Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Застосування методів машинного навчання
при обробці великих даних
(тема)

Виконав:
студент 2 курсу, групи ПМм-21-1
Башкатов Є.О.
(прізвище, ініціали)
Спеціальність
113 Прикладна математика
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Прикладна математика
(повна назва освітньої програми)
Керівник проф. Кіріченко Л.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ _____ Сидоров М.В.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту
Кафедра _прикладної математики
Рівень вищої освіти _другий (магістерський)
Спеціальність _113 Прикладна математика
(код і повна назва)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Прикладна математика
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
“ 07 ” листопада 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _Башкатову Євгену Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи _Застосування методів машинного навчання при обробці великих даних

затверджена наказом по університету від 25 жовтня 2022 р. № 1412 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 7 грудня 2022 р.

3. Вихідні дані до роботи _інформаційні потоки даних з пристроїв IoT, заражених хробаками Mirai та BASHLITE

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області
2. Вибір і обґрунтування методу розв’язання
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Аналіз предметної області
4. Метод чисельного аналізу
5. Результати обчислювального експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	7 – 13 листопада 2022 р.	виконано
2	Вибір та обґрунтування методу	14 – 20 листопада 2022 р.	виконано
3	Розробка алгоритму і програми	21 – 27 листопада 2022 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	28 листопада – 4 грудня 2022 р.	виконано
5	Робота над текстом пояснювальної записки	28 листопада – 6 грудня 2022 р.	виконано
6	Представлення роботи на рецензію в ЕК	7 грудня 2022 р.	виконано

Дата видачі завдання _7_ листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Кіріченко Л.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 52 с., 19 рис., 3 дод., 31 джерел.

AMAZON S3, APACHE AIRFLOW, AWS GLUE, БАГАТОКЛАСОВА КЛАСИФІКАЦІЯ, ВЕЛИКІ ДАНІ, МАШИНЕ НАВЧАННЯ, ХМАРНІ СЕРВІСИ, ХМАРНІ СХОВИЩА, ІНТЕРНЕТ АТАКИ

Об’єкт дослідження – існуючі хмарні сервіси AWS та методи обробки і зберігання великих даних.

Мета роботи – дослідження і побудова ефективної інфраструктури з сервісів Amazon, яка дозволяє зберігати та обробляти великі об’єми даних на прикладі класифікації даних методами машинного навчання.

Методи дослідження – побудова та навчання нейронних мереж, методи обробки та зберігання великих даних.

В результаті кваліфікаційної роботи були розглянуті нові рішення для обробки та зберігання великих даних на основі сервісів AWS та запропоновано ефективну архітектуру, що дозволяє працювати з великими об’ємами даних. Подана в роботі архітектура може бути використана в будь-якій сфері, де використовуються великі дані, зокрема в eCommerce, Sales, Supply Chain, Delivery або Health Care бізнес-проектах. Показана архітектура, завдяки використанню хмарних технологій, легко розширюється, що дозволяє покрити усі потреби бізнесу та користувачів.

ABSTRACT

Introductory note: 52 pages, 19 figures, 3 appendixes, 31 sources.

APACHE AIRFLOW, AWS GLUE, AMAZON S3, BIG DATA, CLOUD SERVICES, CLOUD STORAGE, INTERNET ATTACKS, MULTI-CLASS CLASSIFICATION, MACHINE LEARNING,

The object of research – existing AWS cloud services and methods of processing and storing big data.

Purpose of work – to research and build an effective infrastructure from Amazon services that allows storing and processing large volumes of data using the example of data classification using machine learning methods.

Methods of research – construction and training of neural networks, methods of processing and storing big data.

As a result of the qualification work, new solutions for processing and storing big data based on AWS services were considered and an effective architecture was proposed, which allows working with large volumes of data. The architecture presented in the work can be used in any area where big data is used, in particular in eCommerce, Sales, Supply Chain, Delivery, or Health Care business projects. The presented architecture, because of the use of cloud technologies, is easily expandable, which allows for covering all the needs of businesses and users.

ЗМІСТ

ВСТУП

Актуальність теми. Щодня ми створюємо величезну кількість даних, знаємо ми про це чи ні. Кожен клік в Інтернеті, кожна банківська операція, кожне відео, яке ми дивимося на YouTube, кожен електронний лист, який ми надсилаємо, кожен лайк на нашій публікації в Instagram – це дані для технологічних компаній.

Оскільки збирається така величезна кількість даних, компаніям має сенс використовувати ці дані, щоб краще зрозуміти своїх клієнтів та їх поведінку. Це причина, чому за останні кілька років популярність Data Science зросла в рази.

Big Data дозволяє компаніям вирішувати проблеми, з якими вони стикаються у своєму бізнесі, і ефективно вирішувати ці проблеми за допомогою Big Data Analytics. Компанії намагаються визначити закономірності та витягти ідеї з цього моря даних, щоб на їх основі можна було діяти для вирішення наявної проблеми.

Хоча компанії десятиліттями збирали величезну кількість даних, концепція великих даних набула популярності лише на початку-середині 2000-х років. Корпорації усвідомили кількість даних, які збираються щодня, і важливість ефективного використання цих даних.

Комп’ютерні інструменти для отримання знань і розуміння з великої кількості неструктурованих даних епохи Інтернету швидко набувають популярності. На передньому плані знаходяться технології штучного інтелекту, які швидко розвиваються, такі як обробка природної мови, розпізнавання образів і машинне навчання.

Ці технології штучного інтелекту можна застосовувати в багатьох сферах. Наприклад, пошуковий і рекламний бізнес Google і його експериментальні роботи-автомобілі, які пройшли тисячі миль каліфорнійськими дорогами, використовують набір трюків штучного інтелекту. Обидва виклики викликають великі дані, аналізуючи величезну кількість даних і миттєво приймаючи рішення.

Мета і завдання кваліфікаційної роботи. Метою роботи є дослідження і побудова ефективної інфраструктури з сервісів Amazon, яка дозволяє збергігати та обробляти великі об’єми даних на прикладі класифікації даних методами машинного навчання. Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд і аналіз сучасного стану задачі зберігання та обробки великих даних;

– створити хмарне сховище;

– розробити модель нейронної мережі для багатокласової класифікації вхідних потоків даних.

Для цього будемо використовувати набір данних, в якому є трафіки пакетів від заражених IoT пристроїв. У професійній практиці використовуються методи обробки великих даних для подального їх використання для навчання нейронної мережі

Об’єктом дослідження є існуючі хмарні сервіси AWS та створення ефективної архітектури для обробки великих даних, використовуючи їх для створення моделі класифікатора.

Предметом дослідження є ефективність орієнтованих на роботу з великими даними сервісів AWS та їх комунікація з існуючими модулями для навчання нейронних мереж.

Методи дослідження. У кваліфікаційній роботі використовуються методи побудови та навчання нейронних мереж, ефективний набір сервісів AWS, які створюють цілісний процес обробки та зберігання великих даних.

Публікації. Результати, отримані у кваліфікаційній роботі, було предствлено на XVI Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (м. Дніпро, 14-15 грудня, 2022 року) [31].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Великі дані

Великі дані — це загальний термін для нетрадиційних стратегій і технологій, необхідних для збору, організації, обробки та збору інформації з великих наборів даних. Хоча проблема роботи з даними, які перевищують обчислювальну потужність або сховище одного комп'ютера, не нова, поширеність, масштаб і цінність цього типу обчислень значно розширилися за останні роки [1].

Основні вимоги для роботи з великими даними такі ж, як і для роботи з наборами даних будь-якого розміру. Однак величезний масштаб, швидкість прийому та обробки, а також характеристики даних, з якими потрібно мати справу на кожному етапі процесу, створюють значні нові проблеми при розробці рішень. Метою більшості систем великих даних є виявлення інформації та зв'язків із великими обсягами різнорідних даних, які були б неможливі за допомогою звичайних методів. Big data використовує правило декількох V.

Обсяг (Volume). Саме вирішення проблем зберігання великих даних може бути серйозною справою для багатьох організацій. У сучасному світі компанії нерідко обробляють терабайти, петабайти або навіть ексабайти даних щодня.

Швидкість (Velocity). Значна частина цих даних не є просто статичною та неактивною. У багатьох системах великих даних дані генеруються, перетворюються та аналізуються з високою швидкістю. Деякі програми для великих даних вимагають надзвичайно високої швидкості обробки та аналізу, де секунди або мілісекунди важливі для того, щоб не відставати від вхідних даних.

Різноманітність (Variety). Великі дані доступні в різних структурованих, неструктурованих і напівструктурованих форматах. Окрім даних електронних таблиць і транзакцій, середовищ великих даних нерідко включають відео, зображення, текст, документи, дані датчиків, файли журналів та інші типи даних.

Правдивість (Veracity). Оскільки великі дані зазвичай збираються з різних джерел і в різних формах, якість даних також відрізняється. Правдивість означає точність і достовірність даних. Успішне вирішення проблем правдивості даних вимагає очищення даних для видалення дублікатів записів, виправлення помилок і невідповідностей, зменшення шуму та усунення інших невідповідностей.

Термін дії (Validity). Це ґрунтується на концепції достовірності, зосереджуючись на тому, як застосовувати набори великих даних у різних випадках використання. Те, що дані були згенеровані для однієї програми, не означає, що їх можна застосувати до іншої. Ефективний аналіз даних залежить від визначення правильних даних, щоб уникнути недійсних висновків і думок. Так само старі дані можуть більше не бути актуальними.

Візуалізація (Visualization). Коли люди дивляться на велику кількість даних на екрані, у людей часто виблискують очі. Візуалізація великих обсягів складних даних за допомогою діаграм, графіків, теплових карт та інших типів візуалізації даних є ефективним способом передачі розуміння, знайденого в даних.

Значення (Value). Зрештою, вам потрібно отримати від ваших даних бізнес-цінність. Якщо ви робите всю роботу — і витрачаєте всі гроші — на збір, зберігання, обробку та аналіз наборів великих даних, ви хочете бути впевнені, що ваша організація отримує очікувані переваги, а не просто накопичує дані.

Аналітика великих даних – це загальний процес дослідження та аналізу наборів великих даних. Він включає в себе такі дисципліни, як аналіз даних, прогнозне моделювання, статистичний аналіз і машинне навчання. Наріжний камінь сучасних додатків штучного інтелекту, машинне навчання надає значну цінність організаціям, отримуючи розуміння більш високого рівня з великих даних, ніж інші типи аналітики.

Розвиток великих даних безпосередньо пов’язаний із розвитком хмарної архітектури. Мережеві системи просто не могли забезпечити обсяг роботи, необхідний для розширеної аналітики та машинного навчання. Але з хмарними обчисленнями та пов’язаними технологіями ми побачили зростання штучного інтелекту та машинного навчання як практичних частин сучасної економіки.

Серед головних досягнень в області великих даних можна виділити декілька основних напрямків.

Автоматизація. Хмарні платформи підтримують автоматизовану обробку даних, що звільняє адміністраторів від безпосереднього керування введенням даних та потоками інформації. Перехід до включення в хмарні обчислення спеціалістів з автоматизації та даних експоненціально підвищив ефективність, ефективність і точність систем даних у хмарі.

Розподілені середовища. мережеві системи, на перший погляд, неефективні та залежать від конкретних технологій, які часто слугують перешкодами для продуктивності. Розподілені хмарні середовища, однак, усунули вузькі місця та накопичувачі даних як принцип їхнього дизайну, тому продуктивність і масштабованість є першочерговими. Великі хмарні середовища підтримують дедалі більші та складніші системи обробки даних.

Високопродуктивні обчислення. Хмарна технологія призвела до переосмислення того, що означає підтримка високопродуктивних обчислень (HPC). Сучасні програми HPC-систем, які використовують оптимізоване апаратне та програмне забезпечення, автоматизовану обробку й організацію даних, а також миттєве масштабування, забезпечили машинне навчання та аналітику значно більшим за те, що ми бачили навіть 15–20 років тому.

1.2 Змістовна постановка задачі

Об’єктом дослідження є набір даних з мережі ботів, яка була використана для DDoS-атак. Цей набір даних містить реальні дані трафіку, зібрані з 9 комерційних пристроїв, автентично заражених хробаками Mirai і BASHLITE. Даний набір буде будемо використовувати для класифікації потоків даних за допомогою нейронної мережі.

Набір даних був створений за допомогою агрегацій потоків, містить статистичні дані, що підсумовують останній трафік від хосту пакету (IP); статисти-

чні дані, що підсумовують останній трафік, що йде від хоста цього пакета (IP) до хосту призначення пакета; статистичні дані, що підсумовують останній трафік, що йде від хоста плюс порту (IP) цього пакета до хоста плюс порту призначення пакета; статистичні дані, що підсумовують тремтіння трафіку, що йде від хоста цього пакета (IP) до хосту призначення пакета. Усі дані мають декілька часових вікон. Вхідний набір даних має 115 атрибутів та 7062606 записів.

Кожен з заражених пристроїв генерує шість різних інформаційних потоків:

– нормальний інформаційний потік, мітка класу 0;

– атаку на TCP протокол, мітка класу 1;

– атаку типу Junk, мітка класу 2;

– атаку типу Scan, мітка класу 3;

– атаку типу Udp-flood, мітка класу 4;

– комбіновану атаку, мітка класу 5.

Атака на TCP протокол здійснюється за допомогою функції TCP-reset. Кожен TCP-пакет у межах з'єднання несе заголовок. У кожному з них є біт прапора скидання (RST). У більшості пакетів цей біт встановлений в 0 і нічого не означає, але якщо він встановлений в 1, це означає, що одержувач повинен негайно припинити використовувати дане з'єднання: не надсилати пакетів з поточним ідентифікатором (на поточний порт), а також ігнорувати всі наступні пакети цього з'єднання (згідно з інформацією в їх заголовках). Насправді, скидання TCP моментально розриває з'єднання. Третій комп'ютер міг відстежувати TCP-пакети цього з'єднання і підробити пакет з прапором скидання і відправити одному або обом учасникам від імені іншого. Інформація в заголовках повинна зазначати, що пакет отриманий нібито з іншого боку, а не від нападника. Така інформація включає IP-адреси та номери портів і повинна містити достатньо правдоподібні дані, щоб змусити учасників перервати з'єднання. Правильно сформовані підроблені пакети можуть бути надійним способом порушити будь-яке TCP-з'єднання, доступне для відстеження нападником.

Атака типу Junk – це відправка недійсних та смітникових запросів на сервер. Клієнт надсилає деякий бінарний смітник на HTTP-сервер на атакований

порт і сервер відповідає помилкою HTTP 400. Недійсні HTTP-запити не запускають системи захисту від DDoS атак і не розпізнаються як напад. Велика кількість смітникових запросів навантажує сервер помилками з кодом 400, забираючи ресурси від корисного функціоналу на функціонал обробки помилок.

Атака сканування: зловмисники сканують пристрої в мережі, щоб зібрати інформацію про оточення цих пристроїв, перш ніж почати складні атаки, щоб підірвати безпеку усієї мережі. Методи сканування, які зазвичай використовуються для збору інформації про комп'ютерну мережу, включають сканування IP-адрес, сканування портів і сканування версій. Окрім звичайних методів сканування IP-адрес і портів (наприклад, протокол визначення адреси та сканування TCP SYN) для збору IP-адрес серверів і відкритих портів, зловмисники також можуть здійснювати атаки сканування сегментів.

UDP-flood – мережна атака типу «відмова в обслуговуванні», яка використовує безсеансовий режим протоколу UDP. Полягає у відправленні безлічі UDP-пакетів (як правило, великого об'єму) на певні або випадкові номери портів віддаленого хоста, який для кожного отриманого пакета повинен визначити відповідний додаток, переконатися у відсутності його активності і відправити відповідне повідомлення ICMP «адресат недоступний». У результаті атакована система виявиться перевантаженою: у протоколі UDP механізм запобігання навантаженням відсутня, тому після початку атаки паразитний трафік швидко захопить всю доступну смугу пропускання, і корисному трафіку залишиться лише мала її частина.

Підмінивши IP-адреси джерел у UDP-пакетах, зловмисник може перенаправити потік ICMP-відповідей і цим зберегти працездатність атакуючих хостів, і навіть забезпечити їх анонімність.

1.3 Формальна постановка задачі

Задано множина об'єктів X, множина допустимих відповідей Y, та існує цільова функція $y^* : X \to Y$, значення якої $y_i = y^*\left(x_i\right)$ відомі лише на кінцевій

підмножині об'єктів $\{x_1,...,x_l\} \subset X$. Пари «об'єкт-відповідь» (x_i, y_i) називається прецедентами. Сукупність пар $X^l = (x_i, y_i)_{i=1}^l$ називається навчальною вибіркою.

Задача навчання прецедентами полягає в тому, щоб по вибірці X^l відновити залежність y^*, тобто побудувати вирішальну функцію $a: X \to Y$, котра наближувала б цільову функцію $y^*(x)$, причому не лише на об'єктах навчальної вибірки, але і на всій множині X.

Ознака f об'єкта x – це результат виміряння деякої характеристики об'єкта. Формально ознакою називається відображення $f: X \to D_f$, де D_f – множина допустимих значень ознаки. Зокрема, будь-який алгоритм $a: X \to Y$ також можна розглядати як ознаку.

В залежності від природи множини D_f ознаки діляться на декілька типів:

– якщо $D_f = \{0,1\}$, тоді f – бінарна ознака;

– якщо D_f - кінцева множина, тоді f – номінальна ознака;

– якщо D_f - кінцева упорядкована множина, тоді f – порядкова ознака;

– якщо $D_f = R$, тоді f – кількісна ознака.

Якщо всі ознаки мають однаковий тип, $D_{f_1} = \cdots = D_{f_n}$, тоді вихідні дані називається однорідними, в іншому випадку – різнорідними.

Нехай є набір ознак $f_1,...,f_n$. Вектор $(f_1(x),...,f_n(x))$ називають ознаковим описом об'єкта $x \in X$. В подальшому ми не будемо розрізняти об'єкти із X та їх ознакові описи, покладаючи $X = D_{f_1} \times ... \times D_{f_n}$. Сукупність ознакових описів всіх об'єктів вибірки X^l, записану у виді таблиці розміром $l \times n$, називають матрицею об'єктів-ознак

$$F = \left\| f_j(x_i) \right\|_{l \times n} = \begin{pmatrix} f_1(x_1) & \cdots & f_n(x_1) \\ \vdots & \ddots & \vdots \\ f_1(x_l) & \cdots & f_n(x_l) \end{pmatrix}.$$

В залежності від природи множини допустимих відповідей, Y задачі навчання по прецедентам діляться на декілька типів.

Якщо $Y = \{1,...,M\}$, то це задача класифікації на M непересічних класів. В цьому випадку вся множина об'єктів X розбивається на класи $K_y = \{x \in X : y^*(x) = y\}$, та алгоритм $a(x)$ має давати відповідь на запитання «до якого класу належить x?». У деяких прикладах класи називають образами та говорять про задачу розпізнавання образів.

Якщо $Y = \{0,1\}^M$, то це задачі класифікації на M пересічних класів. В простішому випадку ця задача зводиться до розв'язання M незалежних задач класифікації з двома непересічними класами.

В формальному виді постановка задачі класифікації має такий вигляд. Нехай $x_i \in X, i = \overline{1,n}$ – множина об'єктів ознак, входів моделі, $y_i \in Y, i = \overline{1,n}$ – множина об'єктів відповідей, виходів моделі. Пара $(x_i, y_i) \in X \times Y$ називається розмічений об'єкт, або прецедент. Кінцева множина $\{x_i\}, i = \overline{1,n}$ представляє собою матрицю $\{x_{i,j}\}, i = \overline{1,\mathrm{n}}, j = \overline{1,m}$ розміром $n \times m$, де рядок матриці – це масив ознак одного об'єкта, $\{y_i\}, i = \overline{1,n}$ – вектор відповідей, елемент якого є значення номеру класу. Комбінація $\{x_i\}, i = \overline{1,n}$ та $\{y_i\}, i = \overline{1,n}$ називається навчальною вибіркою. Задача класифікації полягає у визначенні функції залежності $f : X \to Y$ котра пророкує по $x \in X$ відповіді $y \in Y$.

В множину ознак входять агреговані в різних проміжках часу вага потоку (можна розглядати як кількість елементів, що спостерігаються у дельті часу), середнє, дисперсія, радіус (корінь квадрата суми дисперсій двох потоків), магнітуда (сума коренів квадратів середніх значень двох потоків), наближена коваріація між двома потоками та наближений коефіцієнт Пірсона. В множину об'єктів відповідей входять усі шість міток класів.

В якості класифікатора будемо використовувати нейрону мережу, яка буде вирішувати задачу багатокласової класифікації за допомогою методів обробки та зберігання великих даних і з використанням сервісов Amazon.

1.4 Постановка задач дослідження

Метою роботи є застосування методів машинного навчання при обробці великих данних. Для досягнення поставленої мети треба виконати наступні задачі:

– провести огляд і аналіз сучасного стану задачі зберігання та обробки великих даних;

– вивчити проблеми великих даних;

– побудувати хмарне сховище для зберігання вхідних та вихідних потоків даних;

– створити хмарний обчислювальний кластер для роботи з данними;

– розробити модель нейронної мережі для багатокласової класифікації вхідних потоків даних.

– навчити на оброблених даних багатокласову нейронну мережу;

– провести якісний аналіз створеної моделі.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МЕТОДІВ ТА ТЕХНОЛОГІЙ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧ КЛАСИФІКАЦІЇ У ВЕЛИКИХ ДАНИХ

2.1 Використання хмарних сховищ для зберігання великих об'ємів даних

Для навчання якісної моделі потрібно багато даних. Вони займають значне місце в локальних сховищах, таких як жорсткі диски та флеш-носії. Ця проблема стає більшою, коли мова йде про «важкі» формати даних, наприклад картинки, відео та аудіо-ряди. Тоді використання стандартних носіїв інформації не є цілковито доречним, бо значний час навчання моделі витрачається на читання та запис файлів. Здебільшого, ми обмежені технічними характеристиками пристроїв зберігання інформації, що веде до додаткових витрат часу та коштів.

Для вирішення цієї проблеми під час процесу тренування нейронної мережі будемо використовувати хмарне сховище від Amazon Web Service під назвою Amazon Simple Storage Service (S3), яке дозволить зберігати будь-які типи та об'єми інформації. Amazon S3 має плоску неієрархічну структуру, всі об'єкти якої зберігаються в кошиках S3, що можуть бути організовані за допомогою загальних імен, які називаються префіксами. Amazon S3 підтримує паралельні запити, завдяки чому продуктивність S3 можна масштабувати за допомогою коефіцієнта обчислювального кластера, не вносячи зміни до програми [2].

Продуктивність масштабується для кожного префікса, завдяки чому для досягнення необхідної пропускної спроможності можна паралельно використовувати потрібну кількість префіксів. Кількість префіксів не обмежена. В Amazon S3 можна здійснювати не менше 3500 запитів на секунду на додавання даних і 5500 запитів на секунду на їх вилучення. Швидкість запису та зчитування файлів подано на рис. 2.1 та рис. 2.2. Кожен префікс S3 може забезпечувати такі значення, завдяки чому значно підвищити продуктивність досить легко. Під час нав-

чання нейронної мережі доступність даних є дуже важливим фактором. Перевагою над локальними сховищами є наявність доступу до збережених даних у будь-який час та з будь-якої географічної локації. Усі дані в S3 мають декілька реплікацій, тому ризик втратити їх мінімальний. В найгіршому випадку, можна завжди відновити втрачені дані через збережену мета-інформацію, яка зберігається для всіх файлів [3].

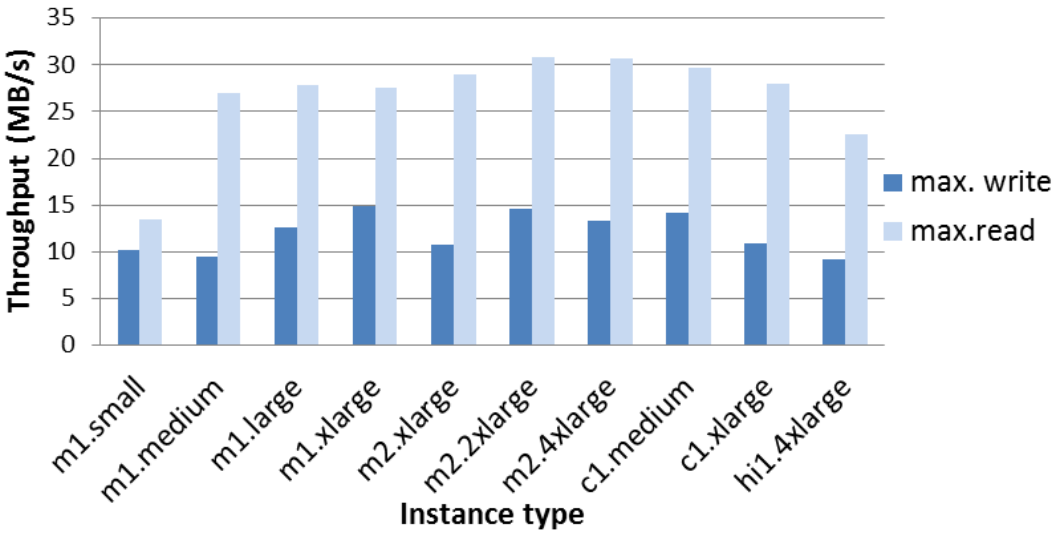

Рисунок 2.1 – Графік максимальної швидкості запису та зчитування файлів різних розмірів

Під час дослідження вхідних даних, розбитті їх на логічні частини та, в результаті, побудови плану попередньої обробки, може знадобитись багато операцій, виконання яких на усьому набору великих за об’ємом даних займе значну кількість часу. Використання S3 дозволить працювати з частиною усього набору, яка доступна в будь-який час. Це дозволяє оминути крок зі зчитуванням усього об’єму та дослідити вхідні дані, застосувавши операції по їх обробці всередині сервісу зберігання та одразу отримати проміжні результати.

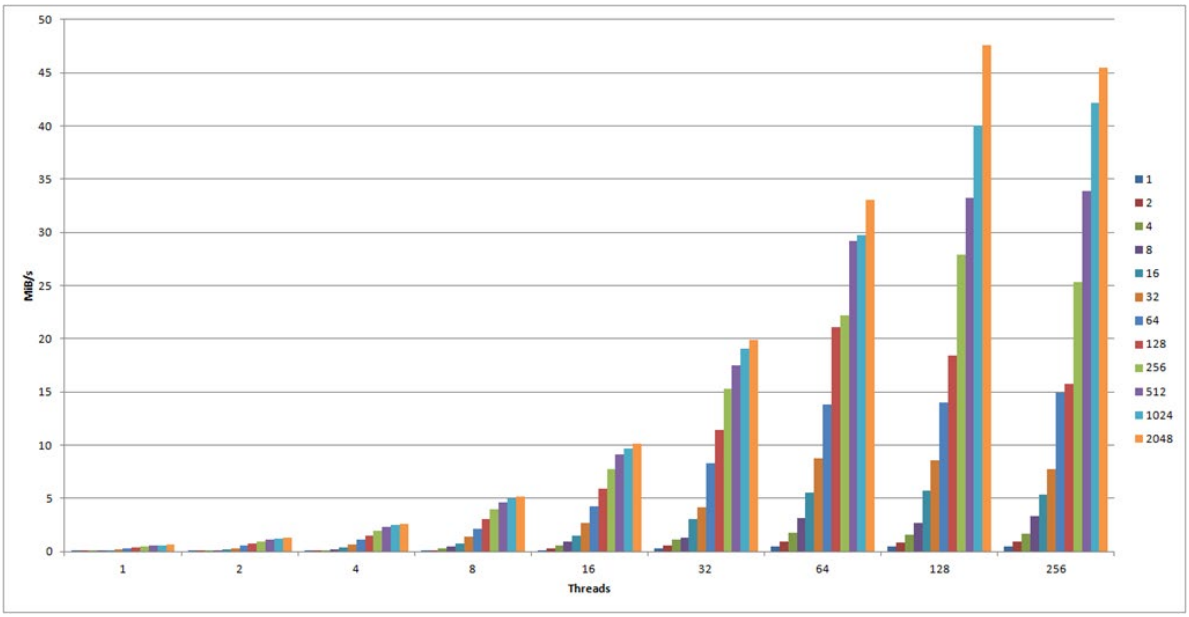

Рисунок 2.2 – Графік залежності швидкості читання від кількості потоків та розміру розбиття даних

Файли у сховищі Amazon S3 можуть мати мітких шести класів, які застосовуються в залежності від частоти звернень, від тих, що потребують швидкого доступу у будь-який час, до тих, що можуть зберігатись протягом великого періоду часу. Приклад міток наведено на рис. 2.3.

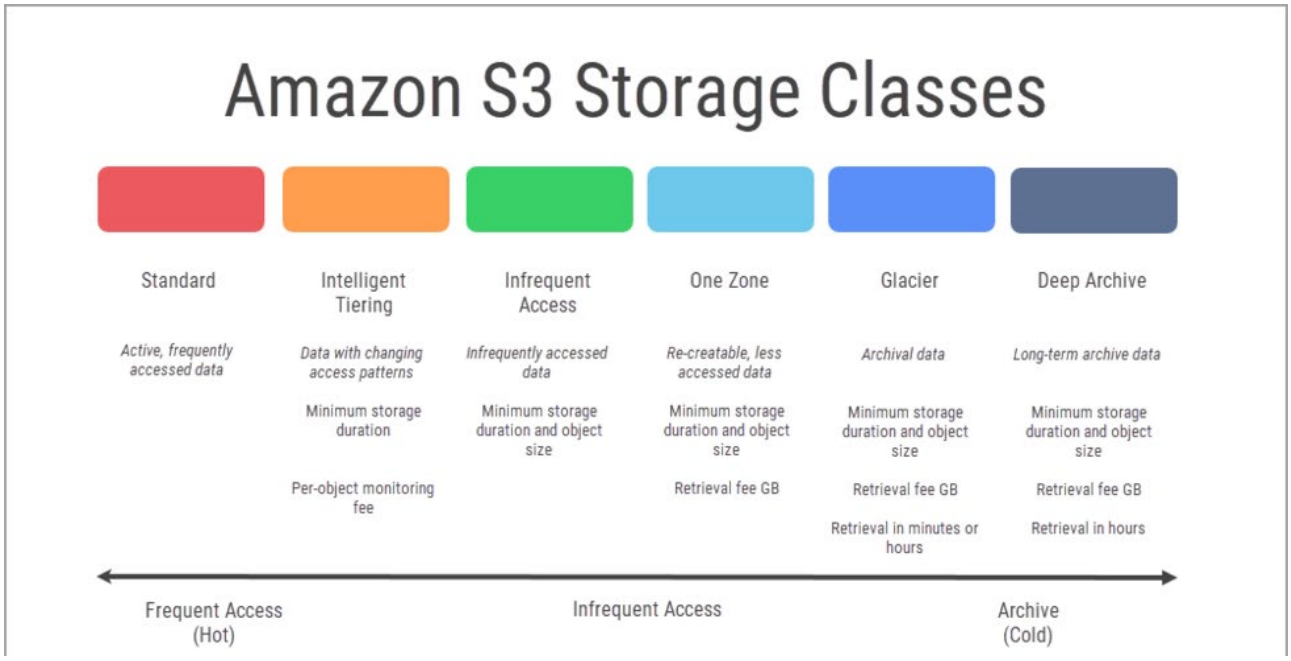

Рисунок 2.3 – Класи зберігання файлів в Amazon S3

2.2 Застосування технології Spark для розподіленої обробки неструктурованих та слабо структурованих даних

Apache Spark – це розподілений фреймворк обробки даних, що став дефакто стандартом обробки великих даних. Apache Spark використовує архітектуру головного/підлеглого з двома основними демонами та менеджером кластера:

Master Daemon – (головний/драйверний процес)

Worker Daemon – (підлеглий процес)

Spark кластер має одного Майстра та будь-яку кількість Підлеглих/Робочих. Драйвер і виконавці запускають свої окремі процеси Java, і користувачі можуть запускати їх на одному горизонтальному кластері spark або на окремих машинах, тобто у вертикальному кластері spark або в змішаній конфігурації машин [4].

Центральний координатор називається Spark Driver, і він спілкується з усіма працівниками. Кожен Worker вузол складається з одного або кількох Worker, які відповідають за виконання Task. Workers реєструються у Driver, який постійно має про них усю інформацію. Структура Spark подана на рис. 2.4.

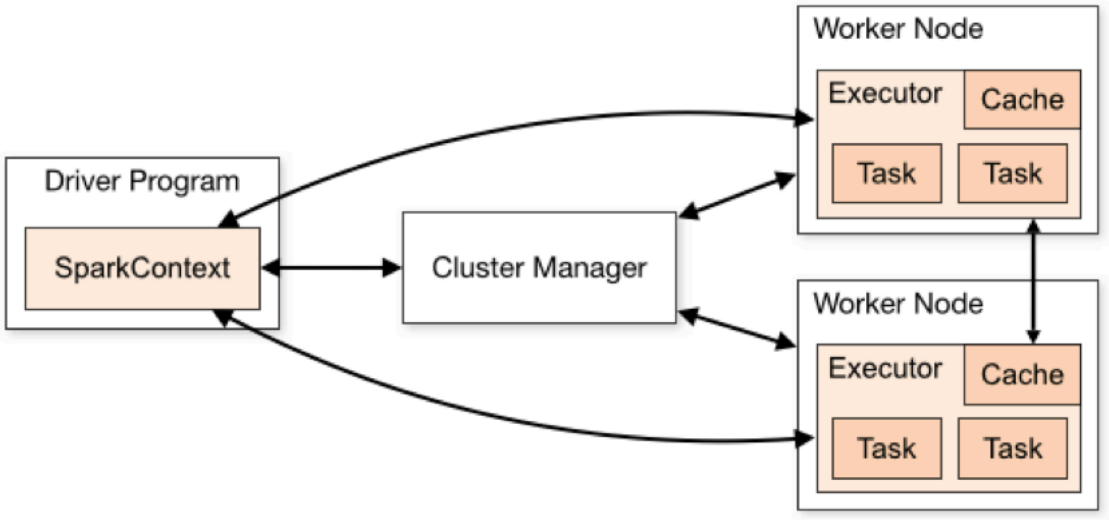

Рисунок 2.4 – Структура додатка Spark

Ця робоча комбінація Driver і Workers відома як Spark Application. Головна мета застосування технології Apache Spark під час розробки нейронної мережі – це підхід до розрахунків та маніпуляцій з вхідними даними. Замість того, щоб виконувати операції прямолінійно вздовж усього набору даних, він розбиває його на логістично однакові блоки, рівномірно та паралельно навантажуючи усі доступні для калькуляцій ресурси. Це дозволяє значно пришвидчити підготовку тренувального набору.

Spark складається з кількох компонентів, до числа яких входять і бібліотеки машинного навчання, як показано на рис. 2.5.

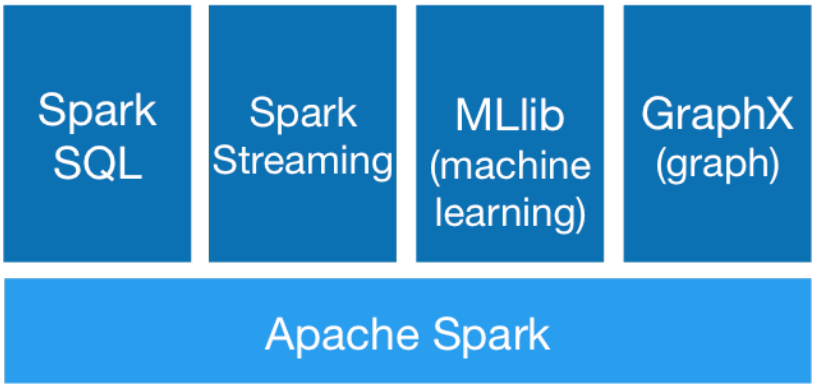

Рисунок 2.5 – Структура екосистеми Apache Spark

Spark ML надає базовий набір інструментів машинного навчання:

– алгоритми класифікації, регресії, кластеризації та спільної фільтрація;

– методи роботи із ознаками;

– конвеєри (pipelines);

– збереження та завантаження моделей та конвеєрів;

– утиліти: лінійна алгебра, статистика, обробка даних та ін.

Spark ML підтримує такі групи алгоритмів для етапу підготовки даних:

а) Extraction – вилучення об'єктів із “необроблених” даних;

б) Transformation – масштабування, перетворення чи зміна об'єктів;

в) Selection - вибір підмножини з більшого набору об'єктів;

г) Locality Sensitive Hashing (LSH) – цей клас алгоритмів поєднує аспекти перетворення ознак з іншими алгоритмами [5].

2.3 Загальна оркестрація задач для створення цілістного процесу ETL (Extract Tranform Load) для створення нейронної мережі за допомогою технології Airflow

Apache Airflow — це платформа з відкритим кодом для розробки, планування та моніторингу пакетно-орієнтованих робочих процесів. Розширювана структура Python від Airflow дає змогу створювати робочі процеси, підключаючись практично до будь-якої технології. Веб-інтерфейс допомагає керувати станом ваших робочих процесів. Airflow можна розгортати багатьма способами, від одного процесу на вашому ноутбуці до розподіленої установки для підтримки навіть найбільших робочих процесів [6, 24].

Airflow — це платформа оркестровки пакетного робочого процесу. Структура Airflow містить оператори для підключення до багатьох технологій і легко розширюється для підключення до нової технології. Якщо ваші робочі процеси мають чіткий початок і кінець і виконуються через рівні проміжки часу, їх можна запрограмувати як Airflow DAG.

DAG (Directed Acyclic Graph) — це основна концепція Airflow, яка збирає разом завдання, організовані за допомогою залежностей і зв’язків, щоб визначити, як вони мають виконуватися [27].

DAG на рис. 2.6 визначає чотири завдання - A, B, C і D - і вказує порядок, у якому вони повинні виконуватися, і які завдання залежать від інших.

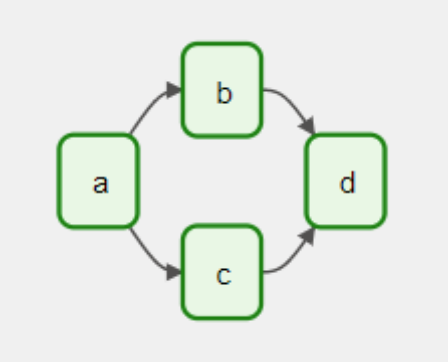

Рисунок 2.6 – Направлений ациклічний граф в Airflow

Сам DAG не має інформації про те, що відбувається всередині завдань; він слідкує за зв'язками, як їх виконувати – порядок їх запуску, скільки разів повторювати, чи є в процесі якась залежність від часу.

DAG повинен мати завдання для виконання, і вони, як правило, мають форму операторів, датчиків або потоку завдань.

Завдання/оператор має залежності від інших завдань (тих, що знаходяться вище за ним), і інші завдання залежать від нього (тих, що знаходяться нижче за ним). Оголошення цих залежностей між завданнями становить структуру DAG (ребра орієнтованого ациклічного графа).

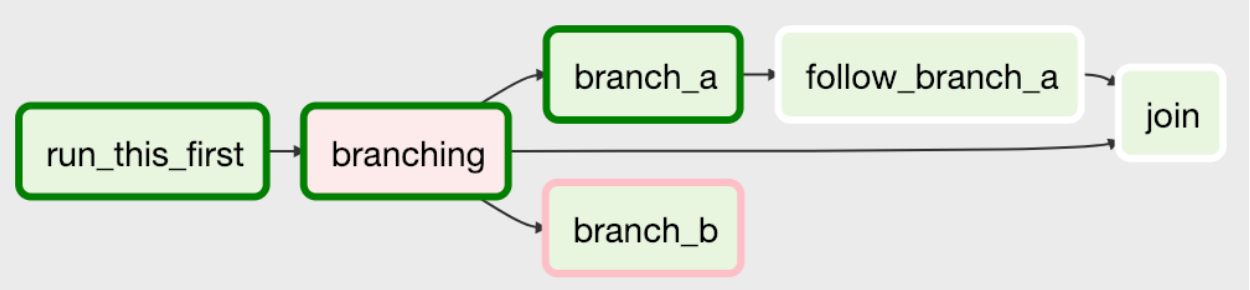

Рисунок 2.7 – Приклад структури зв'язків між задачами в Airflow

Робочі процеси визначаються як код Python, що означає:

– робочі процеси можна зберігати в системі керування версіями, щоб можна було повернутися до попередніх версій;

– робочі процеси можуть розроблятися кількома людьми одночасно;

– для перевірки функціональності можна писати тести;

– компоненти можна розширювати, і ви можете створювати на основі широкого набору існуючих компонентів.

Багата семантика планування та виконання дозволяє легко визначати складні конвеєри, що виконуються через регулярні проміжки часу. Заповнення дозволяє (повторно) запускати конвеєри на історичних даних після внесення змін у вашу логіку [25].

Користувальницький інтерфейс Airflow, показаний на рис. 2.8, забезпечує детальний огляд конвеєрів і окремих завдань, а також огляд конвеєрів за певний

час. З інтерфейсу ви можете перевіряти журнали та керувати завданнями, наприклад повторювати завдання у разі невдачі.

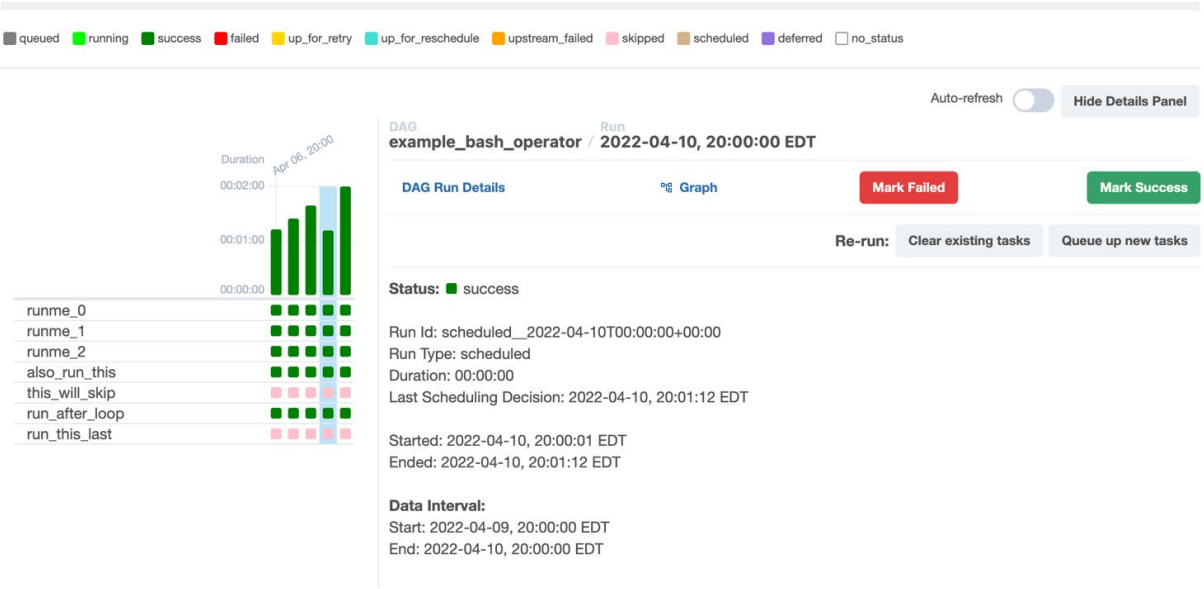

Рисунок 2.8 – Користувальницький інтерфейс Airflow

2.4 Побудова ETL процесу

ETL – це процес у сховищах даних, що означає «вилучення, перетворення та завантаження». Це процес, у якому інструмент ETL витягує дані з різних систем джерел даних, перетворює їх у проміжну область, а потім, нарешті, завантажує в систему сховища даних [26].

Першим кроком процесу ETL є вилучення (extraction). На цьому кроці дані з різних вихідних систем витягуються в різні формати, такі як реляційні бази даних, без SQL, XML і плоскі файли в проміжну область. Важливо витягувати дані з різних вихідних систем і зберігати їх спочатку в проміжній області, а не безпосередньо в сховищі даних, оскільки витягнуті дані мають різні формати і також можуть бути пошкоджені. Тому завантаження безпосередньо в сховище даних може пошкодити його, і відкат буде набагато складнішим. Тому це один із найважливіших етапів процесу ETL.

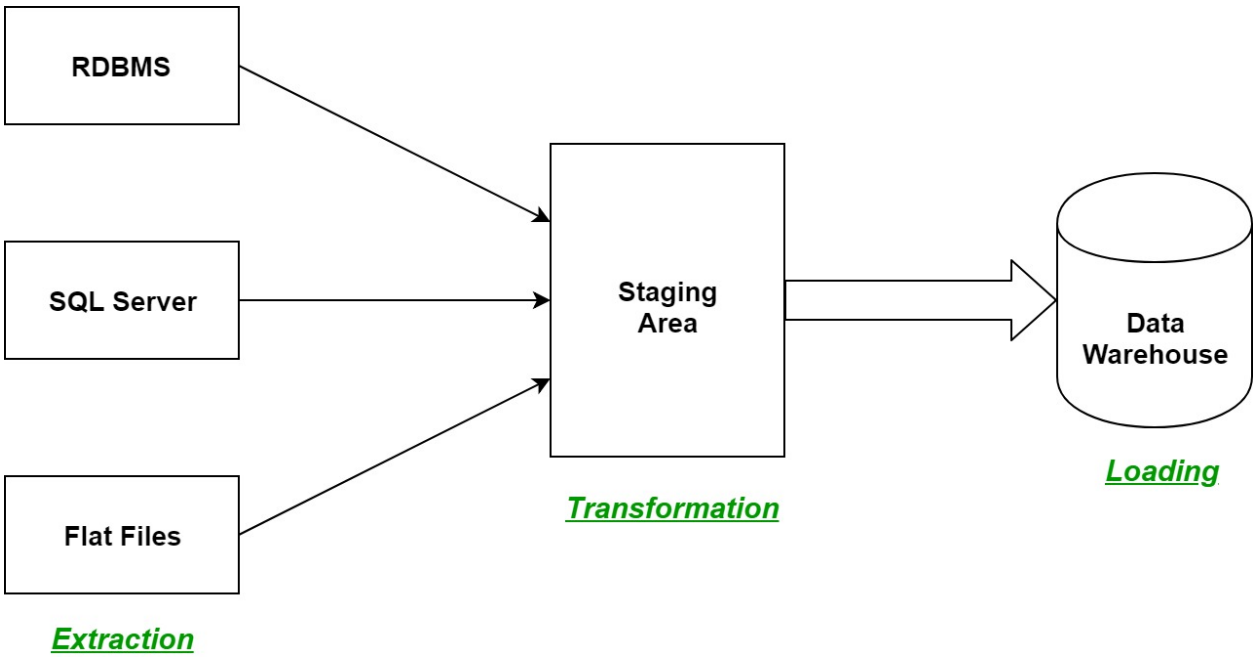

Рисунок 2.9 – Схема ETL процесу

Другим кроком процесу ETL є трансформація (transformation). На цьому етапі набір правил або функцій застосовуються до витягнутих даних, щоб перетворити їх у єдиний стандартний формат. Це може включати такі процеси/завдання:

– завантаження в сховище даних лише певних атрибутів;

– заповнення значень NULL деякими значеннями за замовчуванням;

– об’єднання кількох атрибутів в один;

– розбиття одного атрибута на кілька атрибутів;

– сортування кортежів на основі деякого атрибута (ключ-атрибут).

Третім і останнім кроком процесу ETL є завантаження (load). На цьому етапі трансформовані дані остаточно завантажуються в сховище даних. Іноді дані оновлюються шляхом дуже частого завантаження в сховище даних, а іноді це робиться через триваліші, але регулярні проміжки часу. Швидкість і період завантаження залежать виключно від вимог і відрізняються від системи до системи [29].

Існують інші підходи, які також використовуються для полегшення робочих процесів інтеграції даних.

Change Data Capture (CDC) визначає та фіксує лише вихідні дані, які були змінені, і переміщує ці дані в цільову систему. CDC можна використовувати для зменшення ресурсів, необхідних під час етапу «вилучення» ETL; його також можна використовувати незалежно для переміщення даних, які були перетворені в озеро даних або інше сховище в режимі реального часу.

Реплікація даних копіює зміни в джерелах даних у режимі реального часу або пакетами до центральної бази даних. Тиражування даних часто вказується як метод інтеграції даних. Фактично, він найчастіше використовується для створення резервних копій для аварійного відновлення.

Віртуалізація даних використовує рівень програмної абстракції для створення уніфікованого, інтегрованого, повністю придатного для використання перегляду даних – без фізичного копіювання, перетворення або завантаження вихідних даних у цільову систему. Функціональні можливості віртуалізації даних дозволяють організації створювати віртуальні сховища даних, озера даних і кіоски даних з одних і тих самих вихідних даних для зберігання даних без витрат і складності створення та керування окремими платформами для кожної з них. Хоча віртуалізацію даних можна використовувати разом із ETL, вона все частіше розглядається як альтернатива ETL та іншим методам фізичної інтеграції даних.

Інтеграція потокових даних (SDI) – процес постійно споживає потоки даних у реальному часі, перетворює їх і завантажує в цільову систему для аналізу. Ключове слово тут – постійно. Замість інтеграції знімків даних, отриманих із джерел у певний момент часу, SDI інтегрує дані постійно, коли вони стають доступними. SDI забезпечує сховище даних для аналітики, машинного навчання та додатків у реальному часі для покращення взаємодії з клієнтами, виявлення шахрайства тощо.

В запланованій архітектурі, як на рис. 2.10, для навчання нейронної мережі в якості тимчасового та кінцевого сховища ми будемо використовувати Amazon S3, в якому будуть знаходиться сирі та необроблені дані [29].

Рисунок 2.10 – Схема архітектури сервісів AWS

За допомогою сервісу AWS Glue, який включає в себе кластер, готовий для роботи з технологією Spark, вхідні дані будуть очищені, структуровані та підготовлені для навчання моделі. Також цей сервіс дозволить запустити та відстежувати навчання самої нейронної мережі. Кожний необхідний крок під час усього ETL процесу буде оркеструватись за допомогою Airflow, показувати що виконалося успішно, і які проблеми з'явились. Таким чином ми зможемо створити процес, який не потребує проміжного втручання [7].

2.5 Моніторінг стану та процесів у хмарному середовищі

Під час роботи в хмарному середовище та впродовж усього процесу розробки нейронної мережі потрібно обробляти та зберігати багато поточної інформації, відстежувати та знаходити вузькі місця в алгоритмах навчання, щоб отримати найкращий результат. Тому для вирішення цієї проблеми застосовують бібліотеки або сервіси, які дозволяють виконувати постіне логування. Таким чином, ми будемо мати повну та оновляєму інформацію про статус того чи іншого кроку в нашому процесі.

Кращим рішенням буде використання Amazon CloudWatch, який збирає та візуалізує в режимі реального часу журнали, показники та дані про події на автоматизованих інформаційних панелях, щоб оптимізувати нашу інфраструктуру та обслуговування програм. Він притримується поняття «Спостережливість» – описує, наскільки добре ви можете зрозуміти, що відбувається в системі, часто за допомогою інструментів для збору показників, журналів або трасування. У хмарі може бути важко досягти спостережливості через абсолютну складність системи. У центрах обробки даних чи в хмарі, щоб досягти операційної досконалості та досягти поставлених цілей, нам потрібно розуміти, як працюють наші системи. Рішення для спостереження дозволяють збирати та аналізувати дані з програм та інфраструктури, щоб ми могли розуміти їхній внутрішній стан і отримувати сповіщення, усувати неполадки та вирішувати проблеми з доступністю та продуктивністю програм для покращення якості вихідної нейронної мережі.

Своєчасне виявлення проблеми (в ідеалі до того, як вона вплине на кінцевих результат) є першим кроком у спостереженні. Виявлення має бути проактивним і багатогранним, включаючи сигнали тривоги при перевищенні порогів продуктивності, синтетичне тестування та виявлення аномалій. Загальним показником продуктивності є середній час виявлення (MTTD). Amazon CloudWatch покриває усі можливі ситуації, що можуть виникнути під час розробки.

Інструменти моніторингу записують статистику продуктивності протягом певного часу, щоб можна було визначити моделі використання. Агенти моніторингу записують вибрані показники через встановлені проміжки часу та зберігають отримані дані у форматі часових рядів.

Моніторинг інфраструктури дає змогу співвідносити показники та журнали зі стеку інфраструктури, щоб зрозуміти та вирішити основні причини проблем із продуктивністю.

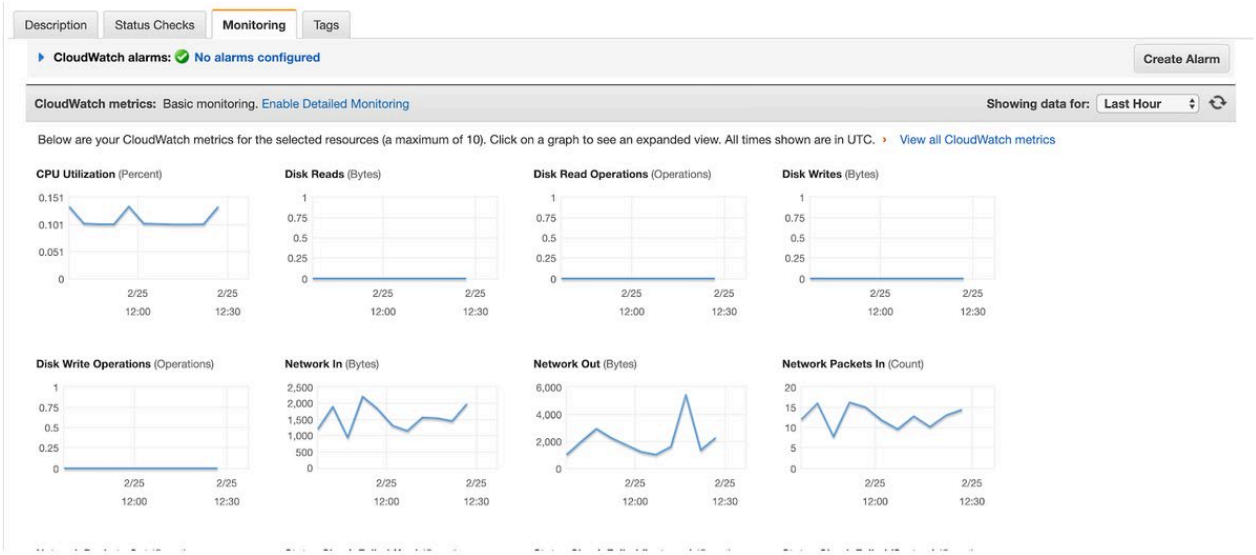

Рисунок 2.11 – Інтерфейс загальної інформації

Нове покоління інструментів тепер використовує потужність штучного інтелекту та машинного навчання для спостереження, використовуючи моделі машинного навчання для виявлення аномальної поведінки додатків і виявлення критичних проблем, перш ніж вони спричинять потенційні збої або збої в роботі.

Розслідування є найбільш трудомістким етапом оперативного заходу. Коли щось йде не так, може бути важко зрозуміти, що найважливіше виправити. Спільне використання кількох джерел спостережування може допомогти вам швидко дослідити, щоб зрозуміти першопричину, але щоб зробити це ефективно, нам потрібно співвідносити дані між показниками, журналами та трасуваннями.

Інструменти візуалізації допомагають осмислити дані, співвідносячи дані спостережування в інтуїтивно зрозумілі графічні відображення.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Опис програми

Побудована програма для багатокласcової класифікації складається з двох основних модулів: main_process_methods і main_process_dag, та трьох основних сервісів Amazon: Amazon S3, AWS Glue, Amazon Managed Workflows for Apache Airflow.

Модуль main_process_methods містить визначення Python методів для обробки даних, які викликаються в процесі виконання Airflow задач, та зберігає статичну інформацію, таку як словник назв пристроїв, структуру таблиць та мітки класів для даних. Цей модуль використовується як поширена бібліотека, та за допомогою модулю wheel . За допомогою сторонньої бібліотеки wheel, main_process_methods перетворюється в пакет з розширенням .whl, який зберігається в Amazon S3 бакеті. Таким чином, Amazon Airflow завжди має до нього доступ.

Модуль main_process_dag містить означення DAG'у, містить виклики Python функцій обгорнутих в Airflow PythonOperator. За допомогою такого підходу, Airflow ідентифікує, надані функції не просто як Python код, а Airflow задачі. Завдяки цьому внутрішній логгер починає збирати усю необхідну інформацію про статус виконання задач, і ми можемо створювати послідовності їх виконання. Сам модуль міститься в сховищі S3, щоб сервіс Airflow мав до нього доступ.

В модулі multi_classification_model міститься означення моделі машинного навчання, її параметри та результати.

3.2 Розгортання Apache Airflow та побудова процесу ETL

Для оркестрації задач створимо спрямований ациклічний граф, вершини якого будуть містить у собі задачі обробки вхідних даних та навчання моделі

нейронної мережі. Основні задачі, які повинні бути вирішені протягом усього процесу:

– завантажити вхідні дані в створений S3 бакет yevhen-001-nure;

– проставити мітки класів для кожного набору даних;

– об’єднати усі дані в єдиний датасет;

– розділити дані на тренувальну та тестову вибірку;

– обробити отримані датасети;

– навчити модель нейронної мережі на оброблених даних;

– вивести та зберегти якісні результати для подальшої оцінки;

– завантажити отриману модель в сховище S3.

Кожна задача буде представлена як Airflow задача та зберігати в собі логічні та інкапсулюють робочий код. Увесь процес можна поділити на чотири логічні частини:

– загрузка сирих даних у сховище S3;

– пре обробка отриманих даних;

– створення та навчання моделі;

– запис тренованої моделі та результатів статистичних метрик у S3.

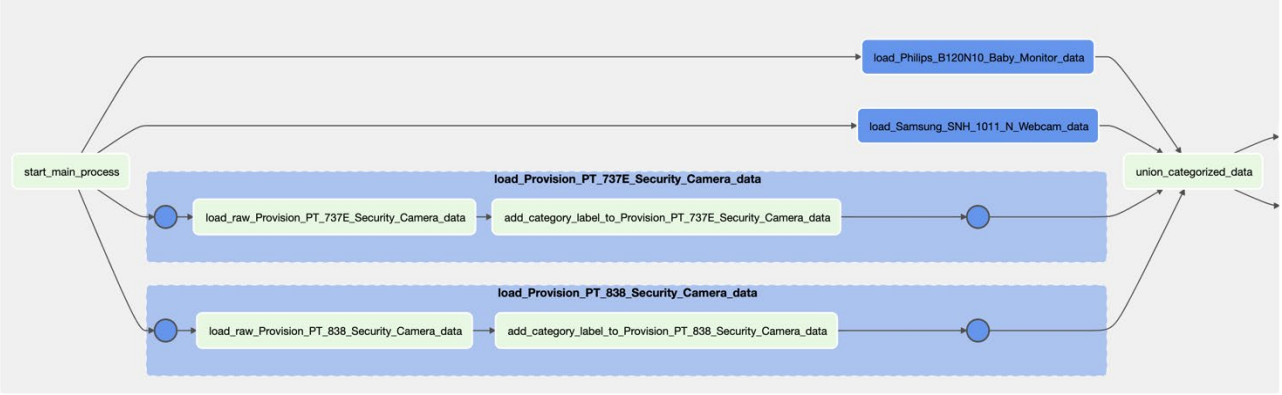

Рисунок 3.1 – Граф загрузки початкових даних у сховище S3

Процес загрузки початкових даних складається з чотирьох паралельних груп задач, позначених на рис. 3.1, як блакитні прямокутники. Кожна група відповідає за виконання двох послідовних задач, а саме загрузки сирих даних та

додавання до них категоріальної мітки. Таким чином, в порівнянні з стандартним підходом пре обробки, за рахунок створення паралельного потоку даних процес загрузки виконається приблизно у 4 рази швидше. Етап закінчується задачою об'єднання даних з паралельних розрахунків у єдиний класифікований датасет, готовий до подальших перетворень.

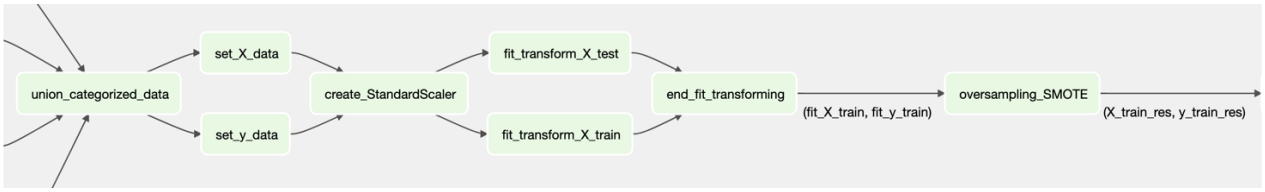

Рисунок 3.2 – Граф процесу пре обробки даних

На рис. 3.2 показано наступний етап процесу, який починається з паралельного розділення даних на навчальний та тестовий набори з включеним параметром stratify. Параметр stratify забезпечить те саме співвідношення розподілу класів для навчання та тестування, що й вихідний набір даних. Це має вирішальне значення у нашому випадку, так як ми обробляємо незбалансований набір даних. Інакше може статися, що навчальні дані складаються лише з основного класу.

Масштабування функцій є важливим кроком у моделюванні алгоритмів із наборами даних. Дані, які зазвичай використовуються для цілей моделювання, отримують різними засобами. Таким чином, отримані дані містять ознаки різних розмірів і масштабів. Різні масштаби характеристик даних негативно впливають на моделювання набору даних. Це призводить до упередженого результату прогнозів з точки зору помилки неправильної класифікації та показників точності. Таким чином, перед моделюванням необхідно масштабувати дані. Багато алгоритмів машинного навчання вимагають відповідного масштабування даних, щоб працювати добре. Отже, ми будемо масштабувати функції за допомогою StandardScaler з бібліотеки sklearn.

Тепер настала черга позбавитися дисбалансу класів. Передискретизація (oversampling) є одним із найбільш широко використовуваних методів боротьби

з класами дисбалансу. Для цього ми генеруємо синтетичні зразки для класів меншин, щоб переконатися, що у нас достатньо даних для навчання моделі. На рис. 3.2 позначена задача oversampling_SMOTE, яка буде використовувати SMOTE (Synthetic Minority Over-sampling Technique) для передискретизації. Даний алгоритм можна знайти в Python модулі imblearn.

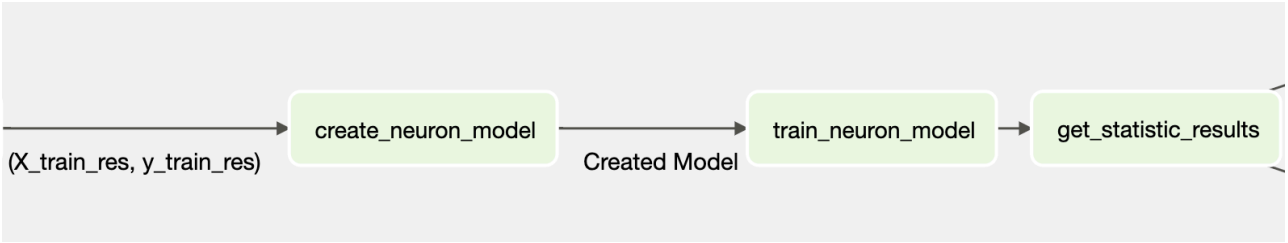

Рисунок 3.3 – Граф процесу створення та навчання моделі

На наступному етапі масштабовані та сбалансовані дані подаються на вхід створеної моделі нейронної мережі, структура якої подана на рис. 3.4.

Метрикою, обраною для оцінки ефективності моделі, є оцінка F1 кожного класу. У багатьох проектах з машинного навчання метрикою оцінки є точність, але не для незбалансованих даних. Якщо модель передбачає мітку класу більшості для кожної точки даних, точність все одно буде хорошою, оскільки 90% буде передбачено правильно.

Ми використали класифікаційний звіт, щоб переглянути наведені вище показники для кожного класу. Він забезпечує краще розуміння загальної продуктивності нашої навченої моделі, відображаючи точність, запам’ятовування, оцінку F1 і підтримку.

Заключний етап включає в себе збір та загрузку якісних результатів за допомогою операторів запису даних в хмарне сховище S3 для подальшої оцінки та висновків про роботу побудованої моделі.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                7424

 dense_1 (Dense)             (None, 32)                2080

 dropout (Dropout)           (None, 32)                0

 dense_2 (Dense)             (None, 16)                528

 dense_3 (Dense)             (None, 6)                 102

=================================================================
Total params: 10,134
Trainable params: 10,134
Non-trainable params: 0
```

Рисунок 3.4 – Структура нейронної мережі

Використовуючи дану архітектуру, даний процес повністю відповідає означенню ETL процесу і є кращим рішенням для моделі машинного навчання [30].

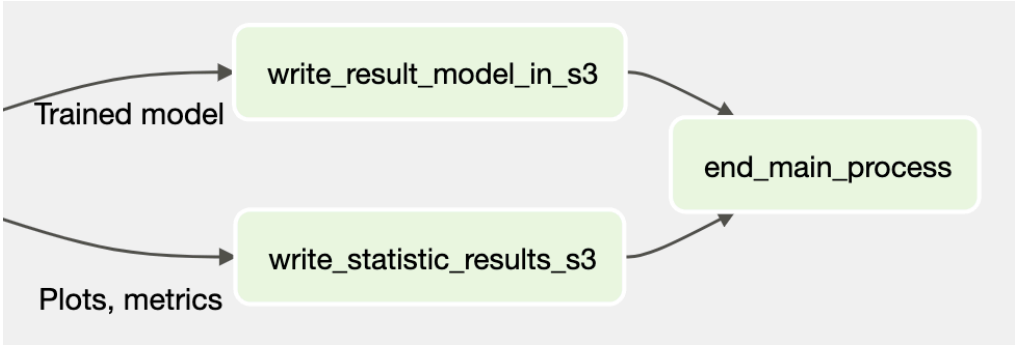

Рисунок 3.5 – Граф процесу збереження результатів

3.3 Локальне тестування Airflow DAG

Amazon Managed Workflows for Apache Airflow сервіс є хмарною обгорткою оригінальної технології, тому під активації Airflow середовища резервуються потужності на розрахунковому кластері. Це дозволяє і розробникам, і

команді підтримки сервісів підтримувати працездатність DAG’ів в будь-який час. Однак це накладає додаткові витрати під час розробки.

Тому для зменшення вартості розробки та тестування було розгорнуто локальну версію Apache Airflow як окремий Docker контейнер, який створює ізольоване середовище та імітує усі Airflow процеси, і дозволяє налагодити правильну роботу його компонентів [22, 23]. Конфігурація Docker контейнеру та його компонентів наведена в Додатку А.

Важливою частиною контейнеру є елемент Vault – локальне захищене сховище, в якому зберігаються ключі доступу до сервісів та аккаунтів AWS. Усі елементи цього сховища проходять через шифрувальні алгоритми та зберігаються в окремому захищеному середовищі. В результаті, ми маємо доступ до сховища S3 з функціональними модулями, і колючи доступа не з’являються в коді програми ані в якості справжніх незашифрованих ключів, ані будь-яких ключових слів, які стосуються прихованих даних.

Після налагодженя правильної роботи DAG’у можна з легкістю перенести створений для нього файл з означенням до S3, де зберігаються DAG’и Amazon Managed Workflows for Apache Airflow сервісу. Головним плюсом такого підходу тестування є значне зменшення вихідної вартості підтримки цієї технології, але локальне тестування позбавлене більшості можливостей оптимізації процесів, наприклад, динамічне збільшення розрахункових потужностей, та збільшений час звернення до сховища S3 і запису в нього даних.

4 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ ТА ЇХ АНАЛІЗ

В результаті ETL процесу була отримана модель, яка здатна розпізнавати тип атаки з пристроїв IoT, а саме атаку на TCP протокол, Junk атаку, Scan атаку, Udp-flood атаку та комбіновану атаку. На рис. 4.1 показано графік точності моделі за метрикою accuracy.

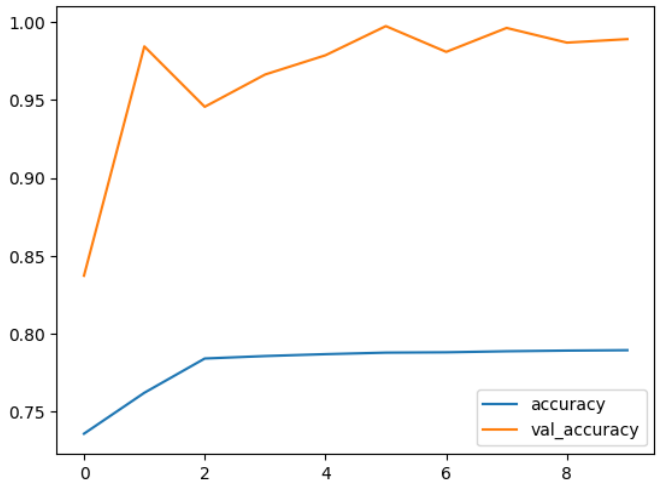

Рисунок 4.1 – Графік точності моделі за метрикою accuracy

Найкраща досягнута точність моделі протягом десяти епох за цією метрикою є 0.77 %. Тестування протягом навчання показує гарний результат, фінальне значення близько до одиниці.

Однак, ця метрика не є об’єктивною для випадку багатокласової класифікації, так як вона є середньою точністю серед класифікаторами для кожного класу.

Для детального аналізу отриманої моделі використаємо метрики precision, recall та f1-score, результати якої наведені на рис. 4.3. На ньому видно, що модель вдало розпізнає мітки класів звичайного трафіку, атаки типу Junk, атаки Scan, Udp-flood та комбінованих атак.

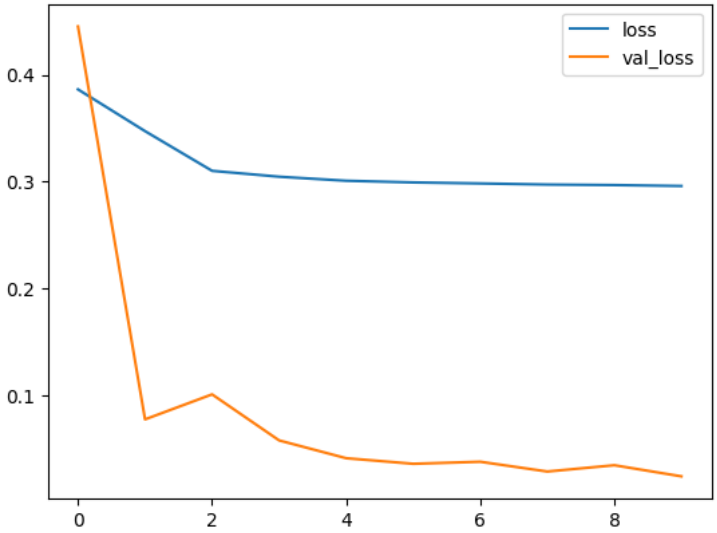

Рисунок 4.2 – Графік функції втрат

Однак атаку на TCP протокол модель тільки вгадує. Це випливає з того, що ця атака має схожі характеристики з усіма іншими. Тож для покращення результатів для цього типу потрібні додаткові якісні характеристики.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	97015
1	0.46	0.00	0.00	96065
2	1.00	1.00	1.00	28313
3	0.95	1.00	0.97	29155
4	0.53	1.00	0.69	106267
5	1.00	0.97	0.99	58933
micro avg	0.76	0.76	0.76	415748
macro avg	0.82	0.83	0.77	415748
weighted avg	0.75	0.76	0.69	415748
samples avg	0.76	0.76	0.76	415748

Рисунок 4.3 – Результати багатокласового класифікатора

ВИСНОВКИ

В кваліфікаційній роботі були досліджені та застосовані методи обробки та зберігання великих даних. Для досягнення поставленої мети дослідження був проведений огляд і аналіз сучасного стану задачі зберігання та обробки великих даних; створене хмарне сховище; розроблена інфраструктура усього процесу за використанням хмарних сервісів. Було запропоновано рішення для обробки значних об’ємів даних та обрана модель нейронної мережі. В результаті досліджень була побудована ефективна інфраструктура з сервісів Amazon, яка дозволить зберігати та обробляти великі об’єми даних для підготовки до навчання нейронної мережі. Для збереження вхідних та оброблених даних було створене та налаштоване хмарне сховище Amazon S3. Сервісом для виконання коду для обробки даних створений розрахунковий кластер, який підтримує технологію Apache Spark, за допомогою AWS Glue. В якості оркестратора задач протягом усього процесу обрана технологія Apache Airflow, яка дозволить розбити задачу на логічні кроки та відстежувати їх статус виконання. Збирання проміжної інформації та логування буде виконано за допомогою сервісу Amazon CloudWatch, який дозволить уникнути помилок під час розробки та оптимізувати отримані результати.

В результаті кваліфікаційної роботи був створений класифікатор на основі моделі нейронної мережі, який здатний розпізнати, чи є серед вхідних потоків даних корисні, або виявити тип атаки на мережу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. An introduction to big data concepts and terminology. digitalocead. URL : https://www.digitalocean.com/community/tutorials/an-introduction-to-big-data-concepts-and-terminology (дата звернення: 01.11.2022).

2. Data Lakes on AWS. Amazon Web Services, Inc. URL : https://aws.amazon .com/big-data/datalakes-and-analytics/datalakes/ (дата звернення: 01.11.2022).

3. Die Funktionsweise von Amazon S3-Buckets Seagate Deutschland. *Seagate.com*. URL : https://www.seagate.com/gb/en/blog/how-amazon-s3-buckets-work/ (дата звернення: 10.11.2022).

4. Apache HTTP Server Test Page powered by CentOS. URL : https://web.ecs.syr.edu/~wedu/seed/Book/book_sample_tcp.pdf (дата звернення: 29.10.2022).

5. PySpark Documentation – PySpark 3.3.1 documentation. Apache Spark™ - Unified Engine for large-scale data analytics. URL : https://spark.apache.org/docs/ latest/api/python/index.html (дата звернення: 10.10.2022).

6. What is Airflow? – Airflow Documentation. Apache Airflow. URL : https://airflow.apache.org/docs/apache-airflow/stable/index.html (дата звернення: 29.10.2022).

7. Serverless Data Integration – AWS Glue – Amazon Web Services. Amazon Web Services, Inc. URL : https://aws.amazon.com/glue/ (дата звернення: 01.11.2022).

8. Bulakh V., Kirichenko L., Radivilova T. Time Series Classification Based on Fractal Properties. 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, 21–25 August 2018. URL : https://doi.org/10.1109/dsmp.2018.8478532 (дата звернення: 29.10.2022).

9. Classification Methods of Machine Learning to Detect DDoS Attacks / Radivilova T. et al. 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Metz, France, 18–21 September 2019. URL : https://doi.org/10.1109/idaacs.2019.8924406 (дата звернення: 1.11.2022).

10. Comparative Analysis of Pseudorandom Number Generation In The Up-to-Date Wireless Data Communication / L. O. Kirichenko et al. Telecommunications and Radio Engineering. 2011. Vol. 70, no. 4. P. 325–333. URL : https://doi.org/10.1615/telecomradeng.v70.i4.20 (дата звернення: 2.11.2022).

11. Intrusion Detection Based on Machine Learning Using Fractal Properties of Traffic Realizations / Radivilova T. et al. 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT), Kyiv, Ukraine, 18–20 December 2019. URL : https://doi.org/10.1109/atit49449.2019.9030452 (дата звернення: 1.11.2022).

12. Kirichenko L., Radivilova T., Bulakh V. Binary Classification of Fractal Time Series by Machine Learning Methods. Advances in Intelligent Systems and Computing. Cham, 2019. P. 701–711. URL : https://doi.org/10.1007/978-3-030-26474-1_49 (дата звернення: 29.10.2022).

13. Kirichenko L., Radivilova T., Bulakh V. Classification of Fractal Time Series Using Recurrence Plots. 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, 9–12 October 2018. URL : https://doi.org/10.1109/infocommst.2018.8632010 (дата звернення: 2.11.2022).

14. Kirichenko L., Radivilova T., Bulakh V. Machine Learning in Classification Time Series with Fractal Properties. Data. 2018. Vol. 4, no. 1. P. 5. URL : https://doi.org/10.3390/data4010005 (дата звернення: 1.11.2022).

15. Kirichenko L., Radivilova T., Zinkevich I. Comparative Analysis of Conversion Series Forecasting in E-commerce Tasks. Advances in Intelligent Systems and Computing II. Cham, 2017. P. 230–242. URL : https://doi.org/10.1007/978-3-319-70581-1_16 (дата звернення: 29.10.2022).

16. Kirichenko L., Saif A., Radivilova T. Generalized Approach to Analysis of Multifractal Properties from Short Time Series. International Journal of Advanced Computer Science and Applications. 2020. Vol. 11, no. 5. URL : https://doi.org/10.14569/ijacsa.2020.0110527 (дата звернення: 29.10.2022).

17. Kirichenko L., Zinchenko P., Radivilova T. Classification of Time Realizations Using Machine Learning Recognition of Recurrence Plots. Advances in Intelli-

gent Systems and Computing. Cham, 2020. P. 687–696. URL : https://doi.org/10.1007/978-3-030-54215-3_44 (дата звернення: 28.10.2022).

18. Kirichenko L., Bulakh V., Radivilova T. Fractal time series analysis of social network activities. 2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkov, 10–13 October 2017. URL : https://doi.org/10.1109/infocommst.2017.8246438 (дата звернення: 1.11.2022).

19. Radivilova T., Kirichenko L., Bulakh V. Comparative analysis of machine learning classification of time series with fractal properties. 2019 IEEE 8th International Conference on Advanced Optoelectronics and Lasers (CAOL), Sozopol, Bulgaria, 6–8 September 2019. URL : https://doi.org/10.1109/caol46282.2019.9019416 (дата звернення: 2.11.2022).

20. The Methods to Improve Quality of Service by Accounting Secure Parameters / T. Radivilova et al. Advances in Computer Science for Engineering and Education II. Cham, 2019. P. 346–355. URL : https://doi.org/10.1007/978-3-030-16621-2_32 (дата звернення: 2.11.2022).

21. Two Approaches to Machine Learning Classification of Time Series Based on Recurrence Plots / Radivilova T. et al. 2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 21–25 August 2020. URL : https://doi.org/10.1109/dsmp47368.2020.9204021 (дата звернення: 2.11.2022).

22. Key features and use cases. Docker Documentation. URL : https://docs.docker.com/compose/features-uses/ (дата звернення: 07.12.2022).

23. Declare default environment variables in file. Docker Documentation. URL: https://docs.docker.com/compose/env-file/ (дата звернення: 29.10.2022).

24. What Is Amazon Managed Workflows for Apache Airflow (MWAA)? - Amazon Managed Workflows for Apache Airflow. URL : https://docs.aws.amazon.com/mwaa/latest/userguide/what-ismwaa.html#architecture-mwaa (дата звернення: 28.10.2022).

25. Managing Python dependencies in requirements.txt - Amazon Managed Workflows for Apache Airflow. URL : https://docs.aws.amazon.com/mwaa/latest/

userguide/best-practices-dependencies.html#best-practices-dependencies-cli-utility (дата звернення: 29.10.2022).

26. Performance tuning for Apache Airflow on Amazon MWAA - Amazon Managed Workflows for Apache Airflow. URL : https://docs.aws.amazon.com/mwaa/latest/userguide/best-practices-tuning.html (дата звернення: 28.10.2022).

27. Adding or updating DAGs - Amazon Managed Workflows for Apache Airflow. URL : https://docs.aws.amazon.com/mwaa/latest/userguide/configuring-dag-folder.html (дата звернення: 28.10.2022).

28. Let's Architect! Optimizing the cost of your architecture | Amazon Web Services. Amazon Web Services. URL : https://aws.amazon.com/blogs/architecture/lets-architect-optimizing-the-cost-of-your-architecture/ (дата звернення: 28.10.2022).

29. Amazon Web Services. Everything You Need to Know About Big Data: From Architectural Principles to Best Practices, 2019. YouTube. URL : https://www.youtube.com/watch?v=MotN5f6_xl8 (дата звернення: 28.10.2022).

30. Enhance your machine learning development by using a modular architecture with Amazon SageMaker projects | Amazon Web Services. Amazon Web Services. URL : https://aws.amazon.com/blogs/machine-learning/enhance-your-machine-learning-development-by-using-a-modular-architecture-with-amazon-sagemaker-projects/ (дата звернення: 29.10.2022).

31. Башкатов Е.О. Побудова ефективної інфраструктури сервісів AMAZON для застосування методів машинного навчання // Тези XVI Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (м. Дніпро, 14-15 грудня 2022 р.). Дніпро : ДІІТ, 2022. С. 141.