Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ВИКОРИСТАННЯ PYSPARK ДЛЯ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ТА ВАЛІДАЦІЇ BIG DATA

Текстова частина
магістерської роботи
за спеціальністю „Інженерія програмного забезпечення” 121

Керівник магістерської роботи
д.т.н., проф. Глибовець А.М.

_____
(підпис)

“____” _____ 2024 р.

Виконав студент
Полінчук К.І.

“____” _____ 2024 р.

Київ 2024

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри інформатики
к.ф-м.н., доц. Гороховський С. С.
_____
(підпис)
“_____” _____2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на магістерську роботу

студенту 2 р.н. магістерської програми Інженерія Програмного Забезпечення Полінчуку Кирилу Ігоровичу
Дослідити Застосування PySpark для забезпечення якості та валідації Big Data

Зміст текстової частини до магістерської роботи:

Зміст
Анотація
Вступ
1 Теоретичні основи валідації Big Data.
2 Технологічні інструменти валідації Big Data.
3 Застосування PySpark для валідації та покращення якості даних у Data Lakehouse архітектурі.
Висновки по роботі та рекомендації для подальших досліджень
Список літератури
Додатки

Дата видачі “_____” _____ 2023 р.

Керівник
А.М. Глибовець, доктор технічних наук, професор _____
(підпис)

Завдання отримав
К.І. Полінчук _____
(підпис)

Тема: Застосування PySpark для забезпечення якості та валідації Big Data

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу	06.09.2023	
2.	Огляд технічної літератури за темою роботи	20.09.2023	
3.	Ознайомлення з технічною документацією рішень на базі Apache Spark	04.10.2023	
4.	Ознайомлення з можливостями машинного навчання для оптимізації процесу валідації	18.10.2023	
5.	Вибір методів для порівняння технічних інструментів валідації з Big Data	14.02.2024	
6.	Огляд можливостей безкоштовної версії DataBricks для побудови ETL конвейєра	28.02.2024	
7.	Побудова ETL конвейєра з валідацією даних	16.04.2024	
8.	Підбір методів для аналізу ефективності застосування PySpark	30.04.2024	
9.	Дослідження ефективності застосування методів машинного навчання	10.05.2024	
11.	Аналіз отриманих результатів, написання доповіді та попередній захист магістерської роботи	17.05.2024	
12.	Коригування роботи за результатами попереднього захисту	30.05.2024	
14.	Захист магістерської роботи (проекту)	10.06.2024	

Студент Полінчук К.І.

Керівник Глибовець А.М. “____”_____ 2024 р.

ЗМІСТ

Анотація 5

Список умовних скорочень 7

ВСТУП 8

РОЗДІЛ 1: Теоретичні засади валідації Big Data 11

1.1 Veracity як одна з ключових характеристик Big Data 11

1.2 Проблеми забезпечення якості Big Data 15

1.3 Валідація даних у ETL і ELT конвейєрах 22

РОЗДІЛ 2: Технологічні інструменти валідації BigData 28

2.1 Застосування Apache Spark у процесі валідації Big Data 28

2.2 Особливості валідації Big Data на платформі DataBricks 40

2.3 Застосування методів машинного навчання для валідації Big Data 50

РОЗДІЛ 3: Застосування PySpark для валідації та покращення якості даних у Data Lakehouse архітектурі 60

3.1 Застосування методів машинного навчання для валідації та покращення якості даних 60

3.2 Аналіз результатів валідації та корекції даних 63

3.3 Рекомендації щодо оптимізації та покращення якості даних у Data Lakehouse архітектурі 67

Висновки 70

Оцінка новизни роботи 72

Список літератури 74

АНОТАЦІЯ

Ця дипломна робота присвячена вивченню та впровадженню методів валідації великих даних (Big Data) з використанням сучасних технологій, таких як машинне навчання, та інструментів, таких як Apache Spark. Робота охоплює теоретичні аспекти валідації даних, розглядає ключові характеристики та проблеми, пов'язані з великими даними, а також описує практичні підходи до їх обробки та забезпечення якості.

У ході дослідження було розроблено ETL-конвеєр з обробки та валідації даних на базі PySpark. Цей конвеєр дозволяє ефективно здійснювати вилкучення, трансформацію та завантаження даних, забезпечуючи їх чистоту та консистентність. З метою оптимізації роботи та зменшення навантаження методами машинного навчання було класифіковано дані, які потребували корекції, та проведено оцінку ефективності застосування методів машинного навчання для покращення процесу валідації Big Data. Було використано алгоритми кластеризації для виявлення аномалій, очищення даних та їх автоматичної корекції.

Результати роботи показують, що використання машинного навчання значно підвищує якість даних, дозволяючи автоматизувати процеси валідації та знижувати потребу в ручній праці. Проведено аналіз продуктивності та точності моделей, які використовуються для валідації даних, а також розглянуто методи їх інтеграції з інструментами обробки великих даних.

Особлива увага була приділена виявленню та корекції аномалій у даних, що включало використання кластеризаці. Це дозволило зменшити кількість помилок та підвищити загальну якість даних, що використовуються для подальшого аналізу та прийняття рішень.

Результати роботи можуть бути використані у різних галузях, де великі обсяги даних є основою для прийняття рішень, таких як фінансовий сектор, охорона здоров'я, маркетинг та виробництво. Застосування

розроблених методів та інструментів дозволяє не тільки забезпечити високу якість даних, але й підвищити ефективність бізнес-процесів, знизити витрати та покращити точність прогнозів та аналітичних висновків.

Ключові слова: Apache Spark, Big Data, DataBricks, ELT, ETL, EtLT, Pandas, Parquet, PySpark, Veracity, Валідація, Великі Дані, Верифікація, Класифікація, Кластеризація, Машинне Навчання.

СПИСОК УМОВНИХ СКОРОЧЕНЬ

1. ACID - Atomicity, Consistency, Isolation, Durability
2. API - Application Programming Interface
3. AWS - Amazon Web Services
4. B2B - Business to Business
5. COVID-19 - Coronavirus Disease 2019
6. DLT - Delta Live Tables
7. DQM - Data Quality Management
8. ETL - Extract, Transform, Load
9. ELT - Extract, Load, Transform
10. GCP - Google Cloud Platform
11. GDPR - General Data Protection Regulation
12. HIPAA - Health Insurance Portability and Accountability Act
13. ISO - International Organization for Standardization
14. IoT - Internet of Things
15. IT - Information Technology
16. ML - Machine Learning
17. NLP - Natural Language Processing
18. NoSQL - Not Only SQL (Structured Query Language)
19. SVM - Support Vector Machine
20. SQL - Structured Query Language
21. USA - United States of America

ВСТУП

Актуальність. Сьогоднішню епоху можна назвати унікальною в людській історії. Світ входить у нову фазу свого розвитку — інформаційну. Філософи та науковці називають її по-різному: інформаційною епохою (Мануель Кастельс), постіндустріальним суспільством (Деніел Белл), "економікою знань" (Пітер Друкер), суспільством "третьої хвилі" (Елвін Тоффлер) чи "плинною модерністю" (Зиґмунт Бауман). Важливим атрибутом цього періоду стають дані, які є найціннішим ресурсом для прийняття стратегічних рішень в різних галузях, але їх обсяг збільшується щосекунди.

Зростає не лише обсяг, а й кількість джерел і напрямків використання: IoT, соціальні мережі, комерційні угоди, B2C послуги – це лише декілька сфер, де постійно утворюються нові пари обміну інформації Producer-Consumer.

Однак, разом із зростанням обсягу даних, виникають виклики з їх обробки, зберігання й аналізу. Окремої уваги в цьому заслуговують валідація та верифікація даних. Ці два різні процеси дозволяють відфільтрувати непотрібні, порожні або дані з низькою інформаційною цінністю, а також перевірити їх якість на кожному етапі, щоб запобігти перевантаження мережі чи сховища низькоякісними даними. У цьому контексті виникає необхідність у високоефективних технічних інструментах та методах, які забезпечують якісну валідацію та верифікацію даних у великих обсягах.

Дана дипломна робота спрямована на дослідження технічних аспектів валідації Big Data, з основним акцентом на використанні інструментів Apache Spark та його компонентів, а також методів машинного навчання – таких як класифікація та кластеризація для оптимізації цього процесу.

Струткту́ра роботи. Перший розділ дипломної роботи присвячений теоретичним аспектам валідації Big Data, розглядає такі ключові поняття,

як Veracity та проблеми, які виникають у процесі валідації. Крім того, буде надано характеристику конвейєра даних ETL (Extract – Transform – Load), який виступає основною для обробки, упорядкування та зберігання даних.

У другому розділі роботи описуються архітектурні рішення та технології, що використовуються для валідації даних у великих обсягах. Цей розділ включає детальний опис архітектури Data Lakehouse та компонентів, що забезпечують її функціонування. Особлива увага приділяється використанню технологій Apache Spark, Delta Lake.

У третьому розділі розглядається застосування методів машинного навчання для валідації та покращення якості даних у Data Lakehouse архітектурі. Основна мета полягає у перевірці ефективності використання методів класифікації та кластеризації для ідентифікації та корекції пошкоджених даних. Описані процеси включають генерацію, збереження, оптимізацію, валідацію, корекцію та моніторинг даних.

Мета дослідження.

Метою даного дослідження є розробка ефективних методів валідації великих обсягів даних з використанням Apache Spark та методів машинного навчання для підвищення якості даних та оптимізації процесу їх обробки.

Завдання дослідження.

1. Провести огляд теоретичних аспектів валідації великих даних.
2. Розробити ETL-конвеєр для обробки великих обсягів даних з використанням PySpark.
3. Впровадити методи машинного навчання для виявлення та корекції аномалій у даних.
4. Оцінити ефективність застосування методів машинного навчання для валідації великих даних.
5. Провести експериментальні дослідження та аналіз результатів.

Предмет дослідження. ETL-пайплайн для обробки даних великого обсягу, практичне застосування Apache Spark для валідації даних та використання методів машинного навчання для підвищення якості даних.

Об'єкт дослідження. Підходи до роботи з Big Data, здійснення валідації великих даних, фреймворк Apache Spark та методи машинного навчання.

Джерела дослідження. Офіційна документація фреймворку Apache Spark, професійна література як у цифровому, так і в друкованому вигляді. Спеціалізовані веб-статті, профільні ресурси на теренах міжнародної мережі Інтернет. Приклади коду Python для роботи з даними.

Розділ 1 ТЕОРЕТИЧНІ ОСНОВИ ВАЛІДАЦІЇ BIG DATA

1.1 VERACITY ЯК ОДНА З КЛЮЧОВИХ ХАРАКТЕРИСТИК BIG DATA

Кембріджський словник визначає поняття BigData як великі набори даних, які виробляються людьми з використанням Інтернету та можуть зберігатися, аналізуватися та використовуватися лише за допомогою спеціальних інструментів і методів [1].

Хоча сама концепція великих даних є відносно новою, витоки великих наборів даних сягають 1960-х і 70-х років, коли світ даних тільки починався з першими центрами обробки даних і розвитком реляційних баз даних. Приблизно в 2005 році люди почали усвідомлювати, скільки даних користувачі генерують через Facebook, YouTube та інші онлайн-сервіси. Того ж року було розроблено Hadoop (платформа з відкритим вихідним кодом, створена спеціально для зберігання та аналізу великих наборів даних). У цей же час також почали набирати популярність і NoSQL бази даних.

Розробка фреймворків з відкритим кодом, таких як Hadoop (і пізніше Spark), була важливим етапом у розвитку Big Data, оскільки вони полегшують роботу з великими даними та здешевлюють їх зберігання. З кожним роком обсяги даних лише зростають. Користувачі все ще генерують величезні обсяги даних, але це роблять не лише люди [2].

З появою Інтернету речей (IoT) більше об’єктів і пристроїв підключаються до Інтернету, збираючи дані вподобання клієнтів і продуктивність застосунків. Поява машинного навчання створила ще більше даних, при чому не лише як виробник, а й як споживач – очищені та підготовлені дані завантажуються у попередньо натреновані моделі для отримання прогнозів.

Тому не дивно, що вже в 2011 році Gartner ставить Big Data на 2 місце серед 10 трендів інфраструктури й операцій IT (перше місце посіла віртуалізація). У коментарі Gartner було сказано наступне. «Великі дані, патерни й аналітика. Дані зростуть на 800% за 5 років, 80% з них будуть не структурованими. Частиною цього є тренд «колективні дані», який включає дані з груп, спільнот і соціальних мереж поза межами бізнесу» [3].

Oracle визначає Big Data як дані, що містять різноманітність, находять у великих обсягах і з великою швидкістю, відомі також як 3V. Це більші та складніші набори даних, настільки, що традиційне програмне забезпечення для обробки даних просто не може ними керувати, але їх використання дозволяє розв'язати нові бізнес проблеми [2].

Одними з перших концепцію 3 V використала MetaGroup у 2001 році, описуючи проблеми, з якими стикнуться центральні дата сховища компаній у зв'язку з розвитком E-commerce [4]. Коли традиційні методи зберігання, обробки та передачі інформації не змогли справлятися зі зростаючими V, прийшов час переглядати концепцію роботи з даними. Але ці три характеристики з часом теж уже не могли описати всі вимоги до даних. З'являється четверта V – Veracity.

Популяризатором цієї четвертої ознаки виступила компанія IBM, яка в 2011 році разом з вищеназваними трьома V назвали «правдивість» або «достовірність» даних ознакою Big Data. Для підтвердження актуальності цього питання було наведено наступні факти. Один з трьох лідерів бізнесу не довіряє інформації на основі якої він приймає рішення. 27% респондентів у ході опитування, що проводилось компанією не змогли вказати, яка кількість даних, яку вони використовують є неточною. Низька якість даних коштує економіці США 3.1 трлн доларів щорічно [5].

З точки зору корисності інформації її можна розділити на «сигнали» і «шуми». Шум – це дані, які не можуть бути перетворені в інформацію і тому не мають цінності, тоді як сигнали мають цінність і дають її. Дані, отримані

в контрольований спосіб, наприклад, через онлайн-реєстрацію клієнтів, зазвичай містять менше шуму, ніж дані, отримані через неконтрольовані джерела, такі як публікації в блогах. Таким чином, відношення сигнал/шум даних залежить від джерела даних і їх типу [6].

Табл. 1 Приклади даних з високим рівнем шуму.

Тип даних	Опис	Приклад
Неточні дані	Дані з некоректними значеннями	Контактні дані з невірними електроними адресами
Неповні дані	Дані з порожніми полями	Не вказано ключові дані користувача, наприклад, номер соціального страхування.
Суперечливі дані	Дані, що мають містити однакові атрибути, але містять різні	Дані одного користувача, але з різними адресами
Дублюючі дані	Ідентичні дані з різних джерел	Два записи про одного користувача з різних джерел
Пошкоджені дані	Дані з нестандартними атрибутами	4 цифри в поштовому індексі
Несинхроновані дані	Дані, що містяться в декількох системах, але не були однаково оновлені.	Адреса користувача була змінена лише в одній з систем.

Згідно звіту Gartner, дані низької якості щорічно вартують компаніям в середньому 12.9 мільйонів доларів США [7]. Це може проявлятися в наступному.

1. Втрачена вигода. Дані низької якості призводять до неточних прогнозів, втраченим можливостям щодо прибутку, втраті клієнтів.

2. Зростання операційних витрат. Такі дані можу призвести до низької ефективності налагоджених процесів, додаткові витрати через збій в роботі та виправлення помилок вручну.

3. Виплати через порушення. Штрафи через помилки при оподаткуванні, невірних виплат клієнтам.

4. Репутаційні ризики. Вплив на репутацію через скандали, втрати клієнтів. Тому забезпечення достовірності інформації, конвертація неструктурованих даних без помилок, перевірка повноти інформації є важливою задачею дата інженера, а якісна Big Data має ось це V серед своїх ознак – Veracity.

Збільшення обсягів і варіантів структурованості даних призводить до зростання можливостей для виникнення помилок, неточностей та викривлень в інформації. Дані з високим вмістом «шуму», помилки вводу, аномальні значення, можуть суттєво впливати на точність та достовірність висновків, зроблених на основі цих даних.

Для вирішення цих питань важливо використовувати методи валідації, очищення, стандартизації і верифікації даних. Автоматизовані засоби виявлення та виправлення помилок, вдосконалені процеси валідації та впровадження механізмів контролю якості можуть покращити достовірність даних.

Загалом можна виділити наступні засоби для підвищення достовірності даних.

1. Алгоритми обробки даних, які включають в себе видалення дублікатів, корекцію помилок та неповних записів, можуть значно покращити якість даних.

2. Застосування правил валідації для перевірки, чи відповідають дані певним стандартам та критеріям. Це може включати перевірку діапазону значень, перевірку формату та інші перевірки якості.

3. Постійний моніторинг та оновлення процесів обробки даних дозволяють вчасно виявляти проблеми та реагувати на них. Метрики якості даних можуть визначати, наскільки добре система впоралася з обробкою та збереженням даних.

1.2 ПРОБЛЕМИ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ BIG DATA

Big Data визначають нову еру інформаційного розвитку - разом з їх появою змінюються отримання, обробка й аналіз інформації. Спільне визначення та стандартизація критеріїв якості для великих даних залишаються актуальними завданнями для науковців і практиків. У цьому розділі ми розглянемо ключові проблеми, пов'язані із забезпеченням якості великих даних, проаналізуємо їх впливи на бізнес-середовище, а також визначимо характеристики якісних даних.

Можна виділити наступні проблеми забезпечення якості Big Data.

1. Різноманітність джерел даних призводить до необхідності використання одночасно даних різних за типом і структурою, що ускладнює процес їх консолідації.

У минулому підприємства використовували дані, що генерувалися власними бізнес-системами, наприклад, дані про продажі та залишки. Але зараз обсяги даних, які збираються та аналізуються підприємствами, виходять за межі цього обсягу. Джерела великих даних дуже різноманітні і включають: 1) набори даних з Інтернету та мобільного Інтернету [8]; 2) дані з Інтернету речей; 3) дані, зібрані різними галузями промисловості; 4) результати наукових експериментів і спостережень [9], такі як дані експериментів у фізиці, біологічні дані та дані спостережень у космосі.

Вище названі джерела генерують різні за структурою дані, які можна розділити на три групи за цією ознакою. Неструктуровані дані, наприклад, відео, аудіо і т.д. Напівструктуровані дані, до яких відносяться: програмні

пакети/модулі, дані у форматі json і т.п. Третій тип - це структуровані дані. Кількість неструктурованих або напівструктурованих даних становить більше 80% від загальної кількості наявних даних [8]. Отже, виникають конфлікти та неузгодженість між даними з різних джерел. У випадку невеликого обсягу даних їх можна перевірити за допомогою ручного пошуку чи програмування, навіть за допомогою ETL (Extract, Transform, Load) або ELT (Extract, Load, Transform) конвейєрів. Однак, навіть ці методи втрачають ефективність, коли мова заходить про терабайти, не говорячи про пента- й екзабайти.

2. Обсяг даних надто великий, і важко визначити якість даних протягом прийнятного проміжку часу.

Після промислової революції обсяги інформації подвоювалася кожні десять років. Починаючи з 1970 року, кількість інформації подвоюється кожні три роки. Сьогодні глобальний обсяг інформації може подвоюватися кожні два роки. У 2011 році обсяг світових даних, створених і скопійованих, досягнув 1,8 зеттабайтів [10].

У 2020 році обсяг створених та реплікованих даних досяг нового рекорду, коли було створено, зафіксовано, скопійовано і використано інформації на 64.2 зеттабайтів. Однак, лише невеликий відсоток цих новостворених даних зберігається у сховищах, оскільки всього лише два відсотки даних, які були створені та збережені в 2020 році, були використані у 2021 році. Але ця тенденція буде мінятися, оскільки прогнозується інтенсивне зростання зберігання даних у сховищах щорічно на рівні 19.2%. У 2020 році встановлений обсяг збереженої інформації досягнув 6,7 зеттабайтів [11].

Протягом наступних п'яти років до 2025 року прогнозується, що глобальне створення даних зросте до понад 180 зеттабайтів. Зростання обсягів відбувається вищими темпами, ніж очікувалося раніше, через

збільшений попит у зв'язку з пандемією COVID-19, оскільки більше людей працювало, навчалося і частіше користувалося варіантами дозвілля вдома [11].

Важко зібрати, очистити, інтегрувати та, нарешті, отримати необхідні високоякісні дані протягом адекватного періоду часу. Завдяки великому відсотку неструктурованих даних у великих обсягах інформації, перетворення неструктурованих типів на структуровані та подальша обробка даних займають багато часу. Це становить серйозні виклики для існуючих підходів обробки, особливо з огляду на збереження якості даних.

3. Зміни даних відбуваються дуже швидко, і "актуальність" даних має короткий термін, що висуває певні вимоги до технології обробки. Через швидкі зміни у Big Data, "актуальність" деяких з них є дуже короткочасною. Якщо компанії не можуть збирати й обробляти необхідні дані в реальному часі або з мінімальною затримкою, то вони можуть робити висновки на основі застарілої чи недійсної інформації. Обробка й аналіз на основі цих даних призведуть до помилок у прийнятті рішень урядами або підприємствами. На даний момент програмне забезпечення для обробки даних у реальному часі все ще знаходиться в стадії розробки або удосконаленні; дійсно ефективні комерційні продукти є малочисельними.

Однією з найбільших проблем, пов'язаних з аналізом даних в реальному часі, є якість даних. Дані в реальному часі можуть генеруватися з різних джерел, таких як датчики, соціальні медіа та веб-додатки. Однак дані, що генеруються з цих джерел, можуть бути неточними або неповними, що може призвести до помилкових висновків та неправильного прийняття рішень. Для вирішення проблеми якості даних бізнесам та організаціям необхідно забезпечити точність, повноту та актуальність аналізованих даних. Це вимагає ретельного планування та використання відповідних інструментів і методик забезпечення якості даних, таких як очищення даних, видалення дублів та інтеграція даних [12].

4. Відсутність у світі єдиних та затверджених стандартів якості даних. З метою гарантування якості інформації та підвищення користі для підприємств, в 1987 році Міжнародна організація зі стандартизації (ISO) опублікувала стандарти ISO 9000, серію стандартів які застосовуються при створенні та удосконаленні систем менеджменту якості організацій.

Стандарти ISO серії 9000 були розроблені технічним комітетом ISO/TK 176 в результаті узагальнення накопиченого національного досвіду різних країн щодо розроблення, впровадження та функціонування систем якості. Комітет керувався попередніми розробками Британського інституту стандартів, що знайшли своє відображення в Британському стандарті BS 5750. Стандарти серії ISO 9000, прийняті більш ніж 90 країнами світу як національні, застосовуються до будь-яких підприємств, незалежно від їх розміру, форм власності та сфери діяльності. На сьогоднішній день понад 100 країн і регіонів у всьому світі активно впроваджують ці стандарти. Впровадження сприяє взаєморозумінню між підприємствами у внутрішній та міжнародній торгівлі та приносить користь у вигляді усунення торгових бар'єрів.

У порівнянні з цим, вивчення стандартів якості даних почалося у 1990-х роках, але тільки у 2011 році ISO опублікувала їх як ISO 8000-1:2011 [13]. Пізніше було опубліковано нову версію ISO 8000-1:2022, яка скасовувала попередню [14].

Концепція великих даних є новою, і в академічному середовищі, так само як в бізнес середовищі, як було зазначено вище, ще не сформовано єдиної думки щодо критеріїв якості. Наукова література пропонує різні визначення якості даних, але одне можна вважати беззаперечним: якість даних залежить не тільки від їх власних характеристик, але й від бізнес-середовища, яке використовує ці дані, включаючи бізнес-процеси та користувачів. Зазвичай стандарти якості даних диктуються самими виробниками. Раніше це було обумовлено тим, що бізнес виробляв дані для

бізнесу, споживачами інформації були аналітичні агентства чи державні установи, а модель обміну інформації зводилась виключно до B2B каналів. Іншими словами, споживачі даних були або безпосередніми, або непрямими виробниками даних, що забезпечувало їх якість. Проте, в епоху великих даних, з огляду на різноманітність джерел даних, споживачі даних не обов'язково є їхніми виробниками.

Для визначення якості даних можна скористатися 5 характеристиками інформації: доступність, корисність, надійність, актуальність, презентабельність [10]. Перші чотири характеристики якості даних можна вважати невід'ємними, а остання - додатковою, яка покращує задоволення споживачів.

Доступність визначається як ступінь зручності для користувачів отримати дані та пов'язану з ними інформацію, складається з трьох елементів: сама доступність інформації, авторизація та своєчасність. Доступність визначається рівнем складності для користувачів при отриманні даних. Доступність тісно пов'язана з відкритістю даних: чим вищий ступінь відкритості даних, тим більше даних може бути отримано, і тим вище рівень доступності. Своєчасність визначається як затримка часу від генерації та отримання даних до їх використання [15]. Дані повинні бути доступні протягом цього періоду, щоб забезпечити можливість їх аналізу. В епоху великих даних зміст даних швидко змінюється, тому своєчасність має велике значення. Авторизація означає, чи має окрема особа або організація право на використання даних. Авторизація важлива для забезпечення контролю над доступом до даних, дотримання політик конфіденційності та безпеки. Особливо важливо, щоб дані використовувалися відповідно до їх призначення та правових вимог.

Корисність визначає, чи є дані корисними та відповідають потребам користувачів. Вона включає в себе визначення даних, надійність та метадані. Визначення складається зі специфікації даних, яка включає назву,

показники, діапазони допустимих значень, стандартні формати, бізнес-правила тощо. Нормативне визначення даних підвищує ступінь використання даних. Достовірність використовується для оцінки некількісних даних. Вона відноситься до об'єктивних і суб'єктивних компонентів правдивості джерела або повідомлення. Достовірність даних має три ключові фактори: надійність джерел даних, нормалізація даних та час, коли дані були вироблені. Зі збільшенням кількості джерел даних і типів даних, оскільки споживачі даних спотворюють значення загальноприйнятої термінології та концепцій даних, використання даних може нести ризики. Тому виробники даних повинні надавати метадані, які описують різні аспекти наборів даних, щоб зменшити проблеми, викликані непорозумінням або невідповідностями.

Надійність даних визначає, чи можемо ми довіряти їм. Вона складається з точності, узгодженості, повноти, відповідності та перевіреності.

Для визначення точності даних, вони порівнюється з відомим довідковим значенням. У деяких ситуаціях точність може бути легко виміряна. Однак у інших випадках відоме довідкове значення відсутнє, що ускладнює вимірювання точності. Оскільки точність до певної міри корелюється з контекстом, точність даних слід вирішувати залежно від ситуації застосування.

Узгодженість даних визначає, чи правильні та повні логічні взаємозв'язки між даними. У галузі баз даних [16] це також означає, що однакові дані, розташовані у різних сховищах, повинні бути еквівалентними.

Термін "цілісність даних" має широкий зміст і може мати різні значення залежно від конкретного контексту. У контексті баз даних [16] під цілісністю мається на увазі повна структура. Також це може значити, що дані стандартизовані відповідно до моделі та/або типу. У галузі

інформаційної безпеки цілісність даних означає підтримку та забезпечення точності та узгодженості даних протягом усього їх життєвого циклу, дані не можуть бути змінені несанкціоновано або непомітно.

Повнота означає, що значення всіх компонентів одного елементу даних є дійсними. Наприклад, для кольору зображення можна використовувати RGB для опису червоного, зеленого та синього, і через такий вектор, що складається з 3 елементів, можна передати будь-який колір. Якщо значення кольору певного компонента відсутнє, зображення не може показати реальний колір, і його повнота порушена [13].

З точки зору можливості застосування перевірки, життєвий цикл даних включає три фази: генерація даних, збір даних та використання даних [13]. Однак, тут мається на увазі, що аудитори можуть справедливо оцінити точність та цілісність даних протягом розумних меж під час фази використання даних.

Актуальність використовується для опису ступеня кореляції між змістом даних та очікуваннями або вимогами користувачів; адаптивність є його елементом якості [17].

Презентабельність характеризує метод опису даних, який дозволяє користувачам повністю розуміти дані, вимірюється більше суб’єктивно завдяки читабельності та структурі.

Отже, однією з ознак початку ери Big Data є експоненційний ріст даних у різних галузях та областях. Забезпечення якості великих даних, а також аналіз інформації, яку приховано у цих даних, стають важливими питаннями для промисловості та наукових кіл. Низька якість даних може призвести до низької ефективності їх використання і навіть спричинити серйозні помилки в прийнятті рішень. У розділі було коротко проаналізовано 4 основні виклики пов’язані із забезпеченням якості Big Data, надано визначення основним характеристикам даних.

1.3 ВАЛІДАЦІЯ ДАНИХ У ETL І ELT КОНВЕЙЄРАХ

В міру зростання обсягів даних, їх джерел і типів у організаціях, збільшується важливість використання цих даних у аналітиці, науці про дані та проектах з машинного навчання для отримання бізнес-інформації. Ця потреба збільшує відповідальність команд дата інженерів, оскільки обробка сирих, неструктурованих даних і їх конвертація у чисті, актуальні, надійні дані є важливим кроком перед подальшою роботою з даними. Тому побудова ефективних процесів конвеєрної обробки даних (Data Pipelines) — одне з найважливіших завдань для Big Data Engineers у будь-якому проєкті. Треба створити структуру процесів обробки даних і, звісно ж, реалізувати її. Багато інших завдань є частиною цього процесу та допомагають його поліпшувати з точки зору ефективності.

Серед них відомий ETL (Extract Transform Load) конвеєр — вилучення даних, проведення трансформацій над ними та завантаження в іншу систему для зберігання і подальшої роботи. Для різних видів даних використовують різні інструменти, у Big Data часто працюють зі статичними або потоковими типами даних. Для таких цілей застосовують фреймворки Apache Spark, Flink, Storm, Kafka та хмарні сервіси AWS, Google Cloud, Azure [18].

Перший крок цього процесу полягає у вилученні даних з цільових джерел, які зазвичай є різними за своєю природою, наприклад, бізнес-системи, API-сервери, маркетинговими аналітичними даними, базами даних транзакцій. Частина цих даних є структурованою, в той час, як журнали серверів є напівструкторованими даними у форматі JSON.

Існують різні способи вилучення даних. Часткове вилучення з можливістю оновлення в потоковому режимі – при появі змін – дані оновлюються автоматично. Часткове вилучення, за якого фіксуються зміни та помічаються оновлені дані. Повне вилучення – завантажуються всі дані, зміни можна помітити лише при порівнянні двох різних версій [19].

Другий крок полягає у трансформації даних, які були витягнуті з джерел, у формат, який може бути використаний. На цьому етапі дані очищаються, обробляються та трансформуються, часто до певної схеми, що відповідає операційним потребам. Цей процес передбачає декілька типів трансформацій, які забезпечують якість і цілісність даних. Дані, як правило, не завантажуються безпосередньо у цільове джерело даних, а натомість зазвичай завантажуються у проміжну базу даних. Цей крок забезпечує швидке відновлення у випадку, якщо щось піде не за планом. На цьому етапі є можливість генерувати звіти про аудит даних або діагностувати та виправляти будь-які проблеми з даними.

Третій етап – завантаження, це процес запису перетворених даних із проміжної зони до цільової бази даних, яка могла існувати раніше, а може й бути створена під конкретну задачу. Залежно від вимог до застосунку, цей процес може бути як досить простим, так і складним. Кожен з цих етапів може бути виконаний за допомогою інструментів ETL (Apache NiFi, Apache Airflow, AWS Glue) або коду.

ETL конвеєри стали невід'ємною частиною сучасної обробки даних, але зі зростанням кількості джерел та типів даних, створення та підтримка надійних конвеєрів є однією з найскладніших завдань у сфері інженерії даних. Побудова надійних конвеєрів є повільною та складною. Дані конвеєри створюються зі складним кодом і мають обмежену можливість повторного використання. Конвеєр, побудований у одному середовищі, не можна використовувати в іншому, навіть якщо базовий код дуже схожий, тому робота інженерів зі створення ETL займає певний час, коли клієнт вже потребує результату.

У таких умовах все складніше забезпечувати якість даних. Погані дані часто пропускаються через конвеєр непоміченими, знецінюючи всю інформацію. Тому валідація потрібна на кожному кроці, щоб відсіяти зайве. Нарешті, зі зростанням масштабів та складності конвеєрів, компанії

стикаються з підвищеним оперативним навантаженням при їх управлінні, що робить підтримку якості надзвичайно важкою. Невдачі конвеєрів важко ідентифікувати та ще важче вирішувати через відсутність видимості та інструментів.

Валідація даних у процесі не є одноразовою операцією, а скоріше неперервним процесом, який охоплює всі етапи ETL. Розглянемо детальніше, як інтегрується валідація даних на кожному етапі [20].

На етапі вилучення дані збираються з різних джерел. Це початковий пункт процесу ETL і перша можливість для валідації даних.

1. Перевірка повноти даних. На цьому етапі необхідно забезпечити, щоб всі необхідні дані були вилучені. Це можна зробити, порівнявши кількість записів або використовуючи контрольні суми з джерела та даних, що були отримані.

2. Перевірка точності даних. Отримані дані мають відповідати даним у джерелах, тому тут потрібна перевірка значень. Якщо якісь джерела викликають сумніви, корисно провести деякі попередні перевірки якості даних, щоб виявити потенційні проблеми на ранніх стадіях.

На етапі перетворення, де дані очищаються та перетворюються у формат, придатний для цільової системи, також важлива валідація даних.

1. Валідація правил трансформації. Під час перетворення даних важливо перевіряти, що застосовані правила та логіка дають очікувані результати.

2. Перевірка відповідності даних після трансформації. Збереження формату, структури, черги.

3. Перевірка на відсутність null. Важливо перевірити, що важливі поля (обов’язкові для заповнення) не виявились порожніми після перетворення.

Останнім етапом ETL є завантаження перетворених даних у цільову систему, зазвичай це сховище даних. Тут валідація забезпечує успішність та точність операції завантаження.

1. Перевірка повноти даних. Подібно до етапу вилучення, важливо переконатися, що всі дані були завантажені у цільову систему.

2. Перевірка цілісності даних. Важливо перевірити, що відносини між елементами даних (зовнішній ключ, наприклад) були збережені під час процесу завантаження.

3. Перевірка відповідності. Це включає порівняння даних у джерелі та цільових системах для переконання, що вони збігаються, що підтверджує успішність процесу ETL.

Розглянемо детальніше техніки, які можуть бути застосовані для вищезазначених перевірок на кожному етапі.

1. Перевірка типів даних - валідація того, що кожен елемент даних є правильного типу. Це може включати перевірку, чи числові поля містять числові дані, а поля дати містять дійсні дати тощо.

2. Перевірки діапазону валідація того, що значення даних вписуються в прийнятні межі. Наприклад, якщо поле даних повинне містити вік особи, перевірка діапазону підтвердить, що значення вписуються у правдоподібний діапазон, наприклад від 0 до 120 років.

3. Перевірки обмежень включають верифікацію того, що дані відповідають попередньо визначеним обмеженням. Це можуть бути унікальні обмеження (наприклад, кожен ідентифікатор клієнта повинен бути унікальним), обмеження первинного ключа (наприклад, кожен рядок повинен мати унікальний ідентифікатор) або обмеження зовнішнього ключа (наприклад, посилання на іншу таблицю, яке повинне існувати).

4. Перевірки відповідності використовуються для забезпечення послідовності значень даних між наборами даних. Наприклад, якщо є дві

таблиці з даними про клієнтів, імена та ідентифікатори клієнтів повинні відповідати одне одному.

5. Перевірки унікальності схожі на перевірки обмежень, але спеціально зосереджені на забезпеченні унікальності значень у певному полі, де це необхідно. Типовим прикладом буде перевірка, що кожен клієнт має унікальний ідентифікаційний номер клієнта.

6. Перевірки референтної цілісності включають валідацію збереження взаємозв'язків між таблицями. Це особливо важливо на етапі завантаження у процесі ETL, де підтримання взаємозв'язків між елементами даних (як-от зовнішні ключі) є критичним.

Окремо слід розглянути конвеєр даних, що існує за формулою ELT – вилучення, завантаження, трансформація. Протягом багатьох років сховища даних з ETL та DataLake з ELT розвивалися паралельно. Однак, останнім часом процес ELT стає поширеним у організаціях, які доповнюють свої традиційні інструменти ETL новими, більш дешевими процесами ELT для наповнення своїх схем сховищ даних — закладаючи основу для методології EtLT.

Звичайним підходом до використання гнучкості та низької вартості DataLake є заміна дорогого традиційного інструмента ETL на DataLake та програмування трансформацій всередині DataLake. Однак, щоб зменшити вплив на бізнес, Data Warehouse все ще залишається у використанні. Це призводить до комбінації патернів ETL та ELT.

Вартість хмарних сховищ даних значно знизилася, що робить підтримку окремого DataLake економічно менш обґрунтованою. До того ж деякі хмарні сховища даних, як-от Snowflake, розширюють свої можливості, наближаючись до різноманітних і гнучких методологій обробки даних, які використовуються в DataLake [21].

Таким чином, обробка перетворень всередині сховищ даних стає доступнішою, і підтримка окремого DataLake більше не потрібна. Це призводить до того, що багато підприємств консолідують свої операції з даними в найновіших поколіннях хмарних сховищ даних.

Тим часом деякі DataLake, такі як Databricks, додають структури, які зручніше використовувати й обробляти, що робить недоцільним використання окремих сховищ для фінальних результатів [21].

Враховуючи це зближення DataLake і DataWarehouse, деякі вендори пропонують гармонійне поєднання в Data Lakehouse, універсальні рішення, здатні зберігати та обробляти будь-який тип даних (DataBricks, Snowflake, Amazon S3 + Amazon Redshift, BigQuery від GCP, Azure Synapse Analytics від Microsoft).

Таким чином виникає новий підхід до побудови конвеєрів з обробки даних EtLT – вилучення, трансформація (коригування), завантаження, трансформація. Міняється глибина валідації та додається більше роботи на більш пізніх стадіях. На перших етапах відбувається перевірка джерел і цілісності завантаження даних. На пізньому етапі – трансформації - відбувається перевірка діапазонів, відповідності, тощо. Інакше кажучи, будь-який з підходів до процесу обробки даних не вимагає нових або специфічних перевірок, лише міняє їх порядок і пріоритет на кожному з етапів.

РОЗДІЛ 2 ТЕХНОЛОГІЧНІ ІНСТРУМЕНТИ ВАЛІДАЦІЇ BIGDATA

2.1 ЗАСТОСУВАННЯ APACHE SPARK У ПРОЦЕСІ ВАЛІДАЦІЇ BIG DATA

У сучасному світі, орієнтованому на дані, обробка, інтерпретація та отримання значущих висновків з величезних обсягів інформації становить складне завдання для підприємств.

Згідно зі звітом Міжнародного інституту McKinsey, всі компанії, що працюють на основі даних, мають на 23 рази більше ймовірностей залучити нових клієнтів, в шість разів більше шансів зберегти існуючих та в 19 разів більше ймовірностей отримати прибуток[22].

Використання Big Data надає численні переваги для організацій, включаючи [23]:

1. Кращі стратегічні рішення. 69% організацій зазначають, що аналіз великих даних допомагає приймати більш обґрунтовані та стратегічні рішення.

2. Покращений контроль операційних процесів. 54% організацій вважають, що використання великих даних дозволяє краще контролювати та оптимізувати операційні процеси.

3. Кращий розуміння клієнтів. 52% респондентів зазначили, що аналіз великих даних дозволяє глибше розуміти потреби та поведінку клієнтів, що допомагає в розробці більш персоналізованих продуктів та послуг.

4. Зменшення витрат. 47% організацій відзначають зниження витрат завдяки оптимізації процесів та виявленню неефективностей.

5. Крім того, ті організації, які змогли кількісно оцінити свої вигоди від аналізу великих даних, повідомили про середнє зростання доходів на 8% і скорочення витрат на 10%.

Фреймворки для великих даних - найкращий вибір для спрощення цього складного процесу організації та обробки даних. Apache Spark - універсальний фреймворк для обробки даних, який допомагає виконувати складні завдання обробки на великих наборах даних. Він також може розподіляти завдання обробки даних між кількома машинами, як самостійно, так і в поєднанні з іншими розподіленими технологіями обчислень. Це робить його фаворитом для компаній, які працюють з великими даними та машинним навчанням. Крім того, надаючи інтуїтивний API, який знімає більшість нудної праці з обробки великих даних та розподілених обчислень, він звільняє розробників від частини програмування, пов'язаного з цими діяльностями [24].

Apache Spark був розроблений в Лабораторії AMP при Університеті Каліфорнії в Берклі у 2009 році. Він має вбудовані зв'язки для мов програмування Python, R, Java, Scala. Крім того, він поставляється з багатьма бібліотеками, що допомагають у потоковій обробці, обробці графів та розробці застосунків машинного навчання.

Використовуючи Apache Spark для валідації даних, інженери можуть легко проводити такі операції, як очищення даних від некоректних значень, перевірка їх точності та повноти, профілювання для розуміння структури та якості даних, а також паралельна обробка, що значно прискорює весь процес. Крім того, Apache Spark інтегрується з іншими інструментами для роботи з великими обсягами даних, що робить його універсальним рішенням для валідації та аналізу інформації.

Можна виділити наступні напрямки застосування Apache Spark для роботи з Big Data [24]:

1. Очищення даних. Apache Spark забезпечує вбудовані функції та бібліотеки, такі як Spark SQL та Spark DataFrames, які можуть очищати та трансформувати дані, видаляючи відсутні значення, нормалізуючи формати даних та виправляючи неконсистентні дані.

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, lower, trim, when

# Ініціалізація змінних
input_path = "input_data.csv"
output_path = "cleaned_data.csv"
column_to_clean = "column_name"
age_column = "age"

# Створення Spark сесії
spark = SparkSession.builder \
    .appName("Data Cleaning with Spark") \
    .getOrCreate()

# Завантаження даних
df = spark.read.csv(input_path, header=True, inferSchema=True)

# Видалення рядків з відсутніми значеннями
df_clean = df.na.drop()

# Нормалізація формату даних (наприклад, перетворення текстових
стовпців до нижнього регістру та видалення пробілів)
df_normalized      =      df_clean.withColumn(column_to_clean,
lower(trim(col(column_to_clean))))

#   Виправлення   неконсистентних   даних   (наприклад,   заміна
некоректних значень)
df_fixed         =         df_normalized.withColumn(age_column,
when(col(age_column) < 0, None).otherwise(col(age_column)))

# Збереження очищених даних
df_fixed.write.csv(output_path, header=True)

# Виведення результату
```

```
df_fixed.show()

# Зупинка Spark сесії
spark.stop()
```

2. Перевірка даних. Apache Spark можна використовувати, щоб перевірити точність даних: діапазон, формат, перехресні перевірки, використовуючи Spark SQL та Spark DataFrames.

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, regexp_extract

# Ініціалізація змінних
input_path = "input_data.csv"
invalid_data_path = "invalid_data.csv"
valid_data_path = "valid_data.csv"
age_column = "age"
email_column = "email"

# Створення Spark сесії
spark = SparkSession.builder \
    .appName("Data Validation with Spark") \
    .getOrCreate()

# Завантаження даних
df = spark.read.csv(input_path, header=True, inferSchema=True)

# Перевірка діапазону значень (наприклад, вік повинен бути між
0 і 120)
valid_age_df = df.filter((col(age_column) >= 0) &
(col(age_column) <= 120))
invalid_age_df = df.filter((col(age_column) < 0) |
(col(age_column) > 120))

# Перевірка формату (наприклад, перевірка формату email)
```

```
email_pattern   =   r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-
Z]{2,}$'
valid_email_df                                                  =
valid_age_df.filter(regexp_extract(col(email_column),
email_pattern, 0) != "")
invalid_email_df                                                =
valid_age_df.filter(regexp_extract(col(email_column),
email_pattern, 0) == "")

# Об'єднання всіх невірних даних
invalid_df = invalid_age_df.union(invalid_email_df)

# Збереження валідних та невалідних даних
valid_email_df.write.csv(valid_data_path, header=True)
invalid_df.write.csv(invalid_data_path, header=True)

# Виведення результату
print("Valid Data:")
valid_email_df.show()

print("Invalid Data:")
invalid_df.show()

# Зупинка Spark сесії
spark.stop()
```

3. Профілювання даних. Apache Spark надає потужні можливості профілювання даних, які дозволяють зрозуміти структуру та якість даних, виявити будь-які проблеми та виправити їх перед подальшою обробкою.

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count, when, isnan, mean,
stddev

# Ініціалізація змінних
```

```
input_path = "input_data.csv"

# Створення Spark сесії
spark = SparkSession.builder \
    .appName("Data Profiling with Spark") \
    .getOrCreate()

# Завантаження даних
df = spark.read.csv(input_path, header=True, inferSchema=True)

# Профілювання даних: обчислення кількості відсутніх значень у кожному стовпці
missing_data = df.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in df.columns])

# Профілювання даних: описова статистика для числових колонок
statistics = df.describe()

# Профілювання даних: обчислення додаткових статистик (наприклад, середнє значення, стандартне відхилення)
additional_stats = df.select([mean(c).alias(f"mean_{c}") for c in df.columns if c != 'string_column']) \
    .union(df.select([stddev(c).alias(f"stddev_{c}") for c in df.columns if c != 'string_column']))

# Виведення результатів
print("Missing Data:")
missing_data.show()

print("Statistics:")
statistics.show()

print("Additional Statistics:")
additional_stats.show()
```

```
# Зупинка Spark сесії
spark.stop()
```

4. Паралельна обробка даних. Ця функція дозволяє паралельно обробляти дані, що значно прискорює процес перевірки даних у середовищі великих даних. Можна легко розподілити завдання валідації даних по кількох вузлах кластера, використовуючи можливості обробки даних в пам'яті Spark для швидкої обробки великих обсягів даних.

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, regexp_extract

# Ініціалізація змінних
input_path = "input_data.csv"
valid_data_path = "valid_data.csv"
invalid_data_path = "invalid_data.csv"
age_column = "age"
email_column = "email"
email_pattern   =   r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-
Z]{2,}$'

# Створення Spark сесії
spark = SparkSession.builder \
    .appName("Parallel Data Processing with Spark") \
    .getOrCreate()

# Завантаження даних
df = spark.read.csv(input_path, header=True, inferSchema=True)

# Паралельна обробка: перевірка діапазону значень (наприклад,
вік повинен бути між 0 і 120)
valid_age_df    =    df.filter((col(age_column)    >=    0)    &
(col(age_column) <= 120))
```

```
invalid_age_df = df.filter((col(age_column) < 0) |
(col(age_column) > 120))

# Паралельна обробка: перевірка формату (наприклад, перевірка
формату email)
valid_email_df =
valid_age_df.filter(regexp_extract(col(email_column),
email_pattern, 0) != "")
invalid_email_df =
valid_age_df.filter(regexp_extract(col(email_column),
email_pattern, 0) == "")

# Об'єднання всіх невірних даних
invalid_df = invalid_age_df.union(invalid_email_df)

# Збереження валідних та невалідних даних паралельно
valid_email_df.write.csv(valid_data_path, header=True)
invalid_df.write.csv(invalid_data_path, header=True)

# Виведення результату
print("Valid Data:")
valid_email_df.show()

print("Invalid Data:")
invalid_df.show()

# Зупинка Spark сесії
spark.stop()
```

5. Інтеграція з іншими інструментами. Apache Spark може бути інтегрований з іншими інструментами для роботи з великими даними, такими як Apache Hive, Apache HBase, Apache Cassandra, щоб створити комплексне рішення для обробки великих даних. Це дозволяє організаціям виконувати перевірку даних як частину більш широкого процесу обробки

великих даних, забезпечуючи високу якість даних перед їх використанням для аналізу та прийняття рішень.

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, regexp_extract
import happybase

# Ініціалізація змінних
input_path = "input_data.csv"
valid_data_path = "valid_data.csv"
invalid_data_path = "invalid_data.csv"
database_name = "database"
hive_table_name = "hive_table"
hbase_table_name = "hbase_table"
cassandra_keyspace = "keyspace"
cassandra_table_name = "cassandra_table"
column_family = "cf"
hbase_host = "localhost"
age_column = "age"
email_column = "email"
email_pattern   =   r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-
Z]{2,}$'

# Створення Spark сесії з підтримкою Hive та Cassandra
spark = SparkSession.builder \
    .appName("Spark Integration Example") \
    .enableHiveSupport() \
    .config("spark.cassandra.connection.host", "localhost") \
    .getOrCreate()

# Створення бази даних у Hive (якщо не існує)
spark.sql(f"CREATE DATABASE IF NOT EXISTS {database_name}")

# Завантаження даних
```

```
df = spark.read.csv(input_path, header=True, inferSchema=True)

# Паралельна обробка: перевірка діапазону значень
valid_age_df    =    df.filter((col(age_column)    >=    0)    &
(col(age_column) <= 120))
invalid_age_df    =    df.filter((col(age_column)    <    0)    |
(col(age_column) > 120))

# Паралельна обробка: перевірка формату email
valid_email_df                                                  =
valid_age_df.filter(regexp_extract(col(email_column),
email_pattern, 0) != "")
invalid_email_df                                                =
valid_age_df.filter(regexp_extract(col(email_column),
email_pattern, 0) == "")

# Об'єднання всіх невірних даних
invalid_df = invalid_age_df.union(invalid_email_df)

# Збереження валідних та невалідних даних
valid_email_df.write.csv(valid_data_path, header=True)
invalid_df.write.csv(invalid_data_path, header=True)

# Збереження даних у Hive
valid_email_df.write.mode("overwrite").saveAsTable(f"{database
_name}.{hive_table_name}")

# Запис даних до HBase
connection = happybase.Connection(hbase_host)
table = connection.table(hbase_table_name)

for row in valid_email_df.collect():
    table.put(str(row['id']), {
        f"{column_family}:name": row['name'],
```

```
        f"{column_family}:age": str(row['age'])
    })

# Читання даних з HBase через Spark
conf = {
    "hbase.zookeeper.quorum": hbase_host,
    "hbase.mapreduce.inputtable": hbase_table_name
}
hbase_df = spark.read \
    .format("org.apache.hadoop.hbase.spark") \
    .options(**conf) \
    .load()

# Збереження даних у Cassandra
valid_email_df.write \
    .format("org.apache.spark.sql.cassandra") \
    .mode("overwrite") \
    .options(table=cassandra_table_name,
keyspace=cassandra_keyspace) \
    .save()

# Читання даних з Cassandra через Spark
cassandra_df = spark.read \
    .format("org.apache.spark.sql.cassandra") \
    .options(table=cassandra_table_name,
keyspace=cassandra_keyspace) \
    .load()

# Виконання запиту до даних у Hive
hive_result = spark.sql(f"SELECT * FROM
{database_name}.{hive_table_name} WHERE age > 30")

# Виведення результату
print("Valid Data:")
```

```
valid_email_df.show()

print("Invalid Data:")
invalid_df.show()

print("Data from Hive:")
hive_result.show()

print("Data from HBase:")
hbase_df.show()

print("Data from Cassandra:")
cassandra_df.show()

# Зупинка Spark сесії
spark.stop()
```

Використовуючи Apache Spark для валідації даних, організації можуть забезпечити високу якість своїх даних, що дозволяє приймати обґрунтовані рішення, покращувати досвід споживачів та стимулювати інновації. Інтеграція Spark з іншими інструментами для роботи з великими даними забезпечує комплексне рішення для обробки та аналізу даних, що є ключовим елементом стратегії цифрової трансформації сучасних організацій.

Apache Spark є потужною системою для обробки великих обсягів даних, яка підтримує широкий спектр операцій для валідації та аналізу даних. Використання методів та інструментів Spark дозволяє забезпечити точність, повноту та узгодженість даних, що є критично важливим для успішної роботи з великими даними.

2.2 ОСОБЛИВОСТІ ВАЛІДАЦІЇ BIG DATA НА ПЛАТФОРМІ DATABRICKS

Одним із недоліків Apache Spark є складність його інсталяції та налаштування. Для коректного функціонування Spark потребує ретельного налаштування кластерів, встановлення необхідних бібліотек та забезпечення сумісності з іншими компонентами інфраструктури. Це може бути трудомістким процесом, який вимагає значних технічних знань і часу. Такі труднощі можуть стати перепоною для швидкого старту проєктів з аналізу даних.

Databricks усуває цю проблему, пропонуючи повністю кероване середовище для роботи зі Spark [25]. Це дозволяє командам зосередитися на аналізі даних і розробці рішень, замість витрачання часу на налаштування та обслуговування інфраструктури.

Databricks – це інтегрована платформа для аналізу та обробки великих обсягів даних, яка забезпечує повну підтримку Apache Spark [25]. Важливою перевагою Databricks є те, що він розроблений творцями Apache Spark, що гарантує оптимальну інтеграцію та ефективність використання цієї потужної технології. Однією з ключових особливостей Databricks є те, що для його використання не потрібно встановлювати або налаштовувати Apache Spark окремо. Databricks пропонує готове до використання середовище, що значно спрощує процес розробки, тестування та впровадження рішень для роботи з великими даними.

Databricks як компанія розробляє та продає хмарну платформу для роботи з даними, використовуючи маркетинговий термін "lakehouse" [21], який об'єднує поняття "data warehouse" (склад даних) і "data lake" (озеро даних). Lakehouse Databricks базується на відкритій платформі Apache Spark, яка дозволяє виконувати аналітичні запити до напівструктурованих даних без традиційної схеми бази даних [21]. У жовтні 2022 року Lakehouse

отримала статус FedRAMP, що дозволяє її використовувати урядовим установам США та їх підрядникам [24].

Databricks надає декілька важливих інструментів для валідації та забезпечення якості даних [24]:

Delta Lake – це розширення Apache Spark, яке додає підтримку ACID-транзакцій та можливості для управління даними на рівні таблиць. Delta Lake забезпечує високу консистентність даних та дозволяє здійснювати перевірку якості даних за допомогою обмежень та перевірок. У червні 2020 року Databricks запустила Delta Engine, новий движок запитів, який накладається на Delta Lake для підвищення продуктивності запитів. Delta Engine сумісний з Apache Spark та MLflow, які також є відкритими проектами, створеними співробітниками Databricks.

```
from pyspark.sql import SparkSession
from delta.tables import DeltaTable

# Ініціалізація змінних
input_path = "input_data.csv"
delta_table_path = "delta_table"
age_column = "age"
email_column = "email"
email_pattern  =  r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'

# Створення Spark сесії з підтримкою Delta Lake
spark = SparkSession.builder \
    .appName("Delta Lake Example") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()
```

```
    # Завантаження даних
    df = spark.read.csv(input_path, header=True,
inferSchema=True)

    # Очищення даних: видалення рядків з відсутніми значеннями
    df_clean = df.na.drop()

    # Нормалізація формату даних (наприклад, перетворення
текстових стовпців до нижнього регістру та видалення пробілів)
    df_normalized = df_clean.withColumn(email_column,
lower(trim(col(email_column))))

    # Виправлення неконсистентних даних (наприклад, заміна
некоректних значень)
    df_fixed = df_normalized.withColumn(age_column,
when(col(age_column) < 0, None).otherwise(col(age_column)))

    # Збереження очищених даних у Delta таблицю
    df_fixed.write.format("delta").mode("overwrite").save(del
ta_table_path)

    # Завантаження Delta таблиці
    delta_table = DeltaTable.forPath(spark, delta_table_path)

    # Перевірка якості даних за допомогою обмежень
    delta_table.update(
        condition=col(email_column).rlike(email_pattern) ==
False,
        set={email_column: None}
    )

    # Виведення результату
    delta_table.toDF().show()
```

```
# Виконання запиту до даних у Delta таблиці
result = spark.sql(f"SELECT * FROM delta.`{delta_table_path}` WHERE age > 30")

# Виведення результату
result.show()

# Зупинка Spark сесії
spark.stop()
```

Delta Live Tables – цей інструмент дозволяє легко створювати та керувати надійними конвеєрами даних, які доставляють високоякісні дані в Delta Lake. Delta Live Tables пропонує вбудовані функції для обробки очікувань і моніторингу якості даних, що допомагає запобігати потраплянню неякісних даних у конвеєри.

```
import dlt
from pyspark.sql.functions import col, lower, trim

# Завантаження сирих даних
@dlt.table(name="raw_data")
def raw_data():
    return (
        spark.read.json("raw_data.json")
    )

# Очищення даних
@dlt.table(name="cleaned_data")
def cleaned_data():
    return (
        dlt.read("raw_data")
        .filter(col("age").isNotNull() & (col("age") >= 0))
        .withColumn("email", lower(trim(col("email"))))
    )
```

```
# Трансформація даних
@dlt.table(name="transformed_data")
def transformed_data():
    return (
        dlt.read("cleaned_data")
        .withColumn("age", col("age").cast("int"))
    )

# Збереження даних у Delta Lake
@dlt.table(name="delta_data")
def delta_data():
    return dlt.read("transformed_data")
```

Databricks SQL – введений у листопаді 2020 року для виконання завдань бізнес-аналітики та звітності. Аналітики можуть виконувати запити до наборів даних за допомогою стандартного SQL або використовувати конектори продукту для інтеграції з інструментами бізнес-аналітики, такими як Holistics, Tableau, Qlik, SigmaComputing, Looker і ThoughtSpot.

```
-- Використання бази даних
USE analytics_db;

-- Запит для отримання даних з таблиці
SELECT * FROM delta_table;

-- Запит для отримання кількості користувачів старше 30 років
SELECT COUNT(*) FROM delta_table WHERE age > 30;

-- Запит для обчислення середнього віку користувачів
SELECT AVG(age) AS average_age FROM delta_table;

-- Запит для групування даних за віком
SELECT age, COUNT(*) AS user_count
FROM delta_table
```

```
GROUP BY age
ORDER BY user_count DESC;
```

Lakehouse Monitoring – інтегрована сервісна платформа для моніторингу якості даних та AI-активів. Lakehouse Monitoring надає готові метрики якості даних та автоматично створювану панель приладів для візуалізації цих метрик. Користувачі можуть визначати користувацькі метрики та отримувати сповіщення про проблеми з якістю даних.

```
from pyspark.sql import SparkSession
from delta.tables import DeltaTable
from pyspark.sql.functions import col, count, when, isnan

# Ініціалізація змінних
input_path = "input_data.csv"
delta_table_path = "delta_table"
email_column = "email"
age_column = "age"
email_pattern   =   r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-
Z]{2,}$'

# Створення Spark сесії з підтримкою Delta Lake
spark = SparkSession.builder \
    .appName("Lakehouse Monitoring Example") \
    .config("spark.sql.extensions",
"io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

# Завантаження даних
df = spark.read.csv(input_path, header=True, inferSchema=True)

# Збереження даних у Delta таблицю
```

```
df.write.format("delta").mode("overwrite").save(delta_table_pa
th)

# Завантаження Delta таблиці
delta_table = DeltaTable.forPath(spark, delta_table_path)

# Перевірка якості даних: кількість відсутніх значень
missing_values                                                  =
delta_table.toDF().select([count(when(col(c).isNull()          |
isnan(c), c)).alias(c) for c in df.columns])

# Перевірка якості даних: некоректні email адреси
invalid_emails                                                  =
delta_table.toDF().filter(~col(email_column).rlike(email_patte
rn)).count()

# Перевірка якості даних: діапазон значень для віку
invalid_ages = delta_table.toDF().filter((col(age_column) < 0)
| (col(age_column) > 120)).count()

# Виведення результатів перевірки якості даних
print("Missing Values:")
missing_values.show()

print(f"Invalid Emails: {invalid_emails}")
print(f"Invalid Ages: {invalid_ages}")

# Зупинка Spark сесії
spark.stop()
```

Архітектура медалей (Medallion Architecture) [26] є шаблоном дизайну даних, який використовується для логічної організації даних у lakehouse з метою поступового і прогресивного покращення структури та якості даних при їх переміщенні через кожен шар архітектури (від Bronze

⇒ Silver ⇒ Gold таблиць). Архітектуру медалей інколи також називають "multi-hop" архітектурою.

Databricks надає інструменти, такі як Delta Live Tables (DLT), які дозволяють користувачам миттєво створювати конвеєри даних з таблицями Bronze, Silver і Gold [26] за допомогою всього кількох рядків коду. Крім того, за допомогою потокових таблиць і матеріалізованих подань, користувачі можуть створювати потокові DLT конвеєри на основі Apache Spark Structured Streaming, які оновлюються та оновлюються інкрементно. Детальніше про це можна дізнатися в документації Databricks щодо комбінування потокових таблиць і матеріалізованих подань в одному конвеєрі [25].

```
import dlt
from pyspark.sql.functions import col, lower, trim

# Завантаження сирих даних у Bronze таблицю
@dlt.table(name="bronze_table")
def bronze_table():
    return (
        spark.read.json("path/to/raw_data.json")
    )

# Очищення та нормалізація даних у Silver таблиці
@dlt.table(name="silver_table")
@dlt.expect("valid_age", "age IS NOT NULL AND age >= 0")
@dlt.expect("valid_email", "email RLIKE '^[a-zA-Z0-9._%+-
]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'")
def silver_table():
    return (
        dlt.read("bronze_table")
        .withColumn("email", lower(trim(col("email"))))
        .withColumn("age", col("age").cast("int"))
    )
```

```
    # Агрегація та підготовка даних для аналітики у Gold таблиці
    @dlt.table(name="gold_table")
    def gold_table():
        return (
            dlt.read("silver_table")
            .groupBy("age")
            .count()
        )
```

Bronze Layer - це шар, де ми зберігаємо всі дані з зовнішніх систем. Структури таблиць у цьому шарі відповідають структурам таблиць системи-джерела "як є", разом з будь-якими додатковими стовпцями метаданих, які фіксують дату/час завантаження, ідентифікатор процесу тощо. Основна мета цього шару - швидке захоплення змін даних (Change Data Capture) і можливість надання історичного архіву джерела (cold storage), лініїї даних, можливість аудиту, повторної обробки у разі потреби без повторного читання даних з системи-джерела.

На Silver Layer lakehouse, дані з Bronze Layer зіставляються, зливаються, уніфікуються та очищуються ("в достатній мірі"), щоб забезпечити "підприємницький погляд" на всі ключові бізнес-об'єкти, концепції та транзакції (наприклад, головні клієнти, магазини, не дубльовані транзакції та таблиці перехресних посилань).

Silver Layer об'єднує дані з різних джерел в єдиний підприємницький погляд і дозволяє самообслуговування для аналітики для ad-hoc звітності, розширеної аналітики та ML. Він служить джерелом для відділів аналітиків, інженерів даних та науковців даних для створення проектів та аналізів для вирішення бізнес-проблем через підприємницькі та відділові проекти даних у Gold Layer.

У парадигмі інженерії даних lakehouse зазвичай використовується методологія ELT, а не ETL [26], що означає, що під час завантаження Silver Layer застосовуються лише мінімальні або "достатні" трансформації та правила очищення даних. Пріоритетом є швидкість і гнучкість для завантаження та доставки даних у data lake, а багато проектних специфічних складних трансформацій і бізнес-правил застосовуються під час завантаження даних з Silver до Gold Layer. З точки зору моделювання даних, Silver Layer має більше моделей даних схожих на 3-тю нормальну форму. У цьому шарі можна використовувати Data Vault-подібні, продуктивні моделі даних.

Дані в Gold Layer lakehouse зазвичай організовані у бази даних, готові до споживання, специфічні для проекту. Gold Layer призначений для звітності і використовує більш денормалізовані та оптимізовані для читання моделі даних з меншим числом з'єднань. Тут застосовуються остаточні трансформації даних і правила забезпечення якості даних. Останній шар перетворення даних включає проекти, такі як аналітика клієнтів, аналітика якості продуктів, аналітика запасів, сегментація клієнтів, рекомендації продуктів, маркетинг/аналітика продажів тощо.

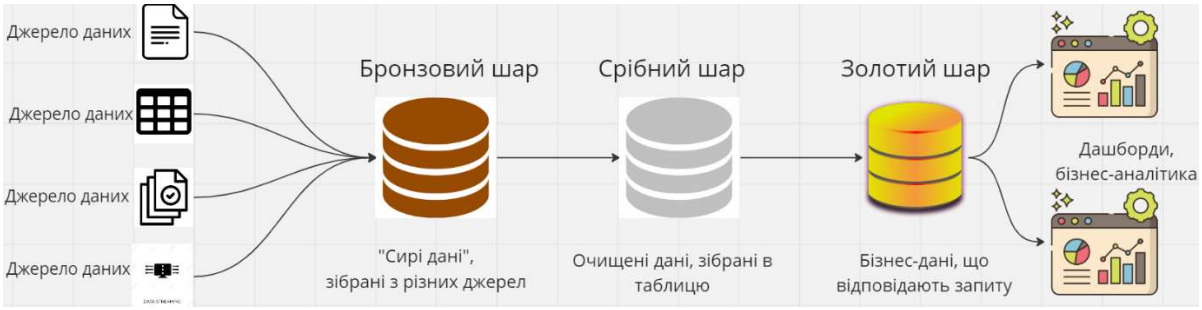


Рис. 1 Медальна архітектура даних

2.3 РОЛЬ МАШИННОГО НАВЧАННЯ В УПРАВЛІННІ ЯКІСТЮ ДАНИХ

Традиційні підходи до управління якістю даних, такі як ручне очищення даних і валідація на основі правил, часто стикаються з проблемами, такими як погана інтеграція даних із розрізнених джерел, неефективність через ручні процеси очищення даних та нездатність масштабуватися зі збільшенням обсягу даних.

Ці методи не мають прогнозних можливостей, які мають штучний інтелект і машинне навчання, які можуть автоматизувати очищення даних, виявляти аномалії в реальному часі та адаптуватися до нових патернів даних. Загалом PricewaterhouseCoopers прогнозує штучному інтелекту та машинному навчанню велике майбутнє, обіцяючи, що потенційний внесок у глобальну економіку до 2030 року від штучного інтелекту становитиме 15.7 $ трлн, а зростання ВВП країн становитиме 26% завдяки йому. Найбільші економічні вигоди від штучного інтелекту будуть в Китаї (26% приріст ВВП у 2030 році) та Північній Америці (14,5% приріст), що еквівалентно загалом $10,7 трлн і становить майже 70% глобального економічного впливу [27].

Машинне навчання для управління якістю даних є ключовим елементом високоякісного управління даними. Алгоритми машинного навчання можуть автоматично виявляти та виправляти помилки в даних, такі як помилки у написанні та некоректні записи, що зменшує потребу у людському втручанні. Просунуті моделі, такі як нейронні мережі, можуть навчитися розпізнавати складні патерни та аномалії в даних. Обробка природної мови (NLP) часто використовується для стандартизації та очищення текстових даних [28].

Алгоритми якості даних на основі машинного навчання можуть ефективно переглядати великі набори даних, щоб виявляти дублікати

записів, навіть коли ці дублікати не є точними збігами (наприклад, варіації у написанні імен чи адрес). Після виявлення дублікатів машинне навчання може допомогти об’єднати записи або запропонувати найточнішу версію дубльованих даних на основі вивчених патернів. Дедуплікація сприяє підтримці цілісності даних.

Виявлення аномалій є ключовим елементом управління якістю даних. Моделі машинного навчання, особливо алгоритми навчання без учителя, такі як кластеризація або one-class SVM, добре виявляють відхилення або аномалії, включаючи помилки та шахрайство [29]. Машинне навчання дозволяє здійснювати моніторинг потоків даних у реальному часі, щоб оперативно виявляти та попереджати про аномалії.

Машинне навчання може аналізувати історичні дані для виявлення патернів і тенденцій, прогнозування майбутніх результатів і виявлення взаємозв'язків, які раніше були непомітними. Використання машинного навчання для управління якістю даних також допомагає у створенні ознак. Алгоритми можуть допомогти визначити найрелевантніші ознаки або створити нові, які покращують прогнозні можливості моделі та інсайти, отримані з даних.

Моделі можуть постійно навчатися та адаптуватися до нових даних, покращуючи продуктивність та забезпечуючи еволюцію процесів управління якістю даних разом зі зміною даних. Інсайти, отримані за допомогою аналізу даних, можуть навіть пропонувати стратегічні рішення. Організації можуть пріоритизувати ініціативи щодо якості даних на основі даних, які мають значний вплив на бізнес-результати.

У 2024 році важко уявити ефективне управління якістю даних без практик, заснованих на штучному інтеленкті чи машинному навчанні. Ось чому варто використовувати управління якістю даних і машинне навчання[30]:

1. Покращена точність та повнота даних. Алгоритми машинного навчання можуть ефективніше виявляти аномалії, непослідовності та помилки у даних, ніж традиційні методи. Ці системи також можуть автоматично виправляти поширені помилки, навчаючись на прикладах, що призводить до підвищення точності даних.

```
import pandas as pd
from sklearn.ensemble import IsolationForest
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from pyspark.sql import SparkSession

# Ініціалізація змінних
input_path = "input_data.csv"
output_path = "output_data.csv"

# Створення Spark сесії
spark = SparkSession.builder \
    .appName("Data Cleaning with ML") \
    .getOrCreate()

# Завантаження даних
df = spark.read.csv(input_path, header=True,
inferSchema=True).toPandas()

# Обробка відсутніх значень
imputer = SimpleImputer(strategy='mean')
df_imputed = pd.DataFrame(imputer.fit_transform(df),
columns=df.columns)

# Масштабування даних
scaler = StandardScaler()
```

```
df_scaled = pd.DataFrame(scaler.fit_transform(df_imputed),
columns=df.columns)

# Використання Isolation Forest для виявлення аномалій
iso_forest = IsolationForest(contamination=0.05)
df['anomaly'] = iso_forest.fit_predict(df_scaled)

# Відфільтровування аномалій
df_clean = df[df['anomaly'] == 1].drop('anomaly', axis=1)

# Збереження очищених даних
df_clean.to_csv(output_path, index=False)

# Виведення результату
print("Cleaned Data:")
print(df_clean.head())

# Зупинка Spark сесії
spark.stop()
```

2. Збільшена ефективність та економія коштів у управлінні даними. Машинне навчання і штучний інтелект автоматизують рутинні завдання з управління якістю даних, такі як очищення та валідація. Це зменшує потребу у ручному обробленні даних, що є трудомістким і схильним до помилок. Крім того, системи штучного інтелекту та машинного навчання можуть обробляти великі обсяги даних у масштабі, допомагаючи організаціям задовольнити їхні зростаючі потреби у даних без збільшення витрат або зусиль.

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
```

```
from sklearn.preprocessing import StandardScaler
from pyspark.sql import SparkSession

# Ініціалізація змінних
input_path = "input_data.csv"
output_path = "cleaned_data.csv"

# Створення Spark сесії
spark = SparkSession.builder \
    .appName("Data Management with ML") \
    .getOrCreate()

# Завантаження даних
df = spark.read.csv(input_path, header=True,
inferSchema=True).toPandas()

# Визначення цільової змінної та ознак
target = 'label'
features = df.drop(columns=[target]).columns

# Розділення даних на тренувальний і тестовий набори
X_train, X_test, y_train, y_test =
train_test_split(df[features], df[target], test_size=0.2,
random_state=42)

# Побудова конвеєра для обробки даних і навчання моделі
pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler()),
    ('classifier', RandomForestClassifier(n_estimators=100,
random_state=42))
])

# Навчання моделі
```

```
pipeline.fit(X_train, y_train)

# Передбачення та очищення даних
predictions = pipeline.predict(X_test)
X_test['predictions'] = predictions

# Збереження очищених даних
X_test.to_csv(output_path, index=False)

# Виведення результату
print("Cleaned Data:")
print(X_test.head())

# Зупинка Spark сесії
spark.stop()
```

3. Покращення можливостей прийняття рішень на основі надійних даних. Завдяки більш точним і повним даним, організації можуть отримувати цінні інсайти, які раніше були приховані через низьку якість даних. Це призводить до кращого, більш обґрунтованого прийняття рішень на всіх рівнях організації.

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from pyspark.sql import SparkSession

# Ініціалізація змінних
input_path = "input_data.csv"
output_path = "insights_report.csv"
```

```
# Створення Spark сесії
spark = SparkSession.builder \
    .appName("Data Insights with ML") \
    .getOrCreate()

# Завантаження даних
df = spark.read.csv(input_path, header=True,
inferSchema=True).toPandas()

# Визначення цільової змінної та ознак
target = 'label'  # Замініть 'label' на вашу цільову змінну
features = df.drop(columns=[target]).columns

# Розділення даних на тренувальний і тестовий набори
X_train, X_test, y_train, y_test =
train_test_split(df[features], df[target], test_size=0.2,
random_state=42)

# Побудова конвеєра для обробки даних і навчання моделі
pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler()),
    ('classifier', RandomForestClassifier(n_estimators=100,
random_state=42))
])

# Навчання моделі
pipeline.fit(X_train, y_train)

# Передбачення на тестовому наборі
predictions = pipeline.predict(X_test)

# Отримання інсайтів: оцінка якості моделі
```

```
report = classification_report(y_test, predictions, output_dict=True)
report_df = pd.DataFrame(report).transpose()

# Збереження інсайтів у файл
report_df.to_csv(output_path)

# Виведення інсайтів
print("Classification Report:")
print(report_df)

# Зупинка Spark сесії
spark.stop()
```

4. Зменшення ризиків, пов'язаних з низькою якістю даних. Покращена якість даних за допомогою машинного навчання допомагає забезпечити відповідність нормативним вимогам, таким як GDPR, HIPAA тощо, шляхом забезпечення точної обробки та захисту персональних даних. Це зменшує ризик юридичних штрафів та шкоди репутації. Виявляючи неточності та непослідовності у даних, машинне навчання та штучний інтелект допомагають зменшити ризики, пов'язані з прийняттям рішень на основі низькоякісних даних, таких як фінансові втрати, стратегічні помилки та операційна неефективність.

```
import pandas as pd
from sklearn.ensemble import IsolationForest
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from pyspark.sql import SparkSession

# Ініціалізація змінних
input_path = "input_data.csv"
cleaned_data_path = "cleaned_data.csv"
```

```
# Створення Spark сесії
spark = SparkSession.builder \
    .appName("Data Quality with ML") \
    .getOrCreate()

# Завантаження даних
df = spark.read.csv(input_path, header=True,
inferSchema=True).toPandas()

# Визначення конвеєра для обробки даних
pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler()),
    ('anomaly_detector', IsolationForest(contamination=0.05,
random_state=42))
])

# Підготовка даних
df_features = df.drop(columns=['id']) # Замініть 'id' на
стовпчик, який не потрібен для обробки

# Обробка даних та виявлення аномалій
pipeline.fit(df_features)
anomalies =
pipeline.named_steps['anomaly_detector'].predict(df_features)
df['anomaly'] = anomalies

# Відфільтровування аномалій
df_cleaned = df[df['anomaly'] == 1].drop('anomaly', axis=1)

# Збереження очищених даних
df_cleaned.to_csv(cleaned_data_path, index=False)
```

```
# Виведення результату
print("Cleaned Data:")
print(df_cleaned.head())

# Зупинка Spark сесії
spark.stop()
```

Штучний інтелект і машинне навчання відіграють ключову роль в управлінні якістю даних, забезпечуючи більш точні, ефективні та масштабовані рішення у порівнянні з традиційними методами. Використання передових алгоритмів машинного навчання дозволяє організаціям автоматизувати процеси очищення даних, виявляти аномалії в реальному часі та адаптуватися до нових патернів даних, що суттєво покращує якість даних та забезпечує більш обґрунтовані рішення на всіх рівнях бізнесу.

РОЗДІЛ: 3 ЗАСТОСУВАННЯ PYSPARK ДЛЯ ВАЛДІАЦІЇ ТА ПОКРАЩЕННЯ ЯКОСТІ ДАНИХ У DATA LAKEHOUSE АРХІТЕКТУРІ

3.1 ЗАСТОСУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВАЛІДАЦІЇ ТА ПОКРАЩЕННЯ ЯКОСТІ ДАНИХ

У цьому розділі розглядається застосування методів машинного навчання для валідації та покращення якості даних у Data Lakehouse архітектурі. Основна мета полягає у перевірці ефективності використання методів класифікації та кластеризації для ідентифікації та корекції пошкоджених даних, а саме оцінці обсягів даних, які будуть валідуватися та коректуватися. Основна мета – зменшення обсягів даних, які підлягають перевірці при збереженні якості. Описані процеси включають генерацію, збереження, оптимізацію, валідацію, корекцію та моніторинг даних. Особлива увага приділяється вибору моделей для класифікації та кластеризації, а також їх впливу на якість даних. Код створено на платформі DataBricks Community [32].

Для проведення експериментів необхідно було створити великий набір даних, який включає як коректні, так і пошкоджені дані. Використовуючи бібліотеку Faker, було згенеровано випадкові значення для таких полів, як email, age та інші. Генерація даних здійснювалась двома функціями: generate_clean_data та generate_corrupted_data. Перша функція генерує коректні дані з нормальним розподілом значень value, середнім значенням 50 та стандартним відхиленням 10. Друга функція створює пошкоджені дані з більшою дисперсією значень value (стандартне відхилення 100), а також містить неправильно форматовані email та некоректні вікові значення. В результаті було створено 8 наборів коректних даних та 2 набори пошкоджених даних, кожен з яких містить по 100000 записів.

Для підвищення продуктивності обробки дані було збережено у форматі Delta Lake. Delta Lake забезпечує ACID-транзакції, масштабоване зберігання та високу продуктивність запитів [21]. Кожен згенерований набір даних було збережено як окрему Delta таблицю. Потім ці таблиці оптимізувалися за допомогою команди OPTIMIZE ZORDER BY, що покращує продуктивність доступу до даних за рахунок упорядкування їх на диску за визначеними стовпцями [21]. Оптимізація таблиць забезпечує швидкий доступ до великих обсягів даних і зменшує час виконання запитів, що є критичним для обробки великих даних.

Data Lakehouse архітектура часто використовує концепцію багаторівневого зберігання даних, що включає бронзові, срібні та золоті слої [26]. Кожен з цих слоїв має своє призначення та забезпечує поступове покращення якості даних на кожному етапі обробки.

На наступному етапі було розроблено модель машинного навчання для класифікації даних на коректні та пошкоджені. Вибір моделі припав на алгоритм RandomForestClassifier. RandomForest є потужним інструментом для класифікації, оскільки він поєднує велику кількість дерев рішень, що дозволяє отримати високу точність і стійкість до перенавчання [29]. Модель тренувалася на згенерованих даних, де було додано стовпець status, що вказує, чи є дані коректними або пошкодженими. Для трансформації даних перед навчанням використовувався VectorAssembler, який об'єднував значення value та age у вектор ознак. Після тренування модель оцінювалася за допомогою метрик точності, повноти та F1-міри. Всі метрики та модель логувалися за допомогою MLflow для подальшого аналізу. Використання MLflow дозволяє зберігати всі експерименти, що робить процес аналізу та відтворення результатів простішим і надійнішим [30].

Процес валідації даних включав два етапи: повну валідацію всіх даних та валідацію лише тих даних, які були визначені моделлю як пошкоджені. Під час повної валідації перевірялися всі записи на наявність

пошкоджених значень. Для цього додавався стовпець validation_status, який вказував, чи є дані валідними чи ні. Потім підраховувалася загальна кількість валідних та пошкоджених даних. Це дозволяє отримати уявлення про загальний стан якості даних у системі. Під час валідації пошкоджених даних, визначених моделлю, аналізувалися лише ті записи, які були класифіковані як пошкоджені моделлю RandomForestClassifier. Це дає можливість оцінити ефективність моделі у визначенні пошкоджених даних і порівняти результати з повною валідацією.

Для цього використовувалися моделі машинного навчання, такі як лінійна регресія, для прогнозування та виправлення значень у пошкоджених записах. Наприклад, неправильні значення email та віку могли бути виправлені за допомогою відповідних функцій. Після корекції дані зберігалися у форматі Delta для подальшого аналізу. Це дозволяє не лише покращити якість даних, але й забезпечити їх надійне зберігання та швидкий доступ до виправлених записів. Корекція даних також включала виправлення значень, які виходять за допустимі межі, такі як надто малі або надто великі значення віку, а також виправлення форматування email адрес.

Усі результати валідації та корекції даних логувалися та зберігалися з використанням MLflow. Це дозволяє відстежувати ефективність застосування методів машинного навчання для валідації даних та забезпечити їх високу якість.

Крім того, було застосовано методи кластеризації, такі як DBSCAN, для виявлення аномалій у даних. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) використовується для виявлення кластерів у даних та ідентифікації викидів, які можуть бути потенційно пошкодженими записами. Це дозволяє додатково перевіряти дані на предмет аномалій, що не були виявлені під час класифікації [30]. Результати кластеризації також логувалися для подальшого аналізу.

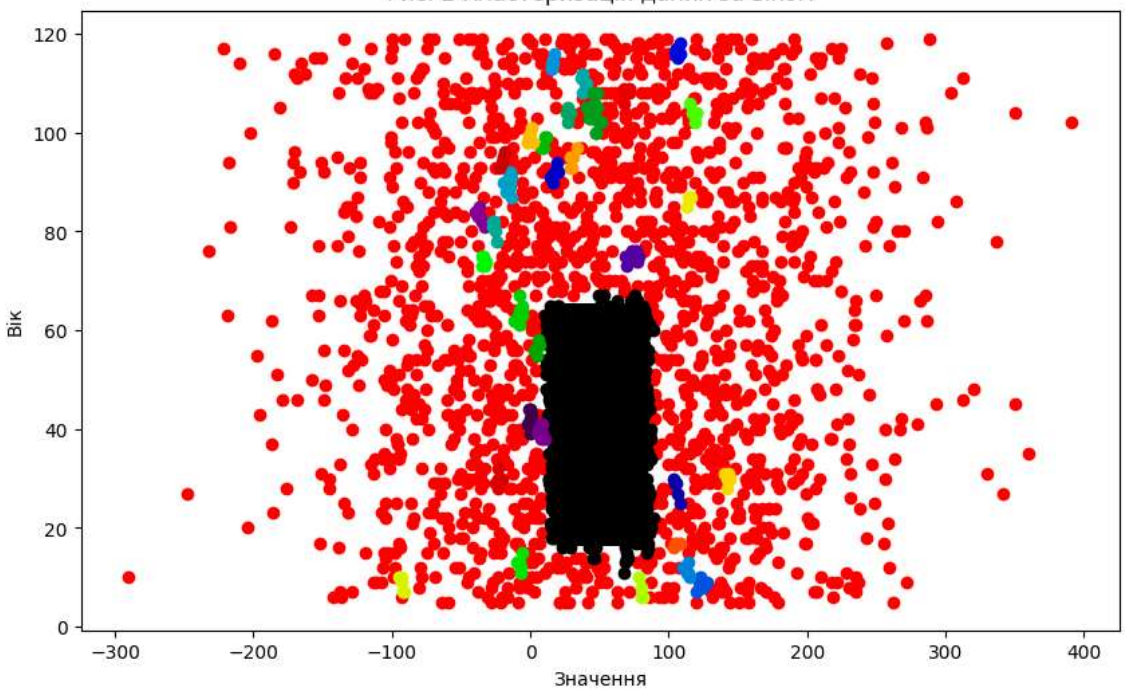
Вибір моделей для класифікації та кластеризації є критично важливим для успішного виконання поставлених завдань. Для класифікації було обрано RandomForestClassifier з кількох причин. По-перше, RandomForest є ансамблевим методом, що комбінує безліч дерев рішень, що дозволяє отримати високу точність класифікації та зменшити ризик перевиучування. По-друге, цей алгоритм є стійким до шуму в даних, що особливо важливо при роботі з великими наборами даних, що містять пошкоджені записи. Нарешті, RandomForest дозволяє отримувати інтерпретовані результати, такі як важливість ознак, що допомагає у подальшому аналізі даних.

Для кластеризації було обрано алгоритм DBSCAN. DBSCAN є методом кластеризації, що базується на щільності, і він ефективно виявляє кластери різних форм та розмірів, а також викиди. Однією з головних переваг DBSCAN є його здатність ідентифікувати викиди (аномалії), що є критично важливим для завдань валідації даних. На відміну від інших алгоритмів кластеризації, таких як K-means, DBSCAN не вимагає заздалегідь визначати кількість кластерів, що робить його більш гнучким для застосування в умовах невизначеності.

3.2 АНАЛІЗ РЕЗУЛЬТАТІВ ВАЛІДАЦІЇ ТА КОРЕКЦІЇ ДАНИХ

Аналіз результатів валідації та корекції даних у Data Lakehouse архітектурі показує значний вплив машинного навчання на скорочення обсягів обробки даних та підвищення їх якості. Результати вказують на те, що використання методів класифікації та кластеризації, таких як RandomForestClassifier та DBSCAN, дозволяє ефективно ідентифікувати пошкоджені дані та забезпечити їх корекцію [31].

Рис. 2 Кластеризація даних за віком



Час валідації всіх даних склав всього 0.07 секунд для обсягу даних у 100000 рядків. Це свідчить про високу продуктивність системи, що базується на Delta Lake та методах машинного навчання. За цей час було валідовано всі 100000 рядків даних, ідеРи3нтифіковано 6532 пошкоджених рядків. Це становить приблизно 6.53% від загального обсягу даних. Стає очевидним, що для такого відсотка пошкоджень валідувати всі дані є неефективним.

Валідація пошкоджених даних, визначених моделлю, зайняла трохи більше часу – 0.12 секунд, але обсяг даних, які потребували перевірки, скоротився до 6345 рядка, з яких усі віявились пошкодженими та відбулась корекція. Це значно менше, ніж загальний обсяг даних, що вказує на ефективність попереднього етапу класифікації. Таким чином, використання машинного навчання дозволило зменшити обсяг даних, що потребували детальної перевірки, майже на 90%, що суттєво знизило витрати часу та обчислювальних ресурсів.

Рис. 3 Відсоток пошкоджених даних

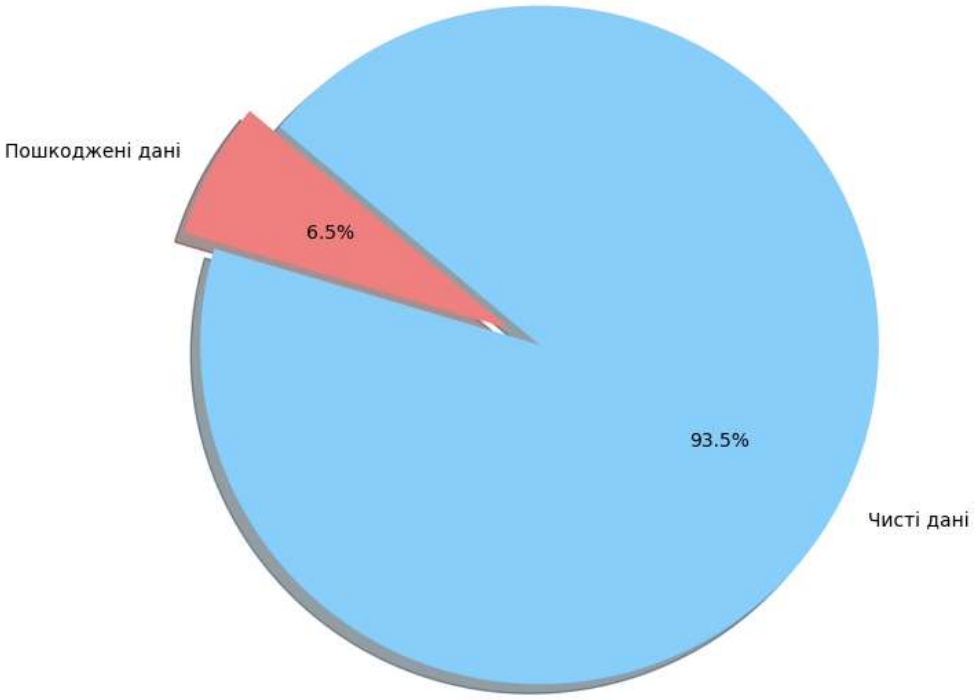
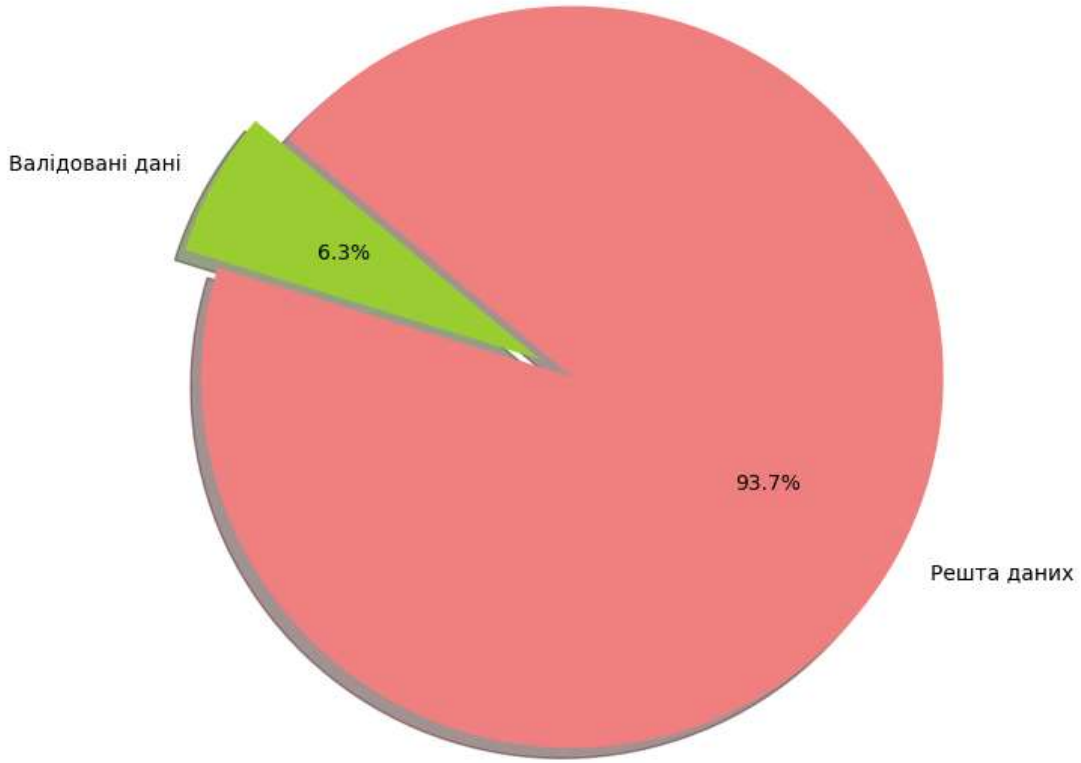


Рис. 4 Відсоток даних, валідованих моделлю



Аналіз пошкоджених даних по джерелах показав, що найбільший відсоток пошкоджених даних припадає на джерела source_8 та source_9, де пошкоджені дані становлять близько 20%. Інші джерела мають значно менший відсоток пошкоджених даних, що коливається в межах від 7.11% до 8.34%. Ці дані дозволяють ідентифікувати найбільш проблемні джерела, на які слід звернути особливу увагу при подальшій обробці та оптимізації.

Ефективність моделі RandomForestClassifier у визначенні пошкоджених даних підтверджується тим, що вона змогла ідентифікувати 98.26% пошкоджених даних. Такий високий відсоток вказує на високу точність моделі у виявленні некоректних записів. Це дозволяє зосередити зусилля на перевірці та корекції лише тих даних, які дійсно потребують уваги, що зменшує загальний обсяг роботи та покращує продуктивність системи.

Корекція даних за допомогою машинного навчання, зокрема використання моделі лінійної регресії для прогнозування та виправлення значень, дозволила ефективно виправити помилки у пошкоджених записах. Наприклад, неправильно форматовані email та некоректні значення віку були успішно виправлені, що підвищило загальну якість даних. Відсоток скоригованих пошкоджених даних також становив 97.14%, що відповідає відсотку валідації, визначеному моделлю.

Таким чином, машинне навчання суттєво впливає на скорочення обсягів обробки даних, оскільки дозволяє автоматизувати процеси валідації та корекції, зменшити обсяг даних, які потребують детальної перевірки, та підвищити точність і швидкість обробки. Використання методів класифікації та кластеризації дозволяє ефективно ідентифікувати проблемні записи та забезпечити їх корекцію, що знижує витрати часу та обчислювальних ресурсів, покращуючи при цьому загальну якість даних.

Крім того, машинне навчання забезпечує постійне вдосконалення процесів обробки даних за рахунок можливості адаптуватися до нових патернів даних. Це особливо важливо в умовах постійно зростаючих обсягів даних, які надходять з різних джерел. Завдяки автоматизованим процесам обробки та аналізу даних, організації можуть швидше реагувати на зміни, підтримувати високу якість даних та приймати більш обґрунтовані рішення на основі надійної інформації.

3.3 РЕКОМЕНДАЦІЇ ЩОДО ОПТИМІЗАЦІЇ ТА ПОКРАЩЕННЯ ЯКОСТІ ДАНИХ У DATA LAKEHOUSE АРХІТЕКТУРІ

Враховуючи результати проведених експериментів та аналізу, можна сформулювати декілька ключових рекомендацій щодо оптимізації та покращення якості даних у Data Lakehouse архітектурі за допомогою методів машинного навчання. Ці рекомендації стосуються як технічних аспектів обробки даних, так і стратегічного підходу до їхньої валідації та корекції.

Однією з найважливіших рекомендацій є впровадження автоматизованих інструментів моніторингу якості даних. Використання таких інструментів дозволяє здійснювати постійний контроль за станом даних у реальному часі, своєчасно виявляти аномалії та інші відхилення від норми. Це забезпечує швидке реагування на проблеми з даними та мінімізує ризики, пов'язані з використанням некоректної інформації для прийняття рішень.

Застосування ансамблевих методів машинного навчання, таких як RandomForest, дозволяє досягти високої точності класифікації та виявлення пошкоджених даних. Ансамблеві моделі комбінують результати кількох базових моделей, що дозволяє отримати більш надійні та стійкі результати [30]. Це особливо важливо при роботі з великими наборами даних, які можуть містити значну кількість шуму та некоректних записів.

Для забезпечення ефективного доступу до даних та зменшення часу виконання запитів рекомендується використовувати сучасні формати зберігання, такі як Delta Lake. Застосування таких форматів дозволяє забезпечити ACID-транзакції, що є критично важливим для підтримання цілісності даних. Крім того, оптимізація таблиць за допомогою команд типу OPTIMIZE ZORDER BY значно покращує продуктивність запитів, що є важливим фактором при роботі з великими обсягами даних [21].

Аналіз результатів показав, що деякі джерела даних є значно більш схильними до помилок. Зокрема, джерела source_8 та source_9 мали дуже високий відсоток пошкоджених даних. Рекомендується проводити додатковий моніторинг та детальний аналіз таких джерел для виявлення причин високого рівня пошкоджених даних. Це може включати аналіз процесів збору даних, використання додаткових фільтрів та правил валідації на етапі завантаження даних.

Застосування моделей машинного навчання для прогнозування та корекції даних дозволяє автоматично виправляти некоректні записи, що суттєво покращує загальну якість даних. Використання таких підходів, як лінійна регресія для прогнозування значень або спеціалізовані алгоритми для корекції форматування email та вікових значень, дозволяє зменшити необхідність ручної обробки даних і забезпечити їх високу точність.

Методи кластеризації, такі як DBSCAN, є ефективними для виявлення аномалій у даних. Застосування таких методів дозволяє ідентифікувати викиди та інші відхилення, що можуть вказувати на наявність пошкоджених записів. Це забезпечує додатковий рівень перевірки даних та допомагає забезпечити їх високу якість.

Моделі машинного навчання повинні постійно навчатися та адаптуватися до нових даних. Це забезпечує їх актуальність і високу продуктивність навіть при зміні умов та характеристик даних. Регулярне

оновлення моделей та їх перевірка на нових даних дозволяє підтримувати високу якість обробки та валідації даних.

Рекомендації, наведені у цьому розділі, спрямовані на покращення якості даних у Data Lakehouse архітектурі за допомогою сучасних методів машинного навчання. Використання автоматизованих інструментів моніторингу, оптимізація процесів зберігання, фокус на найбільш проблемні джерела даних, а також застосування моделей для прогнозування, корекції та виявлення аномалій забезпечують високий рівень якості даних та ефективність їх обробки. Впровадження цих рекомендацій дозволить організаціям не лише покращити якість своїх даних, але й підвищити ефективність прийняття рішень на всіх рівнях бізнесу.

ВИСНОВКИ

У процесі написання даної роботи було досліджено різні аспекти валідації та покращення якості даних у середовищах Big Data. Проведений аналіз дозволяє зробити наступні висновки:

1. Теоретичні основи та значення валідації Big Data:

- Велика кількість даних, що генеруються щодня, потребує ретельної валідації для забезпечення їх точності та надійності.
- Валідація даних є важливою складовою процесу обробки даних, що дозволяє знизити ризики прийняття неправильних рішень на основі некоректних даних.

2. Проблеми забезпечення якості Big Data:

- Основні проблеми включають різноманітність джерел даних, великий обсяг даних, швидкі зміни в даних та відсутність єдиних стандартів якості.
- Ці проблеми ускладнюють процес валідації та потребують застосування спеціалізованих інструментів та методів.

3. Технологічні інструменти валідації:

- Застосування фреймворків, таких як Apache Spark, дозволяє ефективно обробляти та валідувати великі обсяги даних.
- Використання Apache Spark спрощує процеси очищення, трансформації та перевірки даних, забезпечуючи високу продуктивність і точність обробки.
- Платформа Databricks забезпечує інтегроване середовище для роботи зі Spark, спрощуючи процес налаштування та обслуговування інфраструктури.
- Використання Databricks дозволяє зосередитися на аналізі даних та розробці рішень без необхідності витрачати час на налаштування інфраструктури.

4. Роль машинного навчання у валідації даних:

- Алгоритми машинного навчання дозволяють автоматизувати процеси очищення та валідації даних, виявляти аномалії та виправляти помилки у даних.
- Використання машинного навчання забезпечує більш високу точність та ефективність управління якістю даних у порівнянні з традиційними методами.

5. Практичне застосування методів машинного навчання:

- У процесі дослідження було розроблено модель машинного навчання для класифікації та корекції даних, що показала високу ефективність у визначенні пошкоджених даних.
- Модель дозволила зменшити обсяги даних, які потребують перевірки вручну, що знизило витрати часу та ресурсів на валідацію даних.

6. Рекомендації щодо впровадження методів валідації:

- Для забезпечення високої якості даних рекомендується використовувати автоматизовані інструменти для валідації та корекції даних, а також впроваджувати процеси моніторингу та контролю якості даних на всіх етапах їх обробки.
- Рекомендується застосовувати архітектуру Data Lakehouse для організації зберігання та обробки даних, що дозволяє поступово покращувати якість даних на кожному етапі обробки.

Дана робота демонструє важливість валідації даних у середовищах Big Data та пропонує ефективні методи та інструменти для забезпечення високої якості даних, що є критично важливим для прийняття обґрунтованих рішень на основі аналізу даних.

ОЦІНКА НОВИЗНИ РОБОТИ

У процесі виконання даної роботи було досягнуто кілька нових підходів та рішень у сфері валідації та покращення якості Big Data:

1. Інтеграція машинного навчання у процеси валідації даних.

Використання алгоритмів машинного навчання для автоматичної валідації та корекції даних є інноваційним підходом, який дозволяє значно підвищити точність та ефективність управління якістю даних. Традиційні методи, засновані на ручній перевірці та простих правилах, поступаються місцем більш складним та адаптивним алгоритмам, що дозволяють швидко ідентифікувати та виправляти помилки у великих наборах даних.

2. Застосування Data Lakehouse архітектури.

Запропонована архітектура Data Lakehouse, яка поєднує переваги Data Lake та Data Warehouse, є новим підходом до організації зберігання та обробки даних. Вона дозволяє зберігати дані у різних станах обробки (бронзовий, срібний та золотий рівні), що забезпечує поступове покращення їх якості на кожному етапі обробки.

3. Використання Delta Lake для забезпечення цілісності даних.

Використання технології Delta Lake для зберігання та обробки даних дозволяє забезпечити ACID-транзакції та високу консистентність даних. Це забезпечує можливість проведення надійної валідації та корекції даних, а також покращує продуктивність доступу до даних завдяки оптимізації зберігання.

4. Експериментальна оцінка ефективності методів валідації.

Проведені експерименти з використанням згенерованих даних дозволили оцінити ефективність різних методів валідації та корекції даних. Використання бібліотеки Faker для генерації великих наборів даних, що

включають як коректні, так і пошкоджені дані, дозволило створити реалістичні сценарії для тестування моделей машинного навчання.

5. Логування експериментів за допомогою Mlflow.

Використання MLflow для логування експериментів з навчання моделей машинного навчання дозволяє зберігати та аналізувати результати експериментів, що робить процес аналізу та відтворення результатів більш надійним та ефективним. Це дозволяє дослідникам легко відслідковувати ефективність моделей та вдосконалювати їх на основі отриманих даних.

Таким чином, робота має новизну завдяки інтеграції сучасних технологій та методів для забезпечення високої якості даних у середовищах Big Data. Використання машинного навчання, Data Lakehouse архітектури та інструментів для логування експериментів дозволяє значно підвищити точність та ефективність процесів валідації та корекції даних, що є критично важливим для сучасних бізнесів та наукових досліджень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cambridge Dictionary. (2023). Big Data. Retrieved from https://dictionary.cambridge.org/dictionary/english/big-data

2. Oracle. What is Big Data?. Retrieved from https://www.oracle.com/big-data/what-is-big-data/

3. Gartner's Top 10 IT Challenges. Retrieved from https://webcitation.org/6AOqGc0Bs?url=http://www.computerworld.com/s/article/9220975/Gartner_s_Top_10_IT_challenges_include_exiting_baby_boomers_Big_Data

4. Laney, D. (2012). 3D Data Management: Controlling Data Volume, Velocity and Variety. Retrieved from https://web.archive.org/web/20130723080959/http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf

5. IBM Big Data Hub. 4 Vs of Big Data. Retrieved from https://web.archive.org/web/20160616211640/http://www.ibmbigdatahub.com/sites/default/files/infographic_file/4-Vs-of-big-data.jpg

6. Reis, Joe, and Matt Housley. Fundamentals of Data Engineering: Plan and Build Robust Data Systems. O'Reilly Media, 2022.

7. Skillfield. How much is poor quality data really costing your business?. Retrieved from https://www.linkedin.com/pulse/how-much-poor-quality-data-really-costing-your-business-skillfield/

8. Li, J. Z., & Liu, X. M. (2013). An Important Aspect of Big Data: Data Usability. Journal of Computer Research and Development, 50(6), 1147–1162.

9. Demchenko, Y., Grosso, P., de Laat, C., et al. (2013). Addressing Big Data Issues in Scientific Data Infrastructure. Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, California: ACM, 48–55.

10. Cai, L., & Zhu, Y. (n.d.). The Challenges of Data Quality and Data Quality Assessment in the Big Data Era.

11. Statista. Worldwide data created. Retrieved from https://www.statista.com/statistics/871513/worldwide-data-created/

12. Pradhan, A. (n.d.). Challenges associated with real-time data analytics quality. Retrieved from https://www.linkedin.com/pulse/challenges-associated-real-time-data-analytics-quality-atul-pradhan/

13. Wang, R., & Storey, V. (1995). Framework for Analysis of Quality Research. IEEE Transactions on Knowledge and Data Engineering, 1(4), 623–637.

14. ISO 8000-1:2011. Retrieved from https://www.iso.org/obp/ui/#iso:std:iso:8000:-1:ed-1:v1

15. McGilvray, D. (2010). Executing Data Quality Projects: Ten Steps to Quality Data and Trusted Information. Beijing: Publishing House of Electronics Industry.

16. Silberschatz, A., Korth, H., & Sudarshan, S. (2006). Database System Concepts. Beijing: Higher Education Press.

17. Cappiello, C., Francalanci, C., & Pernici, B. (2004). Data quality assessment from user's perspective. Proceedings of the 2004 International Workshop on Information Quality in Information Systems, New York: ACM, 78–73.

18. Big Data Engineering. Retrieved from https://dou.ua/lenta/articles/what-is-big-data-engineering/

19. Extract Transform Load. Retrieved from https://www.databricks.com/glossary/extract-transform-load

20. Data Validation. Retrieved from https://airbyte.com/data-engineering-resources/data-validation

21. ETL, ELT, and ETLT. Retrieved from https://www.linkedin.com/pulse/etl-elt-etlt-evolution-data-pipelines-ascend-io/

22. McKinsey & Company. Retrieved from https://www.mckinsey.com/capabilities/growth-marketing-and-sales/our-insights

23. “Big Data Use Cases 2015 – Getting Real On Data Monetization“, published by BARC. Retrieved from http://barc-research.com/research/big-data-use-cases-2015/?__hstc=158107557.87a26ce36c727a0ca31065dbfb789ae3.1716451106908.1716451106908.1716451106908.1&__hssc=158107557.1.1716451106908&__hsfp=3523199817

24. Apache Spark. Retrieved from https://spark.apache.org/

25. Databricks. Solutions for Data Engineering. Retrieved from https://www.databricks.com/solutions/data-engineering

26. Databricks. Medallion Architecture. Retrieved from https://www.databricks.com/glossary/medallion-architecture

27. PwC. Artificial Intelligence Study. Retrieved from https://www.pwc.com/gx/en/issues/data-and-analytics/publications/artificial-intelligence-study.html

28. Hodgson, J. The 5 Stages of Machine Learning Validation Ensure high-quality machine learning across the ML lifecycle. Towards Data Science.

29. Hapke, H., & Nelson, C. Building Machine Learning Pipelines. O'Reilly Media, 2020.

30. Breck, E., Zinkevich, M., Polyzotis, N., Whang, S., & Roy, S. Data Validation for Machine Learning. Proceedings of SysML (2019). Google Scholar.

31. Agarwal, A. Data Validation in Machine Learning is imperative, not optional.

32. Databricks Community. Retrieved from https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/2634267698425259/614288528932443/4669994005016133/latest.html