Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)
Кафедра Інфокомунікаційної інженерії імені В.В. Поповського
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Аналіз і дослідження методів інтелектуальної маршрутизації в програмно-конфігурованих мережах
(тема)

Виконав:
студент 2 курсу, групи ІКІм-22-1
Майба М.А.
(прізвище, ініціали)

Спеціальність: 172 Телекомунікації та радіотехніка
(код і повна назва спеціальності)
Тип програми: освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма: Інфокомунікаційна інженерія
(повна назва освітньої програми)
Керівник: професор кафедри ІКІ ім. В.В.Поповського
Єременко О.С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ Лемешко О.В.
(підпис) (прізвище, ініціали)

2024 р

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)
Кафедра Інфокомунікаційної інженерії імені В.В. Поповського
(повна назва)
Рівень вищої освіти другий (магістерський)
Спеціальність 172 Телекомунікації та радіотехніка
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Інфокомунікаційна інженерія
(повна назва)

ЗАТВЕРДЖУЮ
Зав. кафедри_____
(підпис)
«____» _____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студент Майба Микола Андрійович
(прізвище, ім'я, по батькові)

1. Тема роботи: Аналіз і дослідження методів інтелектуальної маршрутизації в програмно-конфігурованих мережах.
затверджена наказом по університету від «19» жовтня 2023 р. №1212 Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 20.01.2024 р.
3. Вихідні дані до роботи: методи математичного програмування; база даних мережних пристроїв для проведення класифікації за параметрами безпеки; бібліотеки Python для аналізу даних і машинного навчання; математичні моделі одношляхової маршрутизації з різнотипними метриками; засоби аналітичного моделювання процесів маршрутизації (середовище Python IDLE, GEKKO Optimization Suite, Numpy); вихідні дані для проведення моделювання (структура досліджуваної мережі, пропускна здатність каналів зв'язку).
4. Перелік питань, що потрібно опрацювати в роботі:
1) Провести огляд засобів штучного інтелекту та особливості їх застосування під час управління трафіком в інфокомунікаційних мережах.
2) Розглянути методи інтелектуальної маршрутизації в програмно-конфігурованих мережах.
3) Розв'язати задачу класифікації мережних пристроїв на основі параметрів безпеки за допомогою машинного навчання.

4) Провести аналітичне моделювання одношляхової маршрутизації з різнотипними метриками на основі результатів класифікації пристроїв з машинним навчанням.
5. Перелік графічного матеріалу із зазначенням креслень, плакатів, комп'ютерних ілюстрацій: демонстраційний матеріал у вигляді ppt-презентації (титульний слайд; опис проблеми, об'єкт, предмет і мета дослідження; огляд засобів штучного інтелекту під час управління трафіком і маршрутизації в інфокомунікаційних мережах; класифікація мережних пристроїв на основі параметрів безпеки за допомогою машинного навчання; математичні моделі одношляхової маршрутизації з різнотипними метриками; результати моделювання; висновки).

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		(підпис)	(дата)
Основна частина	професор Єременко Олександра Сергіївна		15.01.2024

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	19.10.2023	Виконано
2	Збір матеріалів для дослідження	30.10.2023	Виконано
3	Розробка 1 розділу	05.11.2023	Виконано
4	Розробка 2 розділу	26.11.2023	Виконано
5	Розробка 3 розділу	03.12.2023	Виконано
6	Розробка 4 розділу	10.12.2023	Виконано
7	Розробка 5 розділу	25.12.2023	Виконано
8	Оформлення атестаційної роботи	15.01.2024	Виконано

Дата видачі завдання 19 жовтня 2023 р

Студент _____ Майба М.А.
(підпис) (прізвище, ініціали)

Керівник роботи _____ професор Єременко О.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 65 с., 28 рис., 5 табл., 42 джерел.

УПРАВЛІННЯ ТРАФІКОМ, МАРШРУТИЗАЦІЯ, SDN, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, ГЛИБОКЕ НАВЧАННЯ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ.

Об’єкт дослідження – процеси інтелектуальної маршрутизації в програмно-конфігурованих мережах.

Предмет дослідження – методи інтелектуальної маршрутизації в програмно-конфігурованих мережах.

Мета роботи – аналіз і дослідження методів інтелектуальної маршрутизації в програмно-конфігурованих мережах.

Методи досліджень – аналіз, формалізація, моделювання, порівняння.

В роботі розглядаються засоби штучного інтелекту, що застосовуються в інфокомунікаційних мережах для підвищення їх ефективності та адаптивності. Основна увага приділяється програмно-конфігурованим мережам, а також застосуванню машинного навчання для інтелектуального управління трафіком та маршрутизації. Особливий інтерес представляють методи глибокого навчання з підсиленням, які дозволяють агентам штучного інтелекту самостійно навчатися оптимальним стратегіям комунікації. Робота містить огляд сучасних досліджень і перспектив розвитку машинного навчання, а також приклади його практичного застосування для інтелектуальної маршрутизації в програмно-конфігурованих мережах. Розв’язано задачу класифікації мережних пристроїв на основі параметрів безпеки за допомогою машинного навчання, для чого розроблено відповідну модель. Проведено аналітичне моделювання одношляхової маршрутизації з різнотипними метриками на основі результатів класифікації з машинним навчанням.

ABSTRACT

The report contains: 65 p., 28 fig., 5 table, 42 sources.

TRAFFIC MANAGEMENT, ROUTING, SDN, ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, DEEP LEARNING, REINFORCEMENT LEARNING.

A research object is a process of intelligent routing in software-defined networks.

The subject of research is methods of intelligent routing in software-defined networks.

The work aims to analyze and research intelligent routing methods in software-defined networks.

Methods of research are analysis, formalization, modeling, and comparison.

The work considers artificial intelligence tools used in infocommunication networks to increase efficiency and adaptability. The main focus is on software-defined networks, as well as the use of machine learning for intelligent traffic management and routing. Of particular interest are reinforcement learning methods that allow artificial intelligence agents to learn optimal communication strategies independently. The work provides an overview of current research and prospects for developing machine learning, as well as examples of its practical application for intelligent routing in software-defined networks. The study solves the problem of classifying network devices based on security parameters using machine learning, for which a corresponding model has been developed. Analytical modeling of single-path routing with different metrics based on machine learning classification results is carried out.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БПЛА – Безпілотний літальний апарат
ІКМ – інфокомунікаційна мережа
МН – машинне навчання
ШІ – штучний інтелект
AI – Artificial intelligence
AlphaGo – go-playing program developed by Google
ANN – Artificial Neural Networks
API – Application Program Interface
CSP – Communications Service Providers
CSV – Comma-Separated Values
CVE – Common Vulnerabilities and Exposures
DBN – Deep Belief Networks
DNN – Deep Neural Network
DRL – Deep Reinforcement Learning
EN – Enterprise Network
ETSI – European Telecommunications Standards Institute
GPU – Graphics Processing Unit
HetNets – Network with complex interoperation between microcell
ICN – Infocommunication Networks
IoT – Internet of Things
IP – Internet Protocol
ISR – Information Security Risk
LAN – Local Area Network
MDP – Markov Decision Process
ML – Machine Learning
MSE – Means Square Errors
NFV – Network Function Virtualization
NIST –National Vulnerability Database
OSPF – Open Shortest Path First
QoE – Quality of Experience
QoS – Quality of Service

RIP – Routing Information Protocol
RL – Reinforcement Learning
SDN – Software Defined Network
SD-WAN – Software Defined Wide Area Network
SL – Supervised Learning
SP – Shortest Path
TPU – Tensor Processing Unit
UL – Unsupervised Learning
WAN – Wide Area Network

ВСТУП

Сучасні інфокомунікаційні мережі (ІКМ) – складні динамічні системи, які забезпечують передачу й обробку інформації між різними користувачами та пристроями. ІКМ постійно розвиваються та модернізуються, щоб задовольнити зростаючі потреби сучасного суспільства у швидкості, якості, надійності та безпеці комунікацій. Одним з напрямків покращення ІКМ є застосування штучного інтелекту (ШІ) для автоматизації та оптимізації різних процесів управління мережею.

Одним з основних інструментів ШІ [1] є машинне навчання (МН) – процес набуття знань або навичок за допомогою даних. МН дозволяє створювати алгоритми, які можуть адаптуватися до змінюваних умов і виконувати завдання, які важко або неможливо програмувати вручну. Одним із сучасних напрямків МН є глибоке навчання [2] (Deep Learning, DL) – підгалузь МН, що використовує багатошарові нейронні мережі для абстрагування високорозмірних даних.

Застосування МН для ІКМ відкриває нові можливості для інтелектуального управління трафіком та маршрутизації в мережах [3]. Інтелектуальне управління трафіком – це процес аналізу, прогнозування, оптимізації та контролю потоків даних в мережах з метою покращення якості обслуговування (Quality of Service, QoS) та якості досвіду (Quality of Experience, QoE) користувачів, а також стійкості та безпеки мережі загалом.

Таким чином, вибір методу машинного навчання для класифікації мережних пристроїв на основі параметрів безпеки, перевірка його достовірності, а також використання отриманих результатів для подальшого рішення завдання безпечної маршрутизації в сучасних інфокомунікаційних мережах є важливою науковою і практичною задачею.

Метою роботи є аналіз і дослідження методів інтелектуальної маршрутизації в програмно-конфігурованих мережах.

Для вирішення поставленої задачі у першому розділі було досліджено використання засобів штучного інтелекту в інфокомунікаційних мережах. З’ясовано, що традиційні мережні архітектури не відповідають вимогам нових сервісів. Було розглянуто основні проривні технології в сфері мережних технологій за останні два десятиліття, а саме програмно-конфігуровані мережі (Software Defined Network, SDN), віртуалізацію мережних функцій (Network Function

Virtualization, NFV) та машинне навчання ML. Також було розглянуто класифікацію алгоритмів машинного навчання.

У другому розділі роботи проаналізовано особливості інтелектуального управління трафіком в інфокомунікаційних мережах. Було акцентовано увагу на використанні глибокого навчання з підкріпленням (Deep Reinforcement Learning, DRL) для вирішення складних проблем оптимізації мережі. Проаналізовано переваги DRL, включаючи можливість отримання рішень для складної оптимізації мережі, навчання і накопичення знань про комунікаційне і мережне середовище, автономне прийняття рішень, покращення швидкості навчання, особливо в задачах з великими просторами станів і дій, та вирішення різних мережних завдань.

У третьому розділі досліджено використання методів інтелектуальної маршрутизації в програмно-конфігурованих мережах. Зроблено акцент на важливості включення штучного інтелекту в системи управління мережним трафіком для забезпечення якості обслуговування в мережах. Розглянуто різні типи методів машинного навчання, включаючи кероване, некероване та підкріплення навчання, а також різні алгоритми. Використане для реалізації інтелектуального прийняття рішень для систем управління мережним трафіком було кероване машинне навчання.

У четвертому розділі зроблено порівняння різних моделей машинного навчання, включаючи лінійну регресію, логістичну регресію, дерева рішень і випадковий ліс, з метою визначення найкращої моделі для поставленої задачі класифікації мережних пристроїв на основі параметрів безпеки. Використано матрицю помилок для оцінки ефективності моделі, що дозволило виявити специфічні помилки класифікації та визначити, які класи модель класифікує найкраще. Зроблено перевірку моделі на валідаційному наборі даних, який не був використаний під час навчання моделі.

У п’ятому розділі розроблено програмний модуль, який використовує прогнозовані дані навченої моделі для розв’язання оптимізаційної задачі щодо безпечної маршрутизації. Використано результати моделі машинного навчання, розробленої в попередньому розділі, для вирішення задач одношляхової маршрутизації. Розглянуто використання різнотипних метрик для визначення оптимального маршруту в мережі.

Окремі результати роботи доповідались на Міжнародних наукових конференціях. Кваліфікаційна робота пов’язана з дослідженнями у межах науково-технічної (експериментальної) розробки 0123U100128 "Розробка алгоритмічно-

програмного забезпечення для кіберстійких інфокомунікаційних систем і мереж критичних інфраструктур", що ведеться на кафедрі інфокомунікаційної інженерії імені В.В. Поповського Харківського національного університету радіоелектроніки.

1 ОГЛЯД ЗАСОБІВ ШТУЧНОГО ІНТЕЛЕКТУ ПІД ЧАС ВИКОРИСТАННЯ В ІНФОКОМУНІКАЦІЙНИХ МЕРЕЖАХ

1.1 Інтелектуалізація програмно-конфігурованих мереж

Традиційні мережні архітектури погано підходять для задоволення вимог нових інтернет-сервісів. Загальноприйнятою практикою в мережах CSP (Communications Service Providers, CSP) є статичний розподіл ресурсів, який відповідає попиту на трафік у години пік. Цей метод призводить до низької мережної та енергетичної ефективності, оскільки поза годинами пік ресурси будуть надлишково надані. Як наслідок, Інтернет став одним з одним з найбільших споживачів енергії у світі. Така ситуація змушує провайдерів послуг зв'язку змінювати базові мережні технології і почали шукати нові технологічні рішення, які підвищують рівень програмованості, керованості та гнучкості конфігурації, знижуючи при цьому витрати на конфігурації, одночасно зменшуючи загальні витрати, пов'язані з експлуатацією мережі.

Основними проривними технологіями у сфері мережних технологій за останні два десятиліття: програмно-конфігуровані мережі (Software-defined networking, SDN), віртуалізація мережних функцій (Network Function Virtualization, NFV) та машинне навчання (Machine Learning, ML).

SDN є одним з найбільш швидкозростаючих ринків інформаційно-комунікаційних технологій: з 289 мільйонів у 2015 році до 8,7 мільярдів у 2020 році, а середньорічний темп зростання становить 98%. Технологія поєднує в собі автоматизацію, гнучкість і розробку додатків для надання мережних послуг у детермінований, динамічний і масштабований спосіб. Розділяючи площину управління і площину даних, SDN повертає мережу до централізованого управління, дозволяючи програмам програмувати правила переадресації мережі через централізовані контролери. SDN є перспективним рішенням для впровадження мережного програмування та прискорення мережних інновацій. Наприклад, мережне програмування дозволяє впроваджувати інтелектуальні методи динамічної оптимізації для підвищення продуктивності та енергоефективності. Застосування SDN в корпоративних мережах (Enterprise network, EN) називається Software Defined Wide Area Network (SD-WAN) [4] і являє собою нову парадигму, яка впроваджує переваги SDN в EN. SD-WAN – це

революційний спосіб спрощення корпоративних мереж і досягнення оптимальної продуктивності додатків за допомогою централізовано керованої віртуалізації WAN. На відміну від традиційних WAN-з'єднань, SD-WAN забезпечує високу гнучкість мережі та економію коштів.

Використовуючи технології віртуалізації, Європейський інститут телекомунікаційних стандартів (ETSI) запропонував NFV для віртуалізації мережних сервісів, які раніше виконувалися спеціальним пропрієтарним обладнанням.

NFV – це парадигма передачі мережних функцій від спеціалізованих апаратних пристроїв до програмних додатків, що працюють на комерційному готовому обладнанні [5]. Мережні пристрої більше не є пропрієтарними, а відкриті для програмного забезпечення різних постачальників, що значно зменшує капітальні та операційні витрати. Хоча впровадження SDN і NFV в мережі приносить революційні переваги в масштабі і гнучкості, воно також приносить абсолютно новий рівень складності. Віртуалізація розбиває традиційну мережу на динамічні компоненти і рівні, які повинні працювати в унісон і можуть змінюватися в будь-який момент часу [6], наприклад, віртуалізований брандмауер може постійно оновлюватися CSP. Висока складність цих нових технологій призвела до того, що дослідження зосередилися на все більш інтелектуальних методах і алгоритмах для оптимізації розподілу ресурсів та маршрутизації у всіх сегментах мережі: доступ, границя та ядро (рис. 1.1).

В останні роки машинне навчання стало основним математичним інструментом, що використовується дослідниками в цій галузі. ML – це підгалузь комп'ютерних наук, яка виникла з вивчення теорії розпізнавання образів та обчислювального навчання в штучному інтелекті. Використовуючи складні математичні та статистичні інструменти, комп'ютери можуть навчатися на основі вхідних даних, як вирішувати конкретні проблеми, які традиційно вирішувала людина [7].

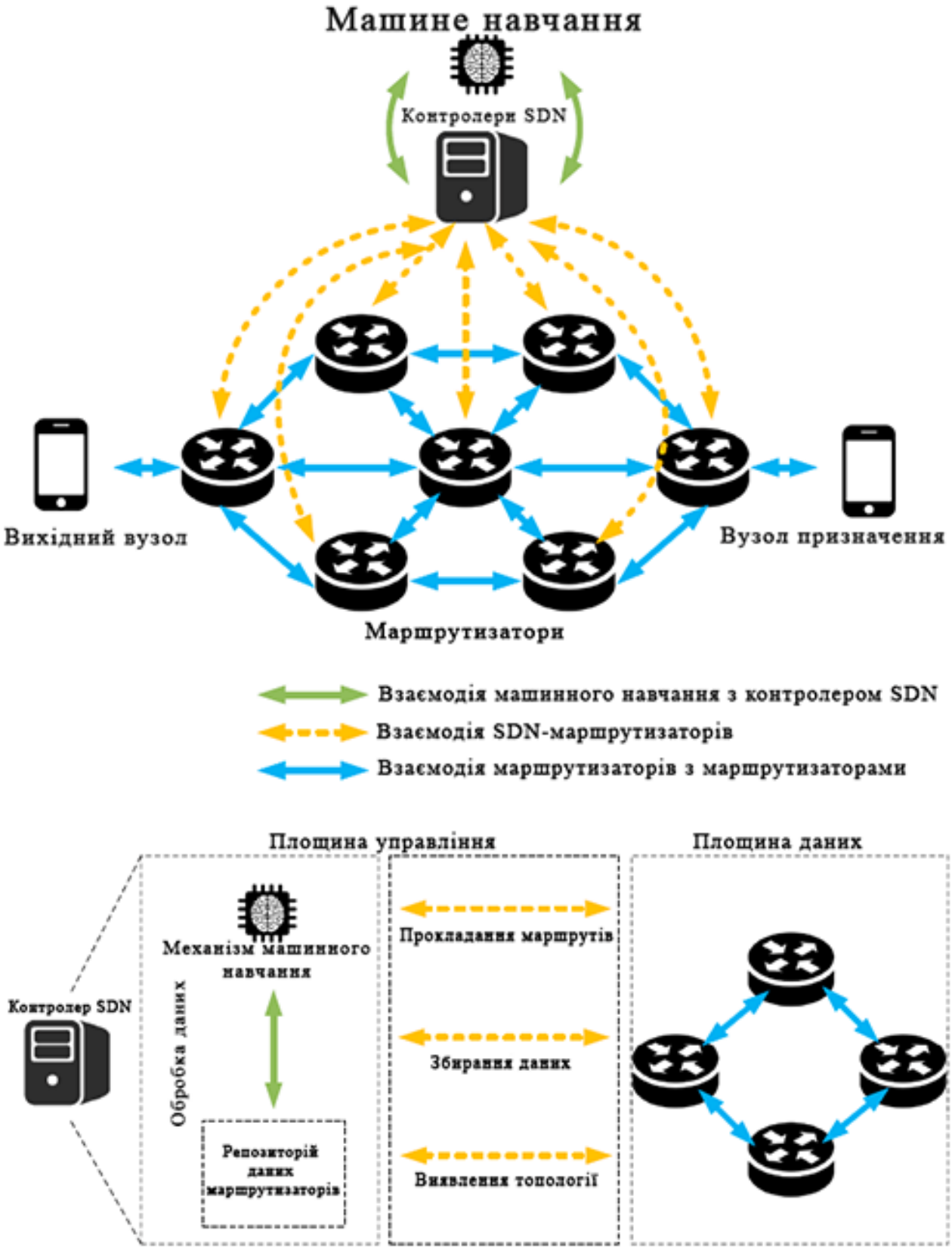

Рисунок 1.1 – Приклад використання ШІ в SDN з використанням засобів машинного навчання [2]

1.2 Класифікація алгоритмів машинного навчання

ML досліджує алгоритми, які можуть «вчитися» на вхідних даних і робити прогнози на їх основі. Етап навчання також називають «тренуванням» алгоритму за допомогою ітеративних механізмів зі зворотним зв'язком для вирішення конкретних завдань. Такі алгоритми працюють шляхом побудови моделі для того, щоб робити прогнози або рішення на основі даних, а не слідувати статичним програмним інструкціям. Алгоритми ML зазвичай класифікуються на три різні категорії:

1. Навчання під контролем (Supervised Learning, SL) використовується в різних додатках, таких як розпізнавання мови розпізнавання мови, виявлення спаму та розпізнавання об'єктів. Його мета полягає в тому, щоб передбачити значення однієї або декількох вихідних змінних за значенням вектора вхідних змінних. Вихідна змінна вихідною змінною може бути неперервна змінна (задача регресії) або дискретна змінною (задача класифікації). Навчальний набір даних складається з набору вибірок вхідних змінних і відповідних вихідних значень.

2. Навчання без вчителя (Unsupervised Learning, UL) полягає в навчанні алгоритму, який використовує лише вхідні векторні дані з метою пошуку прихованої закономірності і подібності. Аналіз соціальних мереж, кластеризація генів та маркетингові дослідження є одними з найуспішніших застосувань методів неконтрольованого навчання. Він може вирішувати різні завдання, такі як кластеризація або кластерний аналіз. Кластеризація - це процес групування даних таким чином, щоб внутрішньо-кластерна схожість між зразками вхідних даних була високою, а міжкластерна - низькою. Міжкластерна схожість є низькою.

3. Навчання з підкріпленням (Reinforcement Learning, RL) - це підхід до ML, який навчає алгоритми за допомогою систему позитивних і негативних заохочень. На відміну від того, що відбувається з контрольованим і неконтрольованим навчанням, в RL ми не маємо тренувального, валідаційного та тестового набору даних, визначених «апріорі». Алгоритм або агент RL навчається тому, як виконувати завдання, відносно навколишньому середовищу. Взаємодія визначається у вигляді у вигляді конкретних дій, спостережень і винагород.

1.3 Висновки до першого розділу

У цьому розділі кваліфікаційної роботи було розглянуто використання засобів штучного інтелекту в інфокомунікаційних мережах. Традиційні мережні архітектури не відповідають вимогам нових інтернет-сервісів.

Розглянуто основні проривні технології в сфері мережних технологій за останні два десятиліття: програмно-конфігуровані мережі SDN, віртуалізація мережних функцій NFV та машинне навчання ML. Класифіковано алгоритми машинного навчання на три різні категорії: SL, UL, RL.

Проаналізовано використання машинного навчання для різних завдань, включаючи розпізнавання мови, виявлення спаму, розпізнавання об’єктів, кластеризацію, а також для вирішення завдань, пов’язаних з взаємодією з навколишнім середовищем.

2 АНАЛІЗ ОСОБЛИВОСТЕЙ ІНТЕЛЕКТУАЛЬНОГО УПРАВЛІННЯ ТРАФІКОМ В ІНФОКОМУНІКАЦІЙНИХ МЕРЕЖАХ

2.1 Визначення особливостей використання навчання з підкріпленням

Під час процесу навчання Reinforcement Learning, агент може періодично приймати рішення, спостерігати за результатами, а потім автоматично коригувати свою стратегію для досягнення оптимальної політики. Однак цей процес навчання, навіть якщо доведено, що він збігається, займає багато часу, щоб досягти найкращої політики, оскільки він повинен дослідити і отримати знання про всю систему, що робить його непридатним і незастосовним для великомасштабних мереж.

Отже, застосування навчання з підкріпленням на практиці дуже обмежене. Нещодавно було представлено DL [8] як нову проривну техніку. Воно може подолати обмеження навчання з підкріпленням і таким чином відкрити нову еру для розвитку навчання з підкріпленням, а саме глибоке навчання з підкріпленням (Deep Reinforcement Learning, DRL).

DRL використовує переваги глибоких нейронних мереж (Deep Neural Network, DNN) для навчання процесу навчання, тим самим покращуючи швидкість навчання та продуктивність алгоритмів навчання з підкріпленням. Як результат, DRL було прийнято в численних додатках навчання з підкріпленням, таких як навчання на практиці, таких як робототехніка, комп'ютерний зір, розпізнавання мови та обробка природної мови [8]. Одним з найвідоміших застосувань DRL є AlphaGo [9], перша комп'ютерна програма, яка може обіграти людину-професіонала без обмежень на дошці 19×19.

2.2 Використання глибокого навчання з підкріпленням для комунікації та мережних технологій

У сфері зв'язку та мережних технологій DRL останнім часом використовується як новий інструмент для ефективного вирішення різноманітних проблем і викликів. Зокрема, сучасні мережі, такі як Інтернет речей (Internet of Things, IoT), гетерогенні мережі (HetNets) і мережі безпілотних літальних апаратів (БПЛА), стають більш децентралізованими, спеціальними і автономними за своєю

природою. Мережні об'єкти, такі як пристрої Інтернету речей, мобільні користувачі і БПЛА, повинні приймати локальні і автономні рішення, наприклад, про доступ до спектру, вибір швидкості передачі даних, управління потужністю передачі і об'єднання базових станцій, щоб досягти цілей різних мереж, включаючи, наприклад, максимізацію пропускної здатності і мінімізацію енергоспоживання. В умовах невизначеності та стохастичності середовища більшість проблем прийняття рішень можуть бути змодельовані за допомогою так званого марковського процесу прийняття рішень (Markov decision process, MDP) [10]. Динамічне програмування [11], [12] та інші алгоритми, такі як ітерація значень, а також методи навчання з підкріпленням, можуть бути застосовані для розв'язання MDP. Однак сучасні мережі є масштабними і складними, а отже, обчислювальна складність методів швидко стає некерованою. Як наслідок, DRL розвивається як альтернативне рішення для подолання цієї проблеми. Загалом, підходи DRL надають наступні переваги:

1. DRL може отримати рішення для складної оптимізації мережі. Таким чином, він дозволяє мережним контролерам, наприклад, базовим станціям, у сучасних мережах вирішувати неопуклі та складні проблеми, наприклад, спільне об'єднання користувачів, обчислення, та графік передачі, для досягнення оптимальних рішень без повної і точної мережної інформації.

2. DRL дозволяє мережним суб'єктам навчатися і накопичувати знання про комунікаційне і мережне середовище. Таким чином, використовуючи DRL, суб'єкти мережі, наприклад, мобільні користувачі, можуть вивчати оптимальні політики, наприклад, вибір базової станції, вибір каналу, рішення про хендовер, рішення про кешування і вивантаження, не знаючи моделі каналу і моделі мобільності.

3. DRL забезпечує автономне прийняття рішень. За допомогою підходів DRL, суб'єкти мережі можуть здійснювати спостереження і отримувати найкращу політику на місцевому рівні з мінімальним обміном інформацією між собою або без нього. або взагалі без обміну інформацією між собою. Це не лише зменшує накладні витрати на зв'язок, але й покращує безпеку і надійність мереж.

4. DRL значно покращує швидкість навчання, особливо в задачах з великими просторами станів і дій. Таким чином, у великомасштабних мережах, наприклад, системах IoT з тисячами пристроїв, DRL дозволяє мережному контролеру або шлюзам IoT динамічно керувати асоціацією користувачів,

доступом до спектру і передачею енергії для величезної кількості пристроїв IoT і мобільних користувачів.

5. Деякі інші проблеми в комунікаціях і мережах, такі як кібер-фізичні атаки, управління завадами і вивантаженням даних, можуть бути змодельовані як ігри, наприклад, некооперативні ігри. Нещодавно DRL почали використовувати як ефективний інструмент для розв'язання ігор, наприклад, для знаходження рівноваги Неша, без повної інформації.

Хоча існують деякі дослідження, пов'язані з машинним навчанням, вони не обговорюють застосування DRL у комунікаціях і мережах. Зокрема, існують дослідження, які обговорюють застосування DRL, такі як [13] і [14], але вони стосуються комп'ютерного зору та обробки природної мови. Зокрема, в огляді [15] обговорюються підходи глибокого навчання для кібербезпеки мереж. Крім того, відсоток робіт, пов'язаних з DRL, для різних мереж та різних завдань у мережах показано на рис. 2.1 [16].

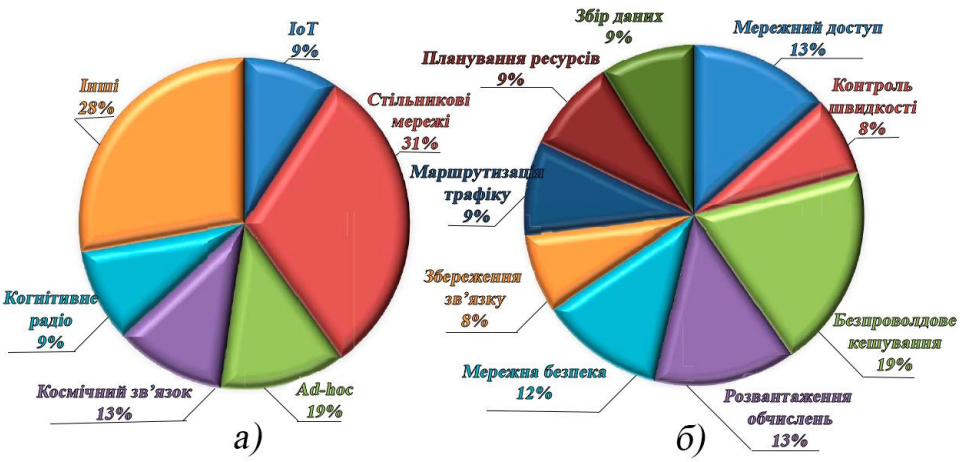

Рисунок 2.1 – Використання DRL

а) – у різних мережах, б) – для виконання різних завдань

З рисунку видно, що більшість робіт, пов'язаних з DRL, стосуються стільникових мереж. Крім того, роботи, пов'язані з безпроводовим кешуванням і розвантаженням, отримали більше уваги, ніж інші.

Таким чином, існуючі дослідження або розглядають застосування DRL для комп'ютерного зору та обробки природної мови, або обговорюють застосування глибокого навчання для роботи в мережі. Не існує жодного дослідження, яке б спеціально обговорювало застосування DRL для комунікацій та створення мереж.

Для зручності роботи класифіковано основні проблеми DRL для комунікації та мережної взаємодії, як показано на рис. 2.2 [16].

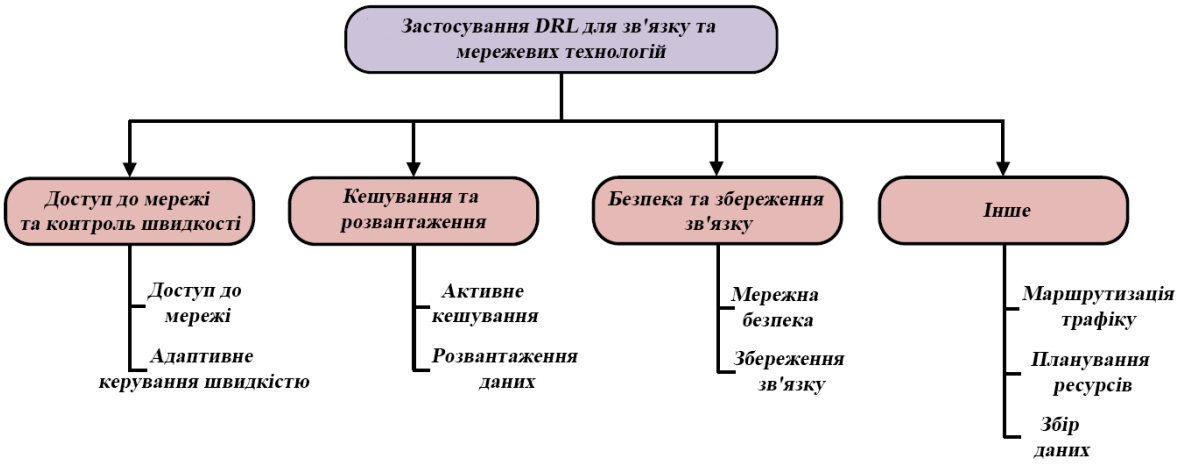

Рисунок 2.2 – Класифікація застосувань DRL для мережної комунікації

Основні проблеми включають доступ до мережі, контроль швидкості передачі даних, безпроводове кешування, вивантаження даних, мережну безпеку, збереження зв'язку, маршрутизацію трафіку та збір даних.

2.3 Висновки до другого розділу

У цьому розділі кваліфікаційної роботи було розглянуто аналіз особливостей інтелектуального управління трафіком в інфокомунікаційних мережах. Акцентовано увагу, на використанні глибокого навчання з підкріпленням (DRL) для вирішення складних проблем оптимізації мережі.

Проаналізовано переваги DRL, включаючи можливість отримання рішень для складної оптимізації мережі, навчання і накопичення знань про комунікаційне і мережне середовище, автономне прийняття рішень, покращення швидкості навчання, особливо в задачах з великими просторами станів і дій, та вирішення різних проблем в комунікаціях і мережах.

Класифіковано основні проблеми DRL для комунікації та мережної взаємодії, включаючи доступ до мережі, контроль швидкості передачі даних, безпроводове кешування, вивантаження даних, мережну безпеку, збереження зв’язку, маршрутизацію трафіку та збір даних.

3 РЕАЛІЗАЦІЯ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОЇ МАРШРУТИЗАЦІЇ В ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖАХ

3.1 Аналіз застосування методів машинного навчання для управління трафіком

Включення ШІ в системи управління мережним трафіком може зіграти важливу роль у забезпеченні якості обслуговування (Quality of Service, QoS) в мережах на основі Інтернет-протоколу (Internet Protocol, IP) [17]. Протягом останніх кількох десятиліть ML використовувався для інтелектуального управління трафіком в дротових/бездротових мережах [18]-[21]. Нещодавно з'явилася ще менша підгрупа методів машинного інтелекту і ML, відома як глибоке навчання, яка має потенціал для створення ще більших порушень [22]-[24].

Для того, щоб зрозуміти, чому системи глибокого навчання повинні замінити своїх попередників (тобто традиційні методи ML), зверніться до різних типів методів ML (SL, RL, UL) і різних алгоритмів (рис. 3.1), які можуть бути використані для реалізації інтелектуального прийняття рішень для систем управління мережним трафіком [32].

Серед методів машинного навчання як керовані, так і некеровані штучні нейронні мережі (Artificial Neural Networks, ANN) використовуються в різних сферах, починаючи від маршрутизації і закінчуючи виявленням вторгнень. Хоча звичайні неглибокі ANN часто використовуються для прогнозування трафіку з метою проактивного управління мережею, їхня продуктивність практично обмежена [25], [26]. Це обмеження пов'язане з тим, що збільшення кількості прихованих шарів ANN не впливає на продуктивність, що стосується покращення рішень щодо управління мережею. Зовсім недавно, однак, відбувся прорив у тому, як системи глибокого навчання, такі як мережі глибокого переконання, глибокі ANNs можуть призвести до значного підвищення продуктивності [27]-[30]. Однак застосування таких систем глибокого навчання обмежувалося переважно розпізнаванням зображень/символів/шаблонів та обробкою природної мови.

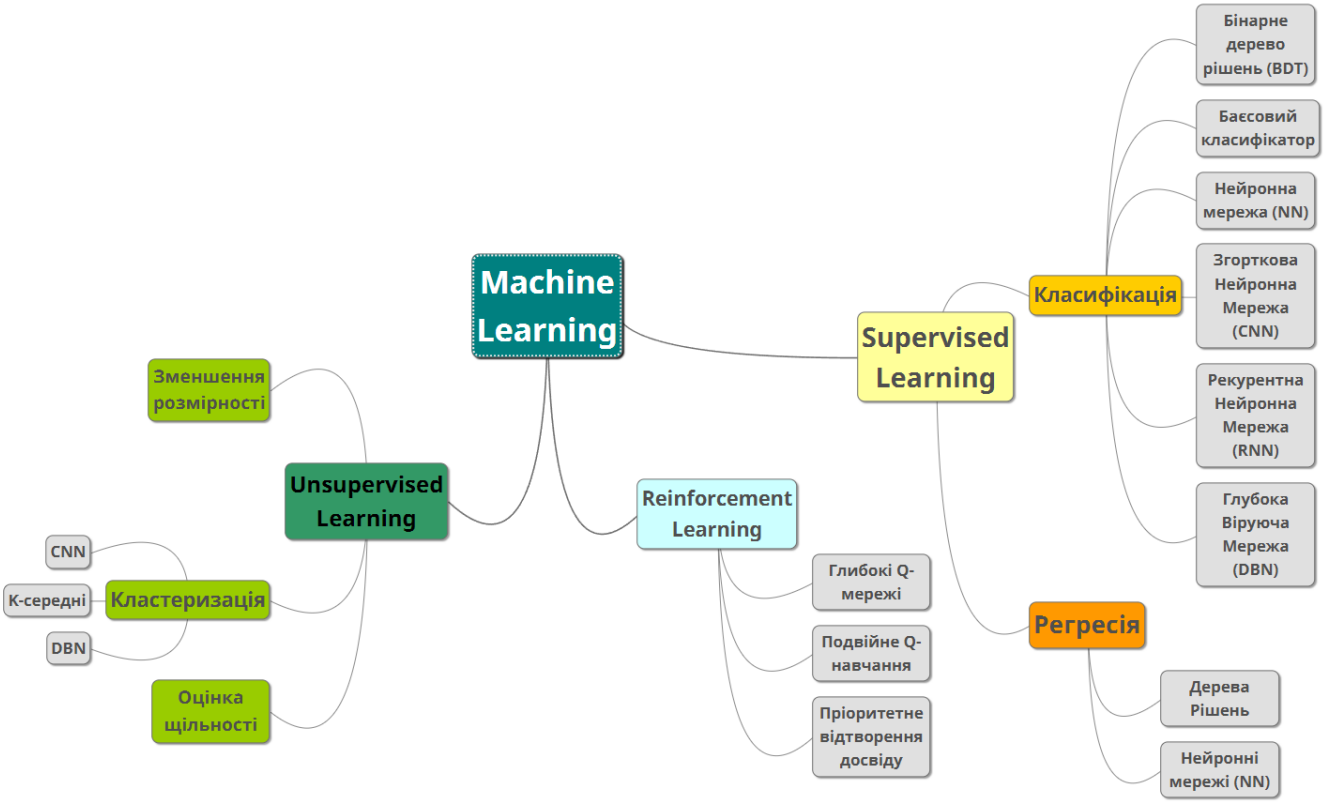

Рисунок 3.1 – Методи машинного навчання, які використовуються для вирішення завдань управління трафіком

3.2 Приклад реалізації інтелектуальної маршрутизації на основі Deep Learning

Як обговорювалося в попередніх розділах, DL можна використовувати в широкому спектрі областей, пов'язаних з мережними технологіями. Крім того, з розвитком нових методів глибокого навчання, в цьому розділі ми представляємо нову сферу інтелектуальних систем управління трафіком, в якій використовується маршрутизація на основі глибокого навчання. Управління маршрутизацією є важливим аспектом управління трафіком, оскільки погано вибрані шляхи можуть призвести до перевантаження мережі, а наступні повторні передачі втрачених пакетів можуть ще більше погіршити перевантаження. У традиційних протоколах маршрутизації основна концепція полягає у виборі шляху, що має максимальне або мінімальне значення, чи метрику, наприклад, алгоритм найкоротшого шляху (Shortest Path, SP) [31]. Щоб вирішити цей недолік традиційних методів маршрутизації, дослідники досліджували можливість застосування машинного навчання для інтелектуального управління маршрутом [17]-[21].

Існує доказ концепції застосування методу глибокого навчання для здійснення інтелектуального управління трафіком в мережах майбутнього[30].

Щоб наочно показати, як архітектура глибокого навчання може бути використана для управління трафіком, ми припустимо бездротову магістральну мережу 4×4, як показано на рис. 3.2 [32].

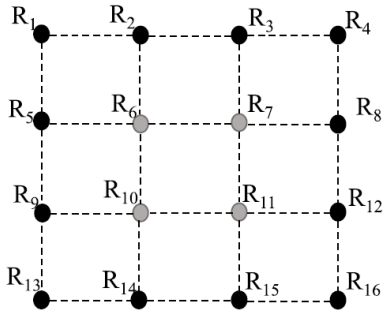

Рисунок 3.2 – Безпроводова магістральна мережа [32]

У розглянутій мережі припустимо, що пакети генеруються тільки в граничних маршрутизаторах і призначені для інших граничних маршрутизаторів, оскільки всі термінали доступу підключені до граничних маршрутизаторів, а внутрішні маршрутизатори виконують лише роль переадресації пакетів. Передбачається, що кожен граничний маршрутизатор запускає кілька мереж глибоких переконань DBN (Deep Belief Networks, DBN) для побудови повних шляхів до інших граничних маршрутизаторів і приєднання своїх пакетів з відповідними шляхами. Внутрішнім маршрутизаторам не потрібно запускати DBN, вони просто зчитують шлях для пересилання пакетів. Для кожного DBN одиниці в нижньому шарі характеризуються як моделі трафіку всіх маршрутизаторів в мережі, в той час як верхній шар представляє наступний вузол для пари «джерело-призначення». Оскільки кількість маршрутизаторів в мережі дорівнює 16, нижній рівень складається з 16 одиниць. Приймається 16-вимірний векторний формат виводу для представлення наступного шляху вузла DBN, так що кожен його елемент має двійкове значення. Крім того, лише один елемент у цьому векторі може дорівнювати 1. Позиція або порядок елемента у векторі, що має значення 1, вказує на наступний вузол. Оскільки кількість граничних маршрутизаторів дорівнює 12, а кожен DBN виводить наступний вузол тільки від одного маршрутизатора до маршрутизатора призначення, то кількість DBN в мережі

дорівнює 180. Щоб зменшити навантаження на навчання, кожен маршрутизатор в мережі навчає DBN, передаючи наступний вузол від себе для кожного з маршрутизаторів призначення.

Навчання DBN можна розділити на два етапи. Перший крок - це попереднє навчання архітектури за допомогою методу багаторівневого навчання, а наступний крок – точне налаштування архітектури за допомогою методу зворотного розповсюдження. За деталями можна звернутися до попередніх робіт [33].

Оскільки навчання архітектури глибокого навчання полягає у виведенні бажаних значень, зазвичай використовується відстань між практичним виходом і бажаним виходом для вимірювання помилок навчання.

Помилка в цій ситуації виникає через додавання стохастичного шуму, який переростає велику вартість в мережі, що може призвести до неконтрольованих або неправильних результатів. Цей ефект шуму може бути важливим в деяких додатках, але в інших він може бути небажаним і потребує контролю та регулювання. Отже, щоб мінімізувати значення функції витрат, потрібно скоригувати значення вагових коефіцієнтів. Для коригування ваги потрібно використати метод зворотного розповсюдження, поки значення функції витрат не досягне необхідного рівня. Використовуючи таблицю 3.1, можна знайти значення ваг та середньоквадратичних помилок (Means Square Errors, MSE), які використовуються для вимірювання значення функції витрат.

Таблиця 3.1 – Значення ваг та середньоквадратичної похибки (MSE) для різної кількості кроків зворотного розповсюдження [32]

Крок	Вага	Значення	MSE
100	w^1_{00}	0.825148	2.918418×10^{-5}
	w^1_{10}	-0.600210	
	…	…	
	$w^3_{11,15}$	-0.074882	
	$w^3_{12,15}$	-0.052039	
200	w^1_{00}	0.825954	6.719093×10^{-6}
	w^1_{10}	-0.598363	
	…	…	
	$w^3_{11,15}$	-0.143467	
	$w^3_{12,15}$	-0.121055	

З таблиці 3.1 видно, що значення MSE зменшується зі збільшенням кількості кроків регресії.

Після фази навчання кожен маршрутизатор отримує значення ваг і зсувів DBN, які передбачають наступні вузли для граничних маршрутизаторів від нього самого. Наступним кроком кожен маршрутизатор пересилає ці значення граничним маршрутизаторам. Таким чином, після того, як кожен граничний маршрутизатор отримає схеми трафіку всіх маршрутизаторів, він може використовувати DBN для побудови повних шляхів до інших граничних маршрутизаторів.

Далі, на основі симуляції топології в [33], оцінюється продуктивність маршрутизації на основі глибокого навчання. Оскільки обчислення всіх маршрутизаторів були аутсорсифіковані на одному комп'ютері, оцінка була обмежена середньомасштабною бездротовою комірчастою магістральною мережею, що складається з 16 маршрутизаторів, як показано на рис. 3.2, а не повномасштабною топологією магістральної мережі. Розмір пакетів даних та керуючих пакетів встановлено на 1 Кб. Пропускну здатність каналу встановлено на рівні 8 Мбіт/с, що є прийнятним для такого масштабу бездротової магістралі [34]. Передбачається, що кожен вузол має необмежений буфер. Загальна швидкість генерації пакетів даних у розглянутій мережі змінюється від 7,68 Мбіт/с до 14,4 Мбіт/с. Для порівняння прийнятої системи глибокого навчання в якості еталонного методу використано OSPF.

Результати дослідження [32] показують, що накладні витрати на передачу сигналів у OSPF набагато вищі, ніж у системи глибокого навчання, незалежно від того, як змінюється частота генерації. Нижчий рівень сигналізації, досягнутий пропозицією дослідження [32], можна пояснити наступним чином. У той час як звичайний метод OSPF вимагає від усіх маршрутизаторів частого обміну інформацією про маршрути з відповідними сусідами, в методі глибокого навчання достатньо лише шаблонів трафіку граничних маршрутизаторів, щоб обчислити всі шляхи з точністю до 95%. Тому в методі глибокого навчання тільки граничні маршрутизатори повинні обмінюватися шаблонами трафіку між собою, а шаблони трафіку внутрішніх маршрутизаторів довільно встановлюються на етапі запуску.

Пропускна здатність, досягнута системою глибокого навчання, майже близька до середньої швидкості генерації даних маршрутизаторами, оскільки вона уникає заторів і втрати пакетів, оцінюючи маршрути до місця призначення набагато швидше, ніж звичайний OSPF.

3.3 Висновки до третього розділу

У цьому розділі кваліфікаційної роботи було розглянуто використання методів інтелектуальної маршрутизації в програмно-конфігурованих мережах. Було зроблено акцент на важливість включення штучного інтелекту в системи управління трафіком для забезпечення якості обслуговування в мережі.

Було розглянуто різні типи методів машинного навчання, включаючи кероване, некероване та навчання з підкріпленням, а також різні алгоритми, які можуть бути використані для реалізації інтелектуального прийняття рішень для систем управління мережним трафіком.

Розглянуто приклад використання глибокого навчання для інтелектуального управління маршрутизацією, а саме приклад безпроводової магістральної мережі, в якій використовується маршрутизація на основі глибокого навчання. Показано, як глибоке навчання може бути використане для побудови повних шляхів до інших граничних маршрутизаторів і приєднання своїх пакетів з відповідними шляхами.

Було оцінено продуктивність маршрутизації на основі глибокого навчання, порівнюючи її з традиційними методами маршрутизації, такими як OSPF. Це показує, що система глибокого навчання може досягти вищої пропускної здатності і нижчого рівня сигналізації, ніж традиційні методи.

4 РОЗВ’ЯЗАННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ ПРИСТРОЇВ НА ОСНОВІ ПАРАМЕТРІВ БЕЗПЕКИ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ

4.1 Особливості засобів машинного навчання для задач класифікації та регресії в мережах

Основна мета машинного навчання полягає в передбаченні результатів на основі вхідних даних. Чим більше різноманітності даних, тим простіше для алгоритму виявити закономірності, що призводить до більш точних результатів. Для того, щоб навчити машину потрібні три основні компоненти.

Перший компонент – це дані. Наприклад, якщо потрібно оптимізувати процес маршрутизації в мережі, то потрібні дані про трафік. Якщо хочемо передбачати затримки передачі даних, тоді потрібна історія затримок. Якщо хочемо з’ясувати патерни використання мережі користувачами, нам потрібні дані про їхню активність. Для ефективного навчання потрібно максимально велика кількість даних.

Другий компонент – це ознаки. Це можуть бути різні характеристики, такі як пропускна здатність мережі, час відгуку сервера, кількість одночасних користувачів або навіть частота використання певних служб. Машина повинна знати, на що їй конкретно дивитися під час аналізу інфокомунікаційних систем.

Третій компонент – це алгоритм. Зазвичай одну й ту ж задачу можна розв'язати різними методами. Від вибору методу залежить точність, швидкість роботи і розмір готової моделі. Однак, якщо дані не якісні, то навіть найкращий алгоритм не допоможе.

Як обговорювалося в попередніх розділах, ML можна класифікувати на три категорії, навчання з вчителем, без та з підкріпленням. У цьому розділі більш детально буде розглянуто категорію навчання з вчителем.

У випадку SL, машина має вчителя, тобто правильну відповідь. Наприклад, якщо існує великий набір даних мережних пристроїв, у ролі вчителя може виступати цільова змінна, яка заздалегідь розділила всі дані на безпечні та небезпечні пристрої, або за будь-яким іншим параметром. Таким чином, машина може вивчити на основі навчальних даних, які пристрої є безпечними, а які – ні.

Використання навчального алгоритму дозволяє машині навчитися швидше та точніше, тому в практичних задачах його використовують значно частіше. Ці задачі

можна поділити на два основних типи – класифікація та регресія. Класифікація передбачає категорію об’єкта, а регресія передбачає значення на числовій осі. Наприклад, класифікація може бути використана для визначення рівня безпечності пристрою, а регресія – для прогнозування імовірності компрометації каналу зв’язку.

4.2 Розробка моделі машинного навчання для класифікації мережних пристроїв за показниками безпеки

Основною метою завдання є розробка моделі машинного навчання, яка зможе прогнозувати, чи є пристрій безпечним на основі його певних характеристик. Як було описано у розділі вище, для виконання цього завдання необхідний набір даних.

За допомогою API (Application Program Interface) NIST (National Vulnerability Database) [35] був зібраний набір даних. Цей набір включає інформацію щодо мережних пристроїв, а саме: назва пристрою (Device), виробник (Vendor), тип пристрою (Type), інформаційний ризик безпеки (Information Security Risk, ISR), кількість вразливостей, що має пристрій (Amount of vulnerabilities), назву вразливості (Common Vulnerabilities and Exposures, CVE), відображення серйозності вразливості у вигляді рівня (Level) та у вигляді числового балу (BaseScore), імовірність того, що вразливість буде використана чи реалізована (Exploitability).

Слід зазначити, що CVE – це система ідентифікації вразливостей в програмному забезпеченні. Кожна вразливість або експлойт отримує унікальний ідентифікатор CVE, що дозволяє вченим і безпековим спеціалістам легко знаходити інформацію про конкретну вразливість.

Ці ідентифікатори використовуються для обміну даними між різними продуктами безпеки та службами, що дозволяє автоматизувати процеси управління вразливостями. Крім того, система CVE допомагає у координації зусиль щодо виявлення та виправлення вразливостей, що забезпечує більшу безпеку цифрових систем на всіх рівнях.

Розмірність набору даних включає 24820 рядків та 9 стовпців. Перші 5 рядків набору даних можна побачити на рисунку 4.1. Вся робота з набором даних буде виконана у Google Colab – сервісі від Google, який надає можливість запускати блокноти Jupyter в хмарі. Він дозволяє користувачам писати та виконувати код

Python без необхідності встановлення його на своєму комп'ютері. Основною причиною вибору цього сервісу стало те, що Google надає безкоштовний доступ до обчислювальних ресурсів, включаючи графічні процесори (GPU) та тензорні процесори (TPU), а це є особливо корисним для завдань машинного навчання. Також сервіс сумісний з більшістю популярних бібліотек Python, що використовуються для аналізу даних та машинного навчання, таких як Pandas, NumPy, matplotlib scikit-learn тощо. Повний список бібліотек, які використовувались для МН у межах дослідження, можна побачити на рисунку 4.2.

Перші 5 рядків:

	Device	Vendor	Type	ISR	Mount of vulnerabilities	CVE	BaseScore	Level	Exploitability
1	Cisco 1000V	Cisco	switch	2.07	1	CVE-2020-3508	7.4	HIGH	0.28
2	Cisco 1100	Cisco	router	6.64	5	CVE-2019-12647	7.5	HIGH	0.39
3	Cisco 1100	Cisco	router	6.64	5	CVE-2020-24587	2.6	LOW	0.12
4	Cisco 1100	Cisco	router	6.64	5	CVE-2020-24588	3.5	LOW	0.21
5	Cisco 1100	Cisco	router	6.64	5	CVE-2020-26139	5.3	MEDIUM	0.16

Рисунок 4.1 – Перші 5 рядків набору даних

Набір даних спочатку був отриманий у форматі CSV (Comma-Separated Values, CSV). Формат CSV – це простий формат файлу, який використовується для зберігання табличних даних, наприклад, бази даних або електронних таблиць. Він використовує коми для розділення значень в кожному рядку і може бути легко читаний і оброблений більшістю програм. Однак, для зручності роботи, набір даних був перетворений у формат звичайної таблиці Excel, бо він дозволяє легко переглядати та редагувати дані у візуально зручному форматі.

З рисунку 4.1 видно, що дані зберігаються у не дуже зручному форматі, бо кожен рядок відображає окрему вразливість CVE, але для одного пристрою може бути декілька вразливостей. Це означає, що інформація про один пристрій розкидана по декількох рядках. Основна проблема у тому, що модель машинного навчання буде обробляти ці рядки, як інформацію про різні пристрої, а не як різні вразливості одного й того ж пристрою. Тому для ефективного аналізу даних та побудови моделі машинного навчання було прийнято рішення залишити для кожного пристрою лише одну вразливість з найвищим значенням BaseScore. Це дозволяє зосередитись на найбільш критичних вразливостях для кожного пристрою та спрощує подальшу обробку даних. У майбутніх роботах буде реалізовано варіант з урахуванням всіх вразливостей для мережного пристрою.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive
import joblib
from google.colab import files
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

Рисунок 4.2 – Бібліотеки, використані під час розробки та дослідження

Наступним кроком є підготовка даних для подальшого навчання. Перш за все, видаляються пробіли на початку та в кінці всіх строкових значень. Це важливо, оскільки пробіли можуть вплинути на результати аналізу даних. Наприклад, якщо спробувати порівняти два однакові значення, але одне з них має пробіл на початку або в кінці, вони будуть вважатися різними значеннями. Це може призвести до помилок в аналізі.

Стовпець «Vendor» видаляється з набору даних, оскільки всі значення в ньому однакові. Він не вносить варіативності в дані, тому не впливає на модель машинного навчання. Стовпець «Mount of vulnerabilities» також видаляється, оскільки він може заважати навчанню, бо було прийнято рішення розглядати пристрій лише за однією, найкритичнішою вразливістю. Це дозволяє зосередитись на найбільш критичних вразливостях для кожного пристрою та спрощує подальшу обробку даних.

Ці кроки є важливою частиною підготовки даних для аналізу та моделювання машинного навчання. Вони допомагають забезпечити те, що дані «чисті», організовані та готові до подальшого аналізу. Програмний код і результат підготовки даних, представлено на рис. 4.3 – 4.4.

```python
# Видалення пробілів на початку та наприкінці всіх строкових значень
df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)
# Видалення пробілів на початку та наприкінці назв стовпців
df.columns = df.columns.str.strip()
# Видалення стовпчика Vendor та (A)Mount of vulnerabilities
df = df.drop('Vendor', axis=1)
df = df.drop('Mount of vulnerabilities', axis = 1)


print(df.head())
print('\n Розмірність набору даних:',df.shape[0])
# Збереження тільки одного рядка Device, у якого найбільший BaseScore
df=df.loc[df.groupby('Device')
['BaseScore'].transform('idxmax')]

df = df.drop_duplicates()
print(df.head())
print(f"\n\nРозмірність набору даних після змін: {df.shape}")
total_devices = len(df)
print(f"Усього пристроїв: {total_devices}")
print('Мінімальне значення ISR:', df['ISR'].min())
print('Максимальне значення ISR:', df['ISR'].max())
```

Рисунок 4.3 – Програмний код для підготовки, очищення набору даних і збереження лише найкритичнішої вразливості для кожного пристрою

```
        Device    Type   ISR            CVE  BaseScore   Level  Exploitability
1  Cisco 1000V  switch  2.07   CVE-2020-3508        7.4    HIGH            0.28
2   Cisco 1100  router  6.64  CVE-2019-12647        7.5    HIGH            0.39
3   Cisco 1100  router  6.64  CVE-2020-24587        2.6     LOW            0.12
4   Cisco 1100  router  6.64  CVE-2020-24588        3.5     LOW            0.21
5   Cisco 1100  router  6.64  CVE-2020-26139        5.3  MEDIUM            0.16

 Розмірність набору даних: 24820
                                Device     Type    ISR            CVE  BaseScore   Level  Exploitability
1                           Cisco 1000V   switch   2.07   CVE-2020-3508        7.4    HIGH            0.28
2                            Cisco 1100   router   6.64  CVE-2019-12647        7.5    HIGH            0.39
10  Cisco 1100 Integrated Services Router   router  10.47   CVE-2020-3474        8.1    HIGH            0.28
12  Cisco 1100 Terminal Services Gateways  unknown   1.82   CVE-2020-3465        6.5  MEDIUM            0.28
13                         Cisco 1100-4p   router  10.72  CVE-2019-12646        7.5    HIGH            0.39


Розмірність набору даних після змін: (2918, 7)
Усього пристроїв: 2918
Мінімальне значення ISR: 0.32
Максимальне значення ISR: 182.54
```

Рисунок 4.4 – Результат підготовки даних

Після підготовки можна перейти до наступного етапу – аналізу даних. Як видно з рисунку 4.4 розмірність набору даних змінилася з 24820 до 2918 рядків, тепер кожен рядок має інформацію про один пристрій. Також видно, що ISR має дуже великий діапазон від 0.32 до 182, а це може ускладнити аналіз та моделювання

даних. Через це, було прийнято рішення розділити цей великий діапазон на менші класи, для того, щоб виявити, яка кількість пристроїв належить до певного класу.

Для реалізації поділу даних на діапазони, був реалізований код, показаний на рисунку 4.5.

```
conditions = [
    (df['ISR'] <= 10), (df['ISR'] > 10) & (df['ISR'] <= 20),
    (df['ISR'] > 20) & (df['ISR'] <= 30),
    (df['ISR'] > 30) & (df['ISR'] <= 40),
    (df['ISR'] > 40) & (df['ISR'] <= 50), (df['ISR'] > 50)
]
colors = ['green', 'yellow', 'orange', 'red', 'brown', 'purple']
ranges = ['<=10', '<=20', '<=30', '<=40', '<=50', '50+']
df['Color'] = np.select(conditions, colors)
# підраховуємо кількість пристроїв у кожному діапазони
device_counts = df['Color'].value_counts()
# функція для відображення кількості пристроїв
def absolute_value(val):
    a  = np.round(val/100.*device_counts.sum(), 0)
    return int(a)
fig, axs = plt.subplots(1, 2, figsize=(20, 10))
# будуємо круговую діаграму з відсотками на сегментах

axs[0].pie(device_counts.values, colors=colors, autopct='%1.1f%%')
axs[0].set_title('Розподіл пристроїв за діапазонами ISR (%)')

# будуємо круговую діаграму з числовими значеннями на сегментах
axs[1].pie(device_counts.values, colors=colors, autopct=absolute_value)
axs[1].set_title('Розподіл пристроїв за діапазонами ISR (числове значення)')
# створюємо легенду з діапазонами ISR
legend_labels = [f'{range}' for color, range in zip(colors, ranges)]
fig.legend(legend_labels, title="Діапазони", loc="center right")
plt.figtext(0.5, 0.01, f"Всього пристроїв: {df.shape[0]}", ha="center", fontsize=12)
plt.show()
print('\n\n', df.head())
df = df.drop('Color', axis=1)
```

Рисунок 4.5 – Приклад коду для поділу даних за діапазонами

Код, який представлено на рисунку 4.5, спочатку сегментує дані ISR в різні діапазони. Кожен діапазон представлений окремим кольором. Отже, код обчислює кількість пристроїв, що належать до кожного діапазону, та будує діаграму (рис. 4.6.), відсоткове значення для кожного діапазону представлено у таблиці 4.1.

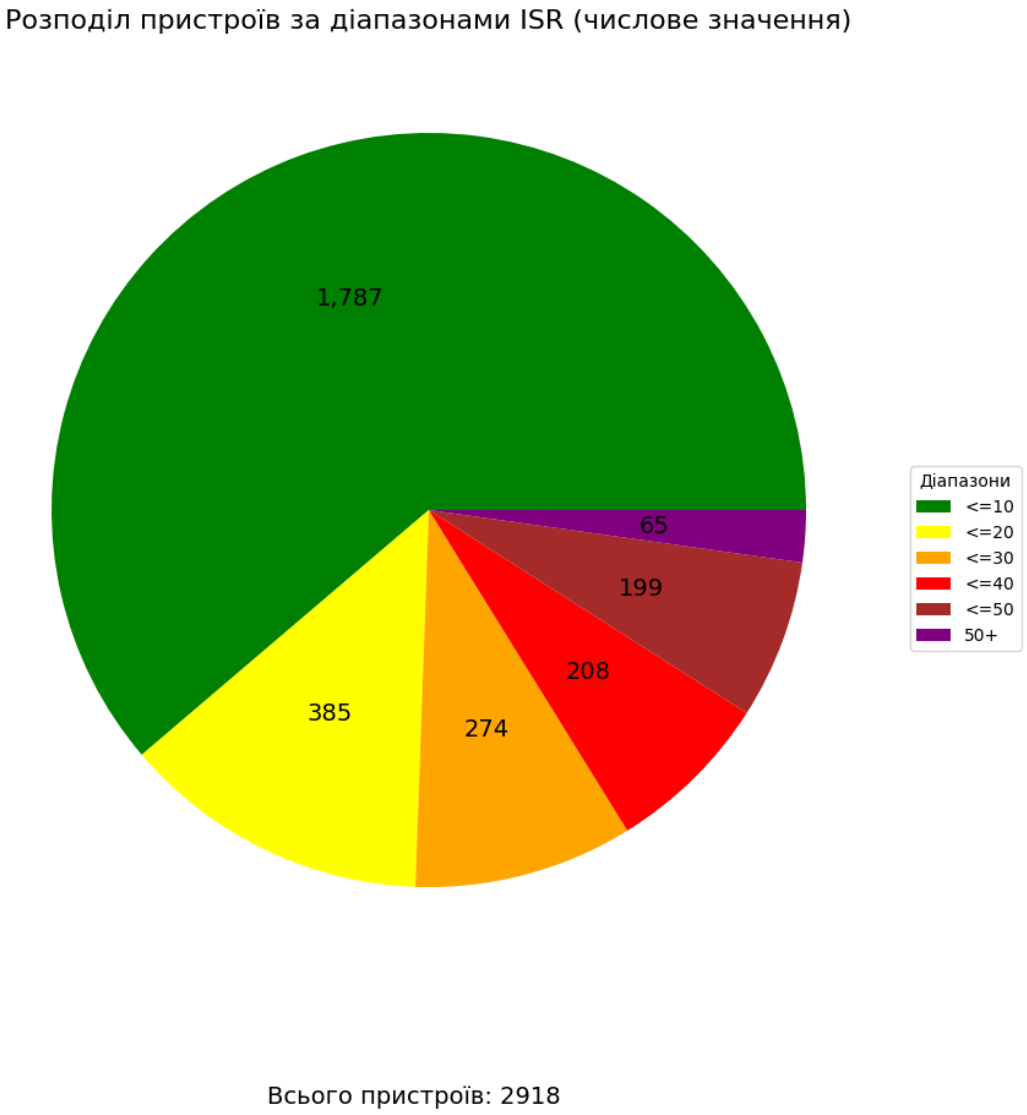

Рисунок 4.6 – Кругова діаграма, що відображає розподіл пристроїв за різними діапазонами ISR

Таблиця 4.1 – Відповідність діапазонів та кількості пристроїв

Діапазон	Кількість пристроїв	Відсоток від загальної кількості пристроїв, %
ISR $\leqslant$ 10	1787	61,3
ISR $\leqslant$ 20	385	13,2
ISR $\leqslant$ 30	274	9,4
ISR $\leqslant$ 40	208	7,1
ISR $\leqslant$ 50	199	6,8
ISR $>$ 50	65	2,2

З рисунку 4.6 видно, що більшість пристроїв має ISR менше 10, що відображено зеленим кольором. Це становить більшість пристроїв у наборі даних.

Моделі машинного навчання працюють краще, коли є достатньо даних для навчання, тому надалі робота буде тільки з діапазоном даних, де ISR менше або дорівнює 10.

Дані, які належать першому діапазону, зберігаються у новому датафреймі для зручності роботи. Це дозволить зосередитись на цьому конкретному діапазоні без утрати загальної структури даних. Результат та перевірка виконання розподілу ISR до нового набору даних представлена на рис. 4.7. Після цього, цей діапазон знову поділяється на 5 класів. Це робиться для того, щоб мати більш детальний розподіл ISR у межах цього діапазону. Кожен з цих класів відображає певний рівень безпеки пристрою.

```
[ ]  newDf = df[(df['ISR'] >= 0) & (df['ISR'] <= 10)].copy()
     print(newDf.head())
     print('\n\nМінімальне значення ISR:', newDf['ISR'].min())
     print('Максимальне значення ISR:', newDf['ISR'].max())
     print('Всього пристроїв:', newDf.shape[0])
```

```
                                  Device     Type   ISR            CVE  BaseScore   Level  Exploitability
1                             Cisco 1000V   switch  2.07  CVE-2020-3508        7.4    HIGH            0.28
2                              Cisco 1100   router  6.64  CVE-2019-12647       7.5    HIGH            0.39
12  Cisco 1100 Terminal Services Gateways  unknown  1.82  CVE-2020-3465        6.5  MEDIUM            0.28
49                         Cisco 1111-4PWE   router  6.71  CVE-2020-3559        8.6    HIGH            0.39
51                    Cisco 1111-8PLTEEEAWB   router  6.71  CVE-2020-3559        8.6    HIGH            0.39


Мінімальне значення ISR: 0.32
Максимальне значення ISR: 9.92
Всього пристроїв: 1787
```

Рисунок 4.7 – Приклад розділення даних за ISR та збереження у новий набір даних

Після розділення даних на діапазони необхідно знову проаналізувати кількість пристроїв у кожному. Для цього використовувалась та ж сама функція, що й для попереднього набору даних, тільки на вхід було надано вже нові значення діапазонів та дані. Результат розподілу пристроїв у межах нового діапазону можна побачити на рис. 4.8.

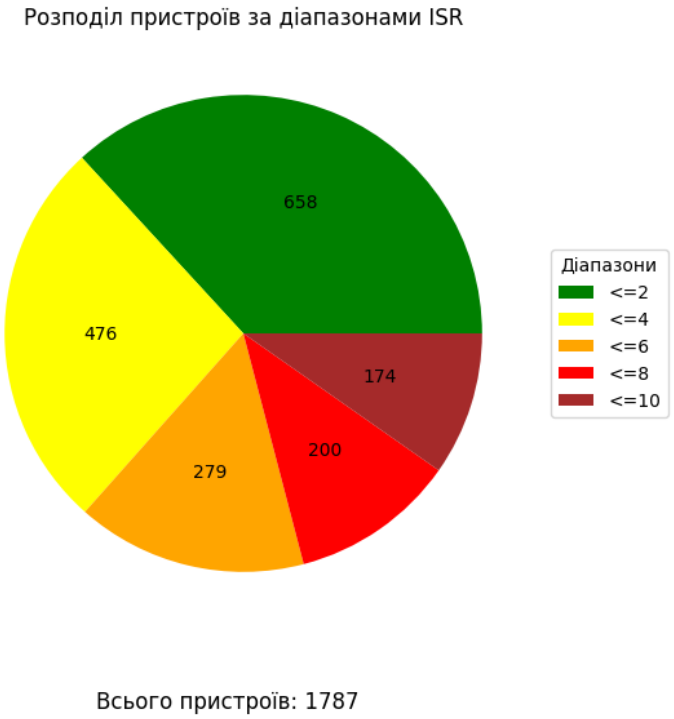

Рисунок 4.8 – Кругова діаграма, що відображає розподіл пристроїв за різними діапазонами ISR у межах першого діапазону (0-10)

Після аналізу даних і визначення розподілу пристроїв за різними діапазонами ISR наступним кроком є підготовка до навчання моделі.

Для цього до набору даних було додано додатковий стовпець «Color». Цей стовпець відображає діапазон ISR для кожного пристрою, який був визначений на попередньому етапі. Кожен діапазон ISR представлений окремим кольором.

Стовпець «Color» буде використовуватися як цільова змінна при навчанні моделі. Це означає, що модель буде навчатися прогнозувати значення цього стовпця на основі інших характеристик пристрою. Результат додавання додаткового стовпця наведено на рис. 4.9. За допомогою цього стовпця можна визначити рівень безпеки пристрою, що робить це завданням класифікації.

	Device	Type	ISR	CVE	BaseScore	Level	Exploitability	Color
1	Cisco 1000V	switch	2.07	CVE-2020-3508	7.4	HIGH	0.28	yellow
2	Cisco 1100	router	6.64	CVE-2019-12647	7.5	HIGH	0.39	red
12	Cisco 1100 Terminal Services Gateways	unknown	1.82	CVE-2020-3465	6.5	MEDIUM	0.28	green
49	Cisco 1111-4PWE	router	6.71	CVE-2020-3559	8.6	HIGH	0.39	red
51	Cisco 1111-8PLTEEEAWB	router	6.71	CVE-2020-3559	8.6	HIGH	0.39	red

Рисунок 4.9 – Перевірка додавання додаткового стовпця «Color» до набору даних для використання як цільової змінної при навчанні моделі.

Це завдання є прикладом багатокласової класифікації, оскільки є більше ніж два можливих класи. Це робить завдання більш складним ніж бінарна класифікація, але також дозволяє отримати більш детальну інформацію про рівень безпеки пристрою.

4.3 Навчання моделі для класифікації пристроїв на основі рівня безпеки

У Python існує велика кількість бібліотек для машинного навчання. У цьому розділі основна увага буде приділена одній з найбільш популярних – Scikit-Learn.

Scikit-Learn значно спрощує процес створення класифікатора, допомагаючи чітко виділити концепції машинного навчання. Вона реалізує ці концепції за допомогою зрозумілої, добре задокументованої та надійної бібліотеки [36]. Це робить Scikit-Learn відмінним інструментом для розробки та впровадження моделей машинного навчання.

У системах машинного навчання або нейронних мережах існують входи та виходи. Те, що подається на входи, зазвичай називають ознаками (features).

Ознаки по суті є тим самим, що й змінні в науковому експерименті – вони характеризують певний спостережуваний феномен, і їх можна кількісно виміряти. Коли ознаки подаються на входи системи машинного навчання, ця система намагається знайти збіги, помітити закономірність між ознаками. На виході генерується результат цієї роботи.

Цей результат зазвичай називають міткою (label), оскільки у виходів є певна позначка, видана їм системою, тобто припущення (прогноз) про те, до якої категорії потрапляє вихід після класифікації.

У контексті машинного навчання класифікація відноситься до навчання з учителем. Такий тип навчання передбачає, що дані, які подаються на входи системи, вже позначені, а важлива частина ознак вже розділена на окремі категорії або класи. Тому мережа вже знає, яка частина входів важлива, а яку частину можна самостійно перевірити. Таке завдання може бути виконане за допомогою дерева рішень – одного з типів класифікатора в Scikit-Learn.

При навчанні без учителя в систему подаються непозначені дані, і вона повинна спробувати сама розділити ці дані на категорії. Оскільки розв’язується задача класифікації, спосіб навчання без учителя розглядатися не буде.

Процес навчання моделі – це подача даних для нейромережі, яка в результаті повинна вивести певні шаблони для даних. У процесі навчання моделі з учителем

на вхід подаються ознаки та мітки, а при прогнозуванні на вхід класифікатора подаються лише ознаки.

Дані, які приймає мережа, діляться на дві групи: набір даних для навчання та набір для тестування. Не варто перевіряти мережу на тому ж наборі даних, на яких вона навчалася, оскільки модель вже буде «налаштована» під цей набір.

Алгоритми машинного навчання можуть бути описані як навчання цільової функції f, яка найкращим чином відповідає вхідним змінним X та вихідній змінній Y: $Y = f(X)$.

Невідомо, що представляє собою функція f. Адже якби ця функція була відома, то використовувалася б безпосередньо, а шукалась б шляхом навчання за допомогою різних алгоритмів.

Найбільш поширеною задачею в машинному навчанні є прогнозування значень Y для нових значень X. Це називається прогностичним моделюванням, і наша мета – зробити якомога більш точне прогнозування.

Це важливий аспект у контексті кваліфікаційної роботи, оскільки намагаємося класифікувати рівень безпеки пристрою на основі його характеристик. Використовуючи Scikit-Learn, є можливість навчити модель, яка найкращим чином відповідає вхідним даним (характеристикам пристрою) до вихідних даних (рівня безпеки). Це дозволить нам робити прогнози про рівень безпеки нових пристроїв.

Scikit-Learn надає доступ до багатьох різних алгоритмів класифікації. Далі буде надано короткий огляд алгоритмів [37], які найчастіше використовуються для задач класифікації.

Лінійна регресія (Linear Regression), безумовно, є одним з найвідоміших та зрозумілих алгоритмів у статистиці та машинному навчанні. Прогностичне моделювання передусім стосується мінімізації помилки моделі. Лінійну регресію можна представити у вигляді рівняння, яке описує пряму лінію, що найточніше відображає взаємозв’язок між вхідними змінними X та вихідними змінними Y. Для складання цього рівняння потрібно знайти певні коефіцієнти B для вхідних змінних. Для оцінки регресійної моделі використовуються різні методи, такі як лінійна алгебра або метод найменших квадратів [37].

Логістична регресія (Logistic Regression) – ще один алгоритм, який прийшов у машинне навчання безпосередньо зі статистики [37]. Вона добре підходить для задач бінарної класифікації. Логістична регресія схожа на лінійну тим, що в ній також потрібно знайти значення коефіцієнтів для вхідних змінних. Різниця полягає

в тому, що вихідне значення перетворюється за допомогою нелінійної або логістичної функції. Логістична функція виглядає як велика літера *S* і перетворює будь-яке значення в число в межах від 0 до 1. Це дуже корисно, оскільки ми можемо застосувати правило до виходу логістичної функції для прив'язки до 0 і 1 (наприклад, якщо результат функції менше 0,5, то на виході отримуємо 1) та прогнозування класу.

Дерево рішень (Decision Tree) можна представити у вигляді двійкового дерева, знайомого багатьом з алгоритмів та структур даних [37]. Кожен вузол представляє собою вхідну змінну та точку розділення для цієї змінної (за умови, що змінна – число). Листові вузли – це вихідна змінна, яка використовується для прогнозування. Прогнози виконуються шляхом проходження по дереву до листового вузла та виведення значення класу. Дерева швидко навчаються та роблять прогнози. Крім того, вони точні для широкого спектра завдань та не вимагають особливої підготовки даних.

Наївний байєсівський класифікатор (Naive Bayes) – це тип класифікатора, який обчислює ймовірність належності об'єкта до певного класу [37]. Ця ймовірність обчислюється з шансу, що певна подія відбудеться, з урахуванням подій, що вже відбулися. Кожен параметр класифікованого об'єкта вважається незалежним від інших параметрів. Це означає, що вплив кожного параметра на прогнозовану ймовірність вважається незалежним від інших параметрів. Це спрощує обчислення, але також може призвести до менш точних прогнозів, якщо в реальності параметри взаємопов'язані.

Випадковий ліс (Random Forest) – дуже популярний та ефективний алгоритм машинного навчання [37]. Це вид ансамблевого алгоритму, який називається пакуванням (bagging). У ньому використовується той самий підхід, але для оцінки всіх статистичних моделей найчастіше використовуються дерева рішень. Навчальні дані розбиваються на багато вибірок, для кожної з яких створюється модель. Коли потрібно зробити прогноз, його робить кожна модель, а потім прогнози усереднюють, щоб дати кращу оцінку вихідному значенню.

У алгоритмі випадкового лісу для всіх вибірок з навчальних даних будуються дерева рішень. При побудові дерев для створення кожного вузла вибираються випадкові ознаки. Окремо отримані моделі не дуже точні, але при їх об'єднанні якість прогнозування значно покращується. Якщо алгоритм з високою дисперсією, наприклад, дерева рішень, показує хороший результат на ваших даних, то цей результат часто можна покращити, застосувавши пакування.

Повертаючись до основного завдання з передбачення безпеки пристрою, першим кроком була підготовка даних. Було визначено числові та категоріальні ознаки в даних. Числові ознаки включали «BaseScore» та «Exploitability», а категоріальні ознаки включали «Type», «CVE» та «Level». Визначення числових і категоріальних ознак допомагає визначити, які методи обробки даних будуть найефективнішими. Наприклад, числові ознаки можуть потребувати нормалізації, а категоріальні ознаки – кодування.

Були створені обробники для цих ознак: StandardScaler для числових ознак та OneHotEncoder для категоріальних ознак. Обробники для ознак, такі як StandardScaler та OneHotEncoder, використовуються для перетворення даних у формат, який можна використовувати для навчання моделей. StandardScaler нормалізує числові ознаки, що допомагає моделям краще інтерпретувати ці ознаки. OneHotEncoder перетворює категоріальні ознаки на бінарний формат, який можна використовувати в моделях машинного навчання. Ці обробники допомогли нормалізувати числові ознаки та перетворити категоріальні ознаки на формат, який можна використовувати для навчання моделей.

Дані були розділені на навчальну, тестову та валідаційну вибірки. Використання валідаційної вибірки допомогло уникнути перевірки моделі на тих самих даних, на яких вона навчалася.

Після підготовки даних, було почато процес навчання моделей, використовуючи різні алгоритми класифікації, доступні в Scikit-Learn. Це включає в себе лінійну регресію, логістичну регресію, дерева рішень, наївний байєсівський класифікатор та випадковий ліс.

Кожна з цих моделей була навчена (рис 4.11), а потім була оцінена їх точність на тестових даних. Це допомогло визначити, яка модель найкраще підходить для поставленої задачі.

```python
# Визначаємо числові та категоріальні ознаки
num_features = ['BaseScore', 'Exploitability']
cat_features = ['Type','CVE', 'Level']

# Створюємо обробники для числових і категоріальних ознак
num_transformer = StandardScaler()
cat_transformer = OneHotEncoder(handle_unknown='ignore')

# Об'єднуємо обробники в один
preprocessor = ColumnTransformer(
    transformers=[
        ('num', num_transformer, num_features),
        ('cat', cat_transformer, cat_features)])

# Розділяємо дані на ознаки (X) і цільову змінну (y)
X = newDf.drop('Color', axis=1)
y = newDf['Color']

# Розділяємо дані на навчальну, тестову та валідаційну вибірки
X_temp, X_val, y_temp, y_val = train_test_split(X, y, test_size=0.15, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X_temp, y_temp, test_size=0.2, random_state=42)

# Зберігаємо валідаційний набір даних у новий файл Excel
val_df = pd.concat([X_val, y_val], axis=1)
val_df = val_df.drop('Color', axis=1)
val_df.to_excel('validation_set.xlsx', index=False)
files.download('validation_set.xlsx')

# Застосовуємо препроцесор до навчальних і тестових даних
X_train = preprocessor.fit_transform(X_train[num_features + cat_features])
X_test = preprocessor.transform(X_test[num_features + cat_features])
```

Рисунок 4.10 – Підготовка даних перед навчанням

```python
# Створюємо і навчаємо модель Random Forest
model_randFor = RandomForestClassifier(random_state=42)
model_randFor.fit(X_train, y_train)
y_pred = model_randFor.predict(X_test)
accuracy_randFor = accuracy_score(y_test, y_pred)
print(f'Точність моделі(Random Forest): {accuracy_randFor}')

# Створюємо і навчаємо модель Decision Tree
model_DesTree = DecisionTreeClassifier(random_state=42)
model_DesTree.fit(X_train, y_train)
y_pred = model_DesTree.predict(X_test)
accuracy_DesTree = accuracy_score(y_test, y_pred)
print(f'Точність моделі(Decision Tree): {accuracy_DesTree}')

# Створюємо і навчаємо модель Logistic Regression
model_LogReg = LogisticRegression(max_iter=1000, random_state=42)
model_LogReg.fit(X_train, y_train)
y_pred = model_LogReg.predict(X_test)
accuracy_LogReg = accuracy_score(y_test, y_pred)
print(f'Точність моделі(Logistic Regression): {accuracy_LogReg}')

# Створюємо і навчаємо модель k-NN
model_knn = KNeighborsClassifier()
model_knn.fit(X_train, y_train)
y_pred = model_knn.predict(X_test)
accuracy_kNN = accuracy_score(y_test, y_pred)
print(f'Точність моделі(k-NN): {accuracy_kNN}')

# Створюємо і навчаємо модель Gradient Boosting
model_gb = GradientBoostingClassifier(random_state=42)
model_gb.fit(X_train, y_train)
y_pred = model_gb.predict(X_test)
accuracy_gb = accuracy_score(y_test, y_pred)
print(f'Точність моделі(Gradient Boosting): {accuracy_gb}')
```

Рисунок 4.11 – Приклад навчання різних моделей, використовуючи бібліотеку Scikit-Learn

Для того, щоб проаналізувати результати кожної моделі, можна порівняти їхню точність та вибрати найкращу модель для розв’язуваної задачі. Також можна провести додаткову оптимізацію моделі, використовуючи методи налаштування гіперпараметрів, такі як пошук сітки або випадковий пошук. Порівняльний графік точності кожної моделі представлено на рис. 4.12.

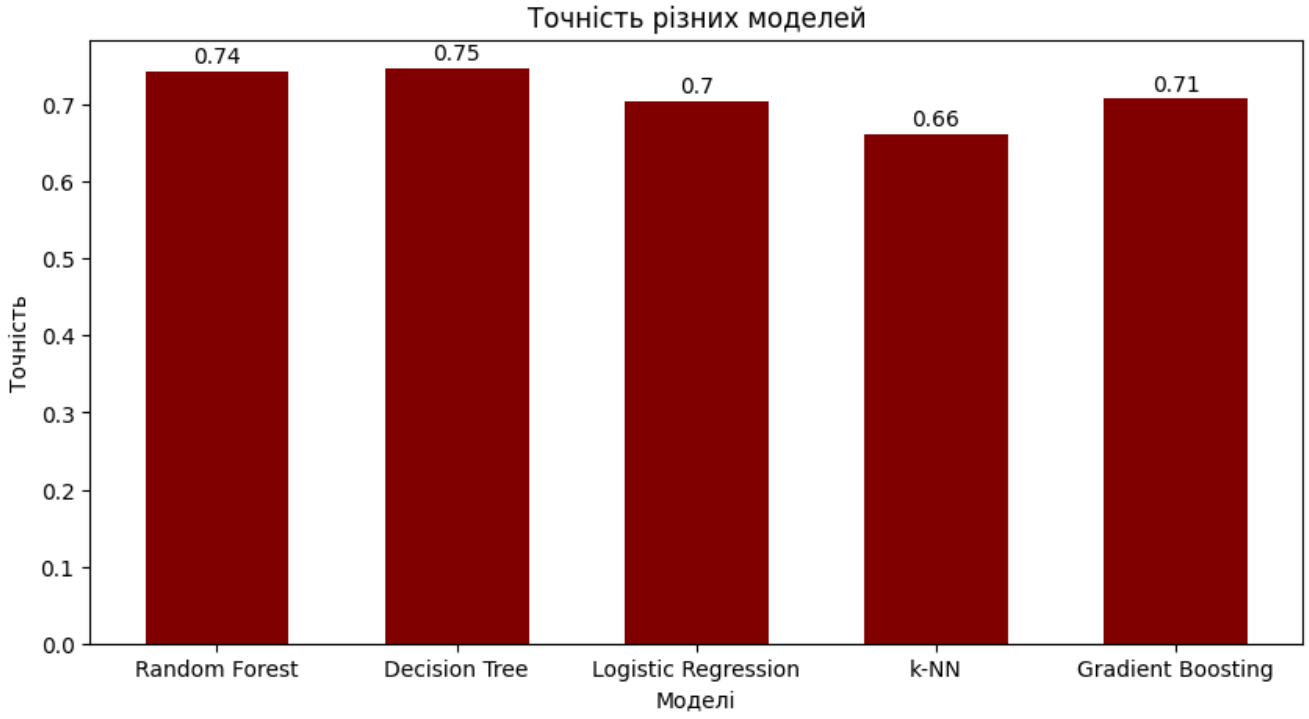

Рисунок 4.12 – Графік точності різних моделей машинного навчання: Random Forest, Decision Tree, Logistic Regression, k-NN та Gradient Boosting.

З рисунку 4.12 видно, що моделі Random Forest та Decision Tree показали найвищу точність у порівнянні з іншими моделями. Це може свідчити про те, що ці моделі були найбільш ефективними для даної задачі класифікації. Logistic Regression, k-NN та Gradient Boosting показали трохи нижчу точність. Це може бути пов’язано з особливостями даних або специфікою задачі. Загалом, ці результати підкреслюють важливість проведення ретельного порівняння різних моделей машинного навчання, щоб вибрати найкращу модель для конкретної задачі.

Наступним етапом була побудована матриця помилок (рис. 4.13) для моделі дерева рішень. Було обрано саме цю модель, бо в неї найкращий результат точності порівняно з іншими моделями. Матриця помилок дозволяє оцінити ефективність

моделі, виявляючи не тільки загальну точність, але й специфічні помилки класифікації.

Використовуючи матрицю помилок, було виявлено, що модель дерева рішень найкраще класифікує «жовтий» клас, але має деякі труднощі з «коричневим» та «помаранчевим» класами. Це може свідчити про те, що модель може потребувати додаткового налаштування для покращення її продуктивності на цих конкретних класах.

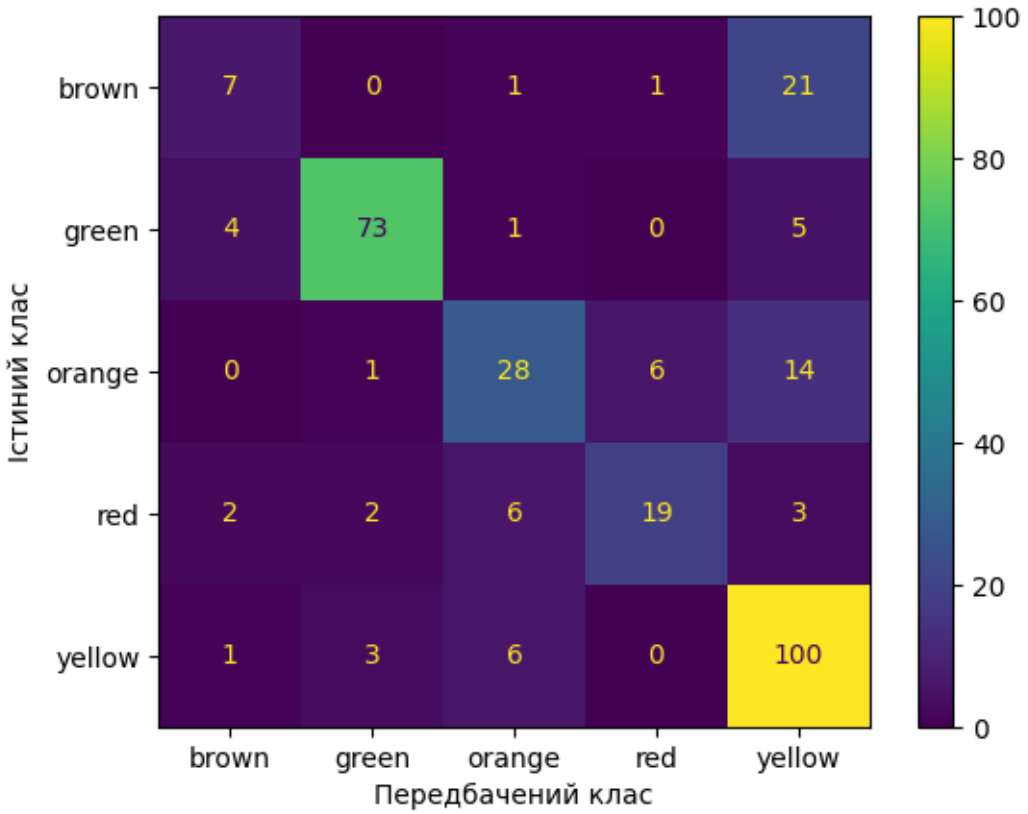

Рисунок 4.13 – Матриця помилок для моделі дерева рішень

Осі графіка позначені як «Передбачений клас» та «Істинний клас», і обидві вони варіюються від п'яти різних класів.

Числа в комірках представляють кількість прикладів для кожної комбінації істинного і передбаченого класів. Числа на діагоналі відображають кількість правильно класифікованих екземплярів для кожного класу. Це називається істинно позитивними передбаченнями. Числа поза діагоналлю представляють помилково класифіковані екземпляри. Це називається «хибно позитивними» і «хибно негативними» передбаченнями.

У даному випадку менша точність моделі для «коричневого» та «помаранчевого» класів може бути пов’язана з недостатнім обсягом даних для цих класів. Тому, одним з можливих шляхів покращення цих результатів може бути збір більшої кількості даних для тренування моделей. Збільшення обсягу даних може допомогти моделям краще вивчити шаблони цих класів і, таким чином, покращити їхню точність класифікації.

Останнім етапом перевірки працездатності навченої моделі є перевірка її на валідаційному наборі даних. Цей набір даних не був використаний під час навчання моделі, тому він дозволяє перевірити, наскільки добре модель може узагальнювати своє навчання на нових, невідомих для неї даних. Код для цих налаштувань представлено на рис. 4.14.

```
# Зберігаємо модель
joblib.dump(model_DesTree, 'model_DesTree.pkl')

# Завантажуємо модель
loaded_model = joblib.load('model_DesTree.pkl')

test_xls = pd.ExcelFile('/content/drive/MyDrive/validation_set.xlsx')
new_data = pd.read_excel(test_xls )
print('Дані до передбаченням:\n', new_data.head())
# Застосовуємо препроцесор до нових даних
new_data_preprocessed = preprocessor.transform(new_data)

# Використовуємо модель для передбачення на нових даних
predictions = loaded_model.predict(new_data_preprocessed)

# Додаємо передбачення у вихідний DataFrame і зберігаємо його в новий файл Excel
new_data['Color'] = predictions
new_data.to_excel('predicted_data.xlsx', index=False)

files.download('predicted_data.xlsx')
# files.download('model_DesTree.pkl')
print('\n\n Дані після передбачення:\n',new_data.head())
```

Рисунок 4.14 – Код для перевірки навченої моделі на валідаційних даних

У цьому коді модель Decision Tree, яка була навчена раніше, зберігається за допомогою методу dump і потім завантажується за допомогою методу load. Це

дозволяє зберегти стан моделі, щоб її можна було використовувати пізніше без необхідності повторного навчання.

Новий набір даних завантажується з файлу Excel, а потім передбачення генеруються за допомогою завантаженої моделі. Ці передбачення потім додаються до вихідного набору даних і зберігаються в новому файлі Excel. Результат передбачення можна побачити на рисунку 4.15.

```
Дані до передбаченням:
                         Device     Type   ISR            CVE  BaseScore     Level  Exploitability
0    Cisco Catalyst 3750E-48PD-EF  unknown  3.82   CVE-2017-3881        9.8  CRITICAL            0.39
1             Cisco Meraki MR53E  unknown  5.54  CVE-2020-26140        6.5    MEDIUM            0.28
2          Cisco Nexus 9364D-GX2A   switch  1.82  CVE-2023-20089        6.5    MEDIUM            0.28
3            Cisco C6800-8p10g-xl   switch  0.54   CVE-2019-1649        6.7    MEDIUM            0.08
4  Cisco Webex Wireless Phone 860  unknown  1.82  CVE-2020-26141        6.5    MEDIUM            0.28


 Дані після передбачення:
                         Device     Type   ISR            CVE  BaseScore     Level  Exploitability   Color
0    Cisco Catalyst 3750E-48PD-EF  unknown  3.82   CVE-2017-3881        9.8  CRITICAL            0.39  yellow
1             Cisco Meraki MR53E  unknown  5.54  CVE-2020-26140        6.5    MEDIUM            0.28  orange
2          Cisco Nexus 9364D-GX2A   switch  1.82  CVE-2023-20089        6.5    MEDIUM            0.28   green
3            Cisco C6800-8p10g-xl   switch  0.54   CVE-2019-1649        6.7    MEDIUM            0.08   green
4  Cisco Webex Wireless Phone 860  unknown  1.82  CVE-2020-26141        6.5    MEDIUM            0.28   green
```

Рисунок 4.15 – Результат роботи моделі машинного навчання на валідаційному наборі даних

З рисунку 4.15 видно, що після передбачення у даних з’явився новий стовпець з кольором. Цей стовбець і є результат роботи моделі МН. З рисунку видно, що перші 5 пристроїв були класифіковані правильно. Результат передбачення для всіх пристроїв показано на рис. 4.16.

```
Всього пристроїв: 269
Кількість правильно передбачених пристроїв: 210
```

Рисунок 4.16 – Результат роботи моделі машинного навчання на валідаційному наборі даних

Результати рисунка 4.16 показують, що модель здатна класифікувати нові дані з високою точністю, бо з 269 пристроїв правильно передбачено було 210, що свідчить про її ефективність. Однак, як і з будь-якою моделлю машинного навчання, важливо пам’ятати, що точність передбачення може залежати від якості

та репрезентативності вхідних даних. Тому, хоча ці результати є обнадійливими, завжди варто проводити додаткову валідацію моделі з використанням нових даних для перевірки її продуктивності.

4.4 Висновки до четвертого розділу

У цьому розділі кваліфікаційної роботи було розглянуто проблему класифікації безпеки пристроїв за допомогою машинного навчання. Було використано числові та категоріальні ознаки для передбачення рівня безпеки пристрою.

Було проведено порівняння різних моделей машинного навчання, включаючи лінійну регресію, логістичну регресію, дерева рішень та випадковий ліс, з метою визначення найкращої моделі для задачі. Використання матриці помилок для оцінки ефективності моделі, дозволило виявити специфічні помилки класифікації та визначити, які класи модель визначає найкраще.

Точність моделі може бути покращена шляхом збору більшої кількості даних для тренування моделей. Це допоможе моделям краще вивчити шаблони цих класів і таким чином покращити їхню точність класифікації.

Була зроблена перевірка моделі на валідаційному наборі даних, який не був використаний під час навчання моделі. Це дозволило перевірити, наскільки добре модель може узагальнювати своє навчання на нових, невідомих для неї даних.

У наступному розділі кваліфікаційної роботи буде розглянуто приклад використання цієї моделі для розв’язання задачі одношляхової маршрутизації.

5 ІНТЕГРАЦІЯ МОДЕЛІ МАШИННОГО НАВЧАННЯ У ЗАВДАННЯ ІНТЕЛЕКТУАЛЬНОЇ МАРШРУТИЗАЦІЇ

У попередньому розділі було розглянуто розробку та валідацію моделі машинного навчання, яка здатна класифікувати пристрої за параметрами безпеки. Модель була навчена та протестована на валідаційному наборі даних, що дозволило оцінити її здатність узагальнювати навчання на нових, невідомих для моделі даних.

У цьому розділі увагу зосереджено на розробці програмного модулю, який використовує прогнозовані дані навченої моделі для вирішення задачі оптимізації. Цей модуль буде використовувати вихідні дані для моделі визначення оптимального шляху в задачі одношляхової маршрутизації з різнотипними метриками.

Цей розділ демонструє, як можна інтегрувати результати роботи моделі машинного навчання в задачу оптимізації, що дозволяє використовувати передбачувану інформацію про безпечність пристроїв для покращення процесу прийняття рішень.

5.1 Розробка програмного модулю для використання прогнозованих даних

У цьому підрозділі буде розглянуто, як можна використати результати класифікації мережних пристроїв на основі моделі машинного навчання, розробленої в попередньому розділі, для вирішення задачі оптимізації.

Модель машинного навчання з попереднього розділу вже навчена та протестована на валідаційному наборі даних. Тепер можна використати цю модель для прогнозування пристроїв з найвищим рівнем безпеки для подальшого їх використання при проектуванні мережі. Ці прогнозовані дані потім можуть бути використані для аналітичного моделювання та розв’язання маршрутної задачі.

Програмний модуль використовує прогнозовані дані моделі для вибору найкращих пристроїв на основі їхнього рівня безпеки за допомогою змінної, яку прогнозувала модель. Модуль повинен дозволяти користувачеві обрати тип пристрою, який він хоче використовувати, та кількість пристроїв, які йому потрібні. Код розробленого модулю представлено на рис. 5.1.

```python
used_devices = []

color_order = ['green', 'yellow', 'orange', 'red', 'brown']
num_devices = int(input(f"Яка кількість пристроїв вам необхідна? (максимум :{new_data.shape[0]})\n"))
device_type = input("Який тип пристрої необхідно? (switch, router, all)\n") #  switch, router, unknown
if num_devices > new_data.shape[0]:
    print("Вибачте, недостатньо пристроїв для вашого запиту.")
else:
    new_data['Color'] = new_data['Color'].astype('category')
    new_data['Color'] = new_data['Color'].cat.set_categories(color_order, ordered=True)
    sorted_data = new_data.sort_values(by=['Color', 'ISR'])

    # Фільтруємо дані за типом пристрою, якщо вказано
    if device_type != 'all':
        sorted_data = sorted_data[sorted_data['Type'] == device_type]

    best_devices = []
    for _, device in sorted_data.iterrows():
        if len(best_devices) == num_devices:
            break
        if device['ISR'] not in used_devices:
            best_devices.append(device)
            used_devices.append(device['ISR'])

    best_devices_df = pd.DataFrame(best_devices)
    print("Ось найбільш підходящі пристрої для вас:\n")
    print(best_devices_df)
```

Рисунок 5.1 – Код програмного модулю для використання прогнозованих даних

Код починається з того, що користувач вводить кількість пристроїв, які йому потрібні, та тип пристрою, який він хоче використовувати. Ці дані потрібні для фільтрації доступних пристроїв.

Дані сортуються за кольором. Колір використовується як індикатор безпеки пристрою. Якщо користувач вказав конкретний тип пристрою, дані фільтруються, щоб включати лише пристрої цього типу. Код проходить через відсортовані та відфільтровані дані, вибираючи найкращі пристрої.

Пристрій вважається «найкращим», якщо його ISR ще не було використано, і він додається до списку найкращих пристроїв. Цей процес продовжується до тих пір, поки не буде вибрано потрібну кількість пристроїв. Наприкінці код виводить список найкращих пристроїв для користувача, це наведено на рис. 5.2.

```
Яка кількість пристроїв вам необхідна? (максимум :269)
6
Який тип пристрої необхідно? (switch, router, all)
router
Ось найбільш підходящі пристрої для вас:

                                Device    Type   ISR            CVE  BaseScore   Level  Exploitability   Color
40                   Cisco NIM-1ce1t1-pri  router  0.54   CVE-2019-1649        6.7  MEDIUM            0.08   green
203  Cisco Cbr8 Converged Broadband Router  router  1.30  CVE-2023-20081        5.9  MEDIUM            0.22   green
148               Cisco 800 Series Routers  router  1.82   CVE-2018-0163        6.5  MEDIUM            0.28   green
211                           Cisco 2951  router  2.60   CVE-2018-0179        5.9  MEDIUM            0.22  yellow
251                         Cisco IR1101  router  2.64   CVE-2019-1950        8.4    HIGH            0.25  yellow
262                Cisco XR 12404 Router  router  2.67  CVE-2019-16027        6.5  MEDIUM            0.28  yellow
```

Рисунок 5.2 – Результат виконання програмного модулю для використання моделі

З рисунку 5.2 видно, що після виконання цього коду було отримано список найкращих пристроїв для використання. Ці пристрої вибрані на основі їхнього рівня безпеки та ISR, що були прогнозовані нашою моделлю машинного навчання.

Отримані результати можна використовувати для вирішення реальних задач, а саме можливо інтегрувати результати моделі машинного навчання в задачу оптимізації, що дозволить використовувати передбачувану інформацію про безпечність пристроїв для покращення процесу прийняття рішень.

5.2 Дослідження рішення одношляхової маршрутизації для різнотипних метрик

У цьому підрозділі буде розглянуто, як вибрані пристрої можуть бути використані в задачі одношляхової маршрутизації.

Для постановки задачі маршрутизації щодо визначення оптимального шляху від вузла-відправника до вузла-отримувача в мережі з використанням різнотипних метрик необхідно зазначити такі параметри:

- кількість каналів (n);
- кількість вузів (m);
- вузол-відправник (s);
- вузол-отримувач (d);
- пропускні здатності каналів ($c_{i,j}$);
- вектор метрик каналів(f);
- імовірності компрометації каналів ($p_{i,j}$).

В мережі з n каналами зв’язку, вектор $\vec{x}$ має відповідну розмірність, де його координати $x_{i,j}$ відображають частку потоку в каналі зв’язку між i-м і j-м вузлами.

Аналогічно, розмірність вектора метрик f також відповідає кількості каналів зв'язку в мережі n, а його координати $f_{i,j}$ представляють метрику каналу зв'язку між i-м і j-м вузлами.

Щодо реалізації одношляхової маршрутизації, на координати вектору $\vec{x}$ накладаються певні обмеження [38], які визначаються за формулою:

$$x_{i,j} \in \{0,1\} \text{ при } i,j = \overline{1,m}, i \neq j\ , \tag{5.1}$$

Булеві значення змінних (5.1) забезпечують відсутність розгалуження потоку по мережі. Це означає, що всі пакети потоку будуть передаватися одним шляхом. Припустимо, що кожному каналу зв'язку присвоюється метрика $f_{i,j}$.

Під час вирішення маршрутної задачі важливо дотримуватися умов збереження потоку [38], що повинні бути виконані в кожному мережному вузлі, а також у мережі в цілому. Це означає, що кількість вхідного та вихідного потоку для кожного вузла повинна бути збалансована:

$$\begin{cases} \sum_{j:(i,j)} x_{i,j} - \sum_{j:(i,j)} x_{j,i} = 1, i = s; \\ \sum_{j:(i,j)} x_{i,j} - \sum_{j:(i,j)} x_{j,i} = 1, i \neq s, d; \\ \sum_{j:(i,j)} x_{i,j} - \sum_{j:(i,j)} x_{j,i} = -1, i = d. \end{cases} \tag{5.2}$$

В сучасних мережних протоколах, задачі одношляхової маршрутизації зазвичай вирішуються шляхом пошуку найкоротшого шляху в мережі, яка відображає структуру телекомунікаційної системи. Задача пошуку найкоротшого шляху в мережі може бути формалізована як задача булевого програмування. Для її розв'язання використовується бібліотека GEKKO Optimization Suite [39].

Згідно з виразом (5.1), при розв'язанні задач булевого програмування потрібно мінімізувати лінійну цільову функцію [38]. Це означає, що ми шукаємо такі значення булевих змінних, які зводять до мінімуму значення цієї функції:

$$min_x \vec{f}^t \vec{x} \tag{5.3}$$

При розв'язанні задач булевого програмування, необхідно враховувати ряд умов, які представлені у вигляді обмежень рівнянь і нерівностей. Ці обмеження

визначають допустимий простір рішень і забезпечують, що отримане рішення відповідає всім необхідним вимогам [38]:

$$A_{eq}\,\vec{x} = \overrightarrow{b_{eq}}, \overrightarrow{lb} \le \vec{x} \le \overrightarrow{ub}\,, \tag{5.4}$$

де A_{eq} – матриця відповідної розмірності.

Припустимо, що структура мережі ілюстрована на рисунку 5.3. Кожний канал зв'язку має вказану пропускну здатність (у чисельнику) та імовірність компрометації (у знаменнику), яка обирається рівною параметру Exploitability отриманих в результаті класифікації найбільш безпечних пристроїв (рис. 5.2). В цьому випадку, загальна кількість вузлів у мережі становить шість (m = 6), а кількість каналів зв'язку – дев'ять (n = 9). Окрім того, вузол, який відправляє пакети – це вузол 1, а вузол, який отримує пакети – це вузол 6.

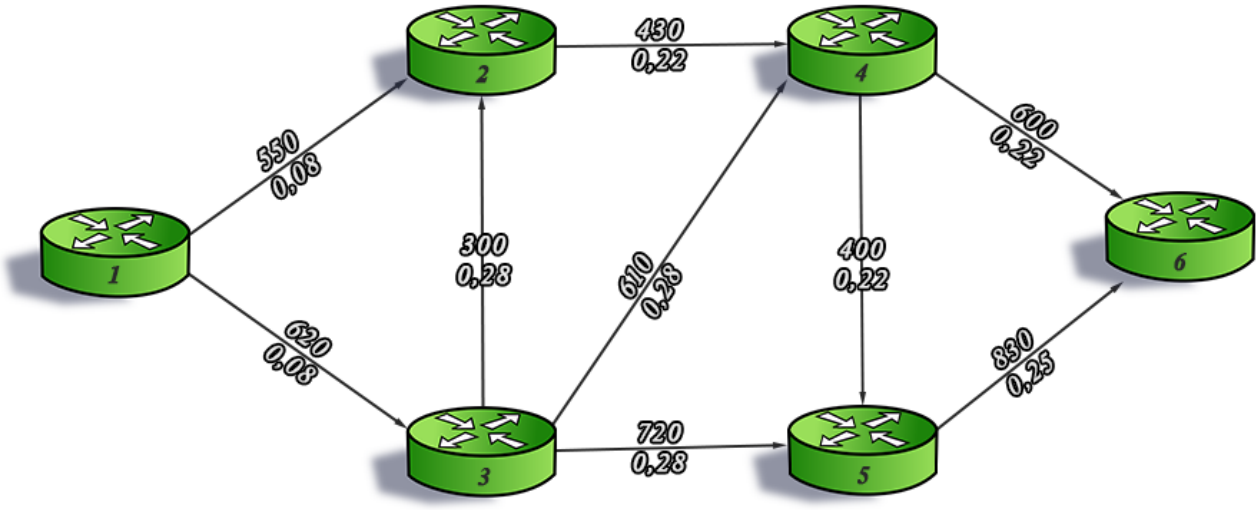

Рисунок 5.3 – Структура мережі для моделювання

Враховуючи структуру мережі, представлену на рисунку 5.3, та враховуючи характеристики кожного каналу зв'язку, включаючи пропускну здатність та імовірність компрометації, можна перейти до важливого аспекту цього дослідження. Цей аспект стосується вибору обраних пристроїв та оцінки їх потенційної вразливості до компрометації.

Ці пристрої були визначені за допомогою моделі, розробленої в попередньому розділі цієї кваліфікаційної роботи. Інформація про них

представлена у таблиці 5.1. Використовуючи цю модель, можливо отримати інформацію щодо імовірності компрометації каналів зв’язку мережі та шляхів передачі пакетів від вузла-відправника (вузол 1) до вузла-отримувача (вузол 6) (табл. 5.2).

Таблиця 5.1 – Пристрої, обрані за допомогою моделі машинного навчання

№	Маршрутизатор	CVE	Exploitability
1	Cisco NIM-1ce1t1-pri	CVE-2019-1649	0,08
2	Cisco Cbr8 Converged Broadband	CVE-2023-20081	0,22
3	Cisco 800 Series	CVE-2018-0163	0,28
4	Cisco 2951	CVE-2018-0179	0,22
5	Cisco IR1101	CVE-2019-1950	0,25
6	Cisco XR 12404	CVE-2019-16027	0,28

Таблиця 5.2 – Характеристики каналів зв’язку фрагмента мережі

№	Канал	Пропускна здатність, пак/с	Імовірність компрометації
1	(1,2)	550	0,08
2	(1,3)	620	0,08
3	(2,4)	430	0,22
4	(3,2)	300	0,28
5	(3,4)	610	0,28
6	(3,5)	720	0,28
7	(4,5)	400	0,22
8	(4,6)	600	0,22
9	(5,6)	830	0,25

Тепер можна сформувати шуканий вектор $\vec{x}$. В рамках моделі, представленої виразами (5.1-5.3), він має наступний вигляд [38]:

$$\vec{x} = \begin{bmatrix} x_{1,2} \\ x_{1,3} \\ x_{2,4} \\ x_{3,2} \\ x_{3,4} \\ x_{3,5} \\ x_{4,5} \\ x_{4,6} \\ x_{5,6} \end{bmatrix}. \quad (5.5)$$

Для використання метрики, аналогічної метриці протоколу RIP (Routing Information Protocol), вектор $\vec{f}$ [38] буде представлений у наступному вигляді:

$$\vec{f} = \begin{bmatrix} f_{1,2} \\ f_{1,3} \\ f_{2,4} \\ f_{3,2} \\ f_{3,4} \\ f_{3,5} \\ f_{4,5} \\ f_{4,6} \\ f_{5,6} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}. \quad (5.6)$$

Формула (5.6) означає, що всі канали зв’язку мають однакову вагу в контексті протоколу RIP, тобто відстань між будь-якими двома вузлами в мережі вважається однаковою, незалежно від їх фактичного розташування або характеристик каналу зв’язку. Це зумовлено особливістю метрик протоколу RIP.

При використанні метрики, аналогічної метриці протоколу OSPF, вектор маршрутних метрик $\vec{f}$ буде мати форму [38]:

$$\vec{f} = \begin{bmatrix} f_{1,2} \\ f_{1,3} \\ f_{2,4} \\ f_{3,2} \\ f_{3,4} \\ f_{3,5} \\ f_{4,5} \\ f_{4,6} \\ f_{5,6} \end{bmatrix} = \begin{bmatrix} {}^{10^8}/_{550} \\ {}^{10^8}/_{620} \\ {}^{10^8}/_{300} \\ {}^{10^8}/_{430} \\ {}^{10^8}/_{610} \\ {}^{10^8}/_{720} \\ {}^{10^8}/_{400} \\ {}^{10^8}/_{600} \\ {}^{10^8}/_{830} \end{bmatrix}. \quad (5.7)$$

При впровадженні безпечної маршрутизації, координати вектора маршрутних метрик $\vec{f}$ приймають наступний вигляд [38]:

$$f_{i,j} = -\lg(1 - p_{i,j}). \quad (5.8)$$

Ця формула (5.8) використовується для визначення метрики каналу зв'язку в контексті безпечної маршрутизації. Вона використовує логарифм для перетворення імовірностей компрометації в ваги, які можна використовувати при розв'язанні маршрутної задачі оптимізації. Це допомагає зменшити вплив високих імовірностей та перетворити імовірності в діапазоні від 0 до 1 на значення, які можуть бути більшими або меншими за 1. Мінус перед логарифмом використовується для перетворення від'ємних значень на додатні, що робить метрику більш інтуїтивно зрозумілою.

Згідно (5.2) можна формалізувати збереження потоку у вузлах мережі:

$$\begin{cases} x_{1,2} + x_{1,3} = 1; \\ x_{1,2} - x_{2,3} + x_{2,4} = 0; \\ -x_{1,3} + x_{3,2} + x_{3,4} + x_{3,5} = 0; \\ -x_{2,4} - x_{3,4} + x_{4,5} + x_{4,6} = 0; \\ -x_{3,5} - x_{4,5} + x_{5,6} = 0; \\ -x_{4,6} - x_{5,6} = -1. \end{cases} \quad (5.9)$$

Тому, згідно цієї системи, можна сформувати матрицю [38] A_{eq} і вектор $\overrightarrow{b_{eq}}$:

$$A_{eq} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix}. \quad (5.10)$$

$$\vec{b}_{eq} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}. \quad (5.11)$$

Через те, що всі координати вектора $\vec{x}$ є булевими (5.1), то необхідно встановити обмеження на значення [38], які можуть приймати елементи вектора:

$$\overrightarrow{lb} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{та} \quad \overrightarrow{ub} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (5.12)$$

В результаті дослідження, змінивши значення маршрутних метрик, необхідно розрахувати за допомогою програмного модулю оптимальні маршрути, встановивши для кожного кількість переходів, пропускну здатність та імовірність компрометації каналу.

Розроблений модуль, для вирішення одношляхової задачі оптимізації, представлено на рис. 5.4 та 5.5.

```python
from gekko import GEKKO  # Імпортуємо бібліотеку GEKKO для оптимізації
import numpy as np  # Імпортуємо бібліотеку numpy для роботи з масивами та математичних операцій

m = GEKKO()  # Ініціалізуємо модель GEKKO
c = [550,620,430,300,610,720,400,600,830] #  вектор пропускних здатностей каналів зв'язку
p = [0.08,0.08,0.22,0.28,0.28,0.28,0.22,0.22,0.25] # вектор імовірностей компрометації каналів зв'язку

def calculate_metrics(c, p):
    fRIP = [1, 1, 1, 1, 1, 1, 1, 1, 1]  # Метрика RIP
    fOSPF = [(10**8)/c[i] for i in range(len(c))]  # Метрика OSPF
    fSEC = [-np.log(1-p[i]) for i in range(len(p))]  # Безпечна метрика
    return fRIP, fOSPF, fSEC

fRIP, fOSPF, fSEC = calculate_metrics(c, p)

def calc(f, type): # Функція calc виконує оптимізацію для заданої метрики
  x = [m.Var(lb=0, ub=1, integer=True) for i in range(9)] # Ініціалізуємо змінні моделі та встановлюємо їхні значення
  for xi in x:
      xi.value = 0
  # Обмеження у формі рівнянь
  m.Equation(x[0]+x[1]==1)  # для вузла-відправника пакетів
  m.Equation(-x[0]+x[2]-x[3]==0)  # для всіх транзитних вузлів
  m.Equation(-x[1]+x[3]+x[4]+x[5]==0)
  m.Equation(-x[2]-x[4]+x[6]+x[7]==0)
  m.Equation(-x[5]-x[6]+x[8]==0)
  m.Equation(-x[7]-x[8]==-1)  # для вузла-отримувача пакетів
  m.Minimize(sum(f[i]*x[i] for i in range(9)))  # Цільова функція
  m.options.SOLVER=1  # Використовуємо APOPT як розв'язувач MINLP
  m.solve(disp = False)  # Розв'язуємо задачу оптимізації
```

Рисунок 5.4 – Перша частина коду для вирішення задачі одношляхової маршрутизації з різнотипними метриками

```python
  m.solve(disp = False)  # Розв'язуємо задачу оптимізації

  print('')   # Виводимо результати розрахунку
  print(f'Результати розрахунку для {type}:')
  print('x1: ' + str(x[0].value)+'   (1,2)')
  print('x2: ' + str(x[1].value)+'   (1,3)')
  print('x3: ' + str(x[2].value)+'   (2,4)')
  print('x4: ' + str(x[3].value)+'   (3,2)')
  print('x5: ' + str(x[4].value)+'   (3,4)')
  print('x6: ' + str(x[5].value)+'   (3,5)')
  print('x7: ' + str(x[6].value)+'   (4,5)')
  print('x8: ' + str(x[7].value)+'   (4,6)')
  print('x9: ' + str(x[8].value)+'   (5,6)')
  # Обчислюємо пропускну здатність та імовірність компрометації маршруту
  x_values = [x[i] for i in range(9)]
  x_elements = [i for i, x in enumerate(x_values) if x.value[0] == 1.0]
  min_metric = min([c[i] for i in x_elements])

  prob = [p[i] for i in x_elements]
  probability_compromise = round(1 - np.prod([1 - p_value for p_value in prob]), 4)
  print(f'Пропускна здатність маршруту з метрикою {type} складає: {min_metric}')
  print(f'Імовірність компрометації маршруту з метрикою {type} складає: {probability_compromise}')

  # Виводимо маршрут
  route_map = ['1-2', '1-3', '2-4', '3-2', '3-4', '3-5', '4-5', '4-6', '5-6']
  route = ' - '.join([route_map[i] for i in x_elements])
  if route:
      print(f'Маршрут: {route}')

calc(fRIP,"RIP")
calc(fOSPF,"OSPF")
calc(fSEC,"Secure Metric")
```

Рисунок 5.5 – Друга частина коду для вирішення задачі одношляхової маршрутизації з різнотипними метриками

Результати виконання коду, а саме, розрахунок пропускної здатності та імовірності компрометації для маршрутів, побудованих за допомогою різних метрик, можна побачити у таблиці 5.3 та рисунках 5.6 і 5.7. Водночас імовірність компрометації маршруту обчислюється наступним чином:

$$P(1 \to 2 \to 4 \to 6) = 1 - (1 - p_{1,2})(1 - p_{2,4})(1 - p_{4,6}), \quad (5.13)$$

$$P(1 \to 3 \to 5 \to 6) = 1 - (1 - p_{1,3})(1 - p_{3,5})(1 - p_{5,6}). \quad (5.14)$$

Таблиця 5.3 – Порядок маршрутизації та характеристики маршрутів, розрахованих за різними метриками

Метрика	Маршрут	Пропускна здатність	Імовірність компрометації
RIP	1 → 2→ 4 → 6	430	0,4403
OSPF	1 → 3 → 5 → 6	620	0,5032
Security	1 → 2→ 4 → 6	430	0,4403

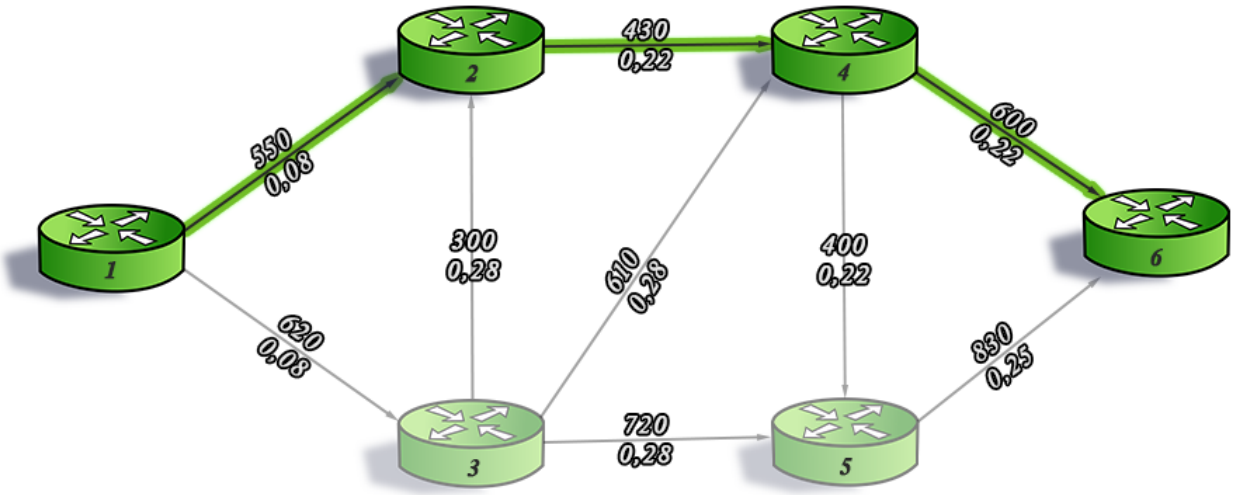

Рисунок 5.6 – Маршрут для метрик по аналогії з RIP та Security

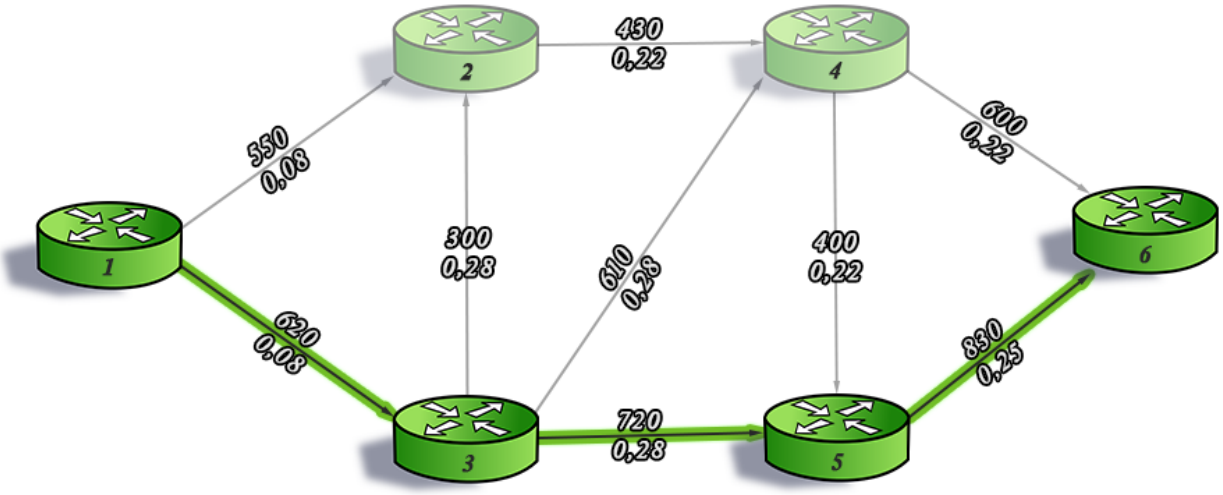

Рисунок 5.7 – Маршрут для метрики по аналогії з OSPF

Отримані результати показують, що для метрик по аналогії з RIP та Security Metric було отримано однаковий маршрут. Це може статися, якщо метрики, які використовуються цими протоколами, призводять до вибору одного й того ж маршруту як оптимального.

Протокол RIP використовує метрику, засновану на мінімальній кількості хопів, тобто кількості вузлів, які потрібно пройти, щоб дійти від початкового до кінцевого вузла. Security Metric використовує метрику, яка враховує безпеку каналу зв’язку, та обирає маршрут, у якого найменша ймовірність компрометації.

Якщо в обох випадках оптимальний маршрут є тим, який мінімізує кількість хопів і максимізує безпеку, то ці два протоколи можуть вибрати один і той же маршрут. З рисунку 5.6 видно, що маршрут з найменшою кількістю переходів також є найбезпечнішим маршрутом.

5.3 Висновки до п’ятого розділу

У цьому розділі кваліфікаційної роботи було розглянуто розробку програмного модулю, який використовує прогнозовані дані навченої моделі для вирішення задачі оптимізації. Були використані результати моделі машинного навчання, розробленої в попередньому розділі, для вирішення задачі одношляхової маршрутизації.

Було досліджено, як вибрані пристрої можуть бути використані при розв’язанні оптимізаційної задачі одношляхової маршрутизації. Визначені критерії

для вибору найкращих пристроїв на основі їхнього рівня безпеки та інших характеристик, які були прогнозовані моделлю машинного навчання.

Розглянуто використання різних метрик для визначення оптимального маршруту в мережі. Використано метрики протоколів RIP, OSPF та безпечної маршрутизації для визначення оптимального маршруту.

ВИСНОВКИ

У кваліфікаційній роботі проведено аналіз та дослідження методів інтелектуальної маршрутизації в програмно-конфігурованих мережах.

З цією метою, було досліджено використання засобів штучного інтелекту в інфокомунікаційних мережах. З’ясовано, що традиційні мережні архітектури не відповідають вимогам нових інфокомунікаційних сервісів. Було розглянуто основні проривні технології в сфері мережних технологій за останні два десятиліття, а саме програмно-конфігуровані мережі SDN, віртуалізація мережних функцій NFV та машинне навчання ML. Також було розглянуто класифікацію алгоритмів машинного навчання трьох різних категорії: SL, UL, RL.

Проведено аналіз особливостей інтелектуального управління трафіком в інфокомунікаційних мережах. Було акцентовано увагу на використанні глибокого навчання з підкріпленням для вирішення складних проблем оптимізації мережі. Проаналізовано переваги DRL, включаючи можливість отримання рішень для складної оптимізації мережі, навчання і накопичення знань про комунікаційне обладнання та мережне середовище, автономне прийняття рішень, покращення швидкості навчання, особливо в задачах з великими просторами станів і дій, та вирішення різних проблем в комунікаційних мережах.

Досліджено використання методів інтелектуальної маршрутизації в програмно-конфігурованих мережах. Акцентовано важливість включення штучного інтелекту в системи управління мережним трафіком для забезпечення якості обслуговування в мережах. Розглянуто різні типи методів машинного навчання, включаючи кероване, некероване та навчання з підкріпленням, а також різні алгоритми. Використане для реалізації інтелектуального прийняття рішень для систем управління мережним трафіком було SL навчання.

Зроблено порівняння різних моделей машинного навчання, включаючи лінійну регресію, логістичну регресію, дерева рішень та випадковий ліс, з метою визначення найкращої моделі для поставленої задачі. Також використано матрицю помилок для оцінки ефективності моделі. Це дозволило виявити специфічні помилки класифікації та визначити, які класи модель класифікує найкраще. Точність моделі може бути покращена шляхом збору більшої кількості даних для тренування моделей. Це допоможе моделям краще вивчити шаблони цих класів і, таким чином, покращити їхню точність класифікації. Перевірено працездатність

моделі на валідаційному наборі даних, який не був використаний під час навчання моделі. Це дозволило виявити, наскільки добре модель може узагальнювати своє навчання на нових даних.

Розроблено програмний модуль, який використовує прогнозовані дані навченої моделі для вирішення задачі оптимізації. Результати розробленої моделі машинного навчання застосовано для розв’язання задачі одношляхової маршрутизації. Показано, як вибрані таким чином пристрої можуть бути використані для підвищення рівня мережної безпеки за рахунок реалізації безпечної одношляхової маршрутизації. Розглянуто використання різнотипних метрик для визначення оптимального маршруту в мережі, а саме метрики по аналогії з протоколами RIP, OSPF та на основі ймовірності компрометації для визначення оптимального маршруту.

Окремі результати роботи доповідались на Міжнародних наукових конференціях, а саме [3, 40-42]. Кваліфікаційна робота пов’язана з дослідженнями у межах науково-технічної (експериментальної) розробки 0123U100128 «Розробка алгоритмічно-програмного забезпечення для кіберстійких інфокомунікаційних систем і мереж критичних інфраструктур», що ведеться на кафедрі інфокомунікаційної інженерії імені В.В. Поповського Харківського національного університету радіоелектроніки.

Кваліфікаційна робота демонструє, як можна інтегрувати результати класифікації мережних пристроїв на основі машинного навчання в процеси оптимізації мереж, що дозволяє використовувати передбачувану інформацію про безпечність пристроїв для покращення процесу прийняття маршрутних рішень. Результати, отримані в роботі, пропонуються для реалізації інтелектуальної маршрутизації в програмно-конфігурованих мережах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Майба М.А., Митцева О.С. Використання штучного інтелекту для покращення наукової продуктивності. Наука та освіта в дослідженнях молодих учених [Електронне видання] : матеріали IV Міжнар. наук.-практ. конф. для студ., аспірантів, докторантів, молод. учених, Харків, 167-168 с.

2. Guo Z. Deep Reinforcement Learning-Based Traffic Engineering in SD-WANs. Bringing Machine Learning to Software-Defined Networks. SpringerBriefs in Computer Science. 2022. P. 7–22. DOI: https://doi.org/10.1007/978-981-19-4874-9_2.

3. Cicioğlu, M. and Çalhan, A., 2023. MLaR: machine-learning-assisted centralized link-state routing in software-defined-based wireless networks. Neural Computing and Applications, 35(7), pp. 5409-5420. DOI: https://doi.org/10.1007/s00521-022-07993-w

4. Uppal, S., Woo, S. and Pitt, D., 2015. Software defined WAN for dummies.

5. Hawilo, H., Shami, A., Mirahmadi, M. and Asal, R., 2014. NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC). IEEE network, 28(6), pp.18-26.

6. Yousaf, F.Z., Bredel, M., Schaller, S. and Schneider, F., 2017. NFV and SDN—Key technology enablers for 5G networks. IEEE Journal on Selected Areas in Communications, 35(11), pp.2468-2478.

7. Musumeci, F., Rottondi, C., Nag, A., Macaluso, I., Zibar, D., Ruffini, M. and Tornatore, M., 2018. An overview on application of machine learning techniques in optical networks. IEEE Communications Surveys & Tutorials, 21(2), pp.1383-1408.

8. Sutton, R.S. and Barto, A.G., 1999. Reinforcement learning: An introduction. Robotica, 17(2), pp.229-235.

9. Goodfellow, I., Bengio, Y. and Courville, A., 2016. Deep learning. MIT press.

10. Makridakis, S., 2017. The forthcoming Artificial Intelligence (AI) revolution: Its impact on society and firms. Futures, 90, pp.46-60.

11. Puterman, M.L., 1990. Markov decision processes. Handbooks in operations research and management science, 2, pp.331-434.

12. Bertsekas, D., 2012. Dynamic programming and optimal control: Volume I. Athena scientific.

13. Li, Y., 2017. Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274., pp. 1–85.

14. Arulkumaran, K., Deisenroth, M.P., Brundage, M. and Bharath, A.A., 2017. A brief survey of deep reinforcement learning. arXiv preprint arXiv:1708.05866, pp. 26–38.

15. Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H. and Wang, C., 2018. Machine learning and deep learning methods for cybersecurity. Ieee access, 6, pp.35365-35381.

16. Luong, N.C., Hoang, D.T., Gong, S., Niyato, D., Wang, P., Liang, Y.C. and Kim, D.I., 2019. Applications of deep reinforcement learning in communications and networking: A survey. IEEE Communications Surveys & Tutorials. DOI: 10.1109/COMST.2019.2916583.

17. Barabas, M., Boanea, G. and Dobrota, V., 2011, November. Multipath routing management using neural networks-based traffic prediction. In The Third International Conference on Emerging Network Intelligence, pp. 118–124.

18. Zhang, L. and Thomopoulos, S.C.A., 1989. Neural network implementation of the shortest path algorithm for traffic routing in communication networks. In International 1989 Joint Conference on Neural Networks, p. 591.

19. Barbancho, J., Leon, C., Molina, F.J. and Barbancho, A., 2007. A new QoS routing algorithm based on self-organizing maps for wireless sensor networks. Telecommunication Systems, 36, pp.73-83.

20. Ali, M.K.M. and Kamoun, F., 1993. Neural networks for shortest path computation and routing in computer networks. IEEE transactions on neural networks, 4(6), pp.941-954.

21. Boanea, G., Barabas, M., Rus, A.B., Dobrota, V. and Domingo-Pascual, J., 2011, June. Performance evaluation of a situation aware multipath routing solution. In 2011 RoEduNet International Conference 10th Edition: Networking in Education and Research, pp. 1–6.

22. Liu, F., Liu, B., Sun, C., Liu, M. and Wang, X., 2013. Deep learning approaches for link prediction in social network services. In Neural Information Processing: 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part II 20, pp. 425–432.

23. Gwon, Y.L. and Kung, H.T., 2014. Inferring Origin Flow Patterns in {Wi-Fi} with Deep Learning. In 11th international conference on autonomic computing, pp. 73–83.

24. Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H. and Ng, A.Y., 2011. Multimodal deep learning. In Proceedings of the 28th international conference on machine learning, pp. 689–696.

25. Zhi-yuan, L., Ru-chuan, W. and Jun-lei, B., 2009, August. A multipath routing algorithm based on traffic prediction in wireless mesh networks. In 2009 Fifth International Conference on Natural Computation, pp. 82–90.

26. Chabaa, S., Zeroual, A. and Antari, J., 2010. Identification and prediction of internet traffic using artificial neural networks. Journal of Intelligent Learning Systems and Applications, 2(03), pp. 147–155, Jul. 2010.

27. Goh, H., Thome, N., Cord, M. and Lim, J.H., 2013. Top-down regularization of deep belief networks. Advances in neural information processing systems, pp. 1878–1886.

28. Srivastava, N. and Salakhutdinov, R.R., 2012. Multimodal learning with deep boltzmann machines. Advances in neural information processing systems,, pp. 2222–2230.

29. Salakhutdinov, R. and Hinton, G., 2009, April. Deep boltzmann machines. In Artificial intelligence and statistics, pp. 448–455.

30. Hinton, G. and Salakhutdinov, R., 2012. An efficient learning procedure for deep Boltzmann machines. Neural Comput, 24(8), pp.1967-2006..

31. Moy, J. *OSPF Version 2*, Ascend Commun., Alameda, CA, USA, Apr. 1998.

32. Fadlullah, Z.M., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T. and Mizutani, K., 2017. State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems. IEEE Communications Surveys & Tutorials, 19(4), pp.2432-2455.

33. Kato, N., Fadlullah, Z.M., Mao, B., Tang, F., Akashi, O., Inoue, T. and Mizutani, K., 2016. The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective. IEEE wireless communications, 24(3), pp.146-153.

34. Raniwala, A. and Chiueh, T.C., 2004, August. Evaluation of a wireless enterprise backbone network architecture. In Proceedings. 12th Annual IEEE Symposium on High Performance Interconnects, pp. 98–104.

35. CVE API. NATIONAL VULNERABILITY DATABASE. URL: https://nvd.nist.gov/developers/vulnerabilities.

36. Scikit-learn: machine learning in Python – scikit-learn 1.3.2 documentation. scikit-learn: machine learning in Python – scikit-learn 0.16.1 documentation. URL: https://scikit-learn.org/stable/.

37. Supervised learning. scikit-learn. URL: https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

38. Лемешко О.В., Невзорова О.С., Єременко О.С., Євсєєва О.Ю. Методичні вказівки до практичних занять з дисципліни «Управління та маршрутизація в ТКС» для студентів денної форми навчання спеціальності 6.050903 – Телекомунікації. Харків: ХНУРЕ, 2016. 64 с.

39. GEKKO Optimization Suite – GEKKO 1.0.5 documentation. GEKKO Optimization Suite – GEKKO 1.0.5 documentation. URL: https://gekko.readthedocs.io/en/latest/.

40. Майба М. А. Розробка програмного модуля для підвищення безпеки персональних даних при використанні месенджера. Радіоелектроніка та молодь у XXI столітті : матеріали 27-го Міжнар. молодіж. форуму, 92-93 с.

41. Maiba, M. Development of a Threat Model when Ensuring Information Security in Messengers Based on Privacy and Anonymity. Інформатика, Математика, Автоматика IMA :: 2023: матеріали та програма Міжнародної наукової конференції молодих учених (м. Суми, 24-28 квітня 2023 р.). Суми : СумДУ, 2023. С. 38.

42. Майба, М.А. Аналіз засобів захисту інформації за допомогою стеганографічного перетворення. XV Міжнародна науково-технічна конференція студентів та аспірантів «Перспективи розвитку інформаційно-телекомунікаційних технологій та систем» ПРІТС 2023: Збірник тез конференції. К.: КПІ ім. Ігоря Сікорського, 2023. С. 359.