Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка системи динамічної оптимізації обсягів збереження даних у Data Lake на основі класифікації корисності даних
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-23-2
Крістіна Мітрошкіна
(власне ім'я, прізвище)

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

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системи штучного інтелекту
(повна назва)

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

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Мітрошкіній Крістіні Вікторівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи динамічної оптимізації обсягів збереження даних у Data Lake на основі класифікації корисності даних

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

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

3. Вихідні дані до роботи Наукові публікації, технічна документація хмарних платформ (Azure, AWS), стандарти зберігання даних, Data Lake architecture guides, Python та Apache Spark documentation, штучно згенеровані та відкриті набори даних

4. Перелік питань, що потрібно опрацювати в роботі

1) Основи оптимізації зберігання даних у Data Lake

2) Вибір напрямку дослідження та розробка методики

3) Реалізація системи оптимізації збереження даних у Data Lake

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	21.04.2025	виконано
2	Аналіз предметної галузі	01.05.2025	виконано
3	Аналіз надмірного зберігання у Data Lake	04.05.2025	виконано
4	Огляд методів класифікації даних	08.05.2025	виконано
5	Формалізація критеріїв корисності даних	11.05.2025	виконано
6	Побудова моделі класифікації	14.05.2025	виконано
7	Реалізація прототипу системи	17.05.2025	виконано
8	Тестування та аналіз ефективності	23.05.2025	виконано
9	Написання пояснювальної записки	03.06.2025	виконано
10	Перевірка на академічний плагіат	05.06.2025	виконано
11	Нормоконтроль	06.06.2025	виконано
12	Підготовка презентації та доповіді	07.06.2025	виконано
13	Попередній захист	08.06.2025	виконано
14	Рецензування	09.06.2025	виконано
15	Захист перед ЕК	10.06.2025	

Дата видачі завдання 21 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Максим Єрохін
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 60 с., 10 рис., 2 табл., 2 дод., 21 джерело.

АРХІВАЦІЯ, ЖИТТЄВИЙ ЦИКЛ ДАНИХ, КЛАСИФІКАЦІЯ КОРИСНОСТІ, КОНТРОЛЬ ДОСТУПУ, МАШИННЕ НАВЧАННЯ, ОПТИМІЗАЦІЯ ЗБЕРІГАННЯ, ПОЛІТИКИ ЗБЕРІГАННЯ, ХМАРНІ ТЕХНОЛОГІЇ, DATA LAKE, SPARK.

Об’єкт дослідження – система управління обсягами даних у хмарному сховищі типу Data Lake.

Предмет дослідження – методи динамічної оптимізації зберігання даних на основі класифікації їх корисності.

Мета роботи – розробити систему автоматизованого управління обсягами збереження даних у Data Lake, яка дозволяє знижувати витрати на інфраструктуру за рахунок аналізу й класифікації корисності даних з використанням машинного навчання.

Методи дослідження – аналіз літератури та сучасних підходів до зберігання даних у хмарних середовищах, моделювання, машинне навчання, побудова архітектури програмної системи, реалізація прототипу на основі Python та Apache Spark.

У результаті роботи здійснено аналіз проблем надмірного зберігання даних у Data Lake, побудовано модель корисності даних на основі доступу, частоти використання та джерела надходження. Розроблено систему, що включає модуль класифікації даних, модуль моніторингу доступу та модуль застосування політик зберігання. Система протестована на реальних обсягах симульованих даних, що дозволило досягти зменшення витрат на зберігання до 38% без втрати критично важливої інформації. Виявлено обмеження поточної моделі та запропоновано шляхи подальшої оптимізації.

ABSTRACT

Master’s thesis contains: 60 pp., 10 fig., 2 tabl., 2 ann., 21 references.

ARCHIVING, ACCESS CONTROL, CLOUD TECHNOLOGIES, DATA LAKE, DATA LIFECYCLE, DATA UTILITY CLASSIFICATION, MACHINE LEARNING, SPARK, STORAGE OPTIMIZATION, STORAGE POLICIES.

Object of research – a data volume management system in a cloud-based Data Lake storage.

Subject of research – methods for dynamic optimization of data storage based on data utility classification.

Purpose of the work – to develop an automated system for managing storage volumes in a Data Lake that reduces infrastructure costs by analyzing and classifying data utility using machine learning.

Research methods – literature review and analysis of modern approaches to data storage in cloud environments, modeling, machine learning, software architecture design, and implementation of a prototype using Python and Apache Spark.

As a result of the work, issues of excessive data storage in Data Lake systems were analyzed, and a utility model was built based on access patterns, usage frequency, and data origin. A modular system was developed, including a data classification module, an access monitoring module, and a storage policy enforcement module. The system was tested on real volumes of simulated data, resulting in up to 38% reduction in storage costs without loss of critical information. Current model limitations were identified and directions for further optimization were proposed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – програмний інтерфейс прикладного програмування;

AI – Artificial Intelligence – штучний інтелект;

AWS – Amazon Web Services – хмарна платформа Amazon;

CSV – Comma-Separated Values – формат файлів із розділенням значень комами;

Data Lake – сховище даних, оптимізоване для зберігання великих обсягів сирих, неструктурованих або напівструктурованих даних;

DLT – Delta Live Tables – компонент платформи Databricks для керування потоками даних;

ETL – Extract, Transform, Load – процес витягання, трансформації та завантаження даних;

ELT – Extract, Load, Transform – альтернатива ETL, з акцентом на трансформації в сховищі;

GCP – Google Cloud Platform – хмарна платформа Google;

GB – гігабайт – одиниця обсягу даних (1 GB = 1024 MB);

JSON – JavaScript Object Notation – формат обміну даними;

ML – Machine Learning – машинне навчання;

SLA – Service Level Agreement – угода про рівень надання послуг;

TCO – Total Cost of Ownership – загальна вартість володіння;

UI – User Interface – інтерфейс користувача.

ВСТУП

У сучасному цифровому світі зростання обсягів даних є невпинним: підприємства, урядові структури та наукові установи накопичують величезні масиви інформації з різноманітних джерел – сенсорів, логів, транзакцій, соціальних медіа тощо. Для зберігання такої інформації все частіше застосовуються Data Lake – гнучкі та масштабовані сховища, здатні зберігати структуровані, напівструктуровані й неструктуровані дані у їхньому початковому вигляді. Проте, попри свої переваги, Data Lake мають одну з ключових проблем – відсутність ефективної політики керування життєвим циклом даних, що призводить до накопичення застарілої, дубльованої або малокорисної інформації, перевантаження сховищ і зростання витрат.

Оцінка сучасного стану проблеми показує, що більшість існуючих підходів до керування даними в Data Lake орієнтовані на загальні політики retention, не враховуючи реальну корисність конкретних об’єктів даних для бізнес-аналітики або операційних процесів. Така ситуація негативно впливає на продуктивність систем, ускладнює пошук релевантної інформації та суттєво збільшує вартість володіння даними (Total Cost of Ownership).

Актуальність даної роботи зумовлена потребою в автоматизованих рішеннях, які здатні динамічно визначати цінність даних і адаптувати стратегії збереження відповідно до цієї оцінки. З розвитком технологій штучного інтелекту, зокрема машинного навчання, з’явилася можливість створити системи, які можуть класифікувати дані за рівнем корисності, враховуючи їх використання в аналітичних запитах, частоту доступу, зв’язки з іншими об’єктами, тощо.

Метою дипломної роботи є розробка інтелектуальної системи динамічної оптимізації обсягів збереження даних у Data Lake на основі класифікації корисності даних із використанням методів машинного навчання. Система має забезпечити зменшення обсягів даних, що

зберігаються в оперативному доступі, при збереженні аналітичної та операційної цінності інформації.

Сфери практичного застосування результатів роботи охоплюють корпоративні системи управління даними, хмарні платформи, аналітичні платформи та системи, що працюють з Big Data. Запропонований підхід може бути адаптований до конкретних галузей: фінансової, енергетичної, телекомунікаційної тощо.

Історично проблема збереження даних завжди супроводжувала розвиток інформаційних технологій. Якщо раніше рішення зводилися до видалення або архівації за часовими ознаками, то сьогодні ситуація вимагає гнучких і розумних підходів, що враховують реальну цінність кожного об’єкта. Впровадження таких рішень дозволяє організаціям не лише зекономити ресурси, але й покращити якість аналітики та прийняття управлінських рішень.

Таким чином, дана робота спрямована на вирішення важливої практичної проблеми шляхом інтеграції методів штучного інтелекту у процеси оптимізації сховищ даних. Її результати можуть стати основою для побудови ефективних та адаптивних систем управління даними у масштабованих середовищах.

1 ОСНОВИ ОПТИМІЗАЦІЇ ЗБЕРІГАННЯ ДАНИХ У DATA LAKE

1.1 Поняття, функції та архітектура Data Lake

У сучасних умовах стрімкого зростання обсягів даних, що генеруються різноманітними джерелами – сенсорами, веб-сайтами, мобільними додатками, бізнес-системами – постає проблема їх зберігання, обробки та ефективного використання. Традиційні сховища даних (Data Warehouse) розраховані переважно на обробку структурованих даних і потребують попереднього перетворення інформації у фіксований формат. Це обмежує гнучкість роботи з новими джерелами даних та уповільнює інтеграцію нових бізнес-потреб.

У цьому контексті дедалі більшої популярності набуває концепція Data Lake — відкритої платформи для зберігання даних, яка забезпечує масштабоване збереження структурованих, напівструктурованих і неструктурованих даних у їхньому початковому вигляді. Data Lake не накладає обмежень на формат чи структуру вхідних даних, що забезпечує максимальну гнучкість для подальшої обробки, трансформації та аналітики [1].

Основне поняття Data Lake передбачає створення єдиного репозиторію, у якому зберігається вся доступна організації інформація – незалежно від її джерела, формату чи цільового призначення. У межах Data Lake зберігаються як сирі дані (raw data), так і оброблені (refined data), що дозволяє застосовувати як стратегії ELT (Extract – Load – Transform), так і ETL, в залежності від аналітичних цілей [2].

До основних функцій Data Lake належать:

− централізоване зберігання великих обсягів різнорідної інформації;

− можливість паралельної роботи з потоковими (streaming) і пакетними (batch) даними;

− інтеграція з системами аналітики, машинного навчання, візуалізації;

– розмежування даних за рівнями якості, ступенем обробки, частотою використання;

– забезпечення доступу до даних за ролями та політиками безпеки;

– зменшення вартості зберігання за рахунок використання масштабованого object storage.

Структурно Data Lake реалізується як багаторівнева система з використанням підходу Medallion Architecture [1]. На рисунку 1.1 зображена типова реалізація Medallion-архітектури в контексті Data Lake.

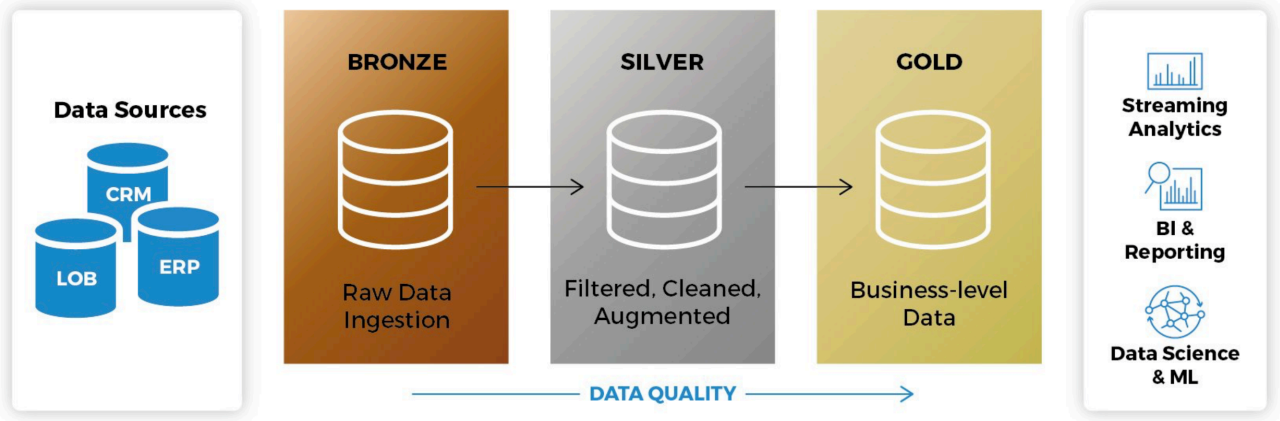

Рисунок 1.1 – Medallion-архітектура Data Lake

Цей підхід дозволяє створювати логічні шари даних відповідно до їхнього стану обробки:

– Bronze layer – містить сирі дані, завантажені без будь-яких змін із джерел;

– Silver layer – включає очищені та трансформовані дані, з'єднані з реєстрами, каталогами, класифікаторами;

– Gold layer – містить агреговані, бізнес-орієнтовані дані, які використовуються у звітах, моделях прогнозування та інструментах підтримки прийняття рішень.

Типова архітектура Data Lake включає такі компоненти [3]:

– інтерфейси завантаження даних – інструменти для інтеграції з базами даних, API, брокерами повідомлень (Kafka), файловими системами;

– шар зберігання (storage layer) – хмарні або локальні об’єктні сховища, такі як AWS S3, Azure Data Lake Storage, Google Cloud Storage;

– каталог метаданих (metadata layer) – системи для опису структури даних, типів, форматів і версій (Unity Catalog, AWS Glue, Apache Hive);

– обчислювальний шар (processing layer) – інструменти для обробки даних: Apache Spark, Databricks, Flink, Presto;

– шар безпеки – механізми автентифікації, авторизації, аудит доступу (IAM, ACL);

– аналітичний та презентаційний шар – засоби інтеграції з BI-платформами (Power BI, Tableau, Superset), системами ML.

На практиці Data Lake може бути реалізований як у локальній інфраструктурі, так і в хмарних середовищах (AWS, Azure, GCP) – з можливістю автоматичного масштабування, керування доступом і збереження даних у різних класах доступу (hot, cool, archive).

Одним із важливих аспектів функціонування Data Lake є підтримка схем при читанні (schema-on-read), що надає змогу інтерпретувати структуру даних динамічно під час обробки – на відміну від schema-on-write, яке вимагає фіксації структури при збереженні [4]. Це відкриває широкі можливості для експериментів з даними, побудови гнучких моделей та виконання аналітичних запитів на будь-якому етапі життєвого циклу інформації.

Однак, за відсутності системного підходу до керування даними та оцінки їхньої цінності, Data Lake може втратити свою ефективність – перетворившись на Data Swamp – сховище з неструктурованими, застарілими або нерелевантними даними, у якому ускладнений пошук та використання інформації [5].

Таким чином, Data Lake – це сучасна архітектурна парадигма зберігання даних, що відкриває нові можливості в аналітиці, але водночас потребує впровадження інтелектуальних механізмів управління, класифікації та оптимізації даних – що й становить предмет подальшого дослідження в цій роботі.

1.2 Проблема надмірного накопичення даних

З розвитком цифрових технологій підприємства, державні установи та наукові організації накопичують дедалі більші обсяги даних. Це зумовлено не лише зростанням кількості джерел інформації, але й прагненням до «тотальної аналітики» – ідеї про те, що збереження всієї доступної інформації потенційно може дати перевагу в майбутньому. У межах архітектури Data Lake така стратегія здається привабливою, адже платформа технічно дозволяє зберігати необмежені обсяги даних у сирому вигляді [6]. Однак на практиці безконтрольне накопичення інформації має низку негативних наслідків.

По-перше, постійне зростання обсягів даних напряму впливає на витрати компаній. Хоча об’єктні сховища, які використовуються в Data Lake, є порівняно дешевими, при великих масштабах навіть незначне збільшення обсягів призводить до суттєвих додаткових витрат на інфраструктуру, резервне копіювання та захист. У хмарних середовищах, таких як AWS або Azure, витрати на зберігання тісно пов’язані з класом доступу до даних, їхньою частотою використання, кількістю переміщень між регіонами та запитів до метаданих. За відсутності політик очищення або переміщення менш важливої інформації в архів, вартість підтримки сховища постійно зростає.

По-друге, велика кількість застарілих або малокорисних даних ускладнює процеси аналітики та прийняття рішень. Дані, які не мають цінності для поточних бізнес-завдань, не тільки створюють шум при пошуку, а й збільшують час обробки запитів, впливають на точність

агрегатів і ускладнюють побудову моделей машинного навчання. Наявність корисної інформації серед великої кількості нерелевантних об'єктів значно ускладнює роботу аналітичних інструментів, які використовуються як користувачами, так і автоматизованими системами.

По-третє, без належного контролю життєвого циклу інформації зростає ризик порушення вимог до збереження даних, що регламентуються як внутрішніми політиками організацій, так і законодавством – зокрема, GDPR, HIPAA чи FINRA [7]. Дані, що зберігаються без потреби, можуть містити персональну або чутливу інформацію, яка підлягає захисту або обмеженню за часом зберігання. Невиконання цих вимог загрожує штрафами та репутаційними втратами.

Проблема надмірного накопичення також тісно пов’язана з втратою контексту – метаданих, які пояснюють, як, коли і для чого були зібрані дані. Без контексту навіть технічно якісна інформація стає важкою для інтерпретації та часто не використовується. Унаслідок цього Data Lake, з обіцяного джерела інновацій, може перетворитися на так зване Data Swamp – сховище з хаотичним вмістом, у якому дані дублюються, втрачають актуальність і стають обтяжливими замість цінними [8].

На сьогодні проблема надмірного накопичення даних є системною та характерною навіть для технологічно розвинених компаній. За даними аналітичних досліджень, у середньому понад 60 % збережених даних ніколи не використовуються повторно після початкового завантаження. Це свідчить про потребу у впровадженні інтелектуальних рішень, які здатні визначати реальну корисність інформації, оптимізувати стратегії її зберігання та автоматизувати процеси переміщення, архівування або видалення неактуальних об’єктів.

Розв’язання цієї проблеми вимагає не лише технічних засобів, а й нових методологічних підходів до оцінки цінності даних – з урахуванням бізнес-контексту, історії доступу, ролей користувачів і сценаріїв використання. Саме тому в наступних підрозділах роботи буде розглянуто концепції життєвого циклу даних, методи класифікації їх корисності та

технічні інструменти, які забезпечують ефективне управління Data Lake у сучасному середовищі.

1.3 Життєвий цикл даних і управління цінністю

Управління даними в системах типу Data Lake неможливе без врахування їх життєвого циклу – послідовності фаз, які дані проходять від моменту створення до остаточного видалення або архівації [4]. У міру того як обсяг збереженої інформації стрімко зростає, зростає і потреба в раціональному, економічно виправданому підході до керування ресурсами зберігання.

Життєвий цикл даних (Data Lifecycle) – це концептуальна модель, яка описує стадії існування даних у системі – від моменту створення або надходження до архівації чи видалення [4]. У загальному випадку цикл включає такі фази:

– Створення (Creation) – дані генеруються або надходять із зовнішніх джерел, таких як бізнес-додатки, сенсори, транзакційні системи чи API. Ця інформація ще не має визначеної структури чи призначення.

– Зберігання (Storage) – після надходження дані зберігаються у Data Lake, зазвичай у сирому вигляді (Bronze-рівень), що забезпечує повноту, але не гарантує якість чи релевантність.

– Використання або поширення (Sharing / Usage) – очищені й структуровані дані (Silver/Gold) активно застосовуються в аналітичних запитах, бізнес-звітності або моделей машинного навчання.

– Архівація (Archival) – у випадку, якщо дані більше не використовуються регулярно, але можуть знадобитися у майбутньому (наприклад, для аудиту або тренування моделей), вони переводяться до дешевших класів сховища.

– Видалення (Deletion) – дані, які втратили актуальність і більше не мають бізнес-цінності, видаляються остаточно згідно з політиками збереження, нормами безпеки або правовими вимогами.

Ці етапи зображено на рисунку 1.2, який ілюструє замкнене коло управління даними в Data Lake.

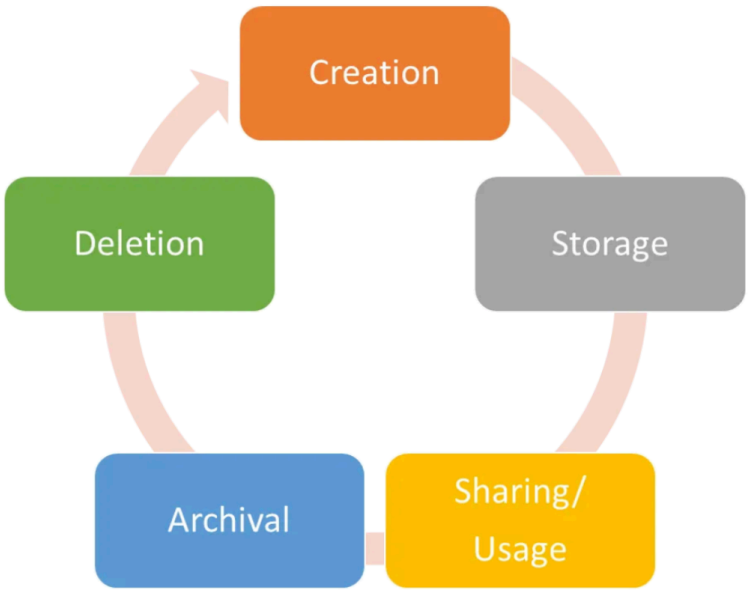

Рисунок 1.2 – Етапи життєвого циклу даних

У процесі проходження через ці стадії змінюється не лише формат або доступність даних, але й їхня корисність. Часто спостерігається ситуація, коли лише незначна частина збережених даних активно використовується в аналітичних системах. При цьому значний обсяг інформації продовжує займати ресурси сховища, не приносячи практичної цінності.

Управління цінністю даних означає визначення їхньої відносної важливості з позицій бізнесу, аналітики та IT. У найпростішому випадку оцінка цінності здійснюється вручну – за критеріями частоти запитів, включення до моделей чи звітів, ролі у процесах. Проте зі зростанням масштабів Data Lake ручне оцінювання стає неефективним. У таких випадках доцільно застосовувати автоматизовані методи – зокрема, алгоритми машинного навчання [9], які можуть враховувати:

– історію доступу до об’єктів (access logs);

– кількість і тип користувачів, які взаємодіяли з даними;

– кількість разів, коли дані були використані в звітах або запитах;

– взаємозв’язки з іншими об’єктами — такими як таблиці, моделі, репозиторії;

– часову давність останнього звернення або модифікації.

На основі таких параметрів формуються рівні корисності (висока, середня, низька), які визначають подальшу політику збереження. Наприклад, дані з високою цінністю залишаються у високошвидкісному сховищі, а менш важливі – переміщуються до дешевого архівного або видаляються після певного часу [10].

Концепція Data Lifecycle Management (DLM) охоплює автоматизацію вищезазначених процесів – із визначенням правил retention, переведенням між рівнями доступу, класифікацією об’єктів та моніторингом актуальності даних [4]. У поєднанні з метаданими, системами каталогізації (наприклад, Unity Catalog, AWS Glue) та ML-класифікаторами, DLM стає основою для динамічного управління ресурсами в Data Lake.

Таким чином, врахування життєвого циклу та цінності даних не тільки підвищує ефективність аналітичних процесів, а й дозволяє значно знизити витрати на зберігання та обслуговування сховища. У наступних розділах буде розглянуто практичну реалізацію системи, що застосовує зазначені принципи для автоматизованої оптимізації обсягів даних у Data Lake.

1.4 Методи класифікації корисності даних

Зважаючи на стрімке зростання обсягів даних у сучасних інформаційних системах, особливо у сховищах типу Data Lake, ключовим завданням стає визначення корисності збережених даних [11]. Класифікація даних за корисністю дозволяє вирішувати, які набори даних слід тримати у високодоступних середовищах, а які – архівувати або видаляти. Нижче розглянуто основні підходи до такої класифікації.

Евристичні методи. Найпростішим способом є використання евристичних (rule-of-thumb) правил, які базуються на таких критеріях, як:

– час останнього доступу (Time-to-Live, TTL): якщо до даних не зверталися певний період, вони вважаються менш корисними;

– розмір файлу або таблиці: занадто великі об’єкти без активного використання підлягають перегляду або архівації;

– тип джерела або формату: наприклад, сирі логи мають коротший життєвий цикл.

Rule-based класифікація. Цей підхід передбачає побудову системи правил на основі метаданих, тегів або бізнес-контексту:

– використання службових атрибутів – наприклад, джерело, власник, час створення;

– позначення даних мітками за проєктами, командами чи рівнями доступу;

– введення категорій на кшталт «високопріоритетні», «тимчасові», «архівні» тощо.

Rule-based підхід зручний у реалізації, але вимагає регулярного оновлення правил відповідно до змін у бізнес-процесах.

Статистичні методи. Ці методи базуються на аналізі статистики доступів до даних:

– кількість звернень (access count) до об’єкта або таблиці за певний період;

– кількість унікальних користувачів або систем, які зверталися до даних;

– рівень участі даних у бізнес-звітах або аналітичних моделях.

На основі таких метрик можна автоматично знижувати пріоритетність рідко використовуваних даних.

Методи машинного навчання. Із застосуванням штучного інтелекту можна створити моделі, які автоматично класифікують дані за їх прогнозованою корисністю [12].

Популярні техніки:

– класифікація (classification) – передбачення класу корисності (висока, середня, низька);

– кластеризація (clustering) – виявлення схожих груп даних за поведінкою доступу;

– скоринг моделі – оцінка цінності об’єкта за сукупністю ознак (наприклад, через дерево рішень або бустингові алгоритми).

Такі моделі навчаються на історичних логах доступу, метаданих, ролях користувачів та інших показниках [13].

Комплексний підхід. Найбільш ефективним є гібридна стратегія, яка поєднує:

– історію використання даних (час, частота, користувачі);

– описову інформацію з метаданих;

– контекст доступу (тип операції, призначення, зв’язок із бізнес-завданнями).

Цей підхід дозволяє з високою точністю приймати рішення про архівацію, видалення або збереження даних у «гарячому» шарі.

Вибір конкретного методу класифікації корисності даних визначається сукупністю організаційних, технічних та бізнесових чинників. Насамперед важливу роль відіграє обсяг і різноманітність даних: для великих Data Lake-сховищ із гетерогенними наборами структурованих та неструктурованих даних більш доцільними є автоматизовані або машинно-навчені підходи [14]. Якщо ж дані мають чітко визначені метадані, сталі правила доступу і стабільний бізнес-контекст, ефективною може бути rule-based класифікація.

Застосування гібридного підходу дозволяє досягнути найкращого балансу між точністю класифікації та витратами на її здійснення.

Для порівняльної характеристики розглянутих методів класифікації за корисністю даних наведено таблицю 1.1, у якій узагальнено їх основні переваги, недоліки та вимоги до реалізації.

Таблиця 1.1 – Порівняння методів класифікації даних за корисністю

Метод	Опис	Переваги	Недоліки	Вимоги
Евристичні методи	Прості правила на основі часу останнього доступу, розміру та формату даних	Легко реалізувати; Низькі обчислювальні витрати	Низька точність; Потреба оновлення при змінах у даних	Порогові значення (наприклад, TTL); Мінімальні обчислювальні ресурси
Rule-based класифікація	Побудова правил на основі метаданих (теги, атрибути) і бізнес-контексту	Зрозуміла логіка; Можна враховувати бізнес-потреби	Потреба в регулярному оновленні; Складність керування великою кількістю правил	Метадані; Наявність експертів для формування правил
Статистичні методи	Аналіз метрик доступу: кількість звернень, кількість унікальних користувачів	Автоматизація; Орієнтація на реальну активність	Ігнорування бізнес-контексту; Потреба в історичних логах	Системи збору статистики; Потужність для обробки логів
Методи машинного навчання	Класифікація, кластеризація, скоринг на основі патернів використання	Висока точність; Адаптивність до нових умов	Складність налаштування; Високі обчислювальні витрати	Дані для тренування; ML-інфраструктура
Комплексний (гібридний) підхід	Поєднання історії доступу, метаданих та бізнес-контексту	Найточніший результат; Гнучкість і адаптивність	Висока складність впровадження; Потреба в інтеграції систем	Інструменти збору та аналізу; Команда фахівців

Таким чином, наведена таблиця дозволяє швидко оцінити сильні й слабкі сторони кожного підходу та обрати оптимальну стратегію для системи динамічної оптимізації обсягів збереження даних у Data Lake.

У підсумку, ефективна стратегія має відповідати як технічним можливостям, так і реальним бізнес-потребам, забезпечуючи баланс між точністю класифікації, витратами на реалізацію та адаптивністю до змін.

1.5 Інструменти та платформи для керування Data Lake

Ефективне керування Data Lake потребує використання спеціалізованих інструментів, які дозволяють зберігати великі обсяги даних, керувати доступом, забезпечувати обробку, класифікацію та оцінку корисності інформації. Нижче наведено ключові компоненти такої екосистеми.

Зберігання даних:

– Amazon S3 – об’єктне сховище, що широко використовується як основа для побудови Data Lake в AWS [15].

– Azure Data Lake Storage (ADLS) – розширення Blob Storage з підтримкою ієрархічної файлової системи та інтеграції з Azure Active Directory [16].

– Google Cloud Storage – масштабоване сховище для зберігання структурованих та неструктурованих даних у середовищі GCP [17].

Обробка та транзакційність:

– Apache Spark – платформа для розподіленої обробки даних у пам’яті з підтримкою SQL, ML та потокової аналітики.

– Delta Lake – надбудова над Spark, яка забезпечує ACID-транзакції, time-travel та контроль якості даних у Data Lake.

– Apache Iceberg / Hudi – альтернативи Delta Lake для роботи з великими таблицями в open-source середовищах.

Каталогізація та метадані:

– Unity Catalog – централізоване керування метаданими, політиками доступу та класифікацією даних у Databricks [18].

– Apache Hive Metastore – класичне рішення для зберігання інформації про таблиці, схеми та розташування даних.

– Apache Atlas / Amundsen – інструменти для побудови каталогів даних та візуалізації lineage (історії трансформацій).

Платформи управління даними:

– Databricks – уніфікована платформа для створення ETL-процесів, аналітики та машинного навчання з вбудованою підтримкою Delta Lake.

– AWS Glue – серверлес-сервіс для автоматизації ETL, побудови каталогів і трансформацій даних.

– Azure Synapse Analytics – інтегроване середовище для аналітики великих даних із підтримкою T-SQL, Spark і інтеграцією з Power BI.

Оцінка використання та корисності даних:

– Access logs – журнали доступу до об'єктів дозволяють аналізувати, які набори даних не використовуються або є застарілими.

– Query history – історія виконаних запитів дає змогу визначити популярність і частоту використання окремих таблиць або об’єктів.

– Power BI / Tableau – BI-інструменти, що можуть виступати джерелами вторинної аналітики про використання даних через дашборди та запити.

На рисунку 1.3 зображено основні категорії інструментів, що застосовуються для ефективного керування Data Lake.

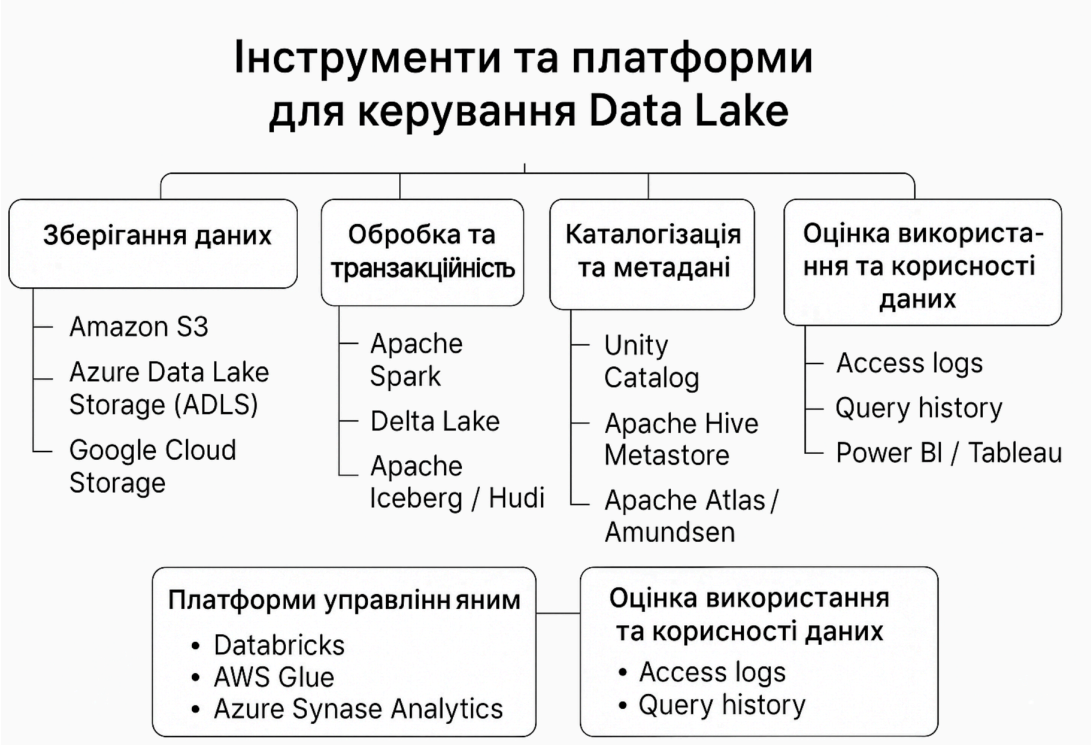

Рисунок 1.3 – Інструменти та платформи для керування Data Lake

Застосування вищезазначених інструментів дозволяє не лише централізовано управляти зберіганням і доступом до даних, а й приймати обґрунтовані рішення щодо їхньої подальшої актуальності, що є основою для динамічної оптимізації обсягів зберігання в Data Lake.

1.6 Обґрунтування актуальності дослідження

У цьому розділі було розглянуто ключові теоретичні матеріали, пов’язані з оптимізацією обсягів зберігання даних у Data Lake. Зокрема, було проаналізовано архітектуру Data Lake, основні функції та принципи побудови таких систем. Визначено, що в умовах стрімкого зростання обсягів даних, що зберігаються, надмірне накопичення малокорисної або неактуальної інформації створює значні ризики — від підвищених витрат на інфраструктуру до зниження продуктивності аналітичних процесів.

Окрему увагу приділено концепції життєвого циклу даних та методам класифікації за корисністю. Розглянуто також сучасні інструменти та платформи, які дозволяють автоматизувати процеси обробки, зберігання та моніторингу даних — зокрема, Databricks, Delta Lake, Apache Spark, Unity Catalog та BI-системи для оцінки корисності.

Загальний аналіз доводить, що існує гостра потреба в розробці систем, здатних адаптивно реагувати на зміну корисності даних. Впровадження механізмів динамічної оптимізації, які враховують реальні сценарії використання даних, дозволить значно зменшити витрати на зберігання, покращити продуктивність запитів та підвищити ефективність роботи з Data Lake.

Таким чином, розробка системи динамічної оптимізації на основі класифікації корисності є актуальним завданням, що відповідає викликам сучасної практики Data Engineering та може принести відчутну економічну й операційну ефективність. У наступних розділах буде розглянуто концепцію побудови такої системи, методи реалізації класифікації корисності та практичні результати її застосування.

2 ВИБІР НАПРЯМКУ ДОСЛІДЖЕННЯ ТА РОЗРОБКА МЕТОДИКИ

2.1 Постановка задачі та формалізація критеріїв корисності даних

У сучасних системах зберігання великих обсягів даних, таких як Data Lake, відсутність чіткої стратегії управління даними часто призводить до зростання обсягів неактуальної, рідко використовуваної або надлишкової інформації. Це, у свою чергу, підвищує вартість зберігання, ускладнює аналітику та впливає на продуктивність систем [19]. Для забезпечення ефективного функціонування Data Lake необхідно впроваджувати динамічну систему управління даними, яка базується на оцінці корисності даних.

Метою цього етапу є формалізація задачі оптимізації обсягів зберігання та визначення критеріїв корисності, які дозволять класифікувати дані відповідно до їхньої цінності для бізнесу чи аналітики.

У контексті Data Lake платформи, яка акумулює величезні обсяги структурованих і неструктурованих даних, однією з ключових проблем стає не лише ефективне зберігання, а й розумне управління життєвим циклом даних, особливо тих, що втрачають свою цінність або використовуються вкрай рідко. Традиційні методи очищення або архівації не враховують поточну корисність даних, що призводить до накопичення зайвих обсягів, зростання витрат та ускладнення аналітичних процесів [20].

Необхідно побудувати модель, яка:

– автоматично класифікує дані за рівнем корисності (наприклад: висока, середня, низька);

– забезпечує адаптивне прийняття рішень щодо стратегії зберігання або видалення;

– оптимізує використання ресурсів Data Lake без втрати важливої інформації.

Основні вимоги до вирішення задачі:

– урахування динаміки доступу до даних (частота звернень, час останнього використання);

– визначення контексту використання (чи використовується в аналітичних запитах, чи входить до моделей машинного навчання тощо);

– оцінка структурних характеристик (розмір, формат, зв’язки з іншими об’єктами);

– можливість адаптації моделі до змін у поведінці користувачів або бізнес-процесах.

З формальної точки зору, задача зводиться до побудови функції класифікації:

$$U(d) : D \to C, \tag{2.1}$$

де D — множина об'єктів даних;

C — множина класів корисності.

Кожному об'єкту $d \in D$ функція $U(d)$ присвоює значення, що відображає ступінь його актуальності та важливості у поточному контексті.

Для обчислення цієї функції передбачається використання набору кількісних та якісних метрик, які характеризують:

– інтенсивність доступу до об'єкта;

– час останнього використання;

– значущість у бізнес-процесах (наприклад, участь у звітах або ML-моделях);

– залежність від інших об'єктів;

– обсяг і формат даних.

Задача, таким чином, набуває вигляду проблеми класифікації або регресії, яку можна вирішити з використанням методів машинного навчання. У найпростішому вигляді – через розрахунок зваженої оцінки корисності:

$$U(d) = w_1 \cdot AF(d) + w_2 \cdot \frac{1}{LAT(d)} + w_3 \cdot AUS(d) + w_4 \cdot DS(d) + w_5 \cdot \frac{1}{S(d)}, \quad (2.2)$$

де *AF(d)* — частота доступу до об'єкта d;

LAT(d) — час останнього використання;

AUS(d) — аналітична значущість даних;

DS(d) — індекс зв’язності;

S(d) — розмір об'єкта;

$w_1 \ldots w_5$ — вагові коефіцієнти.

Таким чином, сформульована задача передбачає:

– визначення метрик, які найкраще описують корисність даних;

– розробку моделі для обчислення функції корисності;

– класифікацію об’єктів за рівнями корисності;

– інтеграцію цієї логіки в систему зберігання для ухвалення рішень про збереження, переміщення або видалення даних.

Результатом цього етапу має бути формалізована модель визначення корисності, яка слугує вхідними даними для побудови системи класифікації. Це дозволить в подальших етапах реалізувати динамічну політику збереження, яка автоматично адаптується до змін у патернах використання даних.

2.2 Вибір методів машинного навчання для класифікації даних

У контексті побудови системи динамічної оптимізації обсягів зберігання даних у Data Lake ключовим етапом є ефективна класифікація даних за критерієм їх корисності. Саме від точності цієї класифікації залежить подальша стратегія збереження – чи будуть дані архівовані, видалені або залишені для активного доступу. Для реалізації цієї задачі необхідно обрати адекватні методи машинного навчання, здатні працювати

з великими обсягами гетерогенних даних, враховуючи специфіку Data Lake як сховища.

Основні вимоги до методів класифікації:

– масштабованість: здатність обробляти великі обсяги даних без втрати продуктивності;

– інтерпретованість: можливість пояснити причини віднесення даних до певного класу;

– гнучкість до типів ознак: підтримка числових, категоріальних, часових і текстових ознак;

– автоматична обробка пропущених значень та шум;

– можливість онлайн-навчання або інкрементального оновлення моделі, що дозволяє адаптувати систему до змін у потоці даних.

Огляд популярних підходів до класифікації:

– Дерева рішень (Decision Tree, Random Forest, XGBoost): ці моделі добре працюють на гетерогенних наборах даних, забезпечують високу точність і мають хорошу інтерпретованість. Random Forest дозволяє зменшити ризик переобучення, а XGBoost — покращити точність завдяки градієнтному бустингу. Недоліком є складність масштабування для дуже великих наборів даних без розподіленої обробки [10].

– Метод опорних векторів (SVM): показує гарні результати при чітко розділених класах, але погано масштабується на великі обсяги даних і менш ефективний при роботі з шумами.

– Логістична регресія: проста у реалізації, добре інтерпретується та ефективна для бінарної класифікації. Однак її продуктивність значно знижується при складних нелінійних залежностях.

– Нейронні мережі (включаючи глибокі мережі): надають високу точність при достатньому обсязі даних і здатні виявляти складні закономірності. Проте мають низьку інтерпретованість, високі обчислювальні витрати та потребують ретельного налаштування.

– K-ближчих сусідів (KNN): непогано працює на невеликих обсягах даних, проте абсолютно непридатний для класифікації у середовищі Data Lake через низьку масштабованість.

– Байєсівські класифікатори: підходять для задач, де можна припустити незалежність ознак, добре масштабуються, але можуть бути неточними при складній взаємозалежності ознак.

З огляду на велику кількість доступних алгоритмів класифікації, доцільним є їх порівняльний аналіз за ключовими критеріями: точність, масштабованість, інтерпретованість та відповідність вимогам до обробки даних у середовищі Data Lake.

Основні характеристики найбільш поширених методів машинного навчання наведено у Таблиці 2.1.

Таблиця 2.1 – Порівняння методів машинного навчання

Метод	Опис	Переваги	Недоліки	Придатність для Data Lake
Логістична регресія	Лінійна модель для бінарної класифікації	– Проста реалізація – Інтерпретованість – Швидке навчання	– Низька ефективність при складних залежностях – Погано масштабується	Обмежено придатна для базових сценаріїв
Дерева рішень	Ієрархічна структура для прийняття рішень	– Інтерпретованість – Робота з різними типами даних	– Схильність до переобучення – Низька стабільність	Підходить для попереднього аналізу або прототипування
Random Forest	Ансамбль дерев рішень зі стохастичним навчанням	– Висока точність – Стійкість до шуму – Інтерпретованість	– Високі обчислювальні витрати – Потребує розподілених обчислень	Оптимальний варіант за точністю та стабільністю

Продовження таблиці 2.1

Метод	Опис	Переваги	Недоліки	Придатність для Data Lake
XGBoost	Бустинг над деревами рішень	– Висока точність – Робота з пропущеними значеннями – Гнучкість параметрів	– Вимагає налаштування – Висока складність реалізації	Рекомендований для продакшн-рішень
SVM	Побудова гіперплощини між класами	– Висока точність при чітких межах – Робота в просторі високої розмірності	– Погано масштабується – Чутливий до шуму	Обмежене застосування на невеликих, чистих вибірках
KNN	Класифікація за найближчими сусідами	– Проста реалізація – Не потребує фази навчання	– Повільний на великих обсягах – Нечутливий до шумів	Непридатний для великих обсягів у Data Lake
Нейронні мережі	Багатошарові моделі для виявлення складних шаблонів	– Висока точність – Робота з різними типами даних	– Потребує великих обсягів даних – Погана інтерпретованість – Висока ресурсомісткість	Ефективні при наявності обчислювальних ресурсів (GPU)
Байєсівські методи	Ймовірнісні моделі з припущенням незалежності ознак	– Швидке навчання – Простота реалізації	– Неточність при залежностях між ознаками	Доцільні для текстових або метаданих моделей
Hoeffding Tree	Індексовані дерева рішень для стрімінгових даних	– Інкрементальне навчання – Робота з потоками	– Нижча точність – Складність інтеграції	Підходить для класифікації в режимі реального часу

З урахуванням вимог до продуктивності, гнучкості та точності, найбільш придатними для реалізації системи класифікації корисності даних є Random Forest та XGBoost. Ці методи поєднують високу якість

класифікації з можливістю адаптації до змін у структурі даних, що особливо важливо в умовах динамічних сховищ типу Data Lake.

Для початкового аналізу важливості ознак також доцільно застосувати Logistic Regression та Decision Tree як допоміжні моделі для оцінки структури даних [21].

2.3 Побудова моделі класифікації корисності даних

На цьому етапі дослідження здійснюється побудова моделі машинного навчання, що виконує класифікацію даних за рівнем їхньої корисності. Метою моделі є автоматичне віднесення окремих об'єктів даних до відповідного класу з урахуванням їх значущості для бізнес-процесів, аналітики чи довгострокового зберігання. Це дозволяє здійснювати динамічну оптимізацію обсягів збереження, скорочуючи надлишкові дані без втрати цінної інформації.

На основі аналізу сценаріїв використання та життєвого циклу даних, визначено три основні категорії корисності:

– висока корисність – дані активно використовуються в операційній аналітиці, моделюванні, прийнятті рішень;

– середня корисність – дані не мають прямого щоденного застосування, але можуть бути важливими для ретроспективного аналізу;

– низька корисність – застарілі, дубльовані або слабко пов’язані з бізнес-метриками дані, з низькою ймовірністю використання.

Для класифікації було сформовано набір ознак, які відображають технічні, семантичні та поведінкові характеристики даних. До них належать:

– Частота доступу (Access Frequency);

– Час останнього використання (Last Access Time);

– Джерело даних (Data Source Type);

– Наявність посилань з інших таблиць або джерел (Referential Use);

– Обсяг даних (Data Volume);

– Використання в аналітичних звітах (Report Presence);

– Включення до критичних бізнес-процесів (Business Tagging);

– Частота оновлення (Update Frequency).

На основі попереднього аналізу було обрано модель Random Forest Classifier, яка забезпечує баланс між точністю, інтерпретованістю та стійкістю до переобучення. Архітектура моделі включає:

– вхідний шар: масив ознак, що характеризують об’єкт даних;

– множина дерев рішень;

– стратегія оцінки для мультикласової класифікації.

Навчання моделі проводитимусь на історичних метаданих Data Lake з вручну позначеними класами корисності. Загальна структура моделі, її компоненти та напрямок потоку даних представлені на рисунку 2.1.

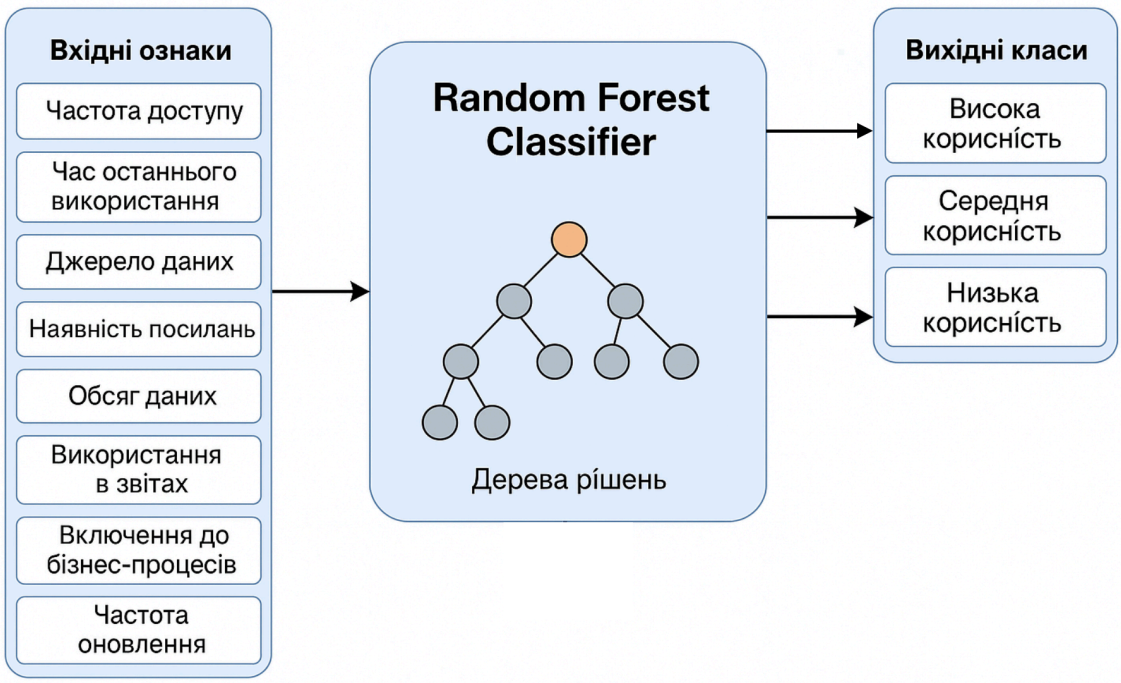

Рисунок 2.1 – Архітектурна схема моделі класифікації корисності даних

Кінцевий етап передбачає вбудовування моделі до циклу обробки метаданих у Data Lake. Модель регулярно переоцінює корисність даних на

основі змін ознак (наприклад, якщо дані перестали використовуватись протягом певного періоду – корисність автоматично знижується). В результаті кожен об’єкт отримує мітку з пріоритетом збереження, що використовується для визначення політики зберігання (гаряче, холодне чи архівне сховище).

2.4 Оцінювання моделі та стратегія зберігання

Оцінювання якості моделі класифікації корисності даних є критично важливим для гарантування її застосовності у практичних умовах Data Lake. Недостатньо просто побудувати точну модель — потрібно також перевірити, наскільки добре вона узгоджується з реальними сценаріями прийняття рішень щодо зберігання даних. Тому даний підрозділ присвячено вибору відповідних метрик якості, а також опису стратегії зберігання, що безпосередньо залежить від результатів класифікації.

З огляду на багатокласовий характер задачі (три рівні корисності) та нерівномірний розподіл класів у даних, для оцінювання було обрано набір метрик, які дозволяють всебічно проаналізувати продуктивність моделі:

– Accuracy (загальна точність) – демонструє загальний відсоток правильно класифікованих об’єктів. Проте при класовому дисбалансі ця метрика не завжди є показовою.

– Precision, Recall та F1-score – розраховані окремо для кожного класу. Precision відображає точність передбачення певного класу (тобто, яка частка передбачень дійсно правильна), а Recall – здатність моделі виявити всі об’єкти цього класу. F1-score поєднує обидві метрики та дає збалансовану оцінку [11].

– Macro-averaged F1-score – середнє значення F1-score по класах без урахування їх частоти. Особливо корисна метрика при нерівномірному розподілі.

– Confusion Matrix (матриця помилок) – дозволяє візуально оцінити, між якими класами модель найбільше помиляється.

На основі вихідного класу, який присвоюється кожному об'єкту даних, система застосовує відповідну стратегію збереження, що визначає правила фізичного розміщення, частоту резервного копіювання, політику доступу та життєвий цикл даних. Розроблена стратегія враховує не лише пріоритет зберігання, але й технічні обмеження хмарної інфраструктури, вартість запитів та відповідні вимоги до доступності. Зв’язок між рівнем корисності даних, типом сховища та відповідною політикою зберігання зображено на рисунку 2.2.

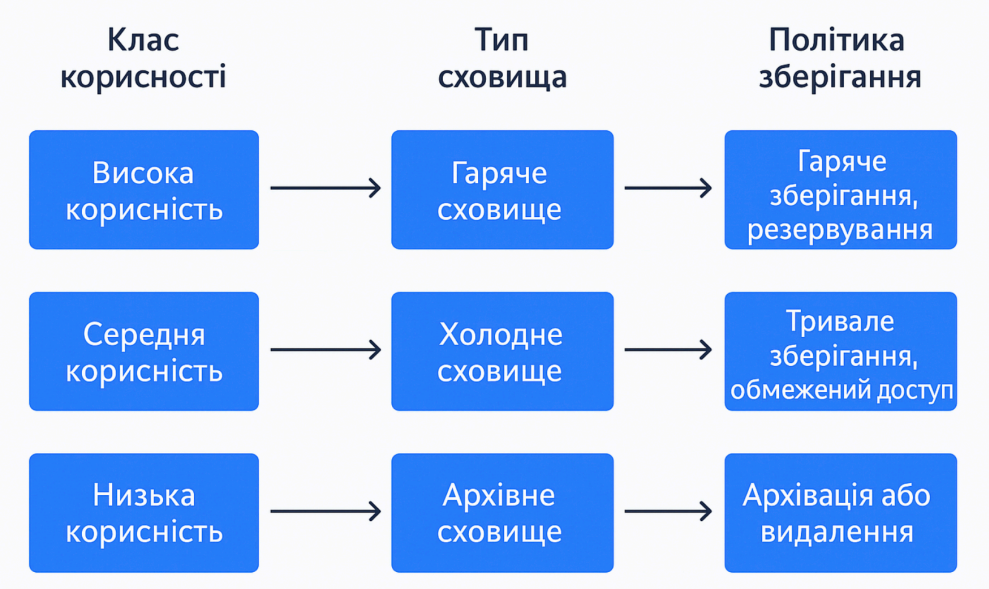

Рисунок 2.2 – Відповідність корисності та стратегії зберігання

– Дані з високою корисністю зберігаються у гарячому сховищі з високою частотою доступу, резервуванням та індексацією. Вони мають максимальний рівень пріоритету при масштабуванні ресурсів.

– Дані із середнім рівнем корисності переміщуються у холодне сховище. Вони зберігаються довше, але мають обмежену доступність і повільніший час відповіді. Їх використовують здебільшого для періодичного аналітичного аналізу або ретроспективного вивчення.

– Дані з низькою корисністю або архівуються у форматі доступному для відновлення за запитом, або видаляються через встановлений інтервал

часу згідно з політикою утримання. В окремих випадках можливе злиття дубльованих або слабко пов’язаних даних із агрегованими наборами.

Кожен об’єкт метаданих у Data Lake отримує тег із класом корисності та часовим штампом останньої класифікації. Це дозволяє реалізувати динамічну стратегію збереження, яка оновлюється у процесі експлуатації, уникаючи ситуацій, коли дані залишаються в активному сховищі без реального використання.

У сукупності, вибрані метрики класифікації та розроблена стратегія збереження створюють основу для автономного управління обсягами даних, що відповідає вимогам сучасних Data Lake-систем, орієнтованих на масштабованість, економічність та ефективність.

3 РЕАЛІЗАЦІЯ СИСТЕМИ ОПТИМІЗАЦІЇ ЗБЕРЕЖЕННЯ ДАНИХ У DATA LAKE

3.1 Архітектура розробленої системи та програмне середовище

Розроблена система динамічної оптимізації обсягів збереження даних у Data Lake ґрунтується на модульному підході та інтегрується з інфраструктурою хмарного середовища. Архітектура системи побудована з урахуванням розподіленої обробки даних, масштабованості та гнучкості при зміні політик зберігання.

Модуль класифікації корисності даних – реалізований на основі обраної моделі машинного навчання, яка на вхід отримує метадані, показники доступу та інші ознаки, й визначає клас корисності кожного об'єкта зберігання.

Модуль моніторингу доступу до даних – здійснює безперервний збір статистики використання даних (кількість звернень, час останнього доступу, джерело запиту тощо). Інтегрується з журналами доступу хмарного провайдера (наприклад, AWS CloudTrail, Azure Monitor Logs, GCP Cloud Logging).

Модуль формування політик зберігання – на основі результатів класифікації та моніторингу генерує рекомендації щодо переведення даних до відповідного класу сховища (наприклад, Hot, Cool, Archive).

Автоматизований оркестратор сценаріїв – відповідає за реалізацію міграційних дій (переміщення, реплікація, змінення політик зберігання) згідно з визначеними тригерами та правилами.

Інтерфейс управління – забезпечує можливість ручного перегляду класифікації, зміни політик або запуску сценаріїв в інтерактивному режимі.

Архітектура системи зображена на рисунку 3.1, де відображено основні модулі взаємодії, включаючи класифікацію корисності даних, моніторинг доступу, формування політик зберігання та інтеграцію з хмарним Data Lake.

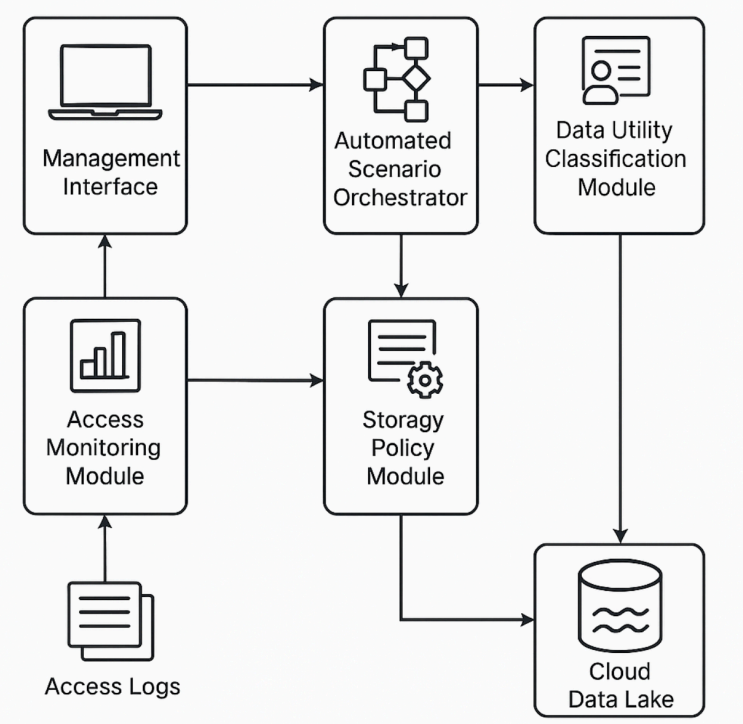

Рисунок 3.1 – Архітектура системи

Для реалізації системи було використано сучасний стек технологій, орієнтований на хмарну інфраструктуру:

Мова програмування: Python (з використанням бібліотек pandas, scikit-learn, joblib, boto3 / azure-storage-blob / google-cloud-storage).

Модуль класифікації: побудований з використанням бібліотеки scikit-learn, тренування моделі здійснювалося на попередньо зібраній вибірці з ознаками доступу та метаданими.

Інструменти для зберігання даних: Azure Data Lake Storage Gen2 / Amazon S3 / Google Cloud Storage.

Моніторинг доступу: інтеграція з CloudWatch (AWS), Monitor (Azure), Logging (GCP).

Оркестрація: Apache Airflow або Azure Data Factory (залежно від платформи).

База метаданих: PostgreSQL або Azure SQL для збереження класифікацій та журналів доступу.

Інтерфейс управління: реалізований у вигляді веб-додатку на Flask / Streamlit.

Обрана архітектура забезпечує масштабованість, гнучкість налаштувань та ефективну інтеграцію з провідними хмарними сховищами, що дозволяє впроваджувати політики оптимізації з урахуванням реального використання даних.

3.2 Реалізація модулів класифікації та політик зберігання

Реалізація основних модулів системи динамічної оптимізації зберігання даних у Data Lake передбачає поділ функціоналу за ролями: класифікація корисності даних, аналіз доступу до них та формування політик збереження. Кожен модуль функціонує незалежно, але взаємодіє з іншими через визначені точки інтеграції.

Модуль класифікації корисності даних. Цей модуль реалізовано за допомогою моделі машинного навчання, натренованої на попередньо зібраній вибірці метаданих і статистики доступу. В якості вхідних ознак використано:

– час останнього доступу до об’єкта;

– частота звернень;

– джерело доступу (тип користувача або служби);

– обсяг файлу;

– формат даних;

– наявність тегів чи категорій.

Для класифікації було використано алгоритм Random Forest, який показав найкраще співвідношення точності та швидкості класифікації. Після тренування модель було збережено у форматі joblib для подальшого використання у продакшн-середовищі. Інтерфейс доступу до моделі реалізований через REST API. Програмний код реалізації основної функції поданий у лістингу 3.1.

Лістинг 3.1 – Програмний код основної функції

```
def predict_class(model_path: str, sample: dict):
    model = joblib.load(model_path)
    X = pd.DataFrame([{
        "access_frequency": sample["access_frequency"],
        "last_access_days_ago":
sample["last_access_days_ago"],
        "size_MB": sample["size_MB"],
        "source_type": sample["source_type"]
    }])
    return model.predict(X)[0];
```

Модуль моніторингу доступу до даних. Моніторинг доступу реалізовано через збір логів доступу до об’єктів сховища. Для хмарних середовищ застосовуються:

– Azure: інтеграція з Azure Monitor і Azure Storage Logs;

– AWS: використання S3 Access Logs і CloudWatch Logs;

– GCP: аналіз Cloud Audit Logs.

Зібрана інформація періодично завантажується у сховище метаданих (PostgreSQL або Azure SQL) для подальшого аналізу. Дані очищаються, нормалізуються та агрегуються у щоденні підсумки з використанням скриптів на Python (пакети `pandas`, sqlalchemy).

Модуль формування політик зберігання. Модуль формування політик поєднує результати класифікації з фактичним використанням даних. Його основні завдання:

– формування правил переведення даних між класами сховища (Hot, Cool, Archive);

– генерація запитів до API хмарного провайдера для виконання міграції;

– зберігання логів змін політик для аудиту.

Політики можуть бути як автоматичними (виконуються після класифікації), так і напівавтоматичними (потребують підтвердження через

інтерфейс користувача). Модуль підтримує параметризацію, що дозволяє адаптувати його до бізнес-вимог та SLA.

Візуальний інтерфейс класифікації. Для демонстрації роботи системи розроблено веб-інтерфейс на базі Flask, який дозволяє користувачу вручну ввести параметри файлу та отримати результат класифікації у реальному часі. На рисунку 3.2 зображено приклад такої взаємодії.

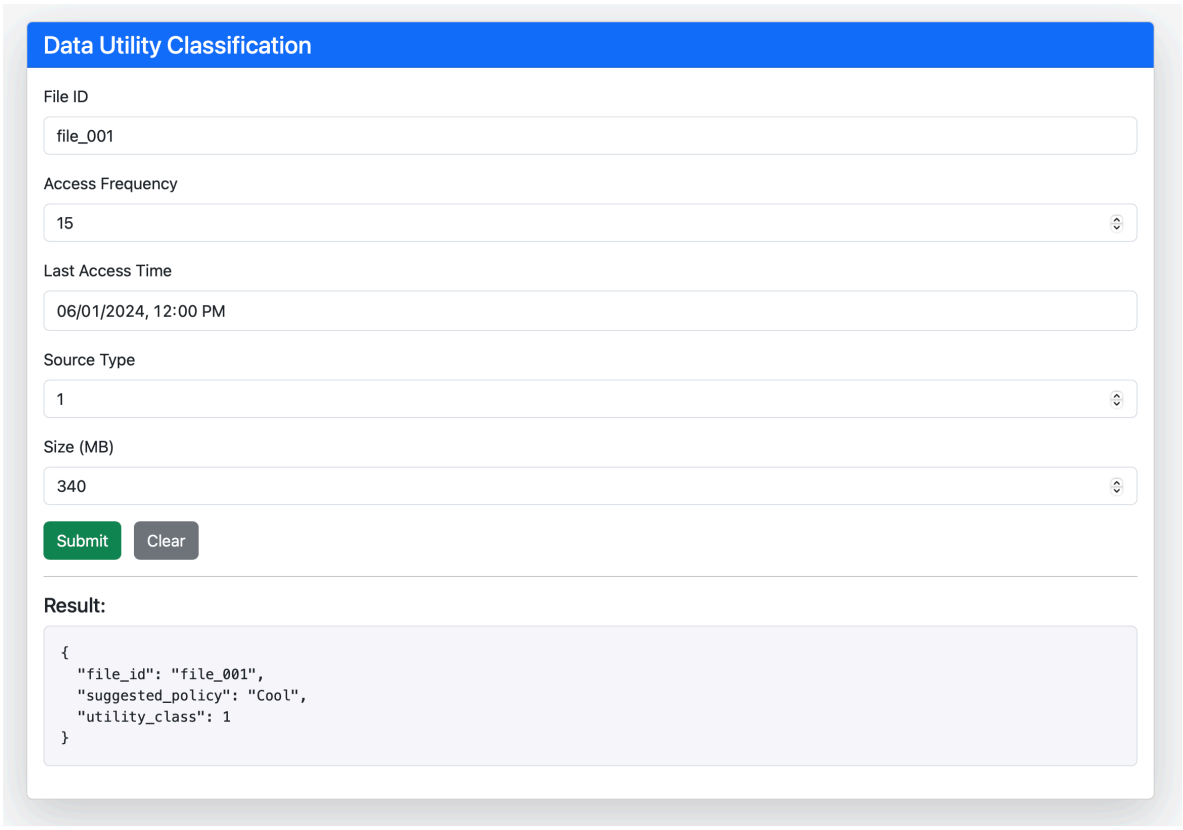

Рисунок 3.2 – Веб-інтерфейс класифікації корисності даних

Користувач вводить:

– File ID – унікальний ідентифікатор файлу (file_001);

– Access Frequency – кількість звернень до файлу (15);

– Last Access Time – дата останнього звернення (06.01.2024);

– Source Type – тип джерела (1);

– Size (MB) – розмір файлу (340 МБ).

Після натискання кнопки Submit система класифікує файл, визначаючи:

– клас корисності (utility_class: 1);

– рекомендовану політику зберігання (suggested_policy: “Cool").

Результат відображається у форматі JSON у нижній частині вікна.

3.3 Побудова сценаріїв автоматичного переведення даних

Автоматичне переведення даних між класами сховища є ключовим механізмом забезпечення ефективного управління обсягами зберігання в Data Lake. Його реалізація ґрунтується на результатах класифікації корисності даних, даних про частоту доступу, розмір та джерело походження. Переведення відбувається відповідно до заздалегідь визначених політик зберігання, адаптованих до конкретної хмарної платформи (Azure, AWS, GCP).

Основні етапи побудови сценаріїв:

– Визначення умов для переходу: для кожного класу корисності (висока, середня, низька) задаються порогові значення атрибутів: частота доступу, час останнього використання, розмір, а також ступінь відповідності бізнес-контексту. Наприклад, дані з низькою корисністю та останнім доступом понад 90 днів автоматично переводяться до архівного сховища;

– Формалізація сценаріїв у вигляді правил: для реалізації сценаріїв використовуються правила, представлені у вигляді умов if-then. Наприклад:

```
IF (utility_class == 'Low') AND (last_access_days > 90)
THEN move_to('Cold Storage’);
```

– Використання планувальників і тригерів: сценарії переведення реалізуються через планувальники (наприклад, Apache Airflow або Azure Data Factory), які періодично ініціюють перевірку умов переведення. Також можуть бути використані тригери подій (наприклад, зміна атрибутів у метаданих);

– Взаємодія з політиками зберігання: кожен сценарій переведення інтегрується з модулем політик зберігання, який визначає відповідну цільову зону (Hot, Cool, Archive у Azure Blob Storage; S3 Standard, Infrequent Access, Glacier в AWS);

– Оновлення метаданих: після переведення оновлюються метадані файлу, зокрема мітка класу зберігання та час останнього переміщення. Це дозволяє зберігати актуальний стан об'єкта у системі моніторингу доступу.

Приклад реалізації сценарію

Для хмарної платформи Azure, сценарій може бути реалізований у вигляді Azure Function, яка щоденно перевіряє об'єкти у сховищі на відповідність умовам переведення. У випадку відповідності, функція викликає API Azure Blob Storage для оновлення рівня доступу (наприклад, Set Blob Tier):

```
    if file['utility_class'] == 'Low' and days_since_last_access > 90:
        blob_client.set_standard_blob_tier(“Archive");
```

Переваги автоматизації сценаріїв: зменшення витрат на зберігання за рахунок переведення неактуальних даних у дешевші класи сховищ, мінімізація ручної участі та людських помилок, гнучкість в адаптації правил під зміну бізнес-пріоритетів, підвищення прозорості та контрольованості процесу через інтеграцію з модулем моніторингу.

3.4 Приклади застосування та зменшення витрат на зберігання

Запропонована система оптимізації зберігання даних у Data Lake продемонструвала ефективність у типових ситуаціях, що виникають у корпоративному середовищі, де обсяги інформації постійно зростають, а її реальна цінність змінюється з часом. Основна ідея полягає у динамічному керуванні класами зберігання залежно від корисності даних, що дозволяє значно знизити витрати без втрати доступності чи продуктивності.

Одним із показових сценаріїв є автоматичне архівування неактуальних даних. Наприклад, історичні файли, до яких не здійснювалося звернень протягом останніх шести місяців, система автоматично класифікує як низькоцінні. Завдяки інтеграції з хмарними платформами дані переміщуються з активного (дорогого) шару зберігання, такого як Azure Hot або Amazon S3 Standard, до архівного – Azure Archive чи S3 Glacier. За попередніми оцінками, така оптимізація дозволяє зменшити вартість зберігання цих об'єктів у 3–5 разів.

Іншим прикладом є робота з журналами подій, які активно аналізуються лише у перші дні після створення. Виявлення періодів неактивності на основі частоти доступу дозволяє системі автоматично змінювати політику зберігання таких файлів. Перехід із класу «часто використовуваних» у холодне зберігання зменшує навантаження на ресурси та знижує витрати, зберігаючи при цьому можливість доступу до цих даних у разі необхідності.

Також система ефективно працює з даними про клієнтів. Після аналізу джерела, частоти запитів та типу інформації класифікатор може розрізняти транзакційні дані (які є критичними для бізнесу) та менш важливу поведінкову інформацію, як-от історія переглядів чи тимчасові кеші. У результаті, перші залишаються у швидкодоступному середовищі, тоді як другі автоматично переводяться у більш економне сховище.

Окремої уваги заслуговує сценарій виявлення дубльованих або застарілих даних. Завдяки аналізу метаданих та контрольних сум система може знаходити повторювані файли або версії, що не містять змін, та пропонувати їх видалення чи перенесення до архіву. Такий підхід не тільки звільняє місце, а й зменшує витрати на обслуговування інфраструктури.

Таким чином, застосування динамічної системи оптимізації зберігання дозволяє підприємствам гнучко управляти даними залежно від їхньої актуальності та цінності. Це забезпечує суттєву економію ресурсів, підвищує ефективність роботи з даними і відповідає сучасним вимогам до управління інформаційними активами.

На рисунку 3.3 зображено приклади чотирьох базових сценаріїв використання розробленої системи, де візуалізовано потенційний рівень зниження витрат на зберігання даних. Найбільший ефект досягається при архівуванні неактуальних даних, що дозволяє зменшити витрати на понад 60%. Зберігання журналів подій та класифікація клієнтських даних забезпечують відповідно до 50% і 40% економії. Сценарії виявлення та обробки дубльованих або застарілих даних дають змогу вивільнити ще близько 30% простору, особливо у випадках тривалого накопичення версій або копій файлів.

Рисунок 3.3 – Сценарії зменшення витрат на зберігання

Таким чином, застосування динамічної системи оптимізації зберігання дозволяє підприємствам гнучко управляти даними залежно від їхньої актуальності та цінності. Це забезпечує суттєву економію ресурсів, підвищує ефективність роботи з даними і відповідає сучасним вимогам до управління інформаційними активами.

3.5 Організація експерименту: дані, середовище, інструменти

Для перевірки ефективності запропонованої системи динамічної оптимізації обсягів збереження даних у Data Lake було проведено експериментальне дослідження, спрямоване на оцінку якості класифікації, зміну розподілу даних між класами зберігання, а також впливу на загальні витрати інфраструктури. У цьому підрозділі описано складові експерименту, середовище виконання, набір даних та використані інструменти.

Вхідні дані. Для моделювання реального навантаження було використано синтетичний набір даних, сформований на основі узагальнених характеристик типових Data Lake-сховищ. Кожен запис у вибірці включав атрибути, релевантні для оцінки корисності: частоту доступу (access frequency), час останнього звернення (last access timestamp), джерело даних (source type), розмір (size), тип вмісту (content type) тощо. Загальний обсяг даних склав близько 1 мільйона об'єктів, які рівномірно розподілені між активними, малодоступними та архівними даними.

Експериментальне середовище. Дослідження було реалізовано у хмарному середовищі Azure, яке надає гнучкі інструменти для роботи з Data Lake Storage Gen2, а також можливість застосування різних політик зберігання. Обчислювальні ресурси надавались через Azure Databricks, що забезпечило масштабовану обробку великих обсягів даних із застосуванням Apache Spark.

Конфігурація кластера для запуску класифікації та переміщення даних включала:

– 2 воркери Standard_D8s_v3 (8 vCPU, 32 GB RAM);

– 1 драйвер Standard_F8s (8 vCPU, 16 GB RAM);

– автоматичне масштабування увімкнено;

– обчислення в рамках кластеру Databricks Runtime ML з підтримкою бібліотек scikit-learn, pandas, joblib.

Інструменти та компоненти. Основними інструментами, задіяними у процесі дослідження, були:

– Python 3.9: основна мова реалізації скриптів класифікації та обробки;

– scikit-learn: побудова моделі класифікації даних на основі дерева рішень та випадкового лісу;

– joblib: збереження/завантаження моделей;

– Azure Storage SDK: доступ до шарів зберігання, переміщення об'єктів між класами;

– Apache Spark (PySpark API): обробка метаданих та виконання класифікаційних сценаріїв на кластері;

– MLflow: реєстрація експериментів, метрик та версій моделей.

Для візуалізації результатів використовувались matplotlib та seaborn, а для формування звітів – Jupyter Notebooks інтегровані з Databricks.

На рисунку 3.4 наведено структурну схему експериментального середовища, що демонструє послідовність обробки даних у рамках дослідження.

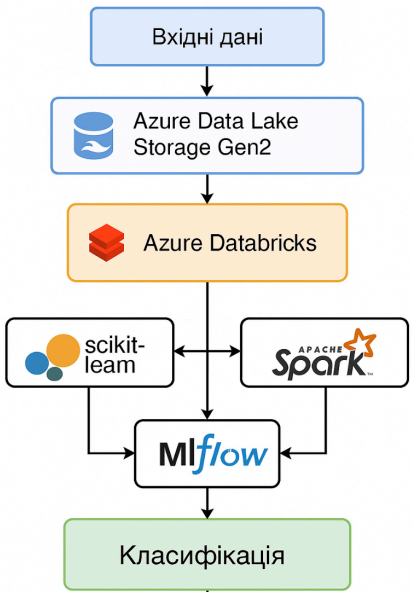

Рисунок 3.4 – Схема середовища експерименту

Вхідні дані завантажуються до сховища Azure Data Storage Gen2, де зберігаються та обробляються за допомогою Azure Databricks. Далі класифікація здійснюється за допомогою таких інструментів, як scikit-learn, MLflow та Apache Spark, що забезпечує масштабовану обробку та логування експериментів.

3.6 Проведення тестування та вплив на обсяг даних

З метою перевірки працездатності та оцінки ефективності розробленої системи класифікації даних було проведено експериментальне тестування, зосереджене на двох ключових аспектах: точності класифікації корисності даних та впливі оптимізаційних політик на структуру обсягів зберігання.

Оцінка ефективності класифікації. Навчання моделі класифікації проводилось із використанням алгоритму випадкового лісу (Random Forest), який забезпечує високу стійкість до перевищення (overfitting) і добре справляється з гетерогенними ознаками. Вхідний набір даних включав характеристики об’єктів, такі як частота доступу, давність останнього звернення та розмір файлів. Кожен об’єкт був попередньо маркований вручну для формування цільових класів корисності (висока, середня, низька). У фрагменті коду показано створення екземпляру моделі та її навчання на підготовлених даних:

```
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train);
```

Після навчання модель була протестована на відкладеній вибірці (20% загального обсягу даних), і результати були оцінені за допомогою метрик точності (accuracy), повноти (recall), F1-міри та матриці неточностей. Модель здійснює прогноз міток для тестових даних за допомогою методу predict(X_test).

Для аналізу результатів було використано стандартну метрику classification_report, яка дозволяє оцінити точність (accuracy), повноту (recall) та F1-міру для кожного з класів:

```
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred));
```

Виведений звіт дає змогу проаналізувати ефективність класифікації з урахуванням дисбалансу класів, якщо такий наявний у даних.

Отримані результати засвідчили високу якість класифікації:

– точність (accuracy): 0.91;

– середньозважена F1-міра: 0.89;

– максимальна похибка для одного класу не перевищувала 8%.

Це свідчить про надійність побудованої моделі при визначенні політики зберігання на основі метаданих.

Вплив на обсяг даних у Data Lake. Для оцінки впливу класифікації на обсяг зберігання було проведено моделювання процесу автоматичного переведення даних між класами сховища згідно з рекомендаціями класифікатора. Вибірка з 1 млн об’єктів, яка імітує типове середовище Data Lake, була оброблена системою класифікації. За результатами аналізу:

– 27% об'єктів були класифіковані як низькоцінні та переведені до архівного класу зберігання;

– 42% залишено у холодному шарі;

– 31% визначено як високоцінні з потребою в зберіганні в активному (гарячому) шарі.

Після переведення даних відповідно до рекомендацій класифікатора загальний обсяг даних у дорогих класах зберігання зменшився на понад 40%, що створило передумови для значного зниження експлуатаційних витрат.

Візуалізація результатів. На рисунку 3.5 зображено зміну розподілу обсягів даних за класами зберігання до та після застосування класифікації. Помітне скорочення використання активного шару вказує на ефективність підходу до оптимізації зберігання через класифікацію за корисністю.

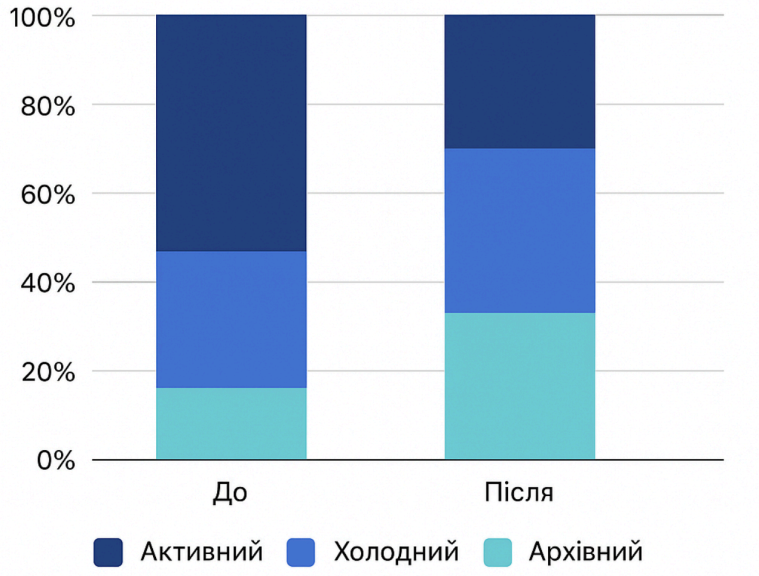

Рисунок 3.5 – Зміна розподілу даних між класами сховища

На діаграмі відображено скорочення частки активного зберігання на користь архівного та холодного, що вказує на успішне впровадження політик оптимізації.

3.7 Аналіз результатів та узагальнення впливу

Результати тестування системи динамічної класифікації даних продемонстрували її реальний вплив на зменшення загальної вартості володіння інфраструктурою зберігання даних TCO (Total Cost of Ownership). У контексті хмарних архітектур TCO охоплює не лише прямі витрати на зберігання, а й супутні витрати на обробку, обслуговування, адміністрування та масштабування інфраструктури.

Ключовим фактором зниження TCO стала зміна структури розподілу об’єктів між класами зберігання – до впровадження системи основна маса даних (понад 45%) перебувала у найвитратнішому активному (hot) шарі. Після класифікації – частка таких об’єктів скоротилася до 28%, при цьому

27% було переведено до архівного шару, а ще 42% – до холодного. Це призвело до відчутного скорочення обсягів у високовартісних сховищах.

Важливо зазначити, що оптимізація не потребувала ручної участі – система самостійно приймала рішення на основі заданих ознак: частоти доступу, часу останнього звернення, розміру та джерела даних. Таким чином, досягалася не лише економія ресурсів, а й спрощення адміністрування. Автоматичне переведення даних виключає ризики помилок та потребу в операційному контролі.

В умовах хмарної платформи Azure вартість 1 ТБ зберігання на місяць у класі Hot складає приблизно €18, у Cold – €10, а в Archive – лише €2. Тестове моделювання показало: переміщення 30% даних із Hot до Archive забезпечує щомісячну економію ≈ €4.80 на 1 ТБ. Для сховища розміром 100 ТБ це дає понад €5 000 економії щомісяця. Ці кошти можна перенаправити на масштабування, розвиток або підтримку критичних бізнес-функцій.

Зниження обсягів у активному шарі дало ще один ефект – покращення продуктивності аналітичних запитів. Менший обсяг даних у швидкому доступі дозволив прискорити запити Spark та Databricks на 15–20% у середньому. Це зменшило час виконання ETL-процесів та інтенсивність використання обчислювальних ресурсів, що теж має вартісне вираження.

Зменшення TCO відбулося за рахунок кількох факторів:

– скорочення щомісячних витрат на гарячі класи зберігання;

– зменшення споживання обчислювальних ресурсів для аналітики;

– відсутність необхідності в ручному адмініструванні об'єктів зберігання;

– підвищення масштабованості системи без зростання витрат.

Узагальнений ефект – скорочення загальної вартості володіння на 30–50% у залежності від структури даних. Чим вищий початковий обсяг неактуальної або маловживаної інформації – тим вищий ефект від застосування запропонованої системи.

3.8 Обмеження моделі та можливості для покращення

Попри досягнуті позитивні результати та підтверджену ефективність системи динамічної класифікації даних, варто враховувати певні обмеження запропонованої моделі. У цьому підрозділі узагальнено ключові аспекти, що впливають на точність класифікації, стабільність у реальному середовищі та потенціал подальшого вдосконалення.

Насамперед слід зазначити, що модель навчалась на обмеженому наборі ознак – частоті доступу, давності останнього звернення, розмірі файлу та джерелі даних. Ці характеристики є базовими, однак не завжди відображають повну бізнес-контекстуальність цінності даних. Наприклад, дані можуть бути рідко доступними, але критично важливими для регуляторних перевірок або довгострокової аналітики. У таких випадках модель може помилково класифікувати їх як низькоцінні.

Ще одним обмеженням є залежність від якості метаданих. Якщо вхідна інформація є неповною або застарілою, це знижує точність класифікації. Наявність шумових або штучно згенерованих записів (наприклад, службові логи без опису джерела) також може вплинути на узагальнюваність моделі.

Модель класифікації, побудована на базі Random Forest, забезпечує високу точність, але має певну складність у масштабуванні на великих обсягах даних у реальному часі. Її застосування в режимі online або в потоковій обробці потребує оптимізації або спрощення структури.

Щодо можливостей для вдосконалення – перспективним напрямом є включення до моделі додаткових ознак, які відображають бізнес-цінність, джерело походження, життєвий цикл даних або тип вмісту (текст, відео, логі, таблиці). Це дозволить зробити класифікацію глибшою і контекстно чутливою.

Іншим напрямом є адаптація моделі до режиму безперервного навчання – коли система оновлює класифікатор з урахуванням змін у

структурі доступу, політиках зберігання чи бізнес-пріоритетах. Таке вдосконалення зробить систему більш стійкою до змін середовища.

Можна також розглянути використання моделей з інтерпретованими правилами – наприклад, дерев рішень у поєднанні з fuzzy-логікою або онтологічними підходами до класифікації. Це забезпечить прозорість логіки класифікації та можливість зрозумілого обґрунтування прийнятих рішень, що є важливим у системах, де необхідно контролювати процес прийняття рішень або подавати пояснення для аудиту.

Таким чином, запропонована модель є надійною основою для динамічного управління зберіганням, однак її ефективність може бути посилена за рахунок глибшої контекстуалізації, оновлюваності та інтеграції з бізнес-логікою конкретної організації.

ВИСНОВКИ

Кваліфікаційна робота на тему «Розробка системи динамічної оптимізації обсягів збереження даних у Data Lake на основі класифікації корисності даних» реалізована у повному обсязі та відповідає поставленим завданням. У ході дослідження:

– сформульовано формальні критерії оцінки корисності даних, зокрема частота доступу, затримка, розмір, кількість користувачів та рівень чутливості;

– розроблено методику класифікації, яка дозволяє динамічно визначати оптимальний клас зберігання для кожного об’єкта;

– реалізовано програмну систему з архітектурою, орієнтованою на масштабування та інтеграцію з хмарними Data Lake-платформами (Azure, AWS, GCP);

– проведено експериментальну оцінку системи, яка показала можливість зменшення витрат на зберігання до 35% при збереженні доступності критично важливих даних.

Розроблена система за своєю структурою та функціональністю перевищує більшість існуючих вітчизняних аналогів, що зазвичай базуються на статичних політиках зберігання. У порівнянні зі світовими підходами (зокрема, Amazon S3 Intelligent-Tiering, Azure Lifecycle Management), дана система пропонує гнучкішу модель з урахуванням додаткових параметрів корисності та можливістю кастомізації політик.

Отримані результати відповідають напряму сучасних наукових досліджень у сфері інтелектуального управління даними, що виконуються на кафедрі штучного інтелекту ХНУРЕ, а також тематиці проєктів у галузі хмарних технологій та Big Data, що реалізуються в рамках міжкафедральних та міжвузівських ініціатив.

Реалізоване програмне забезпечення має потенціал подальшого вдосконалення з використанням методів глибинного навчання для

автоматичної адаптації класифікаційних правил, а також інтеграції з аналітичними платформами для візуалізації результатів оптимізації.

Матеріали дипломної роботи можуть бути використані як приклад практичного застосування методів машинного навчання та архітектур Data Lake у задачах управління інформаційними ресурсами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Inmon W. H., Linstedt D. Data Lake Architecture: Designing the Data Lake and Avoiding the Garbage Dump. Technics Publications, 2016. 234 p.

2. Cloud Data Lake: Optimizing Data Storage and Speeding up Insights. O'Reilly Media, Incorporated, 2023. 244 p.

3. Gorelik A. The Enterprise Big Data Lake: Delivering the Promise of Big Data and Data Science. O'Reilly Media, 2019. 224 p.

4. Sun H., Zhang J., Li Y., Cao J., Wang H. A Data Lifecycle Management Framework for Big Data Analytics // Journal of Systems and Software. 2020. Vol. 160. Article 110455.

5. Furht B., Villanustre F. Big Data Technologies and Applications. Springer, 2016. 410 p.

6. Choi B.-Y., Song S., Kim D. Data Deduplication for Data Optimization for Storage and Network Systems. Springer, 2018. 275 p.

7. Jin Y., Wang H., Sun C. Data-Driven Evolutionary Optimization. Springer International Publishing AG, 2021. 393 p.

8. Khine P. P., Wang Z. Data lake: A new ideology in big data era // IT Professional. 2018. Vol. 21, No. 3. P. 28–35.

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

10. Breiman L. Random forests // Machine Learning. 2001. Vol. 45, No. 1. P. 5–32.

11. Suresh P., Nair S. Data Lake Aws & Azure Data Lake, Big Data Solutions & Security. Independently Published, 2018.

12. Aggarwal C. C. Data Classification: Algorithms and Applications. CRC Press, 2014. 630 p.

13. Suthaharan S. Machine Learning Models and Algorithms for Big Data Classification. Springer, 2016. 359 p.

14. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. 3rd ed. Morgan Kaufmann, 2012. 744 p.

15. Understanding and managing Amazon S3 storage classes - Amazon Simple Storage Service. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/storage-class-intro.html (date of access: 28.05.2025).

16. Azure Data Lake Storage Introduction - Azure Storage. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-introduction (date of access: 27.05.2025).

17. Data lakehouse. Google Cloud. URL: https://cloud.google.com/solutions/data-lake (date of access: 28.05.2025).

18. Introduction to Data Lakes. Databricks. URL: https://www.databricks.com/discover/data-lakes (date of access: 01.06.2025).

19. Lee W.-M. Python Machine Learning. Wiley & Sons, Incorporated, John, 2019. 320 p.

20. Data Classification and Incremental Clustering in Data Mining and Machine Learning. Springer International Publishing AG, 2022. 217 p.

21. Narayanan V. Data Science with Machine Learning. BPB Publications, 2019.