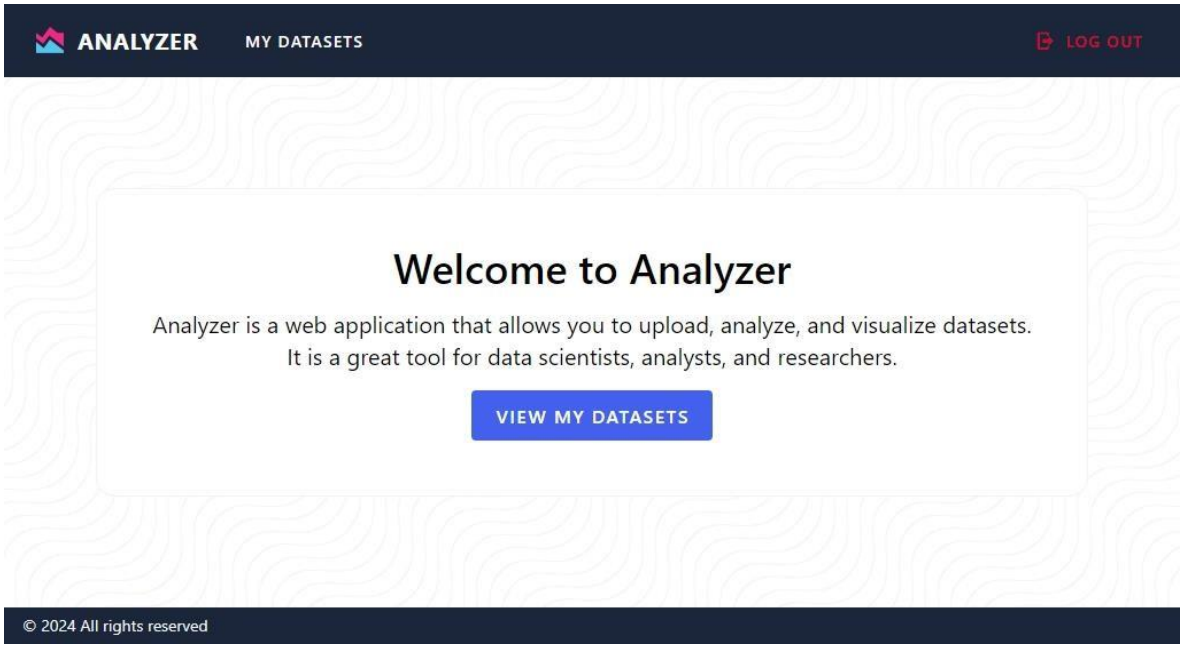


Рисунок 3.8 — Домашня сторінка для авторизованих користувачів (демонструє персоналізований інтерфейс із швидким доступом до основних функцій).

Сторінка датасетів

Сторінка «My Datasets» призначена для керування наборами даних.

Вона включає панель інструментів із кнопками «Add dataset» для додавання нових даних і «Delete» для видалення вибраних наборів (активується після вибору датасету). Список наборів даних відображається у вигляді елементів із назвою, датою оновлення, чекбоксом для вибору та кнопкою видалення (див. Рисунок 3.9).

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.

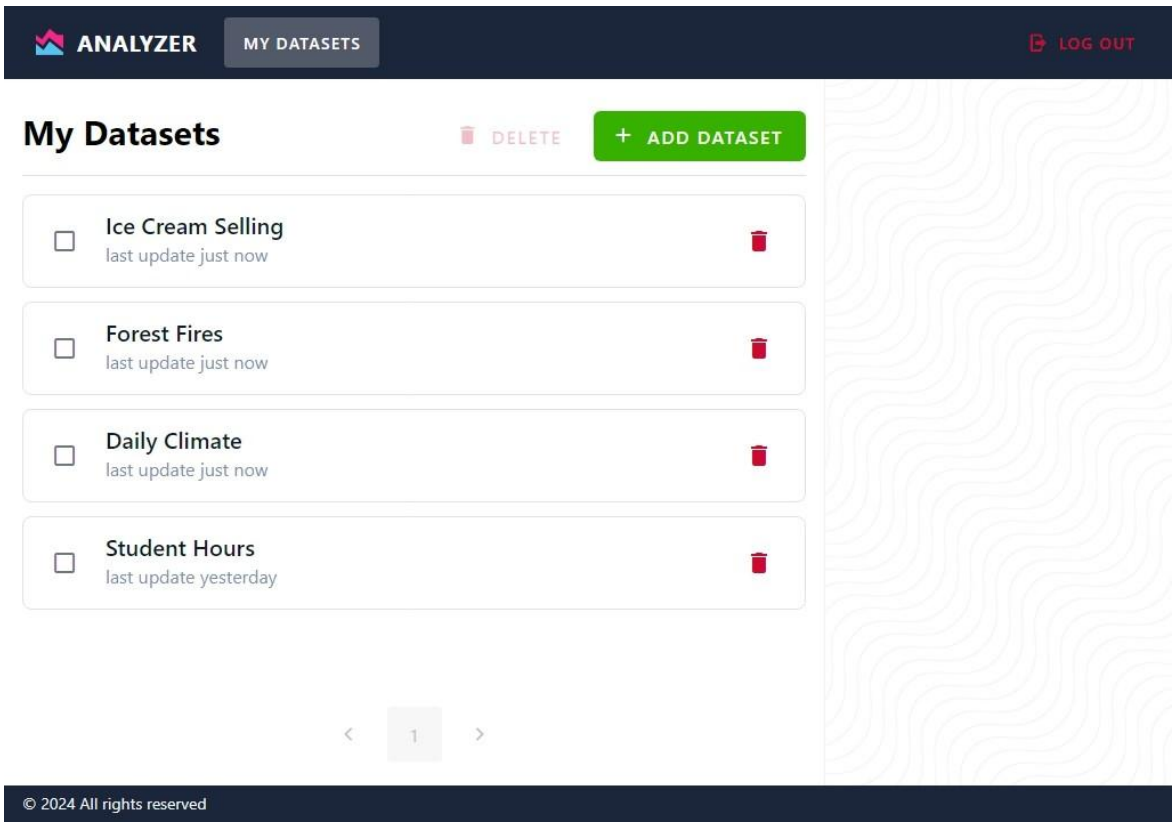


Рисунок 3.9 — Сторінка «My Datasets» (показує зручний інтерфейс для навігації та керування даними)

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.

Діалогове вікно для додавання датасету містить поля для назви, опису та завантаження файлу (CSV, XLSX тощо) із кнопками «Cancel» і «Create» (див. Рисунок 3.10)

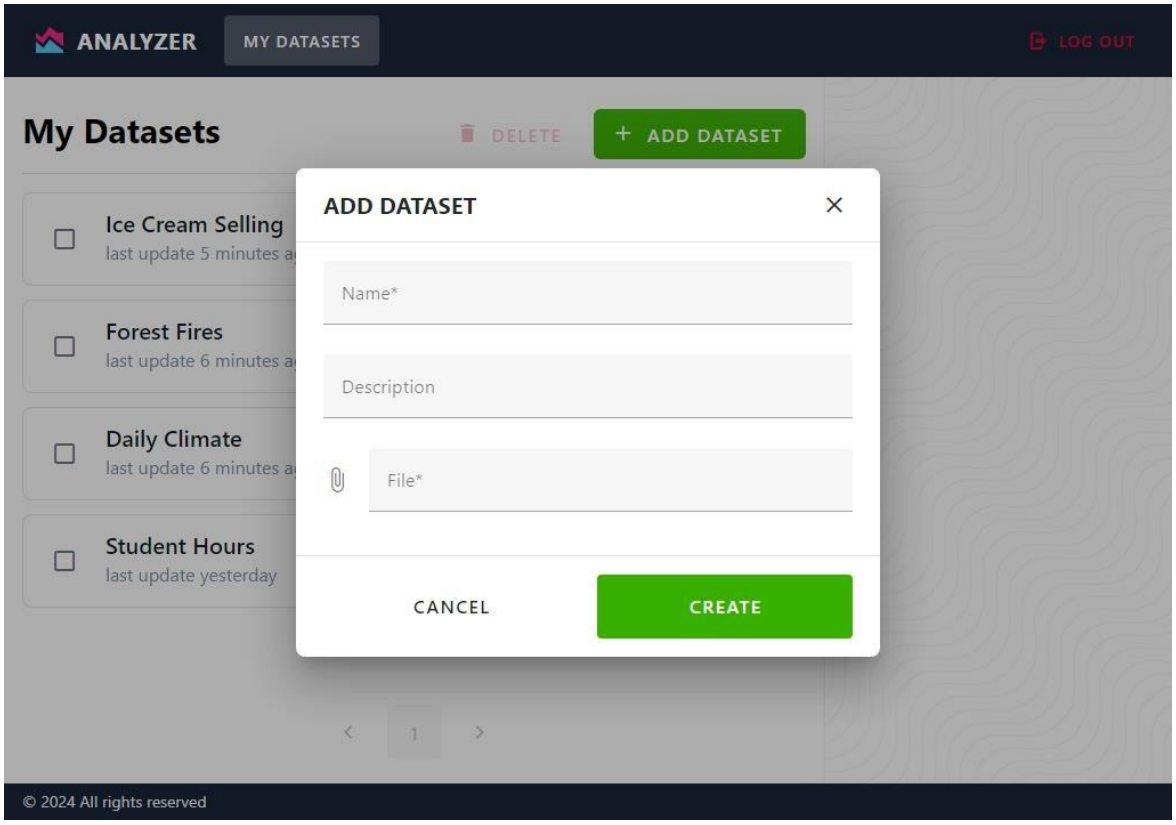


Рисунок 3.10 — Діалогове вікно додавання нового набору даних (ілюструє форму для зручного завантаження даних)

Сторінка детального перегляду набору даних

Сторінка детального перегляду дозволяє користувачам аналізувати дані в табличному та графічному вигляді. Таблиця підтримує сортування, фільтрацію та пагінацію, а1 графік відображає дані для швидкого аналізу

тенденцій (див. Рисунок 311) 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.

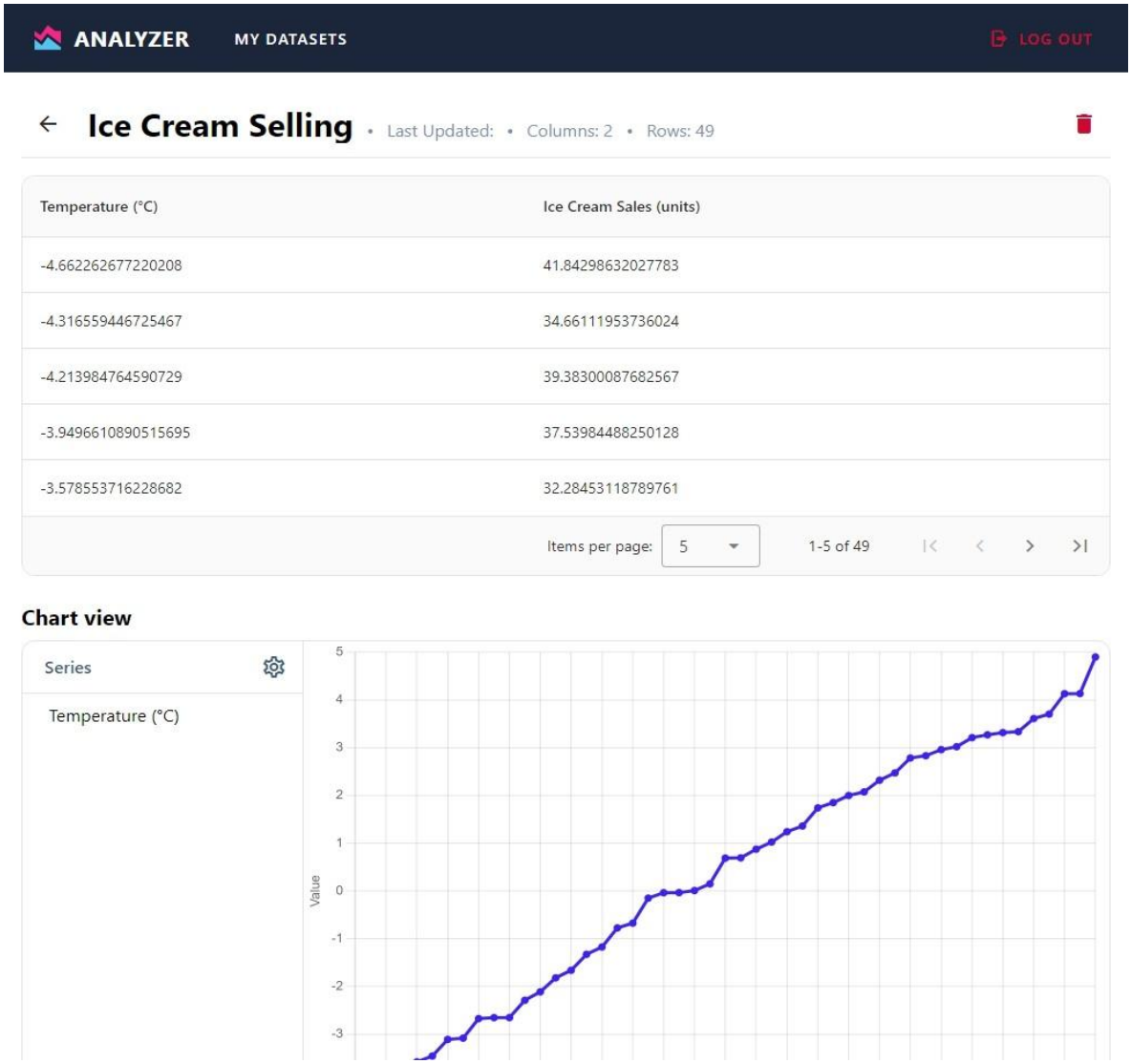


Рисунок 3.11 — Сторінка детального перегляду завантаженого набору даних (демонструє зручне розташування таблиці та графіка).

Налаштування графіка доступні через діалогове вікно, де користувач

може вибрати колонки, тип даних і стиль відображення (див. Рисунок 3.12)

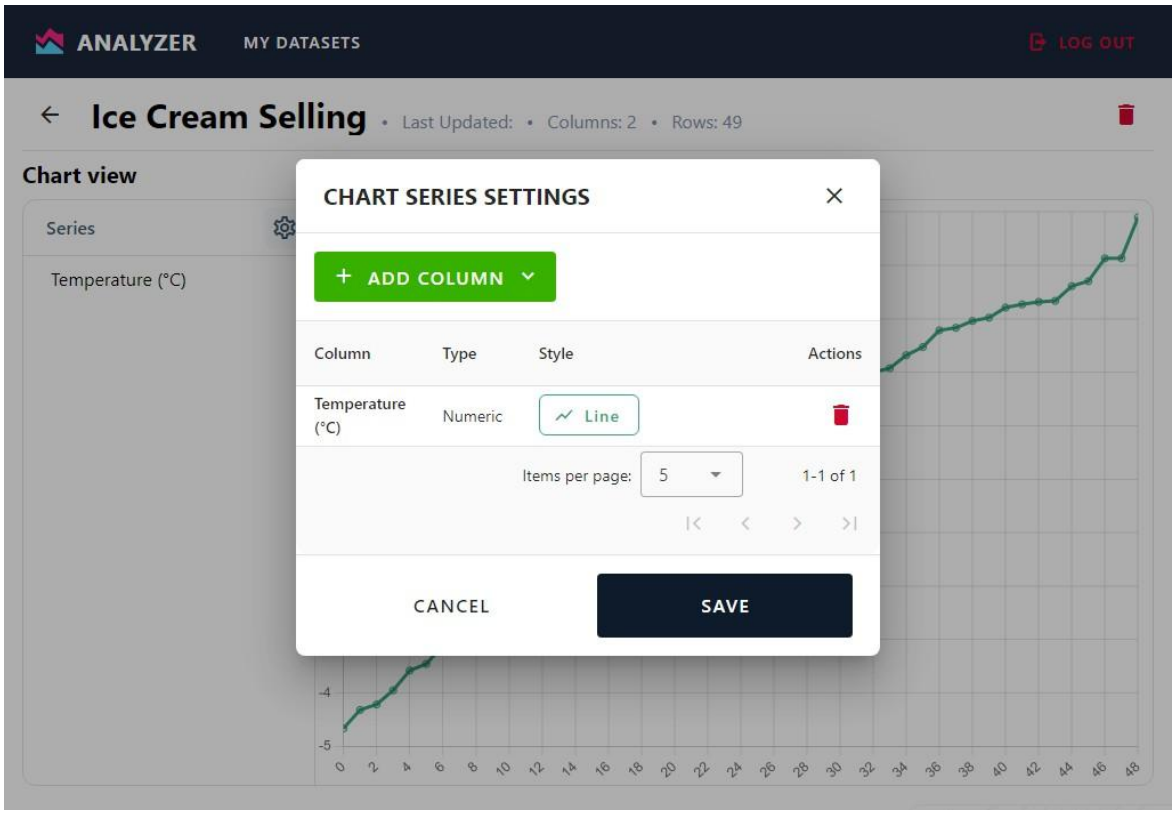


Рисунок 3.12 — Діалогове вікно налаштування графіку перегляду (показує форму для гнучкого налаштування візуалізаці)ї.

Регресійний аналіз

Блок регресійного аналізу дозволяє проводити лінійну або поліноміальну регресію. Форма включає випадаючий список для вибору типу регресії, колонок X і Y, а також кнопку «Calculate» для запуску аналізу (див. Рисунок 3.13)..

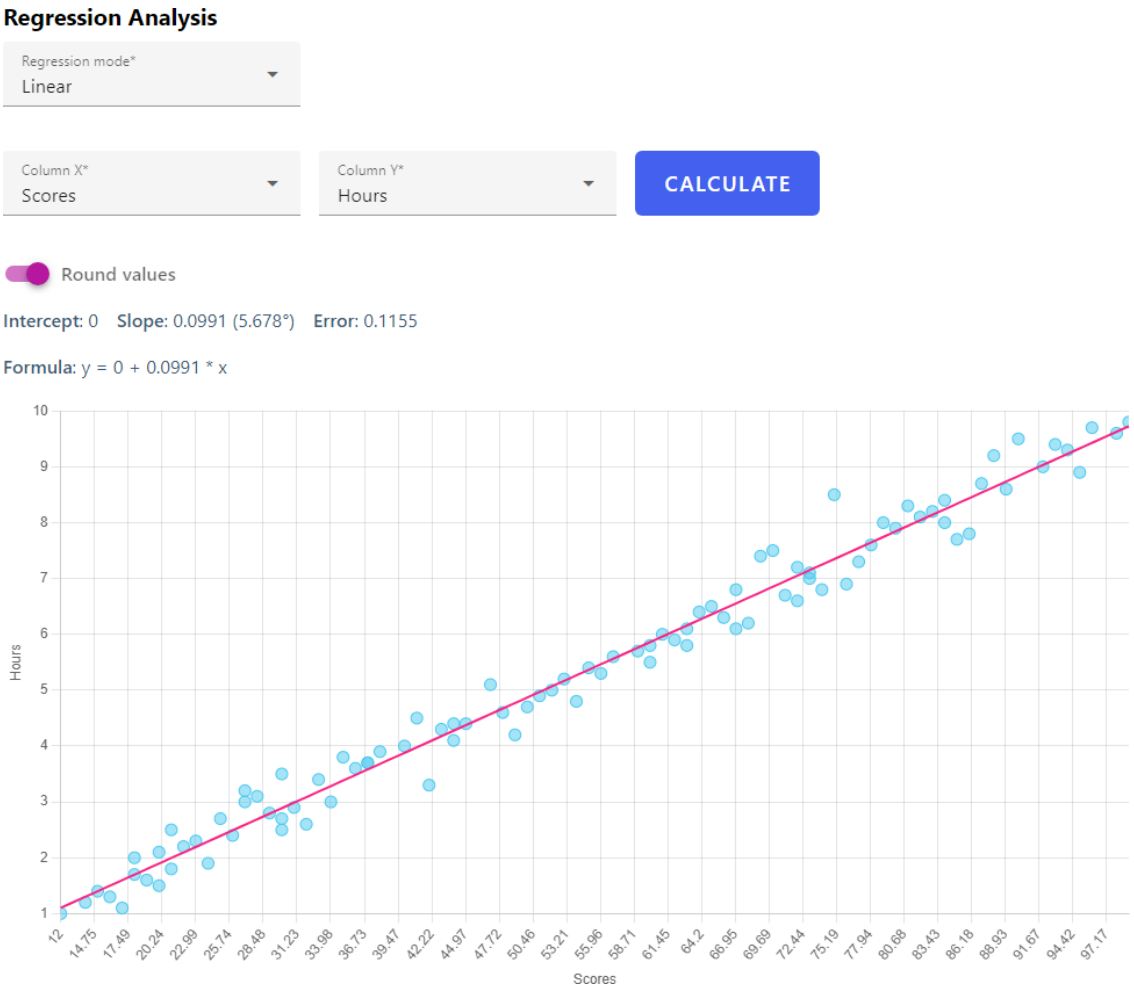


Рисунок 3.13 — Результати лінійної регресії відображає(графік із точками даних і лінією регресії, що допомагає оцінити точність моделі)

Для поліноміальної регресії повертаються додаткові параметри, такі як ступінь полінома (див. Рисунок 3.14)..

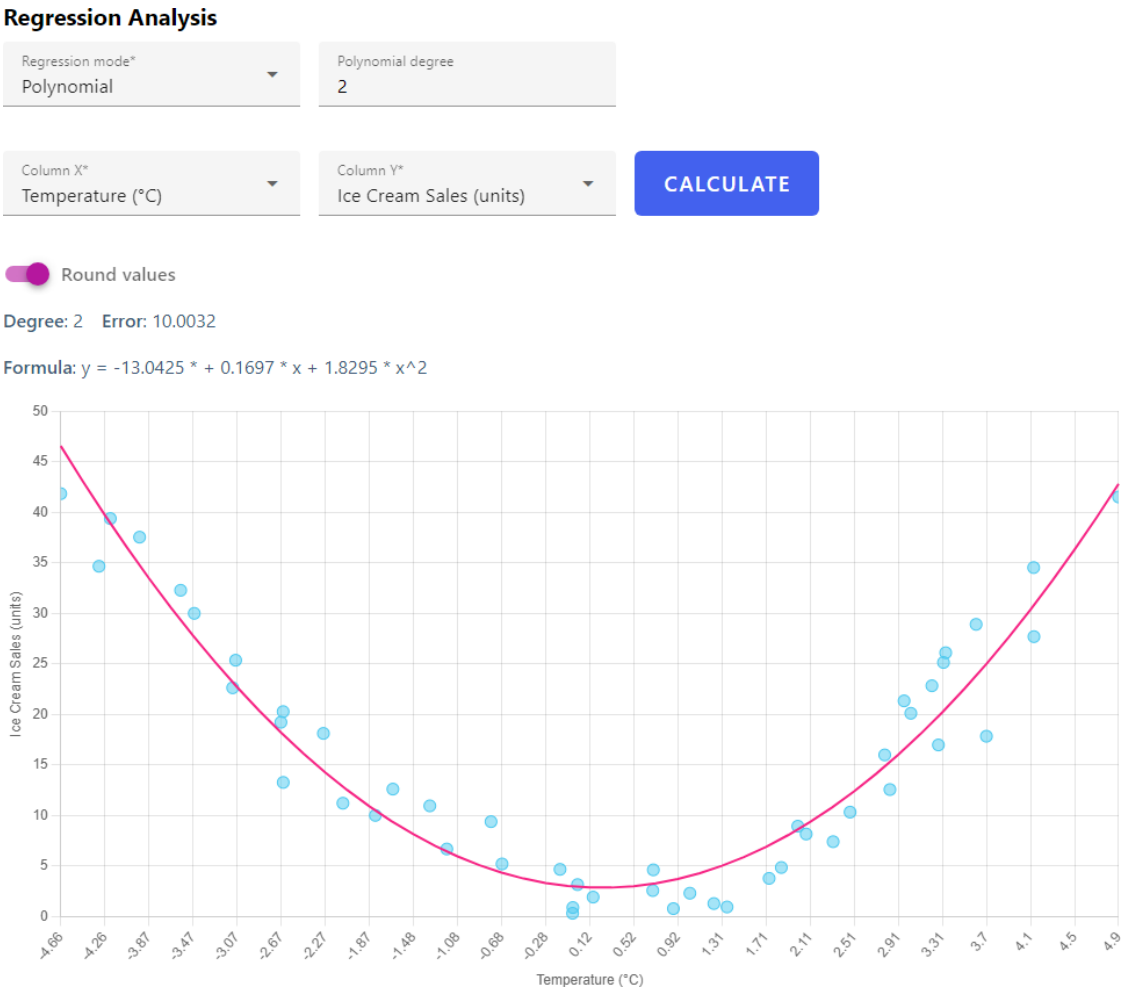


Рисунок 3.14 — Результати поліноміальної регресії другого порядку (показує криву регресії та статистичні показники, округлені до чотирьох знаків після коми, що полегшує інтерпретацію)

Такий інтерфейс забезпечує зручну взаємодію з даними, поєднуючи простоту використання з потужним функціоналом для аналізу, що робить застосунок доступним для широкого кола користувачів.

3.7 Перевірка функціональності та надійності

Перевірка функціональності та надійності веб-застосунку є ключовим етапом розробки, що гарантує коректність роботи системи, зокрема регресійних алгоритмів, які є основою для прогнозування та аналізу даних. Тестування дозволяє виявити помилки, забезпечити точність результатів і підвищити якість коду. У цьому розділі розглянуто методологію тестування, перевірку алгоритмів лінійної та поліноміальної регресії, а також додаткові аспекти, такі як інтеграційне тестування та відладка.

Методологія тестування

Для тестування серверної частини застосунку використано бібліотеку pytest, яка забезпечує зручне створення та виконання тестів. Тести охоплюють основні функціональні компоненти, з акцентом на регресійні алгоритми. Їх результати порівнюються з бібліотекою scikit-learn, відомою своєю надійністю, щоб гарантувати точність власної реалізації. Додатково проводилося інтеграційне тестування для перевірки взаємодії між API, базою даних і алгоритмами аналізу, а також відладка для усунення помилок, виявлених під час виконання тестів. Тестові дані організовано в ізольованому середовищі з використанням фікстур для забезпечення повторюваності.

Ініціалізація тестів

Тестове середовище включає фікстури та тестові датасети, які забезпечують підготовку даних для перевірки алгоритмів. Фікстури створюють ізольовані набори даних, такі як синтетичні масиви чи реальні датасети (наприклад, дані про продажі морозива або оцінки студентів). Реалізація фікстур наведена в Додаток D. Цей підхід спрощує тестування, дозволяючи швидко налаштовувати тестові сценарії та уникати дублювання коду.

Тестування лінійної регресії

Перевірка алгоритму лінійної регресії зосереджена на коректності обчислень параметрів моделі, прогнозів і середньоквадратичної похибки (MSE). Тести порівнюють результати власної реалізації з результатами класу LinearRegression із бібліотеки scikit-learn (див. Додаток F). Основні аспекти

перевірки включають точність коефіцієнтів моделі, прогнозованих значень і MSE, що забезпечує надійність алгоритму для прогнозування лінійних залежностей.

Тестування поліноміальної регресії

Для поліноміальної регресії тести перевіряють коректність обчислень коефіцієнтів полінома, прогнозів і MSE, порівнюючи їх із результатами PolynomialFeatures і LinearRegression із scikit-learn (див. Додаток F). Тести охоплюють різні датасети, включаючи нелінійні залежності, що дозволяє оцінити гнучкість алгоритму. Такий підхід гарантує, що модель правильно обробляє складніші зв’язки між змінними.

Перевірка результатів тестування

Результати тестування, отримані за допомогою команди pytest tests/, підтверджують коректність роботи алгоритмів (див. Додаток G). Усі 18 тестів для лінійної та поліноміальної регресії пройдено успішно, що свідчить про високу надійність реалізації. Виявлено одне попередження, яке не вплинуло на результати. Додаткове інтеграційне тестування підтвердило стабільну взаємодію між компонентами, а відладка дозволила усунути незначні помилки, виявлені на етапі тестування.

Цей підхід до тестування забезпечує високу якість і надійність веб-застосунку, гарантуючи точність прогнозів і стабільність роботи системи.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи створено універсальний веб-застосунок для прикладного статистичного аналізу даних, який відповідає сучасним потребам користувачів. Проведений аналіз підтвердив актуальність розробки таких інструментів в умовах зростання обсягів даних і попиту на доступні аналітичні рішення.

На основі вивчення потреб користувачів і порівняння аналогів (SPSS, Statistica, Datapine, Datatab.net) сформовано вимоги до платформи, що включають підтримку популярних форматів даних, інтуїтивний інтерфейс і потужні аналітичні можливості. Розроблено інформаційну модель із реляційною структурою на базі PostgreSQL, яка забезпечує ефективне зберігання та обробку даних. Серверна частина, реалізована за допомогою FastAPI, підтримує асинхронну обробку, шифрування даних і безпечну авторизацію через JWT. Клієнтська частина, створена на Vue.js із використанням Vuetify і Chart.js, забезпечує інтерактивну взаємодію, включаючи фільтрацію, сортування та візуалізацію даних.

Алгоритми лінійної та поліноміальної регресії, реалізовані через NumPy з оптимізацією градієнтним спуском, дозволяють точно прогнозувати тенденції. Їх надійність підтверджена тестами за допомогою pytest, які показали відповідність результатів бібліотеці scikit-learn. Проведено інтеграційне тестування, що забезпечило стабільну взаємодію між компонентами системи.

Розроблений веб-застосунок є зручним і потужним інструментом, який знижує бар’єри для аналізу даних, роблячи його доступним для широкого кола користувачів — від аналітиків до студентів. Платформа підтримує автоматизацію звітності, інтерактивні дашборди та адаптивність до різних пристроїв, що робить її придатною для використання в бізнесі, освіті та наукових дослідженнях. У перспективі можливе розширення функціоналу, наприклад, інтеграція нових алгоритмів чи підтримка хмарних сервісів, що підвищить її конкурентоспроможність..

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
7. Getting Started: Navigating the Statistica User Interface. URL: https://www.youtube.com/watch?v=W7w2yFz-JwU&list=PLnmoGGHHJldizem1Mzu4AJi7qSR9zdV3o&index=2
8. A software for everyone from managers to data scientists. URL:.
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.
11. Vue.js. URL: https://vuejs.org/
12. Learn React. URL: https://react.dev/learn/describing-the-ui Overview of ASP.NET Core. URL: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0.
13. Welcome to Flask. URL: https://flask.palletsprojects.com/en/3.0.x/
14. FastAPI. URL: https://fastapi.tiangolo.com/
15. About SQLite. URL: https://www.sqlite.org/about.html.
16. About PostgreSQL. URL: https://www.postgresql.org/about/.
17. What is MongoDB. URL: https://www.mongodb.com/company/what-is-mongodb
18. MongoDB Docs. URL: https://www.mongodb.com/docs.

19. Gradient Descent. URL: https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/what-is-gradient-descent .
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: 14.06.2025).

ДОДАТКИ

Д одаток А

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()
```

Д одаток В

Лістинг 2 Реалізація класу лінійної регресії

```
import numpy as np
from core.analysis.regressions.base_regression import
BaseRegression
from core.analysis.utils import LearningRateTooHigh
56
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.
57
: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
58
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
```

Д одаток С

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
```

Д одаток D

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)
```

Д одаток Е

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)
```

Д одаток F

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)
```

Д одаток G

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
======================
```