Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота

за освітнім ступенем «бакалавр»

на тему:

ПРОГРАМНА СИСТЕМА ЗАБЕЗПЕЧЕННЯ ІНЖЕНЕРІЇ ДАНИХ ДЛЯ МАШИННОГО НАВЧАННЯ. ВЗАЄМОДІЯ З ВЕЛИКИМИ ДАНИМИ BIG DATA

Виконала:
здобувачка вищої освіти 4 курсу
спеціальності 121 Інженерія програмного забезпечення
заочної форми навчання
Літвін Анастасії Русланівни

Науковий керівник:
к.т.н., доцент Сяський В. А.

Рівне – 2026

ЗМІСТ

ВСТУП

Стрімке зростання обсягів цифрової інформації є однією з визначальних тенденцій сучасної епохи. За прогнозами аналітиків IDC (International Data Corporation) та Seagate, до 2025 року розміри світової датасфери мали перевищити 175 зетабайт даних [15]. Сьогодні, завдяки розвитку цифрових технологій, за оцінками, щохвилини в інтернеті обробляється 1,74 мільйона гігабайтів даних [49]. Зберігати таку кількість інформації складно, натомість вилучення цілісного результату з таких масивів є ще складнішим завданням, яке не можна вирішити за допомогою традиційних реляційних систем управління базами даних.

Для вирішення цих проблем сформувалася окрема інженерна дисципліна – інженерія даних (data engineering). Джо Рейс і Метт Гауслі визначають її як розробку впровадження, імплементації та підтримки систем і процесів, що приймають сирі дані і виробляють якісну, послідовну інформацію, придатну для аналізу та машинного навчання [16]. Дата інженер – це фахівець, що працює з даними у їх “сирому вигляді” (raw data). Він відповідає за збір, зберігання, трансформацію та підготовку цих даних для споживача: наприклад, дата-саєнтиста чи інших спеціалістів.

Інженерія даних знаходиться на перетині безпеки, управління даними, DataOps (операційних процесів даних), архітектури даних, оркестрації даних, і розробки програмного забезпечення. Інженер по обробці даних управляє життєвим циклом даних, починаючи зі збору даних від джерел даних і закінчуючи наданням даних для різних сценаріїв використання, таких як аналіз і машинне навчання.

Актуальність теми: інженерія даних важлива, оскільки дозволяє компаніям використовувати дані, наявні на кожному етапі бізнесу, для вирішення критичних задач. Мартін Клеппман у праці про системи, інтенсивні щодо даних, називає надійність, масштабованість і зручність супроводу трьома ключовими принципами будь-якої сучасної системи обробки інформації [17]. Низька якість даних та труднощі з доступом до них призводять до нераціональних витрат,

затримок у запуску продуктів і втрати потенційного прибутку. Вирішенням цих проблем є побудова якісної інженерії даних, яка дозволяє оптимізувати інформаційні процеси організації [25].

В даній роботі поставлені наступні **цілі**:

- визначити та встановити концепцію інженерії даних та великих даних зокрема;
- дослідити індивідуальну роль інженерії даних з допомогою відповідних систем в машинному навчанні;
- встановити ключові етапи, інструменти та вимоги до систем даних для машинного навчання;
- знайти переваги та недоліки у підходах управлінням даними;
- розкрити характеристики великих даних та відобразити їх структуру.

Мета роботи: на основі дослідження наукових праць, тематичних статей та технічних специфікацій розглянути різноманітні підходи до системи інженерії даних, їх застосування у наявному програмному середовищі та взаємодію з великими масивами даних (Big Data).

Об’єкт дослідження: система забезпечення інженерії даних для машинного навчання. Взаємодія з великими даними Big Data.

Предмет дослідження: конвеєри даних (data pipelines), що функціонують за допомогою інструментів для машинного навчання та управління великими даними.

Практичне значення: результати роботи можуть бути використані для побудови реальних систем виявлення аномалій у фінансових транзакціях, а систематизований матеріал – для написання рефератів, проведення лекційних та семінарських занять, уроків у навчальних закладах загальноосвітнього типу.

Структура роботи: бакалаврська робота складається зі вступу, трьох розділів, висновку та списку використаних джерел.

РОЗДІЛ 1. ПРОГРАМНА СИСТЕМА ЗАБЕЗПЕЧЕННЯ ІНЖЕНЕРІЇ ДАНИХ ДЛЯ МАШИННОГО НАВЧАННЯ

1.1. Концепція інженерії даних

1.1.1. Визначення інженерії даних

Інженерія даних – це процес створення та проєктування інфраструктури даних, що дозволяє організаціям ефективно працювати з різнорідними, часто неструктурованими потоками інформації з численних джерел. Без неї практично неможливо опрацювати й осмислити колосальні обсяги наявних даних. Рейс і Гауслі у “Fundamentals of Data Engineering” описують цю дисципліну як таку, що охоплює повний “lifecycle” даних – від їх генерації в джерелі до готових аналітичних продуктів, придатних для бізнесу та науки [16].

Перехід до спільних баз даних змінив підхід до розробки інформаційних систем. Було доведено, що логічна структура даних є автономною і не залежить від конкретних програм їхньої обробки. Це наукове положення відоме як принцип логічної незалежності даних. Через постійне ускладнення моделей предметної області, початковий аналіз даних став обов’язковим етапом, тому сучасне проєктування ПЗ насамперед починається з детального дослідження інформаційних сутностей та зв’язків між ними [26]. Кімбалл і Росс, описуючи розвиток сховищ даних, підкреслюють, що проектування на основі бізнес-процесів і моделей даних є фундаментом успішних аналітичних систем [18].

Сучасна інженерія даних базується на розподілені архітектури, паралельні файлові системи та моделі паралельних обчислень, зокрема MapReduce і хмарні платформи. Це дозволяє ефективно оперувати масивами з мінімальною затримкою. Основним завданням цієї галузі є трансформація сирих даних у високоякісні, структуровані та консистентні інформаційні продукти шляхом впровадження механізмів очищення, дедуплікації, вирішення сутностей (entity resolution) та відстеження походження даних (data provenance) [19].

Забезпечуючи точність, повноту і актуальності інформації, інженерія даних стає технологічним фундаментом для машинного навчання – відповідає за безперебійне постачання валідованих вибірок для екстракції ознак, навчання прогнозних моделей та побудови систем інтелектуальної аналітики в реальному часі.

1.1.2. Роль у сучасних IT-системах

Установи покладаються на величезні обсяги інформації для прийняття обґрунтованих рішень, оптимізації операцій та стимулювання інновацій. Ця діяльність відбувається в екосистемі даних – складній взаємопов'язаній структурі платформ, фреймворків та сервісів, котрі спільно забезпечують повний життєвий цикл інформації. У фундаментальних працях з архітектури систем даних наголошується, що ефективність такої екосистеми залежить від балансу між швидкістю обробки та надійністю збереження інформації [13].

Сучасний стек даних складається з кількох важливих рівнів. Корпоративні застосунки, IoT-пристрої та соціальні мережі генерують інформацію, яку динамічні конвеєри даних (Data Pipelines) транспортують і трансформують. Залежно від структури та призначення, інформація розподіляється між класичними базами даних, масштабованими озерами даних (Data Lakes) або централізованими сховищами (Data Warehouses). Білл Інмон, якого вважають "батьком" концепції сховища даних, визначив Data Warehouse як "предметно-орієнтовану, інтегровану, незмінну та хронологічно структуровану колекцію даних для підтримки управлінських рішень" [14].

Інженери даних проєктують і будують конвеєри, що забезпечують вилучення, перетворення і завантаження даних (ETL/ELT) у системи зберігання. Вибір між підходами ETL та ELT впливає на масштабованість системи при підготовці даних для моделей машинного навчання і безпосередньо впливає на здатність організації приймати управлінські рішення у реальному часі [16, 27].

Безпека даних на кожному етапі конвеєра є обов’язковою умовою. Норматив GDPR та його аналоги (CCPA, Закон України “Про захист персональних даних”) встановлюють чіткі вимоги до обробки персональної інформації, і спеціалісти з інженерії даних зобов’язані їх дотримуватися [5].

1.1.3. Відмінності між штучним інтелектом, машинним навчанням і глибоким навчанням

Новітня аналітика корпоративних даних зумовила необхідність застосування передових методів для вилучення прихованої бізнес-інформації. Щоб точно визначити місце прогнозних механізмів у ширшому контексті обчислювальних технологій, необхідно розмежувати три методології (рис. 1.1.): штучний інтелект (AI), машинне навчання (ML) та глибоке навчання (DL). Ці підходи існують у вигляді вкладеної ієрархічної структури, де глибоке навчання є спеціалізованою підгрупою машинного навчання, яке, у свою чергу, виступає спеціалізованим підпростором більш широких систем штучного інтелекту [1].

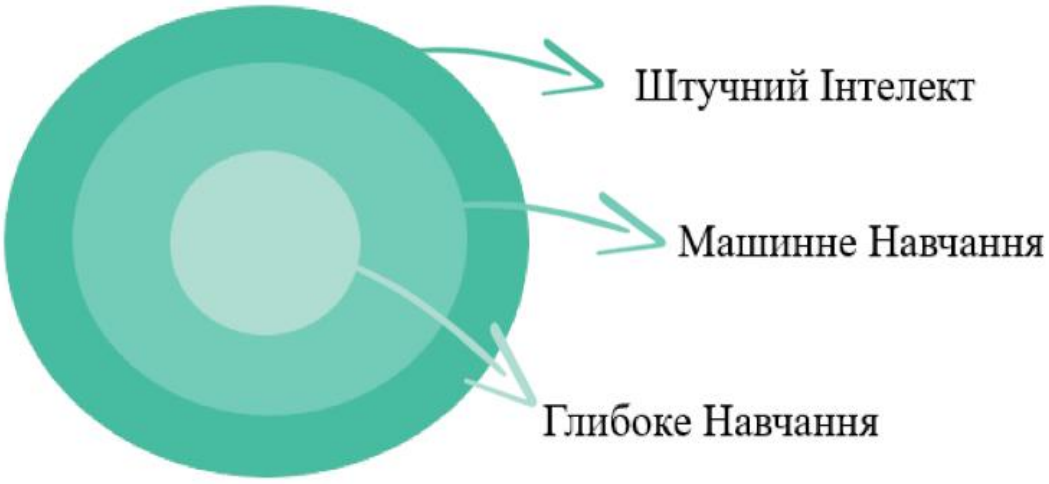

Рисунок 1.1. Різниця між штучним інтелектом, машинним навчанням та глибоким навчанням

Штучний інтелект охоплює обчислювальні архітектури, спрямовані на відтворення когнітивних функцій людини: логіки, розуміння, раціоналізації та здатності до діагностики. Формальне перетворення людського інтелекту на детерміновані або пошукові машинні процеси дозволяє обчислювальним системам самостійно оцінювати складні операційні параметри та приймати рішення, налаштовані на максимізацію ймовірності досягнення визначеної цільової функції. Стюарт Рассел та Пітер Норвіг у класичному підручнику з ШІ описують чотири підходи до визначення: системи, що думають як люди; системи, що діють як люди; системи, що думають раціонально; системи, що діють раціонально [2, 20]. Саме остання концепція є домінуючою у сучасних інженерних застосуваннях.

Машинне навчання, як окрема підкатегорія в рамках структури штучного інтелекту, модифікує класичний підхід до обчислень: замість прописаних інструкцій впроваджуються методи адаптивного аналізу даних. Фреймворки машинного навчання наділяють системи алгоритмічною здатністю індуктивно виводити операційну логіку та поступово вдосконалювати показники ефективності безпосередньо на основі експериментальних даних. Том Мітчелл дав наступне визначення: “комп’ютерна програма навчається з досвіду E стосовно задачі T і міри якості P, якщо її якість P при виконанні T покращується з досвідом E” [21]. Цей підхід замінює необхідність ручного програмування на здатність автоматично виявляти закономірності в даних.

Глибоке навчання становить основу ієрархії машинного навчання, використовуючи багаторівневі нейронні мережі для обробки даних. На відміну від традиційних методів машинного навчання, де ознаки часто розробляються вручну, глибоке навчання спеціально оптимізоване для автономної обробки неструктурованих, багатовимірних та немаркованих наборів даних, у якому кожен шар мережі навчається ієрархічним абстракціям [22].

1.2. Система даних для машинного навчання

1.2.1. Ключові етапи Data Pipeline

Ефективність аналітики та моделей машинного навчання залежить від надійних конвеєрів даних (Data Pipelines) – механізмів, призначених для переміщення та перетворення необроблених даних з різних джерел у цінні результати. Кінцеві продукти (дашборди, прогнозні висновки) є лише видимою вершиною аналітичних процесів, які підтримуються розгалуженою, часто невидимою інфраструктурою складних етапів обробки [16].

Необроблені дані зазвичай хаотичні й відрізняються за якістю, структурою та форматом. Після вилучення вони проходять інтенсивну фазу попередньої обробки: стандартизуються несумісні формати, шифруються рядки та числові схеми. Далі напівструктуровані або неструктуровані вихідні дані перетворюють у єдині реляційні моделі та об’єднують окремі потоки для отримання цілісного подання. Основоположною аксіомою інженерії даних є принцип “Сміття на вході, сміття на виході” (Garbage In, Garbage Out), що характеризує якість результату, який завжди залежить від якості вхідної інформації. Томас С. Редман, автор праці “Data Quality: The Field Guide”, підкреслює, як компанії втрачають у середньому 15-25% доходу через погану якість даних, а надійні конвеєри мають розглядати валідацію як постійну програмну необхідність [50].

Основні операції (вилучення, перетворення, перевірка та завантаження) залишаються незмінними – зазнає змін архітектурна послідовність їх виконання. Історично робочі процеси в галузі інженерії даних суворо дотримувались парадигми ETL (Extract, Transform, Load), та розвиток сучасної хмарної інфраструктури став каталізатором переходу до підходу ELT (Extract, Load, Transform). Відмінність між цими двома методологіями полягає насамперед у тому, де відбувається обчислювально-інтенсивна обробка та як це впливає на масштабованість конвеєра.

Конвеєри ETL суттєво економлять час і ресурси компаній, вирішуючи широкий спектр практичних завдань: від спрощення міграції даних із застарілих баз до сучасних хмарних сховищ та консолідації всієї корпоративної інформації в єдиному центрі до глибокої інтеграції CRM-платформ із системами автоматизації маркетингу (MAP). За допомогою попереднього форматування й обробки, конвеєри формують надійні набори даних для миттєвого вирішення конкретних аналітичних завдань, а також дозволяють заздалегідь виключати або маскувати конфіденційні дані перед їх завантаженням у фінальну систему – це дає змогу дотримуватися нормативних вимог безпеки. Таке ефективне використання ETL-процесів усуває проблему ізольованості даних, створює єдине консолідоване уявлення про всі процеси в організації та стає фундаментом для прийняття якісних управлінських рішень за допомогою інструментів аналітики, візуалізацій та інтерактивних інформаційних панелей.

Порівнюючи конвеєр даних з ETL-конвеєром, стає зрозуміло, що конвеєр даних – це більш комплексний набір процедур, який забезпечує передачу даних. А конвеєр ETL підпадає під цю категорію як особливий різновид конвеєра даних. Стає можливим виділити три фундаментальні відмінності між конвеєром даних та ETL:

1. Конвеєри даних можуть або змінювати дані після завантаження, або не змінювати їх взагалі, тоді як ETL-конвеєри змінюють дані перед завантаженням у систему призначення;
2. Конвеєри даних не завжди завершуються після завантаження даних. З огляду на те, що багато сучасних конвеєрів даних передають дані в потоці, їхня процедура завантаження може сприяти створенню звітів у реальному часі або запуску діяльності в інших системах. На відміну від них, процеси ETL завершуються, коли дані завантажуються в сховище призначення;
3. Не всі конвеєри даних працюють пакетно. Сучасні конвеєри даних часто використовують потокові обчислення для обробки в реальному часі. Це дозволяє постійно оновлювати дані, уможливлюючи аналітику та звітність у

реальному часі, а також активацію додаткових систем. Конвеєри ETL передають дані до системи призначення партіями за заздалегідь визначеним розкладом [11, 28, 29].

1.2.2. Інструменти систем обробки даних

Конвеєр даних – це рух даних від джерела до місця призначення. Конвеєр даних ETL використовує програмний код для виконання трьох основних операцій.

1. Вилучення даних (Data Ingestion/Extraction)

Перший етап, на якому дані збираються з різних джерел. Наприклад баз даних (SQL, NoSQL), API, файлів (CSV, JSON, Parquet), лог-файлів, CRM-систем чи SFTP-серверів. Серед типів є пакетна обробка Batch (дані збираються та обробляються великими порціями за розкладом) і потокова обробка Streaming (дані надходять і обробляються в реальному часі; наприклад, через Apache Kafka).

2. Трансформація даних (Data Transformation)

Перетворення “сирих” даних у формат, придатний для аналізу. Найважливіший етап, що включає: очищення – видалення дублікатів, обробку пропущених значень (NULL), виправлення помилок. Збагачення – додавання інформації з інших джерел. Фільтрацію та агрегацію – відбір потрібних даних, групування та узагальнення. Форматування – зміна типів даних, структурування для сховища.

3. Завантаження даних (Data Loading)

Запис оброблених та очищених даних у цільове місце призначення. Data Warehouse: Сховище даних (BigQuery, Snowflake, Redshift) – для аналітики. Data Lake: Озеро даних (S3, Azure Blob Storage) – для зберігання неструктурованих даних [28, 29].

1.2.3. Вимоги до систем обробки даних, переваги та недоліки ETL, ELT

Проєктування надійного та ефективного конвеєра даних вимагає дотримання низки технічних та експлуатаційних вимог заради автоматизованого, точного та своєчасного переміщення інформації. Ключовим фактором життєздатності такої системи є її масштабованість, що дозволяє опрацьовувати зростаючі обсяги даних за допомогою динамічного розширення хмарних обчислювальних потужностей або інструментів на зразок Apache Spark. Наскрізна автоматизація процесів збору, трансформації та завантаження зводить до мінімуму людський фактор, тоді як вбудовані суворі перевірки якості й цілісності даних (Data Quality) гарантують їхню абсолютну точність, повноту та відповідність бізнес-правилам на кожному етапі. Для злагодженої роботи архітектура конвеєра повинна мати високий рівень надійності та відмовостійкості, гарантуючи доставку даних без жодних втрат, шляхом механічного автоматичного перезапуску та відновлення роботи після збоїв. Безперервний контроль працездатності інфраструктури здійснюється за допомогою впровадження інструментів моніторингу в реальному часі та систем миттєвого оповіщення про помилки через корпоративні канали зв’язку (Slack, Email). Сучасний пайплайн має демонструвати високу продуктивність для суворого дотримання регламентованих угод про рівень послуг (SLA) щодо швидкості доставки даних, а також володіти гнучкою модульною структурою, яка дозволяє легко інтегрувати нові джерела або модифікувати логіку обробки без ризику руйнації всієї системи. Увесь цикл обов’язково підпорядковується жорстким стандартам безпеки та сумісності (наприклад, GDPR або PCI DSS), що передбачає надійний захист і шифрування чутливих даних під час їх транспортування, а також чітке розуміння та аудит місць зберігання конфіденційної інформації [30, 31].

Впровадження класичного процесу ETL у сховищах даних має чітко виражений двосторонній характер, поєднуючи в собі стратегічні переваги і технологічні й фінансові обмеження. Завдяки стандартизації форматів, виправленню помилок та автоматизації перевірок, ETL забезпечує високу якість даних, гарантуючи їхню точність, повноту та актуальність. Також значно покращує інтеграцію, об’єднуючи розрізнені потоки з багатьох джерел у єдиний, доступний і зручний для використання масив [9, 32].

Без належної уваги система підготовки даних у форматі, придатному для машинного навчання, може перетворитися на заплутану мережу операцій з вилучення даних, об’єднання та вибірки, що часто супроводжується створенням проміжних файлів. Такі дані перетворюються в “Джунглі конвеєрів”. Вони можуть органічно розвиватися у міру виявлення нових сигналів та поступового додавання нових джерел даних і надзвичайно важко піддаються управлінню, виявленню помилок і відновленню [12]. Разом з тим, новітні ETL-інструменти дозволяють підвищити рівень безпеки через суворий контроль авторизованого доступу, оптимізувати масштабованість для управління великими обсягами інформації та автоматизувати рутинні процеси оновлення сховища, мінімізуючи часові й трудові витрати команди.

Однак, цей підхід також супроводжується недоліками, серед яких висока вартість впровадження та подальшої підтримки інфраструктури, що може становити складність для організацій з обмеженими ресурсами. Загальна архітектурна складність реалізації вимагає від компанії дефіцитної технічної експертизи, а сама специфіка традиційного ETL накладає обмеження на гнучкість системи, яка часто виявляється нездатною ефективно обробляти неструктуровані дані чи динамічні потоки в реальному часі. У певних сценаріях масштабованість ETL-інструментів може досягати своєї межі при зіткненні з надвеликими масивами даних (Big Data), а централізований збір, обробка та зберігання колосальних обсягів інформації неминуче породжують додаткові ризики й

занепокоєння щодо забезпечення конфіденційності та захисту персональних даних [9, 32].

Архітектурний підхід ELT демонструє баланс між технологічною гнучкістю та експлуатаційними викликами, змінюючи принципи роботи з інформацією. До безумовних переваг системи ELT належить висока швидкість роботи, оскільки дані завантажуються без попередньої підготовки, а їх подальша трансформація здійснюється потужними хмарними сервісами, що робить інтерактивну аналітику самообслуговування доступною майже в реальному часі. Цей підхід забезпечує виняткову гнучкість, адже сирі дані перетворюються під конкретні запити користувачів безпосередньо в момент звернення, що дозволяє використовувати їх багаторазово для різних цілей без складних налаштувань на рівні самого конвеєра. Крім того, ELT гарантує високу прозорість доступної інформації та безмежну масштабованість у хмарі, поєднуючи це з низькими експлуатаційними витратами завдяки відсутності потреби в локальному обладнанні та гнучкій моделі оплати виключно за фактично використані хмарні ресурси [9, 33].

Серед недоліків – високі вимоги до інфраструктури, адже перенесення етапу трансформації всередину сховищ (як-от Snowflake чи BigQuery) створює колосальне навантаження на обчислювальні потужності та, як наслідок, провокує вищі хмарні витрати за умови відсутності жорсткої оптимізації процесів. Гострою проблемою також є загроза безпеці та конфіденційності, оскільки невідфільтровані сирі дані, включно з персонально ідентифікованою інформацією (PII), завантажуються в систему ще до очищення чи анонімізації, що ускладнює комплаєнс. Також без суворого управління метаданими та чіткої документації існує високий ризик перетворення сховища на хаотичне “болото даних” (Data Swamp). Сам процес вимагає від інженерів глибоких знань SQL для написання складних запитів усередині сховища замість використання звичних ETL-інструментів, а необхідність обробки та перетворення масивів безпосередньо перед використанням може спричинити затримки в отриманні фінальної бізнес-аналітики [34].

Методології ETL та ELT, мають свої сильні та слабкі сторони (рис. 1.2.), і вибір між ними залежить від різних факторів, таких як обсяг даних, складність, вимоги до затримки та пріоритети організації. Хоча ETL пропонує надійні можливості очищення та трансформації даних, ELT забезпечує масштабованість, гнучкість і переваги обробки в реальному часі. Зрештою, організаціям потрібно оцінити свої конкретні вимоги та обмеження, щоб визначити найбільш відповідний підхід для інтеграції даних і потреб у складі.

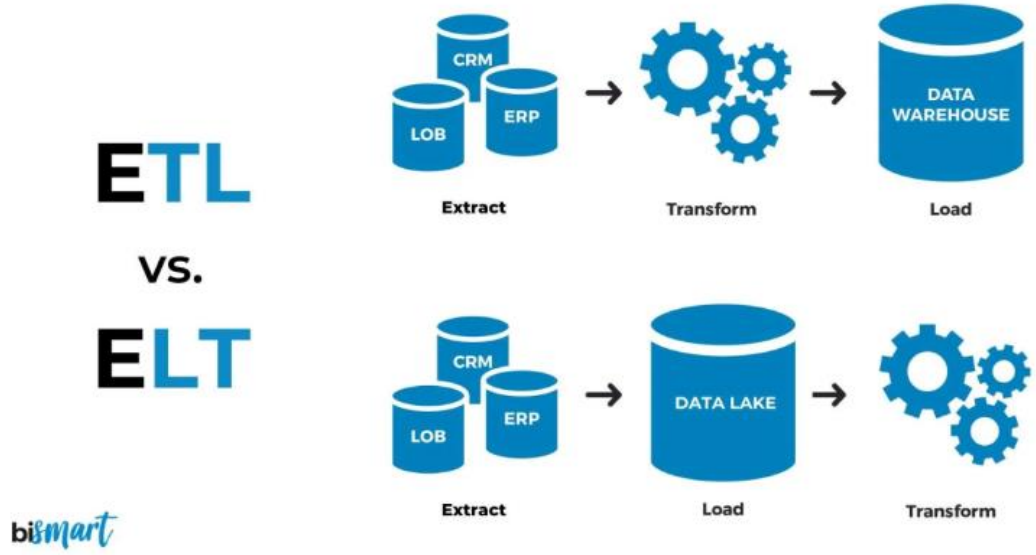

Рисунок 1.2. ETL проти ELT: розуміння відмінностей

1.3. Особливості даних відповідно до типу навчання

1.3.1. Особливості даних для навчання з учителем, без вчителя

Навчання з учителем. У цьому разі машина має “наставника” – учителя, який говорить їй, як правильно вчинити див (рис 1.3.).

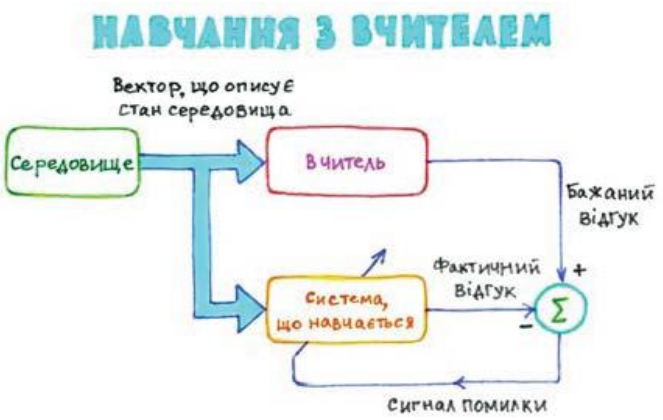

Рисунок 1.3. Основи машинного навчання – навчання з вчителем

Вчитель заздалегідь відмічає всі потрібні дані, щоб машина навчалася на конкретних прикладах. Так він показує, що на цій світлині – автомобіль, на іншій – велосипед. Учителем не завжди буває програміст, який стоїть над машиною й контролює кожну її дію. У термінах машинного навчання вчитель – це саме втручання людини в процес оброблення інформації [35]. У навчанні з учителем модель тренується на даних, які включають як вхідні дані, так і правильні відповіді (тобто цільові дані) [36].

Класифікація – найпопулярніше завдання в усьому машинному навчанні – розділяє об’єкти за заздалегідь відомими ознаками. При потребі, дата інженер може надати даним прості ознаки, розділити і впорядкувати за допомогою таблиць [35]. Існує безліч різних способів перетворення табличних даних, і кожен із них може по-різному впливати на результати навчання [8]. Для класифікації завжди потрібен вчитель – розмічені дані за ознаками та категоріями, за якими машина вчитиметься визначати. Це використовують для: спам-фільтрів; визначення мови; аналізування тональності; пошуку подібних документів; розпізнавання прописних літер та цифр; визначення підозрілих транзакцій [35].

Навчання без учителя. У навчанні без учителя машині просто викладають купу фотографій на образний стіл і кажуть: “Розберися, що тут до чого”. Дані, в такому разі, не розмічені, і машина сама намагається знайти закономірності (рис. 1.4.).

Рисунок 1.4. Основи машинного навчання – навчання без вчителя

На практиці використовують і алгоритми, як методи аналізування та підготовки даних, а не як основний алгоритм, що виконує конкретні завдання за допомогою цих даних. У разі, якщо розмічення даних неможливе, вдаються до методів навчання без учителя [35].

Навчання без учителя відбувається без правильних відповідей (немає цільових даних), на відміну від попереднього методу. Модель намагається знайти структури, групи або закономірності, спираючись лише на вхідні дані. Прикладом цього є рекомендаційні системи в інтернет-магазинах: можна вивчити, як знаходити схожі продукти (або користувачів), аналізуючи поведінку користувачів, наприклад, які товари вони часто купують разом [38].

Кластеризація у навчанні без вчителя (Unsupervised Learning) – це метод машинного навчання, який автоматично групує набір даних (об’єктів) на групи (кластери) за схожими характеристиками, не маючи попередньої інформації про те, які це класи [35].

1.3.2. Особливості даних для навчання з підкріпленням

Інженерія даних (Data Engineering) для навчання з підкріпленням (Reinforcement Learning, RL) – це специфічна сфера, яка фокусується на створенні надійних пайплайнів для збору, обробки та зберігання даних, необхідних для навчання агентів. Основна різниця між класичним Data Engineering та інженерією для RL полягає в тому, що в RL дані часто генеруються агентом у реальному часі, взаємодією із середовищем [37]. Наявні наступні рекомендації Data Engineering для RL:

- Динамічне середовище: інженер даних має забезпечити високу швидкість обробки даних (streaming), оскільки агент навчається на основі отриманих винагород і станів;
- Генерація досвіду (Experience Replay): створення ефективних сховищ (буферів), де зберігаються послідовності “стан-дія-винагорода-наступний стан” для повторного навчання;
- Підготовка симуляцій: часто дата-інженери налаштовують середовища симуляції, звідки RL-агент отримує дані для тренування [35].

1.3.3. Особливості даних для ансамблів, нейромережі та глибоке навчання

Класичні ансамблеві методи машинного навчання демонструють найкращу ефективність при роботі з табличними, чітко структурованими даними (CSV-файли чи SQL-таблиці). Серед численних форматів даних табличні залишаються переважаючими в дослідженнях. Вадим Борисов та ін. у систематичному огляді глибоких нейронних мереж для табличних даних встановили, що деревоподібні моделі (наприклад, XGBoost, LightGBM) стабільно переважають нейронні мережі на типових структурованих датасетах, що пояснюється природою розподілів

числових ознак і важливістю категоріальних змінних [6]. Грінштайн, Ояльон та Вароку підтвердили ці результати в нейро-інформаційній конференції NeurIPS 2022: деревоподібні ансамблі залишаються стандартом для середніх табличних датасетів, тоді як нейронні мережі починають конкурувати лише на великих датасетах з виразними неявними залежностями [7]. Подібний формат схожий до процесу збору та зберігання інформації різними організаціями, наприклад, формат збору даних пацієнтів, даних сенсорів та характеристик поведінки споживачів [6].

Напрям глибокого навчання (Deep Learning) розроблений для ефективної роботи з колосальними обсягами неструктурованих даних, володіючи при цьому унікальними інженерними та функціональними особливостями. Визначальним фактором є масштаб і характер інформації. Для досягнення високої точності глибокі моделі потребують величезних повсякденних масивів, як-от зображення, відео, аудіозаписи, тексти чи складні часові ряди, де діє пряма закономірність: що більше даних надходить у систему, тим якісніше відбувається процес навчання нейромережі. З технічного погляду, такі процеси вимагають обов’язкового масштабування інформації шляхом її нормалізації або стандартизації (наприклад, приведення числових значень до діапазону [0, 1] або до матриці із середнім значенням 0 і дисперсією 1), а також ретельного попереднього оброблення, яке полягає у конвертації різнорідних сирих форматів у чіткі числові структури – тензори. Гудфеллоу, Бенджіо та Курвіль підкреслюють, що головна перевага глибокого навчання над класичними алгоритмами полягає у здатності автоматично виявляти ієрархічні ознаки безпосередньо з “сирих” масивів, звільняючи інженерів від складного ручного проектування ознак (Feature Engineering) [22].

РОЗДІЛ 2. ВЗАЄМОДІЯ З ВЕЛИКИМИ ДАНИМИ BIG DATA

2.1. Концепція та характеристики великих даних

2.1.1. Концепція великих даних

Великі дані (Big Data) – це галузь, що займається аналізом, обробкою та зберіганням великих колекцій даних, які часто походять з різних джерел і не можуть бути ефективно оброблені традиційними реляційними системами. Big Data відповідає на різні вимоги, такі як об’єднання кількох непов’язаних наборів даних, обробка великих обсягів неструктурованих даних та збір прихованої інформації з обмеженим часом [10].

Термін “Великі дані” (Big Data) з’явився 3 вересня 2008 р., коли вийшов спеціальний випуск британського журналу “Nature”, присвячений питанням наукових можливостей роботи з великими даними. Згідно зі звітом інституту McKinsey “Big Data: The Next Frontier for Innovation, Competition, and Productivity” (2011), термін відноситься до наборів даних, розмір яких перевищує ємність звичайних баз даних для видобування, зберігання, управління і аналізу інформації. Аналітики “Gartner” запропонували широко цитоване визначення “3V”: Volume (обсяг), Velocity (швидкість) і Variety (різноманітність). Пізніше модель була доповнена до “5V” елементами Veracity (достовірність) і Value (цінність).

Великі дані (Big Data) в інформаційних технологіях (за визначенням К. Лінча) – набір методів та засобів опрацювання структурованих і неструктурованих різнотипних динамічних даних великих обсягів з метою їх аналізу та використання для підтримки прийняття рішень. Є альтернативою традиційним системам управління базами даних і рішенням класу Business Intelligence. До цього класу відносять засоби паралельного опрацювання даних (NoSQL, алгоритми MapReduce, Hadoop). На думку компанії “DCA” (Data-Centric

Alliance) під Big Data розуміють не конкретний об’єм даних, а методи їх обробки, які дозволяють розподілено обробляти інформацію. Ці методи можна застосовувати як до великих масивів даних (таких як дані всіх сторінок в мережі Інтернет), так і до малих масивів (інформація про денні поступлення товару в магазин).

Сукупність технологій великих даних дозволяють виявляти приховані від обмеженого людського сприйняття закономірності. Це дає безпрецедентні можливості оптимізації багатьох сфер життя: державного управління, медицини, телекомунікацій, фінансів, транспорту, виробництва і т. д. Характерно, що у жовтні 2015 року компанія “Gartner” виключила Big Data з числа трендів. Своє рішення аналітики пояснили тим, що пов’язані технології вже активно застосовуються на підприємствах, частково стосуються інших популярних тенденцій і стали повсякденним робочим інструментом [3].

Серед прикладів генераторів великих даних наявний Airbus A380 Engine, що генерує 1 петабайт даних під час рейсу з Лондона в Сінгапур; великий адронний колайдер – 1 гігабайт даних щосекунди; квадратний кілометровий масив – 20 екзабайтів даних на день (еквівалентно 20 мільярдам гігабайт на день) [4].

2.1.2 Характеристики великих даних

Повна модель характеристик великих даних у сучасній літературі налічує від 5 до 10 “V”. Найбільш вживаними є сім:

1. Volume (обсяг) – загальна кількість даних, що зберігаються і обробляються. Навіть одна організація може мати як десятки терабайт, так і сотні петабайт даних. Наприклад, обробки страхових виплат, продажу нових полісів та внесення змін до існуючих полісів. Однак коротке обговорення показує, що великі обсяги неструктурованих даних, як всередині, так і за межами компанії, можуть виявитися корисними для досягнення цілей ETI. Ці дані включають

медичні карти, документи, подані клієнтами під час подання заяви на страхування, переліки майна, дані про автопарк, дані соціальних мереж та дані про погоду. Аналіз такої кількості інформації неможливий без спеціальних технологій.

2. Velocity (швидкість) – темп генерації нових даних та необхідної їх обробки. Завдання Big Data полягає в тому, щоб впоратися з швидкістю надходження даних та проаналізувати їх у реальному часі. Інакше нові дані встигли б створитися, доки триває аналіз попередніх, роблячи його неактуальним. Висока швидкість спостерігається у даних з веб-сервісів та страхові пропозиції. Дані управління катастрофами та страхового відшкодування повинні оброблятися швидко, аби запобігти збиткам. Технології потокової обробки (Apache Kafka, Apache Flink) орієнтовані саме на цю характеристику

3. Variety (різноманітність) – структуровані таблиці, неструктурований текст, аудіо, відео та дані геолокації. Кожен формат потребує індивідуального аналізу. Наприклад, користувач може зв’язатися з компанією через соціальні мережі за допомогою комп’ютера, переглядати вебсайт компанії на смартфоні, робити покупки за допомогою планшета та написати електронний лист у службу підтримки. Усі дані генеруються від однієї особи, але мають різні форми.

4. Veracity (достовірність) – ступінь точності та надійності даних. Якщо компанія працює з неперевіреною інформацією, вона не зможе приймати управлінські рішення, а її ініціативи зазнають невдач. Наприклад, контакти клієнтів були збережені з помилками, і тепер бізнес не може надати їм інформацію про нові послуги та акції.

5. Variability (мінливість) – зміна контексту та значення даних з часом. Регулярне фіксування змін є важливим аспектом роботи з Big Data, завдяки цьому можна виявляти закономірності та будувати маркетингову стратегію для конкретної категорії споживачів.

6. Visualization (візуалізація) – необхідність представлення результатів у доступній формі. Задля зручного опрацювання інформації необхідна якісна візуалізація аналізу великих даних за допомогою діаграм, кольорових графіків та інтерактивних карт.

7. Value (цінність) – здатність отримувати корисні інсайти з величезних масивів. Великі дані дозволяють приймати ефективні управлінські рішення. Наприклад, після проведеного аналізу, компанія може розширити товарну лінійку і додати пропозиції, актуальні саме для її клієнтів [4, 9].

2.2. Системи зберігання великих даних

Найбільш очевидними прикладами децентралізованих сховищ є хмарні сховища, такі як iCloud, Google Drive та Dropbox. Використовуючи ці хмарні сховища, клієнти можуть легко завантажувати будь-яку інформацію, яка одразу ж зберігатиметься на кількох безпечних і надійних серверах. Для зручності користувачів, сервіси пропонують обмін посиланнями з іншими, щоб інформація була легко доступною для завантаження. Прикладом надзвичайно популярного хмарного сховища є Amazon S3. Ця децентралізована система зберігання даних здебільшого орієнтована на об’єктне зберігання. Всі об’єкти в системі ідентифікуються за допомогою ключа і зберігаються по всьому світу.

HDFS або Hadoop File System використовується переважно для зберігання великих обсягів даних, пов’язаних з аналітикою. Ця система функціонує на стандартному обладнанні, тому ціни на неї досить прийнятні. Azure Blob Storage – хмарне сховище для зберігання величезних обсягів неструктурованої інформації: починаючи від файлів і закінчуючи зображеннями та відео. Фреймворк Ceph – масштабований варіант для зберігання файлів, об’єктів або навіть блокчейну. Google Cloud Storage – універсальний варіант для широкого кола користувачів,

яким потрібно зберігати величезні обсяги інформації для аналітики, резервного копіювання, веб-хостингу та аварійного відновлення [39].

Об’єктне сховище – це технологія зберігання та управління даними у неструктурованому форматі, який називають об’єктами. У сучасних організаціях створюються та аналізуються великі обсяги неструктурованих даних, таких як фотографії, відео, електронні листи, вебсторінки, сенсорні дані та аудіофайли. Хмарні об’єктні системи зберігання розподіляють ці дані по безлічі фізичних пристроїв, при цьому надаючи користувачам можливість доступу до контенту з єдиного репозиторію віртуального сховища. Рішення об’єктних сховищ ідеально підходять для розробки хмарних додатків, для яких потрібна гнучкість і можливість масштабування. Крім того, ці сховища можна використовувати для імпорту даних з існуючих сховищ з метою аналітики, резервного копіювання або архівування [40].

База даних NoSQL – це тип нереляційної бази даних, призначеної для зберігання та управління даними, які не вміщуються в таблиці з фіксованими схемами та заздалегідь визначеними структурами. Вони надають пріоритет гнучкості, масштабованості та продуктивності, що робить їх ідеальними для великих обсягів розподілених, напівструктурованих або даних, що швидко змінюються. NoSQL відрізняються від баз даних SQL способом зберігання даних, масштабування систем та обробки мінливих вимог додатків. Вебсайти, мобільні додатки та хмарні сервіси почали обробляти великі обсяги інформації, саме тоді багатьом командам знадобилися бази даних, які могли б легко розширюватися та адаптуватися у міру зміни вимог [41].

NAS (Network-attached Storage) – це централізована система зберігання даних, що містить один або кілька накопичувачів, і підключена до локальної мережі через Ethernet або, частіше, Wi-Fi. Як частина домашньої або офісної мережі, NAS забезпечує доступ до даних із різних пристроїв, таких, як ноутбуки, телефони, планшети, Smart-телевізори та навіть ігрові консолі. Такий своєрідний інтелектуальний пристрій для централізованого зберігання даних, який надає

змогу керувати й поширювати персональні файли, зокрема фотографії, відео, музику й документи. Підключивши NAS до домашньої чи офісної мережі, можна створити безпечний простір для зберігання даних, надавати доступ до якого членам родини або колегами можна через ПК або мобільний пристрій [42].

Мережа зберігання даних (МЗД) (англ. Storage Area Network (SAN)) – архітектурне рішення для підключення зовнішніх пристроїв зберігання даних, таких як дискові масиви, стрічкові бібліотеки, оптичні накопичувачі до серверів таким чином, щоб операційна система розпізнала підключені ресурси як локальні. Побудова мережі SAN вирішує проблеми зниження сукупної вартості володіння системою зберігання даних, а також надає інструменти для організації надійного зберігання інформації. В простому випадку SAN складається з СХД, комутаторів і серверів, об’єднаних оптичними каналами зв’язку. Окрім дискових СХД в SAN можна підключити дискові бібліотеки, стрічкові бібліотеки, пристрої для зберігання даних на оптичних дисках (CD/DVD і другі) і т. д. Така схема високонадійної інфраструктури, в якій сервери ввімкнені одночасно в локальну мережу і в мережу зберігання даних, забезпечує доступ до даних, що знаходяться на СХД при виході з ладу будь-якого процесорного модуля, комутатора або шляху доступу.

МЗД (SAN) характеризується наданням так званих мережевих блокових засобів (як правило за допомогою протоколів AoE), в той час як мережеві сховища даних (NAS (англ. Network Attached Storage)) націлені на надання доступу до даних, які зберігаються на їх файловій системі за допомогою мережевої файлової системи (такої як NFS, SMB/CIFS iSCSI, або AppleTalk).

Категоричний поділ зразку “SAN – це тільки мережеві диски, NAS – це тільки мережева ФС” є штучним: з появою iSCSI почалося взаємне проникнення технологій з метою підвищення гнучкості і зручності їх застосування. Наприклад, в 2003 році NetApp вже надавали iSCSI на своїх NAS, а EMC і HDS – навпаки, пропонували NAS-фронтенди для своїх SAN-масивів [28]. Використання SAN дозволяє забезпечити:

- Централізоване управління ресурсами серверів і систем зберігання даних;
- Підключення нових дискових масивів і серверів без зупинки роботи всієї системи зберігання;
- Використання раніше придбаного обладнання спільно з новими пристроями зберігання даних;
- Оперативний і надійний доступ до накопичувачів даних, що знаходяться на великій відстані від серверів без значних втрат продуктивності;
- Прискорення процесу резервного копіювання і відновлення даних – BURA [43].

2.3. Структури обробки великих даних та інструменти

2.3.1. Структури обробки великих даних

З точки зору обробки, в основу технологій Big Data покладені два фундаментальних принципи: розподіленого зберігання даних і розподіленої обробки з урахуванням локальності даних.

Розподілене зберігання вирішує проблему великого обсягу даних, дозволяючи організовувати сховище з довільного числа окремих простих носіїв. Зберігання може бути організовано з різним ступенем надмірності, забезпечуючи стійкість до збоїв окремих носіїв. Розподілена обробка, з урахуванням локальності даних, означає, що програма обробки доставляється на обчислювач, що знаходиться якомога ближче до оброблюваних даних. Це принципово відрізняється від традиційного підходу, коли обчислювальні потужності і підсистема зберігання розділені і дані повинні бути доставлені на обчислювач. Завдяки цьому, технології Big Data спираються на обчислювальні кластери з безлічі обчислювачів, забезпечених локальної підсистемою зберігання. Доступ до даних і їх обробка здійснюються спеціальним програмним забезпеченням.

Найбільш відомим проєктом у галузі Big Data є Apache Hadoop, який представляє собою програмний фреймворк, що дозволяє зберігати і обробляти дані за допомогою комп'ютерних кластерів, використовуючи парадигму MapReduce. Широке використання MapReduce в машинному навчанні було зумовлене відсутністю потужних абстракцій для розподіленого навчання. Разом з тим, цей підхід не є найкращою абстракцією для ітеративних алгоритмів машинного навчання. [3, 13].

Основними складовими платформи Hadoop є: відмовостійка розподілена файлова система Hadoop Distributed File System (HDFS); програмний інтерфейс MapReduce для паралельної обробки; Apache Hadoop YARN як менеджер ресурсів кластера. Відповідно до підходу MapReduce обробка складається з двох кроків: Map (паралельна попередня обробка на вузлах кластера) і Reduce (зведення результатів у єдиний підсумок) [3].

2.3.2. Інструменти обробки великих даних

Apache Spark – це уніфікована аналітична система з відкритим кодом для великомасштабної обробки даних. Початково розроблений як альтернатива Hadoop MapReduce, Spark пропонує вищу швидкість обробки, особливо для обчислень у пам'яті. Здатність утримувати й обробляти масиви безпосередньо в оперативній пам'яті мінімізує звернення до дискових сховищ, що критично важливо для ітеративних робочих навантажень. Тоді як вбудована відмовостійкість гарантує стабільність бізнес-додатків через автоматичне виявлення збоїв вузлів кластера та миттєве відновлення процесів без втрати даних. Захарія та ін. у статті "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing" продемонстрували прискорення до 100× порівняно з Hadoop MapReduce для ітеративних алгоритмів машинного навчання [23].

Унікальність Spark полягає в її багатофункціональній єдиній екосистемі. Вона гармонійно поєднує інструменти для класичної пакетної обробки історичних даних та генерації звітів, механізми потокового аналізу (Streaming), а також спеціалізовані бібліотеки для масштабованого машинного навчання, аналізу складних взаємозв'язків у графових структурах (наприклад, для рекомендаційних систем чи виявлення шахрайства) та виконання аналітичних запитів за допомогою знайомого синтаксису SQL. Висока популярність і простота впровадження цієї платформи забезпечуються її крос-мовною доступністю, адже нативні API для Scala, Python, Java та R дозволяють розробникам із різним технічним бекграундом легко створювати рішення, прискорюючи розробку та спрощуючи інтеграцію Spark в існуючу корпоративну інфраструктуру [44].

Apache Hadoop – вільна програмна платформа і каркас для організації розподіленого зберігання і обробки наборів великих даних з використанням моделі програмування MapReduce. При ній завдання ділиться на багато дрібніших відособлених фрагментів, кожен з яких може бути запущений на окремому вузлі кластера, що складається з серійних комп'ютерів. Всі модулі в Hadoop спроєктовані з урахуванням припущення, що апаратне забезпечення часто виходить з ладу і такі ситуації повинні автоматично опрацьовуватись фреймворком. Коли вузли маніпулюють лише даними, до яких мають доступ, це дозволяє обробляти набір даних швидше і ефективніше, ніж в традиційній суперкомп'ютерній архітектурі [45, 46].

Apache Kafka – це розподілена система обміну повідомленнями з відкритим кодом для обробки великих обсягів потоків даних у реальному часі. Вона поєднує високу пропускну здатність із низькою затримкою доставки даних, написана на Java та Scala. Kafka розробили інженери LinkedIn для обробки даних, генерованих платформою, бажаючи зберігати та фіксувати всі дії користувачів заради надання більш персоналізованого досвіду. Завдяки її масштабованості у лютому 2020 компанія змогла збільшити обсяг обміну повідомленнями до понад семи трильйонів [24]. Високу доступність та продуктивність Kafka забезпечує шляхом розподілення даних між різними серверами, а не зберіганням в одному місці. У

2011 році проєкт став опенсорсним і його передали Apache Software Foundation. Наразі Kafka використовують розробники та дата інженери Netflix, Uber, Airbnb, Twitter та безліч інших компаній у сфері фінансів, ігор тощо [47].

На відміну від традиційних черг повідомлень, Kafka гарантує довготривале збереження даних (Persistency) завдяки їх упорядкованому запису на диск – це дозволяє надійно фіксувати історію подій та гнучко обробляти їх як у реальному часі, так і в ретроспективі. В основі взаємодії компонентів платформи лежить класична модель Publish-Subscribe, де постачальники (producers) публікують дані у спеціальні тематичні канали – топіки (topics), а споживачі (consumers) підписуються на них для читання. Завдяки швидкодійному механізму розділення (Partitioning), кожна тема розбивається на окремі партиції, дозволяючи різним споживачам одночасно й незалежно обробляти ізольовані сегменти даних [48].

РОЗДІЛ 3. ВИКОРИСТАННЯ МАШИННОГО НАВЧАННЯ НА BIG DATA ДЛЯ ВИЗНАЧЕННЯ ФІНАНСОВОГО ШАХРАЙСТВА

3.1. Інструменти та технології розробки

У цьому проєкті були використані такі інструменти і технології як:

- Apache Airflow – планувальник конвеєрів на основі DAG. Виконує вилучення → перевірку → перетворення → оцінку по порядку з паралельним навчанням моделі;
- Docker + Compose – контейнеризує вебсервер, планувальник та Airflow. Забезпечує ідентичну роботу конвеєра на будь-якій машині;
- Apache Spark – spark_fraud_processing.py – розподілена інженерія функцій, зчитує перевірений CSV, інженерію нових стовпців, записує вивід Parquet;
- Parquet – стовпцевий формат файлу, написаний Spark. Ефективний для великих наборів даних. Використаний для завантаження у ETL (Витяг, Завантаження, Трансформація);
- Scikit-learn – надає можливість використання Pipeline, StandardScaler, OneHotEncoder, LogisticRegression та RandomForestClassifier, які необхідні для машинного навчання;
- XGBoost – дерева, підсилені за допомогою Gradient Boosting. Виправляє дисбаланс класів через scale_pos_weight. Часто є лідером за результатами аналізу табличних даних щодо шахрайства;
- Joblib – послідовно зберігає налаштовані конвеєри моделей у файли .pkl. Також дозволяє паралельну обробку всередині RandomForestClassifier;
- Pandas – використовувалося на межі між Spark та scikit-learn, перетворюючи DataFrames Spark у Pandas для розділення, тренування та введення моделі;
- Jupyter Notebook (.ipynb) – analysis_model.ipynb – дослідницький аналіз даних та створення прототипів моделі до формалізації конвеєра;
- Streamlit – панель керування фронтендом (fraud_detection.py). Слугує для демонстрації результатів виявлення шахрайства через веб-панель

інструментів або інтерфейс користувача. Налаштовується через .streamlit/config.toml;

3.2. Структура проєкту та його реалізація

Для повноцінного запуску проєкта необхідно перевірити наявність всіх необхідних інструментів та технології розробки (рис 3.1.). У разі їх відсутності, завантаження можливе командою “pip install -r requirements.txt”.

requirements.txt

```
pandas
numpy
scikit-learn
xgboost
joblib
matplotlib
seaborn
apache-airflow==2.7.1
pyspark==4.1.1
```

Рисунок 3.1. requirements.txt

Після інспектування наявності всіх необхідних бібліотек, слід запустити Docker командою “docker-compose up --build -d”. Параметр ’--build’ виконує перекомпіляцію на основі файлу Dockerfile (враховуючи всі зміни), а ’-d’ запускає процес у фоновому режимі, щоб звільнити термінал. Ці дії дозволяють налаштувати базу метаданих – “docker-compose run --rm webserver airflow db init” (під час цього процесу створюються таблиці, необхідні для роботи Airflow). Створимо власні облікові дані – “docker-compose run --rm webserver airflow users create --username admin --password admin --firstname Admin --lastname User --role Admin --email admin@example.com”. Файл docker-compose.yaml запускає вебсервер, планувальник та обробник Airflow у вигляді ізольованих контейнерів. Перевіримо служби, які запустилися – “docker-compose ps”. Тепер Airflow має

сервер для роботи, відкриємо його через localhost:8080 і зайдемо, використовуючи логін і пароль admin/admin (рис 3.2.).

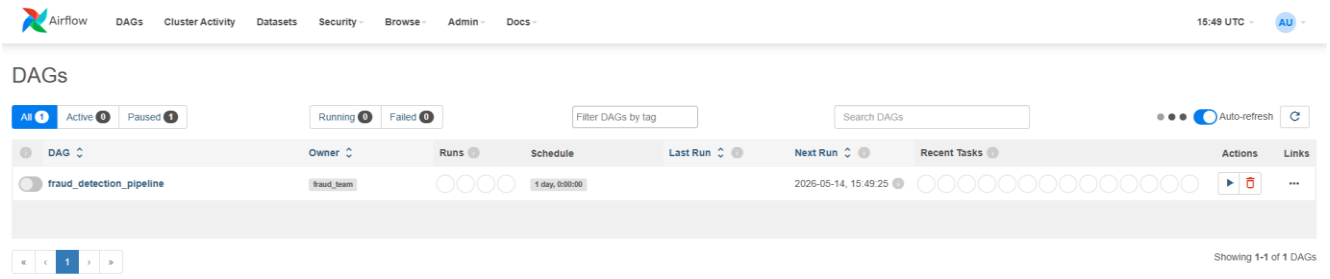

Рисунок 3.2. Apache Airflow

Переглянемо DAG (конвеєр ETL + навчання) і активуємо його вручну (рис. 3.3.). Це запускає весь конвеєр у такому порядку:

1. Витяг даних (читає файл AIML Dataset.csv);
2. Перевірка даних (схеми, нульові значення, дублікати);
3. Попередня обробка даних (формування функцій за допомогою Spark, запис їх у Parquet);
4. Завантаження і реєстрація (перевіряє Parquet, створює лог аудиту ETL);
5. Паралельне тренування трьох моделей (Logistic Regression, Random Forest, XGBoost);
6. Оцінка та застосування (обирає найкращу модель і зберігає її).

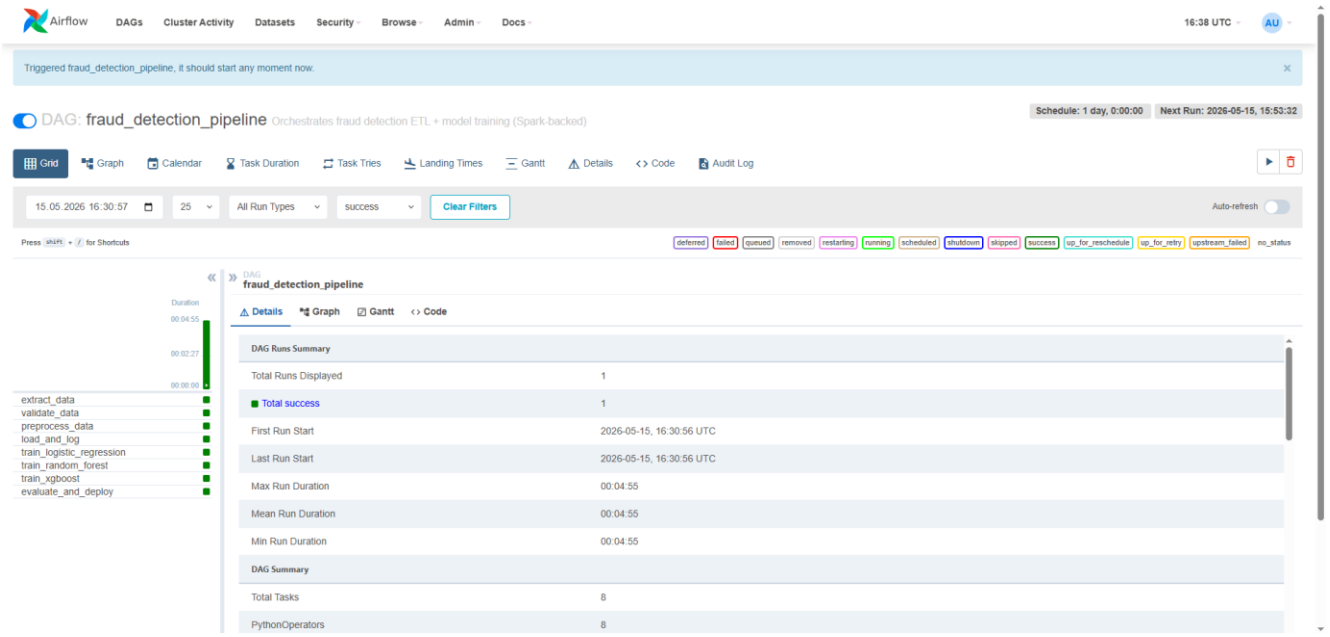

Рисунок 3.3. Успішний запуск DAG

Тепер запустимо панель управління Streamlit у файлі fraud_detection.py командою “streamlit run fraud_detection.py”, який відкриється за адресою localhost:8501 (рис. 3.4.). Тут можна побачити інтерфейс користувача, який використовує найкращу модель з трьох натренированих – best_fraud_detection_model.pkl. У файлі config.toml було збільшено максимальний розмір бази даних, яка може бути використана для аналізу великого обсягу даних за допомогою Spark.

Рисунок 3.4. Запуск Streamlit у терміналі Visual Studio Code

Інтерфейс містить дві вкладки. Перша слугує для прогнозування шахрайських операцій щодо окремої транзакції (рис. 3.5.).

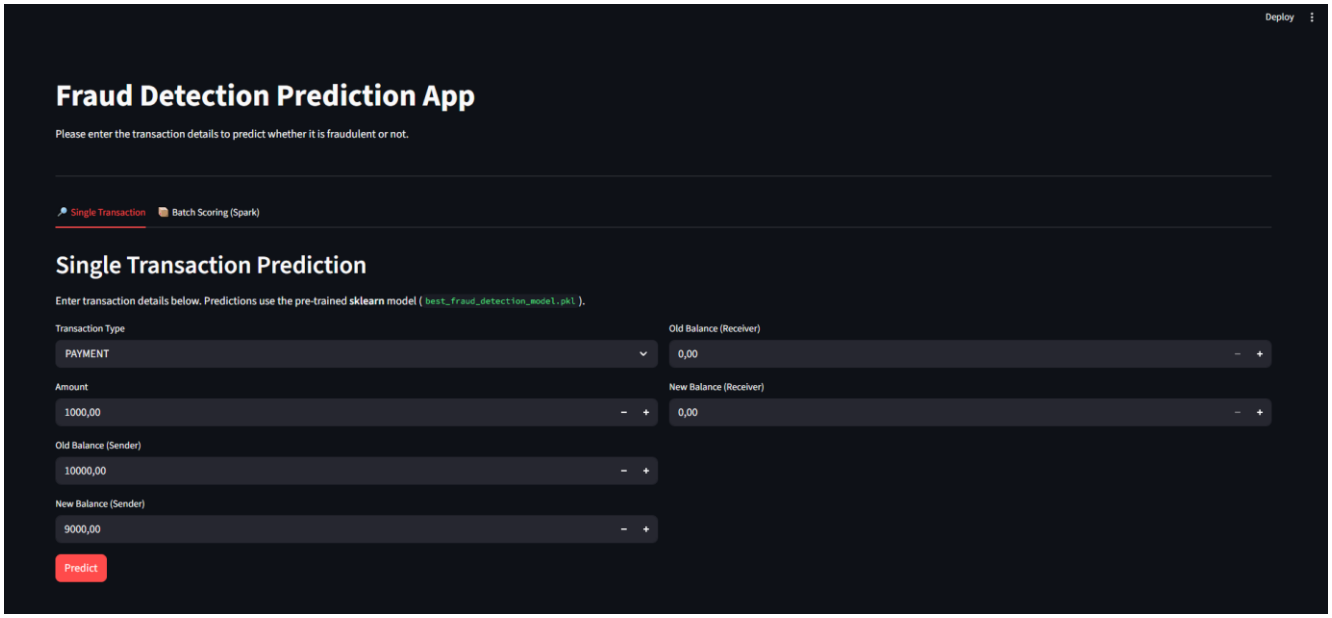

Рисунок 3.5. Інтерфейс Streamlit

Користувачі вручну вводять конкретні деталі транзакції за допомогою веб-форм, включаючи тип транзакції (наприклад, платіж, переказ), суму та залишок як

відправника, так і одержувача до та після транзакції. Після натиснення кнопки ’Predict’ програма відображає класифікацію транзакції як шахрайську – fraudulent (рис. 3.6.).

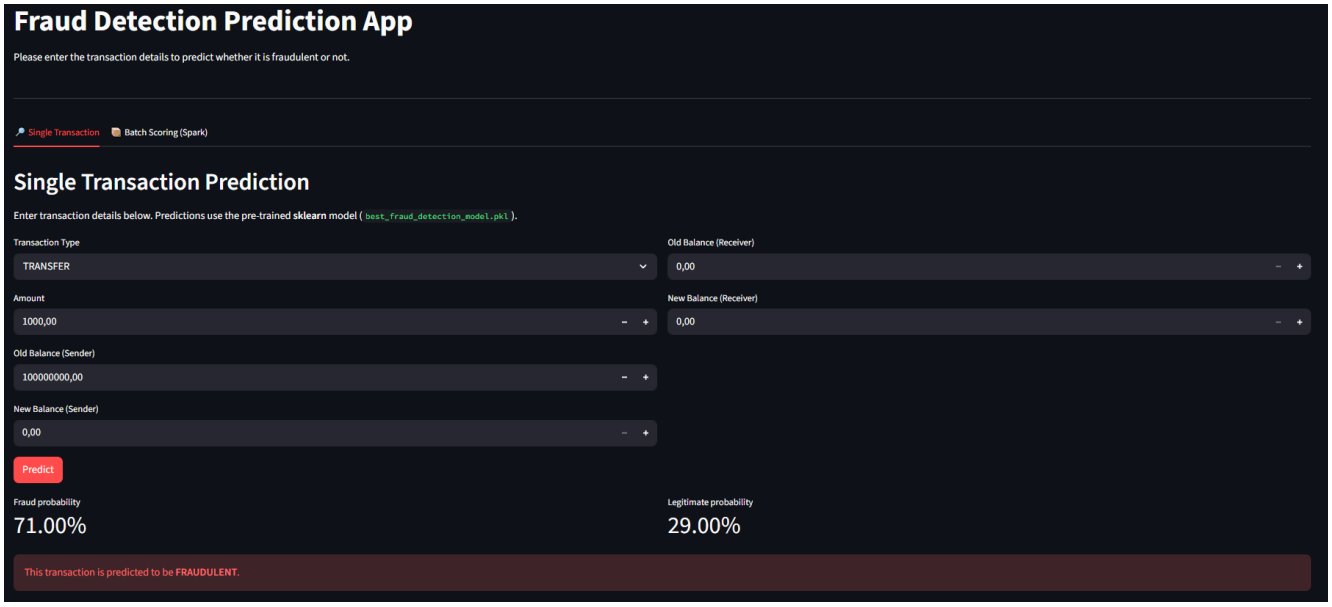

Рисунок 3.6. Ймовірність шахрайської операції

В іншому випадку імовірність шахрайства занадто мала, тому операція вважається законною – legitimate (рис. 3.7.).

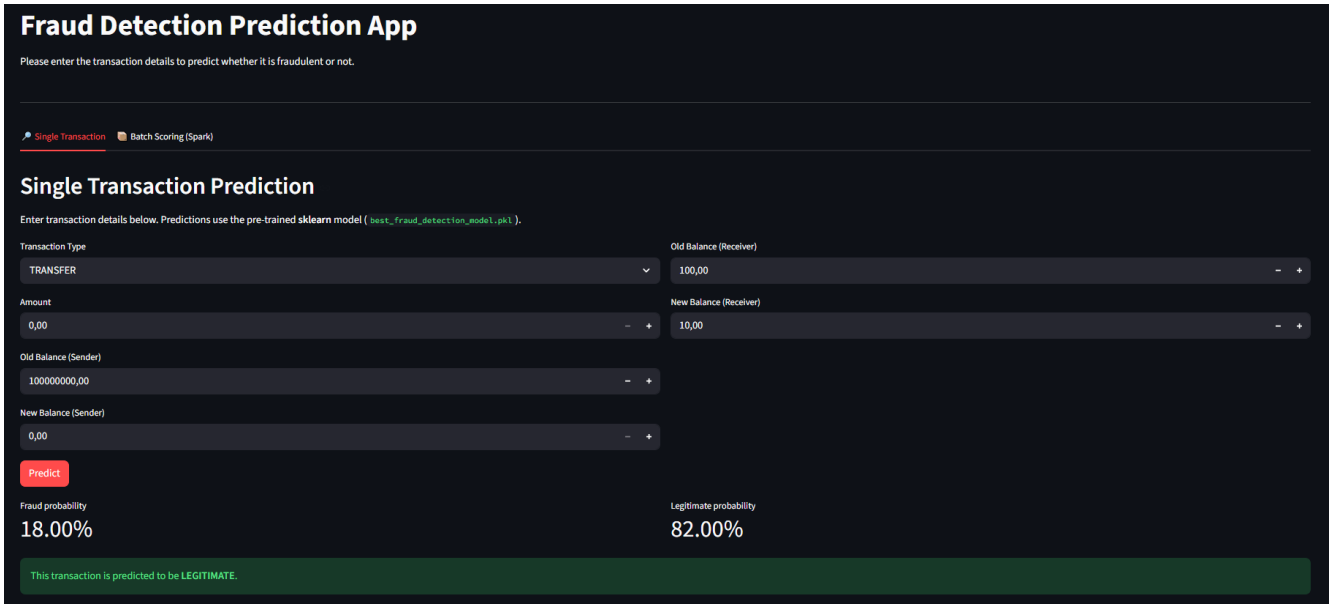

Рисунок 3.7. Візуальний результат законної операції

Пакетна ж оцінка за допомогою PySpark потребує попереднього завантаження бази даних користувачами, після цього можна розпочати обробку, натиснувши ’Run Batch Scoring’ (рис. 3.8.). Додаток очікує наявність певних стовпців, таких як тип платежу, сума та дані про баланс.

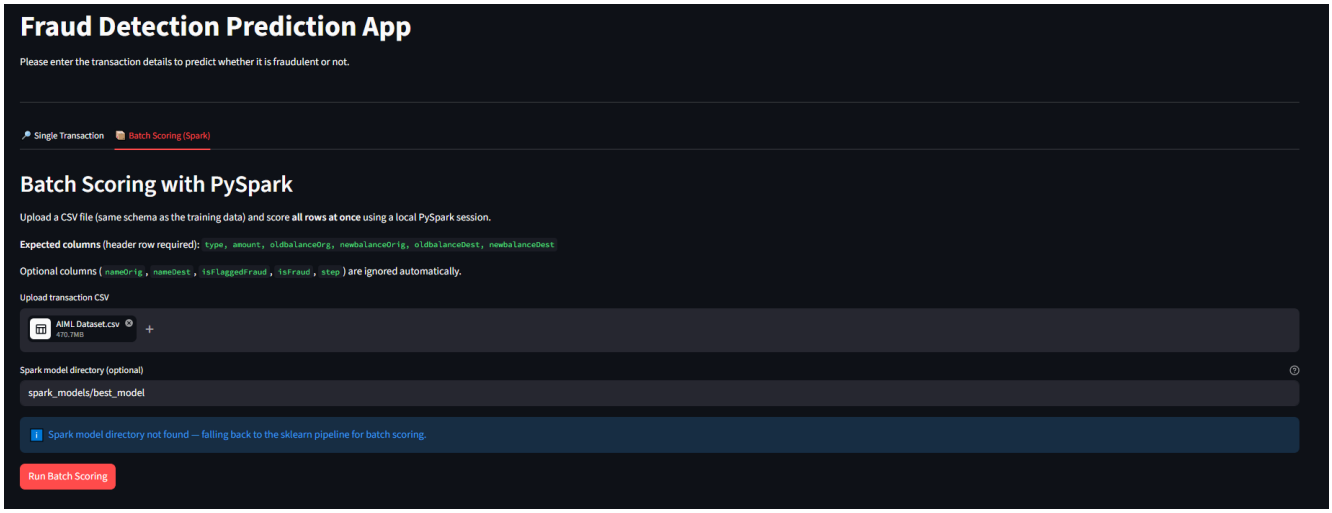

Рисунок 3.8. Друга вкладка інтерфейсу Streamlit

Додаток автоматично очищає дані, видаляючи непотрібні стовпці (наприклад, step або nameOrig), а також виконує обробку даних для обчислення різниці в залишках як для відправника, так і для одержувача (balanceDiffOrig та balanceDiffDest). Також є підтримка подвійного бекенду – додаток намагається використовувати PySpark PipelineModel для розподіленого оцінювання, якщо вказано шлях до моделі. Якщо ж модель Spark не знайдено, додаток автоматично переходить до використання моделі scikit-learn для обробки пакета даних (рис. 3.9.). Після опрацювання додаток надає вичерпний звіт, що включає в себе:

- Загальну кількість оцінених транзакцій;
- Кількість передбачених випадків шахрайства;
- Загальний рівень шахрайства;
- Інтерактивна таблиця, що відображає перші 200 результатів із індивідуальною ймовірністю махінацій;

- Список позначених операцій: спеціальний розділ, у якому перелічено лише транзакції, позначені як шахрайські, відсортовані за найвищою ймовірністю;
- Кнопка для завантаження всього набору прогнозів у вигляді нового файлу CSV.

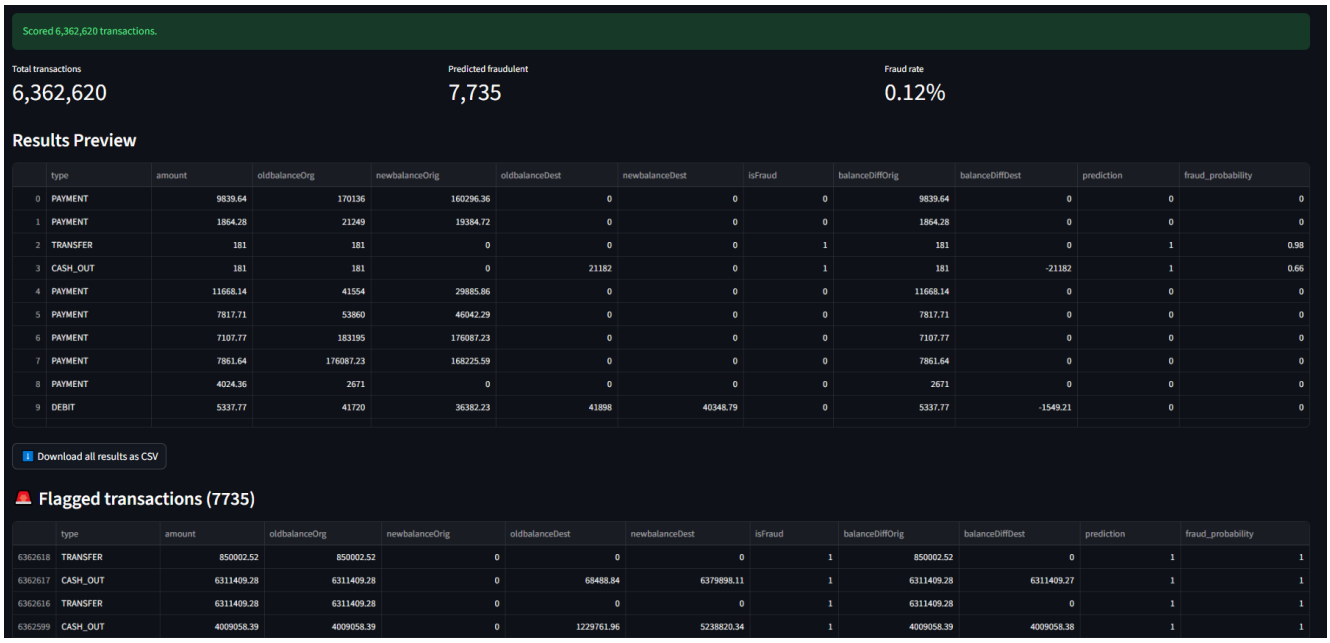

Рисунок 3.9. Результати роботи PySpark

3.3. Числові результати проєкту

Датасет містить 6 362 620 транзакцій і 11 ознак, займаючи в пам’яті понад 534 МБ – на межі того, що можна комфортно обробляти локально без розподіленого середовища, звідси й обґрунтованість використання Spark.

Розподіл цільової змінної критично нерівномірний: шахрайських транзакцій лише 8 213 з 6 354 407 легітимних – це 0,13% від загального обсягу. Саме такий дисбаланс є головним викликом задачі: модель, яка просто класифікує все як “не шахрайство”, одразу матиме 99,87% точності, не навчившись нічого корисного. Система isFlaggedFraud (вбудований у датасет детектор шахрайства) позначила лише 16 транзакцій зі 8 213 реальних – тобто пропустила 99,8% шахрайства. Це наочно показує, чому потрібен ML замість простих правил.

Середня сума транзакції – 179 861, медіана – 74 871, максимум – 92 445 516. Такий великий розкид (стандартне відхилення 603 858) пояснює необхідність логарифмічного масштабування при візуалізації та стандартизації перед навчанням. 1 399 253 транзакцій мають від'ємний balanceDiffOrig (баланс відправника зріс після транзакції) – аномалія, яка може вказувати на технічні помилки в даних або складні схеми переказів. Аналогічно, 1 238 864 записів мають від'ємний balanceDiffDest. Ці артефакти – важлива деталь: вони потрапляють у модель як ознаки і потенційно можуть вносити шум.

Кореляційний аналіз показав, що жодна окрема ознака не корелює з isFraud сильніше ніж сума транзакції, тобто 0,077 (рис. 3.10). Це означає, що шахрайство не вловлюється лінійними залежностями – потрібні нелінійні методи.

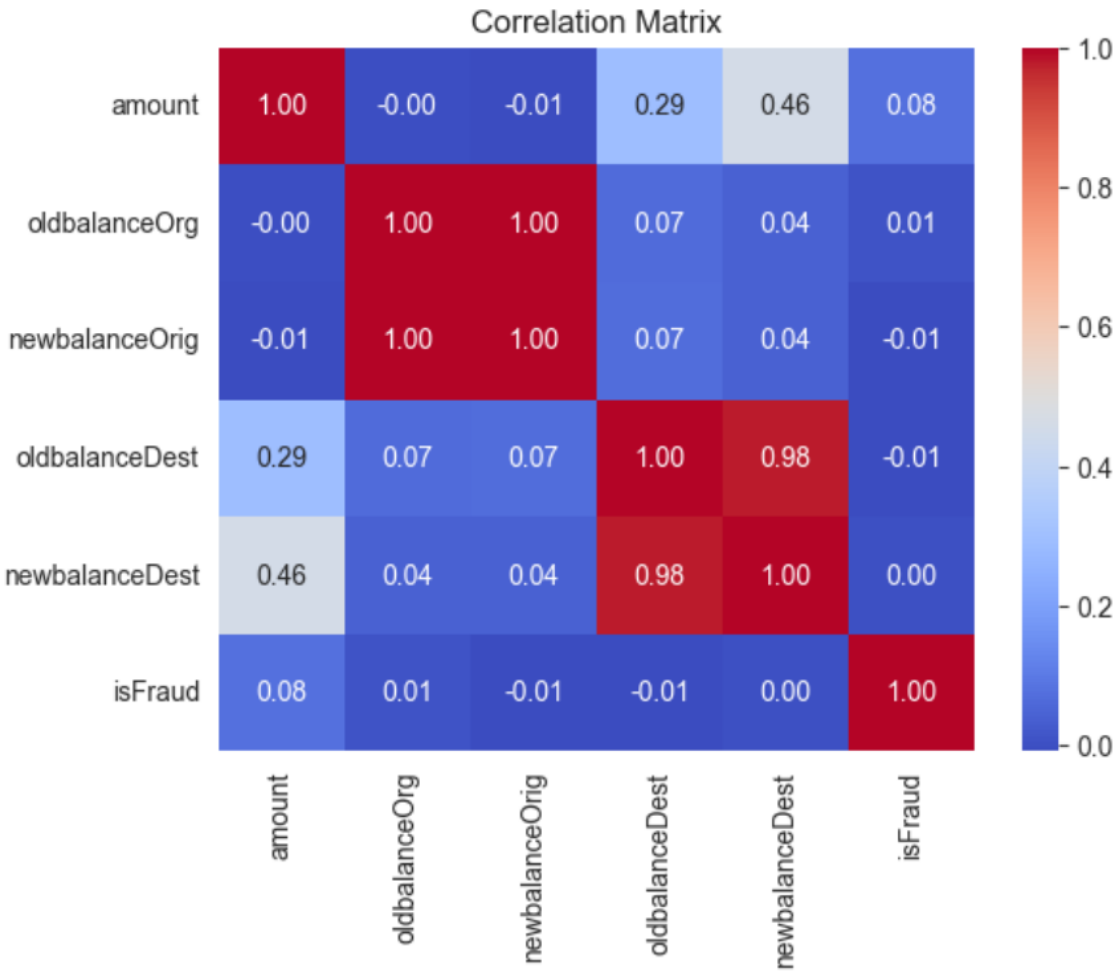

Рисунок 3.10. Матриця лінійної кореляції ознак датасету

Показово, що шахрайство зустрічається у двох типах транзакцій: TRANSFER і CASH_OUT. Серед відфільтрованих 2 770 409 таких операцій – 2 237 500 CASH_OUT і 532 909 TRANSFER. Ця закономірність логічна: саме ці типи дозволяють вивести кошти за межі системи. Ще один маркер – 1 188 074 транзакцій з ненульовим балансом відправника до переказу та нульовим після. Це майже кожна п'ята операція TRANSFER/CASH_OUT. Вони не всі шахрайські, але "обнулення рахунку" є одним із найсильніших сигналів для моделі.

Усі три моделі навчались на 70% датасету (~4,45 млн записів), тестувались на 30% (~1,91 млн), із стратифікованим розподілом класів.

Модель	Точність	F1 (шахрайство)	Достовірність	Повнота
Logistic Regression	94,47%	0,042	0,02	0,94
XGBoost	99,20%	0,243	0,14	1,00
Random Forest	99,97%	0,862	0,97	0,78

Logistic Regression – показова провальна точка відліку. Точність 94,47% виглядає пристойно, але F1 по класу шахрайства лише 0,042. Модель знайшла 2 308 шахрайських транзакцій, але видала 105 368 хибних спрацювань (false positives). Тобто з кожних 100 підозрілих транзакцій реальними шахрайськими є лише 2 – неприйнятно для практичного використання.

XGBoost з scale_pos_weight пішов в іншу крайність: повнота 1,00 (не пропустив жодного шахрайства в тесті), але достовірність лише 0,14. З 2 464 тестових шахрайств модель виявила всі, проте разом із ними позначила величезну кількість легітимних операцій.

Random Forest знайшов баланс, якого не вдалось досягти іншим. Достовірність 0,97 означає, що 97% підозрілих транзакцій справді є

шахрайськими. Повнота 0,78 – модель пропускає приблизно кожне п’яте реальне шахрайство, але при цьому не завалює аналітиків хибними спрацюваннями. Саме тому він був обраний як best_fraud_detection_model.pkl.

ВИСНОВКИ

Бакалаврська робота присвячена побудові системи інженерії даних для задач машинного навчання, зосередженої на виявленні фінансового шахрайства в транзакціях. Теоретична частина охопила концепцію інженерії даних, порівняння парадигм ETL та ELT, характеристики великих даних (Big Data) та інструменти їх обробки. Практична частина – реалізований повний конвеєр обробки даних і навчання моделей на реальному датасеті фінансових транзакцій.

Проєкт побудовано на стеку Apache Airflow + Apache Spark + Scikit-learn + XGBoost і контейнеризовано через Docker Compose. DAG-конвеєр послідовно виконував шість кроків: вилучення CSV, валідацію схеми, трансформацію через Spark із записом у Parquet, завантаження та реєстрацію, паралельне навчання трьох моделей (Logistic Regression, Random Forest, XGBoost), і кінцевий вибір найкращої за метриками. Для обробки дисбалансу класів у XGBoost застосовувався параметр scale_pos_weight, що критично важливо при частці шахрайства менше 1%. Підсумкова модель збережена у форматі .pkl і передана до Streamlit-інтерфейсу як для покрокового введення транзакцій, так і для пакетного оцінювання через PySpark.

Система оброблена на датасеті обсягом 6 362 620 транзакцій. З них модель позначила як шахрайські 7 735 операцій, що дає загальний рівень шахрайства 0,12% – це реалістичне значення для фінансових датасетів, де шахрайські транзакції завжди становлять мізерну частку загального потоку. Показовим є приклад із двох тестових сценаріїв: переказ із повністю спорожненим рахунком відправника (баланс після транзакції = 0) отримав 71% ймовірності шахрайства, тоді як аналогічна операція з адекватними залишками – лише 18%. Це свідчить, що модель коректно засвоїла ознаку “обнулення рахунку” як основний маркер підозрілої активності.

Розроблена система вирішує конкретну бізнес-задачу: автоматична перевірка транзакцій без участі людини на всіх шести мільйонах записів, з видачею ймовірності шахрайства та окремим звітом по позначених операціях.

Інтерфейс Streamlit дозволяє аналітику як перевіряти поодинокі транзакції в реальному часі, так і завантажувати CSV-файл для пакетного оброблення – результат доступний для завантаження у тому самому форматі. Практична цінність роботи не обмежується виявленням шахрайства. Реалізований конвеєр – це шаблон, який можна перевикористати для будь-якого схожого завдання: детекції аномалій у страхових виплатах, аудиту медичних записів, моніторингу мережевого трафіку.

Серед перспектив подальшого розвитку можна зазначити кілька напрямків:

По-перше, перехід до потокової обробки в реальному часі через Apache Kafka. Зараз конвеєр працює пакетно за розкладом; підключення Kafka дозволило б перевіряти транзакції ще до їх підтвердження.

По-друге, впровадження MLflow або аналогічного трекера експериментів для версіонування моделей. Нині вибір “найкращої моделі” зафіксований логікою конвеєра; в промисловому середовищі потрібен реєстр, який дозволяє відкат до попередньої версії при деградації якості.

По-третє, розширення ознакового простору з залученням графових методів – адже шахрайські схеми часто реалізуються через мережі пов’язаних рахунків, а не через окремі транзакції. GraphML або Apache Spark GraphX могли б виявляти такі патерни там, де табличні моделі не справляються.

Також система потребує механізму моніторингу дрейфу даних (data drift). Шахрайські схеми змінюються; модель, навчена сьогодні, може втратити точність через кілька місяців без перенавчання на нових прикладах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Основи машинного навчання : навч. посіб. / В. О. Харченко. – Суми : Сумський державний університет, 2023. – 264 с
2. Pankiv V. I., Storozhuk O. L. Використання нейронних мереж та технік глибокого навчання для аналізу великих обсягів даних у реальному часі. Forestry Education and Science: Current Challenges and Development Prospects. International Science-Practical Conference, October 23-25, 2024, Lviv, Ukraine. 2024.
3. Талах М.В., Технології обробки Big Data. Навчальний посібник/ Талах М.В., – Чернівці: Чернівецький нац.ун-т, 2024. – 454 с.
4. Технології оброблення великих даних: конспект лекцій з дисципліни «Технології оброблення великих даних» [Електронний ресурс] : навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення» (освітня програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»)/ Л.М. Олещенко; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 5,55 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 227 с.
5. Захист інформації в комп’ютерних системах та мережах. Частина ІІ: підручник / Ю.В. Костюк, П.М. Складанний. – Київ : Київський столичний університет імені Бориса Грінченка, 2026. – 386 с.
6. Deep Neural Networks and Tabular Data: A Survey / V. Borisov та ін. IEEE Transactions on Neural Networks and Learning Systems, vol. 35, no. 6, 2024 P. 1–21.
7. Grinsztajn L., Oyallon E., Varoquaux G. Why Do Tree-Based Models Still Outperform Deep Learning on Typical Tabular Data?. Advances in Neural Information Processing Systems 35, New Orleans, Louisiana, USA, 28 November – 9 December 2022. San Diego, California, USA, 2022. P. 507–520.

8. Hancock J. T., Khoshgoftaar T. M. Survey on categorical data for neural networks. Journal of Big Data. 2020. Vol. 7, no. 1, pp. 1–41.
9. Erl T., Khattak W., Buhler P. Big Data Fundamentals: Concepts, Drivers and Techniques. Pearson Education Canada, 2016. 240 p.
10. Densmore J. Data Pipelines Pocket Reference: Moving and Processing Data for Analytics. O'Reilly Media, 2021. 276 p.
11. R. Ananthanarayanan, V. Basker, S. Das, A. Gupta, H. Jiang, T. Qiu, A. Reznichenko, D. Ryabkov, M. Singh, and S. Venkataraman. Photon: Fault-tolerant and scalable joining of continuous data streams. In SIGMOD ’13: Proceedings of the 2013 international conference on Management of data, pages 577– 588, New York, NY, USA, 2013.
12. T. M. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In 11th USENIX Symposium on Operating Systems Design and Implementation, OSDI ’14, Broomfield, CO, USA, October 6-8, 2014., pages 571–582, 2014.
13. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, Incorporated, 2017. 616 p.
14. Inmon W. H. Building the Data Warehouse. Wiley & Sons, Incorporated, John, 2002.
15. Reinsel D., Gantz J., Rydning J. Data Age 2025: The Evolution of Data to Life-Critical : IDC White Paper. Seagate, 2018. 28 p.
16. Reis J., Housley M. Fundamentals of Data Engineering: Plan and Build Robust Data Systems. O'Reilly Media, Incorporated, 2022. – 414 p.
17. Kleppmann M., Riccomini C. Designing Data-Intensive Applications. 2nd ed. (Early Release). O'Reilly Media, 2024.
18. Kimball R., Ross M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. 3rd ed. – Indianapolis: John Wiley & Sons, 2013. – 600 p.
19. Chan Y., Talburt J., Talley T. M. Data Engineering: Mining, Information and Intelligence. Springer, 2010. 468 p.

20. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. – Hoboken: Pearson, 2020. – 1132 p.
21. Mitchell T. M. Machine Learning. McGraw-Hill Science/Engineering/Math, 1997. 432 p.
22. Bengio Y., Courville A., Goodfellow I. Deep Learning. MIT Press, 2016. 800 p.
23. Zaharia M. et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing // NSDI ’12: Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation. – San Jose, CA, 2012. – P. 2–14.
24. Narkhede N., Shapira G., Palino T. Kafka: The Definitive Guide. Real-Time Data and Stream Processing at Scale. 2nd ed. – Sebastopol: O’Reilly Media, 2021. – 488 p.
25. Що таке інженерія даних? - Integral Solutions. URL: https://integralsolutions.pl/uk/czym-jest-inzynieria-danych/ (дата звернення: 02.05.2026).
26. Основні принципи програмної інженерії. StudFiles. URL: https://studfile.net/preview/9295990/page:2/ (дата звернення: 02.05.2026).
27. Belgaokar P. The Role of Data Engineers in Modern Data Ecosystems. LinkedIn: Log In or Sign Up. URL: https://www.linkedin.com/pulse/role-data-engineers-modern-ecosystems-pratik-belgaokar-hseyf?utm_source=share&utm_medium=guest_desktop&utm_campaign=copy (дата звернення: 03.05.2026).
28. Довідник по Machine Learning – ETL Pipeline | База знань IT технологій. URL: https://itwiki.dev/data-science/ml-reference/ml-glossary/etl-pipeline (дата звернення: 03.05.2026).
29. Підручник з тестування ETL. Guru99. URL: https://www.guru99.com/uk/utlimate-guide-etl-datawarehouse-testing.html (дата звернення: 04.05.2026).

30. Reeves J. Яким має бути хороший дата-пайплайн?. LinkedIn Україна: увійдіть або зареєструйтеся. URL: https://ua.linkedin.com/pulse/how-good-data-pipeline-should-joseph-reeves-uolqc?tl=uk (дата звернення: 04.05.2026).
31. Налаштування конвеєра даних для надійної та масштабованої моделі ML | Шайп. Shaip. URL: https://uk.shaip.com/blog/setting-up-data-pipeline-to-build-scalable-ml-model/ (дата звернення: 05.05.2026).
32. GeeksforGeeks. ETL Process in Data Warehouse - GeeksforGeeks. URL: https://www.geeksforgeeks.org/dbms/etl-process-in-data-warehouse/ (дата звернення: 05.05.2026).
33. ETL або ELT: який процес роботи з даними дає оптимальний результат? – RBC group. URL: https://www.rbcgrp.com/ua/etl-ili-elt-kakoj-process-raboty-s-dannymi-daet-optimalnyj-rezultat/ (дата звернення: 06.05.2026).
34. Amjad A. ETL проти ELT: розуміння відмінностей, переваг і недоліків. LinkedIn Україна: увійдіть або зареєструйтеся. URL: https://ua.linkedin.com/pulse/etl-vs-elt-understanding-differences-advantages-asad-amjad-qvmkf?tl=uk (дата звернення: 06.05.2026).
35. Що таке навчання з підкріпленням? Розбираємо теорію і реальні кейси. Evergreen - web розробка і діджиталізація бізнесу за gдопомогою AI продуктів. URL: https://evergreens.com.ua/ua/articles/reinforcement-learning.html (дата звернення: 07.05.2026).
36. Методи машинного навчання | Розділ 4. Ключові методи ШІ | Застосування штучного інтелекту у сфері П(ПТ)О Навчальна Програма | Profosvita. URL: https://profosvita.online/courses/course-v1:Profosvita+CM-R042-OEP+2025/courseware/1ca33a8f1bb34f02ba777dcd36616c91/b394a5e241f24933830ccf5fc1849463/3?activate_block_id=block-v1:Profosvita+CM-R042-OEP+2025+type@vertical+block@541503a1faf848188f6610fe549c8657 (дата звернення: 07.05.2026).
37. Учасники проєктів Вікімедіа. Навчання з підкріпленням – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Навчання_з_підкріпленням (дата звернення: 07.05.2026).

38. GeeksforGeeks. Unsupervised Machine Learning - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/machine-learning/unsupervised-learning/ (дата звернення: 09.05.2026).
39. Розподілена система зберігання даних: Типи та реальні приклади - Блог - HostZealot. HostZealot. URL: https://www.hostzealot.com.ua/blog/about-servers/rozpodilena-sistema-zberigannya-danix-tipi-ta-realni-prikladi (дата звернення: 11.05.2026).
40. What is Object Storage? - Cloud Object Storage Explained - AWS. Amazon Web Services, Inc. URL: https://aws.amazon.com/what-is/object-storage/ (дата звернення: 11.05.2026).
41. What Is NoSQL? NoSQL Databases Explained. MongoDB: The World's Leading Modern Data Platform | MongoDB. URL: https://www.mongodb.com/resources/basics/databases/nosql-explained (дата звернення: 13.05.2026).
42. Що таке NAS і навіщо він потрібен? Основні випадки, за яких застосовується NAS | ERC Ukraine. URL: https://erc.ua/magazine/24598/shcho-take-nas-i-navishcho-vin-potriben-osnovni-vipadki-za-iakikh-zastosovuietsia-nas (дата звернення: 13.05.2026).
43. Учасники проєктів Вікімедіа. Мережа зберігання даних – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Мережа_зберігання_даних (дата звернення: 14.05.2026).
44. Staff D. What is Apache Spark?. Databricks. URL: https://www.databricks.com/blog/what-is-apache-spark (дата звернення: 14.05.2026).
45. Data Locality: HPC vs. Hadoop vs. Spark | Data Science Association. Wayback Machine. URL: https://web.archive.org/web/20170910220353/http://www.datascienceassn.org/content/data-locality-hpc-vs-hadoop-vs-spark (дата звернення: 15.05.2026).
46. IBM - What is the Hadoop Distributed File System (HDFS) - United States. Wayback Machine. URL:

https://web.archive.org/web/20170108001542/http://www-01.ibm.com/software/data/infosphere/hadoop/hdfs/ (дата звернення: 15.05.2026).

47. Apache Kafka: гайд для початківців за 8 хвилин. IT Education Center Blog. URL: https://itedu.center/ua/blog/sysadministration/apache-kafka-guide-for-beginners/ (дата звернення: 15.05.2026).
48. What is Kafka? Topics, Producers, Consumers, Brokers Explained | Confluent Documentation. Confluent Documentation. URL: https://docs.confluent.io/kafka/introduction.html#partitions (дата звернення: 15.05.2026).
49. What Happens in an Internet Minute in 2026 – Media Usage Statistics. businesstats.com. URL: https://businesstats.com/new-user-generated-content-uploaded-by-users-per-minute/ (дата звернення: 15.05.2026).
50. Redman T. C. Seizing Opportunity in Data Quality. MIT Sloan Management Review. URL: https://sloanreview.mit.edu/article/seizing-opportunity-in-data-quality/ (дата звернення: 15.05.2026).