МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна технологія розпізнавання трафіку алгоритмів генерації доменів на основі глибинного машинного навчання»

Завідувач випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Москаленко В. В.**

Студента групи ІНм – 91н **Міщенко Н. В.**

СУМИ 2021

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *«Інформатика»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Міщенко Нікіті Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна технологія розпізнавання трафіку алгоритмів генерації доменів на основі глибинного машинного навчання*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____
2. Термін здачі студентом закінченого проекту (роботи)_____
3. Вхідні данні до проекту (роботи) _____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що належить розробити)
1) Огляд методів та алгоритмів розпізнавання трафіку алгоритмів генерації доменів; 2) Постановка задачі і формування завдань дослідження; 3) Вибір програмних засобів для розробки технології розпізнання алгоритмів генерації доменів; 4) Формування початкової вибірки для навчання та критеріїв валідації; 5) Розробка системи; 6) Аналіз результатів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____
_____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	Огляд методів та алгоритмів розпізнавання трафіку алгоритмів генерації доменів		
2.	Постановка задачі та формування завдань дослідження		
3.	Опис вибору моделей та критеріїв вибірки для навчання		
4.	Розробка інформаційної системи та формування результатів		
5.	Оформлення магістерської роботи		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 63 стор., 20 рис., 4 таблиці, 3 додаток, 51 літературних джерел.

Об'єкт дослідження — розпізнавання трафіку алгоритмів генерації доменів.

Мета роботи — розробка інформаційної системи розпізнавання трафіку алгоритмів генерації доменів на основі глибинного машинного навчання.

Результати — проведений аналіз літератури, методів та підходів, які використовуються при розпізнавання трафіку алгоритмів генерації доменів (DGA), вивчено особливості та принципи роботи DGA. В ході дослідження було обрано засоби та технології для реалізації шляхом аналізу предметної області, сформовано початкову вибірку та опис її ознак. Після ознайомлення з існуючими рішеннями було знайдено дослідницьку прогалину та було обрано моделі для навчання, часових згорткових мереж (TCN), що засновані на методах глибинного машинного навчання та лягли у основі інформаційної технології, яка була розроблена в результаті роботі. Дана інформаційної технологія дозволяє з високою точністю виявляти домени згенеровані DGA. Технологія була реалізований за допомогою мови програмування Python, хмарного середовища розробки Google Colab та бібліотеки машинного навчання TensorFlow та SciKit-Learn.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗПІЗНАВАННЯ ТРАФІКУ АЛГОРИТМІВ ГЕНЕРАЦІЇ ДОМЕНІВ НА ОСНОВІ ГЛИБИННОГО МАШИННОГО НАВЧАННЯ, INFORMATION TECHNOLOGY FOR TRAFFIC DETECTION OF DOMAIN GENERATION ALGORITHMS BASED ON DEEP MACHINE LEARNING

ЗМІСТ

ВСТУП

З бурхливим розвитком Інтернету кіберпростір став найпопулярнішим середовищем для обміну інформацією майже для всіх аспектів повсякденного життя. Безпека кіберпростору страждає від постійно зростаючих загроз. В останні роки різноманітні спалахи шкідливого програмного забезпечення спричинили значні збитки уряду, енергетиці, виробництву та іншій ключовій інформаційній інфраструктурі [1]. Викрадена інформація включала конфіденційні дані розвідки, фінансові записи та особисту інформацію. Використання подібної інформації може бути катастрофічним, особливо коли вона стає загальнодоступною або продається на чорному ринку. Деякі статистичні дані щодо впливу кібербезпеки на підприємства, організації та приватних осіб:

- За останні роки кіберзлочинність спричинила викрадання коштів на суму понад 400 мільярдів доларів та витрати на покриття збитків, заподіяних злочинами [2];
- В даний час зловмисники отримують понад 1 мільярд доларів США щорічного доходу від атак Ransomware, таких як атаки Wannacry та CryptoWall [3];
- Передбачається, що організації у всьому світі щорічно витрачатимуть щонайменше 100 мільярдів доларів на захист кібербезпеки [4];

Захист від шкідливого програмного забезпечення є критично важливим для безпеки кіберпростору. Незважаючи на досягнення в галузі технологій безпеки, досі важко ефективно реагувати на кібератаки за допомогою командно-контрольних (C&C) серверів. Сервери C&C слугують командними центрами, які шкідливе програмне забезпечення, пов'язане з цілеспрямованими атаками, використовує для зберігання викрадених даних або отримання команд. За останні роки кібератаки з використанням C&C серверів значно зросли. Такі сервери зазвичай управляють бот-мережами,

набором заражених персональних комп'ютерів (PC), видаючи їм команди або керуючи кодом, отриманим віддалено від зловмисників. Щоб приховати свої C&C сервери, зловмисники часто використовують алгоритм генерації доменів (DGA), який автоматично генерує доменні імена для C&C серверів. Налагодження комунікаційних та комунікаційних послуг є життєво важливим кроком для зловмисників, щоб вони могли рухатись побічно всередині мережі.

Проблема галузі полягає у тому, що протидіяти зростаючій витонченості кібератак, значну частину з яких складають DGA, стає все більш складним завданням, оскільки оборонні засоби швидко застарівають. Однією з основних сфер, яка в значній мірі впливає на кібербезпеку, є машинне навчання (ML). Виявлення вторгнень, аналіз шкідливих програм і виявлення спаму – основні області застосування методів ML. Відповідно до дослідження методів своєчасного виявлення кібератак та оригінальної таксономії [5] яка стверджує, що алгоритми поверхневого машинного навчання (SL) застосовуються до всіх основних областей виявлення кібератак, але алгоритми глибинного навчання (DL) є менш застосованими та загальна кількість алгоритмів, заснованих на DL, значно менше, ніж на SL. Ключовим чинником цього є те, що пропозиції DL, засновані на величезних нейронних мережах, є набагато більш новою ініціативою, з першої комп'ютерної реалізацією, досягнутої в 2006 році [6]. Існує безліч визначень DL і глибинних нейронних мереж (DNN). Просте визначення свідчить, що DL – це набір алгоритмів машинного навчання, які намагаються навчатися на декількох рівнях, що відповідають різним рівням абстракції. Ці рівні відповідають окремими рівнями концепцій, де концепції більш високого рівня визначаються з понять більш низького рівня, і одні і ті ж концепції нижчого рівня можуть допомогти визначити багато концепції більш високого рівня [7]. У певних областях, таких як розпізнавання образів та у методах обробки мови, алгоритми ПН значно програють алгоритмам DL.

Задля підвищення ефективності виявлення та значного зменшення збитків, що можуть задати DGA, необхідно розробити інформаційну систему розпізнавання трафіку DGA на основі глибинного машинного навчання. Вирішення цієї задачі є корисним для компаній, розробників засобів захисту від кібератак, та користувачів PC які хочуть збільшити ступінь захисту від кібератак.

Мета – розробка технології розпізнавання трафіку алгоритмів генерації доменів на основі глибинного машинного навчання.

Об'єкт – розпізнавання трафіку алгоритмів генерації доменів.

Предмет – розпізнавання трафіку алгоритмів генерації доменів на основі глибинного машинного навчання.

Гіпотеза:

Модель розпізнавання навчена за методом глибинного навчання, буде розпізнавати трафік алгоритмів генерації доменів з більшою точністю відносно аналогів створених за алгоритмами поверхневого машинного навчання.

Обґрунтування новизни:

Недослідженою є можливість використання часових згорткових нейронних мереж (TCN), що є методом глибинного навчання, для своєчасного виявлення трафіку алгоритмів генерації доменних імен, що являє собою дослідницьку прогалину і надає роботі наукову новизну.

1. АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Сучасний стан та тенденції розвитку методів захисту від DGA

З урахуванням зростаючої загрози у кібербезпеці, дослідження зосереджуються на ML та його широкому наборі інструментів та прийомів для виявлення, зупинки та реагування на складні кібератаки [8]. ML може бути використано в різних сферах кібербезпеки, щоб забезпечити аналітичні підходи для виявлення та реагування на атаки. Це також може покращити процеси безпеки шляхом автоматизації рутинних завдань та полегшення аналітикам безпеки швидкої роботи з напів-автоматизованими завданнями.

1.1.1. Передумови та попередні відомості

Ботнет - це мережа комп'ютерів, підключених до Інтернету, якими керує майстер ботів віддалено за допомогою C&C каналу зв’язку (серверу). Заражений комп'ютер у мережі називається ботом. Майстер ботів використовує бота для розміщення зловмисних дій. Здебільшого розмір або структура бот-мереж може відрізнятися. Однак вони проходять однакові етапи у своєму життєвому циклі [9]. Щоб адаптуватися до нових технологій, ботнет розвивається разом із майстром ботів. Канал C&C забезпечує точку зв'язку між майстером ботів та іншими ботами для передачі даних між ними. Комунікації C&C є такими:

- Майстер ботів контактує з бот-мережами, видаючи команду;
- На основі команди ботнет виконує свою діяльність;
- Ботнет передає результати виконаних дій майстру ботів.

Ботнет можна виявити лише тоді, коли визначено місце розташування сервера C&C. Крім того, майстер ботів не зможе керувати ботнетом без встановлення надійного зв’язку між серверами C&C та компрометованими комп’ютерами. Насамперед, існує три типи архітектури C&C, засновані на способі використання комунікації, тобто централізована, децентралізована та гібридна. У централізованій архітектурі ботнетів майстер ботів керує всіма активними ботами в ботнеті з централізованого сервера C&C. Іншими

словами, бот спілкується з сервером C&C для всіх операцій, таких як отримання та відповідь на команди. Централізована архітектура мереж ботів використовує ієрархічну зіркову топологію. Інтернет-ретрансляційний чат (IRC) та протокол передачі гіпертексту (HTTP) є ключовими протоколами в централізованій архітектурі. Цей тип архітектури легше контролювати, оскільки він має лише одну центральну точку. Однак дизайн централізованої архітектури є складним. Крім того, затримка повідомлення та живучість є короткими, і шанси на збій більше, ніж у інших архітектурах. Децентралізовані архітектури або однорангові архітектури (peer-to-peer/P2P) містять більше одного C&C сервера для управління всіма активними ботами в бот-мережі. Кожен бот діє як C&C сервер, а також як клієнт. Виявлення бот-мережі, яка використовує децентралізовану архітектуру, є складним завданням, порівняно з централізованою архітектурою. Це пов’язано з використанням P2P протоколів. Дизайн децентралізованої архітектури є складним, тоді як затримка повідомлення та живучість повідомлення вища, а шанси на провал менші порівняно з іншими централізованими архітектурами. Гібридна архітектура – це поєднання централізованої та децентралізованої архітектури. Завдання виявлення та моніторингу в гібридній архітектурі складніші, ніж централізовані та децентралізовані архітектури. Однак дизайн гібридної архітектури простіший порівняно з іншими архітектурами.

Система доменних імен (DNS) – це прикладний протокол в інтернет-інфраструктурі, який має розподілену базу даних, що перехресно посилає доменні імена на відповідні їм адреси інтернет-протоколу (IP) та навпаки. DNS підтримує ієрархію управління своєю розподіленою базою даних, і вона складається з кореневого рівня, доменів верхнього рівня, доменів другого рівня, піддоменів та хостів. Доменні імена складаються з домену верхнього рівня (TLD) та домену другого рівня (SLD), які розділені крапкою. Наприклад, у YouTube.com, com – це TLD, а YouTube – SLD. Ієрархія DNS

розділена на зони, які контролюються менеджером зон. Кожен вузол в ієрархії DNS має мітку, яка включає інформацію, пов’язану з іменем домену. Ієрархія DNS на (рис. 1.1.1) – це іменований простір доменів, який схожий на eDirectory з інвертованою деревоподібною структурою. Організації можуть використовувати власний простір імен доменів для створення приватних мереж. Ці приватні мережі не видно в інтернеті. Ім'я домену, що складається з одного або декількох розділів або піддоменів, називається міткою. Шлях від піддомену до кореня називається повноцінним іменем домену. Ієрархія DNS може мати не більше 127 рівнів. Мітки розділені крапками. Кожна мітка може містити від 0 до 63 символів. Мітка довжиною 0 присвоюється кореневій зоні. Повна довжина доменного імені може містити 253 символи. Доменне ім’я переспрямовує запит кінцевого користувача до певного джерела, де інформація існує у просторі імен DNS. Вся інформація про доменні імена зберігається на DNS-сервері у вигляді ресурсних записів. Рекурсивний та нерекурсивний (або ітераційний) - це два типи DNS-серверів. Нерекурсивні DNS-сервери діють як початковий запис (Start of Authority), вони відповідають на запити всередині керованих доменів без запитів інших DNS-серверів, навіть якщо нерекурсивний DNS-сервер не може надати відповідь на запит. Рекурсивні DNS-сервери, зображені на (рис. 1.1.2), відповідають на запити всередині керованих доменів без запитів інших DNS-серверів, навіть якщо нерекурсивний DNS-сервер не може надати запитану відповідь. Рекурсивні DNS-сервери зазвичай страждають від DDoS-атак, отруєння кешу DNS, несанкціонованого використання ресурсів та погіршення продуктивності сервера кореневих імен. Крім того, запити до серверів DNS зазвичай не шифруються [10], таким чином можна виявити веб-сайти, які переглядаються. Аналіз (рис. 1.1.2) показує відмінності у роботі інтерактивних та рекурсивних DNS-серверів.

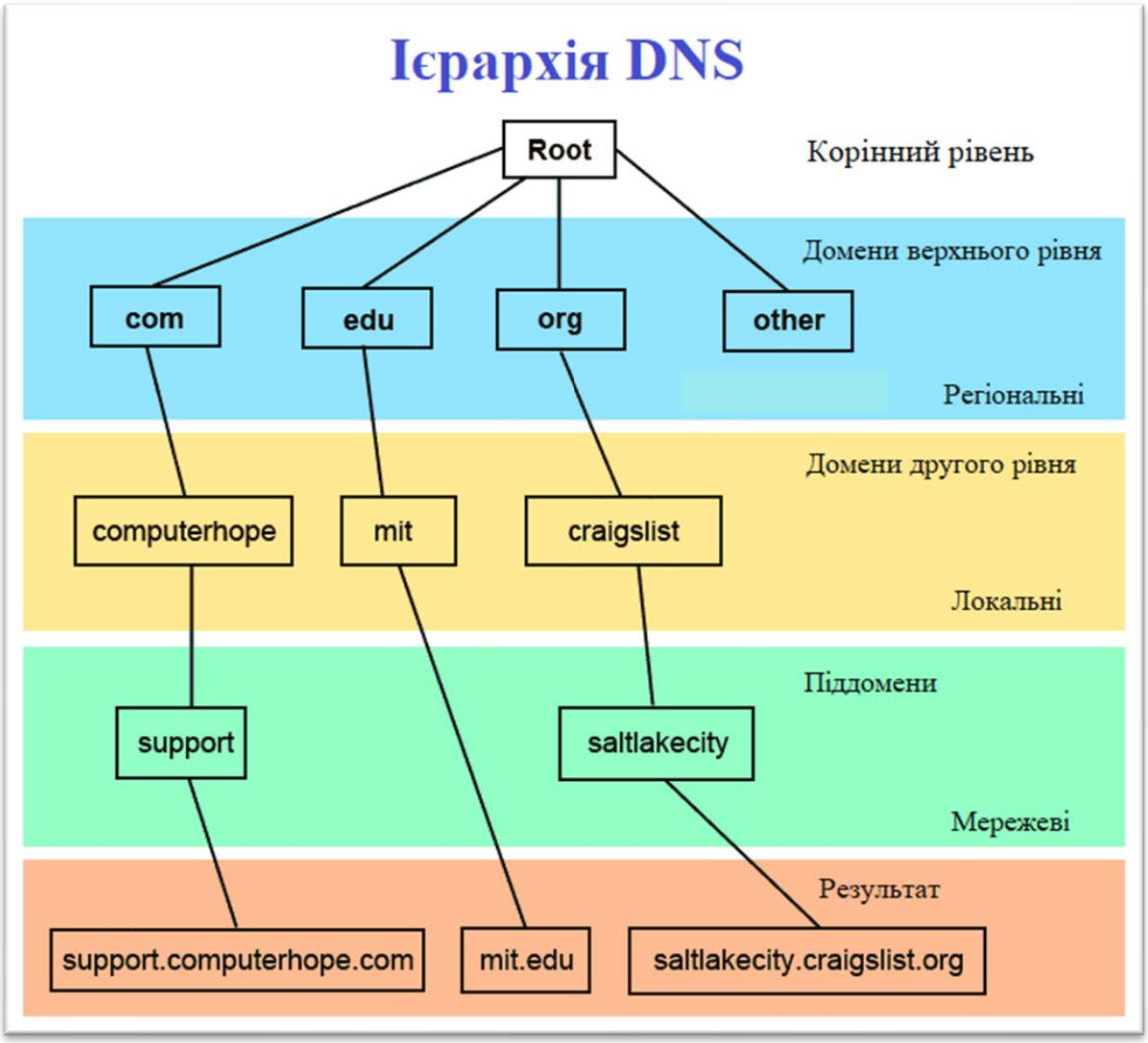

Рисунок 1.1.1 – Ієрархічна система доменних імен

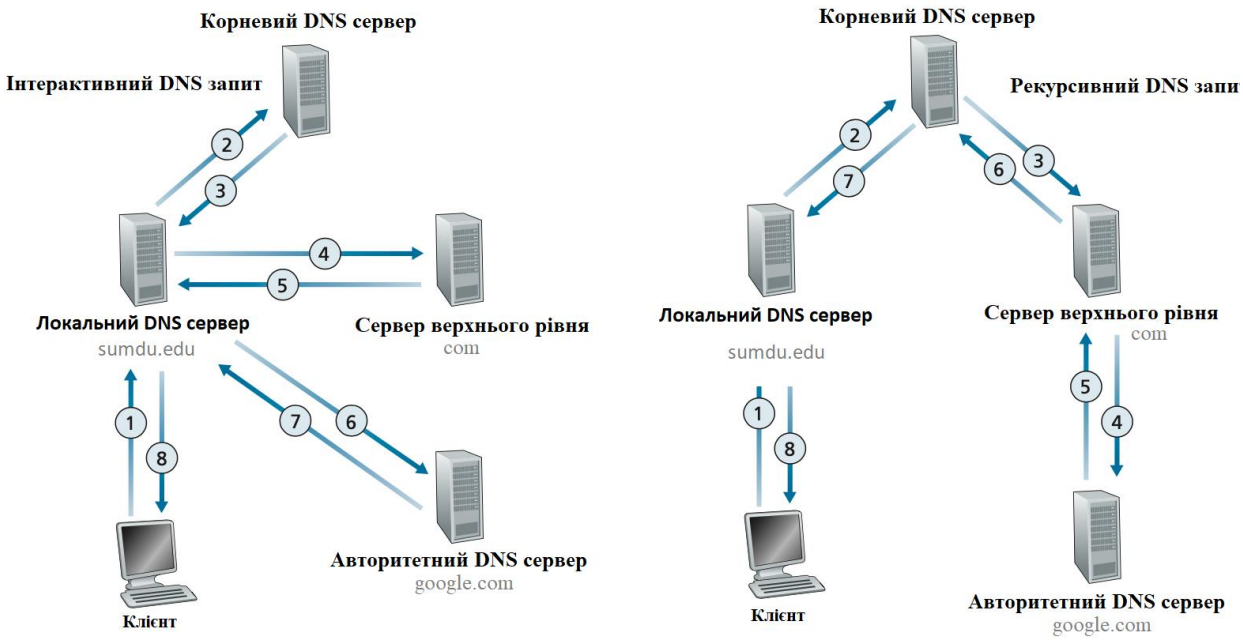

Рисунок 1.1.2 – Інтерактивні та Рекурсивні DNS-сервери

Domain fluxing – це техніка, зображена на (рис. 1.1.3), при якій повноцінні доменні імена, пов’язані з IP-адресою серверів C&C, часто змінюються, щоб підтримувати роботу ботнетів. Для здійснення цієї операції майстер ботів використовує DGA для генерації доменних імен у великих масштабах, які можуть обійти чорний список та евристичні методи виявлення доменних імен DGA. Бот використовує два домени, abc.com та def.com. Домен abc.com не зареєстрований і отримує відповідь NXDomain від DNS-сервера. def.com зареєстрований і, отже, він може зв'язатися з C&C сервером. Під час атак Domain-Flux бот-мережі зв’язуються зі своїм бот-майстром із власноруч створеними доменними іменами за допомогою методу звернення та випробування. У більшості випадків цей метод генерує кілька неіснуючих (NX) запитів для доменів, які не мають IP-адрес (NXDomains). Моніторинг трафіку DNS є важливим завданням, і допомагає виявити ботнет за допомогою аналізу DNS. Цей метод є більш ефективним, ніж статичний та двійковий аналіз шкідливих програм. Головним чином це тому, що для даних методів потрібно набагато менше часу для зворотного проектування двійкових файлів для призначення підпису [11].

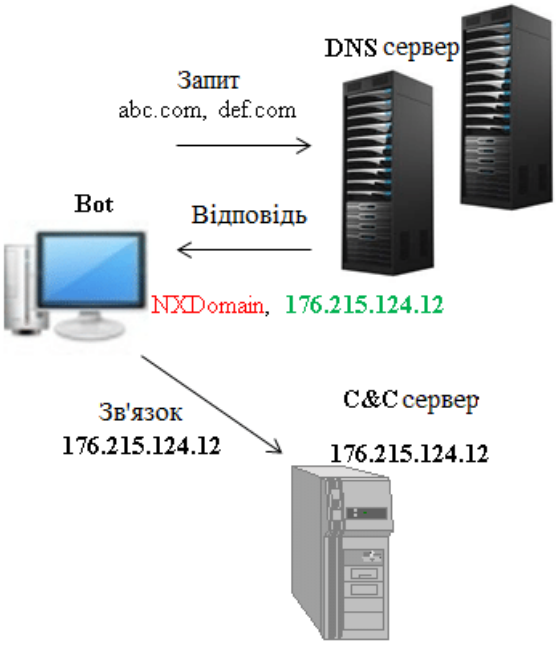

Рисунок 1.1.3 – Діаграма domain-flux атак

Діаграма на (рис. 1.1.3) пояснює, принцип роботи domain-flux атак.

Аналіз запитів DNS допомагає виявити генеровані DGA-домени, а також допомагає своєчасно відстежити ботнет і заблокувати точку зв'язку між бот-мастером та C&C сервером. Виявлення згенерованих DGA імен шляхом аналізу даних DNS має різні переваги. Наприклад, протокол DNS включає лише невелику кількість трафіку у всій мережі, що робить його придатним для аналізу навіть у великомасштабних мережах. Крім того, трафік DNS зазвичай кешований, що зменшує трафік. Деякі функції домену, такі як номер автономної системи (AS) та власник домену, можуть бути додані до функцій трафіку DNS для подальшого збільшення рівня виявлення доменних імен DGA. Крім того, аналіз запитів DNS допомагає ідентифікувати атаки на ранніх стадіях або навіть до їх виникнення.

DGA генерує велику кількість доменних імен, використовуючи початкове випадкове значення, яке може бути датою, числом або будь-якими випадковими символами. Для побудови доменних імен генератор DGA використовує комбінацію операцій бітового зсуву, xor, ділення, множення та модуля для створення послідовності символів, які дотримуються певного розподілу, наприклад нормального або рівномірного розподілу. Генератори DGA зазвичай використовують різні початкові значення, які часто змінюються. Наприклад, ці значення можуть змінюватися протягом одного дня. Це робить стратегії чорного списку неефективними, оскільки DGA постійно створюють різні нові імена доменів. Однак генеровані DGA доменні імена містять унікальні статистичні властивості, які відрізняються від справжніх доменних імен.

1.1.2. Принцип роботи DGA та проблема розпізнання

Шкідливе програмне забезпечення (ПЗ) заражає безліч пристроїв по всьому світу, і після зараження йому необхідно встановити зв'язок з C&C сервером. Якщо ця адреса чи домен жорстко закодований в шкідливій програмі, його можна легко знайти і заблокувати. З цієї та багатьох інших причин шкідливі програми використовують DGA.

DGA – це алгоритми, що зустрічаються у різних сімействах шкідливих програм, які використовуються для періодичного генерування великої кількості псевдовипадкових неіснуючих доменних імен, які можуть використовуватися як точки з’єднання з C&C сервером. Після генерації неіснуючих доменних імен, зловмисне програмне забезпечення намагається встановити з’єднання зі C&C-сервером, надсилаючи запити DNS на згенеровані доменні імена, доки один із доменів не перенаправить запит на IP-адресу сервера C&C. Згенеровані DGA доменні імена функціонують як точки з’єднання шкідливого програмного забезпечення та його C&C-сервера. Доменні імена, генеровані DGA, також відомі як алгоритмічно генеровані домени. DGA використовується для запобігання видаленню сервера C&C та перешкоджання спробам створення чорного списку.

Велика кількість потенційних точок з’єднання ускладнює правоохоронним органам ефективне вимкнення ботнетів, оскільки шкідливе ПЗ намагатиметься зв’язуватися з деякими із згенерованих доменних імен щодня, щоб отримувати оновлення або команди. Було виявлено, що автори шкідливого ПЗ Dyre зареєстрували деякі домени за 2 роки до моменту впровадження ПЗ, деякі автори реєструють домени всього за пару днів або навіть за пару годин до того, як шкідливе ПЗ починає вступати в контакт з C&C сервером. В результаті шкідлива програма може генерувати тисячі запитів, але в більшості випадків вона шукає тільки одне успішне з'єднання або один зареєстрований домен.

Використання криптографії із відкритим ключем у коді шкідливих програм робить неможливим для правоохоронних органів та інших суб'єктів імітувати команди контролерів шкідливого програмного забезпечення, оскільки деякі черв'яки автоматично відхиляють будь-які оновлення, не підписані контролерами шкідливого програмного забезпечення.

Наприклад, заражений комп’ютер може створити тисячі імен доменів, таких як: www. <ginperish> .com, і намагатиметься зв’язатись із їх частиною з

метою отримання оновлення або команд. Ця методика була популяризована сімейством шкідливих програм Conficker.a та .b, яке спочатку генерувало 250 доменних імен на день. Починаючи з Conficker.c, зловмисне ПЗ буде генерувати 50 000 доменних імен щодня, з яких воно намагатиметься зв’язуватися з 500, даючи зараженому комп'ютеру можливість 1% оновлення щодня, якщо контролери шкідливих програм реєструють лише один домен на день. Щоб заражені комп’ютери не оновлювали шкідливе програмне забезпечення, правоохоронцям потрібно було б попередньо реєструвати 50 000 нових доменних імен щодня. З точки зору власника ботнету, їм потрібно зареєструвати лише один або кілька доменів з кількох доменів, які кожен бот генерує щодня. Таку ж саму методику застосували інші автори шкідливих програм. За даними фірми, що займається мережевою безпекою, Damballa, топ-5 найпоширеніших сімей злочинних програм на базі DGA – це Conficker, Murofet, BankPatch, Bonnana та Bobax станом на 2011 рік [12].

DGA також може поєднувати слова зі словника для створення доменів. Ці словники можуть бути закодовані у зловмисному програмному або взят із загальнодоступного джерела [13]. Домени, створені словниковою DGA, як правило, важче виявити через їх схожість із легітимними доменами.

1.1.3. Застосування ML та заходи безпеки від DGA атак

Алгоритми ML можуть бути реалізовані в додатках для виявлення і реагування на кібератаки, перш ніж вони зададуть явного впливу [14]. Зазвичай це досягається за допомогою моделі, розробленої шляхом аналізу наборів великих даних про події безпеки та виявлення шаблону зловмисних дій. Як результат, коли виявляються подібні дії, вони автоматично обробляються. Набір навчальних даних моделей, як правило, складається з попередньо виявлених та записаних індикаторів скомпрометованості (IOC) (фрагменти шкідливих даних, таких як дані, знайдені в записах або файлах системного журналу, що ідентифікують потенційно шкідливу діяльність у системі або мережі). Крім того, завдяки наявності наборів даних IOC, ми

можемо використовувати алгоритми класифікації ML для виявлення різних видів поведінки шкідливих програм у наборах даних та відповідно їх класифікувати, таким чином IOC використовуються для побудови моделей та систем, які можуть контролювати, ідентифікувати та реагувати на загрози в режимі реального часу. Були проведені дослідження на основі методів поведінкового аналізу, які використовують методи кластеризації та класифікації ML для аналізу поведінки тисяч шкідливих програм [6]. Це дослідження дає можливість використовувати вивчені шаблони для автоматизації процесу виявлення та класифікації нових шкідливих програм. Також, це може допомогти аналітикам безпеки або іншим автоматизованим системам швидко ідентифікувати та класифікувати новий тип загрози та відповісти на неї відповідно, використовуючи рішення, прийняті на основі даних. Наприклад, використовуючи історичний набір даних, що містить докладні події атак-вимагачів WannaCry, модель ML може навчитися ідентифікувати подібні атаки, тим самим даючи можливість автоматизувати процес ідентифікації та реагування на подібні атаки. Методи ML також використовувались у класифікації трафіку IP [15] [16], а це може допомогти автоматизувати процес систем виявлення вторгнень, які можуть бути використані для виявлення поведінкових моделей, як у випадку DDOS-атак. Зі збільшенням числа технік ML інші дослідження були зосереджені на аналізі безлічі рішень ML для систем виявлення вторгнень, включаючи одиночні, гібридні та ансамблеві класифікації [17].

При оцінці та визначенні ризиків у мережі використовуються кількісні показники для присвоєння оцінок ризику різним розділам мережі, тим самим допомагаючи організаціям розставити пріоритети своїм ресурсам кібербезпеки відповідно до різних показників ризику. Машинне навчання може бути використано для автоматизації цього процесу шляхом аналізу історичних наборів даних про кібератаки та визначення того, які ділянки мереж в основному брали участь у певних типах атак. Використання ML є

вигідним у тому сенсі, що отримані оцінки базуватимуться не лише на знаннях доменів мереж, а найголовніше, що оцінки базуватимуться на реальних даних. Цей показник може допомогти кількісно визначити ймовірність та вплив нападу на певну область мережі і, таким чином, може допомогти організаціям зменшити ризик стати жертвами нападів. Були проведені дослідження щодо використання алгоритмів ML, таких як K-Nearest Neighbor, Support Vector Machines та алгоритми Random Forest, для аналізу та кластеризації мережевих активів на основі їх зв’язку [18]. Інші дослідження зосереджувались на тому, як пристрої інтернету речей, підключені до малих та середніх підприємств, можуть бути використані для обхідних атак на ці підприемства [19]. Розроблені системи ML, які використовують принцип взаємного підсилення для аналізу величезних обсягів оповіщень в організаційних мережах для визначення показників ризику з урахуванням асоціацій різних мережевих об'єктів [20].

Імена доменів DGA [21] можна заблокувати, використовуючи чорні списки, але охоплення цих чорних списків є або поганим (загальнодоступні чорні списки), або надзвичайно непослідовним (чорні списки комерційних постачальників) [22]. Методи детектування належать до двох основних класів: реакційного і в режимі реального часу. Реакційне виявлення спирається на неконтрольовані методи кластеризації та контекстну інформацію, наприклад відповіді мережі NXDOMAIN [11], інформація WHOIS [23] та пасивний DNS [24], щоб зробити оцінку легітимність імен доменних імен. Нещодавні спроби виявити доменні імена DGA за допомогою методів DL були надзвичайно успішними, показники F1 перевищували 99% [25]. Ці методи DL, як правило, використовують архітектури довгої короткочасної пам'яті (LSTM) та згорткової нейронної мережи (CNN) [26], хоча глибинні вбудовані слова - це тип подання слів, який дозволяє словам з подібним значенням мати подібне подання. показали велику перспективу для виявлення словника DGA [27]. Однак ці підходи DL можуть бути вразливими

до методів шкідливого машинного навчання (Adversarial Machine Learning) [28] [29].

Також машинне навчання може бути використано для автоматизації повторюваних завдань, що виконуються аналітиками з безпеки під час діяльності з безпеки. Це можна зробити за допомогою аналізу записів / звітів про минулі дії аналітиків безпеки для успішної ідентифікації та реагування на певні атаки та використання цих знань для побудови моделі, яка може ідентифікувати подібні атаки та відповідати відповідно без втручання людини. Хоча важко автоматизувати повний процес безпеки та повністю замінити аналітика безпеки, проте є деякі аспекти аналізу, які машинне навчання може автоматизувати, включаючи виявлення шкідливого програмного забезпечення, аналіз мережевих журналів та оцінку вразливості, наприклад аналіз мережевого ризику. Включаючи машинне навчання у робочий потік безпеки, «людина і машина» можуть об’єднати зусилля та виконувати речі зі швидкістю та якістю, які в іншому випадку були б неможливими. З експоненціальним зростанням штучного інтелекту, все більша кількість завдань стає автоматизованою. Спокусливо думати, що штучний інтелект (AI) підвищить автоматизацію, а певні завдання, які в даний час виконують люди, переймуть на себе машини. Це може бути правдою в деяких випадках, проте є численні випадки, коли поєднання AI та людської діяльності дають набагато кращі результати, ніж кожен з них сам по собі. Саме з цієї причини в даний час можна спостерігати зростання компаній що використовують AI, та зосереджені не тільки на створенні продукту AI для автоматизації завдань, але і створенні продуктів, що підвищують і доповнюють продуктивність людських аналітиків. Відомим прикладом такої компанії є Palantir [30], яка створює продукти, що полегшують аналітикам агрегування та використання величезних обсягів даних.

В інших працях з метою посилення діяльності аналітиків безпеки були проведені дослідження щодо використання алгоритмів машинного навчання, таких як генетичні алгоритми та спроби прийняття рішень щодо створення додатків, що генерують правила класифікації мережевих з'єднань [31]. Інші підходи стосуються впровадження когнітивної архітектури для створення автоматизованої системи прийняття рішень в галузі кіберзахисту, що має здібності експертного рівня, натхненну тим, як люди міркують і навчаються [32]. Аналітикам кібер-безпеки, як правило, доводиться витрачати час, реагуючи на численні події, які іноді включають помилкові спрацьовування, що в основному виявляється марною тратою свого часу. Проведено дослідження, щоб показати, що класифікатори машинного навчання можна навчити за даними попереджень, щоб ідентифікувати та розрізнити помилкові спрацьовування від справжніх спрацьовувань, тим самим даючи можливість створити автоматизовану систему, яка буде попереджати аналітика лише за сценаріями, що включають справжні позитивні дані [33].

1.2. Аналіз та класифікація алгоритмів машинного навчання для розпізнавання трафіку алгоритмів генерації доменів

В останні роки дослідженню ідентифікації DGA приділяється велика увага. Від методів, що спираються на підході конструювання ознак [34], заснованих на ML, до застосування DL.

1.2.1. Огляд алгоритмів DL

Всі алгоритми DL засновані на глибинних нейронних мережах (DNN), які представляють собою великі нейронні мережі, організовані на багатьох рівнях, здатних до навчання автономному поданням, (рис. 1.2.1).

Контрольовані алгоритми DL:

– Нейронна мережа прямого поширення (FNN). Це варіант DNN, де кожен нейрон пов'язаний з усіма нейронами попереднього шару. FNN робить ніяких припущень щодо вхідних даних і надає

гнучке універсальне рішення для класифікації за рахунок високих обчислювальних витрат;

- Згорткові нейронні мережі (CNN) з прямим зв'язком. Це варіант DNN, де кожен нейрон отримує вхідні дані тільки від підмножини нейронів попереднього шару. Ця характеристика робить CNN ефективними при аналізі просторових даних, але їх продуктивність знижується при застосуванні до непросторових даними. У CNN нижча вартість обчислень, ніж у FNN;
- Рекурентні глибокі нейронні мережі (RNN). Варіант DNN, де зв'язки між елементами утворюють спрямовану послідовність. Завдяки цьому з'являється можливість обробляти серії подій у часі або послідовні просторові ланцюжки. RNN можуть використовувати свою внутрішню пам'ять для обробки послідовностей довільної довжини. Тому мережі RNN застосовні в таких завданнях, де щось цілісне розбите на частини, наприклад: розпізнавання рукописного тексту або розпізнавання мови. Останнім часом найбільшого поширення набули моделі довгої короткочасної пам'яті (LSTM) та вентильного рекурентного вузла (GRU).

Неконтрольовані алгоритми DL:

- Глибинні мережі переконань (DBN). Вони моделюються за допомогою композиції обмежених машин Больцмана (RBM), класу нейронних мереж без вихідного шару. DBN можна успішно використовувати для завдань попереднього навчання, оскільки вони чудові в функції вилучення ознак. Вони вимагають фази навчання, але з наборами даних без міток;
- Складені автоенкодери (CAE). Вони складаються з декількох автоенкодерів, класу нейронних мереж, в яких кількість вхідних і вихідних нейронів однаково. CAE відмінно справляється з

завданнями попереднього навчання, як і DBN, і досягає кращих результатів на невеликих наборах даних.

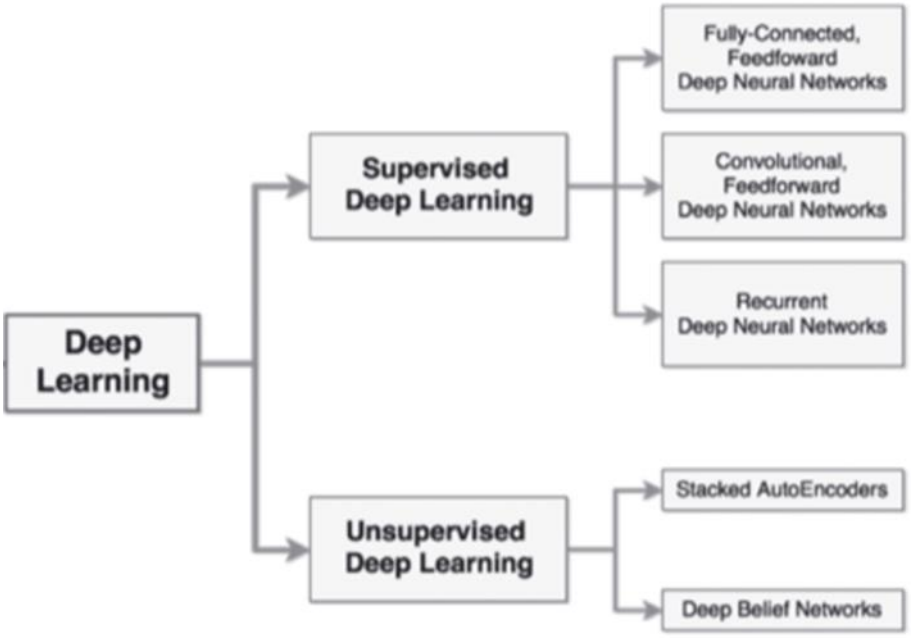

Рисунок 1.2.1 – Діаграма алгоритмів DL

1.2.2. Застосування алгоритмів DL для розпізнавання DGA

Загальні засоби захисту від зловмисних доменів DGA включають чорні списки [23, 35], класифікатори випадкового лісу (RF) [27, 36, 37] та методи кластеризації [11, 26]. Коли списки добре підтримуються, а ознаки відбираються ретельно, ці методи мають прийнятну ефективність. Однак як чорні списки, так і ці моделі мають серйозні обмеження: по-перше, вони покладаються на відібрані вручну ознаки, що вимагають багато часу для розробки, а по-друге, вони позбавлені можливості узагальнення за допомогою декількох впроваджених вручну ознак і вимагають постійного технічного обслуговування. Необхідні більш вичерпні тактики для виявлення безперервних нових DGA, що походять від шкідливого програмного забезпечення на основі мережі.

Останні інновації з використанням глибокого навчання мають найсучаснішу точність виявлення DGA. Такі моделі відрізняються високою гнучкістю і мають доведений успіх у складних мовних завданнях. Вони не

вимагають ознак оброблених вручну, які вимагають багато часу і їх легко уникнути. Автори [38] першими представили мережу довгострокової короткочасної пам’яті (LSTM) для класифікації DGA. Пізніше були застосовані інші архітектури, такі як подальші варіації LSTM [37, 38, 39, 40, 41], CNN [42, 43] та гібридної моделі CNN-LSTM [44]. Хоча ці класифікатори успішні для доменів DGA із випадковими символами, вони в основному були неефективними при ідентифікації доменів DGA словників. Ці моделі також добре працюють на різних наборах тестувань, але їх продуктивність може постраждати при спробі узагальнення на нові сімейства DGA або нові версії раніше відомих моделі сімейств.

В дослідженнях глибинного навчання було розглянуто питання виявлення доменів DGA. Автори [41] використали п’ять репрезентативних моделей класифікації зображень на основі CNN для класифікації доменів DGA: Alex Net, VGG, Squeeze Net, Inception та ResNet. Їхні експериментальні результати показали високу продуктивність, навіть незважаючи на те, що вони використовували моделі, специфічні для зображення, зокрема модель Inception v4. У роботі [23], автори порівняли складені моделі CNN та паралельні моделі CNN, використовуючи символи доменів DGA як вхідні дані. Їх експериментальні результати показали, що обидві моделі CNN перевершували моделі машинного навчання, такі як RF та багатошаровий персептрон, особливо паралельні CNN, які мали вищу точність.

Моделі RNN, зокрема моделі LSTM, також вивчались у галузі виявлення доменів DGA. Автори [45] класифікували виявлення DGA на дві категорії – ретроспективне виявлення та виявлення в реальному часі, та запропонували модель LSTM для виявлення в реальному часі. Експериментальні результати показали, що ефективність моделі LSTM без вилучення ознак була кращою, ніж у випадкового лісу з ручним вилученням ознак. У роботі [28] застосовано механізм уваги до існуючої моделі LSTM і продемонстрував, що важливість кожного символу в доменному імені

відрізняється в класифікації доменів DGA. Та експериментально класифікувано 15 типів доменних імен DGA та не DGA і показано, що запропонований метод перевершує не лише застарілі моделі машинного навчання, але й моделі LSTM. Хоча було багато досліджень щодо виявлення доменів DGA шляхом застосування LSTM [27, 37, 45, 46], усі вони використовують виключно LSTM з односпрямованою інформацією, що є основним обмеженням, оскільки вони не використовують всю наявну вхідну інформацію в поточному стані. У роботі [47] автори пропонують модель виявлення на основі BiLSTM, яка використовує високопродуктивні моделі LSTM і вивчає двонаправлену інформацію.

У роботі [25], автори використали просту модель LSTM для ідентифікації доменів DGA, які мали високий рівень ефективності, за винятком класів DGA, що нагадують англійські слова. А у [48] запропоновано нову модель, засновану на LSTM, з метою виявлення ботнетів, які використовують DGA для генерації великої кількості доменів як C&C сервери. У [42] автори запропонували модель на основі CNN для виявлення доменних імен DGA. Але вони не перевірили ефективність моделі на основі DGA, що складається зі списку слів, яка є різновидом DGA, що імітує склад та методи іменування звичайних доменних імен, та ускладнює їх виявлення. Наприклад, доменне ім'я, таке як dbkortrdffa.com (генерується шкідливим програмним забезпеченням), є значно підозрілішим, ніж домен, такий як mightyapple.net (згенерований DGA Suppobox), ці методи були в першу чергу навчені виявляти випадкові рядки, що призводить до низького ступеня виявлення DGA на основі списку слів. Проблема виявлення доменів DGA на основі списку слів є доволі складною та актуальною.

1.3. Постановка задачі

Головне завдання роботи – побудувати модель даних та навчити обрану модель розпізнання з подальшою оптимізацією її параметрів для

максимізації розпізнавання трафіку алгоритмів генерації доменів (DGA) за тестовою вибіркою. В результаті буде отримано технологію розпізнання трафіку DGA, яка буде корисною для розробників засобів захисту від кібератак та компаній що бажають посилити захист всередині власної мережі.

Для досягнення поставленої мети сформульовані наступні задачі:

– проаналізувати сучасні моделі і методи машинного навчання для захисту та розпізнавання трафіку DGA;
– обрати засоби та технології для реалізації;
– сформувати опис ознак початкової вибірки;
– запропонувати моделі аналізу даних;
– виконати обробку вхідного набору даних, нормалізацію, оптимізацію, навчання моделі та тестування;
– зробити висновки.

Для цього необхідно виконати такі етапи:

– підібрати достовірний набір даних;
– побудувати модель даних;
– проаналізувати ти оптимізувати (нормалізувати) вхідні дані;
– навчити модель на тренувальному наборі даних;
– валідувати модель на тестовому наборі даних;
– обчислити метрики та побудувати графіки.

Таким чином, було проаналізовано сучасний стан і тенденції розвитку методів захисту від DGA, також було проведено аналіз алгоритмів машинного навчання для розпізнавання трафіку алгоритмів генерації доменів і класифікацію методів глибинного навчання в цілому. Та було сформульовано постановку задачі.

2. ОПИС ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ТРАФІКУ АЛГОРИТМІВ ГЕНЕРАЦІЇ ДОМЕНІВ

2.1. Вибір мови програмування

Насправді, немає найкращої мови для машинного навчання, кожна мова гарна там, де вона найкраще підходить та є більш доречною. Також не існує єдиної мови машинного навчання як найкращої мови для машинного навчання. Однак, безумовно, є деякі мови програмування, які більше підходять для завдань машинного навчання, ніж інші. Багато інженерів машинного навчання вибирають мову машинного навчання, виходячи з тієї бізнес-проблеми, над якою вони працюють. Наприклад, більшість інженерів машинного навчання воліють використовувати Python для проблем NLP(Natural Language Processing), водночас віддаючи перевагу використовувати R або Python для завдань аналізу тональності тексту, а деякі, ймовірно, використовують Java для інших завдань машинного навчання, таких як безпека та виявлення загроз.

Python – це найпопулярніша високорівнева мова програмування з динамічною семантикою. Він досить простий для роботи і читання: його використання знижує вартість розробки та обслуговування програм. Python вважається найпростішим мовою програмування, а саме тому він найпоширеніший. По суті, машинне навчання – це технологія, яка допомагає додаткам на основі штучного інтелекту навчатися і видавати результати автоматично, без людського втручання.

Робота фахівця по машинному навчання полягає у тому, що він повинен збирати, систематизувати і аналізувати дані, а потім на основі отриманої інформації створювати алгоритми для штучного інтелекту. Python найкраще підходить для виконання таких завдань, тому що він досить зрозумілий в порівнянні з іншими мовами. Більш того, у нього відмінна

продуктивність при обробці даних. Одна з основних причин, чому Python використовується для машинного навчання полягає в тому, що у нього є безліч фреймворків, які спрощують процес написання коду і скорочують час на розробку. У наукових розрахунках використовується Numpy, в просунутих обчисленнях - SciPy, в добуванні і аналізі даних - SciKit-Learn. Ці бібліотеки працюють в таких фреймворків, як TensorFlow, CNTK і Apache Spark. Ще одна перевага Python – це велика підтримка і якісна документація. Існує безліч корисних ресурсів про Python, на яких програміст може отримати допомогу і консультацію, перебуваючи на будь-якому етапі розробки.

Для початку роботи з проектами ML та DL дуже важливо мати певні бібліотеки, API та середовища. Google активно бере участь у дослідження ML і пропонує хмарне середовище розробки під назвою Colaboratory, відоме як Google Colab, яке є безкоштовним для загального користування. Ця хмарна платформа на основі браузера дозволяє створювати складні моделі на великих масивах даних за допомогою Jupyter Notebook. Платформа пропонує оболонку Python для виконання коду Python. Різні функціональні можливості, що надаються Google Colab, перелічені нижче:

- Безкоштовна послуга для використання графічних процесорів і TPU (тензорних процесорних одиниць): програми ML і DL мають великі потреби в обчисленнях і для адекватного виконання вимагають потужних машин, які називаються GPU або TPU. Однак ці машини дорогі. Google Colab пропонує безкоштовне використання графічного процесора, який може працювати постійно протягом 12 годин;
- Середовище розробки: більшість популярних бібліотек, таких як NumPy, Pandas, Matplotlib, Scikit-learn, попередньо встановлені в Google Colab. Він також пропонує підтримку TensorFlow та Keras, а також PyTorch та Caffe. Більше того, будь-яку іншу

бібліотеку Python можна встановити в Google Colab за допомогою команди !pip install <ім'я_бібліотеки>;

– Завантаження / імпорт наборів даних: Програми ML та DL вимагають великого набору даних для навчання. Це можуть бути загальнодоступні набори даних або створені спеціально для аналізу та навчання. Google Colab пропонує функціонал для монтування Google Drive за допомогою коду авторизації. Набір даних також можна завантажити з GitHub або Kaggle;
– Команди терміналу: набір команд терміналу можна безпосередньо виконувати за допомогою комірки Google Colab. Знак оклику (!) Потрібно поставити як префікс перед кожною командою - наприклад, !ls, !pwd тощо.

2.2. Моделі та методи екстракції опису ознак розпізнавання трафіку алгоритмів генерації доменів

По-перше, постає питання навчальної вибірки. Набір даних для навчання можна отримати аналізуючи та компануючи набори даних для розпізнавання трафіку DGA, що розміщені у відкритому доступі та(або) використані у схожих роботах. Для роботи нам потрібні справжні дані, що містять як різноманітний сучасний звичайний мережевий трафік, так і різноманітні сценарії вторгнень із поглибленою структурованою інформацією про трафік мережі. Наша вибірка буде складатися з 2 класів. Перший – легітимний (Legit), був узятий зі списку реальних доменних імен Alexa Top Million. Другий – DGA, був складений з комбінування доменів 360 Lab DGA Domains(колекція доменів, згенерованих DGA, її підтримує та оновлює щодня 360 – китайський постачальник безпеки) та списку доменів Bambenek DGA feeds створений автором, шляхом зворотної розробки алгоритмів генерації шкідливих доменних імен, взятих з примірників шкідливих програм, існуючих в мережі Інтернет.

Для формування вибірки із необроблених даних буде використовуватися наступний список параметрів:

– Тип домену (легітимний чи DGA);
– Доменне ім’я;
– Тип сімейства DGA (для легітимних доменів - відсутній).

Сама класифікація буде проводитись за принципом 75 до 25, тобто навчання відбуватиметься на 75% вихідних даних, а тестування алгоритму на тих, що залишилися 25%.

2.3. Алгоритм машинного навчання для розпізнавання трафіку алгоритмів генерації доменів

Насправді, для створення універсального класифікатора, здатного ідентифікувати трафік DGA в режимі реального часу, добре пасує така архітектура нейронної мережі, що може аналізувати інформацію, отриману з попередніх кроків для аналізу даних на поточному кроці навчання. Такі властивості присутні у RNN (рис. 2.3.1). Рекурентні нейронні мережі, головним чином, відрізняються наявністю циклу. Вони дозволяють зберігати і використовувати інформацію, отриману з попередніх кроків нейронної мережі. Кожен домен в нашому випадку розглядається як послідовність символів з фіксованого словника, яка подається на вхід RNN. Навчання такої нейронної мережі проводиться методом зворотного поширення помилки таким чином, щоб максимізувати ймовірність правильного вибору відповідного класу. Однак, на практиці виявляється, що якщо розрив між минулою інформацією і справжньою досить великий, то цей зв'язок втрачається, і подібна мережа не здатна її обробляти. Вирішення цієї проблеми було знайдено в 1997 році вченими Hochreiter та Schmidhuber. У роботі [49] вони запропонували нову модель, а саме LSTM (рис. 2.3.2), яку через багато років потому використали автори [25]. В даний час дана модель має багато різних модифікацій та широко використовується для розв'язання

різноманітних класів задач, таких як: розпізнавання мови, обробка природних мов та ін. LSTM складається з ряду постійно пов'язаних підмереж, відомих як блоки пам'яті. Замість одного шару нейронної мережі, в LSTM використовується 4 шари, взаємодіючих особливим чином. Гарним прикладом LSTM є вентильний рекурентний вузол (Gated Recurrent Unit, GRU) (рис. 2.3.3), який використовувався у [23] та [26]. GRU – це вентильний механізм у рекурентних нейронних мережах, представлений 2014 року. Він подібний до LSTM з вентилем забування, але мають менше параметрів, оскільки не мають вентиля виходу. Було виявлено, що їхня продуктивність на моделюванні поліфонічної музики та мовленнєвого сигналу аналогічна продуктивності LSTM.

Дивлячись на досягнення RNN та SL у сфері виявлення DGA доцільно постає питання щодо можливості ще більше покращити якість розпізнавання проблеми DGA. І відповідь на це питання може дати доволі новий підхід для розпізнання відхилень у часових рядах – часові згорткові мережі (TCN) (рис. 2.3.4). TCN – це структура, яка використовує випадкові звивини та розширення, саме тому вона адаптивна до послідовних даних із темпоральністю та великими рецептивними полями. Часовий ряд – це ряд точок даних, проіндексованих (або перелічених, або відкладених на графіку) в хронологічному порядку. Найчастіше часовий ряд є послідовністю, взятою на рівновіддалених точках в часі, які йдуть одна за одною. У роботі [50] 2016 року були розроблені Часові згорткові мережі (TCN) для сегментації дій на основі відео. Два етапи цього звичайного процесу включають: по-перше, обчислення функцій низького рівня з використанням (зазвичай) CNN, що кодує просторово-часову інформацію, а по-друге, введення цих функцій низького рівня в класифікатор, який фіксує тимчасову інформацію високого рівня за допомогою (зазвичай) RNN. Основним недоліком такого підходу є те, що для нього потрібні дві окремі моделі. TCN забезпечує уніфікований підхід до ієрархічного захоплення всіх двох рівнів інформації.

До недавнього часу тема моделювання послідовностей в контексті глибокого навчання в основному була пов'язана з повторюваними архітектурами нейронних мереж, такими як LSTM та GRU. У [51] автори припускають, що такий спосіб мислення застарілий і що CNN слід враховувати як одного з основних кандидатів при моделюванні послідовних даних. Вони змогли показати, що згорткові мережі можуть досягти кращої продуктивності, ніж RNN, у багатьох завданнях, уникаючи загальних недоліків повторюваних моделей, таких як проблема вибуху/зникнення градієнта або нестача пам'яті. Крім того, використання CNN замість RNN може призвести до поліпшення продуктивності, оскільки вона дозволяє паралельно обчислювати результати. Архітектура, яку вони пропонують використовувати і є TCN. Доведено, що CNN є особливо корисними для вилучення ознак високого рівня в структурних даних.

У цій роботі буде використано TCN для виявлення трафіку DGA. Ми тренуємо TCN на створеному наборі даних і використовуємо його для прогнозування тенденції за кілька часових кроків.

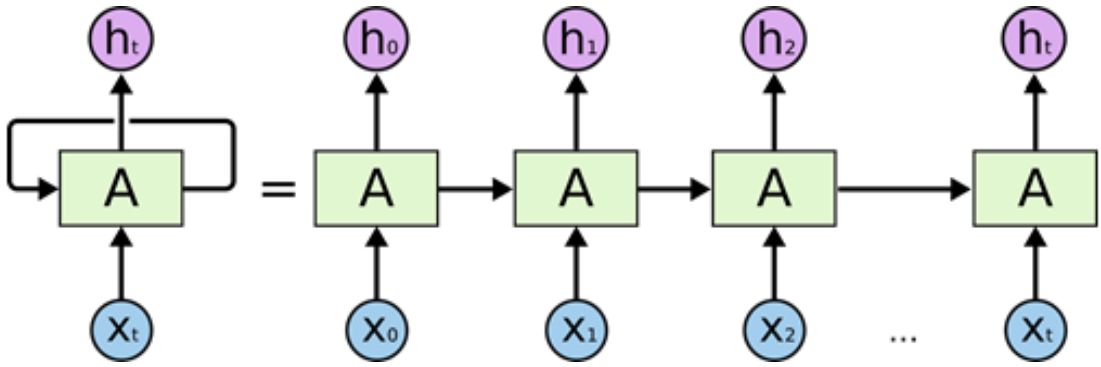

Рисунок 2.3.1 – Схема рекурентної нейронної мережі

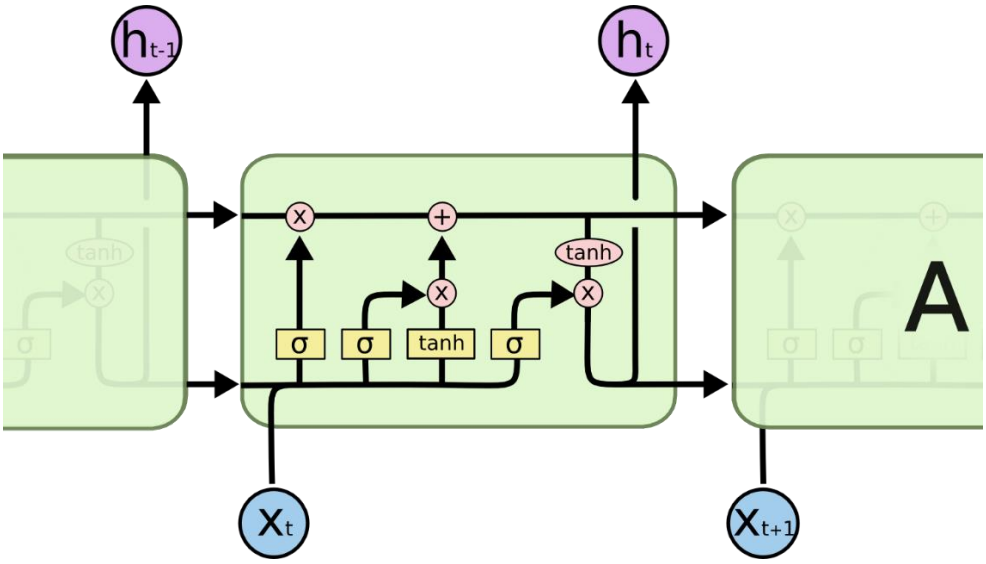

Рисунок 2.3.2 – Схема моделі LSTM, що містить чотири взаємодіючих шари

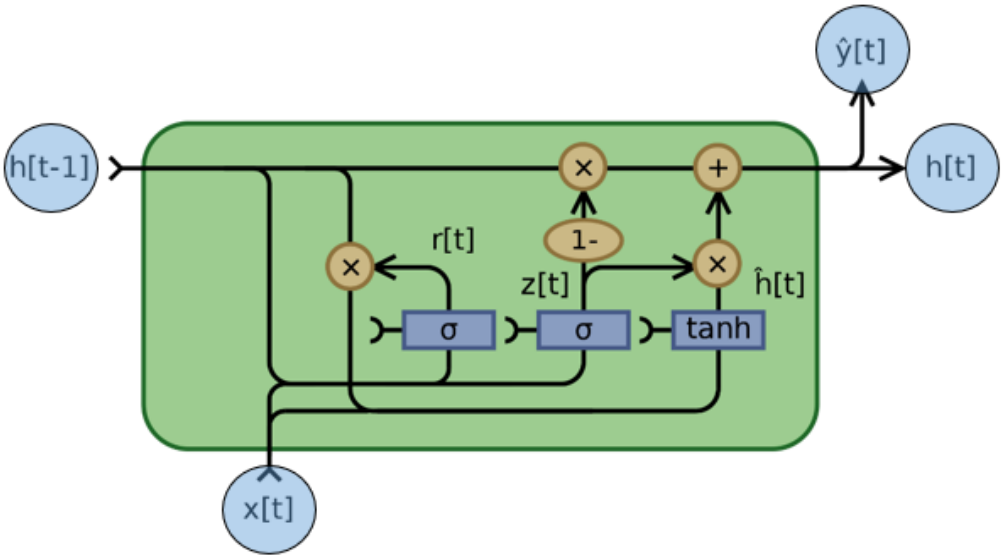

Рисунок 2.3.3 – Схема моделі GRU

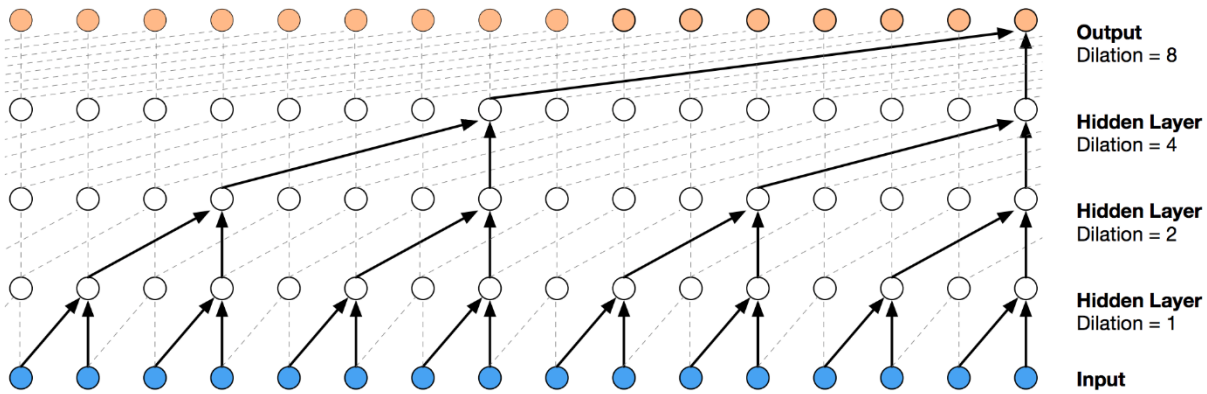

Рисунок 2.3.4 – Схематична візуалізація стека TCN

2.4. Критерії валідації алгоритмів машинного навчання

Після навчання моделі нам потрібно оцінити, наскільки точно класифікатор може передбачити цільовій результат: нормальна чи анормальна мережева активність, базуючись на відібраних ознаках. Також ми використовували різні алгоритми машинного навчання, з різноманітною підготовкою даних та гіперпараметрів. Саме завдяки критеріям валідації ми зможемо порівняти різні алгоритми навчання та моделі даних.

Як основа використовується Позитивно-Негативна техніка валідації:

- **Positive** (**P**) – Позитивні розпізнання;
- **Negative** (**N**) – Негативні розпізнання;
- **True positive** (**TP**) – Справжні позитивні: це позитивні розпізнання, які правильно було визначено класифікатором;
- **True nagative** (**TN**) – Справжні негативні: це негативні розпізнання, які були правильно визначені класифікатором;
- **False positive** (**FP**) – Хибні позитивні: це негативні розпізнання, які неправильно було визначені позитивними;
- **False negative** (**FN**) – Хибні негативні: це позитивні розпізнання, які були неправильно визначені як негативні;

Ці значення підсумовуються у матриці помилок (confusion matrix) (табл. 2.4.1). Вона підсумовує, наскільки класифікатор може розпізнати класи даних. Таким чином, помилки класифікації бувають двох видів: False Negative (FN) і False Positive (FP).

Таблиця 2.4.1 – Confusion matrix

	$\hat{y} = 1$ (Predicted Positives)	$\hat{y} = 0$ (Predicted Negatives)
y = 1 (Positives)	TP	FN
y = 0 (Negatives)	FP	TN

$\hat{y}$ - це відповідь алгоритму на об'єкті розпізнавання, а y - справжня мітка класу на цьому об'єкті.

У (табл. 2.4.2) описані різні допоміжні показники оцінки класифікатора при прогнозуванні.

Таблиця 2.4.2 – Міри оцінки

Міра	Формула	Визначення
Accuracy (точність розпізнавання)	$\frac{TP+TN}{P+N}$	Міра розпізнавання до загальної кількості. Тобто частка правильних відповідей алгоритму.
Коефіцієнт помилок	$\frac{FP+FN}{P+N}$	Похибка розпізнавання (1 – Accuracy)
Повнота (чутливість, Recall)	$\frac{TP}{P}$	Справжня позитивна ставка, показує, яку частку справжніх позитивних об'єктів класу з усіх об'єктів позитивного класу знайшов алгоритм.
Специфічність	$\frac{TN}{N}$	Справжня негативна ставка, показує, яку частку об'єктів негативного класу з усіх об'єктів негативного класу знайшов алгоритм.
Точність (Precision)	$\frac{TP}{TP+FP}$	Міра точності (тобто, який відсоток позитивних, позначених позитивом, насправді такий)
Втрата (Loses)	$MSE = \frac{1}{N}\sum_{(x,y)\in D}(y - prediction(x))^2$	Втрата - це покарання за поганий прогноз. Тобто, втрата - це число, яке

		вказує, наскільки поганим було прогнозування моделі на окремому прикладі. Якщо прогноз моделі ідеальний, втрата дорівнює нулю; інакше втрати більші. Метою навчання моделі є пошук набору ваг та упереджень, які мають низькі втрати в середньому для всіх прикладів.
F-міра, середнє гармонійне значення точності і повноти	$\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$	Агрегований критерій якості, альтернативний спосіб використання точності та повноти, поєднуючи їх разом
F_β, де β – негативне дійсне число	$(1 + \beta^2) \cdot \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$	Зважений показник точності та повноти

Вивчення параметрів функції прогнозування і тестування її на одних і тих же даних є методологічною помилкою. Модель, яка буде просто повторювати мітки зразків, які вона щойно побачила, матиме ідеальну оцінку, до того моменту коли не зможе передбачити нові дані не видимі до цього. Така ситуація називається перенавчанням. Щоб цього уникнути, при проведенні експерименту з машинним навчанням зазвичай використовується частина наявних даних у вигляді набору тестів. Але існує вірогідність, що якась частина даних буде використовуватися тільки як тренувальній набір і ніколи не буде протестований. Щоб вирішити цю проблему, ще одна частина набору даних може бути представлена як так званий «набір перевірки»:

навчання триває на навчальному наборі, після чого виконується оцінка на перевірочному наборі, і коли експеримент здається успішним, остаточна оцінка може бути зроблена на тестовому наборі. Вирішенням цієї проблеми є процедура перехресного затверджування (Cross-validation).

Перехресне затверджування – це метод оцінки аналітичної моделі і її поведінки на незалежних даних. При оцінці моделі наявні дані розбиваються на k частин. Потім на k-1 частинах даних проводиться навчання моделі, а частина, що залишилася даних використовується для тестування. Процедура повторюється k разів; в результаті кожна з k елементів даних використовується для тестування. В результаті виходить оцінка ефективності обраної моделі з найбільш рівномірним використанням наявних даних.

Для правильної оцінки якості роботи створеного класифікатора буде використано перехресне затверджування та пораховані основні метрики що використовуються в задачах класифікації (Precision, Recall та F-міра).

Таким чином, було зроблено наступне: обрано мову програмування для реалізації інформаційної системи; описано метод формування початкової вибірки даних для навчання моделі; обґрунтовано та обрано алгоритм машинного навчання для створення моделі розпізнання; проведено опис критеріїв валідації алгоритму навчання які будуть застосовані для оцінки якості роботи створеної інформаційної системи.

3. РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ТРАФІКУ АЛГОРИТМІВ ГЕНЕРАЦІЇ ДОМЕНІВ НА ОСНОВІ ГЛИБИННОГО МАШИННОГО НАВЧАННЯ

3.1. Формування навчальних та тестових вибірок

Навчання моделі буде проводитися на наборах даних, які представлені у вигляді .csv файлів з даними (в кожному з яких кома виступає в ролі роздільного знаку між ознаками), де кожна новий рядок – це новий набір даних. Файли можна відкрити для перегляду простим текстовим редактором. Дані легітимних доменів(не DGA) були взяті з Alexa Top Million, а дані DGA були скомбіновані з двох джерел доменів: 360 Lab DGA та списку доменів Bambenek DGA feeds. При чому дані Alexa Top Million представлені у csv файлі, а дані DGA у звичайних текстових файлах, тому спочатку необхідно обробити дані(привести їх до однакового виду) та виділити основні ознаки даних. В результаті обробки даних будо сформовано 3 початкових фала даних: nonDGA_domains.csv, 360_dga_domains.csv та bambenek_dga_domains.csv статистичний опис вмісту яких відображено на (рис. 3.1.1 – 3.1.3) відповідно.

	Вид_DGA	Домен	Тип
count	1000000	1000000	1000000
unique	1	1000000	1
top	Відсутній	aknology.com	Легітимний
freq	1000000	1	1000000

Рисунок 3.1.1 – Статистичний опис nonDGA_domains.csv

	Вид_DGA	Домен	Тип
count	1380024	1380024	1380024
unique	52	1380024	1
top	banjori	pbgmvinskycattederifg.com	DGA
freq	469990	1	1380024

Рисунок 3.1.2 – Статистичний опис 360_dga_domains.csv

	Вид_DGA	Домен	Тип
count	830202	830202	830202
unique	35	828202	1
top	banjori	fycejfsotscqwcwgaibmdnpbkf.ru	DGA
freq	439223	2	830202

Рисунок 3.1.3 – Статистичний опис bambenek_dga_domains.csv

Далі файли що містять DGA необхідно об’єднати та видалити дублікати. В результаті отримано файл allDGA_domains.csv (рис. 3.1.4). Щоб отримати кінцевий набір даних необхідно об’єднати файл легітимних доменів nonDGA_domains.csv з даними файлу усіх DGA. Створений файл (all_domains.csv) містить 2689083 записів та 3 стовпця (Вид_DGA, Домен, Тип). Приклад перших десяти рядків файлу all_domains.csv зображено на (рис. 3.1.6).

	Вид_DGA	Домен	Тип
count	1689083	1689083	1689083
unique	69	1671886	1
top	banjori	ekwxmwiugkeq.biz	DGA
freq	470169	2	1689083

Рисунок 3.1.4 – Статистичний опис allDGA_domains.csv

	Вид_DGA	Домен	Тип
count	2689083	2689083	2689083
unique	70	2671886	2
top	Відсутній	wglqssuiwcymao.biz	DGA
freq	1000000	2	1689083

Рисунок 3.1.5 – Статистичний опис all_domains.csv

	Вид_DGA	Домен	Тип
1	Відсутній	google.com	Легітимний
2	Відсутній	youtube.com	Легітимний
3	Відсутній	facebook.com	Легітимний
4	Відсутній	baidu.com	Легітимний
5	Відсутній	wikipedia.org	Легітимний
6	Відсутній	amazon.com	Легітимний
7	Відсутній	qq.com	Легітимний
8	Відсутній	yahoo.com	Легітимний
9	Відсутній	taobao.com	Легітимний
10	Відсутній	tmall.com	Легітимний

Рисунок 3.1.6 – Перші 10 рядків файлу all_domains.csv

Після створення файлів даних потрібно застосувати певну попередню обробку, щоб зробити дані придатними для навчання нейронної мережі. У нас є 1 мільйон доменів не DGA та 2 мільйони доменів DGA загалом. Згідно з цим великим списком, який містить усі наявні дані, формується словник дійсних символів. Дійсні символи – це унікальні символи, що містяться в даних. Отримавши дійсні символи, ми можемо розрахувати максимальну кількість ознак для даних. Це необхідно для представлення тексту домену у

вигляді числової послідовності. Тобто кількість ознак дорівнює кількості дійсних символі в словнику плюс один символ, а саме пустий символ, що представляє собою відступ для тих доменів, довжина яких менша за довжину максимального домену. Отримавши словник, ми можемо перетворити символи в послідовності. Нейронні мережі нічого не можуть зробити з ознаками. Градієнтний спуск і метод зворотного поширення помилки працюють з числами. Щоб перетворити домени в числа достатньо співставити кожному знаку доменного імені відповідний номер зі словника (рис. 3.1.7).

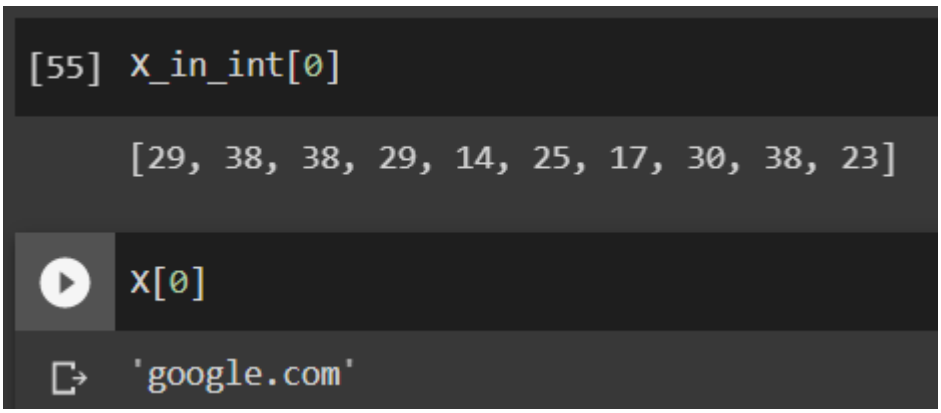

Рисунок 3.1.7 – Приклад представлення доменного імені у вигляді чисел

Через вимогу до форми вхідних нейронних мереж, яка повинна бути однаковою, нам потрібно переконатися що всі вибірки (домени) мають однакову довжину. Є два варіанти:

– Обрати довжину, яка охоплює більшість зразків;
– Обрати довжину, яка охоплює всі зразки.

Плюси і мінуси відповідно зростають. Наприклад для першого випадку у ситуації, коли довжини даних є незбалансовано-розподіленими, тобто більшість даних мають помірну довжину, і лише невелика частина має велику довжину. Графік цього розподілу буде виглядати як логнормальний розподіл (рис. 3.1.8).

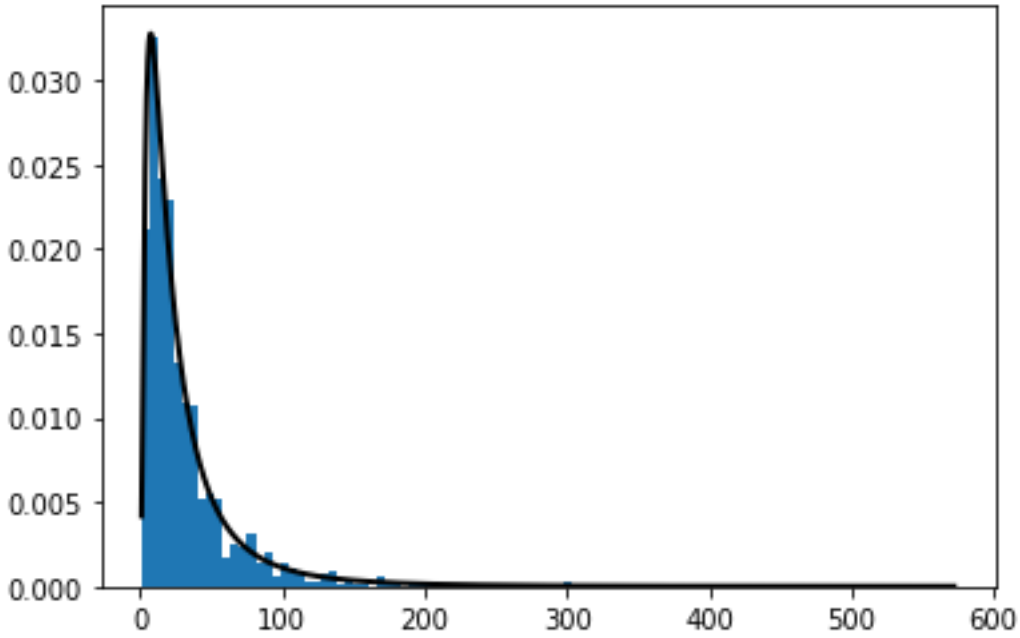

Рисунок 3.1.8 – Приклад логнормального розподілу

У такому випадку, коли ми вибираємо довжину, що не охоплює всі довжини доменних імен, довжина відступу може бути набагато меншою, ніж при виборі максимальної довжини, яка охоплює всі довжини даних. Оскільки така довжина заповнення може охоплювати більшість даних, ми все одно можемо добре представляти дані, не втрачаючи занадто багато інформації (лише ті надто довгі дані будуть відрізані), а це означає, що обравши таку довжину, ми можемо уникнути упередженості (що э неминучим) моделі, яку потрібно навчити. Тим часом менша довжина заощадить нам багато обчислювальних ресурсів. Проте найбільша доменна довжина нашої моделі становить 73 знаки, що є не великим числом для даної моделі. Таким чином, було вирішено встановити 73 як довжину заповнення. В результаті отримано проста, впорядкована комбінація легітимних та DGA доменів, яка представлена у вигляді чисел. І останнім кроком необхідно створити маску приналежності до DGA, тобто співставити результуючій впорядкованій комбінації відповідне значення приналежності до DGA (значення 0 чи 1, де 0 – не належить, а 1 – належить до DGA).

Отже, в результаті отримано два набори (масиви) даних, кожне значення першого являє собою числове представлення доменних імен, а значення другого набору (масиву) даних відповідають числовому значенню приналежності до DGA, значень першого набору.

3.2. Короткий опис програмного забезпечення

Основною мовою програмування для всіх створюваних додатків в дипломному проекті обраний Python. Це високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читання коду. Вибір Python, як мови програмування, не випадковий. Ця мова має низький поріг входження, можливість, без особливих проблем, запуститися на будь-який операційній системі (ОС) і, найголовніше, на цій мові написано безліч бібліотек для побудови різних моделей.

Pandas – це дуже відомий пакет для роботи зі структурованими даними, що мають відкритий код та ліцензування BSD. Ця бібліотека – це високопродуктивні, прості у використання структури даних та інструменті аналізу даних.

NumPy – один із найпопулярніших програмних пакетів Python. Це ефективний та зручний інструмент для роботи із векторними та матричними операціями. Завдяки присутності корисним вбудованим функціям, дозволяє уникнути перевантаження коду. Багато інших бібліотек можуть напряму взаємодіяти з NumPy. Pandas – не менш відомий пакет для роботи зі структурованими даними, що мають відкритий код та ліцензування BSD. Ця бібліотека – це високопродуктивні, прості у використання структури даних та інструменті аналізу даних.

Pyplot Matplotlib та Seaborn – чудовий набір для візуалізації даних. Методи цих двох бібліотек дозволяють легко конфігурувати будь-які типи графіків, гістограм і т.ін. Це, мабуть, найкращий спосіб графічного

представлення даних. Seaborn є надбудовою над базовою бібліотекою візуалізації Matplotlib. Модуль pyplot - це колекція функцій в стилі команд, яка дозволяє використовувати Matplotlib майже так само, як MATLAB. Кожна функція pyplot працює з об'єктами Figure і дозволяє їх змінити. Наприклад, є функції для створення об'єкта Figure, для створення областей побудови, представлення ліній, додавання міток і так далі. Pyplot є залежним від стану (stateful). Він відслідковує статус об'єкта Figure і його області побудови. Функції виконуються на поточному об'єкті.

TensorFlow – це комплексна платформа з відкритим вихідним кодом для машинного навчання, розроблена компанією Google. Вона має всеосяжну гнучку екосистему інструментів, бібліотек, ресурсів та спільнот, які дозволяють дослідникам просувати новітні досягнення в області машинного навчання, а розробникам легко створювати і розгортати додатки на основі машинного навчання. Платформа спочатку розроблена командою Google Brain і використовується в сервісах Google для розпізнавання мови, виділення обличь на фотографіях, визначення схожості зображень, відсіювання спаму в Gmail, підбору новин у Google News і організації перекладу з урахуванням смислу. Розподілені системи машинного навчання можна створювати на типовому обладнанні, завдяки вбудованій підтримці в TensorFlow рознесення обчислень на кілька CPU або GPU. TensorFlow добре підходить для автоматизованої анотації зображень в таких системах як DeepDream. Також з 26 жовтня 2015 року Google використовує систему RankBrain для збільшення релевантності ранжування пошукової видачі Google. RankBrain заснований на TensorFlow. TensorFlow дозволяє проводити навчання генеративно-змагальних мереж (GAN). Інтеграція TensorFlow з Python забезпечується дистрибутивом Anaconda. Іншими застосуванням є використання у складі програм FakeApp з метою безшовного поєднання фото та відеозображень для створення підробних, але правдоподібних відео, відомих під назвою Deepfake. TensorFlow надає бібліотеку готових

алгоритмів чисельних обчислень, реалізованих через графи потоків даних (data flow graphs). Вузли в таких графах реалізують математичні операції або точки входу/виводу, в той час як ребра графа представляють багатовимірні масиви даних (тензори), які перетікають між вузлами. Вузли можуть бути закріплені за обчислювальними пристроями і виконуватися асинхронно, паралельно обробляючи разом все підходящі до них тензори, що дозволяє організувати одночасну роботу вузлів в нейронної мережі за аналогією з одночасною активацією нейронів в мозку.

Scikit-learn – найкращий набір функціоналу для машинного навчання. Бібліотека має великий вибір компонент для навчання із вчителем та без нього. Не дивлячись на об’ємну та складну предметну частину, він забезпечує зручний, зрозумілий та добре описаний інтерфейс для роботи з бібліотеками та функціями. Ще однією перевагою є наявність великої кількості навчальних посібників та довідкової інформації, в яких використовуються алгоритми класичного машинного мислення. Саме завдяки документації та великій кількості прикладів, ці бібліотеки використовуються як професіоналами, так і початківцями, які щойно знайомляться з машинним навчанням.

3.3. Результати машинного навчання інформаційної технології розпізнавання трафіку алгоритмів генерації доменів на основі глибинного машинного навчання

Було обрано дві моделі TCN. Перша являє собою модель для завдання регресії, а друга – модель завдання класифікації кіновідгуків залежно від оцінки IMDB. Моделі було навчено на основі платформи TensorFlow, кількість поколінь навчання першої становить – 2, а другої моделі – 10.

Для навчання моделі загальний набір даних було розділено на тестові дані та дані для тренування з використанням функції train_test_split бібліотеки Scikit-learn у співвідношенні 75% до 25% відповідно. Загалом,

таке співвідношення ми можемо забезпечити через великий набір даних, що містить приблизно 3 мільйони зразків. Перед початком навчання необхідно виконати дії, вказані у пункті 3.1, бо значення знаходяться в різних одиницях виміру.

В результаті навчання моделей та тестуванні їх на тестовому наборі було отримано відповіді моделей, що зображені на (рис. 3.3.1 – 3.3.2), та на основі яких було пораховано основні метрики оцінки класифікаторів при прогнозуванні, значення яких відображені у (табл. 3.3.1 – 3.3.2). А на (рис. 3.3.3 – 3.3.4) зображено залежність точності (Accuracy) від епохи навчання моделі.

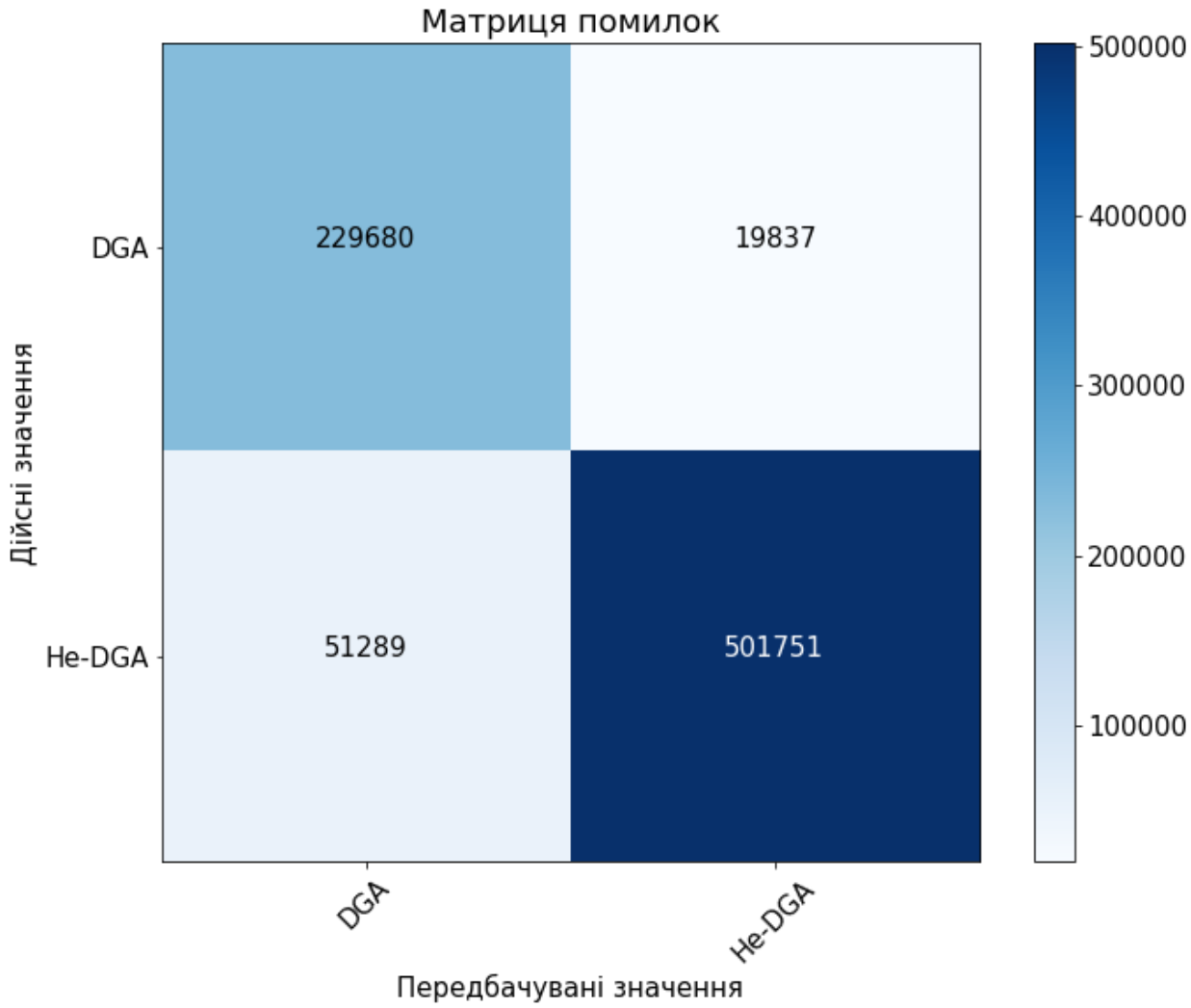

Рисунок 3.3.1 – Матриця помилок першої моделі

Таблиця 3.3.1– Основні метрики для першої моделі

Міра	Формула	Отримане значення
Accuracy (точність розпізнавання)	$\frac{TP+TN}{P+N}$	0.9114
Коефіцієнт помилок	$\frac{FP+FN}{P+N}$	0.0886
Повнота (чутливість, Recall)	$\frac{TP}{P}$	0.8174
Специфічність	$\frac{TN}{N}$	0.9619
Точність (Precision)	$\frac{TP}{TP+FP}$	0.9204
Втрата (Loses)	$MSE = \frac{1}{N}\sum_{(x,y)\in D}(y - prediction(x))^2$	0.0657
F-міра, середнє гармонійне значення точності і повноти	$\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$	0.86592295
F_β, де β – негативне дійсне число	$(1 + \beta^2) \cdot \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$	0.93381338

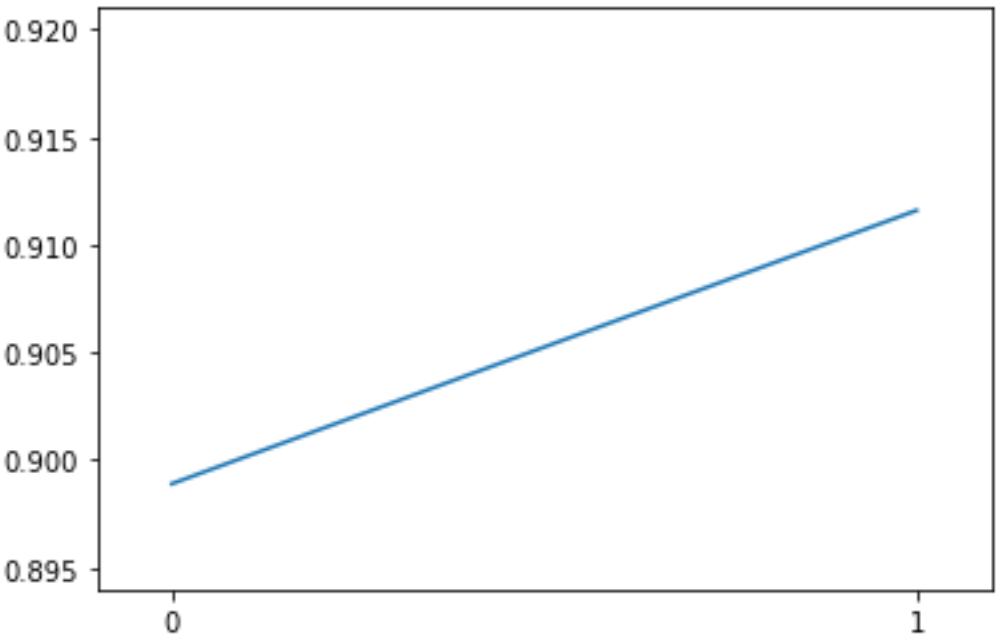

Рисунок 3.3.2 – Графік точності першої моделі залежно від епохи навчання

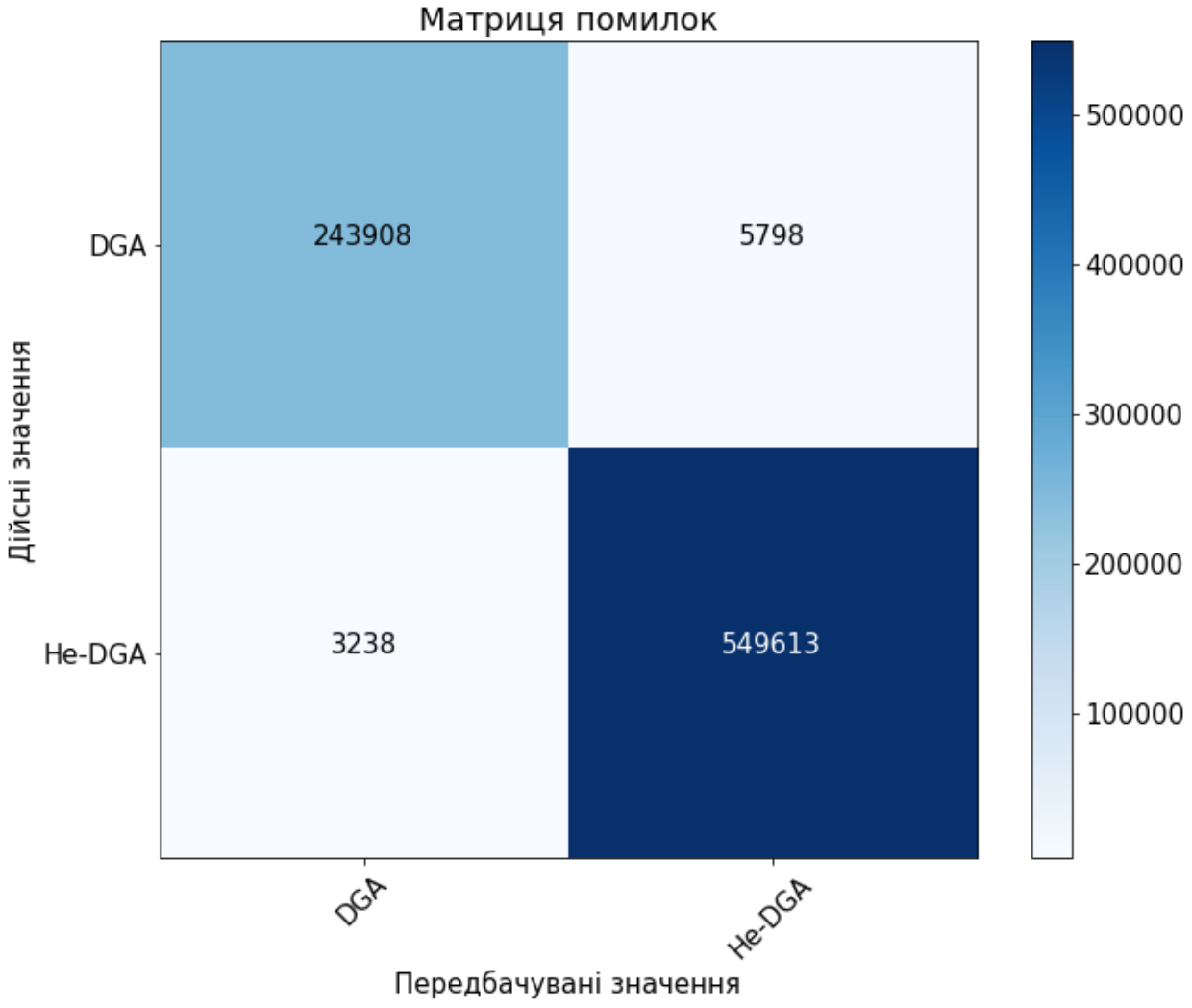

Рисунок 3.3.3 – Матриця помилок другої моделі

Таблиця 3.3.2 – Основні метрики для другої моделі

Міра	Формула	Отримане значення
Accuracy (точність розпізнавання)	$\frac{TP+TN}{P+N}$	0.9887
Коефіцієнт помилок	$\frac{FP+FN}{P+N}$	0.0113
Повнота (чутливість, Recall)	$\frac{TP}{P}$	0.9869
Специфічність	$\frac{TN}{N}$	0.9941
Точність (Precision)	$\frac{TP}{TP+FP}$	0.9768
Втрата (Loses)	$MSE = \frac{1}{N}\sum_{(x,y)\in D}(y - prediction(x))^2$	0.03616
F-міра, середнє гармонійне значення точності і повноти	$\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$	0.9818135
F_β, де β – негативне дійсне число	$(1 + \beta^2) \cdot \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$	0.99184669

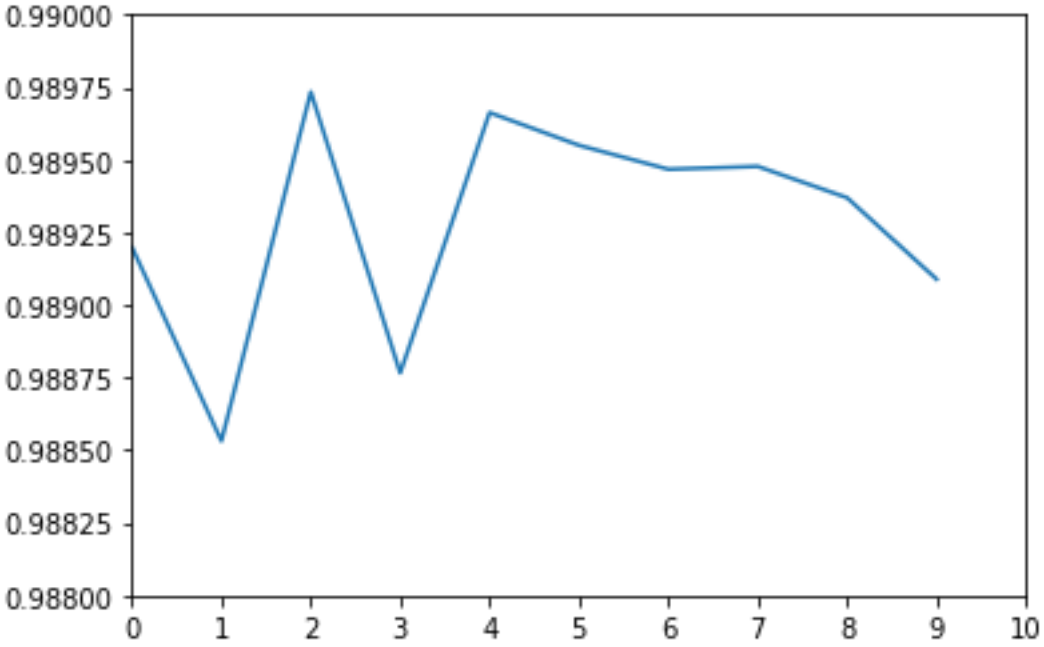

Рисунок 3.3.4 – Графік точності другої моделі залежно від епохи навчання

Таким чином, було сформовано навчальну та тестову вибірки для обраних моделей; проведено короткий опис програмного забезпечення, яке було використано у роботі; показано детальні результати навчання обраних моделей.

В результаті можна сказати, що обидві моделі мають дуже високий рівень правильного розпізнання на тестовій вибірці. Такі показники значно кращі за показники відомих моделей машинного навчання, як навчених методами поверхневого навчання так і методами глибинного навчання. Друга модель, а саме модель для завдання класифікації кіновідгуків залежно від оцінки IMDB показала дещо кращі результати за модель для завдання регресії, а саме точність розпізнання – 98,87 %, Precision – 98,69%, Recall – 98,69% та F міра – 99,18%. Значення показників першої моделі дещо гірші за значення другої, а це може бути зумовлено тим, що кількість поколінь навчання першої моделі – 2, на відміну від кількості поколінь навчання другої моделі – 10. Підвищення кількості поколінь може призвести до покращення результатів розпізнання.

ВИСНОВКИ

DGA – це алгоритми, що зустрічаються у різних сімействах шкідливих програм, які використовуються для періодичного генерування великої кількості псевдовипадкових неіснуючих доменних імен. Згенеровані DGA доменні імена функціонують як точки з'єднання шкідливого програмного забезпечення та його C&C-сервера, який в свою чергу відсилає команди зараженому пристрою з метою викрадення конфіденційної інформації та завдання шкоди. Доменні імена, генеровані DGA, також відомі як алгоритмічно генеровані домени. DGA використовується для запобігання видаленню сервера C&C та перешкоджання спробам створення чорного списку. Аналізуючи мережевий трафік, ми можемо розпізнати атакуючі дії зловмисників та вчасно вжити заходи для зменшення шкідливого впливу.

За результатами аналізу наявних методів та рішень щодо розпізнання трафіку DGA, було виявлено, що для вирішення проблеми розпізнання в цілому використовуються моделі RNN, а саме LSTM та GRU і їх різновиди та комбінації з іншими моделями. Також було встановлено, що моделі часових згорткових мереж (TCN), які виникли нещодавно і показують кращі результати за LSTM і GRU. Тож було прийнято рішення використати TCN для проблеми розпізнавання трафіку алгоритмів генерації доменів.

В ході дослідження було обрано засоби та технології для реалізації шляхом аналізу предметної області; сформовано початкову вибірку та опис її ознак, запропоновано модель валідації вибірки і обрано моделі для навчання, а саме моделі часових згорткових мереж (TCN), що засновані на основі глибинного машинного навчання.

Для розробки інформаційної технології були використанні різноманітні засоби та знання, зокрема:

- методи глибинного машинного навчання;
- методи аналізу великих даних;
- хмарне середовище розробки Google Colab;

– мова Python;
– бібліотеки машинного навчання TensorFlow та SciKit-Learn.

У результаті було реалізовані продумані методи та технології для ефективної роботи інформаційної технології, що дозволяє виявляти домени згенеровані DGA. Було побудовано дві моделі даних та навчено їх з подальшою оптимізацією параметрів для максимізації розпізнавання трафіку алгоритмів генерації доменів (DGA) за тестовою вибіркою. В результаті було отримано інформаційну технологію розпізнання трафіку DGA, яка буде корисною для розробників засобів захисту від кібератак та компаній що бажають посилити захист всередині власної мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Larson S. 10 biggest hacks of 2017. 2017, December 20. Retrieved: November 3, 2018.

[2] Rimo T., Walth M. “McAfee and CSIS: Stopping Cybercrime Can Positively Impact World Economies”, McAfee, June 9, 2014. URL: https://www.businesswire.com/news/home/20140609005306/en/McAfee-and-CSIS-Stopping-Cybercrime-Can-Positively-Impact-World-Economies.

[3] Cuthbertson A. Ransomware attacks have risen 250 percent in 2017, hitting the U.S. hardest. 2017, May та . Retrieved: September 21, 2018.

[4] Worldwide Revenue for Security Technology Forecast to Surpass $100 Billion in 2020, According to the New IDC Worldwide Semiannual Security Spending Guide. 2016, October 12. Retrieved: September 21, 2018. URL: https://www.businesswire.com/news/home/20161012005102/en/Worldwide-Revenue-Security-Technology-Forecast-Surpass-100.

[5] Berman D. S., Buczak A. L., Chavis J. S., Corbett C. L. A Survey of Deep Learning Methods for Cyber Security. 10, 122,2019.

[6] Rieck K., Trinius P., Willems C., Holz T. Automatic analysis of malware behavior using machine learning. Journal of Computer Security, 19(4), 639-668, 2011.

[7] Deng L., Yu D. Deep learning: Methods and applications. Found. Trends Signal Process. 7, 197–387, 2014.

[8] Fraley J. B., Cannady J. The promise of machine learning in cybersecurity. In SoutheastCon, 2017, pp. 1-6. IEEE. March 2017.

[9] Silva S. S., Silva R. M., Pinto R. C., Salles R. M. Botnets: A survey. Computer Networks, vol. 57, no. 2, pp. 378-403, 2013.

[10] Jolfaei A., Vizandan A., Mirghadri A. Image encryption using HC-128 and HC-256 stream ciphers. International Journal of Electronic Security and Digital Forensics, vol. 4, no. 1, pp. 19-42, 2012.

[11] Antonakakis M., Perdisci R., Nadji Y., Vasiloglou N., Abunimeh S., Lee W., Dagon D. From throw-away traffic to bots: detecting the rise of dga-based malware. In Proceedings of the 21st USENIX conference on Security symposium, 2012.

[12] Damballa. Top-5 Most Prevalent DGA-based Crimeware Families. p. 4. 2011, URL: https://www.damballa.com/downloads/r_pubs/WP_DGAs-in-the-Hands-of-Cyber-Criminals.pdf

[13] Plohmann D., Yakdan K., Klatt M., Bader J., Gerhards-Padilla E. A Comprehensive Measurement Study of Domain Generating Malware. 25th USENIX Security Symposium, pp. 263–278, 2016.

[14] Dolev S., Lodha S. Cyber Security Cryptography and Machine Learning. In Proceedings of the First International Conference, CSCML 2017, Beer-Sheva, Israel, June 29-30, 2017.

[15] Nguyen T., Armitage G. A survey of techniques for Internet traffic classification using machine learning. IEEE Communications Surveys and Tutorials, 10(4), 56-76. 2008.

[16] Zander S., Nguyen T., Armitage G. Automated traffic classification and application identification using machine learning. In Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on (pp. 250-257). IEEE, November 2005.

[17] Tsai C. F., Hsu Y. F., Lin C. Y., Lin W. Y. Intrusion detection by machine learning: A review. Expert Systems with Applications, 36(10), 11994-12000, 2009.

[18] Arora D., K. Li F., Loffler A. Big data analytics for classification of network enabled devices. In Advanced Information Networking and Applications Workshops (WAINA), 2016 30th International Conference on pp. 708-713. IEEE, March, 2016.

[19] Saleem J., Adebisi B., Ande R., Hammoudehs M. A state of the art survey-Impact of cyber-attacks on SME's. In Proceedings of the International Conference on Future Networks and Distributed Systems. p. 52, ACM, July, 2017.

[20] Hu X., Wang T., Stoecklin M. P., Schales D. L., Jang J., Sailer R. Asset risk scoring in enterprise network with mutually reinforced reputation propagation. In 2014 IEEE Security and Privacy Workshops (SPW), pp. 61-64. IEEE, May, 2014.

[21] Shateel A. Chowdhury. DOMAIN GENERATION ALGORITHM – DGA IN MALWARE. Aug 30, 2019. URL: https://hackersterminal.com/domain-generation-algorithm-dga-in-malware/

[22] Kührer M., Rossow C., Holz T. Paint It Black: Evaluating the Effectiveness of Malware Blacklists. Research in Attacks, Intrusions and Defenses, Springer International Publishing, 8688, pp. 1–21, 2014.

[23] Curtin R. R., Gardner A. B., Grzonkowski S., Kleymenov A., Mosquera A. Detecting DGA domains with recurrent neural networks and side information. In Proceedings of the 14th Int'l Conference on Availability, Reliability, and Security, pp. 1–10, Canterbury, UK, August 2019.

[24] Pereira M., Coleman Sh., Yu B., Cock M. De, Nascimento A. Dictionary Extraction and Detection of Algorithmically Generated Domain Names in Passive DNS Traffic. Research in Attacks, Intrusions, and Defenses, Lecture Notes in Computer Science, 11050, Springer International Publishing, pp. 295–314, 2018.

[25] Woodbridge J., Anderson H., Ahuja A., Grant D. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. 2016.

[26] Yu B., Pan J., Hu J., Nascimento A., Cock M. Character Level based Detection of DGA Domain Names. 2018 International Joint Conference on Neural Networks (IJCNN). Rio de Janeiro: IEEE: 1–8., 2018.

[27] Koh J. J., Rhodes B. Inline Detection of Domain Generation Algorithms with Context-Sensitive Word Embeddings. 2018 IEEE International Conference on Big Data (Big Data). Seattle, WA, USA: IEEE: 2966–2971, 2018.

[28] Anderson H., Woodbridge J., Bobby F. DeepDGA: Adversarially-Tuned Domain Generation and Detection. 2016.

[29] Sidi L., Nadler A., Shabtai A. MaskDGA: An Evasion Attack Against DGA Classifiers and Adversarial Defenses. in IEEE Access, vol. 8, pp. 161580-161592, 2020.

[30] About. (n.d.). Retrieved: November 03, 2018, URL: http://www.palantir.com/.

[31] Sinclair C., Pierce L., Matzner S. An application of machine learning to network intrusion detection. In Computer Security Applications Conference, 1999. (ACSAC'99) Proceedings. 15th Annual (pp. 371-377). IEEE, 1999.

[32] Benjamin D. P., Pal P., Webber F., Rubel P., Atigetchi M. Using a cognitive architecture to automate cyberdefense reasoning. In Bioinspired Learning and Intelligent Systems for Security, 2008. BLISS'08. ECSIS Symposium on (pp. 58-63). IEEE, 2008, August.

[33] Zomlot L., Chandran S., Caragea D., Ou X. Aiding intrusion analysis using machine learning. In Machine Learning and Applications (ICMLA), 2013 12th International Conference on (Vol. 2, pp. 40-47). IEEE, 2013, December.

[34] Zhauniarovich Y., Khalil I., Yu T., Dacier M. A survey on malicious domains detection through dns data analysis. ACM Comput Surveys (CSUR) 51(4):67. 2018.

[35] Chaz L., Platon K., Davide B., Juan C., Antonakakis M. A lustrum of malware network communication: evolution and insights. 2017.

[36] Ahluwalia A., Traore I., Ganame K., Agarwal N. Detecting broad length algorithmically generated domains. 2017.

[37] Yu B., Gray D., Pan J., Cock M. D., Nascimento A. C., Inline DGA detection with deep networks. In Proceedings of the 2017 IEEE Int'l Conference on Data Mining Workshops, pp. 683–692, New Orleans, LO, USA, November 2017.

[38] Lison P., Mavroeidis V. Automatic detection of malware-generated domains with recurrent neural models. 2017.

[39] Akarsh S., Sriram S., Poornachandran P., Menon V. K., Soman K. P. Deep learning framework for domain generation algorithms prediction using long short-term memory. 2019.

[40] Vinayakumar R., Soman K. P., Poornachandran P. Detecting malicious domain names using deep learning approaches at scale. 2018.

[41] Tran D., Mac H., Tong V., Tran H. A., Nguyen L. G. A LSTM based framework for handling multiclass imbalance in DGA botnet detection. 2018.

[42] Saxe J., Berlin K. A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. 2017.

[43] Shaofang Z., Lanfen L., Yuan J., Wang F., Ling Z., Cui J. CNN-based DGA detection with high coverage. 2019.

[44] Yu B., Pan J., Hu J., Nascimento A., Cock M. D. Character level-based detection of DGA domain names, 2018.

[45] Feng Z., Shuo C., Xiaochuan W. Classification for DGA-based malicious domain names with deep learning architectures. Int'l Journal of Intelligent Information Systems, vol. 6, no. 6, pp. 67–71, 2017.

[46] Qiao Y., Zhang B., Zhang W., Sangaiah A. K., Wu H. DGA domain name classification method based on long short-term memory with attention mechanism. Applied Sciences, vol. 9, no. 20, 2019.

[47] Namgung J., Son S. Moon Y. Efficient Deep Learning Models for DGA Domain Detection. Security and Communication Networks. 2021.

[48] Mac H., Tran D., Tong V., Nguyen L. G., Tran H. A. Dga botnet detection using supervised learning methods. In: Proceedings of the Eighth International Symposium on Information and Communication Technology, 211–218. ACM, Nha Trang. 2017.

[49] Hochreiter S., Schmidhuber J. Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735. 1997.

[50] Lea C., Vidal R., Reiter A., Hager G.D. Temporal Convolutional Networks: A Unified Approach to Action Segmentation. In: Hua G., Jégou H. (eds) Computer Vision – ECCV 2016 Workshops. ECCV 2016. Lecture Notes in Computer Science, vol 9915. Springer. 2016.

[51] Shaojie B., Kolter J., Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. 2018.

ДОДАТКИ

Додаток А

Програмна реалізація формування моделі даних та підготовка їх до навчання моделей

```
# підключення до гугл драйв
from google.colab import drive
drive.mount('/content/gdrive')
# шлях до папки з даними на моєму google drive
data_folder = "gdrive/My Drive/GW/data"

# завантаження необхідних ресурсів
!pip install publicsuffixlist
!pip install keras-tcn
# ініціалізація бібліотек і необхідних засобів
import numpy as np
import pandas as pd
import re
from publicsuffixlist import PublicSuffixList
import gc
import math
import collections
import random
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc, roc_auc_score
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.layers.embeddings import Embedding
from keras.layers import LSTM, Conv1D, MaxPooling1D, Input, Flatten
from keras import regularizers

RANDOM_SEED = 1
random.seed(RANDOM_SEED)
%matplotlib inline
```

```
# Загрузка легітимних доменів з .csv файлу та встановлення лейблів
nonDGA_domains = pd.read_csv(data_folder + '/top-
1m.csv', header=None, names=['Домен'])
nonDGA_domains['Вид_DGA'] = 'Відсутній'
nonDGA_domains['Тип'] = 'Легітимний'
nonDGA_domains = nonDGA_domains[['Вид_DGA','Домен','Тип']]
# Зберігаємо набір даних nonDGA_domains.csv
nonDGA_domains.to_csv(data_folder + '/nonDGA_domains.csv', index = False)
nonDGA_domains.describe()

# Загрузка DGA доменів з 360_DGA.txt файлу та встановлення лейблів
dga_domains = pd.read_table(data_folder + '/360_DGA.txt', names = ['Вид_DG
A','Домен','Початковий_час','Кінцевий_чвс'])
dga_domains = dga_domains.iloc[:, 0:2]
dga_domains['Тип']='DGA'
dga_domains = dga_domains.drop_duplicates()
# Зберігаємо набір даних 360_dga_domains.csv
dga_domains.to_csv(data_folder + '/360_dga_domains.csv', index = False)
dga_domains.describe()

# Загрузка DGA доменів з bambenek_dga_domains.txt файлу та встановлення ле
йблів
bambenek_dga_domains = pd.read_csv(data_folder + '/bambenek_dga-
feed.txt', header=None, sep=',', names = [ 'Домен','Вид_DGA','Час','Опис_u
rl'])
bambenek_dga_domain = bambenek_dga_domain[['Вид_DGA', 'Домен']]
bambenek_dga_domains['Тип']='DGA'
dga_familiy_list = list()
bambenek_dga_domain['Вид_DGA'] = bambenek_dga_domain['Вид_DGA'].apply(lamb
da x: x.split(' ')[3])
Зберігаємо набір даних bambenek_dga_domains.csv
bambenek_dga_domains.to_csv(data_folder + '/bambenek_dga_domains.txt', ind
ex = False)
bambenek_dga_domains.describe()

# Тепер отримаємо домени з pandas DataFrame та поєднаємо два джерела
доменів DGA.
nonDGA_domains = nonDGA_domains['Domain'].tolist()
dga_domains = dga_domains['Domain'].tolist() + bambenek_dga_domains['Domai
n'].tolist()
len(nonDGA_domains)

len(dga_domains)
# Загалом у нас є 1 мільйон доменів не DGA та 2 мільйони доменів DGA.

# Після зчитування даних нам потрібно застосувати певну попередню обробку,
щоб зробити наші дані придатними для нейронних мереж. Спочатку ми
об'єднуємо всі домени nonDGA та DGA як один великий список.
X = nonDGA_domains + dga_domains
```

```
# Згідно з цим великим списком, який містить усі наявні у нас дані,
сформуємо словник дійсних символів.
unique_chars = enumerate(set(''.join(X)))
chars_dict = dict()
for i, x in unique_chars: #індекс у enum починається з 0
    print('i: ' + str(i+1) + '  x: ' + x)
    chars_dict[x] = i + 1 #залишаємо 0 для знаку відступу

# Отримавши дійсні символи, ми можемо розрахувати максимальну кількість
ознак для наших даних.

# індекс 0 також буде ознакою для символу заповнення відступу або
невідомого значення
max_features_num = len(chars_dict) + 1
max_features_num

Примітка: довжина chars_dict - це кількість дійсних символів, але це не
справжня кількість ознак. Це пояснюється тим, що дійсні символи - це
ознаки що моють значення, кожен із них означає унікальний символ. Але ми
повинні знати, що "порожнє / нічого" - це також ознака.

# Отримавши chars_dict, ми можемо перетворити символи в послідовності.
Нейронні мережі нічого не можуть зробити з символами. Будь-який
градієнтний спуск або зворотне поширення працює з числами. Тут
перетворення можна розглядати як основу представлення даних для глибинного
навчання в нашому проекті.

# Перетворення символів chars в int
X_in_int = []
for domain in X:
    domain_in_int = []
    for c in domain:
        domain_in_int.append(chars_dict[c])
    X_in_int.append(domain_in_int)

# оновлення значення X
X = X_in_int
# max length - це максимальна довжина домену в нашому наборі даних
maxlen = np.max([len(x) for x in X])
maxlen

# додаємо відступи до значеннь щоб всі вони були максимальної довжини
X = sequence.pad_sequences(X, maxlen=maxlen)
X.shape

# В даний час X є простою впорядкованою комбінацією доменів nonDGA та DGA.
На основі сукупності кожного типу доменів легко генерувати Y.
# Генерація відповідних Y, 0 для 'не DGA'; 1 для 'dga'
Y = np.hstack([np.zeros(len(nonDGA_domains)),np.ones(len(dga_domains))])
```

```
embed_size = maxlen # максимальна довжина домену
max_features = max_features_num # скільки унікальних знаків
використовувати

# розбиття даних на тестову та тренувальну вибірку
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25
, random_state=1)


# Метрики для оцінки оделей
def recall(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1(y_true, y_pred):
    precision = precision(y_true, y_pred)
    recall = recall(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))


# Відображення матриці помилок
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)
```

```
thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j],
             horizontalalignment="center",
             color="white" if cm[i, j] > thresh else "black")

plt.tight_layout()
plt.ylabel('Дійсні значення')
plt.xlabel('Передбачувані значення')
```

Додаток Б

Програмна реалізація методу машинного навчання першої моделі TCN

```
from tcn import TCN, tcn_full_summary
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential

# if time_steps> tcn_layer.receptive_field, тоді ми не зможемо вирішити це
завдання.
batch_size, time_steps, input_dim = None, 73, 1

tcn_layer = TCN(input_shape=(time_steps, input_dim))
# steps.receptive_field  повідомляє, як далеко може бачити модель з точки
зору кроків часу.
print('Receptive field size =', tcn_layer.receptive_field)

model1 = Sequential([ tcn_layer, Dense(1) ])

model1.compile(optimizer='adam', loss='mse', metrics=['accuracy', f1, prec
ision, recall])

tcn_full_summary(model1, expand_residual_blocks=False)

history1 = model1.fit(x_train, y_train, epochs=2, validation_split=0.2)

# зберігаємо модель
model1.save(data_folder + '/TCN_2_Epoch')

# робимо прогнозування на тестовій вибірці
y_pred1 = model1.predict(x_test) > 0.5

# створення матриці помилок
from sklearn.metrics import confusion_matrix
cm1=confusion_matrix(y_test, y_pred1)
print(cm1)

# відображення матриці помилок
font = {'size' : 15}
plt.rc('font', **font)
cnf_matrix = confusion_matrix(y_test, y_pred1)
plt.figure(figsize=(10, 8))
plot_confusion_matrix(cnf_matrix, classes=['DGA', 'He-DGA'],
      title='Матриця помилок')
      plt.savefig(data_folder + "/conf_matrix1.png")
plt.show()
```

```
# відображення точності
acc = (cm1[0][0] + cm1[1][1])/cm1.sum()
acc

# відображення precision
prec = cm1[0][0]/(cm1[0][0] + cm1[0][1])
prec

# відображення recall
rec = cm1[0][0]/(cm1[0][0] + cm1[1][0])
rec

# відображення F1
f1_score(y_test, y_pred1, average=None)


# відображення графіку точності залежно від епохи навчання
plt.plot(history1.history['accuracy'])
plt.show()
```

Додаток В

Програмна реалізація методу машинного навчання другої моделі TCN

```
from keras import Model, Input
from keras.datasets import imdb
from keras.layers import Dense, Dropout, Embedding
from keras.preprocessing import sequence
from tcn import TCN

max_features = max_features #40
# відрізати значення доменів після цієї кількості знаків
maxlen = np.max([len(x) for x in X]) #73
batch_size = 32

print('Loading data...')

print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)

i = Input(shape=(maxlen,))
x = Embedding(max_features, 128)(i)
x = TCN(nb_filters=64,
        kernel_size=6,
        dilations=[1, 2, 4, 8, 16, 32, 64])(x)
x = Dropout(0.5)(x)
x = Dense(1, activation='sigmoid')(x)

model2 = Model(inputs=[i], outputs=[x])

model2.summary()

history2 = model2.compile('adam', 'binary_crossentropy',
      metrics = ['accuracy', f1, precision, recall])

# зберігаємо модель
Model2.save(data_folder + '/TCN_IMDB_10E')


# робимо прогнозування на тестовій вибірці
y_pred2 = model2.predict(x_test) > 0.5

# створення матриці помилок
from sklearn.metrics import confusion_matrix
cm2=confusion_matrix(y_test, y_pred2)
print(cm2)

# відображення матриці помилок
font = {'size' : 15}
```

```
plt.rc('font', **font)
cnf_matrix = confusion_matrix(y_test, y_pred1)
plt.figure(figsize=(10, 8))
plot_confusion_matrix(cnf_matrix, classes=['DGA', 'He-DGA'],
      title='Матриця помилок')
      plt.savefig(data_folder + "/conf_matrix1.png")
plt.show()


# відображення точності
acc = (cm2[0][0] + cm2[1][1])/cm2.sum()
acc

# відображення precision
prec = cm2[0][0]/(cm2[0][0] + cm2[0][1])
prec

# відображення recall
rec = cm2[0][0]/(cm2[0][0] + cm2[1][0])
rec

# відображення F1
f1_score(y_test, y_pred2, average=None)


# відображення графіку точності залежно від епохи навчання
plt.plot(history1.history['accuracy'])
plt.show()
```