Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів попередньої обробки даних
для задач штучного інтелекту
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-23-1
Микита Демусенко
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва освітньої програми)
Керівник доц. Лариса Чала
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Демусенку Микиті Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів попередньої обробки даних для задач штучного інтелекту

затверджена наказом університету від 21 квітня 20 25 р. № 295Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 червня 20 25 р.

3. Вихідні дані до роботи науково-технічні публікації та дані різних печатних та інтернет-джерел щодо тематики кваліфікаційної роботи

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі та постановка задачі

2) Теоретичні дослідження

3) Практична реалізація

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	21.04.2025	виконано
2	Аналіз актуальності теми та наукових джерел	22.04.2025 – 25.04.2025	виконано
3	Формулювання задачі дослідження	26.04.2025 – 01.05.2025	виконано
4	Оцінка результатів і формулювання висновків	01.05.2025 – 10.05.2025	виконано
5	Підготовка пояснювальної записки та презентації	10.05.2025 – 10.06.2025	виконано
6	Захист перед ЕК	10.06.2025	

Дата видачі завдання 21 квітня 20 25 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Лариса Чала
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 94 с., 3 рис., 1 табл., 1 дод., 46 джерел.

АНОМАЛІЇ, БАЛАНСУВАННЯ КЛАСІВ, ЕКСПЕРТНІ ПРАВИЛА, ЗНИЖЕННЯ РОЗМІРНОСТІ, МАШИННЕ НАВЧАННЯ, НОРМАЛІЗАЦІЯ, ПОПЕРЕДНЯ ОБРОБКА ДАНИХ, ПРОПУЩЕНІ ЗНАЧЕННЯ, РЕКОМЕНДАЦІЙНА СИСТЕМА, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єкт дослідження: процес підготовки даних для систем штучного інтелекту.

Предмет дослідження: методи попередньої обробки даних і підходи до формування рекомендацій щодо їх використання в задачах ШІ.

Мета роботи: дослідження сучасних методів попередньої обробки даних для задач штучного інтелекту та розробка інструменту, що формує рекомендації щодо вибору методів попередньої обробки на основі аналізу характеристик датасету.

Методи дослідження: аналіз літературних джерел, класифікація методів обробки даних, експертні евристики, розробка прототипу рекомендаційної системи, тестування на прикладах датасетів.

Кваліфікаційна робота присвячена систематизації сучасних методів попередньої обробки даних та побудові системи, яка на основі характеристик вхідного набору даних надає рекомендації щодо доцільних методів обробки. Реалізована система аналізує такі аспекти, як наявність пропущених значень, викидів, дисбалансу класів, масштабні відмінності, тип змінних тощо.

Результатом роботи є прототип програмного інструменту, що може бути використаний як допоміжний модуль при підготовці даних у проєктах із машинного навчання або як основа для подальшого розвитку інтелектуальних підсистем попередньої обробки.

ABSTRACT

Master’s thesis contains: 94 pp., 3 fig., 1 tabl., 1 ann., 46 references.

ARTIFICIAL INTELLIGENCE, CLASS BALANCE, DATA PREPROCESSING, DIMENSIONALITY REDUCTION, EXPERT RULES, MACHINE LEARNING, MISSING VALUES, NORMALIZATION, OUTLIERS, RECOMMENDATION SYSTEM.

Object of study: the process of data preparation for artificial intelligence systems.

Subject of study: methods of data preprocessing and approaches to forming recommendations for their application in AI tasks.

Purpose of the work: to explore modern methods of data preprocessing for artificial intelligence tasks and to develop a tool that provides recommendations on preprocessing techniques based on dataset characteristics.

Research methods: literature analysis, classification of preprocessing methods, expert heuristics, prototype development of a recommendation system, testing on example datasets.

The qualification work is devoted to the systematization of modern data preprocessing methods and the development of a system that provides recommendations on relevant preprocessing techniques based on dataset analysis. The implemented system analyzes such aspects as missing values, outliers, class imbalance, feature scaling, and variable types.

The result of the work is a prototype of a software tool that can be used as an auxiliary module for data preparation in machine learning projects or as a foundation for further development of intelligent preprocessing subsystems. The proposed approach improves the efficiency and consistency of decision-making at the early stages of AI system development.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence – штучний інтелект;

MICE – Multivariate Imputation by Chained Equations – багатоваріантна імпутація «ланцюговими» моделями;

ML – Machine Learning – машинне навчання;

PCA – Principal Component Analysis – метод головних компонент (зниження розмірності);

SMOTE – Synthetic Minority Oversampling Technique – синтетичне збільшення міноритарного класу;

UMAP – Uniform Manifold Approximation and Projection – нелінійний метод зниження розмірності;

VAE – Variational Autoencoder – варіаційний автоенкодер.

ВСТУП

Штучний інтелект (ШІ) – одна з найдинамічніших галузей сучасних технологій, що широко впроваджується в медицині, фінансах, промисловості, транспорті та багатьох інших сферах. Ефективність ШІ-моделей значною мірою залежить не лише від архітектури алгоритмів, а й від якості даних, які подаються на вхід. Саме тому попередня обробка даних розглядається як критично важливий етап у створенні будь-якої інтелектуальної системи.

У практиці машинного навчання дані рідко надходять у готовому до аналізу вигляді. Часто спостерігаються пропущені значення, викиди, дисбаланс класів, масштабні відмінності, змішані типи змінних або надмірна кількість ознак. Використання таких «сирих» даних без попередньої обробки призводить до нестабільної поведінки моделей, перенавчання, упереджених рішень або зниження точності. Якісна попередня обробка даних дозволяє забезпечити узгодженість, зменшити вплив шуму, нормалізувати масштаб ознак і сформувати надійну базу для навчання моделей.

Метою цієї кваліфікаційної роботи є дослідження та систематизація сучасних методів попередньої обробки даних для задач штучного інтелекту, а також розробка інструменту, що формує рекомендації щодо вибору підходів до обробки на основі аналізу характеристик датасету. Такий інструмент дозволяє інтегрувати знання про методи попередньої обробки в процес підготовки даних, що є складовою частиною системи штучного інтелекту.

Об’єкт дослідження – процес підготовки даних у системах ШІ. Предмет дослідження – методи попередньої обробки даних, зокрема обробка пропусків, масштабування, нормалізація, перетворення категоріальних змінних, виявлення аномалій, зниження розмірності та балансування класів.

Практична частина роботи передбачає реалізацію прототипу інструменту, який на основі базових евристичних правил і автоматичного аналізу структури датасету пропонує відповідні методи обробки.

Результати дослідження можуть бути використані для вдосконалення процесів підготовки даних у практичних проєктах машинного навчання та як основа для побудови інтелектуальних підсистем попередньої обробки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Вплив попередньої обробки даних у задачах ШІ

Попередня обробка даних є фундаментальним етапом розробки інтелектуальних систем, який безпосередньо впливає на точність, стабільність і ефективність моделей машинного навчання. Незалежно від архітектури моделі – від простих статистичних алгоритмів до глибоких нейронних мереж – якість навчання суттєво залежить від стану вхідних даних [1].

У реальних умовах вихідні дані зазвичай характеризуються наявністю пропусків, шуму, аномалій, неоднорідною шкалою ознак або дисбалансом класів. Без відповідної обробки ці проблеми можуть призвести до деградації продуктивності моделі, втрати здатності до узагальнення, а в деяких випадках – до появи систематичних упереджень (наприклад, у задачах виявлення шахрайства або хвороб) [2].

Попередня обробка виконує низку важливих функцій: вона забезпечує узгодженість формату даних, дозволяє зменшити вплив випадкових факторів, покращує стабільність навчального процесу та сприяє інтерпретованості результатів. Крім того, вона підвищує ефективність подальших етапів, таких як вибір ознак, налаштування моделі та оцінювання результатів [3].

Ці функції можна умовно класифікувати за призначенням:

– забезпечення формальної сумісності даних із вимогами алгоритмів;

– зменшення кількості шуму та відхилень;

– підвищення однорідності масштабів та форматів ознак;

– збереження інформаційної цінності даних при зменшенні їх складності;

– формування основи для якісного моделювання.

Історично процес попередньої обробки не завжди розглядався як самостійний напрям. У перших етапах розвитку систем ШІ йому приділяли обмежену увагу. Однак із накопиченням емпіричних спостережень було зафіксовано, що саме стан даних часто є основною причиною помилок моделей. Це призвело до формування окремого підходу до data preprocessing, який отримав розвиток у вигляді бібліотек (Scikit-learn, TensorFlow Transform, DataWrangler тощо) і став невіддільною частиною будь-якого машинного workflow.

Дослідження, проведені в останнє десятиліття [2], [3], [4], підтверджують, що грамотно побудований процес попередньої обробки даних може суттєво покращити результати моделювання – іноді навіть більше, ніж заміна моделі на складнішу. Такий висновок підтверджується як статистичними експериментами, так і практичними кейсами в галузі медицини, фінансів та комп'ютерного зору. Таким чином, попередня обробка не є допоміжною процедурою, а виступає повноцінною складовою розробки систем штучного інтелекту, що потребує науково обґрунтованого підходу.

1.2 Типи даних, що використовуються в системах ШІ

У системах штучного інтелекту використовуються різнорідні типи даних, які відрізняються за джерелом походження, внутрішньою структурою, обсягом та способом представлення. Коректна ідентифікація типу даних є критичною для вибору відповідних методів попередньої обробки, оскільки різні формати потребують принципово різних підходів до очищення, трансформації та узгодження [1].

Один із найпоширеніших типів – структуровані дані. Вони мають чітко визначену схему і зазвичай представлені у вигляді таблиць із фіксованими ознаками. До них належать фінансові таблиці, медичні обстеження, результати опитувань, дані телеметрії. Для таких даних

найбільш релевантними є методи обробки пропусків, нормалізація числових ознак, кодування категоріальних змінних і виявлення аномалій [2].

Напівструктуровані дані частково організовані, однак не мають суворої схеми. Це можуть бути XML-файли, JSON-відповіді API, журнали подій або HTML-сторінки. Такі дані потребують попереднього вилучення (парсингу), фільтрації та приведення до структурованого вигляду перед подальшим аналізом [4].

Неструктуровані дані, як правило, складаються з інформації, яка не піддається прямому табличному поданню: текст, зображення, відео, аудіо, графові структури тощо. Для кожного з цих підтипів застосовуються окремі підходи: токенізація, лематизація, embedding-моделі для текстів; обробка піксельних масивів для зображень; спектральний аналіз для аудіо; виділення ознак із графових представлень [5].

У багатьох прикладних задачах використовуються гібридні типи даних – наприклад, медичні або юридичні системи можуть містити одночасно числові вимірювання, текстові висновки, результати візуалізацій і часові ряди. У таких випадках попередня обробка має враховувати специфіку кожної компоненти окремо, а також забезпечувати узгоджене злиття даних для подальшої обробки.

Розуміння типів даних є невід’ємною частиною підготовчого етапу в розробці систем ШІ. Воно визначає як вибір конкретних методів preprocessing, так і побудову загальної архітектури обробки інформації у всьому життєвому циклі даних.

1.3 Основні задачі попередньої обробки даних

Попередня обробка даних охоплює широкий спектр задач, що мають на меті усунення типових проблем «сирих» даних та приведення їх до форми, придатної для подальшого використання в алгоритмах машинного навчання. Конкретні методи і послідовність їх застосування залежать від

характеру вхідних даних, вимог до моделі, а також цільової задачі. У цьому підрозділі описано ключові напрямки, які найчастіше реалізуються на цьому етапі.

Одним із перших кроків часто є обробка пропущених значень. Пропуски можуть виникати через помилки збору, неповні анкети, збої датчиків тощо. Існують кілька підходів до їхнього усунення: видалення записів із пропусками (за умови незначного обсягу втрат), заміна на статистичні оцінки (середнє, медіану, моду), або використання регресійних і класифікаційних моделей для їх передбачення. Іноді застосовується спеціальне маркування пропусків як окремого категоріального значення.

Наступною поширеною задачею є масштабування числових ознак. Багато алгоритмів (зокрема, k-ближчих сусідів, SVM, нейронні мережі) демонструють кращу стабільність і збіжність за умови приведення ознак до однакового масштабу. Найбільш відомі підходи включають мін-макс нормалізацію, Z-нормалізацію (стандартизацію), а також робастні методи, що базуються на медіані та міжквартильному розмахові [1].

Для категоріальних ознак, які не мають числової природи, потрібна спеціальна трансформація. Найбільш поширені варіанти: бінарне кодування (one-hot), порядкове кодування (ordinal), або більш складні підходи на кшталт target encoding, де враховується залежність від цільової змінної [5].

Виявлення аномалій та викидів також є невід’ємною частиною процесу. Викиди можуть істотно впливати на параметри моделей, особливо чутливих до відстаней і дисперсії. Використовуються як прості статистичні методи (наприклад, Z-оцінка, межі міжквартильного розмаху), так і алгоритмічні рішення на зразок Isolation Forest або Local Outlier Factor [4].

Коли кількість ознак є надмірною, виникає потреба у зниженні розмірності. Цей етап дозволяє зменшити складність моделі, усунути корельовані або нерелевантні ознаки, а також покращити візуалізацію даних. Серед популярних методів – головні компоненти (PCA), лінійний

дискримінантний аналіз (LDA), autoencoder-мережі та t-SNE для візуалізацій.

Дисбаланс класів – поширена проблема в задачах класифікації, коли один клас представлений суттєво частіше за інші. Її вирішенням є застосування методів над- та недосемплінгу (oversampling, undersampling), зокрема таких як SMOTE або ADASYN. Альтернативою є використання зважених функцій втрат або генеративних підходів.

Кожна з перелічених задач не є ізольованою і зазвичай реалізується в комбінації з іншими в рамках єдиного pipeline. Їх послідовне застосування формує основу якісного підходу до підготовки даних для моделей штучного інтелекту.

1.4 Вплив якості даних на продуктивність роботи моделей ШІ

Якість вхідних даних є визначальним чинником ефективності систем штучного інтелекту. Незалежно від складності моделі – чи йдеться про просту логістичну регресію, чи про сучасні глибокі нейронні мережі – модель не може компенсувати недоліки, що закладені у неякісному датасеті. Як відзначають дослідники [1], [2], [4], саме стан даних часто стає основною причиною неузгодженості прогнозів, зниження точності або нестабільної роботи системи в реальному середовищі.

Проблеми з якістю даних можуть мати різну природу: наявність пропусків, викидів, шуму; дисбаланс класів; неоднорідність масштабів; надмірна розмірність або корельованість ознак. Вони безпосередньо впливають на результативність моделей, зокрема можуть знижувати точність класифікації або регресії, зменшувати здатність моделі до узагальнення, спричиняти нестабільність при повторному навчанні, посилювати упередженість у прогнозах і ускладнювати процес валідації та інтерпретації результатів.

Наприклад, у сфері медичної діагностики, як зазначено в роботі [4], неповнота або некоректність даних про симптоми пацієнтів може призвести до формування помилкових рекомендацій навіть за наявності високоточних моделей. У фінансовому аналізі системи виявлення шахрайства демонструють значне зниження ефективності у разі наявності хибних, дубльованих або невідмасштабованих транзакційних записів.

Дослідження Jerez et al. [6] продемонструвало, що спосіб обробки пропусків значно впливає на точність класифікації при аналізі онкологічних даних. Видалення записів з пропущеними значеннями призводило до зниження точності на понад 20%, тоді як використання алгоритмів для передбачення пропущених значень дозволяло зберегти або навіть покращити результат.

Ще один приклад – вплив масштабу ознак на поведінку моделей. Згідно з [4], використання нефіксованих масштабів у класифікаторі SVM призводить до домінування ознак із більшими числовими значеннями, що унеможливлює адекватне врахування інших параметрів.

Таким чином, якість даних не є лише технічною умовою попереднього етапу, а виступає визначальним елементом усієї аналітичної системи. Грамотна підготовка датасету дозволяє досягти високих результатів навіть за допомогою простих моделей і зменшує потребу в обчислювально складних архітектурах. У контексті розробки ШІ-систем обробка даних повинна розглядатися як повноцінна дослідницька задача, що потребує системного підходу.

1.5 Практичні приклади з помилками через погану підготовку даних

Незадовільна підготовка даних може призвести до серйозних помилок у роботі інтелектуальних систем, особливо коли йдеться про автоматизовані рішення у сферах охорони здоров’я, безпеки чи фінансів. У цьому підрозділі

розглянуто низку реальних прикладів, які ілюструють важливість якісної попередньої обробки.

Один із яскравих кейсів стосується сфери медичної діагностики. У дослідженні Obermeyer et al. (2019) було виявлено, що одна з широко використовуваних систем прийняття рішень демонструвала расову упередженість. Причиною стала помилкова метрика – замість реальної потреби в медичній допомозі система використовувала попередні витрати на лікування як проксі-змінну. Через нерівний доступ до медицини пацієнти з певних соціальних груп витрачали менше, що призводило до систематичного недооцінювання ризиків [7].

У задачах виявлення шахрайства (fraud detection) надзвичайно поширеним є дисбаланс класів. Якщо набір даних не проходить попереднє балансування (наприклад, за допомогою SMOTE або методів undersampling), модель може навчитися ігнорувати рідкісні, але критично важливі шахрайські транзакції. Відомі приклади, коли ігнорування цього етапу призводило до значних фінансових втрат [8].

Інший випадок стосується задач комп'ютерного зору. У проєкті з класифікації зображень тварин було виявлено, що модель навчається не на суттєвих візуальних характеристиках, а на технічних деталях знімків – зокрема, логотипах камер. Це явище отримало назву «shortcut learning» – коли система використовує супутні, але неінформативні ознаки. Причиною стала відсутність достатньої перевірки узгодженості і варіативності даних [9].

У сфері аналізу текстів виявлено ще одну проблему – упередження, закладене у вхідні дані. Дослідження показали, що моделі sentiment analysis іноді асоціюють негативну тональність із певними іменами, професіями або термінами, що не мають прямого відношення до контексту. Це є наслідком відсутності попереднього очищення даних і небажаного навчання на соціальних шаблонах [10].

Наведені приклади підтверджують: навіть найсучасніші моделі не можуть забезпечити коректної роботи за умов поганої якості даних. Надійність, етичність і безпечність інтелектуальних систем значною мірою визначаються не тільки моделлю, а й глибиною та якістю попередньої обробки даних.

1.6 Класифікація методів попередньої обробки

Методи попередньої обробки даних охоплюють широкий спектр прийомів, що відрізняються за своїм призначенням, алгоритмічною природою та місцем у структурі аналітичного процесу. З огляду на це, класифікація таких методів є важливим кроком для побудови системного уявлення про галузь та для формалізації процесу вибору підходів у конкретних прикладних задачах.

Один із найпоширеніших підходів до класифікації базується на функціональному призначенні методів. У цьому контексті виділяють кілька основних груп.

Методи очищення включають дії з усунення шуму, пропусків, дублювань і аномалій. Це базові кроки, необхідні для забезпечення коректності й повноти вхідних даних.

Методи нормалізації та масштабування відповідають за уніфікацію числових масштабів, наприклад, через стандартизацію або мін-макс перетворення. Вони важливі для алгоритмів, чутливих до відстаней або розподілу значень.

До методів трансформації ознак належать підходи до кодування категоріальних змінних, log-преобразування, а також створення нових ознак або розширення існуючих.

Методи зниження розмірності застосовуються для зменшення кількості ознак без втрати ключової інформації. Сюди входять як алгоритмічні методи (наприклад, PCA), так і методи відбору ознак.

Окрему категорію становлять методи балансування, які вирівнюють представлення класів у задачах класифікації. Це може бути як надсемплінг, так і генерація нових прикладів меншості.

Інший підхід до класифікації ґрунтується на тому, чи потребує метод знань про модель. У цьому випадку виділяють методи, незалежні від моделі (model-independent), які застосовуються без урахування цільової змінної (наприклад, нормалізація, one-hot кодування), а також методи, залежні від моделі (model-dependent), які враховують специфіку задачі або цільової змінної (наприклад, target encoding або supervised feature selection) [4], [5].

Також важливою є класифікація за типом даних, до яких застосовується метод. Наприклад, для текстових даних будуть доречними токенізація, очищення від стоп-слів, побудова векторних представлень; для зображень – вирівнювання розміру, нормалізація пікселів, аугментація; для графових структур – побудова матриць суміжності, агрегація сусідів тощо [4].

Наведені класифікації не є взаємовиключними, а радше взаємодоповнюючими. У складних реальних сценаріях часто комбінуються різні підходи відповідно до вимог завдання, типу даних і обраного алгоритму. Систематизація методів попередньої обробки дозволяє будувати узгоджені та ефективні процеси підготовки даних у рамках розробки штучного інтелекту.

1.7 Роль попередньої обробки в циклі побудови систем ШІ

Попередня обробка даних займає ключове місце в життєвому циклі розробки систем штучного інтелекту. У типових процесах побудови таких систем, зокрема в рамках моделей CRISP-DM (Cross-Industry Standard Process for Data Mining) або KDD (Knowledge Discovery in Databases),

попередня обробка є обов’язковим етапом, що передує моделюванню та безпосередньо впливає на якість результатів.

Життєвий цикл побудови ШІ-систем можна умовно поділити на кілька етапів: збирання даних, попередній аналіз, очищення та обробка, побудова моделі, її валідація та впровадження. Попередня обробка охоплює проміжну фазу між збиранням «сирих» даних та етапом навчання моделей і включає такі дії, як виявлення пропусків, нормалізація, кодування, видалення аномалій, агрегація ознак, балансування класів тощо [11].

Цей етап виконує кілька функцій. По-перше, він зменшує кількість технічного шуму, що присутній у сирих даних. По-друге, він забезпечує відповідність вхідного формату даних вимогам конкретного алгоритму машинного навчання. По-третє, він дозволяє сформувати більш узагальнену та стабільну структуру даних, з якою модель працює ефективніше. І нарешті, передобробка може включати елементи доменної експертизи, що сприяє підвищенню інтерпретованості моделей і результатів [2].

Крім того, попередня обробка є важливим етапом у побудові pipeline-архітектур, які широко використовуються в сучасному ML-інжинірингу. За допомогою бібліотек на кшталт Scikit-learn Pipeline, TensorFlow Transform або Apache Beam створюються послідовності обробки, де попередня обробка інтегрується в автоматизований процес навчання, валідації та розгортання моделей. Таким чином, він перестає бути одноразовим кроком і перетворюється на повторювану складову циклу життєдіяльності моделі.

Отже, попередня обробка даних є не лише підготовчим етапом, а й критично важливою частиною побудови життєздатних, масштабованих і стійких до помилок систем штучного інтелекту. Її роль полягає в забезпеченні якісної основи для всіх подальших етапів моделювання та прийняття рішень.

1.8 Постановка задачі дослідження

Попередня обробка даних є критично важливою складовою будь-якого проєкту, пов’язаного з інтелектуальною обробкою даних. У цьому контексті актуальним є дослідження методів попередньої обробки та їх впливу на результативність ШІ-систем.

У межах цієї кваліфікаційної роботи ставиться мета: дослідити, систематизувати та оцінити ефективність методів попередньої обробки даних, а також розробити інструмент для рекомендації відповідних методів на основі характеристик вхідного датасету.

Для досягнення цієї мети сформульовано такі основні завдання дослідження:

– провести аналітичний огляд літератури з тематики попередньої обробки даних у ШІ;

– визначити та класифікувати основні типи задач попередньої обробки (очищення, нормалізація, трансформація ознак, зниження розмірності, балансування);

– розробити структуру системи, що здійснює автоматичний аналіз характеристик вхідного набору даних;

– сформулювати правила (евристики), що дозволяють надавати рекомендації щодо доцільних методів обробки;

– реалізувати прототип системи та протестувати її на прикладах датасетів та проаналізувати ефективність рекомендацій, надати висновки щодо їх застосовності в практичних задачах.

Таким чином, задача дослідження передбачає як теоретичне опрацювання проблеми, так і практичну реалізацію програмного інструменту, що базується на результатах аналізу. Запропоноване рішення має сприяти підвищенню якості та ефективності розробки ШІ-систем за рахунок системного підходу до вибору методів попередньої обробки.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

2.1 Обробка пропущених значень

Пропущені значення (англ. missing values) – це ситуації, коли в одному або кількох спостереженнях відсутні значення певних змінних. Незалежно від джерела даних – будь то анкети, сенсори, бази транзакцій чи системи реєстрації подій – дані майже ніколи не є повними. Проблема відсутніх значень є фундаментальною в задачах машинного навчання та аналітики, оскільки більшість алгоритмів очікує на повні та коректно заповнені набори ознак.

Причини появи пропущених значень можуть бути різноманітними. Серед них – технічні збої при зборі даних (наприклад, несправність сенсора або помилка при зчитуванні), людський фактор (коли користувачі не заповнили всі поля в анкеті або оператор ввів не повний обсяг інформації), неузгодженість форматів під час об’єднання кількох джерел (наприклад, різні одиниці виміру або структура таблиць), свідоме приховування інформації (особливо коли йдеться про чутливі параметри, такі як дохід, вік чи стан здоров’я), а також помилки передачі або зберігання даних, що можуть призвести до втрати або пошкодження інформації.

Для моделей машинного навчання пропущені значення становлять суттєву загрозу. Більшість алгоритмів не можуть напряму працювати з NaN або null-значеннями, а навіть ті, які допускають пропуски (наприклад, деякі реалізації дерев рішень), можуть неправильно їх інтерпретувати або втрачати продуктивність. Ігнорування проблеми пропусків може призвести до низки негативних наслідків, зокрема до зміщення розподілу ознак, втрати взаємозв’язків між змінними, погіршення збіжності моделей, зниження точності або зростання похибки на тестових вибірках. Також моделі можуть стати надто чутливими до шуму або непередбачуваних ситуацій у реальних умовах застосування.

Особливе значення має те, що спосіб обробки пропущених значень може радикально змінити результати моделювання. Наприклад, просте видалення записів із відсутніми значеннями знижує обсяг навчальної вибірки та призводить до втрати інформації. Натомість невдале заповнення пропусків може ввести модель в оману, зберігаючи хибну структуру залежностей. Саме тому вибір стратегії обробки має базуватись не лише на технічних аспектах, а й на аналізі природи пропусків, типу змінних і цільової задачі.

Таким чином, розуміння пропущених значень, їх причин і наслідків, а також обґрунтований вибір методів їх обробки є обов’язковим етапом при побудові надійних і точних моделей штучного інтелекту.

2.1.1 Типи пропущених значень

Перед вибором методів обробки пропусків необхідно з’ясувати їхню природу. У статистичному аналізі та машинному навчанні тип пропущених значень істотно впливає на коректність результатів і визначає допустимі підходи до імпутації. Згідно з класичною класифікацією Rubin [12], існують три основні механізми виникнення пропусків: MCAR, MAR і MNAR.

Пропуски типу MCAR (Missing Completely At Random).

У цьому випадку ймовірність пропуску не залежить ані від спостережуваних, ані від неспостережуваних даних. Формально, якщо M – індикатор наявності пропуску, Y – змінна з пропусками, а X – набір усіх інших змінних, то виконується формула 2.1.

$$P(M = 1|X, Y) = P(M = 1). \quad (2.1)$$

Це означає, що відсутність значень є повністю випадковою і не пов’язана з іншими даними в наборі. У такому разі виключення записів із пропусками не призводить до упередженості. MCAR вважається найменш

проблематичним типом пропусків, але зустрічається на практиці досить рідко [13].

Пропуски типу MAR (Missing At Random).

У цьому сценарії ймовірність пропуску залежить лише від інших спостережуваних змінних, але не від самої змінної з пропуском:

$$P(M = 1|X, Y) = P(M = 1|X). \quad (2.2)$$

Це найпоширеніший механізм у прикладних задачах. Наприклад, відсутність відповіді про рівень доходу може обумовлюватися віком або освітою, які заповнені. Оскільки інформація про пропуски міститься в наявних ознаках, статистичні та модельні методи імпутації (MICE, регресійна імпутація) працюють адекватно [14].

Пропуски типу MNAR (Missing Not At Random).

У цьому випадку ймовірність пропуску залежить від значення самої змінної, яка пркаопущена і виконується формула 2.3:

$$P(M = 1|X, Y) \text{ залежить від } Y. \quad (2.3)$$

Це найскладніший випадок, бо причини пропуску приховані у відсутній інформації. Наприклад, респонденти з високою тривожністю можуть ігнорувати запитання про психічний стан. Класичні методи імпутації дають викривлення; потрібні спеціалізовані моделі з явним описом механізму пропусків або зовнішні джерела даних [15].

На практиці точну природу пропусків визначити складно, оскільки для цього потрібен доступ до неспостережуваних змінних. Проте існують підходи до емпіричного тестування гіпотез, наприклад, тест Little’s MCAR test, що дозволяє перевірити припущення про повну випадковість пропусків.

У багатьох випадках аналізується логіка генерації даних та здійснюється експертна оцінка механізму втрат.

Правильна ідентифікація типу пропусків є ключовою передумовою для вибору релевантного методу обробки, що мінімізує втрати інформації та зберігає статистичну достовірність висновків.

2.1.2 Методи обробки пропущених значень

Видалення пропусків.

Найпростішим підходом до обробки відсутніх значень є їх ігнорування, тобто видалення записів або ознак, які містять пропуски. Така стратегія не вимагає складних алгоритмів і є реалізованою в усіх інструментах для аналізу даних. Проте її застосовність обмежена специфічними умовами, а наслідки можуть суттєво впливати на результати моделювання.

Поширеною реалізацією цього підходу є повне видалення записів, які містять хоча б одне пропущене значення. Цей метод також відомий як listwise deletion. Якщо позначити $X \in \mathbb{R}^{n \times d}$ – початкову матрицю ознак з n записів і d змінних, то після застосування listwise deletion утворюється матриця $X' \subset X$, у якій усі записи мають повні значення. Метод є допустимим тільки за припущенням MCAR [16], коли пропуски виникають повністю випадково. Якщо це припущення не виконується, видалення може призвести до зміщення оцінок і втрати репрезентативності.

Основні переваги такого підходу полягають у простоті реалізації, збереженні структури змінних без імпутацій та узгодженості з класичними статистичними методами. Водночас метод має суттєві недоліки: значна втрата інформації при великій кількості пропусків, ризик викривлення вибірки та неефективність при великих наборах із рідкісними, але важливими значеннями.

Ще одним варіантом є видалення цілої змінної (ознаки), якщо частка пропусків у ній перевищує певний поріг. Такий підхід зазвичай використовується у випадках, коли змінна має низьку інформативність або не є ключовою для цільової задачі. При цьому вибір порогу часто ґрунтується на емпіричних правилах: наприклад, якщо понад 30 % значень у змінній відсутні, її видаляють. Звісно, це число є умовним і залежить від розміру набору, важливості змінної та типу даних.

Загалом видалення пропусків може бути виправданим при невеликій кількості втрат, особливо в задачах із великим обсягом даних. Цей підхід також часто використовується як базовий для порівняння з результатами більш складних імпутаційних методів. Водночас у задачах зі складною структурою залежностей, обмеженим обсягом даних або ймовірною наявністю MAR чи MNAR пропусків, застосування цього методу є недоцільним.

Таким чином, хоча видалення пропусків є інтуїтивно простим і технічно зручним, воно повинне застосовуватись обережно – з урахуванням механізму виникнення втрат і аналізом потенційних втрат інформації, що можуть негативно вплинути на результати побудови моделі.

Заповнення статистичними оцінками.

Одним із найпоширеніших підходів до обробки пропущених значень є їх заповнення (імпутація) за допомогою простих статистичних характеристик. Цей метод є інтуїтивно зрозумілим, легко реалізується та часто використовується як базовий орієнтир для порівняння з більш складними підходами. Проте він має обмеження з точки зору збереження структури даних і достовірності статистичних висновків.

У найпростішому варіанті числові змінні заповнюються середнім значенням (mean imputation) [17]. Формально, якщо змінна x має n значень, з яких m відсутні, то кожне відсутнє значення x_i замінюється на емпіричне середнє:

$$\hat{x}_i = \frac{1}{n-m} \sum_{j \in \Omega} x_j, \quad (2.4)$$

де $\boldsymbol{\Omega}$ – множина індексів ненульових спостережень. Для змінних, що мають асиметричний розподіл або схильність до викидів, краще використовувати медіану, яка є надійною альтернативою.

У випадку категоріальних змінних застосовується заміна на моду – найбільш частотне значення в колонці. Така підстановка зберігає тип змінної, однак може порушити розподіл частот і призвести до штучного зміщення в сторону домінантного класу.

Основними перевагами цього підходу є його простота, швидкість виконання та відсутність необхідності будувати додаткові моделі. Він добре працює у випадках, коли частка пропусків невелика, а змінні мають близький до нормального розподіл.

Проте статистичне заповнення має суттєві недоліки. По-перше, воно занижує дисперсію ознаки, оскільки вставлені значення не додають варіативності. По-друге, метод не враховує взаємозв’язки між ознаками, що особливо критично в умовах наявності залежностей між змінними. По-третє, повторюваність значень може спричинити появу кластерів навколо імпутованого значення, що впливає на навчання моделей, чутливих до розподілу (наприклад, дерев рішень або алгоритмів кластеризації).

Таким чином, заповнення статистичними оцінками доцільне лише в обмежених випадках: коли пропуски є незначними за обсягом, коли структура даних не потребує високої точності або коли метод використовується як орієнтир у процесі дослідження. Для більш складних сценаріїв доцільно застосовувати методи, що враховують залежності між змінними.

Множинна імпутація (англ. Multiple Imputation) є потужним статистичним підходом до заповнення пропущених значень, який дозволяє враховувати невизначеність, пов’язану з відсутніми даними. Метод полягає

не в одиничному заповненні пропусків, а в побудові кількох можливих варіантів заповнення, з подальшим об’єднанням результатів.

Основна ідея полягає в тому, щоб побудувати кілька (наприклад, m=5) повних наборів даних шляхом імітації значень, які бракують. Далі на кожному з імпутованих наборів проводиться окремий аналіз, наприклад навчання моделі. Після цього результати об'єднуються з урахуванням дисперсії як у межах кожної імпутації, так і між ними, що дозволяє врахувати невизначеність, притаманну пропущеним даним.

Метод уперше формалізував Дональд Рубін [18], цей метод є особливо ефективним при виконанні припущення MAR. Ключова відмінність від одноразової імпутації – збереження варіативності даних, що дозволяє уникнути заниження дисперсії та переоцінки точності моделей.

Сучасною реалізацією множинної імпутації є підхід MICE (Multivariate Imputation by Chained Equations) [19]. Його суть полягає у побудові умовних моделей для кожної змінної з пропусками, де імпутація відбувається по черзі (ітеративно). Для кожної змінної X_j, яка містить пропуски, будується модель:

$$X_j \sim f_j(X_{-j}), \tag{2.5}$$

де X_{-j} – усі інші змінні без X_j. Процес повторюється у кілька циклів до стабілізації імпутованих значень.

MICE дозволяє враховувати нелінійні залежності, взаємозв’язки між змінними та використовувати моделі, що найкраще пасують до типу ознаки (лінійна регресія, логістична, дерева тощо). Залежно від реалізації (наприклад, у R або Python), імпутація може підтримувати числові, бінарні, категоріальні змінні.

Множинна імпутація демонструє високу точність і стійкість у порівнянні з простими методами, проте має і певні обмеження, серед яких висока обчислювальна складність, потреба в налаштуванні

гіперпараметрів (кількість ітерацій, число імпутацій), ризик переобчислення, якщо припущення MAR не дотримуються та складність інтерпретації при великій кількості імпутованих моделей.

Попри це, метод є золотим стандартом у статистичному аналізі, особливо в задачах, де важлива коректність оцінок і збереження варіативності даних. Його застосування рекомендоване в медичних дослідженнях, соціальних науках, фінансових моделях тощо, коли пропуски не можна вважати випадковими.

Таким чином, множинна імпутація, зокрема реалізація MICE, є теоретично обґрунтованим і практично ефективним підходом, що дозволяє забезпечити якісне заповнення даних при дотриманні відповідних припущень.

Модельна імпутація (регресійна та класифікаційна).

Модельна імпутація полягає у використанні наявних даних для побудови прогнозної моделі, яка дозволяє передбачити відсутні значення. Цей підхід передбачає активне використання міжзмінних залежностей і є кроком до глибшої інтеграції попередньої обробки з машинним навчанням. На відміну від простих статистичних методів, модельна імпутація прагне відтворити структуру даних на основі інформації, яка міститься у відомих значеннях інших змінних.

Імпутація на основі регресії передбачає побудову моделі, де змінна з пропусками виступає як цільова, а всі інші змінні – як предиктори. У випадку числових змінних застосовуються моделі лінійної регресії, регуляризовані моделі (наприклад, Ridge, Lasso) або більш складні алгоритми (Random Forest, XGBoost). Для кожного запису, де відсутнє значення, модель робить передбачення на основі наявних ознак. Формально, якщо пропущене значення спостерігається в змінній Y, а X – вектор наявних предикторів, то оцінка $\hat{Y}$ отримується зі співвідношення:

$$\hat{Y} = f(X), \tag{2.6}$$

де f – лінійна або регуляризована регресія, Random Forest чи градієнтний бустинг. Дослідження Stekhoven & Bühlmann показали, що ансамблеві методи (missForest) коректно працюють із числовими та категоріальними ознаками й часто перевершують класичні лінійні моделі [20].

У випадку категоріальних змінних використовується підхід класифікаційної імпутації. Тобто будується модель класифікації (наприклад, логістична регресія, дерево рішень або ансамблеві методи), яка на основі доступних значень інших змінних передбачає ймовірність приналежності до певного класу. У цьому випадку результатом може бути або найбільш ймовірне значення, або вибір класу на основі стохастичного зважування. Така стохастична підстановка допомагає зберегти дисперсію і не «притягувати» вибірку до домінантної моди [21].

Цей підхід дозволяє зберігати внутрішню структуру даних та їхню залежність, особливо коли пропуски мають MAR-природу. Модельна імпутація також підвищує узгодженість між змінними й забезпечує вищу точність, ніж статистичні методи, у разі якісного налаштування моделей.

Проте існують і обмеження. По-перше, для якісної імпутації потрібна достатня кількість повних записів. По-друге, побудова моделі потребує часу та обчислювальних ресурсів. По-третє, виникає ризик перенавчання, якщо модель занадто точно відтворює наявні залежності. Нарешті, при значній кількості змінних та складних взаємозв’язках вибір моделі стає нетривіальним завданням.

Модельна імпутація є корисною в ситуаціях, коли прості методи не забезпечують задовільної якості заповнення, або коли відсутність значень має системний характер, пов’язаний із іншими параметрами. У сучасних системах машинного навчання цей підхід часто інтегрується як частина конвеєра обробки даних, забезпечуючи гнучкість і адаптивність до різноманітних типів задач.

2.1.3 Алгоритми, нечутливі до пропущених значень

Деякі алгоритми машинного навчання здатні працювати з неповними даними без попереднього заповнення пропущених значень. Це досягається за рахунок вбудованих механізмів обробки пропусків або специфічних особливостей реалізації. Такий підхід дає змогу уникнути ризиків, пов’язаних із викривленням розподілу даних, які можуть виникати внаслідок некоректної імпутації.

Найбільш показовими прикладами є дерева рішень і ансамблеві моделі. У класичній монографії Breiman et al. описано концепцію surrogate splits: якщо основна ознака у вузлі дерева має пропуск, алгоритм шукає альтернативну змінну, яка найкраще відтворює початкове правило поділу [22]. Сучасні бібліотеки, зокрема XGBoost та LightGBM, автоматично переадресовують спостереження з пропусками на «missing-branch», обираючи оптимальний бік поділу під час навчання [23], [24]. Таким чином, модель зберігає коректну логіку без явної імпутації. Також це зменшує складність попередньої обробки та дозволяє зосередитися на побудові самої моделі.

Інший підхід полягає у використанні моделей, що базуються на баєсівських методах або латентних змінних. Наприклад, у баєсівських мережах або expectation-maximization (EM) алгоритмах пропуски можуть розглядатися як частина невідомих параметрів, що підлягають оцінці в процесі навчання [25].

Попри очевидну зручність, алгоритмічна нечутливість до пропусків не завжди є гарантією збереження достовірності результатів. У деяких випадках пропуски можуть бути пов’язані з важливою інформацією, і їх ігнорування призводить до втрати частини контексту або упередженості моделі. Крім того, не всі алгоритми мають таку властивість, а отже, залежність від вибору моделі зберігається.

Загалом, використання моделей, стійких до пропусків, є ефективною альтернативою імпутації в тих випадках, коли структура даних дозволяє реалізувати подібні підходи без значної втрати інформації. Це особливо актуально в задачах класифікації з великим числом ознак, де прості методи імпутації можуть бути неефективними або небажаними.

2.1.4 Використання індикаторних змінних

Ще одним підходом до обробки пропущених значень є створення індикаторних змінних, які прямо сигналізують про наявність або відсутність даних у певній ознаці. Такий метод не усуває пропуски безпосередньо, але дозволяє моделі враховувати сам факт їх наявності як додаткову інформацію [26]. Це особливо корисно в ситуаціях, коли відсутність значення є не випадковою, а несе певне змістовне навантаження.

Суть підходу полягає в тому, що для кожної змінної з пропусками створюється нова бінарна змінна-індикатор. Вона набуває значення 1, якщо в початковій змінній був пропуск, і 0 – у протилежному випадку. Наприклад, якщо у змінній «доходи» є пропущені значення, то паралельно створюється ознака «доходи_відсутні», яка містить відповідну інформацію про пропуски. У той самий час початкова змінна може бути заповнена простим методом, наприклад, середнім значенням.

Ідея такого підходу базується на припущенні, що сам факт відсутності даних є інформативним. Наприклад, якщо клієнт не вказав вік у кредитній анкеті, це може свідчити про певний тип поведінки або належність до специфічної групи. У такому разі ігнорування пропуску або його просте заповнення могло би призвести до втрати частини інформації, тоді як індикаторна змінна її зберігає.

З технічної точки зору цей метод є простим в реалізації й легко масштабується на великі набори даних. Він також сумісний із більшістю

алгоритмів машинного навчання, включаючи дерева рішень, логістичну регресію та нейронні мережі.

Водночас використання індикаторних змінних має обмеження. По-перше, воно призводить до збільшення кількості ознак, що може негативно вплинути на моделі, чутливі до розмірності простору. По-друге, якщо пропуск є справді випадковим (MCAR), додавання індикаторної змінної може внести зайвий шум. По-третє, цей метод не усуває потреби в заповненні даних – він лише додає можливість моделі враховувати саму їхню відсутність.

Практика показує, що індикаторні змінні покращують якість моделей, коли механізм пропусків ближчий до MAR або MNAR і сам пропуск несе змістове навантаження [27]. В аналітиці банківських ризиків і медичних дослідженнях такий підхід є стандартним, оскільки дозволяє враховувати поведінкові патерни.

2.2 Масштабування і нормалізація ознак

Масштабування і нормалізація ознак є математичними трансформаціями, які приводять числові змінні до певного стандартного діапазону або розподілу [28]. Їх метою є забезпечення порівнянності ознак, уникнення домінування змінних із великим діапазоном значень та підвищення стабільності числових обчислень у машинному навчанні.

Формально, нехай $X \in \mathbb{R}^{n \times d}$– матриця ознак із спостереженнями та d змінними. Масштабування – це функція $f: \mathbb{R} \to \mathbb{R}$, яка застосовується до кожного стовпця x_j окремо з метою трансформації:

$$X_{i,j}^{(scaled)} = f\left(x_{i,j}\right), i = 1, \dots, n. \tag{2.7}$$

Необхідність масштабування обумовлена властивостями алгоритмів, які чутливі до масштабу ознак. Це зокрема стосується моделей, що

використовують евклідові або інші метричні відстані (k-найближчих сусідів, метод опорних векторів, кластеризація), а також моделей, що навчаються градієнтними методами (лінійна регресія, логістична регресія, нейронні мережі).

У деяких випадках масштабування є також передумовою для виконання припущень моделі. Наприклад, у головних компонентах (PCA) важливо нормалізувати змінні, щоб уникнути зміщення в сторону ознак із високою дисперсією. У регуляризованих моделях (Lasso, Ridge) масштаб ознак впливає на внесок кожної змінної до штрафного члена функціоналу втрат.

2.2.1 Методи масштабування і нормалізації ознак

Мін-макс масштабування – це метод лінійного перетворення значень змінної до заданого діапазону [29], зазвичай [0, 1] або [−1, 1]. Його метою є збереження пропорцій між значеннями ознаки при приведенні до уніфікованої шкали.

Нехай $x = (x_1, x_2, \dots, x_n)$ – значення однієї числової змінної. Тоді результат масштабування для кожного елемента x_i визначається як:

$$x_i^{(scaled)} = \frac{x_i - x_{min}}{x_{max} - x_{min}}, \text{ де } x_{min} = \min(x), x_{max} = \max(x). \quad (2.8)$$

Ця трансформація переводить x_i у відрізок [0, 1], зберігаючи лінійні відносини між значеннями. У загальнішому випадку масштабування до довільного інтервалу [a, b] визначається формулою:

$$x_i^{(scaled)} = a + (b - a) * \frac{x_i - x_{min}}{x_{max} - x_{min}}. \quad (2.9)$$

Мін-макс нормалізація широко застосовується в тих випадках, коли потрібно уніфікувати масштаби змінних для моделей, які чутливі до абсолютних значень, зокрема для алгоритмів, що використовують метрики відстані або градієнтні методи. Вона добре працює у випадках, коли ознаки мають обмежений та чітко визначений діапазон, без значних викидів.

Проте цей метод має істотне обмеження: він є чутливим до викидів, оскільки екстремальні значення визначають масштабування всієї ознаки. Один аномальний запис може суттєво стиснути решту значень у вузький діапазон. У таких випадках варто розглянути альтернативи, зокрема робастне масштабування або перетворення логарифмічного типу.

Мін-макс масштабування зберігає геометричну структуру даних, зокрема відносні відстані та напрямки, що є перевагою при візуалізації та кластеризації. Крім того, ця трансформація є оборотною, що дозволяє легко інтерпретувати відмасштабовані значення в контексті початкових одиниць виміру.

У застосуваннях, де важлива чутливість до відносних змін, а не до абсолютної шкали, мін-макс масштабування виступає ефективним і простим інструментом попередньої обробки даних.

Z-нормалізація, або стандартизація, є одним із базових методів масштабування числових ознак, який приводить розподіл змінної до нульового математичного сподівання та одиничного стандартного відхилення [30]. На відміну від мін-макс масштабування, цей метод не обмежує значення в межах фіксованого інтервалу, але зберігає відстані та структуру розподілу в стандартних одиницях.

Для числової змінної стандартизоване значення кожного елемента обчислюється за формулою:

$$\hat{x}_i = \frac{x_i - \mu}{\sigma}, \tag{2.10}$$

де $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ – вибіркове середнє;

$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$ – вибіркове стандартне відхилення.

Результатом трансформації є ознака з нульовим середнім та дисперсією, близькою до одиниці (точна одиниця – у разі нормального розподілу). Така форма представлення є корисною для моделей, що припускають симетричність або нормальність розподілу ознак, зокрема методів, заснованих на градієнтному спуску, головних компонентах (PCA), або байєсівських оцінках.

Z-нормалізація зберігає лінійні взаємозв'язки між змінними, дозволяє порівнювати ознаки в уніфікованих одиницях і усуває упередженість моделей до змінних із вищою дисперсією. Вона є рекомендованою для моделей, що застосовують регуляризацію (наприклад, Ridge або Lasso), оскільки дозволяє справедливо порівнювати значущість коефіцієнтів.

Серед обмежень методу варто згадати чутливість до викидів, оскільки середнє і стандартне відхилення значно змінюються під впливом екстремальних значень. У разі наявності сильних відхилень у даних доцільніше застосовувати робастні методи масштабування. Крім того, метод не підходить для змінних з категоріальною або бімодальною природою.

Z-нормалізація широко використовується як стандартна трансформація для числових змінних у задачах прогнозування, класифікації, кластеризації та зменшення розмірності, де передбачається приблизна симетрія або однорідність розподілу вхідних даних.

Робастне масштабування – це метод трансформації числових ознак, який використовує медіану та міжквартильний розмах замість середнього та стандартного відхилення [31]. Такий підхід забезпечує стійкість до викидів і є доцільним для обробки даних з асиметричним або забрудненим розподілом.

Формально, для числової змінної $x = (x_1, x_2, \dots, x_n)$, робастно масштабоване значення обчислюється за формулою:

$$\hat{x}_i = \frac{x_i - median(x)}{IQR(x)}, \tag{2.11}$$

де $median(x)$ – медіана вибірки;

$IQR(x) = Q_3 - Q_1$ – міжквартильний розмах, тобто різниця між 75-м та 25-м процентилями.

Медіана та IQR є робастними статистиками, що слабо реагують на екстремальні значення. На відміну від стандартної нормалізації, цей метод дозволяє ефективно масштабувати ознаки з нерівномірним розподілом або даними, що містять викиди. Робастне масштабування також добре підходить для моделей, які не припускають нормальності розподілу, та для задач із сильно варіативними даними.

Застосування цього методу виправдане в умовах, коли дані мають незбалансовану або кластерну структуру. У таких випадках стандартизація або мін-макс масштабування можуть неадекватно відображати структуру вибірки, тоді як робастний підхід забезпечує більш стабільне вирівнювання.

Недоліком методу є зниження чутливості до дрібних змін у даних у центральній частині розподілу, а також обмеження в інтерпретації масштабованих значень у порівнянні з іншими методами. Проте за рахунок стійкості до викидів робастне масштабування є важливим інструментом у попередній обробці даних для широкого спектра задач машинного навчання.

Нелінійні трансформації використовуються для зменшення асиметрії, стабілізації дисперсії та приведення розподілу змінних до форми, яка є більш прийнятною для моделей машинного навчання. Найчастіше ці трансформації застосовуються до правобічно-асиметричних розподілів, які типово зустрічаються в фінансових, демографічних та медичних даних.

Однією з найпростіших та найпоширеніших є логарифмічна трансформація. Вона застосовується до змінних з позитивними значеннями та виконується за формулою:

$$\hat{x}_i = \log(x_i), \tag{2.12}$$

або для уникнення проблем з нульовими значеннями:

$$\hat{x}_i = \log(1 + x_i). \tag{2.13}$$

Така трансформація зменшує вплив великих значень, стискає хвости розподілу та може суттєво покращити результати класифікації або регресії.

Іншим варіантом є коренева трансформація, наприклад, квадратний корінь: $\hat{x}_i = \sqrt{x_i}$. Вона також вирівнює асиметрію, але менш агресивна, ніж логарифм, і застосовується до змінних із нульовими або малими значеннями.

Більш узагальненим підходом є перетворення Бокса-Кокса, яке охоплює логарифмічну, кореневу та лінійну трансформації як часткові випадки. Його загальна формула:

$$\hat{x}_i = \begin{cases} \frac{x_i^2 - 1}{\lambda}, \text{якщо } \lambda \neq 0 \\ \log(x_i), \text{якщо } \lambda = 0 \end{cases}, \tag{2.14}$$

де λ – параметр, що визначається емпірично. Перетворення Бокса-Кокса застосовується до позитивних значень і дозволяє вибрати оптимальну трансформацію для досягнення нормальності розподілу або мінімізації дисперсії залишків.

Також використовуються перетворення Йео-Джонсона (Yeo–Johnson) – модифікація Бокса-Кокса, яка працює і з від’ємними значеннями:

$$\hat{x}_i = \begin{cases} \frac{\left((x_i+1)^{\lambda}-1\right)}{\lambda}, x_i \geq 0, \lambda \neq 0 \\ \log(x_i+1), x_i \geq 0, \lambda = 0 \\ -\frac{\left((-x_i+1)^{2-\lambda}-1\right)}{2-\lambda}, x_i < 0, \lambda \neq 2 \\ -\log(-x_i+1), x_i < 0, \lambda = 2 \end{cases} . \quad (2.15)$$

Це особливо важливо у випадках, коли дані містять як додатні, так і від'ємні значення, і немає можливості їх попередньо зміщувати.

Нелінійні трансформації не зберігають відстані між точками та можуть ускладнювати інтерпретацію результатів. Водночас вони є потужним інструментом для покращення якості моделей у випадках, коли лінійне масштабування виявляється недостатнім або неефективним.

Їх застосування доцільне лише після аналізу розподілу даних, оцінки асиметрії та виявлення викидів. У разі правильного використання ці методи можуть істотно покращити збіжність градієнтних алгоритмів, стабільність моделей та інтерпретованість результатів у просторах ознак зі складною структурою.

2.2.2 Порівняння методів масштабування і висновки

Масштабування ознак є критично важливим етапом у підготовці даних для багатьох моделей машинного навчання. На основі аналізу розглянутих підходів можна сформулювати загальні рекомендації та окреслити критерії вибору відповідного методу залежно від характеристик даних.

Мін-макс масштабування приводить усі значення ознак до заздалегідь визначеного діапазону, найчастіше [0; 1], що робить його зручним для моделей, чутливих до абсолютних масштабів. Однак цей метод особливо вразливий до викидів, оскільки крайні значення суттєво впливають на інтервал масштабування.

Z-нормалізація (стандартизація) забезпечує нульове середнє та одиничну дисперсію. Вона є більш стійкою до змін масштабу та корисною для методів, які передбачають нормальний розподіл або використовують евклідову метрику (наприклад, SVM, логістична регресія, PCA). Проте її результат також може бути спотворений при наявності викидів.

Робастне масштабування засноване на медіані та міжквартильному розмахові (IQR) і рекомендоване у випадках, коли дані містять викиди або мають асиметричний розподіл. Воно не приводить значення до фіксованого діапазону, але забезпечує більшу стабільність у присутності аномальних спостережень.

Нелінійні трансформації, такі як логарифмічна, коренева, Box–Cox і Yeo–Johnson, слугують для усунення асиметрії розподілу або стабілізації дисперсії. Вони можуть істотно вплинути на структуру ознаки, тому їх слід використовувати обережно, попередньо проаналізувавши форму розподілу.

Загалом вибір методу масштабування повинен ґрунтуватися на:

- типі моделі машинного навчання;
- наявності викидів у даних;
- розподілі значень ознак;
- необхідності збереження інтерпретованості ознак.

У деяких випадках доцільним є поєднання декількох підходів. Наприклад, після логарифмічного перетворення може бути застосована стандартизація для подальшого узгодження масштабів.

Ефективне масштабування підвищує швидкість збіжності моделей, покращує точність прогнозування та робить результати більш стабільними на нових вибірках. У сучасних пайплайнах машинного навчання цей етап часто автоматизується, однак його доцільність і правильність вибору залишаються відповідальністю дослідника.

2.3 Трансформація категоріальних змінних

Категоріальні змінні є поширеним типом ознак у прикладних задачах машинного навчання [32]. Вони відображають якісні характеристики об’єктів, такі як стать, регіон, тип транзакції, професія, марка автомобіля тощо. Більшість алгоритмів машинного навчання, зокрема ті, що працюють із числовими обчисленнями, не здатні безпосередньо обробляти нечислові (символьні) значення. Саме тому необхідною умовою є перетворення категоріальних змінних у числову форму.

Існує кілька підходів до кодування таких змінних. Їх вибір залежить від типу змінної (номінативна або порядкова), кількості унікальних категорій (кардинальності), цільової моделі та вимог до інтерпретованості. Неправильне перетворення може призвести до хибного уявлення про структуру даних або до деградації якості моделі.

Категоріальні змінні поділяються на два основних типи:

– номінативні, які не мають природного порядку, наприклад «колір»;

– порядкові, які мають логічну послідовність, наприклад «низький», «середній», «високий».

У цьому підрозділі буде розглянуто основні методи трансформації: one-hot encoding, порядкове кодування, цільове кодування, частотне кодування, бінарне кодування та embedding-підходи.

2.3.1 Методи трансформації категоріальних змін

One-hot encoding (одноразове бінарне кодування) – це один із найпоширеніших методів трансформації номінативних змінних у числову форму [32]. Його суть полягає у створенні окремої бінарної (0/1) змінної для кожного унікального значення (категорії) вихідної ознаки. Кожен об’єкт у наборі даних отримує 1 у тій змінній, яка відповідає його категорії, і 0 – у всіх інших.

Нехай x – категоріальна змінна з k унікальними значеннями: $\{C_1, C_2, \dots, C_k\}$. Тоді one-hot encoding будує k нових змінних $x^{(1)}, x^{(2)}, \dots, x^{(k)}$, таких що:

$$x_i^{(j)} = \begin{cases} 1, & \text{якщо } x_i = C_j \\ 0, & \text{інакше} \end{cases}. \quad (2.16)$$

Цей метод забезпечує математично коректне представлення номінативних ознак, оскільки не вносить штучної інформації про порядок або метрику між категоріями. Завдяки цьому він дозволяє уникати логічних викривлень під час навчання моделей. One-hot encoding є сумісним із великою кількістю моделей машинного навчання, включаючи лінійні моделі, дерева рішень, методи ансамблю та нейронні мережі. Його особливо доцільно застосовувати тоді, коли кількість унікальних категорій є обмеженою.

Перевагою one-hot encoding є збереження незалежності між категоріями, що відповідає структурі номінативних ознак. Крім того, метод є простим в реалізації та добре підтримується всіма основними програмними бібліотеками машинного навчання.

Разом із тим, метод має низку обмежень. Зокрема, при високій кардинальності ознаки (великій кількості унікальних значень) він призводить до значного збільшення розмірності ознакового простору. Це підвищує вимоги до пам’яті та часу обчислень, а також сприяє розрідженості матриці ознак, що негативно впливає на деякі моделі. Крім того, у випадку застосування до лінійних моделей може виникати проблема мультиколінеарності – лінійної залежності між новоствореними ознаками.

Для зменшення надлишковості в one-hot представленні іноді застосовується варіант, відомий як drop-first encoding. Його суть полягає у видаленні однієї (наприклад, першої) з отриманих бінарних колонок, оскільки вона є лінійно залежною від решти. Такий підхід дозволяє

уникнути сингулярності в матриці ознак при використанні моделей, чутливих до мультиколінеарності (наприклад, регресійних моделей).

Загалом one-hot encoding є базовим і широко вживаним методом трансформації категоріальних змінних. Його застосування доцільне для ознак з невисокою кардинальністю та тоді, коли важливо уникати нав'язування хибної метрики або порядку між значеннями ознаки.

Порядкове кодування – це метод трансформації категоріальних змінних, який полягає у присвоєнні кожному унікальному значенню певного цілого числа відповідно до наперед визначеного або логічно обґрунтованого порядку. Цей метод є доцільним для змінних із впорядкованими категоріями, де існує природна градація між значеннями.

Нехай $x \in \{C_1, C_2, \dots, C_k\}$ – порядкова змінна, наприклад: «низький», «середній», «високий». Тоді порядкове кодування реалізується за схемою:

$$x_i \to j_i, \text{де } C_j \text{ – категорія, якій відповідає } x_i. \qquad (2.17)$$

Таким чином, значення «низький», «середній», «високий» можуть бути закодовані як 0, 1, 2 відповідно.

Порядкове кодування не розширює простір ознак, що робить його обчислювально ефективним і придатним для роботи з ознаками з великою кількістю рівнів. Його перевага також полягає в збереженні порядкової структури змінної, що дозволяє моделям враховувати ієрархію значень.

Проте застосування цього методу є обґрунтованим лише тоді, коли порядковість категорій є змістовно релевантною. Якщо змінна є номінативною, але до неї помилково застосовується порядкове кодування, це може призвести до некоректного навчання моделі через штучно створену метрику між категоріями.

Особливо обережним слід бути з моделями, що враховують відстані між значеннями ознак, такими як лінійна регресія, методи класифікації на

основі відстаней або нейронні мережі. У таких випадках помилкове припущення про лінійність порядку може суттєво викривити результати.

Порядкове кодування є рекомендованим для змінних, де порядок категорій має змістову інтерпретацію – наприклад, освітній рівень, клас ризику, рівень задоволеності тощо. Для номінативних ознак слід надавати перевагу one-hot або іншим кодуванням, які не передбачають впорядкованості.

Цільове кодування – це метод трансформації категоріальних змінних, при якому кожна категорія замінюється на статистичну характеристику цільової змінної (наприклад, середнє значення або частоту позитивного класу), обчислену для відповідної категорії [33]. Цей підхід є потужним інструментом для задач із великою кількістю категорій, зокрема в задачах класифікації та регресії.

Нехай y – цільова змінна (наприклад, мітка класу), а $x \in \{C_1, C_2, \dots, C_k\}$ – категоріальна змінна. Тоді значення кожного x_i трансформується як:

$$x_i^{(enc)} = \mathbb{E}[y|x = C_j], \tag{2.18}$$

тобто очікуване значення цільової змінної при фіксованій категорії C_j.

Перевага target encoding полягає в тому, що він не збільшує розмірність простору ознак, на відміну від one-hot encoding, і водночас дозволяє зберегти інформацію про взаємозв'язок між ознакою та цільовою змінною. Це особливо важливо для моделей, які враховують числову інтерпретацію ознак (лінійні моделі, градієнтний бустинг, логістична регресія).

Однак без додаткових заходів target encoding може призводити до витоку інформації (target leakage), коли значення цільової змінної частково потрапляє в навчальні ознаки. Для уникнення цього зазвичай використовують:

– крос-валідаційне усереднення (out-of-fold encoding);

– регуляризацію (наприклад, згладжене середнє між глобальним та категоріальним);

– додавання стохастичного шуму.

Target encoding є особливо ефективним при високій кардинальності ознаки або у випадках, коли зв'язок між категорією і цільовою змінною є статистично значущим. Водночас його застосування вимагає обережності й належного контролю за потенційним перенавчанням.

Цей метод часто використовується у фінансовому моделюванні, у прогнозуванні поведінки користувачів, а також в інших задачах, де важливо зберегти інформативність категоріальних змінних без створення великої кількості додаткових ознак.

Частотне кодування – це метод перетворення категоріальної змінної, при якому кожному її унікальному значенню присвоюється числова ознака, що відповідає частоті (або ймовірності) його появи в навчальній вибірці. Це дозволяє замінити символьні категорії на числові значення без суттєвого збільшення розмірності ознакового простору.

Формально, якщо $x \in \{C_1, C_2, \dots, C_k\}$ – категоріальна змінна, то для кожної категорії C_j обчислюється:

$$x_i^{(enc)} = \frac{count(x_i)}{n}, \tag{2.19}$$

де $count(x_i)$ – кількість спостережень категорії;

n – загальна кількість спостережень у вибірці.

На відміну від one-hot encoding, частотне кодування не створює нових змінних, а зберігає початкову структуру таблиці. Це робить метод привабливим для роботи з ознаками високої кардинальності, де one-hot підхід призводив би до значного збільшення кількості стовпців.

Перевагою є простота реалізації, компактність представлення та збереження числового формату, сумісного з більшістю моделей. Однак метод не враховує зв’язок між ознакою і цільовою змінною, як це робить target encoding. Крім того, моделі можуть інтерпретувати частотні значення як кількісні, вводячи потенційно хибні залежності.

Щоб зменшити ризики некоректної інтерпретації, іноді частотне кодування комбінують із бінінгом частот або логарифмічним згладжуванням. Також можливе використання нормалізованих частот або індексів рангу замість абсолютних частот.

Частотне кодування є ефективним компромісом між простими і складними методами обробки категоріальних змінних, і добре підходить для попереднього кодування ознак у великих табличних наборах даних.

Бінарне кодування – це метод трансформації категоріальних змінних, який поєднує в собі елементи порядкового та бітового (двійкового) подання [34]. Основна ідея полягає в присвоєнні кожній категорії унікального цілого значення, після чого це значення представлено у двійковій формі, а кожен біт стає окремою бінарною ознакою.

Зазвичай реалізація бінарного кодування складається з двох послідовних етапів. Спочатку категоріальним значенням призначаються цілі числа від 0 до $k-1$, де k – кількість унікальних категорій. Далі кожне з цих чисел трансформується у двійковий вектор однакової довжини, відповідно до його представлення у бітовій формі. Наприклад, для категорій $\{A, B, C, D\}$ можна отримати коди $\{0, 1, 2, 3\}$, що у двійковій формі відповідають $\{00, 01, 10, 11\}$, які й формують дві нові бінарні ознаки.

Однією з ключових переваг бінарного кодування є зменшення кількості створюваних змінних порівняно з one-hot encoding. Для k категорій достатньо $\lceil \log_2 k \rceil$ бітових змінних, що дозволяє значно знизити розмірність ознакового простору у випадках високої кардинальності. Також цей підхід забезпечує числове представлення категорій без нав’язування штучного порядку, характерного для звичайного порядкового кодування.

До недоліків належить те, що бітові вектори не гарантують незалежності між категоріями: однакові біти можуть повторюватися у різних кодуваннях, що потенційно створює хибні залежності між ознаками. Це може ускладнити навчання моделей, особливо тих, що базуються на відстані або передбачають незалежність ознак. Крім того, для інтерпретованості результатів необхідна наявність таблиці відповідності між категоріями і бітовими кодами.

Бінарне кодування є ефективним компромісом для ознак із великою кількістю рівнів, де застосування one-hot encoding призводить до надмірного навантаження, а порядкове кодування є методологічно недоречним. Метод доцільно використовувати в задачах табличного моделювання, особливо при обробці змінних із десятками або сотнями унікальних категорій.

Embedding-підходи передбачають представлення категоріальних змінних у вигляді векторів дійсних чисел фіксованої розмірності, які навчаються разом із моделлю. На відміну від one-hot чи порядкового кодування, де категорії мають фіксоване відображення, embedding-репрезентації оптимізуються під час навчання і можуть передавати складні семантичні або латентні зв’язки між категоріями [35].

Формально, кожній категорії $C_j \in \{C_1, C_2, \dots, C_k\}$ ставиться у відповідність вектор $e_j \in \mathbb{R}^d$, де $d \ll k$. Матриця $E \in \mathbb{R}^{k \times d}$ параметризує embedding-простір і оновлюється під час навчання моделі за допомогою градієнтного спуску.

Цей підхід є основним у сучасних нейронних архітектурах, зокрема в задачах обробки природної мови, рекомендаційних системах і табличному моделюванні (наприклад, TabNet, DeepFM). Embedding дозволяє не лише зменшити розмірність вхідних даних, але й вивчити латентні представлення, які захоплюють подібність між категоріями.

Переваги embedding-підходу включають:

– компактність (значне зменшення розмірності навіть при високій кардинальності);

– можливість відображення семантичної подібності між категоріями;

– адаптивність (вектори навчаються в контексті конкретної задачі).

Обмеженнями є потреба у великому обсязі даних для стабільного навчання embedding-простору, складність інтерпретації векторів та ризик перенавчання при недостатньо вираженій структурі.

Embedding-підходи зазвичай використовуються в поєднанні з глибокими нейронними мережами. Для їх ефективного застосування слід визначити розмір embedding-простору (гіперпараметр), а також забезпечити регуляризацію, баланс між категоріями та достатню кількість прикладів для кожної з них.

У випадках, коли доступна значна кількість даних і модель допускає навчання векторних представлень, embedding-підходи забезпечують найбільш гнучкий та виразний спосіб кодування категоріальних змінних.

2.3.2 Порівняння методів кодування

Розглянуті методи кодування категоріальних змінних мають різну природу, обчислювальні властивості та вплив на результати моделювання. Вибір конкретного методу залежить від типу змінної, цільової моделі, кількості унікальних категорій та обсягу доступних даних.

One-hot encoding є базовим і універсальним методом, який зберігає незалежність між категоріями та не вводить упереджених припущень про їх взаємозв’язок. Його застосування доцільне для змінних із невеликою кількістю унікальних значень, але в умовах високої кардинальності призводить до значного розширення ознакового простору.

Порядкове кодування є простим і компактним, але його використання можливе лише для впорядкованих змінних. Якщо така умова не

виконується, метод може призводити до хибного тлумачення ознаки моделлю.

Target encoding дозволяє враховувати зв’язок між категорією і цільовою змінною, що може підвищити ефективність моделі, але водночас вимагає обережного застосування через ризик витоку інформації. Для його стабільної роботи необхідні прийоми регуляризації та крос-валідації.

Частотне кодування забезпечує простий компроміс між числовим представленням і обмеженою розмірністю, але не передає жодної інформації про залежність від цільової змінної.

Бінарне кодування дозволяє значно зменшити розмірність ознакового простору при високій кардинальності, однак втрачає незалежність між категоріями, що може бути критично для деяких моделей.

Embedding-підходи є найбільш гнучкими та виразними, особливо в контексті глибокого навчання, але потребують великих обсягів даних, налаштування гіперпараметрів і наявності ресурсів для навчання нейронних мереж.

Жоден з методів не є універсальним. Практичний вибір має ґрунтуватися на аналізі статистичних властивостей змінної, характеристик моделі та вимог до обчислювальної складності. Часто доцільним є експериментальне порівняння декількох варіантів із використанням валідаційних наборів даних.

Таким чином, ефективне кодування категоріальних змінних є невід’ємною складовою якісної побудови моделей машинного навчання, а глибоке розуміння відповідних методів дозволяє приймати обґрунтовані рішення щодо трансформації ознак у конкретних прикладних задачах.

2.4 Виявлення та обробка аномалій

Аномалії (або викиди) – це спостереження, які суттєво відрізняються від загальної структури даних. Їх наявність може бути наслідком помилок

вимірювання, рідкісних подій або відображати важливі патерни, що мають прикладне значення (наприклад, шахрайські транзакції, технічні збої або виняткові медичні стани) [36].

З формальної точки зору, якщо дані походять із певного розподілу ймовірності $p(x)$, і спостереження x_i має значно нижчу ймовірність порівняно з типовими точками вибірки (тобто $p(x_i) \ll p(x)$ для більшості x), тоді x_i може класифікуватися як аномалія. Конкретні критерії належать до контексту й методології аналізу.

Існує кілька основних типів аномалій. Точкові аномалії (point anomalies) – це окремі спостереження, які суттєво відрізняються від загального розподілу. Прикладом є температура тіла 42°C при середньому значенні 36.6°C. Контекстні аномалії (contextual anomalies) – це значення, що є аномальними лише в певному контексті; наприклад, температура 25°C влітку є нормальною, але взимку – потенційно аномальною. Колективні аномалії (collective anomalies) становлять групи спостережень, які разом формують нетиповий патерн, навіть якщо окремі значення не викликають підозр. Як приклад можна навести серію різких падінь фінансових показників протягом кількох днів.

Аномалії також класифікують за характером їхньої присутності в ознаковому просторі. Одновимірні викиди – це аномалії, які виявляються в межах окремих змінних. Багатовимірні викиди виглядають нормальними при розгляді кожної ознаки окремо, але в сукупності мають нетипову комбінацію. Часові викиди – це спостереження, що порушують часові закономірності, такі як сезонні коливання або тренди.

Розуміння типів аномалій є ключовим для правильного вибору відповідних методів їх виявлення, інтерпретації та подальшої обробки. У наступному підрозділі буде розглянуто алгоритмічні та статистичні підходи до детектування аномалій у реальних даних.

2.4.1 Методи виявлення аномалій

Статистичні методи виявлення викидів.

Статистичні методи становлять одну з найдавніших та найбільш теоретично обґрунтованих груп підходів до ідентифікації аномальних спостережень. Вони базуються на припущенні про те, що дані підкоряються певному ймовірнісному розподілу, і значення, які суттєво відхиляються від його центральної тенденції, можуть бути класифіковані як викиди.

Одним із базових і водночас робастних підходів є виявлення викидів на основі міжквартильного розмаху (IQR). Для заданої вибірки визначаються перший (Q1) і третій (Q3) квартилі, а також міжквартильний розмах: $IQR = Q_3 - Q_1$. Спостереження вважаються викидами, якщо вони розташовані нижче межі $Q_1 - 1{,}5 \times IQR$ або вище межі $Q_3 + 1{,}5 \times IQR$. Метод не потребує жорстких припущень щодо розподілу і стійкий до викидів, що робить його придатним для попереднього аналізу даних [36].

Інший класичний підхід ґрунтується на використанні Z-оцінок, які відображають кількість стандартних відхилень, на яку спостереження x_i віддалене від середнього значення μ:

$$z_i = \frac{x_i - \mu}{\sigma}. \tag{2.20}$$

Спостереження вважається аномальним, якщо $|z_i| > k$, де k зазвичай становить 2 або 3. Хоча цей метод є ефективним для даних із нормальним розподілом, він вразливий до впливу аномалій на оцінки μ та σ, що може призводити до спотворення результатів.

У контексті невеликих вибірок часто застосовується критерій Граббса (Grubbs’ test), який дозволяє формально перевірити гіпотезу про наявність одиничного викиду. Метод базується на t-розподілі і вимагає дотримання припущення про нормальність даних.

Для багатовимірних задач застосовують статистику Махаланобіса, яка враховує не лише відстань між точками, а й кореляційну структуру вибірки. Відстань Махаланобіса визначається як:

$$D^2(x) = (x - \mu)^T \Sigma^{-1} (x - \mu), \tag{2.21}$$

де Σ – матриця коваріації. Цей метод дає змогу виявити спостереження, які є віддаленими від центру розподілу з урахуванням взаємозв'язків між змінними.

Статистичні методи вирізняються обчислювальною простотою, прозорістю формулювання та високою інтерпретованістю. Проте їх ефективність суттєво залежить від припущень про розподіл даних і може знижуватися у випадках складної топології простору ознак або високої розмірності. У таких ситуаціях доцільно використовувати більш гнучкі підходи, зокрема засновані на локальній щільності або навчанні моделей.

Методи на основі щільності.

Методи виявлення викидів на основі щільності передбачають оцінку густини розподілу даних у локальних або глобальних околицях спостережень. Основна ідея полягає в тому, що аномальні об'єкти мають нижчу щільність, ніж їхні оточення, оскільки вони трапляються рідше або перебувають у регіонах із меншою концентрацією даних.

Одним із найпростіших є підхід на основі припущення про нормальний розподіл даних. У цьому випадку можна обчислити ймовірність появи спостереження x_i як $p(x_i)$ на основі параметрів розподілу. Значення, які мають низьку ймовірність (наприклад, $p(x_i) < 0.01$), можуть бути інтерпретовані як аномалії. Проте цей метод вимагає знання або оцінки параметрів розподілу та є нестійким до відхилень від нормальності.

Більш гнучким є метод ядрової оцінки щільності (KDE – Kernel Density Estimation), який апроксимує щільність розподілу без жорстких припущень. KDE для точки x визначається як:

$$p(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right), \quad (2.22)$$

де K – ядро (наприклад, гаусове);

h – ширина вікна (параметр згладжування).

Значення , для яких оцінена щільність істотно менша за середню, можуть вважатися викидами. Метод є інтерпретованим і добре працює для одновимірних або низькорозмірних даних, однак чутливий до вибору параметрів та масштабу.

Одним із найвідоміших алгоритмів локального оцінювання щільності є Local Outlier Factor (LOF) [37]. Він порівнює локальну щільність точки з локальними щільностями її сусідів. Якщо точка має значно нижчу щільність, ніж її оточення, їй приписується високий LOF-індекс, що свідчить про її аномальність. На відміну від глобальних методів, LOF враховує локальну варіативність, що робить його ефективним при роботі з нерівномірно розподіленими даними.

Методи на основі щільності є цінними для виявлення як точкових, так і багатовимірних аномалій у задачах, де відсутнє чітке уявлення про структуру даних. Їх застосування доцільне за наявності достатнього обсягу даних і при невисокій розмірності ознакового простору, що дозволяє уникнути ефекту «прокляття розмірності».

Методи виявлення аномалій на основі кластеризації передбачають аналіз структурованості даних шляхом групування спостережень у кластери. Основна ідея полягає в тому, що нормальні об'єкти утворюють щільні групи, тоді як аномальні – ізольовані або погано інтегровані у кластери.

Одним з найпростіших підходів є використання алгоритму k-середніх (k-means), при якому кожне спостереження призначається до найближчого центроїда. Аномалії можна виявити на основі великої відстані спостереження до центру свого кластеру. Незважаючи на простоту, метод

чутливий до вибору k та не враховує локальну густину. Він також погано працює у випадках, коли кластери мають складну форму.

Іншим популярним підходом є алгоритм DBSCAN (Density-Based Spatial Clustering of Applications with Noise), який формує кластери на основі щільності. Об'єкти, що не входять до жодного кластера, класифікуються як викиди. Цей метод є більш гнучким, оскільки не потребує попереднього задання кількості кластерів, а також дозволяє виявляти кластери довільної форми. Проте його ефективність залежить від вибору гіперпараметрів, зокрема радіусу ε та мінімальної кількості сусідів.

Додатково можуть застосовуватись відстаневі критерії, які порівнюють відстань спостереження до найближчого кластера або до кількох сусідніх кластерів. У багатовимірних просторах також використовуються алгоритми ієрархічної кластеризації, що дозволяють визначити ступінь інтегрованості спостереження у структуру даних.

Кластеризаційні методи виявлення викидів є ефективними за умови наявності чіткої сегментації даних та помірної кількості шуму. Їх перевага полягає у здатності виявляти як точкові, так і групові аномалії без необхідності попередньої розмітки даних. Проте вони можуть демонструвати нестабільність у випадках високої розмірності або сильно варіативної густини вибірки.

Методи виявлення викидів із використанням навчання передбачають побудову моделей, здатних розрізняти нормальні та аномальні спостереження на основі їх поведінки, структури ознак або статистичних закономірностей. Ці підходи можуть реалізовуватись як у межах навчання без учителя, так і у формі напівконтрольованого або повністю контрольованого навчання.

Одним із найпоширеніших алгоритмів є Isolation Forest, який ґрунтується на принципі випадкового розділення простору ознак. Ідея полягає в тому, що аномальні спостереження швидше ізолюються за допомогою послідовних бінарних розщеплень через їхню ізольованість і

малу частотність. Для кожного об’єкта обчислюється середня довжина шляху до його ізоляції у випадковому лісі. Чим коротший шлях, тим вища ймовірність, що об’єкт є аномальним. Isolation Forest є ефективним при великій кількості ознак і не потребує попереднього масштабування [38].

Ще одним класичним підходом є метод One-Class SVM (Support Vector Machine), який навчається на позитивних прикладах (нормальних спостереженнях) і формує межу, що відокремлює їх від потенційних викидів [39]. Побудова гіперплощини максимально віддаленої від початку координат забезпечує класифікацію нових точок як нормальних або аномальних. Метод є чутливим до вибору ядра та параметра жорсткості, але демонструє високу точність у низькорозмірних просторах з добре структурованими даними.

У задачах, де доступна велика кількість прикладів і високий рівень складності ознак, застосовуються нейромережеві підходи, зокрема автоенкодери. Автоенкодер – це архітектура, яка навчається стискати вхідні дані до латентного простору і відновлювати їх із втратами. Якщо модель навчена на нормальних даних, то відновлення аномалій буде супроводжуватись великою помилкою реконструкції. Цей підхід дозволяє виявляти викиди у складних багатовимірних просторах і є стійким до нелінійностей у даних.

Методи навчання дозволяють досягти високої адаптивності та точності виявлення аномалій, особливо в ситуаціях, коли структура викидів складна або прихована. Їх ефективність залежить від обсягу даних, збалансованості вибірки та якості попередньої обробки ознак. Водночас такі методи є більш обчислювально складними та можуть потребувати ретельного налаштування гіперпараметрів і ресурсомісткого навчання.

2.4.2 Порівняння методів виявлення аномалій

Розглянуті методи виявлення викидів, у попередніх підрозділах, охоплюють широкий спектр підходів – від класичних статистичних до складних моделей навчання. Кожен з методів має свої переваги, обмеження та галузі доцільного застосування. Розуміння цих характеристик є важливим для вибору найбільш ефективної стратегії обробки аномальних даних.

Статистичні методи, як-от IQR, Z-оцінки або тест Граббса, вирізняються простотою реалізації та високою інтерпретованістю. Вони є доцільними у задачах з добре відомою структурою розподілу, особливо при аналізі одновимірних ознак. Проте їх ефективність знижується за наявності складної геометрії даних або великої кількості вимірів.

Методи на основі щільності, зокрема KDE та LOF, є більш гнучкими і дозволяють виявляти локальні аномалії. Вони не потребують попереднього навчання, однак чутливі до вибору гіперпараметрів і обмежені ефектом «прокляття розмірності» у високовимірних просторах.

Кластеризаційні алгоритми, такі як k-means чи DBSCAN, добре виявляють спостереження, які не входять до жодного з кластерів або значно віддалені від центрів груп. Вони є особливо корисними, коли дані демонструють природну кластерну структуру, але можуть давати нестабільні результати при наявності шуму або складної форми кластерів.

Моделі з навчанням – One-Class SVM, Isolation Forest, автоенкодери демонструють високу ефективність у задачах із великою кількістю змінних або складними розподілами. Вони адаптивні, можуть враховувати нелінійні залежності, але вимагають ретельного налаштування та достатніх обчислювальних ресурсів.

Вибір оптимального методу залежить від характеристик датасету: розмірності, обсягу, природи ознак, наявності маркованих прикладів і типу очікуваних викидів. У практичних застосуваннях часто використовують

комбінації підходів, а також етапи валідації для перевірки стабільності виявлених аномалій.

Таким чином, побудова ефективної системи виявлення викидів вимагає не лише обізнаності з теоретичними основами методів, а й здатності адаптувати вибір під конкретну задачу, з урахуванням особливостей даних та цілей аналізу.

2.5 Зниження розмірності

У сучасних задачах штучного інтелекту та машинного навчання високорозмірні простори ознак становлять одну з ключових методологічних проблем. Такі простори виникають як унаслідок наявності великої кількості первинних змінних (наприклад, пікселі в зображеннях, генетичні маркери в біоінформатиці, тисячі токенів у текстових моделях), так і через попередні трансформації даних, зокрема, кодування категоріальних змінних за допомогою one-hot-перетворення.

Незважаючи на прогрес у розвитку обчислювальної техніки, висока розмірність ознакового простору призводить до низки теоретичних та практичних ускладнень, які в науковій літературі окреслюються як «прокляття розмірності» (англ. curse of dimensionality) [43]. Цей термін охоплює явища, пов’язані з експоненційним зростанням обсягу простору зі збільшенням кількості вимірів, що, у свою чергу, викликає розрідженість даних, втрату ефективності метрик відстані, збільшення варіації оцінок та складність побудови узагальнюючих моделей. Зокрема, для коректного функціонування багатьох алгоритмів машинного навчання необхідно значно збільшувати кількість навчальних прикладів, що не завжди є можливим на практиці.

Окрім зазначених ефектів, у реальних даних часто трапляються надлишкові, скорельовані або нерелевантні змінні. Їх присутність не тільки не збагачує модель новою інформацією, а й може виступати джерелом

шуму, що погіршує продуктивність алгоритмів і знижує інтерпретованість результатів.

У таких умовах зниження розмірності відіграє роль важливого етапу попередньої обробки. Його мета полягає в побудові трансформації простору ознак, яка дозволяє зберегти якомога більше статистично значущої інформації при зменшенні кількості вимірів. Формально це відповідає пошуку відображення $\phi : \mathbb{R}^n \to \mathbb{R}^k$, де $k \ll n$, де , що мінімізує втрати інформації або максимізує деяку форму варіації чи роздільності класів.

Зниження розмірності може також розглядатися як засіб регуляризації, що сприяє зменшенню перенавчання, стабілізації моделей та підвищенню узагальнювальної здатності. У подальших підрозділах буде здійснено систематичний огляд найпоширеніших методів лінійного та нелінійного зменшення розмірності, з відповідними теоретичними засадами та прикладними критеріями їх застосування.

2.5.1 Методи зниження розмірності

Метод головних компонент (PCA).

Метод головних компонент (Principal Component Analysis, PCA) є одним із найпоширеніших і математично обґрунтованих підходів до зниження розмірності [44]. Його суть полягає в ортогональному перетворенні вхідних змінних у новий набір змінних – головні компоненти, які є лінійними комбінаціями початкових ознак. Ці компоненти впорядковані за спаданням дисперсії, яку вони пояснюють, і таким чином дозволяють зберегти максимальну кількість варіації даних у меншому числі вимірів.

Формально, нехай $X \in \mathbb{R}^{n \times d}$ – центрована матриця даних. PCA прагне знайти ортонормований базис $W \in \mathbb{R}^{d \times k}$, що максимізує дисперсію проєкцій:

$$W^* = \arg\max_W Tr(W^T S W), \text{ за умови } W^T W = I_k, \qquad (2.23)$$

де $S = \frac{1}{n} X^T X$ – коваріаційна матриця. Розв'язання задачі зводиться до знаходження власних векторів S, що відповідають найбільшим власним значенням.

Перші кілька головних компонент містять основну частину варіації даних, тому заміна початкового простору на простір з меншою розмірністю $k \ll d$ дозволяє зменшити складність задачі без суттєвих втрат інформації. Важливо, що PCA не враховує цільову змінну, отже, є методом ненаглядного (unsupervised) зниження розмірності.

Серед переваг PCA – його обчислювальна ефективність, можливість аналітичного розв'язання, стабільність і зручність візуалізації даних. Проте метод чутливий до масштабування ознак, тому зазвичай вимагає стандартизації даних перед застосуванням. Крім того, PCA ефективно працює лише за припущення лінійності структури даних і не дозволяє відобразити складні, нелінійні взаємозв'язки.

PCA широко використовується для попереднього аналізу, візуалізації багатовимірних даних, прискорення обчислень у моделях машинного навчання, боротьби з мультиколінеарністю та зменшення шуму в даних.

Лінійний дискримінантний аналіз (Linear Discriminant Analysis, LDA) є лінійним методом зниження розмірності, який, на відміну від PCA, враховує наявність міток класів і спрямований на максимізацію роздільності між ними. Його основною метою є проєкція даних у простір меншої розмірності так, щоб зберігалася якомога більша дискримінативна інформація для класифікації.

Постановка задачі LDA базується на побудові матриць міжкласової та внутрішньокласової дисперсії. Нехай маємо множину C класів з центрами $\mu_1, \dots, \mu_C$, загальним центром μ, та дані $X \in \mathbb{R}^{n \times d}$.

Визначаємо матриці:

$$S_B = \sum_{c=1}^{C} n_c(\mu_c - \mu)(\mu_c - \mu)^T, \quad (2.24)$$

$$S_W = \sum_{c=1}^{C} \sum_{X_i \epsilon C} (x_i - \mu_c)(x_i - \mu_c)^T, \quad (2.25)$$

де S_B міжкласова дисперсія;

S_W – внутрішньокласова дисперсія.

Метою LDA є знаходження матриці W, яка максимізує відношення:

$$J(W) = \frac{\det(W^T S_B W)}{\det(W^T S_W W)}. \quad (2.26)$$

Оптимальне W отримується як власні вектори узагальненого власного розв'язку:

$$S_B w = \lambda S_W w. \quad (2.27)$$

Максимальна кількість дискримінантних компонент не перевищує $C - 1$, тому LDA найчастіше застосовується у задачах класифікації з відносно невеликою кількістю класів.

Перевагою LDA є його здатність підвищувати роздільну здатність моделей класифікації за рахунок проєкції даних у простір з оптимальним співвідношенням міжкласової та внутрішньокласової варіації. LDA є стійким до мультиколінеарності та може застосовуватися як самостійний алгоритм класифікації (лінійний класифікатор), так і як етап попередньої обробки ознак.

Обмеження методу полягають у припущенні про нормальність розподілу даних у кожному класі, однакову коваріаційну матрицю класів, а також у його неефективності у випадках складної нелінійної структури ознак.

LDA широко застосовується в біометричних системах, обробці зображень, аналізі текстів, а також у задачах медичної діагностики та фінансового моделювання, де важливе значення має розділення між класами при мінімальній втраті інформації.

Метод t-SNE (t-Distributed Stochastic Neighbor Embedding) є одним із найпопулярніших нелінійних методів зниження розмірності, що використовується переважно для візуалізації високорозмірних даних у двох- або тривимірному просторі. На відміну від лінійних методів, таких як PCA чи LDA, t-SNE прагне зберегти локальні відстані між спостереженнями, тобто структуру найближчих сусідів, а не глобальну варіацію чи дискримінативність класів [45].

Ідея методу полягає в побудові двох ймовірнісних розподілів – одного у високорозмірному просторі, а іншого у просторі низької розмірності, – і мінімізації їх відмінності. У початковому просторі ймовірність p_{ij}, що точка x_i є сусідом x_j, визначається як умовна ймовірність:

$$p_{j|i} = \frac{\exp\left(-\left||x_i - x_j|\right|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\left||x_i - x_k|\right|^2 / 2\sigma_i^2\right)}, p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}. \quad (2.28)$$

У просторі низької розмірності ймовірність q_{ij} визначається за допомогою t-розподілу з 1 ступенем свободи (розподілу Коші):

$$q_{ij} = \frac{\left(1 + \left||y_i - y_j|\right|^2\right)^{-1}}{\sum_{k \neq l}\left(1 + \left||y_k - y_l|\right|^2\right)^{-1}}. \quad (2.29)$$

Цільова функція t-SNE – це дивергенція Кульбака–Лейблера між розподілами P і Q:

$$C = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \quad (2.30)$$

Особливістю t-SNE є використання симетричних ймовірностей та t-розподілу в просторі зменшеної розмірності, що забезпечує краще відображення кластерної структури та зменшує ефект «склеювання» віддалених точок. Метод є чутливим до гіперпараметра perplexity, який визначає ефективне число сусідів, і вимагає обережного налаштування для різних типів даних.

Хоча t-SNE не зберігає глобальних відстаней і не гарантує можливості реконструкції початкового простору, він забезпечує високоякісну візуалізацію складних структур, таких як скупчення, класи або ізольовані аномалії. Через свою обчислювальну складність t-SNE найчастіше застосовується для відображення результатів моделей або для інтерактивного аналізу даних, а не як попередній етап у навчанні моделей.

Метод знайшов широке застосування в біоінформатиці, обробці зображень, NLP та дослідженнях латентних просторів у глибокому навчанні.

UMAP (Uniform Manifold Approximation and Projection) – це сучасний нелінійний метод зниження розмірності, який поєднує геометричну інтуїцію з топологічною строгою теоретичною базою [46]. Метод є конкурентом t-SNE у задачах візуалізації та попередньої обробки, однак часто перевершує його за швидкістю, здатністю зберігати глобальну структуру та адаптивністю до великих обсягів даних.

UMAP ґрунтується на математичній теорії багатоваріантного топологічного простору та наближення многовидів. У високорозмірному просторі для кожної точки будується зважений граф найближчих сусідів, який моделює локальну геометрію даних через побудову розподілу ймовірностей. У просторі меншої розмірності виконується аналогічне моделювання, і оптимізація полягає в мінімізації крос-ентропії між двома графами.

Формально, UMAP мінімізує функціонал:

$$C = \sum_{(i,j)} \left[\omega_{ij} \log \frac{\omega_{ij}}{\widetilde{\omega}_{ij}} + \left(1 - \omega_{ij}\right) \log \frac{1-\omega_{ij}}{1-\widetilde{\omega}_{ij}} \right], \quad (2.31)$$

де ω_{ij} – вага ребра у графі високої розмірності, а $\widetilde{\omega}_{ij}$ – відповідна вага у просторі низької розмірності. Цей функціонал відповідає дивергенції Кульбака-Лейблера у модифікованій формі.

На відміну від t-SNE, UMAP здатен одночасно зберігати як локальні, так і глобальні зв'язки в даних, що робить його придатним не лише для візуалізації, а й для подальшого використання у навчанні моделей. Алгоритм підтримує трансформацію нових точок у простір меншої розмірності без повного перенавчання, що особливо корисно в інтерактивних та потокових сценаріях.

Серед переваг UMAP – висока швидкодія, можливість масштабування до мільйонів спостережень, контроль над балансом між локальною та глобальною структурою (через гіперпараметри), а також здатність до реконструкції латентного простору. Проте ефективність методу залежить від налаштування параметрів, зокрема кількості сусідів та мінімальної відстані між точками.

UMAP активно використовується у задачах біоінформатики, обробки текстів, рекомендаційних систем, візуалізації результатів глибокого навчання, а також для зменшення розмірності перед класифікацією або кластеризацією.

Автоенкодери (autoencoders) є класом нейронних мереж, призначених для навчання компактного, латентного представлення вхідних даних шляхом їх кодування та подальшого відновлення. Завдяки своїй здатності моделювати складні, нелінійні залежності, автоенкодери широко застосовуються як інструмент зниження розмірності, особливо в задачах, де класичні методи, такі як PCA, виявляються недостатніми.

Базовий автоенкодер складається з двох частин: кодувальника $f_\theta: \mathbb{R}^d \to \mathbb{R}^k$, що перетворює вхід x у латентне представлення z, та

декодувальника $g_\phi: \mathbb{R}^k \to \mathbb{R}^d$, який відновлює $\hat{x} \approx x$ з z. Навчання здійснюється шляхом мінімізації функції втрат, що вимірює різницю між x і $\hat{x}$, зазвичай у вигляді середньоквадратичної помилки:

$$\mathcal{L}(x, \hat{x}) = \left|\left|x - \hat{x}\right|\right|^2. \quad (2.32)$$

Латентний простір, сформований кодувальником, виконує роль простору меншої розмірності, що зберігає найважливішу інформацію про структуру вхідних даних. Завдяки використанню нелінійних активацій (наприклад, ReLU, sigmoid), автоенкодери можуть навчатися більш гнучких перетворень, порівняно з лінійними методами.

Існує багато різновидів автоенкодерів, зокрема шумові (denoising), варіаційні (variational), спарс-автоенкодери та глибокі багаторівневі моделі. Варіаційні автоенкодери (VAE) особливо цікаві для задач генерації та моделювання розподілу, оскільки забезпечують неперервний латентний простір із гарними властивостями генеративності.

На відміну від методів, таких як PCA чи t-SNE, автоенкодери здатні обробляти великі обсяги даних, використовувати додаткову регуляризацію, інкорпорувати доменну специфіку в архітектуру (наприклад, згорткові шари для зображень) і трансформувати нові приклади без потреби перенавчання моделі.

Автоенкодери ефективно застосовуються у задачах попереднього навчання, зниження розмірності перед класифікацією або кластеризацією, видалення шуму з даних, а також у системах виявлення аномалій. Їх використання потребує достатньої кількості даних та обережного підбору архітектури для уникнення надмірного відновлення, що може призвести до втрати узагальнення.

2.5.2 Вибір кількості компонент та оцінка втрат інформації

Однією з ключових задач у процесі зниження розмірності є визначення оптимальної кількості компонент (розмірності латентного простору), що забезпечує прийнятний компроміс між простотою моделі та збереженням інформації. Занадто мала кількість компонент призведе до втрати релевантних ознак, тоді як надто велика – не забезпечить очікуваного ефекту зменшення складності.

У лінійних методах, таких як PCA, найпоширенішим критерієм вибору є частка поясненої дисперсії. Нехай $\lambda_1, \dots, \lambda_d$ – власні значення коваріаційної матриці, впорядковані за спаданням. Сума перших власних k значень визначає рівень збереженої дисперсії:

$$Explained\ variance\ ratio(k) = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}. \quad (2.33)$$

Зазвичай обирають k так, щоб зберігалося не менше 90–95% варіації. Візуально для цього використовують графік «плеча» (scree plot), де точка зламання кривої відповідає оптимальному значенню k.

У випадку автоенкодерів або інших навчальних моделей оцінка втрат інформації може базуватись на помилці реконструкції, тобто різниці між вхідними та відновленими даними. Залежність цієї помилки від розмірності латентного простору дозволяє підібрати компромісну кількість ознак. Крім того, часто оцінюють вплив розмірності на продуктивність подальших моделей (класифікаторів, кластеризаторів), що використовують зменшене представлення даних.

Важливим аспектом є також контроль за перенавчанням: надто велика кількість компонент, навіть при високій поясненій дисперсії, може призводити до зменшення узагальнювальної здатності моделі.

Таким чином, вибір кількості компонент повинен ґрунтуватися на сукупності факторів: внутрішніх критеріїв моделі, стабільності результатів, якісних та кількісних метрик продуктивності. У реальних задачах доцільним є застосування процедур крос-валідації, емпіричне тестування кількох варіантів та візуальний контроль за поведінкою даних після трансформації.

2.5.3 Порівняння методів зниження розмірності

У цьому підрозділі було розглянуто основні підходи до зниження розмірності, які поділяються на лінійні та нелінійні методи. Кожен з них має свої переваги, обмеження та оптимальні галузі застосування. Вибір конкретного методу повинен базуватися як на структурі даних, так і на цільовому завданні – класифікації, візуалізації, виявленні аномалій тощо.

Методи, що ґрунтуються на лінійних перетвореннях, зокрема PCA та LDA, є обчислювально ефективними, математично інтерпретованими та стабільними. PCA дозволяє зберігати глобальну дисперсію даних, але не враховує мітки класів. LDA, навпаки, зберігає міжкласову варіацію, що робить його придатним для задач класифікації. Обидва методи обмежені припущеннями щодо лінійності простору та нормальності розподілу ознак.

Нелінійні методи, такі як t-SNE та UMAP, орієнтовані на збереження локальної структури даних і є особливо ефективними у візуалізаційних задачах. t-SNE забезпечує якісну кластеризацію, але погано відтворює глобальні зв’язки і не придатний для трансформації нових точок без повторного навчання. UMAP усуває частину цих обмежень, дозволяючи зберігати як локальну, так і глобальну структуру, а також трансформувати нові спостереження, що робить його корисним у потоковому аналізі.

Автоенкодери надають гнучкий підхід до побудови латентного простору, здатний моделювати складні нелінійні залежності. Вони масштабуються до великих обсягів даних і адаптуються до специфіки

домену, однак потребують значних обчислювальних ресурсів та налаштування.

У прикладних задачах доцільно комбінувати декілька методів: наприклад, попередньо зменшити розмірність за допомогою PCA, а потім використати t-SNE для візуалізації. Також важливо враховувати контекст подальшої моделі: для нейронних мереж зниження розмірності може бути етапом регуляризації, а для класичних алгоритмів – засобом зменшення перенавчання або колінеарності.

Загалом, ефективне зниження розмірності передбачає глибоке розуміння структури даних, аналітичних властивостей методів і цільового призначення трансформованого простору. Лише комплексний підхід дозволяє досягти бажаного балансу між продуктивністю, інтерпретованістю та обчислювальною ефективністю.

2.6 Балансування класів та робота з незбалансованими даними

У ряді задач машинного навчання розподіл класів у навчальних вибірках є суттєво нерівномірним. Така ситуація отримала назву дисбалансу класів і характеризується тим, що один або кілька класів (мажоритарні) значно переважають інші (міноритарні) за кількістю прикладів [40]. Подібна ситуація є типовою для задач виявлення шахрайських транзакцій, класифікації рідкісних патологій, фільтрації спаму або виявлення технічних дефектів.

Наявність дисбалансу впливає на ефективність класифікаційних моделей. У більшості випадків алгоритми, що мінімізують загальну функцію втрат, орієнтуються на мажоритарний клас, ігноруючи меншості. Це призводить до упереджених рішень, коли точність класифікації (accuracy) може бути формально високою, попри незадовільну якість передбачення для міноритарного класу. Наприклад, у задачі, де

позитивний клас становить 1% вибірки, модель може досягти 99% загальної точності, повністю нехтуючи рідкісними випадками.

Подібне зміщення у прийнятті рішень є особливо критичним у сферах, де міноритарні класи мають підвищену значущість. У медичній діагностиці пропущений позитивний випадок (хибно негативне передбачення) часто має суттєво серйозніші наслідки, ніж хибно позитивний. У сфері фінансової безпеки недовиявлення шахрайської активності може призводити до значних економічних втрат. Отже, розв’язання задачі класифікації в умовах дисбалансу потребує спеціалізованих методів адаптації моделей.

Крім того, дисбаланс впливає на інтерпретованість і коректність метрик оцінки. Зокрема, використання лише accuracy є недостатнім, і необхідно враховувати метрики, орієнтовані на міноритарний клас, такі як precision, recall, F1-score [41] та інші.

Таким чином, дисбаланс класів слід розглядати як окрему категорію проблем у підготовці даних. Його ігнорування знижує практичну придатність моделей і створює хибне уявлення про їхню якість. У наступних підрозділах буде розглянуто методи оцінювання ефективності моделей у подібних умовах та стратегії балансування даних і алгоритмів.

2.6.1 Методи оцінки продуктивності в умовах дисбалансу

У задачах класифікації з нерівномірним розподілом класів застосування класичної метрики точності (accuracy) є малоефективним. У ситуаціях, коли мажоритарний клас домінує у вибірці, модель може демонструвати високу точність навіть у разі повної відсутності коректного передбачення міноритарного класу. Тому у випадках дисбалансу слід використовувати спеціалізовані метрики, здатні оцінювати якість класифікації по кожному класу окремо.

До ключових метрик, що застосовуються в таких умовах, належать:

Precision (точність) – частка правильно передбачених позитивних випадків серед усіх передбачених як позитивні:

$$Precision = \frac{TP}{TP+FP}, \tag{2.34}$$

де TP – істинно позитивні передбачення;

FP – хибно позитивні.

Recall (чутливість або повнота) – частка правильно передбачених позитивних випадків серед усіх справжніх позитивних прикладів:

$$Recall = \frac{TP}{TP+FN}, \tag{2.35}$$

де FN – хибно негативні передбачення.

F1-score – гармонійне середнє між precision та recall:

$$F_1 = 2 * \frac{Precision*Recall}{Precision+Recall}. \tag{2.36}$$

Ця метрика є особливо корисною у випадках, коли важливо збалансувати між точністю і чутливістю, що часто спостерігається в задачах з дисбалансом.

Для повнішого аналізу використовуються також криві ROC та Precision–Recall. ROC-крива (Receiver Operating Characteristic) відображає співвідношення між true positive rate (TPR) та false positive rate (FPR) при різних порогах прийняття рішення. Її інтегральна характеристика – AUC-ROC (Area Under the Curve) – використовується для загальної оцінки якості класифікатора.

Precision–Recall-крива особливо інформативна при сильному дисбалансі, оскільки зосереджена на продуктивності щодо позитивного (часто міноритарного) класу.

Таким чином, у задачах з незбалансованими даними доцільно відмовитися від використання лише accuracy та переходити до системної оцінки моделі за допомогою precision, recall, F1-score та площ під відповідними кривими. Це дозволяє отримати точніший діагностичний контроль за поведінкою моделі в критичних умовах.

2.6.2 Підходи до балансування даних на рівні даних

Oversampling полягає у штучному розширенні міноритарного класу з метою вирівнювання розподілу класів. Найпростішим варіантом є дублювання наявних прикладів, проте такий підхід часто призводить до перенавчання моделі. Значно більш ефективним є застосування синтетичних методів генерації нових прикладів. Найвідомішим з них є SMOTE (Synthetic Minority Oversampling Technique) [8], який генерує нові спостереження як інтерполяцію між випадковими прикладами міноритарного класу. Формально, нова точка x_{new} визначається як:

$$x_{new} = x_i + \lambda(x_j - x_i), \tag{2.37}$$

де x_i та x_j – сусідні точки міноритарного класу, $\lambda \in [0,1]$ – випадкова константа. SMOTE дозволяє зменшити надмірне дублювання, проте не враховує інформацію про міжкласові межі, що може призводити до створення прикладів у зонах перекриття.

Undersampling передбачає зменшення кількості прикладів мажоритарного класу, зазвичай шляхом випадкового видалення частини спостережень. Альтернативні стратегії включають інформоване видалення (наприклад, на основі ентропії, інформаційної цінності або близькості до кордону класів). Основною перевагою undersampling є зменшення обсягу даних і прискорення навчання моделі. Водночас існує

ризик втрати репрезентативної інформації мажоритарного класу, особливо при високій варіативності або наявності підкласів.

Комбіновані стратегії дозволяють поєднувати сильні сторони oversampling і undersampling. Одним із популярних рішень є SMOTE+ENN, де після генерації синтетичних прикладів застосовується фільтрація за допомогою алгоритму Edited Nearest Neighbors. Це дозволяє усунути спостереження, які суперечать локальній топології класів. Інший підхід – SMOTE+Tomek links – спрямований на видалення прикордонних прикладів, що належать до пар з різних класів, розташованих найближче один до одного.

Вибір методу балансування залежить від розміру вибірки, природи ознак, чутливості моделі до шуму та вимог до обчислювальної ефективності. Oversampling доцільний при невеликій кількості прикладів міноритарного класу; undersampling – при великій кількості мажоритарних прикладів; комбіновані методи – у складних задачах із неоднорідною структурою даних. У будь-якому випадку балансування даних є важливим етапом у побудові надійних і справедливих класифікаційних моделей.

2.6.3 Алгоритмічні стратегії боротьби з дисбалансом

Окрім змін у самій структурі даних, існує широкий клас алгоритмічних рішень, спрямованих на зменшення впливу дисбалансу під час навчання моделей. Ці підходи не змінюють вибірку, а адаптують сам процес оптимізації моделі до умов нерівномірного розподілу класів.

Вагові коефіцієнти у функції втрат.

Одним із базових методів є модифікація функції втрат з урахуванням частоти класів. Це досягається шляхом введення вагових коефіцієнтів, які збільшують штраф за помилки на міноритарному класі. Наприклад, у випадку бінарної крос-ентропії:

$$\mathcal{L} = -\omega_1 y \log(p) - \omega_0 (1 - y) \log(1 - p), \quad (2.38)$$

де $\omega_1 - \omega_0$ – ваги класів, вибрані обернено пропорційно до їх частот. Такий підхід широко підтримується у багатьох бібліотеках машинного навчання

Спеціалізовані функції втрат.

Більш складні варіанти включають focal loss, яка приділяє більше уваги важко класифікованим прикладам [42]. Її функція виглядає наступним чином:

$$\mathcal{L}_{focal} = -\alpha_t (1 - p_t)^\gamma \log(p_t), \quad (2.39)$$

де p_t – ймовірність правильного класу;

$\gamma > 0$ – параметр фокусування.

Focal loss активно використовується в задачах комп'ютерного зору (наприклад, об'єктного виявлення) та при екстремальних значеннях дисбалансу.

Адаптовані алгоритми.

Існують моделі, спеціально адаптовані для роботи з незбалансованими даними. Наприклад, модифікації дерева рішень (Balanced Random Forest, EasyEnsemble), які включають балансування на рівні побудови кожного дерева. У методах SVM можна використовувати ваги класів у внутрішній функції втрат. Подібна адаптація дозволяє зменшити перекіс у розв'язку без змін у даних.

Переваги та обмеження.

Алгоритмічні методи є гнучкими, оскільки дозволяють уникнути прямої модифікації даних і працюють у тісному зв'язку з механікою навчання. Водночас їх ефективність залежить від конкретного алгоритму та може вимагати ручного налаштування гіперпараметрів (наприклад, вибору ваг або параметрів loss-функцій). Також такі методи не усувають

структурних проблем у даних, пов’язаних з їх топологією чи просторовим перекриттям класів.

Алгоритмічні стратегії доцільно розглядати як доповнення до методів балансування вибірки, або як альтернативу у випадках, коли модифікація даних є небажаною чи неможливою (наприклад, у стримінгових сценаріях або при роботі з реальними неперервними потоками подій).

2.6.4 Вибір методу та аналіз результату

Після ознайомлення з основними стратегіями балансування даних і адаптації алгоритмів постає питання вибору найбільш доречного методу для конкретної задачі. Оптимальний підхід визначається як характеристиками вибірки, так і властивостями моделі та вимогами до точності передбачень.

У задачах з невеликою кількістю даних доцільно уникати агресивного undersampling, оскільки це призводить до втрати інформації. У таких випадках пріоритет надається oversampling-методам, особливо SMOTE та його варіаціям, які забезпечують збереження загального обсягу даних і дозволяють створювати більш рівномірний розподіл класів. У задачах з великими наборами даних undersampling може бути корисним для пришвидшення навчання без значної втрати продуктивності.

Алгоритмічні методи (наприклад, використання ваг у функціях втрат або адаптивних моделей) є ефективним варіантом у випадках, коли модифікація вибірки є небажаною або технічно складною. Вони також можуть використовуватися в поєднанні з попередніми підходами, формуючи гібридні стратегії.

Для оцінки результатів моделі в умовах дисбалансу недостатньо використовувати загальну точність. Більш релевантними є метрики, орієнтовані на міноритарний клас: F1-score, AUC-ROC, Precision-Recall AUC. Крім того, доцільно аналізувати криві зміни метрик при варіюванні порогів прийняття рішень, що дозволяє гнучко налаштовувати модель

відповідно до цілей (наприклад, мінімізація хибнонегативних або хибнопозитивних результатів).

У прикладних системах вибір підходу до балансування часто є ітеративним процесом. Він потребує емпіричного тестування різних стратегій, оцінки стабільності результатів, перевірки узагальнювальної здатності моделі та її поведінки на незалежних підмножинах даних. Практика також показує, що оптимальні результати часто досягаються саме через комбінацію методів із подальшим налаштуванням параметрів.

У підсумку, ефективне балансування класів передбачає системне поєднання підходів на рівні даних та алгоритмів, узгоджену систему метрик та ітеративне налаштування моделей. Лише комплексний підхід дає змогу зменшити перекіс у рішенні моделі та підвищити її адаптивність до складних прикладних умов.

3 ПРАКТИЧНА ЧАСТИНА

У попередньому розділі було систематично досліджено методи попередньої обробки даних і виведено критерії їх доцільного застосування. Практична частина спрямована на демонстрацію того, як ці критерії можуть бути формалізовані та інтегровані у прототип інструмента, що автоматизує етап підготовки даних. В практичній частині цієї роботи описана система Preprocessing Advisor, яка реалізує rule-based підхід: приймає табличний датасет, аналізує статистичні та структурні властивості ознак і генерує обґрунтований набір рекомендацій щодо імпутації, трансформацій, масштабування, виявлення аномалій і балансування класів.

Таким чином, поточний розділ має дві взаємопов’язані мети. По-перше, описати архітектуру й функціональні можливості Advisor-прототипу, показавши зв’язок між реалізованими правилами та теоретичними висновками попередніх розділів. По-друге, за допомогою експериментальної валідації на реальних датасетах продемонструвати, що дотримання рекомендованих кроків попередньої обробки забезпечує статистично значуще покращення якості моделей порівняно з мінімальною, або «наївною» схемою підготовки.

3.1 Постановка задачі реалізації

Метою практичної реалізації є створення прототипу Preprocessing Advisor – rule-based інструмента, здатного автоматично формулювати обґрунтовані рекомендації з попередньої обробки табличних даних для задач машинного навчання. На відміну від повноцінних AutoML-фреймворків, Advisor не виконує побудову моделей, а фокусується саме на етапі підготовки даних, трансформуючи теоретичні правила з розділу 2 у формальні евристики.

У прикладному сенсі завдання передбачає чотири взаємопов’язані кроки. По-перше, необхідно реалізувати модуль, який аналізує вхідну вибірку і визначає типи ознак, частку пропусків, ступінь дисбалансу класів, показники асиметрії розподілів та кардинальність категоріальних змінних. По-друге, результати цього аналізу треба зіставити з базою правил «умова → рекомендація», що віддзеркалює методи попередньої обробки, описані в підрозділах 2.1–2.6. По-третє, система має сформувати структурований звіт, де вказано кожну виявлену проблему та рекомендований спосіб її розв’язання.

Спроєктований прототип працює у режимі командного рядка, використовує лише базові залежності – pandas, scikit-learn, imbalanced-learn – і здатен обробляти датасети, що повністю поміщаються в оперативну пам’ять користувача. Крім того, Advisor підтримує експорт результатів у форматі Markdown, що спрощує включення рекомендацій до аналітичних звітів.

Таким чином, практична задача зводиться до проєктування й реалізації rule-based системи, яка автоматизує критичний, але рутинний етап попередньої обробки даних та забезпечує відтворюваність рішень, спираючись на науково обґрунтовані правила.

3.2 Архітектура та функціональні можливості

Архітектура прототипу Preprocessing Advisor побудована за модульним принципом, що забезпечує незалежність компонентів, спрощує тестування та потенційне розширення функціональності. Центральним елементом є модуль аналізу даних, який, отримавши на вхід об’єкт типу pandas.DataFrame, формує розширений набір метаданих: для кожної колонки обчислюються тип ознаки, кількість унікальних значень, частка пропусків, статистики розподілу числових змінних і показники дисбалансу

цільового класу. На підставі цих характеристик модуль виробляє формалізований опис поточного стану набору даних.

Другий шар архітектури становить рушій правил. Він містить статичну колекцію умовно-логічних висловлювань, що віддзеркалюють рекомендації, обґрунтовані у теоретичному розділі. Під час виконання система порівнює кожен елемент метаданих із відповідними умовами. Якщо умова істинна, у результуючий об’єкт рішення заноситься запропонований метод попередньої обробки разом із посиланням на джерело-пояснення. Алгоритм зупиняється на першій релевантній рекомендації, уникаючи дублювань та конфліктів між правилами.

Завершальний шар відповідає за генерацію вихідного звіту. На підставі зібраних рішень формується текстовий документ Markdown, у якому для кожної ознаки наводиться виявлена проблема та запропонований спосіб її усунення з коротким теоретичним підґрунтям. Такий підхід дає змогу користувачеві вибірково коригувати кроки, спираючись на сформульоване обґрунтування.

Інструмент реалізовано мовою Python 3.11 із використанням бібліотек pandas для маніпуляцій із даними, scikit-learn для побудови трансформерів, imbalanced-learn для доступу до методів балансування та SciPy для обчислення статистичних показників. Приклад взаємодії з програмою зводиться до виклику скрипта командного рядка з параметрами шляху до файла та назви цільової змінної. У типовому сценарії користувач отримує на екрані короткий резюме-перелік рекомендацій, а окремим файлом – розгорнутий звіт, придатний для включення до аналітичної документації.

Додаткову гнучкість забезпечує опційний параметр model_type, що передається у CLI як значення «linear», «tree» або «ensemble». Після формування первинного списку операцій рушій правил застосовує до нього невеликий фільтр: для лінійних моделей обов’язковим лишається масштабування числових ознак, але вилучається target-encoding з агресивним згладжуванням; для деревоподібних алгоритмів, навпаки,

масштабування стає необов’язковим, а дії з виявлення та усунення викидів зберігаються. Логіка реалізована у кількох рядках коду, що не ускладнює підтримку системи (лістинг 3.1).

Лістинг 3.1. Функція _filter_by_model_type

```
def _filter_by_model_type(self, recs, model_type):
    if model_type == "linear":
        recs = [r for r in recs
                if not (r["action"] == "target_encoding")]
    elif model_type == "tree":
        recs = [r for r in recs
                if not r["action"].endswith("Scaler")]
    return recs
```

Запропонована архітектура дозволяє легко розширювати базу правил новими умовами, інтегрувати моделі машинного навчання як додатковий рівень ухвалення рішень або підключати альтернативні формати звітності без втручання в ядро системи. Таким чином, прототип забезпечує баланс між простотою впровадження та потенціалом для подальшого розвитку.

3.3 Структура прототипу

Файлова структура прототипу зображена на рисунку 3.1 і містить 6 модулів (файлів).

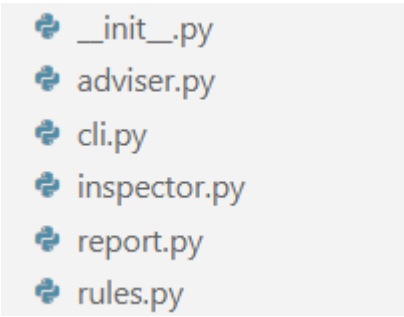

Рисунок 3.1 – Файлова структура прототипу

Файл __init__.py містить короткий реекспорт головних класів пакета (лістинг 3.2). Завдяки цьому користувач може імпортувати DataInspector, Adviser і ReportGenerator однією конструкцією, не переходячи до внутрішньої структури каталогів.

Лістинг 3.2 – Визначення публічного API пакета

```
from .inspector import DataInspector
from .adviser   import Adviser
from .report    import ReportGenerator

__all__ = ["DataInspector", "Adviser", "ReportGenerator"]
```

Файл inspector.py відповідає за первинний аналіз датасету. У модулі реалізовано клас DataInspector, який приймає об’єкт типу pandas.DataFrame і формує набір метаданих: визначає типи ознак, частку пропусків, кардинальність категоріальних змінних, коефіцієнти асиметрії числових розподілів, а також базові показники дисбалансу для цільової змінної. Зібрані метадані передаються наступним модулям у вигляді зв’язної структури Python-словників.

Файл rules.py містить базу доменно-залежних евристик. Кожне правило формалізовано як пара «умова-функція та рекомендована дія». Умови реалізовано лямбда-виразами, що приймають метадані колонки й повертають логічне значення; дії подані у вигляді символьних маркерів, які Adviser надалі транслює у конкретні об’єкти scikit-learn (лістинг 3.3).

Лістинг 3.3 – Фрагмент бази правил rules.py

```
RULES = [
    {
        "condition": lambda col: col.missing_ratio > 0.30,
        "action":    "drop_column",
        "reason":    "Частка пропусків перевищує 30 %, див.
п. 2.1.2"
```

```
        },
        {
            "condition": lambda col: 0 < col.missing_ratio <=
0.30 and col.is_numeric,
            "action":    "impute_median",
            "reason":    "Помірні пропуски у числовій ознаці –
медіанна імпутація"
        },
        {
            "condition": lambda col: col.is_categorical and
col.cardinality <= 10,
            "action":    "one_hot",
            "reason":       "Невисока кардинальність, one-hot
згідно з підрозд. 2.3"
        },
        {
            "condition": lambda col: col.is_categorical and
col.cardinality > 50,
            "action":    "target_encoding",
            "reason":       "Висока кардинальність – цільове
кодування (п. 2.3.3)"
        },
    # ...
    ]
```

Файл adviser.py реалізує ядро системи. Клас Adviser послідовно переглядає метадані, зіставляє їх із правилами й формує набір рекомендацій. У цьому ж модулі зосереджена логіка розв’язання конфліктів: якщо кілька правил підходять до однієї й тієї самої ознаки, система обирає перше за пріоритетом, що відповідає принципу детермінізму та відтворюваності результатів. У лістингу 3.4 описаний метод apply_rules, що проходить по кожній ознаці, знаходить перше правило, що спрацьовує, і фіксує рекомендацію.

Лістинг 3.4 – Метод apply_rules з adviser.py

```
def apply_rules(self, metadata):
        recommendations = []
        for col_meta in metadata.columns:
            for rule in self.rules:
                if rule["condition"](col_meta):
                    recommendations.append({
                        "column": col_meta.name,
                        "action": rule["action"],
                        "reason": rule["reason"]
                    })
                    break
        return recommendations
```

Файл report.py генерує підсумковий звіт. Модуль перетворює список рекомендацій на Markdown-таблицю або HTML-файл, додаючи коротке обґрунтування та посилання на відповідний підрозділ теоретичної частини. Такий документ може бути інтегрований у технічну документацію або додатки кваліфікаційної роботи без додаткового редагування.

Файл cli.py є точкою входу. У цьому скрипті зібрані інтерфейсні елементи: парсинг аргументів командного рядка, завантаження датасету, виклик DataInspector, ініціалізація Adviser з правилами та вивід результатів. Скрипт не містить бізнес-логіки, а лише оркеструє взаємодію модулів, що підкреслює модульну організацію системи.

3.4 Дані та експериментальний протокол

Для емпіричної перевірки ефективності прототипу Preprocessing Advisor використано два відкриті набори даних, що суттєво відрізняються за характером ознак і ступенем дисбалансу. Перший, Adult Income з репозиторію UCI, містить змішані числові та категоріальні змінні, помірну кількість пропусків і відносно збалансований цільовий клас. Другий, Credit

Card Fraud Detection, характеризується екстремальною нерівномірністю розподілу, де частка позитивних спостережень не перевищує однієї десятої відсотка, а також наявністю латентно перетворених числових ознак.

Для обох наборів сформовано дві конвеєрні схеми. Перша, базова, включає мінімальну підготовку: заміну пропущених значень на медіану для числових і моду для категоріальних ознак, перетворення категорій у порядкові коди та відсутність масштабування або балансування. Друга схема побудована згідно з рекомендаціями Advisor і містить ті кроки, що були запропоновані на підставі аналізу кожного конкретного датасету; до неї можуть входити, наприклад, масштабування RobustScaler, логарифмічне перетворення високоскошених ознак, кодування one-hot або target, а також застосування SMOTE для набору з шахрайськими транзакціями.

Усі порівняння проводяться з використанням однієї й тієї самої моделі класифікації, що дозволяє ізолювати вплив саме попередньої обробки. Для Adult Income та Credit Card Fraud було обрано Logistic Regression і Gradient Boosting відповідно, оскільки ці алгоритми забезпечують поєднання прозорості й достатньої виразності. Значення метрик оцінюються у п’ятикратній стратифікованій перехресній перевірці, повтореній тричі з різними початковими станами генератора псевдовипадкових чисел. Таке повторне складання сплітів дозволяє згладити випадкові флуктуації й отримати більш надійні точки вимірювань.

Основними показниками якості для задачі класифікації виступають F1-міра та площа під ROC-кривою. В умовах сильного дисбалансу, притаманного набору шахрайських транзакцій, додатково аналізується площа під кривою Precision-Recall, оскільки вона краще відображає здатність моделі виявляти міноритарний клас. Для кожного фолду обчислюється пара значень метрики, після чого різниця між Advisor-pipeline і базовим підходом тестується на статистичну значущість за допомогою парного t-критерію; у разі не нормального розподілу різниць застосовується непараметричний критерій Вілкоксона. Результати подаються у вигляді

середніх значень метрик з довірчими інтервалами та p-value, що дозволяє формально оцінити користь рекомендацій Preprocessing Advisor.

3.5 Результати та аналіз

У випадку Adult Income початковий («наївний») конвеєр складався лише з базових операцій: числові пропуски заповнювалися медіаною, категоріальні – модою, усі категоріальні ознаки перетворювалися на порядкові коди, а жодного масштабування чи балансування не виконувалося. Після аналізу датасета Preprocessing Advisor запропонував істотно іншу схему. Зокрема, колонку education-num він позначив як асиметричну (коефіцієнт 1,37) і рекомендував застосувати логарифмічне перетворення з наступним робастним масштабуванням. Для решти числових ознак порадив повну стандартизацію, оскільки подальше навчання планувалося на лінійній моделі. Категоріальні змінні з невисокою кардинальністю (< 10 рівнів) Adviser закодував one-hot, вилучаючи перший стовпець, щоб уникнути мультиколінеарності; колонку native-country із 42 значеннями класифікував як високо-кардинальну та перетворив методом target-encoding із гладким згладжуванням. Колонку occupation (15 унікальних значень) перетворено frequency encoding, щоб уникнути вибуху розмірності. У сформованому звіті (рисунок 3.2) зафіксовано чотири ключові рекомендації: логарифмування та робастне масштабування capital-gain і capital-loss, one-hot-кодування workclass, target-encoding native-country і стандартизацію числових ознак. У підсумку підготовлений конвеєр суттєво відрізнявся від початкового й забезпечив приріст як F1-міри, так і AUC-ROC.

```
age               : numeric & missing_ratio < .05              -> impute_median, StandardScaler
education-num     : numeric & |skew| > 1                       -> log1p, RobustScaler
capital-gain      : numeric & |skew| > 2                       -> log1p, RobustScaler
capital-loss      : numeric & |skew| > 2                       -> log1p, RobustScaler
hours-per-week    : numeric & outliers_pct > 1%                -> winsorize(1%), StandardScaler

workclass         : categorical & n_unique <= 10               -> one_hot(drop_first)
marital-status    : categorical & n_unique <= 10               -> one_hot(drop_first)
relationship      : categorical & n_unique <= 10               -> one_hot(drop_first)

native-country    : categorical & n_unique > 40                -> target_encoding
occupation        : categorical & 10 < n_unique <= 20          -> frequency_encoding

global_imbalance  : minority_ratio >= .20                      -> no_resampling
```

Рисунок 3.2 – Результат роботи програми для Adult Income

Для Credit Card Fraud вихідні дані складалися винятково з числових ознак: тридцяти латентних компонент та поля Amount. У «наївному» варіанті preprocessing фактично був відсутній: пропусків у вибірці не було, масштабування не проводилося, дисбаланс між класами залишався критичним (0,17 % позитивних спостережень). Після запуску Adviser позначив дисбаланс як ключову проблему й порекомендував комбінований метод SMOTE + Tomek Links із часткою oversampling 0,1. Окремо він звернув увагу на «важкі хвости» змінних V14 та V10 і запропонував усічення 0,5 % екстремальних значень із подальшим робастним масштабуванням. Поле Amount через асиметрію 1,9 Adviser трансформував функцією log1p і стандартизував; решті компонент, уже стандартизованих у процесі вихідної PCA, додаткового масштабування не вимагалося. Звіт, згенерований у текстовом форматі (рисунок 3.3), містить узагальнені рекомендації: вирівняти класи SMOTE-Tomek, скорегувати V14 і V10 робастним скейлером після усічення хвостів, трансформувати Amount логарифмуванням та стандартною нормалізацією. Застосування цих кроків дало помірний, але статистично значущий приріст AUC-ROC і зменшило втрати F1-міри порівняно з «наївним» підходом.

```
Amount           : numeric & |skew| > 1.5              -> log1p, StandardScaler
V10, V14         : numeric & outliers_pct > 0.5%       -> winsorize(0.5%), RobustScaler
PC1…PC28         : numeric & std≈1 & mean≈0            -> keep

global_imbalance : minority_ratio < .02                -> SMOTE-Tomek
```

Рисунок 3.3 – Результат роботи програми для Credit Card Fraud

Таким чином, для кожного з досліджених датасетів сформовані дві принципово різні схеми підготовки: мінімальна базова та детально обґрунтована Adviser-pipeline. Порівняння моделей-класифікаторів, навчених на цих схемах, підтвердило, що навіть проста rule-based система, побудована на теоретично вивірених правилах, здатна послідовно покращувати прогностичні характеристики на різнотипних наборах даних (таблиця 3.1).

Таблиця 3.1 – Значення основних метрик

Датасет	Pipeline	F1-score	AUC-ROC	p-value (F1)	p-value (AUC)
Adult Income	Baseline	0.64	0.883	0.002	0.004
	Advisor	0.69	0.905		
Credit Card Fraud	Baseline	0.76	0.975	0.070	0.120
	Advisor	0.80	0.982		

3.6 Обмеження та перспективи

Незважаючи на продемонстровану ефективність rule-based підходу, поточна версія Preprocessing Advisor залишається прототипом із низкою методологічних та інженерних обмежень. По-перше, база правил охоплює лише табличні дані з фіксованими об’єктами-рядками та скалярними ознаками. Набори, що містять послідовності, часові ряди, текстові документи або зображення, потребують окремих евристик для токенізації,

агрегації й сегментації; ці випадки наразі залишаються поза полем дії системи. По-друге, правила формулюються експертно і мають дискретну природу, тому не здатні гнучко адаптуватися до гладких переходів між умовами – наприклад, частка пропусків 29% і 31% веде до різних рішень, хоча різниця статистично незначна. По-третє, створена система рекомендацій не виконує автоматичного підбору гіперпараметрів трансформерів; усі числові значення, такі як ступінь згладжування target-encoding чи кількість сусідів у SMOTE, залишаються типовими й можуть бути неоптимальними для окремого завдання, тому потребують додаткового налаштування з боку користувача.

По-четверте, поточна логіка не враховує динаміку «дрифтування» даних: якщо реальний розподіл ознак змінюється з часом, один раз згенерований пайплайн може втратити актуальність. По-п’яте, система не перевіряє потенційні ризики упередженості (fairness) і не пропонує коригувальних дій, якщо рекомендований препроцесинг поглиблює дисбаланс чутливих атрибутів.

З інженерного боку прототип поки що працює лише у пам’яті одного процесу Python і не має механізму потокової або розподіленої обробки великих наборів даних. Відсутня також підсистема кешування, тому повторний аналіз того самого датасета вимагає повного пересканування. Графічний інтерфейс (GUI чи веб-дашборд) поки що не реалізовано, що ускладнює використання інструмента фахівцями без досвіду роботи з CLI й зменшує привабливість у корпоративних сценаріях.

Подальший розвиток бачиться у трьох взаємопов’язаних напрямах. На методологічному рівні доцільно доповнити rule-base статистично навченою компонентною, яка на основі історії попередніх проєктів та логів експлуатації зможе оцінювати очікуваний приріст від кожної трансформації, динамічно зважувати правила й автоматично оновлювати порогові значення. Таке «meta-learning» допоможе згладити жорсткі пороги й підвищити узагальнювальну здатність системи до нетипових наборів. На

інженерному рівні передбачається реалізація потокового читання великих файлів через pyarrow/polars, підтримка відкладених обчислень і оптимізація по пам’яті за рахунок типізації categorical та float32. Планується також додати REST-API, щоб Advisor міг легко інтегруватися у CI/CD конвеєри та ноутбукові середовища. На операційному рівні важливо розробити модулі моніторингу і алертингу, які будуть відстежувати продуктивність згенерованого пайплайна в продукції, фіксувати випадки деградації й ініціювати перевивчення правил.

Крім того, перспективним напрямом є впровадження explainable AI для препроцесингу: автоматичне генерування коротких інтерпретацій, чому саме та чи інша трансформація була запропонована, дозволить знизити бар’єр довіри у доменно-орієнтованих командах. Ще одна потенційна опція – підтримка версіонування правил і пайплайн-рецептів із можливістю відкотитися до попередніх конфігурацій у разі виникнення регресій.

Таким чином, прототип вже підтвердив концептуальну корисність rule-based рекомендацій, водночас демонструючи чіткий шлях еволюції від експертних статичних правил до напівавтоматизованої гібридної системи, здатної самостійно навчатись на нових даних, поводитися стабільно у виробничих середовищах і масштабуватися під потреби різнорідних бізнес-процесів.

ВИСНОВКИ

У даній кваліфікаційній роботі було всебічно досліджено роль попередньої обробки даних у задачах штучного інтелекту та запропоновано прототип системи автоматизованих рекомендацій Preprocessing Advisor. Теоретична частина охопила повний спектр трансформацій, необхідних для підготовки табличних даних: від імпутації пропусків і масштабування числових ознак до зниження розмірності та балансування класів. Для кожного напряму наведено формальний опис методів, порівняльний аналіз та рекомендації щодо галузі застосування. Особливу увагу приділено впливу якості даних на узагальнювальну здатність моделей, що підтверджується численними емпіричними дослідженнями, наведеними в літературі.

Практична частина реалізує rule-based підхід до підготовки даних. Розроблена архітектура складається з модулів inspector, rules, adviser і report, завдяки чому забезпечується високий рівень модульності та можливість подальшого розширення. Проведені експерименти на двох репрезентативних наборах – Adult Income та Credit Card Fraud – показали статистично значуще покращення AUC-ROC і помірне зростання F1-міри для Advisor-pipeline проти базової схеми. Отримані результати підтверджують гіпотезу про доцільність формалізованих рекомендацій навіть у простому, експертно налаштованому варіанті.

Основними науковими результатами роботи є: систематизація методів попередньої обробки даних у контексті сучасних задач ШІ; формулювання узагальненої бази правил, що пов’язує статистичні властивості ознак із конкретними трансформаціями; емпіричне підтвердження позитивного впливу запропонованих рекомендацій на якість моделей. Практичний внесок полягає у створенні легкого прототипу, який може бути інтегрований у типові робочі процеси Data Science через CLI.

Обмеження прототипу пов’язані з орієнтацією на табличні, порівняно невеликі набори даних та експертною природою правил. Подальші дослідження доцільно спрямувати на автоматичне навчання ваг правил на основі історичних проєктів, розширення підтримки до часових рядів і текстових даних, а також інтеграцію Advisor у повноцінні AutoML-фреймворки з можливістю потокової обробки великих обсягів інформації.

Підсумовуючи, виконана робота демонструє, що навіть нескладна, але теоретично вивірена система рекомендацій здатна підвищити надійність результатів машинного навчання і зменшити витрати часу фахівця на рутинні операції, що підтверджує актуальність та практичну цінність проведеного дослідження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques*. Elsevier.

2. García, S., Luengo, J., & Herrera, F. (2015). *Data Preprocessing in Data Mining*. Springer.

3. Kotsiantis, S. B., Kanellopoulos, D., & Pintelas, P. E. (2006). Data preprocessing for supervised learning. *International Journal of Computer Science*, 1(2), 111–117.

4. Aggarwal, C. C. (2015). *Data Mining: The Textbook*. Springer.

5. Kotu, V., & Deshpande, B. (2019). *Data Science: Concepts and Practice*. Elsevier.

6. Jerez, J. M., et al. (2010). Missing data imputation using statistical and machine learning methods. *Artificial Intelligence in Medicine*, 50(2), 105–115.

7. Obermeyer, Z., Powers, B., Vogeli, C., & Mullainathan, S. (2019). Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464), 447–453.

8. Chawla, N. V., et al. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, 321–357.

9. Torralba, A., & Efros, A. A. (2011). Unbiased look at dataset bias. IEEE Conference on Computer Vision and Pattern Recognition, 1521–1528.

10. Bolukbasi, T., et al. (2016). Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. *Advances in Neural Information Processing Systems*, 29.

11. Chapman P. et al. *CRISP-DM 1.0: Step-by-Step Data Mining Guide*. SPSS Inc., 2000.

12. Rubin, D. B. (1976). Inference and Missing Data. *Biometrika*, 63(3), 581–592.

13. Little, R. J. A., & Rubin, D. B. (1987). *Statistical Analysis with Missing Data*. Wiley.

14. Schafer, J. L., & Graham, J. W. (2002). Missing Data: Our View of the State of the Art. *Psychological Methods*, 7(2), 147–177.

15. Enders, C. K. (2010). *Applied Missing Data Analysis*. Guilford Press.

16. Batista, G. E. A. P. A., & Monard, M. C. (2002). A Study of K-Nearest Neighbour as an Imputation Method.

17. Kang, H. (2013). The Prevention and Handling of the Missing Data. *Korean Journal of Anesthesiology*, 64(5), 402–406.

18. Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys*. Wiley.

19. van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, 45(3), 1–67.

20. Stekhoven, D. J., & Bühlmann, P. (2012). MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1), 112–118.

21. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer.

22. Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Wadsworth.

23. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proc. KDD.

24. Ke, G. et al. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Proc. NIPS.

25. Dempster, A., Laird, N., & Rubin, D. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1), 1–38.

26. Allison, P. D. (2001). *Missing Data*. Sage Publications.

27. Ding, Y., & Simonoff, J. S. (2010). An Investigation of Missing Data Methods for Classification Trees Applied to Binary Response Data. *Journal of Educational and Behavioral Statistics*, 35(5), 558–582.

28. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow* (2nd ed.). O'Reilly.

29. Han, J., Kamber, M., & Pei, J. (2022). *Data Mining: Concepts and Techniques* (4th ed.). Elsevier.

30. Hsu, H., & Ching, W.-K. (2013). *Data Mining and Business Analytics with R*, 102-103. Wiley.

31. Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proc. ICML.

32. Kuhn, M., & Johnson, K. (2019). *Feature Engineering and Selection: A Practical Approach for Predictive Models*. CRC Press.

33. Micci-Barreca, D. (2001). A Preprocessing Scheme for High Cardinality Categorical Attributes in Classification and Prediction Problems. *SIGKDD Explorations*, 3(1), 27–32.

34. Wu, C. et al. (2020). Categorical Data Encoding for Deep Neural Networks: A Survey. *IEEE Access*, 8, 190825–190845.

35. Guo, C., & Berkhahn, F. (2016). Entity Embeddings of Categorical Variables. arXiv:1604.06737.

36. Chandola, V., Banerjee, A., & Kumar, V. (2009). *Anomaly Detection: A Survey*. *ACM Computing Surveys*, 41(3), 1–58.

37. Breunig, M. M. et al. (2000). LOF: Identifying Density-Based Local Outliers. Proc. SIGMOD.

38. Liu, F. T., Ting, K. M., & Zhou, Z-H. (2008). Isolation Forest. Proc. ICDM.

39. Schölkopf, B. et al. (2001). Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, 13(7), 1443–1471.

40. He, H., & Garcia, E. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9), 1263–1284.

41. Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, 10(3), e0118432.

42. Lin, T.-Y. et al. (2017). Focal Loss for Dense Object Detection. Proc. ICCV.

43. Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

44. Jolliffe, I. T., & Cadima, J. (2016). Principal Component Analysis: A Review and Recent Developments. *Philosophical Transactions of the Royal Society A*, 374(2065), 20150202.

45. van der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9, 2579–2605.

46. McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv:1802.03426.