БАКАЛАВРСЬКА РОБОТА

БР. ІП - 74.00.00.000 ПЗ

Група ІП-22-3

Магомета Тарас

2026

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Магомета Тарас Євгенович

(прізвище, ім'я, по батькові)

УДК 004.4

(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка програмної системи тестування моделей машинного навчання

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач освітнього рівня Магомета Т.Є.

(підпис, ініціали та прізвище здобувача)

Науковий керівник Юрчишин Володимир Миколайович, д.т.н., професор

(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2026

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ ____ ” _____ 2026 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Магометі Тарасу Євгеновичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “ Розробка програмної системи тестування моделей машинного навчання”

керівник проекту (роботи) Юрчишин Володимир Миколайович, д.т.н., професор

затверджені наказом закладу вищої освіти від “ 21 ” квітня 2026р. № 179/7

2. Строк подання студентом проекту (роботи) 10 червня 2026 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної області використання машинного навчання в освіті
2. Алгоритмічне забезпечення системи тестування моделей машинного навчання
3. Проектування системи тестування моделей машинного навчання
4. Програмна реалізація системи візуалізації моделей машинного навчання
5. Архітектурна реалізація та програмна інженерія системи

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Функціональні завдання розробки системи (рис. 1.1, ст. 17)
2. Архітектура програмної системи (рис. 1.2, ст. 19)
3. Інтерфейс Orange Data Mining (рис. 1.3, ст. 21)
4. Інтерфейс платформи машинного навчання MLflow (рис. 1.4 ст. 25)
5. Інтерфейс платформи KNIME (рис. 1.5, ст. 25)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **21 квітня 2026 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Аналіз предметної області використання машинного навчання в освіті	04.05.2026	виконано
2	Алгоритмічне забезпечення системи тестування моделей машинного навчання	15.05.2026	виконано
3	Проектування системи тестування моделей машинного навчання	21.05.2026	виконано
4	Програмна реалізація системи візуалізації моделей машинного навчання	28.05.2026	виконано
5	Архітектурна реалізація та програмна інженерія системи	03.06.2026	виконано
6	Оформлення пояснювальної записки бакалаврської роботи завідувачем кафедри	10.06.2026	виконано

Студент – дипломник _____Магомета Т.Є.
(підпис)

Керівник роботи _____ Юрчишин В.М.
(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 81 сторінку, 21 рисунок, список використаних джерел із 34 найменуваннями.

Мета роботи: розробка програмної системи тестування моделей машинного навчання з інтерактивним візуальним середовищем, орієнтованої на використання в освітньому процесі.

Об'єктом дослідження є процес тестування та аналізу моделей машинного навчання в освітньому середовищі.

Предметом дослідження є методи, алгоритми та програмні засоби реалізації інтерактивної системи тестування моделей машинного навчання.

В першому розділі проведений аналіз предметної області дозволив обґрунтувати необхідність створення інтерактивної системи тестування моделей машинного навчання для освітніх цілей.

В другому розділі розроблено математичне та алгоритмічне забезпечення системи, визначено її вимоги та архітектуру

В третьому розділі виконано проєктування системи з використанням UML-моделей, що забезпечило формалізацію її структури та поведінки

В четвертому розділі реалізовано програмну систему з інтерактивним інтерфейсом та функціональними модулями обробки і тестування моделей

Висновок: розроблено ефективну програмну систему тестування моделей машинного навчання, що забезпечує інтерактивне навчальне середовище для аналізу даних і підвищення якості освітнього процесу

КЛЮЧОВІ СЛОВА: МАШИННЕ НАВЧАННЯ, ТЕСТУВАННЯ МОДЕЛЕЙ, ІНТЕРАКТИВНЕ НАВЧАННЯ, ВІЗУАЛІЗАЦІЯ ДАНИХ, ЛІНІЙНА РЕГРЕСІЯ, АНАЛІЗ ДАНИХ, ПРОГРАМНА СИСТЕМА, ОСВІТНІ ТЕХНОЛОГІЇ, ПРЕПРОЦЕСИНГ ДАНИХ

ANNOTATION

The bachelor's thesis contains 81 pages, 21 figures, a list of used sources with 34 names.

Purpose of the work: development of a software system for testing machine learning models with an interactive visual environment, focused on use in the educational process.

The object of the study is the process of testing and analyzing machine learning models in the educational environment.

The subject of the study is methods, algorithms and software tools for implementing an interactive system for testing machine learning models.

In the first section, the analysis of the subject area made it possible to justify the need to create an interactive system for testing machine learning models for educational purposes.

In the second section, the mathematical and algorithmic support of the system is developed, its requirements and architecture are determined

In the third section, the design system is performed using UML models, which provided formalization of its structure and behavior.

In the fourth section, a software system with an interactive interface and functional modules for processing and testing models is implemented.

Conclusion: an effective software system for testing machine learning models has been developed, which provides an interactive learning environment for data analysis and improving the quality of the educational process.

KEYWORDS: MACHINE LEARNING, MODEL TESTING, INTERACTIVE LEARNING, DATA VISUALIZATION, LINEAR REGRESSION, DATA ANALYSIS, SOFTWARE SYSTEM, EDUCATIONAL TECHNOLOGIES, DATA PREPROCESSING

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ML – Machine Learning

MI – Mutual Information

MSE – Mean Squared Error

CSV – Comma-Separated Values

HTML – HyperText Markup Language

NaN – Not a Number

$\mathbf{R^2}$ – Coefficient of Determination

SaaS – Software as a Service

L1 / L2 – Lasso and Ridge Regularization (types of regularization)

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується стрімким впровадженням методів машинного навчання у різні сфери діяльності, зокрема в освітню галузь. У зв’язку з цим зростає потреба у формуванні у здобувачів освіти не лише теоретичних знань, а й практичних навичок роботи з моделями машинного навчання, їх аналізу, тестування та інтерпретації результатів. Особливого значення набуває створення інтерактивних програмних середовищ, які дозволяють візуалізувати процеси навчання моделей та експериментувати з даними у зручній і доступній формі.

Традиційні підходи до викладання дисциплін, пов’язаних з аналізом даних, часто не забезпечують достатнього рівня залученості студентів, що знижує ефективність навчального процесу. Водночас існуючі програмні інструменти, хоча й мають потужний функціонал, зазвичай не адаптовані до освітніх потреб або є надто складними для початкового рівня підготовки. Це зумовлює необхідність розробки спеціалізованих систем, які поєднують простоту використання, наочність і функціональну повноту.

У межах даної роботи розглянуто питання створення програмної системи тестування моделей машинного навчання, орієнтованої на використання у навчальному процесі. Розроблена система забезпечує можливість роботи з наборами даних, їх попередньої обробки, побудови моделей, їх валідації та аналізу результатів у візуальній формі. Особлива увага приділяється інтерактивності, модульності архітектури та доступності інтерфейсу користувача.

Актуальність роботи

Актуальність теми дослідження зумовлена необхідністю підвищення якості підготовки фахівців у галузі інформаційних технологій в умовах цифровізації освіти. Машинне навчання стає ключовим інструментом аналізу даних, тому формування практичних навичок роботи з відповідними

моделями є критично важливим. Однак існуючі програмні засоби або орієнтовані на професійних користувачів, або не забезпечують достатнього рівня інтерактивності та наочності для освітнього процесу.

Крім того, сучасні освітні підходи передбачають використання інтерактивних і візуалізаційних технологій, які сприяють кращому розумінню складних математичних і алгоритмічних концепцій. Відсутність спеціалізованих навчальних систем для тестування моделей машинного навчання обмежує можливості ефективного засвоєння матеріалу.

Таким чином, розробка програмної системи, що поєднує інструменти обробки даних, побудови моделей і їх тестування в інтерактивному середовищі, є актуальним науково-прикладним завданням.

Метою роботи є розробка програмної системи тестування моделей машинного навчання з інтерактивним візуальним середовищем, орієнтованої на використання в освітньому процесі.

Об’єктом дослідження є процес тестування та аналізу моделей машинного навчання в освітньому середовищі.

Предметом дослідження є методи, алгоритми та програмні засоби реалізації інтерактивної системи тестування моделей машинного навчання.

Завдання дослідження

- провести аналіз предметної області та існуючих програмних рішень;
- обґрунтувати вибір математичних методів і алгоритмів;
- визначити функціональні та нефункціональні вимоги до системи;
- розробити архітектуру програмної системи;
- реалізувати модулі обробки даних, побудови та тестування моделей;
- забезпечити інтерактивну візуалізацію результатів;
- провести аналіз ефективності розробленої системи.

Методи дослідження

У роботі використано методи системного аналізу, методи математичного моделювання, зокрема метод найменших квадратів, методи

об’єктно-орієнтованого проєктування, UML-моделювання, а також методи програмної інженерії та аналізу даних.

Наукова новизна

Наукова новизна роботи полягає у розробці інтегрованого підходу до створення навчального середовища тестування моделей машинного навчання, що поєднує візуалізацію, інтерактивність і модульну архітектуру. Запропоновано підхід до організації процесу валідації та аналізу моделей, адаптований до освітніх потреб користувачів. Удосконалено методи інтерактивної роботи з даними шляхом поєднання інструментів препроцесингу, моделювання та оцінювання в єдиній системі.

Практичне застосування

Розроблена система може бути використана у навчальному процесі закладів вищої освіти для викладання дисциплін, пов’язаних з машинним навчанням і аналізом даних. Вона сприяє підвищенню якості підготовки студентів за рахунок практичної орієнтації навчання. Також система може бути застосована для самостійного навчання та підвищення кваліфікації фахівців у сфері ІТ.

Бакалаврська робота містить 81 сторінку, 21 рисунок, 4 розділи список використаних джерел із 34 найменуваннями.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ВИКОРИСТАННЯ МАШИННОГО НАВЧАННЯ В ОСВІТІ

1.1. Аналіз актуальності інтерактивних методів навчання машинного навчання в умовах цифровізації освіти

Зі стрімким розвитком та впровадженням високонавантажених інтелектуальних систем у стратегічно важливі сфери — фінанси, біомедичну інженерію, соціальні комунікації та освітні технології — машинне навчання (ML) трансформувалося у фундаментальну складову сучасного обчислювального середовища. В умовах глобальної орієнтації на парадигму штучного інтелекту (AI-first), забезпечення ефективної та інклюзивної підготовки фахівців у галузі ML стає критичним завданням для вищої школи.

Проте аналіз існуючих навчальних програм виявляє суттєвий когнітивний дисонанс між теоретичною базою та прикладними аспектами дисципліни:

- теоретична домінанта: більшість вступних курсів базуються на глибокому вивченні математичного апарату (лінійної алгебри, математичної статистики та теорії оптимізації). Хоча фундаментальна підготовка є необхідною, надмірна концентрація на абстрактних моделях часто створює високий поріг входження для студентів некогнітивних спеціальностей.

- бар’єр програмної реалізації: традиційний підхід передбачає вивчення ML через код-насичені середовища. Для початківців необхідність одночасного засвоєння синтаксису програмування та складних алгоритмічних структур призводить до розпорошення уваги.

- відсутність наочності конвеєра даних (ML Pipeline): етапи попередньої обробки даних, інженерії ознак та гіперпараметричної оптимізації сприймаються як ізольовані процеси, що ускладнює розуміння їх кумулятивного впливу на метрики ефективності моделі.

З огляду на це, виникає потреба у розробці та впровадженні інструментів інтерактивної візуалізації. Такі інструменти дозволяють реалізувати концепцію "навчання через експеримент", детермінуючи причинно-наслідкові зв'язки між вхідними даними та вихідними результатами моделі без необхідності написання коду на початкових етапах.

Переваги візуально-орієнтованого підходу та впровадження low-code або no-code візуальних інтерфейсів у навчальний процес дозволяє:

- знизити когнітивне навантаження, фокусуючи увагу студента на логіці функціонування алгоритму, а не на налагодженні програмного коду.

- забезпечити миттєвий зворотний зв'язок через динамічну зміну графіків та метрик (Accuracy, Precision/Recall) у реальному часі.

- демократизувати доступ до знань, залучаючи до розробки інтелектуальних систем фахівців доменних галузей (лікарів, економістів, педагогів), які мають експертизу в даних, але не володіють навичками програмування.

Таким чином, розробка методів візуального навчання є необхідним кроком для подолання розриву між теоретичними знаннями та практичними навичками побудови ефективних систем машинного навчання.

1.2. Концептуальні засади та постановка завдання розробки візуального навчального середовища машинного навчання

1.2.1. Обґрунтування розробки та передумови виникнення проєкту

Аналіз наявних освітніх ресурсів у галузі машинного навчання демонструє виражену дихотомію між фундаментальною теоретичною підготовкою та практичною імплементацією алгоритмів. Більшість академічних курсів забезпечують ґрунтовний математичний базис, проте приділяють недостатньо уваги прикладним аспектам безпосереднього

маніпулювання даними без залучення складного програмного інструментарію.

Традиційні етапи підготовки даних, такі як інженерія ознак та очищення наборів даних, зазвичай потребують значних часових витрат на написання коду. Це створює бар'єр, при якому другорядні технічні аспекти програмування нівелюють можливість спостереження за впливом окремих параметрів на продуктивність моделей у реальному часі. Отже, виникає об’єктивна потреба у розробці інструментарію, що дозволив би заповнити цей методологічний розрив шляхом надання візуального та інтерактивного інтерфейсу для дослідження ключових технік ML.

1.2.2. Постановка проблеми

Сучасна парадигма навчання ML значною мірою детермінована рівнем володіння мовами програмування (зокрема Python або R). Для початківців, які не мають достатнього досвіду в розробці ПЗ, такі завдання, як попередня обробка даних, трансформація ознак або оптимізація гіперпараметрів, стають «технологічним тупиком».

Існуючі освітні платформи часто апріорі припускають наявність у користувача навичок кодування. Такий підхід має суттєві недоліки:

- зниження мотивації, бо початківці витрачають більше часу на налагодження синтаксису, ніж на розуміння логіки алгоритмів.

- обмеженість експериментування, оскільки висока ціна помилки в коді стримує підхід до зміни параметрів моделі.

- когнітивне перевантаження, бо одночасне засвоєння математичної логіки та програмної реалізації знижує ефективність навчання.

Даний проєкт спрямований на створення альтернативного no-code середовища, яке дозволяє інтерактивно досліджувати концепції машинного навчання, абстрагуючись від синтаксичних складнощів реалізації.

Метою роботи є проєктування та розробка інтуїтивно зрозумілого програмного додатка для візуалізації повного циклу розробки та тестування ML-моделей. Програмний комплекс має функціонувати як інтерактивний навчальний засіб, що забезпечує проведення експериментів з мінімальними витратами на попередню підготовку середовища.

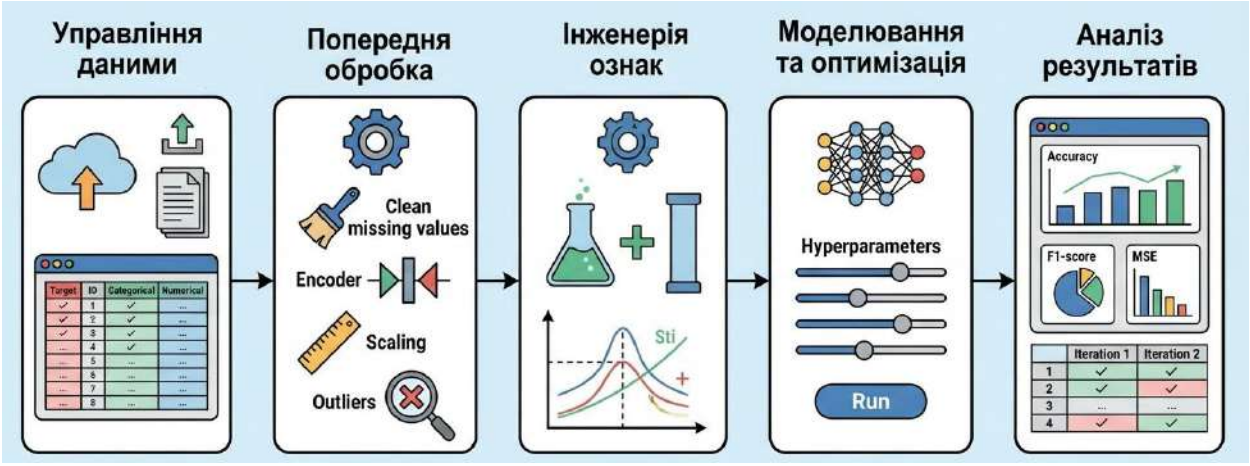

Рисунок 1.1 – Функціональні завдання розробки системи

Для досягнення поставленої мети визначено наступні функціональні завдання (рис. 1.1):

1. Управління даними. Реалізація можливості завантаження користувацьких датасетів та вибору еталонних зразків; забезпечення інтерфейсу для маркування атрибутів (визначення цільової змінної, ідентифікаторів, категоріальних та числових ознак).

2. Попередня обробка. Інтеграція інструментів для автоматизації обробки пропущених значень, кодування категоріальних ознак, масштабування (Scaling) та ідентифікації аномалій (Outlier detection).

3. Інженерія ознак. Забезпечення можливості відбору та генерації ознак на основі методів статистичного аналізу безпосередньо через графічний інтерфейс.

4. Моделювання та оптимізація. Симуляція процесів навчання моделей з можливістю динамічного налаштування гіперпараметрів.

5. Аналіз результатів: Візуалізація метрик ефективності (Accuracy, F1-score, MSE тощо) та порівняльний аналіз результатів різних ітерацій навчання.

6. Ергономіка інтерфейсу. Створення доступного UI/UX-дизайну, адаптованого для користувачів з нульовим або мінімальним рівнем підготовки в галузі програмування.

Використання зазначеного інструментарію дозволить змістити фокус навчання з синтаксичних конструкцій мов програмування на концептуальне розуміння архітектури систем штучного інтелекту.

1.3. Функціональні можливості та технічні характеристики розробленого програмного середовища

Розроблене програмне рішення спроєктоване як інтерактивна екосистема, орієнтована на забезпечення повного циклу операцій з даними в межах освітнього процесу. Пріоритетним завданням системи є надання користувачеві інструментарію для візуального маніпулювання параметрами моделей машинного навчання та оперативного спостереження за результатами змін.

1.3.1. Функціональний склад системи

Архітектура програмного комплексу охоплює наступний перелік функціональних модулів:

- Модуль управління даними - забезпечує механізми імпорту, структурування та редагування наборі даних.

- Модуль попередньої обробки містить базовий інструментарій для нормалізації, очищення та підготовки вибірки до аналізу.

- Блок інженерії ознак дозволяє реалізовувати первинну трансформацію змінних для підвищення предиктивної здатності моделей.

- Ітераційне моделювання надає інтерфейс для симуляції процесів навчання з можливістю динамічного корегування гіперпараметрів.

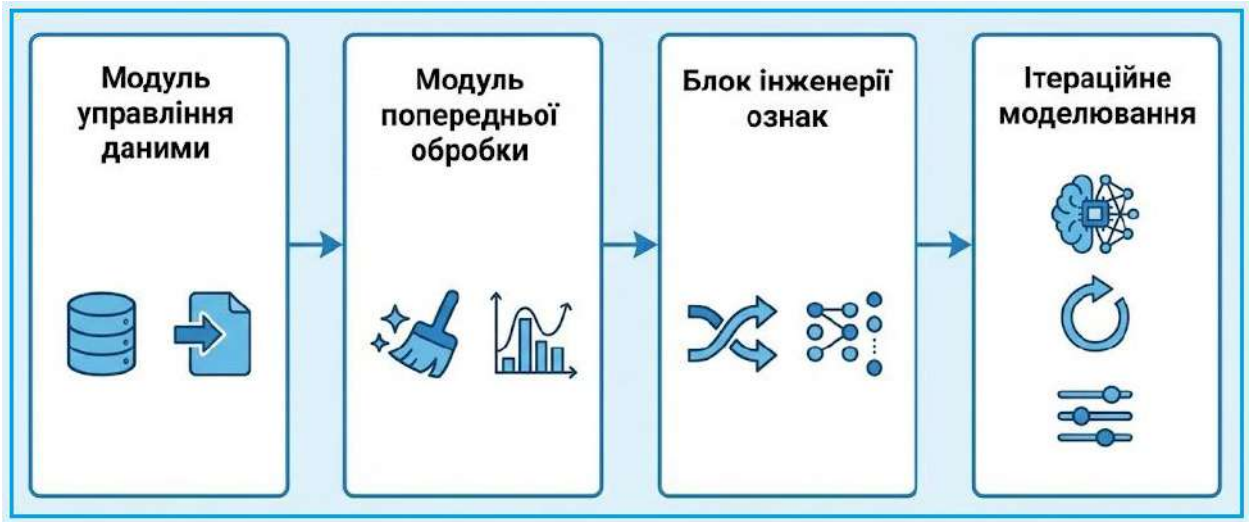

Рисунок 1.2 – Архітектура програмної системи

Ключовою особливістю системи є орієнтація на візуальну взаємодію. Це дозволяє досліднику проводити серію експериментів з різними конфігураціями середовища, миттєво оцінюючи їхній вплив на метрики продуктивності та стабільність результатів.

1.3.2. Технічні обмеження та специфікації

Поточна версія програмного забезпечення є концептуальним прототипом, що обумовлює наявність ряду детермінованих обмежень:

- на даному етапі реалізовано підтримку виключно моделей лінійної регресії. Такий вибір зумовлений високим рівнем інтерпретованості результатів, що є критичним для початкового етапу навчання.

- кількість доступних методів препроцесингу та інженерії ознак свідомо обмежена. Дана стратегія спрямована на запобігання «когнітивному перевантаженню» користувача та фокусування уваги на фундаментальних принципах ML-конвеєра.

- додаток розроблено з використанням фреймворку Streamlit [1]. Важливою характеристикою обраної технології є використання сесійної моделі зберігання даних. Це передбачає, що стан середовища скидається до початкових значень при оновленні сторінки браузера або завершенні сеансу.

Систему оптимізовано для роботи з датасетами малого та середнього обсягів (ліміт до 200 МБ), що відповідає типовим завданням навчального рівня. Для стабільного функціонування необхідна наявність постійного інтернет-з'єднання та використання сучасного десктопного веббраузера з підтримкою актуальних стандартів рендерингу.

Незважаючи на статус прототипу, запропоноване рішення дозволяє вирішити головну проблему — відсутність наочного зв'язку між маніпуляціями з даними та фінальною якістю моделі. Використання low-code підходу мінімізує поріг входження, дозволяючи зосередитися на логіці статистичного висновку, а не на синтаксичних аспектах мов програмування.

1.4. Порівняльний аналіз існуючих рішень та позиціонування розроблюваного програмного забезпечення

На сьогодні в галузі інтелектуального аналізу даних існує низка розвинених інструментальних засобів, призначених для підтримки візуальних робочих процесів машинного навчання. Серед найбільш репрезентативних систем варто виділити Orange Data Mining [3], MLflow [4] та KNIME [5]. Проведений аналіз дозволяє класифікувати ці платформи як багатофункціональні середовища, орієнтовані на фахівців із базовим або просунутим досвідом у сфері Data Science.

1.4.1. Програмний комплекс Orange Data Mining

Orange Data Mining є відкритим багатофункціональним програмним комплексом, призначеним для інтелектуального аналізу даних, машинного

навчання та візуалізації. Платформа реалізована на мові програмування Python із використанням графічної бібліотеки Qt та інтегрованих модулів наукових обчислень, таких як Scikit-learn, NumPy та SciPy.

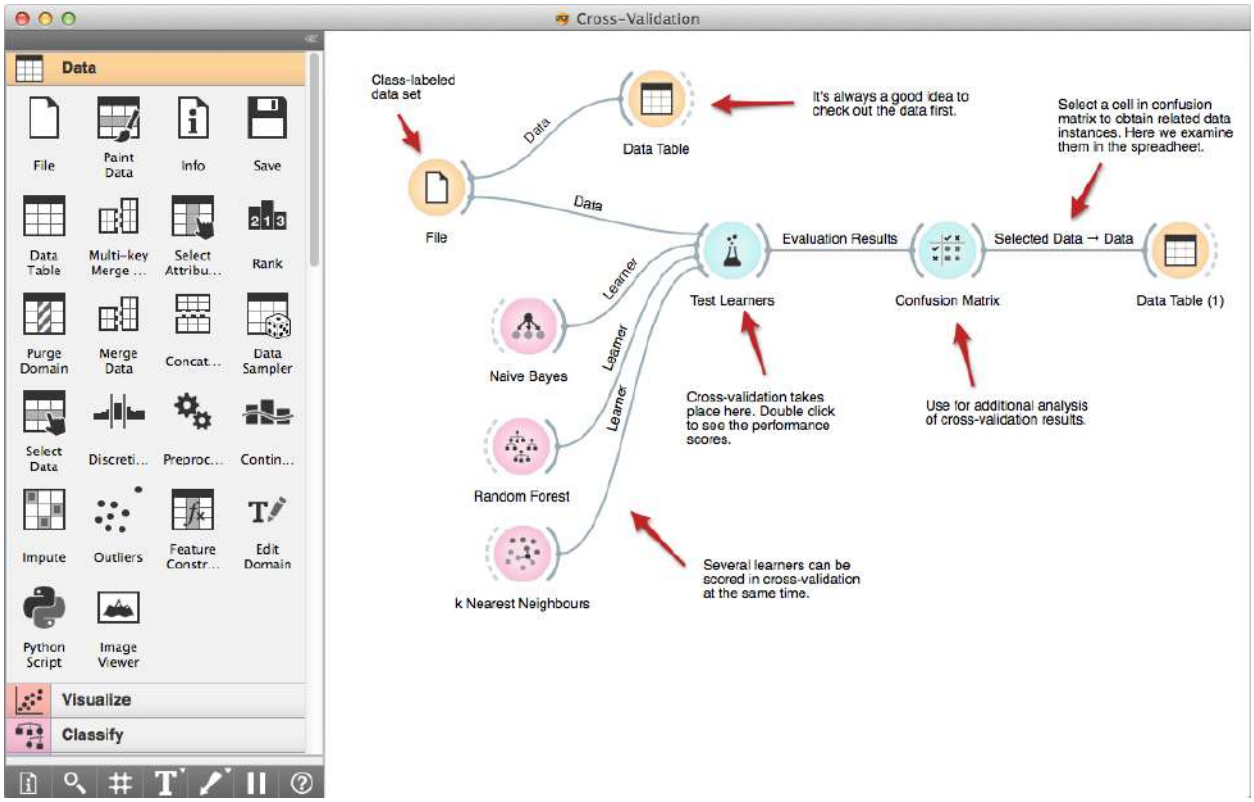

Рисунок 1.3 – Інтерфейс Orange Data Mining

Фундаментальною особливістю Orange є реалізація концепції візуального програмування. На відміну від традиційного написання коду, процес розробки аналітичного конвеєра здійснюється шляхом маніпулювання графічними об'єктами — віджетами.

Кожен віджет є інкапсульованою функціональною одиницею, що відповідає за конкретну операцію: зчитування даних, фільтрацію атрибутів, навчання моделі або побудову діаграми. Користувач формує робочий процес (workflow), з’єднуючи віджети лініями зв’язку, через які передаються структуровані дані. Така архітектура забезпечує високу наочність і дозволяє

досліднику оперативно змінювати параметри експерименту без необхідності переписування програмних інструкцій.

Функціональний спектр Orange охоплює повний життєвий цикл аналізу даних:

- Попередня обробка даних. Програмний засіб надає інструменти для імпорту даних у різних форматах, обробки пропущених значень, нормалізації, дискретизації та відбору ознак.

- Машинне навчання та моделювання. Бібліотека Orange містить широкий набір алгоритмів, що включає лінійні та логістичні регресії, випадкові ліси (Random Forest), метод опорних векторів (SVM), нейронні мережі та методи кластеризації (k-means, ієрархічна кластеризація).

- Аналітична візуалізація. Платформа вирізняється потужними інструментами інтерактивної графіки — від класичних гістограм та розсіювань (Scatter Plots) до складних методів багатовимірної візуалізації, таких як t-SNE та PCA (метод головних компонентів).

Orange Data Mining часто розглядається як інструмент для швидкого прототипування та апробації дослідницьких гіпотез. Завдяки відкритому вихідному коду, система може бути розширена шляхом інтеграції власних скриптів на мові Python, що робить її гнучкою для спеціалізованих наукових завдань.

Для освітнього процесу Orange має особливу цінність як «міст» між теоретичною статистикою та практичною розробкою. Він дозволяє студентам візуалізувати внутрішні процеси алгоритмів (наприклад, перегляд структури дерев рішень у реальному часі), що сприяє глибшому засвоєнню концепцій математичного моделювання та аналізу даних.

Таким чином, Orange Data Mining представляє собою складне програмне середовище, яке поєднує професійні аналітичні можливості з інтуїтивно зрозумілим візуальним інтерфейсом, забезпечуючи високу ефективність дослідницької роботи в умовах сучасної цифровізації науки.

1.4.2. Опис платформи машинного навчання MLflow

MLflow представляє собою відкриту платформу з відкритим вихідним кодом, спроєктовану для управління повним життєвим циклом машинного навчання (Machine Learning Lifecycle Management). Основна цінність системи полягає у вирішенні проблем відтворюваності експериментів, стандартизації пакування моделей та автоматизації процесів їх розгортання у продуктивних середовищах (парадигма MLOps).

Рисунок 1.4 – Інтерфейс платформи машинного навчання MLflow

Архітектура MLflow детермінована чотирма модульними компонентами, які інтегруються в єдину аналітичну екосистему:

1. MLflow Tracking. Централізований інтерфейс (API та GUI) для логування параметрів, версій вихідного коду, метрик продуктивності та вихідних артефактів. Це дозволяє досліднику проводити компаративний аналіз сотень ітерацій навчання, фіксуючи найменші зміни у функціях втрат або точності алгоритмів.

2. MLflow Projects. Специфікація для пакування коду в самодостатні контейнери (наприклад, за допомогою Conda або Docker). Це забезпечує

сувору відтворюваність (reproducibility) результатів, гарантуючи, що код буде виконуватися ідентично на різних обчислювальних вузлах незалежно від локальної конфігурації середовища.

3. MLflow Models. Стандартизований формат для представлення моделей машинного навчання, що підтримує концепцію «універсального виводу» (Inference). Завдяки цьому модель, навчена за допомогою будь-якої бібліотеки (наприклад, PyTorch, Scikit-learn або XGBoost), може бути легко інтегрована в сторонні сервіси або хмарні платформи.

4. MLflow Model Registry: Система управління версіями моделей, що підтримує повний життєвий цикл — від розробки (Staging) до промислового використання (Production) та архівування.

MLflow виступає як потужний засіб систематизації експериментальних даних. На відміну від навчальних середовищ, де фокус спрямований на візуалізацію окремих етапів, MLflow забезпечує промислову надійність та масштабованість.

Платформа дозволяє зберігати повний «цифровий слід» кожної моделі, що є критично важливим для верифікації наукових гіпотез. Модельний реєстр дозволяє групам дослідників працювати над спільними проєктами, уникаючи конфліктів версій та втрати проміжних результатів. MLflow позиціонується як професійне середовище для спеціалістів, які володіють навичками програмування та потребують інструментів для автоматизації складних, багатокомпонентних процесів машинного навчання. В освітньому контексті вивчення MLflow є логічним наступним кроком після освоєння концептуальних основ у візуальних навчальних середовищах.

1.4.3. Відкрита програмна платформа для ІАД KNIME

KNIME (Konstanz Information Miner) — це відкрита програмна платформа для інтелектуального аналізу даних, яка базується на модульній архітектурі та парадигмі візуального програмування. Розроблена на базі

середовища Eclipse, система надає потужний інструментарій для створення аналітичних конвеєрів, охоплюючи всі етапи роботи з даними: від первинного видобутку (ETL) до розгортання складних моделей машинного навчання.

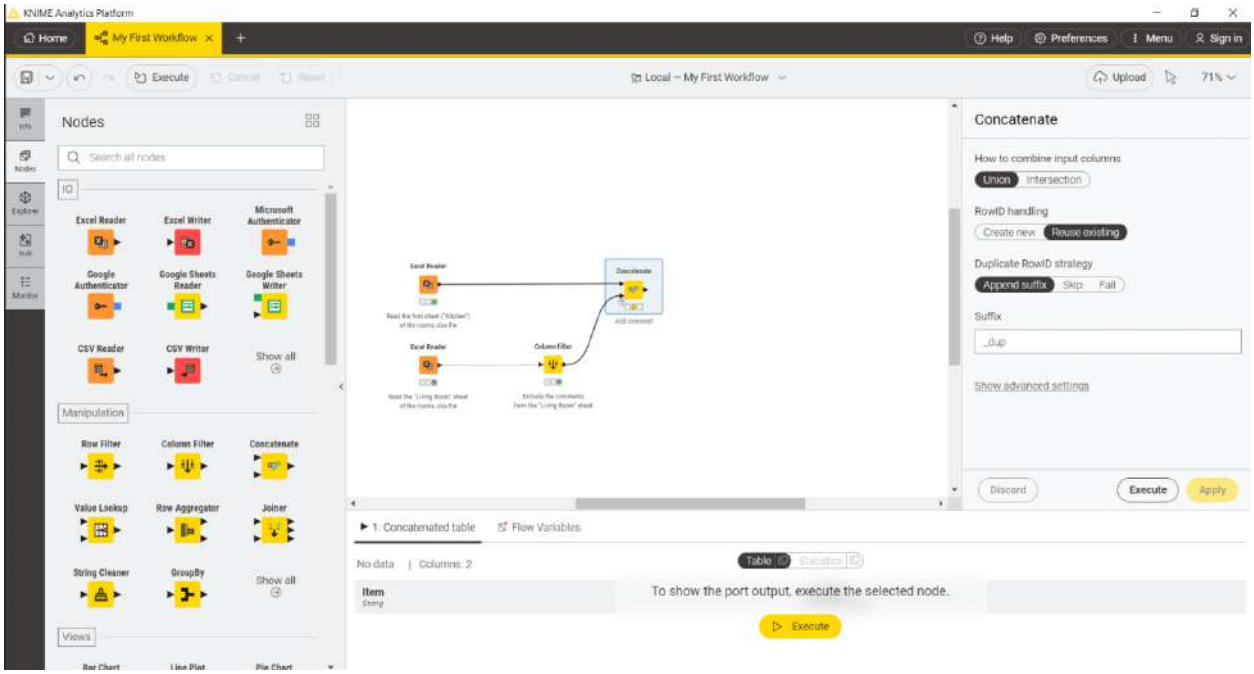

Рисунок 1.5 – Інтерфейс платформи KNIME

Фундаментальною одиницею логіки в KNIME є вузол. Кожен вузол виконує строго детерміновану атомарну операцію (наприклад, зчитування файлу, групування даних або навчання нейронної мережі). Користувач формує робочий потік (workflow), поєднуючи вузли через вхідні та вихідні порти. Кожен порт має строго визначений тип даних (таблиця, модель, база даних), що забезпечує структурну цілісність усього конвеєра.

Вузли мають світлофорну систему індикації стану: «червоний» (не налаштовано), «жовтий» (готово до виконання) та «зелений» (успішно виконано). Це дозволяє досліднику поетапно налагоджувати складні аналітичні процеси, зберігаючи проміжні результати обчислень.

KNIME вирізняється своєю екстенсивністю та здатністю до інтеграції з різнорідними технологічними стеками:

- Платформа підтримує роботу з великими даними через інтеграцію з Apache Spark та Hadoop, а також дозволяє безпосередньо взаємодіяти з реляційними та NoSQL БД за допомогою спеціалізованих конекторів.

- KNIME не обмежує користувача лише візуальними інструментами. Завдяки вузлам-скриптам (Python Script, R Snippet), дослідники можуть впроваджувати власний програмний код безпосередньо в середину візуального конвеєра, поєднуючи гнучкість програмування з наочністю workflow-підходу.

- Система містить розширені модулі для автоматичного підбору гіперпараметрів та оцінки ефективності алгоритмів, що значно прискорює процес пошуку оптимальних предиктивних моделей.

KNIME цінується за забезпечення високого рівня відтворюваності досліджень. Робочі потоки можуть бути експортовані та передані іншим дослідникам, гарантуючи ідентичність результатів обробки даних незалежно від операційної системи.

Окрім наукових цілей, платформа широко використовується в корпоративному секторі для бізнес-аналітики, біоінформатики та фінансового моделювання. Можливість автоматичної генерації звітів та візуалізації даних робить KNIME універсальним інструментом для трансляції складних математичних результатів у зрозумілі бізнес-рішення. Таким чином, KNIME представляє собою аналітичну екосистему, яка поєднує низький поріг входження для візуального моделювання з необмеженими можливостями для професійного аналізу даних та розробки систем ШІ.

1.4.4. Аналіз наявних аналогів та їхні обмеження для освітнього процесу

Незважаючи на потужний аналітичний потенціал, згадані платформи мають низку особливостей, що створюють суттєві бар’єри для студентів-початківців:

- більшість професійних інструментів потребують локального встановлення, конфігурації специфічних залежностей та наявності відповідних обчислювальних потужностей на боці користувача.

- такі системи, як KNIME або Orange, базуються на парадигмі вузлового програмування (node-based programming). Хоча це полегшує розробку складних конвеєрів, для абсолютного новачка інтерфейс із сотнями функціональних блоків може бути дезорієнтуючим.

- платформи на кшталт MLflow фокусуються на управлінні життєвим циклом моделей (MLOps), що вимагає розуміння процесів версіонування, реєстрів моделей та відстеження експериментів — концепцій, які є заскладними на етапі ознайомлення з основами лінійної регресії.

1.4.5. Концептуальні переваги розробленого середовища

Розроблений програмний додаток не претендує на конкуренцію з повномасштабними аналітичними системами, а натомість пропонує спеціалізований полегшений підхід, адаптований до потреб початкової освіти.

Ключові відмінності розробленого рішення:

- веб-орієнтована архітектура, оскільки доступ до інструментарію здійснюється через браузер без необхідності попереднього встановлення ПЗ, що забезпечує кросплатформенність та миттєвий початок роботи.

- на відміну від складних графів у аналогах, запропоноване середовище веде користувача через послідовні етапи ML-конвеєра, що відповідає дидактичному принципу «від простого до складного».

- система забезпечує максимально прозорий «no-code» досвід, де фокус зміщено з налагодження інструмента на розуміння поведінки алгоритму.

Таким чином, позиціонування розробленого продукту визначається як «точка входу» у дисципліну. Він заповнює нішу між теоретичними підручниками та складними професійними платформами, надаючи

доступний та інтерактивний стартовий майданчик для формування базових компетенцій у галузі машинного навчання.

1.5. Роль інтерактивної візуалізації моделей в інклюзивності освіти

Сучасна педагогічна практика в галузі інтелектуального аналізу даних дедалі частіше спирається на візуальні методи інтерпретації складних алгоритмічних структур. Візуальні інструменти визнаються критично важливими для полегшення когнітивного навантаження під час дослідження та розуміння моделей машинного навчання, особливо в академічному середовищі [2].

Одним із пріоритетних векторів проєктування розробленого програмного комплексу стала освітня доступність. Впровадження підходу з низьким порогом входження дозволяє студентам та дослідникам-початківцям зосередитися на семантиці процесів ML, оминаючи етап технічної реалізації коду.

Ключові переваги реалізованого інтерфейсу:

- Інтуїтивність взаємодії, оскільки користувач взаємодіє безпосередньо з об'єктами даних через графічні елементи керування, що робить процес навчання більш детермінованим.

- Зворотний зв'язок у реальному часі і миттєва візуалізація змін у продуктивності моделі після коригування методів препроцесингу дозволяє емпіричним шляхом встановити закономірності впливу вхідних параметрів на вихідний результат.

- Можливість швидкої ітерації різних конфігурацій сприяє глибшому розумінню теоретичних концепцій через безпосередню практичну дію.

Попри функціональну завершеність базових модулів, поточна реалізація програмного забезпечення розглядається як фундаментальний етап

створення комплексної інтелектуальної платформи. Для досягнення повної методичної відповідності вимогам сучасної ІТ-освіти визначено наступні напрями вдосконалення:

- Розширення інструментарію обробки даних, бо необхідно імплементувати ширший спектр методів попередньої обробки (наприклад, складніші техніки імпутації даних та розширені методи інженерії ознак, такі як PCA — метод головних компонентів).

- Подальша розробка передбачає додавання нелінійних моделей, алгоритмів класифікації (логістична регресія, дерева рішень) та методів ансамблювання.

- Додавання контекстних підказок, інтерактивних «гайдів» та покрокових інструкцій дозволить перетворити інструментальний додаток на повноцінну систему самостійного навчання.

- Впровадження розширених графіків аналізу залишків та матриць кореляції для глибшої діагностики моделей.

Розроблене візуальне середовище демонструє високу ефективність у подоланні технологічного бар'єра між теоретичними знаннями та практичними навичками. Створення альтернативи традиційному програмуванню на початкових етапах вивчення ML дозволяє сформувати у користувача стійку логічну базу, що є необхідною умовою для подальшого переходу до професійної розробки складних інтелектуальних систем.

1.6 Висновки по розділу

У першому розділі проведено системний аналіз предметної області використання машинного навчання в освіті, що дозволило встановити зростаючу актуальність інтерактивних підходів у навчальному процесі. Доведено, що в умовах цифровізації освіти традиційні методи викладання є недостатньо ефективними для формування практичних компетентностей у

сфері аналізу даних. Обґрунтовано доцільність розробки спеціалізованого візуального середовища, орієнтованого на навчальні потреби користувачів. Проведений порівняльний аналіз існуючих програмних рішень виявив їх функціональні переваги та водночас обмеження щодо застосування у навчальному процесі.

У результаті сформовано концептуальні засади та визначено ключові переваги розроблюваної системи, зокрема інтерактивність, доступність і орієнтацію на освітню інклюзивність.

РОЗДІЛ 2. АЛГОРИТМІЧНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ТЕСТУВАННЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

2.1. Математичний апарат та теоретичне обґрунтування вибору методу лінійної регресії

Вибір алгоритму лінійної регресії як базового методу в розробленому інтерактивному середовищі зумовлений його високою інтерпретованістю, математичною прозорістю та фундаментальною роллю у статистичному аналізі. Лінійна регресія дозволяє наочно продемонструвати базові принципи машинного навчання, такі як мінімізація функції втрат та оптимізація параметрів.

2.1.1. Формальна постановка моделі

Завдання багатофакторної лінійної регресії полягає у знаходженні лінійної залежності між вектором незалежних ознак $X = (x_1, x_2, \ldots, x_n)$ та цільовою змінною y. Математично модель описується рівнянням:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$

де:

β_0 — вільний член (intercept);

β_i — регресійні коефіцієнти, що визначають вагу кожної ознаки;

ε — випадкова похибка (шум), що враховує фактори, які не увійшли до моделі.

У матричному вигляді модель записується як:

$$\hat{Y} = X\beta$$

2.1.2. Метод найменших квадратів

Для знаходження оптимальних значень коефіцієнтів β у системі реалізовано класичний метод найменших квадратів (Ordinary Least Squares). Суть методу полягає у мінімізації суми квадратів залишків (різниці між фактичними та прогнозованими значеннями):

$$RSS(\beta) = \sum_{i=1}^{m}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{m}(y_i - (\beta_0 + \sum_{j=1}^{n}\beta_j x_{ij}))^2 \to \min$$

Аналітичне розв'язання для вектора коефіцієнтів β знаходиться через нормальне рівняння:

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

2.1.3. Обґрунтування вибору для навчального середовища

Використання саме цього математичного апарату в межах проєкту аргументовано наступними чинниками:

- Пряма інтерпретованість: коефіцієнт βі безпосередньо вказує, на скільки одиниць зміниться цільова змінна при зміні ознаки хі на одиницю. Це ідеальний інструмент для навчання студентів поняттю «важливості ознак».

- Діагностика допущень: модель дозволяє візуалізувати фундаментальні поняття статистики: гомоскедастичність, нормальність розподілу залишків та відсутність мультиколінеарності.

- Обчислювальна ефективність: отримання ваг моделі через аналітичне розв'язання або ітеративні методи (Gradient Descent) відбувається майже миттєво для наборів даних до 200 МБ, що забезпечує необхідну інтерактивність інтерфейсу.

- Статистична значущість: лінійна регресія надає можливість обчислення p-значень для кожного коефіцієнта, що дозволяє реалізувати в програмі функцію відбору ознак на основі їхньої статистичної значущості.

2.1.4. Метрики оцінки адекватності моделі

Для аналізу результатів у програмному комплексі використовуються наступні математичні метрики.

Коефіцієнт детермінації (R^2) показує частку дисперсії залежної змінної, яка пояснюється моделлю:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Середня абсолютна похибка (MAE) визначає середнє відхилення прогнозу від факту:

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i|$$

Середньоквадратична похибка (MSE) - чутлива до великих відхилень (викидів):

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

Впровадження даного математичного базису дозволяє користувачеві не просто отримати результат прогнозування, а й зрозуміти внутрішню логіку роботи алгоритму через зміну параметрів β та аналіз похибок.

2.2. Специфікація функціональних вимог до інтерактивного середовища тестування моделей

Розроблене програмне забезпечення базується на парадигмі no-code, що дозволяє реалізувати повний цикл типового робочого процесу машинного навчання через графічний інтерфейс. Функціональні вимоги до системи сформовані таким чином, щоб забезпечити дослідникові можливість маніпулювання даними, детермінованого застосування математичних перетворень та верифікації моделей без необхідності володіння навичками програмування.

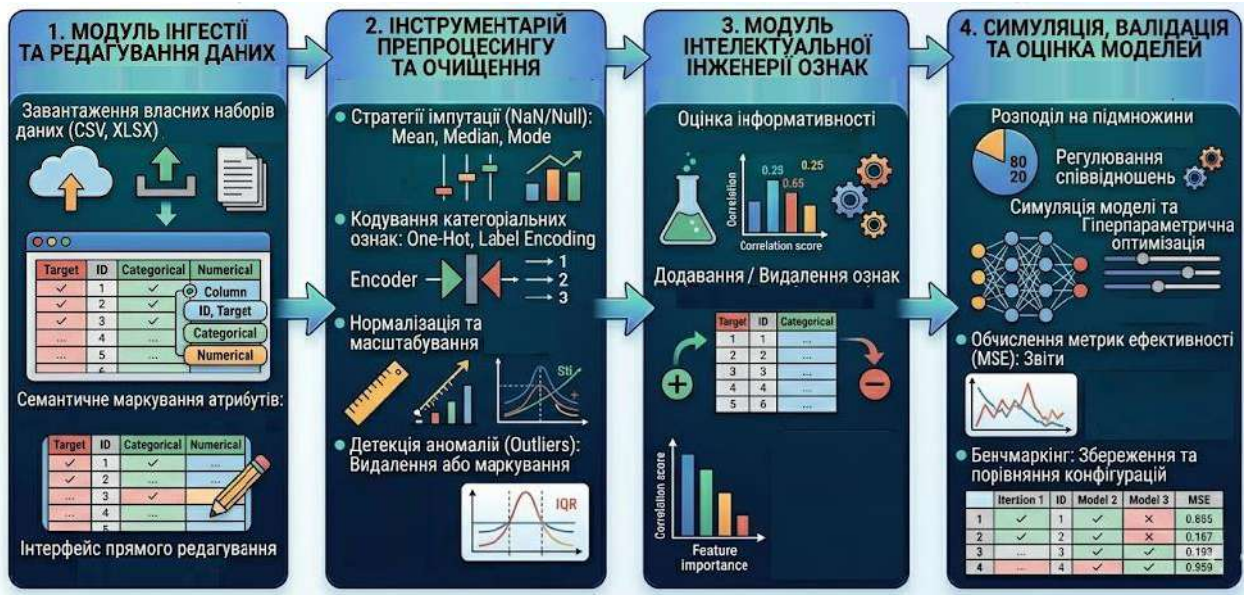

Рисунок 2.1 – Основні функціональні модулі системи

Нижче наведено детальну декомпозицію основних функціональних модулів системи.

2.2.1. Модуль підготовки та редагування наборів даних

Даний сегмент відповідає за первинну обробку вхідної інформації та її підготовку до структурного аналізу:

- Управління вхідними потоками. Реалізація механізмів імпорту користувацьких датасетів у поширених форматах (CSV, XLSX) або вибір еталонних вибірок із вбудованої бібліотеки для валідації алгоритмів.

- Семантичне маркування атрибутів. Можливість інтерактивного призначення ролей для кожного стовпця (ідентифікатор, цільова змінна, категоріальна або числова ознака).

- Інтерфейс прямого редагування. Впровадження динамічної табличної структури, що дозволяє проводити оперативну корекцію значень безпосередньо в середовищі додатка.

2.2.2. Інструментарій препроцесингу та очищення даних

Модуль забезпечує математичну підготовку даних для мінімізації похибок навчання:

- Стратегії імпутації (Data Imputation): Заміна відсутніх значень (NaN/Null) з використанням різних статистичних стратегій (середнє, медіана, мода або константне значення).

- Кодування категоріальних ознак: Перетворення текстових міток у числовий формат за допомогою методів One−Hot Encoding або Label Encoding.

- Нормалізація та масштабування: Приведення числових ознак до єдиного діапазону (Min−Max Scaling або Z−score Standardization) для забезпечення стабільної збіжності алгоритмів.

- Детекція аномалій (Outlier Detection): Автоматизоване виявлення викидів за допомогою статистичних методів (наприклад, Z−score або інтерквартильний розмах IQR) з можливістю їх вилучення або маркування.

2.2.3. Модуль інтелектуальної інженерії ознак

Функціонал спрямований на оптимізацію вхідного простору ознак:

- Оцінка інформативності: Розрахунок метрик взаємної інформації (Mutual Information) для кількісного оцінювання кореляції між вхідними параметрами та цільовою змінною.

- Трансформація ознак: Можливість додавання деривативних (виведених) ознак або деактивація малозначущих атрибутів на основі отриманих статистичних показників.

2.2.4. Симуляція, валідація та оцінка моделей

Заключний етап конвеєра, що відповідає за отримання предиктивних результатів:

- Валідаційна стратегія: автоматизований розподіл вибірки на тренувальну та тестову підмножини з можливістю регулювання пропорцій користувачем.

- Гіперпараметрична оптимізація: інтерфейс для прямого налаштування внутрішніх параметрів моделі (наприклад, коефіцієнтів регуляризації) перед запуском ітерації навчання.

- Обчислення метрик ефективності: генерація аналітичних звітів, що включають показники похибок, зокрема середньоквадратичну помилку (MSE).

- Бенчмаркінг: Механізм збереження результатів декількох запусків для проведення порівняльного аналізу та вибору найбільш ефективної конфігурації.

Синергія зазначених функцій створює цілісне навчальне середовище, що нівелює технічний бар'єр між теорією та практикою.

Завдяки модульній архітектурі, програмний комплекс володіє високим потенціалом масштабованості, що дозволяє в майбутньому інтегрувати складніші моделі глибинного навчання та додаткові інструменти аналітичної візуалізації.

2.3. Специфікація нефункціональних вимог, технічних обмежень та припущень системи

Ефективність функціонування розробленого програмного забезпечення визначається не лише його інструментальним складом, а й низкою якісних характеристик, що забезпечують стабільність, доступність та ергономічність навчального процесу.

2.3.1. Нефункціональні вимоги до програмного продукту

Нефункціональні вимоги зосереджені на забезпеченні високого рівня користувацького досвіду (UX), продуктивності та інклюзивності для цільової аудиторії, яка може не володіти глибокими технічними навичками.

- Ергономічність та інтуїтивність інтерфейсу. Архітектура графічного інтерфейсу спроєктована за принципом мінімізації когнітивного навантаження. Структура макета відповідає логічній послідовності етапів ML-конвеєра, що дозволяє користувачеві орієнтуватися в системі без необхідності попереднього вивчення інструкцій.

- Продуктивність та латентність. Усі алгоритми обробки даних та візуалізації оптимізовані для забезпечення швидкої реакції системи. Час відгуку інтерфейсу на дії користувача (зміна параметрів, застосування фільтрів) зведений до мінімуму для підтримки безперервності навчального процесу.

- Кросбраузерна сумісність. Програмний комплекс забезпечує коректне відображення та стабільну роботу в актуальних версіях провідних веббраузерів, включаючи Google Chrome, Mozilla Firefox, Microsoft Edge та Safari.

- Адаптивність пристроїв. Основним середовищем експлуатації визначено настільні комп'ютери, що зумовлено необхідністю візуалізації складних графічних об'єктів та табличних даних.

Розробка системи базувалася на певних пресупозиціях, що визначають контекст її використання:

- припускається, що кінцевий користувач є дослідником-початківцем з мінімальним досвідом у програмуванні або статистичному моделюванні.

- робота з додатком вимагає наявності сучасного десктопного браузера та стабільного інтернет-з'єднання для завантаження бібліотек та динамічного рендерингу компонентів.

2.3.2. Технічні та архітектурні обмеження

На відміну від промислових ML-платформ, орієнтованих на роботу з великими даними та складними ансамблями моделей [6], даний проєкт має детерміновані обмеження, обумовлені його освітньою спрямованістю та обраним технологічним стеком:

- поточна версія підтримує виключно моделі лінійної регресії. Це обмеження є свідомим вибором для підтвердження концепції (Proof of Concept) та фокусування на базових принципах інтерпретації результатів.

- використання фреймворку Streamlit обумовлює безстатусну (stateless) модель роботи. Стан сесії скидається при оновленні сторінки або розриві з'єднання, що обмежує можливість тривалого збереження проміжних етапів роботи без експорту результатів.

- максимальний розмір вхідного набору даних встановлено на рівні 200 МБ. Даний ліміт відповідає типовим навчальним датасетам і є оптимальним для обробки в оперативній пам'яті, проте може бути недостатнім для вирішення складних наукових завдань.

- стислі терміни реалізації обмежили загальний функціонал, проте архітектура системи побудована за модульним принципом. Це забезпечує високу масштабованість та можливість інтеграції нових методів препроцесингу та складніших моделей (наприклад, нейронних мереж) у майбутніх ітераціях проєкту.

Визначені нефункціональні вимоги та обмеження формують чіткі межі експлуатації системи, гарантуючи її стабільність у межах освітнього процесу. Незважаючи на наявні технічні ліміти, платформа забезпечує необхідну гнучкість для трансформації у повнофункціональне середовище візуального моделювання.

2.4. Організація та архітектура управління даними

Ефективність функціонування інтерактивного навчального середовища безпосередньо залежить від моделі обробки та зберігання інформації. Враховуючи динамічний характер взаємодії користувача з платформою, архітектура даних базується на принципах тимчасової детермінованості та оперативного доступу.

2.4.1. Логічна структура та обробка даних у пам'яті

Для забезпечення високої швидкодії при роботі з наборами даних обсягом до 200 МБ, система використовує модель In-Memory Data Management. Основною структурною одиницею збереження даних на рівні логіки додатка є об'єктно-орієнтовані структури, що дозволяють здійснювати маніпуляції з ознаками в режимі реального часу.

Рисунок 2.2 - Логічна схема даних

Логічна схема даних включає:

- Метадані набору даних: структура, що містить інформацію про типи змінних (категоріальні, числові), виявлені пропуски та статистичні характеристики (середнє значення, медіана, стандартне відхилення).

- Конфігурація стану моделі: словник параметрів, де зберігаються поточні значення гіперпараметрів лінійної регресії та обрані методи препроцесингу.

- Історія ітерацій: масив результатів попередніх запусків, що дозволяє проводити порівняльний аналіз метрик ефективності (R2, MAE, MSE).

2.4.2. Керування станом сеансу

Оскільки програмний комплекс реалізовано на базі фреймворку Streamlit, традиційна архітектура з персистентною базою даних замінена на модель керування станом сесії.

Це рішення дозволяє:

- Ізоляція даних: кожен користувач працює у власному ізольованому контейнері даних, що забезпечує конфіденційність завантажених датасетів.

- Мінімізація затримок: відсутність необхідності постійного звернення до зовнішніх SQL-серверів значно прискорює процес перерахунку моделі при зміні ознак.

- Тимчасове зберігання: дані існують протягом життєвого циклу поточної вкладки браузера, що нівелює потребу в адмініструванні складних систем зберігання для навчальних цілей.

2.4.3. Схема потоків даних

Процес проходження даних через систему можна представити у вигляді наступної послідовності:

- Input Layer: Завантаження файлу (CSV/XLSX) через буфер пам’яті.

- Processing Layer: Серіалізація даних та їх валідація на відповідність типам. У разі потреби застосовуються алгоритми нормалізації, результати яких оновлюють поточний стан у пам'яті.

- Storage Layer: Тимчасове кешування результатів інженерії ознак для запобігання повторним обчисленням при зміні візуальних компонентів інтерфейсу.

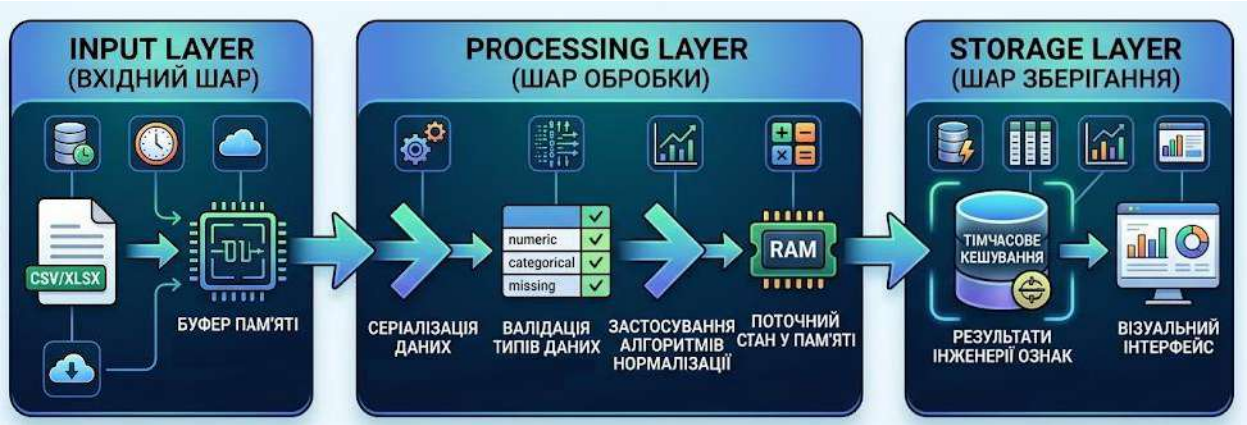

Рисунок 2.3 - Схема потоків даних

Така архітектура дозволяє збалансувати продуктивність системи та зручність користувача, фокусуючи ресурси на оперативності відгуку інтерфейсу, що є критично важливим для освітнього процесу.

2.5. Процес валідації даних та інструментальна база обробки даних

2.5.1. Механізми валідації та автоматичного визначення структурних параметрів даних

Для забезпечення безперебійної взаємодії користувача з платформою реалізовано алгоритм адаптивного завантаження даних. Цей етап є критичним, оскільки некоректне зчитування структури файлу на початковому етапі унеможливлює подальше моделювання.

Процес валідації охоплює три рівні перевірки:

1. Детекція кодування та формату

Оскільки вхідні набори даних можуть бути сформовані у різних операційних середовищах, система використовує механізми автоматичного визначення кодування (наприклад, UTF-8, Windows-1251). Це дозволяє уникнути артефактів при відображенні кириличних символів у назвах ознак. Паралельно здійснюється ідентифікація роздільників (кому, крапка з комою, табуляція) для коректного парсингу табличної структури.

2. Інференс типів даних

Після зчитування файлу система проводить автоматичний аналіз кожного стовпця:

- Числові ознаки (Numerical): Виявлення цілих чисел та чисел із рухомою комою. На цьому етапі перевіряється наявність нечислових вкраплень у колонках, що за логікою мають бути квантитативними.

- Категоріальні ознаки (Categorical/Object): Визначення текстових міток та оцінка їхньої кардинальності (кількості унікальних значень).

- Часові мітки (Datetime): Спроба автоматичного розпізнавання форматів дати для їх подальшої коректної обробки.

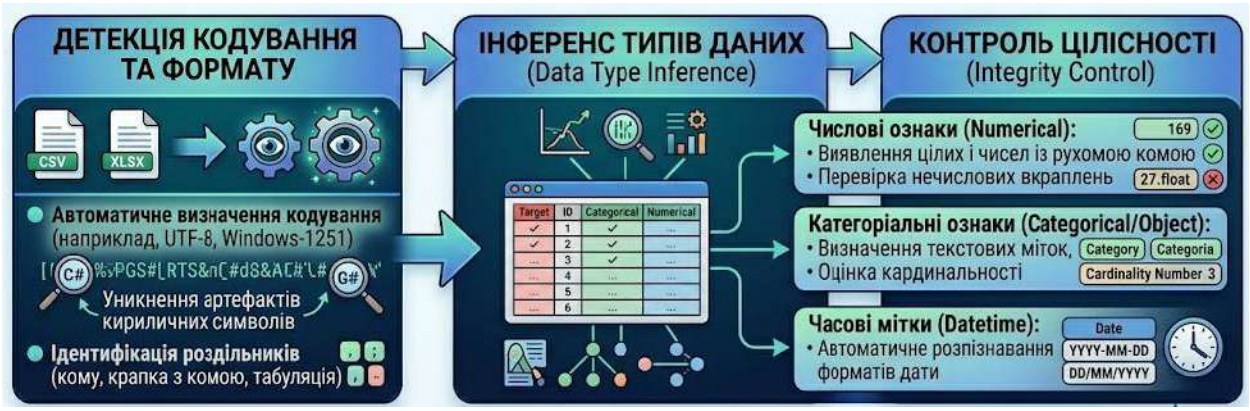

Рисунок 2.4 – Процес валідації даних

3. Контроль цілісності

Алгоритм проводить первинний аудит наявності порожніх значень (NaN/Null) та дублікатів записів. У разі виявлення критичних помилок (наприклад, невідповідність кількості стовпців у рядках) система генерує інформативне повідомлення для користувача, запобігаючи аварійному завершенню роботи додатка.

2.5.2. Інтерфейс маркування атрибутів

Після успішної валідації користувачеві надається можливість ручного коригування метаданих через графічний інтерфейс. Це включає:

- Визначення цільової ознаки — показника, який модель буде прогнозувати.

- Виключення ідентифікаторів — унікальних маркерів, що не мають статистичної значущості для навчання.

- Примусова зміна типу даних, якщо автоматичне визначення не відповідає дослідницькій меті (наприклад, інтерпретація числових кодів як категорій).

Такий підхід до управління даними гарантує, що навчальний процес фокусуватиметься на змістовній частині ML-конвеєра, мінімізуючи технічні помилки на етапі введення інформації.

2.5.3. Технологічний стек та інструментальна база обробки даних

Для реалізації описаних механізмів валідації та управління даними було обрано стек бібліотек мови програмування Python, які є галузевим стандартом у сфері Data Science та забезпечують високу детермінованість обчислень:

- Pandas використовується як основний рушій для маніпуляції табличними даними. Завдяки високорівневим структурам DataFrame, бібліотека забезпечує ефективне зчитування файлів, автоматичне виявлення типів даних та управління індексацією атрибутів.

- NumPy залучається для виконання векторизованих математичних операцій та обробки числових масивів. Зокрема, вона забезпечує коректну роботу з об’єктами NaN (Not a Number) при ідентифікації пропущених значень.

- Chardet / Charset-normalizer інтегровані для реалізації механізму автоматичного детектування кодування вхідних текстових файлів, що мінімізує ризик виникнення помилок при парсингу нестандартних форматів.

- Scikit-learn слугує базою для реалізації методів попередньої обробки (наприклад, класів StandardScaler або OneHotEncoder). Це гарантує, що трансформації даних у візуальному середовищі відповідають загальноприйнятим математичним стандартам машинного навчання.

2.5.4. Оптимізація ресурсів пам'яті

Враховуючи обмеження обсягу даних до 200 МБ, у системі реалізовано стратегію Lazy Loading та оптимізацію типів даних. Зокрема, при завантаженні великих таблиць програма автоматично понижує розрядність числових типів (наприклад, з float64 до float32), якщо це не впливає на точність обчислень лінійної регресії. Це дозволяє зберігати високу швидкість відгуку інтерфейсу навіть при одночасній роботі з декількома етапами препроцесингу.

2.5.5. Аналіз ризиків та системні вимоги

Проєктування будь-якої інтелектуальної системи передбачає ідентифікацію потенційних ризиків, що можуть вплинути на цілісність даних або стабільність навчального процесу.

Таблиця 2.1 – Категорії ризиків проекту

Категорія ризику	**Опис загрози**	**Заходи з мінімізації**
Втрата стану сесії	Раптове оновлення сторінки або розрив з'єднання призводить до повного скидання прогресу обробки даних.	Впровадження модульної структури, що дозволяє швидко повторити кроки обробки за збереженим алгоритмом дій.
Дефіцит пам'яті	При роботі з файлами на межі ліміту (близько **200 МБ**) можливе уповільнення інтерфейсу або аварійне завершення вкладки браузера.	Використання оптимізованих типів даних (наприклад, `float32`) та механізмів очищення кешу після виконання ітерації моделювання.
Конфлікт типів	Некоректне введення даних користувачем (наприклад, змішування числових та текстових значень) може призвести до помилок при навчанні моделі.	Багаторівнева валідація на етапі інгестії та блокування виконання моделі до моменту усунення невідповідностей у типах ознак.

Оскільки розроблене середовище виконує інтенсивні обчислення на стороні клієнта та сервера (залежно від конфігурації розгортання), визначено наступні мінімальні та рекомендовані системні вимоги для стабільної роботи:

Апаратні вимоги:

- Процесор: 2-ядерний процесор із частотою не менше 2.0 ГГц (рекомендовано 4-ядерний для паралелізації процесів візуалізації).

- Оперативна пам'ять: мінімально: 4 ГБ (для роботи з малими датасетами до 50 МБ). Рекомендовано: 8 ГБ і більше (для забезпечення безперебійної роботи з наборами даних до 200 МБ та одночасного запуску декількох візуалізацій).

- Відеокарта: Інтегроване рішення з підтримкою апаратного прискорення рендерингу графіки в браузері.

Програмні вимоги:

- Операційна система: Windows 10/11, macOS 11+, або актуальні дистрибутиви Linux (Ubuntu 20.04+).

- Веббраузер: будь-який сучасний браузер на базі рушія Chromium

(Chrome, Edge), Firefox або Safari. Наявність увімкненого JavaScript та підтримки WebGL є обов'язковою.

- Мережа: стабільне підключення до мережі Інтернет зі швидкістю не менше 5 Мбіт/с (для завантаження інтерфейсу та передачі метаданих).

2.6. Програмно-апаратна реалізація та технологічний стек системи

Архітектура розроблюваного програмного продукту базується на принципах модульності та кросплатформенності. Вибір технологічного стека зумовлений необхідністю забезпечення високої швидкодії при обробці статистичних даних та створення інтуїтивного інтерфейсу, що відповідає сучасним стандартам UX/UI у науковому програмному забезпеченні.

2.6.1. Реалізація інтерфейсу користувача та візуалізації

Клієнтська частина додатка побудована на базі Streamlit — реактивного фреймворку для розробки вебзастосунків засобами мови Python. Використання даного фреймворку дозволило уніфікувати розробку фронтенд- та бекенд-частин, забезпечуючи високу ремонтопридатність коду.

Для побудови графічного інтерфейсу та систем візуалізації було використано наступні компоненти:

- Стандартна бібліотека компонентів Streamlit: форми введення, інтерактивні повзунки для регулювання гіперпараметрів, кнопки управління станом та бічна панель навігації. Це дозволило структурувати робочий процес ML-конвеєра у вигляді послідовних логічних блоків.

- Кастомізація через HTML/CSS ін'єкції: для реалізації специфічних елементів (модальних вікон та плаваючих навігаційних кнопок), що виходять за межі стандартних шаблонів, використано пряму вставку стилів та розмітки.

- Streamlit-AgGrid: інтеграція високопродуктивного табличного

компонента для забезпечення інтерактивної взаємодії з наборами даних (сортування, фільтрація та редагування значень у реальному часі).

- Vega-Altair: бібліотека декларативної візуалізації, обрана для побудови статистичних графіків. Вона забезпечує високу якість рендерингу та нативну підтримку реактивності інтерфейсу.

2.6.2. Архітектура бекенд-частини та логіка обробки даних

Бекенд системи реалізовано як сукупність автономних модулів на мові Python. Основна логіка розподілена між компонентами, що відповідають за управління станом сесії, виконання математичних перетворень та симуляцію навчання моделей.

Ключовий інструментарій бекенд-розробки включає наступні бібліотеки. Pandas виконує роль основного рушія для маніпулювання табличними даними. Бібліотека забезпечує зчитування, серіалізацію та тимчасове зберігання об'єктів у межах користувацької сесії. NumPy залучається для виконання векторних обчислень, необхідних на етапах нормалізації та інженерії ознак. Scikit-learn надає математичну базу для реалізації моделей лінійної регресії, алгоритмів розподілу вибірки на навчальну/тестову підмножини та розрахунку метрик адекватності (MSE, R2). Модульна структура коду дозволяє легко масштабувати функціонал, додаючи нові типи моделей або методи препроцесингу без порушення цілісності основної системи. Керування станом сесії (Session State) забезпечує послідовність збереження параметрів між ітераціями взаємодії користувача з інтерфейсом.

2.6.3. Стратегія розгортання та хостинг

Для забезпечення публічного доступу до інтерактивного середовища обрано модель PaaS. Як первинну платформу розгортання було використано

Heroku, що надає стабільне середовище для виконання Python-додатків із підтримкою автоматичного масштабування ресурсів.

Як альтернативний варіант для майбутніх ітерацій розглядається Streamlit Community Cloud, що пропонує спеціалізовану інфраструктуру для хостингу аналітичних застосунків. Завдяки легкій архітектурі, програмний комплекс є мобільним і може бути розгорнутий на будь-якому сучасному хмарному сервісі з підтримкою Docker-контейнеризації.

2.6.4. Методологія підбору наборів даних

Для забезпечення можливості негайної експлуатації системи початківцями, у склад програмного комплексу інтегровано бібліотеку еталонних наборів даних. Вибірка здійснювалася на основі платформи Kaggle за наступними критеріями:

- придатність до регресійного аналізу - наявність чітко вираженої цільової змінної.

- гетерогенність ознак - наявність як числових, так і категоріальних атрибутів для демонстрації етапів кодування та масштабування.

- включення датасетів із пропущеними значеннями та аномаліями для візуалізації роботи алгоритмів очищення даних.

Це дозволяє користувачеві опанувати функціонал системи без попередньої підготовки власних даних, використовуючи кураторські приклади як базу для експериментів.

Таким чином було проведено детальний аналіз архітектурних рішень, функціональних можливостей та математичного обґрунтування розробленого програмного середовища. Описані модулі управління даними, препроцесингу, інженерії ознак та моделювання утворюють цілісну екосистему для вивчення основ машинного навчання.

Впровадження парадигми no-code та орієнтація на візуальну взаємодію дозволили створити інструмент із низьким порогом входження, що зберігає

при цьому високу наукову достовірність результатів. Врахування технічних обмежень та ризиків на етапі проєктування гарантує стійкість системи у навчальних сценаріях, забезпечуючи надійну платформу для подальшого масштабування та інтеграції складніших алгоритмів штучного інтелекту.

2.7 Висновки по розділу

У другому розділі сформовано методологічну основу функціонування системи тестування моделей машинного навчання. Обґрунтовано вибір методу лінійної регресії як базового інструменту навчального середовища з огляду на його інтерпретованість і простоту реалізації. Визначено ключові функціональні та нефункціональні вимоги до системи, що забезпечують її ефективність, надійність і зручність використання. Розроблено архітектуру управління даними, яка охоплює процеси валідації, обробки та збереження інформації. Окрему увагу приділено технологічному стеку, оптимізації ресурсів і аналізу ризиків, що забезпечує стабільність та масштабованість системи.

РОЗДІЛ 3. ПРОЕКТУВАННЯ СИСТЕМИ ТЕСТУВАННЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

3.1. Проєктування функціональної моделі взаємодії у формі діаграми варіантів використання

Для детермінації логіки взаємодії між користувачем та розробленою системою побудовано діаграму варіантів використання (Use Case Diagram). Ця модель дозволяє візуалізувати функціональні межі системи та визначити ролі основних акторів у процесі машинного навчання.

Побудована діаграма варіантів використання слугує високорівневою моделлю функціональних можливостей інтерактивного середовища, визначаючи межі системи та патерни взаємодії між основними суб'єктами процесу машинного навчання. Модель структурована таким чином, щоб відобразити ітеративну природу дослідження даних та логіку переходу від первинної обробки до оцінки предиктивних моделей.

У межах системи детерміновано два типи акторів, кожен з яких володіє специфічним набором повноважень та функцій:

- Користувач (User) виступає як основний ініціатор дослідницького процесу. Його роль полягає у прийнятті рішень щодо вибору методів обробки, налаштування архітектури моделі та інтерпретації отриманих результатів.

- Система (System) виконує роль інтелектуального середовища виконання, забезпечуючи автоматизацію рутинних обчислювальних операцій, контроль цілісності даних та візуалізацію станів.

Основними сценаріями, що реалізують програмна система є управління даними: завантаження датасетів з обов'язковим включенням процедури маркування ознак, маніпуляції з вибіркою, застосування препроцесингу, редагування таблиць та модифікація простору ознак, робота з історією.

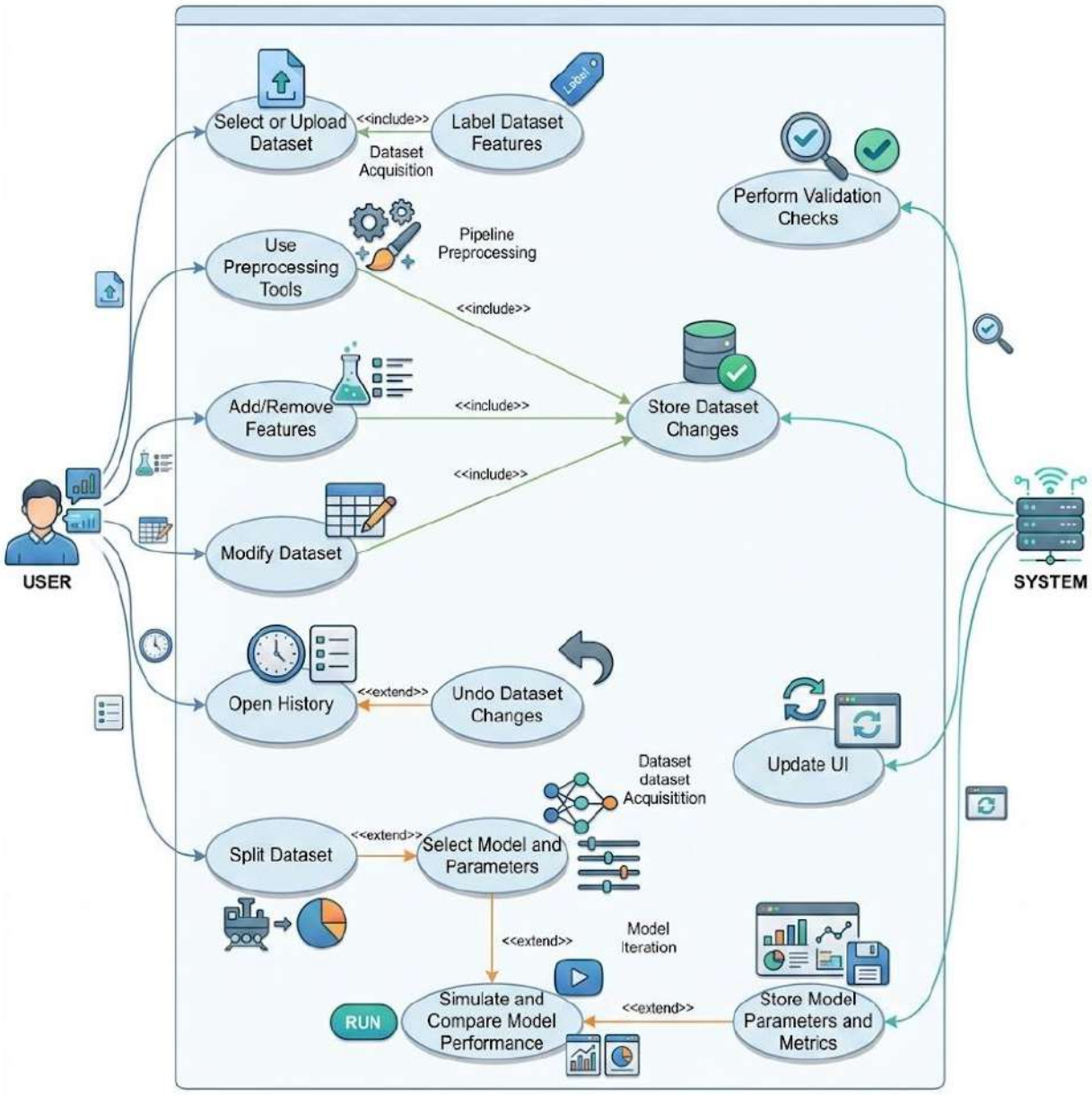

Рисунок 3.1 – Діаграма варіантів використання

Діаграма відображає логічну послідовність, розподілену на чотири основні функціональні домени:

1. Домен отримання та структурування даних (Dataset Acquisition).

Процес розпочинається з варіанта використання «Select or Upload Dataset». Даний сценарій має безумовний зв’язок <<include>> із прецедентом «Label Dataset Features», що підкреслює необхідність семантичного визначення типів атрибутів (цільова змінна, категоріальні ознаки) як обов’язкову умову для подальшого аналізу.

2. Домен конвеєрної передобробки (Pipeline Preprocessing):

Цей кластер охоплює операції «Use Preprocessing Tools», «Add/Remove Features» та «Modify Dataset». Ключовою архітектурною особливістю є інтеграція зв'язку <<include>> між цими операціями та сценарієм «Store Dataset Changes». Це гарантує, що будь-яка трансформація даних автоматично фіксується в оперативній пам'яті системи, забезпечуючи консистентність стану сесії.

3. Домен управління станом та ретроспективного аналізу:

Сценарій «Open History» дозволяє користувачеві відстежувати еволюцію змін. Він розширюється (<<extend>>) функціоналом «Undo Dataset Changes», що надає механізм відкату до попередніх станів, мінімізуючи ризики незворотної втрати результатів під час експериментування.

4. Домен ітераційного моделювання (Model Iteration):

Заключний етап життєвого циклу починається з розподілу даних («Split Dataset»). Цей процес ініціює ланцюжок розширень: вибір моделі та параметрів («Select Model and Parameters»), безпосередню симуляцію та порівняння («Simulate and Compare Model Performance»). Кінцевою точкою є автоматизоване збереження параметрів та метрик («Store Model Parameters and Metrics»), що дозволяє проводити бенчмаркінг різних конфігурацій.

Роль Системи на діаграмі представлена через низку критичних сервісних сценаріїв:

- Perform Validation Checks: безперервний моніторинг вхідних даних на відповідність типам та відсутність критичних похибок.

- Update UI: динамічне оновлення графічних інтерфейсів та інфографіки у відповідь на маніпуляції користувача.

- Store Model Parameters and Metrics: автоматизоване логування результатів, що забезпечує наукову достовірність та можливість відтворення експерименту.

Представлена модель варіантів використання підтверджує, що архітектура системи спрямована на створення безшовного переходу між теоретичними концепціями та їх практичною реалізацією. Чітке розмежування обов'язкових та опціональних сценаріїв дозволяє користувачеві гнучко керувати процесом навчання, зберігаючи при цьому строгу логіку побудови аналітичного рішення.

3.2. Моделювання динамічного аспекту системи на основі діаграми активностей

Діаграма активностей, представлена на рис. 3.2, детермінує логічну послідовність операцій та керуючих потоків у межах розробленого програмного середовища. Вона слугує інструментом для візуалізації алгоритмічного базису системи, охоплюючи всі етапи трансформації даних: від ініціалізації вхідного потоку до фінальної верифікації предиктивної моделі.

Модель відображає ієрархічну структуру процесів — від ініціалізації вхідного набору даних до фінальної верифікації та бенчмаркінгу предиктивних моделей.

1. Етап інгестії та початкової верифікації

Процес ініціюється станом «Завантаження або вибір набору даних», після чого слідує процедура семантичної «Розмітки ознак». Ключовим вузлом прийняття рішень на початковому етапі є перевірка «Набір даних дійсний?».

У разі виявлення структурних невідповідностей система переходить до стану інформування користувача про критичні помилки, що завершує поточний потік виконання до моменту їх усунення.

При успішній верифікації активується стан візуалізації даних, що відкриває доступ до багатовекторного інструментарію обробки.

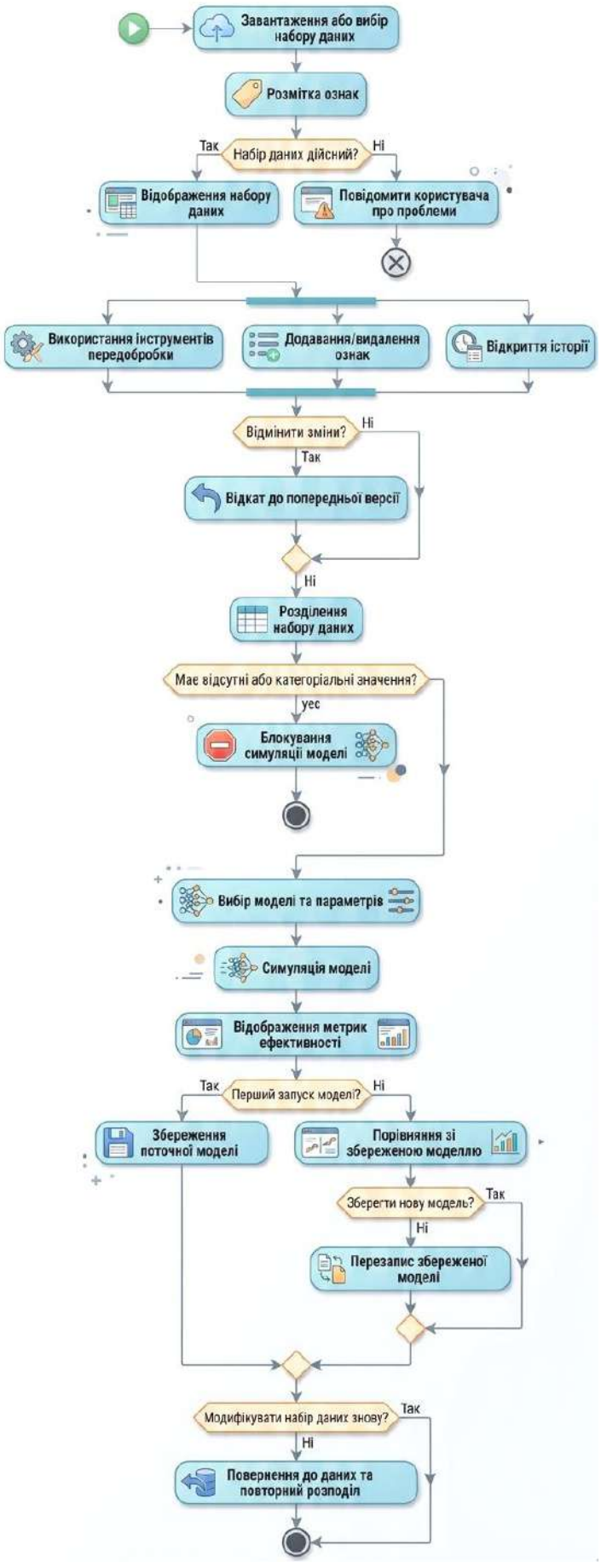

Рисунок 3.2 – Діаграма активностей

2. Паралельні процеси обробки та управління станом

Система надає користувачеві можливість вибору між трьома основними напрямами маніпуляцій, які можуть виконуватися ітераційно:

- Використання інструментів передобробки: виконання операцій очищення та нормалізації.

- Додавання/видалення ознак: активна інженерія простору параметрів.

- Відкриття історії: ретроспективний аналіз змін із можливістю активації сценарію «Відкат до попередньої версії» через відповідний вузол розгалуження.

3. Стратифікація вибірки та контроль готовності моделі

Важливим етапом алгоритму є «Розділення набору даних» на навчальну та тестову підмножини. Перед безпосереднім переходом до моделювання система виконує критичну перевірку на наявність пропущених значень або некодованих категоріальних ознак.

Якщо дані не відповідають вимогам обраного алгоритму (наприклад, лінійної регресії), система активує стан «Блокування симуляції моделі», що запобігає виникненню помилок часу виконання та забезпечує надійність обчислень.

4. Цикл симуляції, порівняльного аналізу та ітераційного вдосконалення

Процес моделювання охоплює вибір гіперпараметрів, безпосередню симуляцію та генерацію аналітичного звіту з метриками ефективності (MSE,R2). Логіка збереження результатів детермінована умовою «Перший запуск моделі?»:

- при первинній ітерації здійснюється автоматичне збереження поточної конфігурації.

- при повторних запусках активується блок порівняльного аналізу, де користувач приймає рішення щодо перезапису існуючої моделі більш ефективною версією.

Алгоритм завершується вузлом зворотного зв'язку «Модифікувати набір даних знову?». Даний цикл дозволяє реалізувати парадигму машинного навчання, де користувач повертається до етапу інженерії ознак для досягнення цільових показників точності, що робить робочий потік нелінійним та адаптивним.

3.3. Динамічне моделювання системних взаємодій на основі діаграми послідовностей

Для деталізації часових аспектів функціонування системи та визначення логіки обміну повідомленнями між об'єктами побудовано діаграму послідовностей. Ця модель дозволяє відстежити життєвий цикл запиту користувача та взаємодію між фронтенд-інтерфейсом, менеджером стану сесії та функціональними модулями обробки.

У процесі взаємодії задіяні наступні ключові сутності:

- Актор (User): ініціатор подій.

- Інтерфейс користувача (UI): прошарок візуалізації та збору вхідних параметрів.

- Менеджер стану (SessionState): центральний вузол збереження актуальних даних у межах поточної сесії.

- Інструмент історії (HistoryTool): модуль контролю версій набору даних.

- Функціональні модулі (Preprocessing, Feature Engineering, Split): блоки логіки для трансформації даних.

- Симулятор та Візуалізатор (ModelSimulator, ResultsDisplay): обчислювальне ядро та модуль генерації аналітичних звітів.

Процес взаємодії структуровано за логічними етапами:

1. Ініціалізація та версіонування. При завантаженні набору даних через UI, повідомлення передається до SessionState для кешування. Одночасно

спрацьовує виклик до HistoryTool для створення початкової контрольної точки («Save version»).

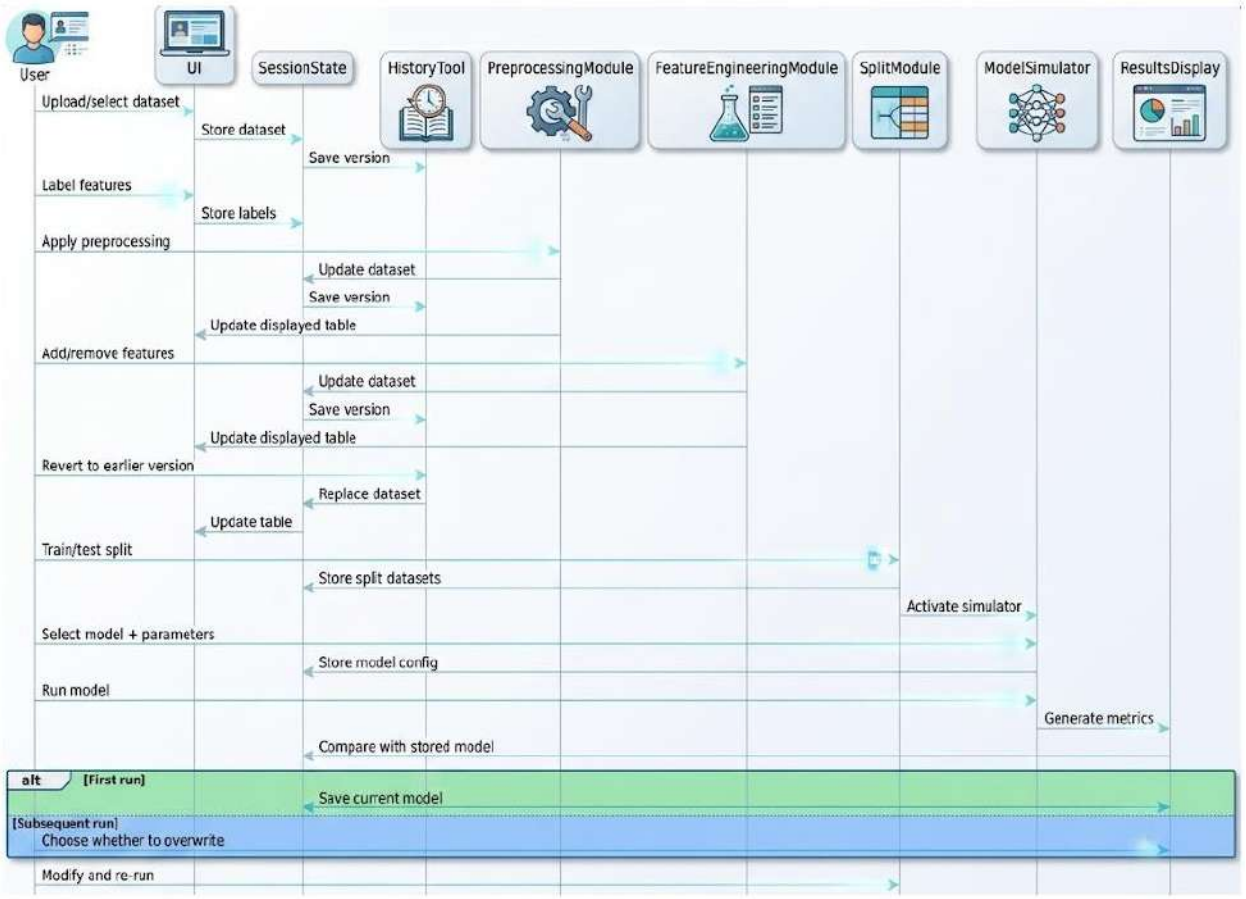

Рисунок 3.3 – Діаграма послідовності

2. Ітераційна трансформація. При застосуванні інструментів передобробки або інженерії ознак спостерігається синхронний обмін: модуль обробки повертає оновлений масив даних до SessionState, ініціює збереження нової версії в історії та надсилає сигнал до UI для динамічного оновлення табличного відображення.

3. Обчислювальний конвеєр: Етап навчання активується через SplitModule, який після сегментації даних ініціює запуск ModelSimulator.

4. Аналітична обробка та логіка розгалуження (alt): Після генерації метрик у ResultsDisplay система виконує порівняльний аналіз.

- У сценарії «Перший запуск» модель зберігається автоматично.

- У сценарії «Наступні запуски» реалізовано механізм запиту до користувача щодо необхідності перезапису існуючої конфігурації («Choose whether to overwrite»).

Застосована модель підтверджує, що архітектура системи забезпечує високу цілісність даних завдяки централізованому управлінню станом (SessionState). Використання механізму зворотних викликів між модулями обробки та інструментом історії гарантує відтворюваність експериментів, що є критично важливим для освітніх застосувань у галузі машинного навчання.

3.4 Висновки по розділу

У третьому розділі виконано проєктування системи тестування моделей машинного навчання з використанням сучасних методів моделювання. Побудовано діаграми варіантів використання, що дозволило чітко визначити основні сценарії взаємодії користувача із системою. Розроблено діаграми активностей, які відображають послідовність виконання процесів і логіку обробки даних. Також сформовано діаграми послідовностей, що деталізують взаємодію між окремими компонентами системи. Отримані результати забезпечили цілісне бачення архітектури системи та створили основу для її подальшої програмної реалізації.

РОЗДІЛ 4. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВІЗУАЛІЗАЦІЇ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

4.1. Архітектурна організація та функціональна декомпозиція графічного інтерфейсу системи

Проєктування графічного інтерфейсу системи візуалізації та апробації моделей машинного навчання ґрунтується на принципах ергономічності, когнітивної доступності та інтуїтивно зрозумілої навігації.

Архітектура макета має вертикальну орієнтацію, що детермінує послідовну реалізацію етапів типового конвеєра та спрямовує користувача системи крізь логічні блоки обробки даних.

Програмне середовище розділене на дві основні функціональні зони, що забезпечують комплексне управління процесом моделювання моделей машинного навчання:

- Панель управління (Sidebar) - контрольний елемент системи, де зосереджено основний інструментарій препроцесингу, параметри конфігурації моделей та опції вибору методів інженерії ознак. Таке розміщення дозволяє зберігати постійний доступ до налаштувань без перекриття зони візуалізації.

- Центральна область візуалізації (Main Display) - призначена для відображення динамічних табличних структур (датасетів), інтерактивних графіків та результатів аналітичного виводу. Дана зона забезпечує пряму візуальну верифікацію результатів застосованих перетворень.

Стартова робоча область, представлена на рисунку 3.4, є фундаментальною точкою входу в систему, де ініціюється життєвий цикл аналізу даних.

Рисунок 4.1 - Інтерфейс стартової робочої області

Даний екран реалізує наступні критичні функції:

- Інгестія даних.

Забезпечення механізмів імпорту користувацьких наборів даних або вибору еталонних зразків із вбудованого репозиторію для проведення пілотних експериментів.

- Семантичне маркування атрибутів.

Можливість визначення метаданих стовпців (ідентифікація цільової ознаки, категоріальних та числових змінних), що є обов'язковим пререквізитом для коректного функціонування математичного апарату лінійної регресії та методів попередньої обробки.

4.2. Аналітичний функціонал основного робочого середовища та первинна діагностика даних

Після завершення процедури семантичного маркування ознак програмний комплекс ініціює перехід до основного робочого середовища (Main Workspace), архітектура якого представлена на рис. 3.5. Даний інтерфейс виступає центральним вузлом взаємодії користувача з системою,

забезпечуючи поглиблену візуалізацію структури та якості завантаженої вибірки.

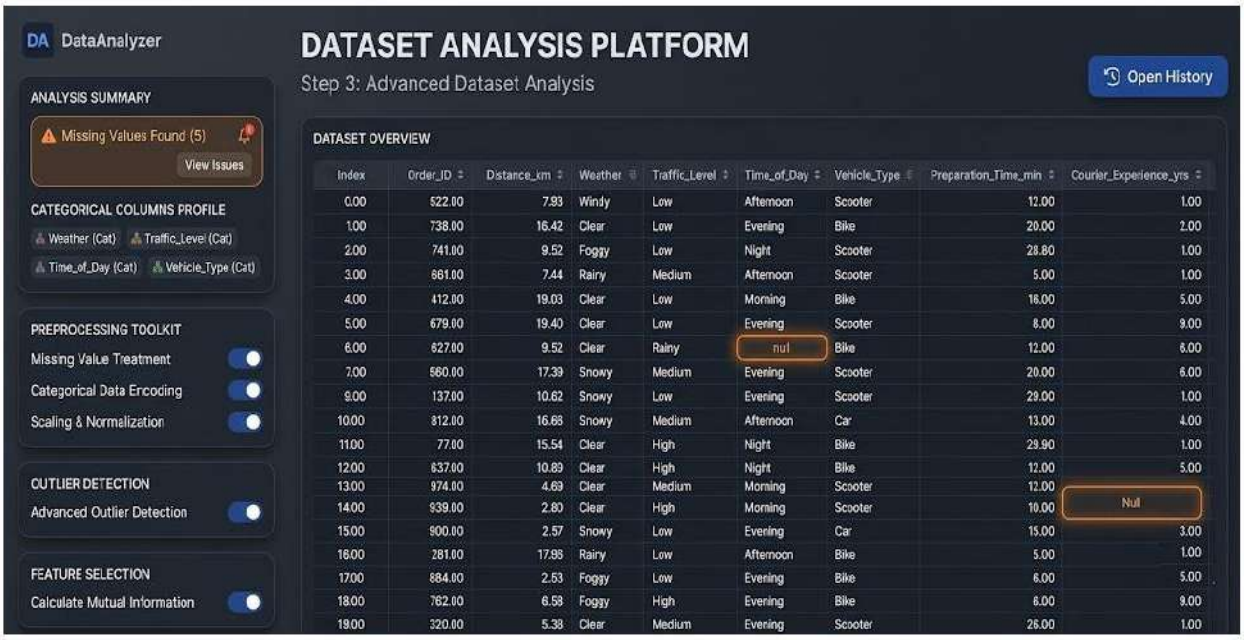

Рисунок 4.2 - Основне робоче середовище інтерфейсу користувача

Модуль статистичної діагностики та аудиту цілісності

На цьому етапі система проводить автоматизований технічний аудит, результати якого виводяться в інтерактивній формі. Ключові аспекти діагностики включають:

- Контроль цілісності даних: Ідентифікація та кількісна оцінка пропущених значень (NaN/Null). Виявлення пустот у даних є критичним фактором для вибору подальшої стратегії імпутації, що безпосередньо впливає на стійкість регресійної моделі.

- Огляд метаданих вибірки: Відображення зведеної статистики за кожною ознакою, що дозволяє дослідникові оцінити діапазони значень, розподіли та потенційні аномалії ще до початку етапу моделювання.

Основний дисплей також виконує роль оркестратора доступних функціональних можливостей:

- Інструментальний стек: Користувачеві надається доступ до контекстно-залежних інструментів препроцесингу та інженерії ознак, склад яких може змінюватися залежно від виявлених характеристик даних.

- Журналювання та контроль станів (History Option): Інтерфейс передбачає наявність модуля історії, який дозволяє відстежувати хронологію застосованих трансформацій. Дана функція забезпечує відтворюваність експерименту та можливість ітераційного повернення до попередніх станів системи у разі погіршення показників адекватності моделі.

Представлений на рисунку інтерфейс (рис. 3.5) реалізує концепцію багатомодульної інформаційної панелі (Dashboard), що призначена для виконання етапів розвідувального аналізу даних та їх підготовки до статистичного моделювання. Дизайн системи базується на принципах функціонального зонування, де центральна робоча область інтегрована з контекстними інструментами управління.

Архітектура інтерфейсу детермінована логікою ML-конвеєра і розділена на два основні вектори взаємодії: аналітичний супровід та пряму маніпуляцію даними.

1. Модульна архітектура панелі управління (Sidebar)

Ліва частина інтерфейсу об'єднує в собі діагностичні та керуючі модулі, що дозволяють здійснювати тонке налаштування процесів обробки:

- Analysis Summary (Аудит цілісності): Реалізує функцію оперативного моніторингу якості вибірки. Система автоматично ідентифікує кількість критичних похибок (наприклад, 5 виявлених пропущених значень), надаючи досліднику миттєвий сигнал про необхідність вжиття заходів щодо очищення даних.

- Categorical Columns Profile: Модуль семантичного аналізу, що візуалізує метадані атрибутів. Виокремлення категоріальних ознак (наприклад, Weather, Traffic_Level) у вигляді інтерактивних тегів дозволяє швидко оцінити структуру простору ознак.

- Preprocessing Toolkit (Інструментарій трансформації): Використання перемикачів (Toggles) замість статичних форм забезпечує динамічне управління етапами імпутації («Missing Value Treatment»), кодування та нормалізації. Це дозволяє досліднику в реальному часі експериментувати з конфігурацією даних.

- Outlier Detection & Feature Selection: Додаткові інтелектуальні модулі для детекції аномалій та обчислення взаємної інформації, що спрямовані на підвищення репрезентативності вибірки.

2. Центральна область візуалізації та контролю (Main Workspace)

Основна робоча область представлена у вигляді високопродуктивної інтерактивної таблиці, яка реалізує принцип прямого візуального зворотного зв'язку:

- Візуальна індикація аномалій: Відсутні значення (null/Null) виділяються за допомогою кольорового акцентування (м'який помаранчевий контур). Така візуальна ієрархія дозволяє локалізувати дефекти в наборі даних, не перевантажуючи дослідника цифровим шумом.

- Інформативність заголовків: Кожна колонка містить іконки-індикатори типу даних та елементи сортування, що полегшує навігацію великими масивами інформації.

- Механізм ретроспекції: Кнопка «Open History» із піктограмою годинника забезпечує доступ до журналювання станів, що є критично важливим для наукової відтворюваності експериментів.

Даний інтерфейс успішно трансформує складні математичні процедури попередньої обробки у візуально зрозумілий процес, знижуючи ймовірність помилок людського фактора на етапі підготовки даних.

Таким чином, основне робоче середовище реалізує принцип прозорості даних, надаючи користувачеві вичерпну інформацію для прийняття обґрунтованих рішень щодо стратегії подальшого інтелектуального аналізу.

Це мінімізує ризики «сліпого» моделювання та сприяє розвитку аналітичної інтуїції у користувача.

4.3. Методологія та програмна реалізація модуля імпутації відсутніх даних

Першим етапом у конвеєрі підготовки даних, представленим у розробленому середовищі, є модуль відновлення цілісності інформації. На рис. 4.3 продемонстровано інтерфейс інструментарію обробки пропущених значень, який дозволяє мінімізувати втрати репрезентативності вибірки без видалення цілих записів.

Рисунок 4.3 - Функціональний інтерфейс модуля імпутації відсутніх значень

На відміну від методів автоматичного видалення рядків (listwise deletion), реалізований інструментарій надає користувачеві можливість селективної імпутації. Це дозволяє застосовувати індивідуальні математичні стратегії до кожної окремої ознаки залежно від її статистичної природи:

- для числових ознак: заміна відсутніх значень середнім арифметичним (mean), медіаною або константним значенням. Вибір медіани є пріоритетним у випадках наявності аномальних відхилень (викидів) у розподілі ознаки.

- для категоріальних ознак: використання моди або маркування пропуску як окремої категорії («Unknown»), що дозволяє зберегти структурну цілісність даних для алгоритмів кодування.

Використання даного модуля в межах інтерактивної сесії забезпечує:

- Збереження обсягу вибірки: критично важливо для малих наборів даних, де видалення навіть 5–10% записів може призвести до недонавчання (underfitting) моделі.

- Контрольованість трансформацій: користувач може візуально спостерігати за тим, як заповнення пропусків впливає на статистичні показники колонки в режимі реального часу.

- Запобігання помилкам виконання: автоматична валідація наявності NaN-значень перед запуском симуляції регресії нівелює ризик аварійного завершення обчислювальних процесів.

Таким чином, реалізований механізм імпутації виступає базовим фільтром, що трансформує «сирі» дані у валідну структуру, готову до подальшого аналізу та моделювання.

4.4. Архітектура та математичне обґрунтування модуля кодування категоріальних ознак

Наступним етапом у конвеєрі попередньої обробки даних є трансформація якісних (категоріальних) атрибутів у числовий формат, придатний для обробки алгоритмами регресійного аналізу. На рис. 4.4 представлено інтерфейс модуля кодування, який забезпечує гнучкість вибору стратегій квантифікації даних.

Рисунок 4.4 - Програмний інтерфейс модуля кодування категоріальних змінних

Програмне середовище надає користувачеві можливість вибору між декількома фундаментальними методами перетворення, кожен з яких має специфічну область застосування:

- One-Hot Encoding - створення фіктивних змінних для категорій, що не мають внутрішнього ранжування (наприклад, типи транспортних засобів або погодні умови). Це дозволяє уникнути хибної інтерпретації моделлю числових ваг як порядкових значень.

- Label Encoding - присвоєння унікального цілочисельного значення кожній категорії. Даний метод є ефективним для ознак із природною ієрархією (наприклад, рівень інтенсивності трафіку: "Low", "Medium", "High").

Критично важливим аспектом реалізованого інтерфейсу є блок інтерактивної превізуалізації. Перш ніж підтвердити внесення змін до основної структури датасету, система генерує фрагмент трансформованої таблиці. Це дозволяє:

- оцінити збільшення кількості ознак після застосування One-Hot Encoding.

- впевнитися, що кодування відбулося коректно і не призвело до появи артефактів або втрати інформації.

- проаналізувати нові вектори ознак на предмет лінійної залежності.

Впровадження модуля кодування з функцією попереднього перегляду забезпечує високу наукову достовірність підготовки даних. Це гарантує, що вхідні параметри моделі будуть математично коректними, що є необхідною умовою для отримання адекватних статистичних оцінок та мінімізації похибки апроксимації.

4.5. Методологія аналізу взаємної залежності та ітераційного синтезу ознак

На рисунку 4.5 продемонстровано архітектурну реалізацію компонентів інженерії ознак, що включають блоки оцінки взаємної інформації та моделювання міжфункціональних взаємодій.

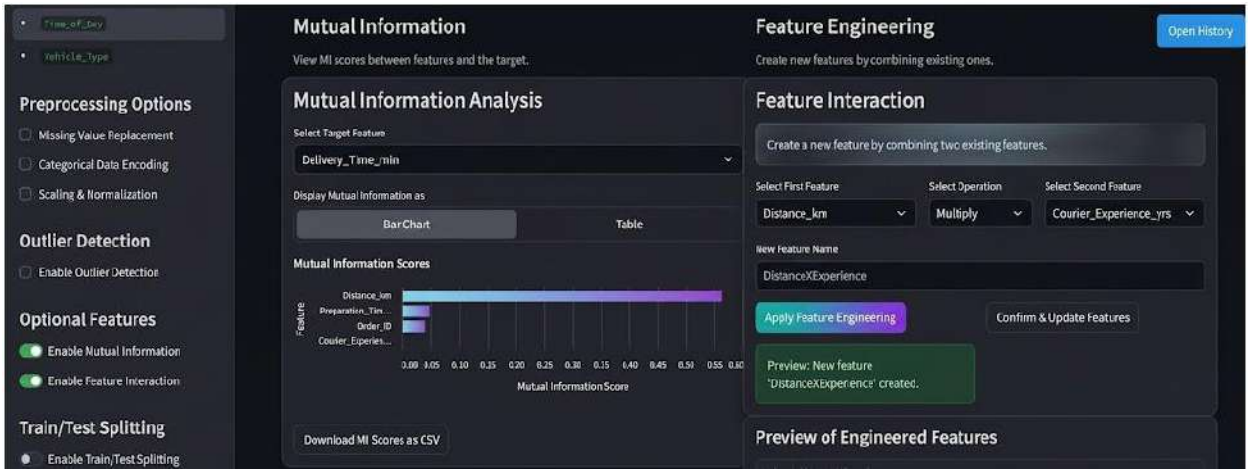

Рисунок 4.5 - Інтерфейс модулів інженерії ознак та аналізу взаємозв'язків

Даний інструментарій призначений для підвищення предиктивної здатності моделей шляхом виявлення нелінійних залежностей та оптимізації простору ознак.

Модуль аналізу взаємної інформації базується на принципах теорії інформації та дозволяє визначити ступінь статистичної залежності між вхідними атрибутами та цільовою змінною. На відміну від лінійної кореляції, даний підхід:

- Ідентифікує нелінійні зв'язки, що є критичним для вибору найбільш інформативних предикторів.

- Ранжує ознаки за їхньою вагою дозволяючи досліднику відсіювати малоінформативний «шум», що безпосередньо запобігає перенавчанню моделі.

Функціонал «взаємодії ознак» надає можливість генерації нових, комбінованих атрибутів на основі існуючих даних. Процес ітераційного моделювання включає:

- Конструювання нових змінних: шляхом застосування математичних операцій до декількох ознак одночасно.

- Валідацію впливу: оцінку зміни метрик якості моделі (наприклад, R2 або MSE) після впровадження нових параметрів.

- Динамічне управління вектором ознак: можливість оперативного видалення надлишкових або корельованих атрибутів для досягнення найбільш компактної та ефективної архітектури даних.

4.6. Архітектурна реалізація модуля вибору алгоритмів та параметризації моделей

На рисунку 4.6 представлено програмний інтерфейс блоку моделювання, що забезпечує комплексний контроль над процесом навчання та налаштування предиктивних алгоритмів. Даний розділ є критичною ланкою системи, де здійснюється перехід від етапу підготовки ознак до безпосередньої побудови математичних залежностей.

Рисунок 4.6 - Інтерфейс вибору архітектури моделі та управління простором гіперпараметрів

Поточна ітерація програмного продукту сфокусована на використанні сімейства алгоритмів лінійної регресії, що виступають еталонними методами для аналізу неперервних числових послідовностей. Гнучкість інтерфейсу дозволяє:

- Обирати базовий алгоритм: здійснювати перемикання між різними модифікаціями регресійних моделей залежно від статистичних характеристик вибірки.

- Масштабувати стек моделей: архітектура системи розроблена з урахуванням подальшої інтеграції складніших методів (наприклад, випадкових лісів або нейронних мереж) без втрати цілісності користувацького досвіду.

Для забезпечення балансу між функціональною повнотою та ергономічністю, модуль налаштування параметрів реалізовано за принципом ієрархічного розкриття:

- за замовчуванням користувачеві пропонуються найбільш значущі гіперпараметри (наприклад, коефіцієнт швидкості навчання або параметри

регуляризації L1 та L2), що визначають здатність моделі до узагальнення та запобігають ефекту перенавчання.

- користувач має можливість активувати розширений режим, що відкриває доступ до всіх внутрішніх змінних конкретного алгоритму. Це дозволяє здійснювати прецизійну детермінацію параметрів оптимізації, що є необхідним для досягнення максимальних показників адекватності моделі.

Таким чином, реалізований модуль симуляції трансформує процес математичного моделювання у візуально контрольовану процедуру. Це дозволяє досліднику оперативно оцінювати вплив окремих гіперпараметрів на результуючу точність прогнозування, що істотно скорочує час на проведення ітераційних експериментів.

4.7. Модуль аналітичної верифікації та інтерпретації показників ефективності

На рисунку 4.7 представлено графічний інтерфейс модуля аналітичного оцінювання, призначений для візуалізації та об’єктивної інтерпретації результатів апробації побудованих моделей. Даний блок забезпечує дослідника вичерпним набором статистичних індикаторів, необхідних для верифікації адекватності математичного апарату.

Програмна реалізація системи передбачає динамічну зміну складу метрик залежно від обраної архітектури моделі та типу аналітичної задачі. Це забезпечує релевантність оцінювання та дозволяє уникнути методологічних помилок при порівнянні різних алгоритмів.

Для моделей лінійної регресії, що інтегровані в поточний модуль, система автоматично генерує наступний набір аналітичних даних:

- Графічний аналіз залишків: візуалізація розподілу помилок на тренувальній та тестовій вибірках, що дозволяє ідентифікувати наявність гетероскедастичності або системних зміщень у моделі.

- Середньоквадратична помилка (MSE): кількісний показник відхилення прогнозних значень від фактичних, що слугує основним критерієм мінімізації цільової функції.

- Коефіцієнт детермінації (R2): статистична міра, що визначає частку дисперсії залежної змінної, яка пояснюється даною моделлю, і характеризує загальну прогнозну силу алгоритму.

Рисунок 4.7 - Візуалізація результатів моделювання та аналіз метрик продуктивності

Варіативність метрик залежно від типу моделі (наприклад, перехід до матриць невідповідностей — confusion matrix — при класифікації) дозволяє досліднику проводити прецизійний аналіз помилок першого та другого роду. Такий підхід гарантує високу достовірність отриманих результатів та забезпечує наукову базу для подальшої оптимізації гіперпараметрів.

Інтеграція модуля візуалізації метрик у єдине робоче середовище дозволяє в режимі реального часу відстежувати вплив змін у препроцесингу або конфігурації моделі на кінцеву точність прогнозування. Це забезпечує ітераційну вдосконалюваність моделі та сприяє виявленню найбільш ефективних стратегій обробки даних для конкретних прикладних задач.

4.8. Архітектурна реалізація та програмна інженерія системи

4.8.1 Дизайн та ергономіка інтерфейсу користувача

Графічний інтерфейс системи побудований на принципах ієрархічного робочого процесу, що відображає стандартний життєвий цикл машинного навчання. Компоненти середовища організовані лінійно для забезпечення послідовного (крок за кроком) керівництва користувачем.

Основні проектні рішення включають:

• Концепція єдиного вікна - для мінімізації когнітивного навантаження та уникнення частого перемикання між вкладками, основний робочий простір залишається статичним, а перехід між етапами здійснюється шляхом вертикальної навігації.

• Використання панелі, що згортається, дозволяє раціонально використовувати екранний простір, розміщуючи інструменти препроцесингу та інженерії ознак окремо від зони візуалізації даних.

• Центральним елементом інтерфейсу є вбудована динамічна таблиця, яка дозволяє здійснювати фільтрацію, редагування та перегляд атрибутів набору даних у режимі реального часу.

• Функції журналювання (історія змін) та механізми відкоту реалізовані у вигляді спливаючих вікон, що запобігає захаращенню макету.

4.8.2. Архітектура бекенду та принцип модульності

Серверна частина системи спроектована як детермінована структура з високим ступенем роз'єднання компонентів. Логіка реалізована за допомогою об'єктно-орієнтованого підходу на мові Python, де кожен аспект робочого циклу ML інкапсульований у незалежний програмний модуль.

Така організація забезпечує:

1. Ізольовану обробку завдань: Зміни в одному модулі не призводять до дестабілізації всієї системи.

2. Масштабованість: Модульна архітектура спрощує інтеграцію нових алгоритмів у майбутніх ітераціях розробки.

3. Інтеграцію зі станом сесії: Забезпечується синхронізація між діями користувача в інтерфейсі та станом об'єктів у пам'яті сервера.

Кожен інструмент трансформації даних (імпутація пропущених значень, квантифікація категоріальних ознак, масштабування та ідентифікація аномалій) реалізований як автономний функціональний блок.

Механізм взаємодії базується на передачі параметрів від компонентів GUI до відповідних бекенд-функцій. Всі модифікації застосовуються до копії набору даних, збереженого в стані сесії. Важливою особливістю є система реєстрації станів: будь-яка зміна фіксується в журналі історії, що дозволяє реалізувати версійність даних і можливість повернення до попередніх етапів обробки без втрати цілісності вибірки.

Функціонал інженерії ознак надає досліднику інструменти для інтелектуального маніпулювання векторами ознак. Ключовим компонентом є алгоритм аналізу взаємної інформації.

• Обчислюється статистична значимість кожного вхідного атрибута відносно цільової змінної.

• Результати ранжуються, надаючи чітке уявлення про інформативну потужність ознак.

• Система підтримує створення деривативних ознак шляхом математичної взаємодії існуючих стовпців. Кожна операція миттєво відображається в основній таблиці та реєструється в системі контролю станів.

Процес моделювання ініціюється після фіналізації підготовки даних і включає наступні етапи:

1. Поділ вибірки на тренувальну та валідаційну підмножини із записом їхніх станів.

2. Надання доступу до гіперпараметрів моделі через інтерфейс користувача для проведення прецизійної настройки.

3. Після фази навчання система генерує аналітичний звіт, що включає числові та графічні метрики якості.

4. Кожен запуск симуляції реєструється, що дозволяє порівнювати різні конфігурації та зберігати параметри оптимальної моделі.

Лістинг 3.1. Реалізація модуля імпутації та управління станом

```
import pandas as pd
import streamlit as st
from sklearn.impute import SimpleImputer

# --- backend/preprocessing.py ---
class PreprocessingModule:
    """Модуль для виконання операцій очищення даних."""

    @staticmethod
    def impute_missing_values(df: pd.DataFrame, column: str, strategy: str) -> pd
        """
        Виконує заміну пропущених значень.
        strategy: 'mean', 'median', 'most_frequent' (mode)
        """
        data_copy = df.copy()
        imputer = SimpleImputer(strategy=strategy)
        # Перетворення колонки (Reshape для sklearn)
        data_copy[[column]] = imputer.fit_transform(data_copy[[column]])
        return data_copy

# --- main_interface.py ---
def render_imputation_ui():
    st.subheader("Налаштування імпутації")

    # Отримання списку колонок з пропущеними значеннями
    columns_with_nans = st.session_state.df.columns[st.session_state.df.isnull()

    if columns_with_nans:
        selected_col = st.selectbox("Оберіть колонку для обробки", columns_with_nan
        strategy = st.selectbox("Метод кодування", ["mean", "median", "most_freque

        if st.button("Застосувати зміни", key="apply_impute"):
            # Виклик незалежного модуля бекенду
            updated_df = PreprocessingModule.impute_missing_values(
                st.session_state.df, selected_col, strategy
            )
```

```
            # Оновлення стану сесії (Session State) та логування історії
            st.session_state.history.append(st.session_state.df.copy())
            st.session_state.df = updated_df

            st.success(f"Значення у колонці '{selected_col}' успішно оновлено!")
            st.rerun()
    else:
        st.info("Пропущених значень не виявлено.")
```

Метод impute_missing_values не залежить від Streamlit. Його можна тестувати окремо або використовувати в іншому середовищі. Модуль створює копію датафрейму (df.copy()), що дозволяє уникнути небажаних побічних ефектів (SettingWithCopyWarning). Перед оновленням поточного стану (st.session_state.df), попередня версія зберігається у списку history, що забезпечує роботу описаної раніше функції скасування. Використання st.rerun() гарантує, що інтерактивна таблиця (наприклад, AgGrid) миттєво відобразить оновлені дані.

Система містить розгалужену мережу перевірок валідності вхідних даних. У разі виникнення некоректних операцій (наприклад, запуск моделі без визначення цільової змінної), система активує протокол сповіщення, який у доступній формі інформує користувача про необхідність корекції дій, блокуючи подальший прогрес до вирішення проблеми.

4.9 Висновки по розділу

У четвертому розділі здійснено програмну реалізацію системи візуалізації та тестування моделей машинного навчання відповідно до визначених вимог і спроєктованої архітектури. Реалізовано функціонально завершений графічний інтерфейс користувача, що забезпечує інтуїтивну взаємодію з системою. Розроблено основні модулі обробки даних, включаючи інструменти імпутації, кодування ознак та аналізу залежностей. Запроваджено механізми вибору, налаштування та оцінювання моделей, що

дозволяють проводити комплексний аналіз їх ефективності. У результаті створено програмний продукт, який відповідає сучасним вимогам до навчальних систем і може бути ефективно використаний у освітньому процесі.

ВИСНОВКИ

У результаті виконання бакалаврської роботи було проведено дослідження предметної області, обґрунтовано вибір методів і засобів реалізації, а також спроєктовано й реалізовано програмну систему тестування моделей машинного навчання, орієнтовану на використання в освітньому процесі.

У першому розділі здійснено ґрунтовний аналіз сучасного стану використання технологій машинного навчання в освіті в умовах цифрової трансформації. Встановлено, що інтерактивні методи навчання є ключовим чинником підвищення ефективності засвоєння складних концепцій, зокрема у галузі машинного навчання. Обґрунтовано необхідність створення візуального навчального середовища, яке дозволяє поєднати теоретичні знання з практичними навичками через експериментування з моделями та даними. Проведений аналіз існуючих програмних рішень, таких як Orange Data Mining, MLflow та KNIME, засвідчив наявність функціонально потужних інструментів, які, однак, мають обмеження щодо адаптації до освітнього середовища, зокрема складність інтерфейсу, надлишковість функціоналу або відсутність інтерактивних навчальних сценаріїв. У результаті було сформульовано концептуальні переваги розроблюваної системи, зокрема орієнтацію на навчальні цілі, інтерактивність, візуалізацію та інклюзивність.

У другому розділі визначено алгоритмічне та математичне забезпечення системи. Обґрунтовано використання методу лінійної регресії як базового інструменту для навчального середовища завдяки його інтерпретованості, простоті реалізації та можливості наочного представлення результатів. Формалізовано математичну модель та застосовано метод найменших квадратів для оцінювання параметрів. Визначено систему метрик оцінювання якості моделей, що забезпечує комплексну перевірку їх

адекватності. Сформовано функціональні та нефункціональні вимоги до системи, які охоплюють аспекти обробки даних, масштабованості, продуктивності та зручності використання. Особливу увагу приділено оптимізації використання ресурсів та аналізу потенційних ризиків функціонування системи.

У третьому розділі виконано проєктування системи з використанням сучасних методів моделювання. Побудовано діаграми варіантів використання, що відображають взаємодію користувача із системою, а також діаграми активностей і послідовностей, які деталізують динамічні аспекти функціонування. Це дозволило формалізувати логіку роботи системи, визначити основні сценарії використання та забезпечити узгодженість між функціональними вимогами і реалізацією.

У четвертому розділі здійснено програмну реалізацію розробленої системи. Реалізовано архітектурну модель із розподілом на клієнтську та серверну частини, що забезпечує модульність, масштабованість і зручність подальшого розвитку. Розроблено графічний інтерфейс користувача з урахуванням принципів ергономіки та інтуїтивності, що сприяє ефективній взаємодії користувача із системою. Реалізовано модулі обробки даних, включаючи імпутацію пропущених значень, кодування категоріальних ознак, аналіз залежностей та синтез ознак. Запроваджено механізми вибору моделей, їх параметризації, а також модуль аналітичної верифікації, який дозволяє інтерпретувати результати моделювання та оцінювати якість побудованих моделей.

Таким чином, у ході виконання дипломної роботи досягнуто поставленої мети — розроблено програмну систему тестування моделей машинного навчання, яка поєднує інструменти обробки даних, побудови моделей та їх оцінювання в інтерактивному візуальному середовищі.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ersozlu Z., Taheri S., Koch I. A Review of Machine Learning Methods Used for Educational Data // Education and Information Technologies. – Cham: Springer Nature, 2024. – Vol. 29. – P. 22125–22145.
2. Munir H., Vogel B., Jacobsson A. Artificial Intelligence and Machine Learning Approaches in Digital Education: A Systematic Revision // Information. – Basel: MDPI, 2022. – Vol. 13(4). – P. 203.
3. Zhai X., Lu M. Machine Learning Applications in Educational Studies // Frontiers in Education. – Lausanne: Frontiers Media, 2023. – Vol. 8. – P. 1225802.
4. Korkmaz C., Correia A.-P. A Review of Research on Machine Learning in Educational Technology // Educational Media International. – London: Taylor & Francis, 2019. – Vol. 56(3). – P. 250–267.
5. Bishop C. Pattern Recognition and Machine Learning. – New York: Springer, 2006. – 738 p.
6. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. – New York: Springer, 2009. – 745 p.
7. Goodfellow I., Bengio Y., Courville A. Deep Learning. – Cambridge: MIT Press, 2016. – 800 p.
8. Mitchell T. Machine Learning. – New York: McGraw-Hill, 1997. – 432 p.
9. Murphy K. Machine Learning: A Probabilistic Perspective. – Cambridge: MIT Press, 2012. – 1104 p.
10. Shalev-Shwartz S., Ben-David S. Understanding Machine Learning: From Theory to Algorithms. – Cambridge: Cambridge University Press, 2014. – 410 p.
11. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. – Boston: Pearson, 2021. – 1152 p.

12. Domingos P. The Master Algorithm. – New York: Basic Books, 2015. – 352 p.

13. James G., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning. – New York: Springer, 2021. – 611 p.

14. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. – Sebastopol: O'Reilly Media, 2019. – 856 p.

15. Abadi M. et al. TensorFlow: A System for Large-Scale Machine Learning // OSDI Conference. – Savannah: USENIX, 2016. – P. 265–283.

16. Zaharia M. et al. MLflow: A Platform for Managing the Machine Learning Lifecycle // KDD Conference. – New York: ACM, 2018. – P. 1–4.

17. Berthold M. et al. KNIME: The Konstanz Information Miner // Data Analysis, Machine Learning and Applications. – Berlin: Springer, 2008. – P. 319–326.

18. Pedregosa F. et al. Scikit-learn: Machine Learning in Python // Journal of Machine Learning Research. – Cambridge: JMLR, 2011. – Vol. 12. – P. 2825–2830.

19. Breiman L. Random Forests // Machine Learning. – New York: Springer, 2001. – Vol. 45. – P. 5–32.

20. Cortes C., Vapnik V. Support-Vector Networks // Machine Learning. – New York: Springer, 1995. – Vol. 20. – P. 273–297.

21. Friedman J. Greedy Function Approximation: A Gradient Boosting Machine // Annals of Statistics. – Institute of Mathematical Statistics, 2001. – Vol. 29. – P. 1189–1232.

22. Kingma D., Ba J. Adam: A Method for Stochastic Optimization // ICLR Conference. – San Diego: OpenReview, 2015. – P. 1–15.

23. LeCun Y., Bengio Y., Hinton G. Deep Learning // Nature. – London: Nature Publishing Group, 2015. – Vol. 521. – P. 436–444.

24. Krizhevsky A., Sutskever I., Hinton G. ImageNet Classification with Deep Convolutional Neural Networks // NIPS Conference. – Lake Tahoe: MIT Press, 2012. – P. 1097–1105.

25. Vaswani A. et al. Attention Is All You Need // NIPS Conference. – Long Beach: MIT Press, 2017. – P. 5998–6008.

26. Hochreiter S., Schmidhuber J. Long Short-Term Memory // Neural Computation. – Cambridge: MIT Press, 1997. – Vol. 9. – P. 1735–1780.

27. Quinlan J. Induction of Decision Trees // Machine Learning. – New York: Springer, 1986. – Vol. 1. – P. 81–106.

28. Cover T., Hart P. Nearest Neighbor Pattern Classification // IEEE Transactions on Information Theory. – New York: IEEE, 1967. – Vol. 13. – P. 21–27.

29. MacQueen J. Some Methods for Classification and Analysis of Multivariate Observations // Proceedings of the Fifth Berkeley Symposium. – Berkeley: University of California Press, 1967. – P. 281–297.

30. Kohavi R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation // IJCAI Conference. – Montreal: Morgan Kaufmann, 1995. – P. 1137–1145.

31. Stone M. Cross-Validatory Choice and Assessment of Statistical Predictions // Journal of the Royal Statistical Society. – London: Wiley, 1974. – Vol. 36. – P. 111–147.

32. Dietterich T. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms // Neural Computation. – Cambridge: MIT Press, 1998. – Vol. 10. – P. 1895–1923.

33. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. – Amsterdam: Elsevier, 2011. – 703 p.

34. Witten I., Frank E., Hall M. Data Mining: Practical Machine Learning Tools and Techniques. – Burlington: Morgan Kaufmann, 2016. – 654 p.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: “Розробка програмної системи тестування моделей машинного навчання ”

Обсяг пояснювальної записки: 81 аркуш.

Дата закінчення роботи: 9 червня 2026 р.

Підпис студента _____