Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню бакалавра

студента *Дзядека Максима Івановича*

академічної групи *125-20-2*

спеціальності *125 Кібербезпека*

спеціалізації[1]

за освітньо-професійною програмою *Кібербезпека*

на тему *Методи машинного навчання для виявлення аномалій мережевого трафіку інформаційно-комунікаційної системи підприємства*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	проф. Магро В.І.			
розділів:				
спеціальний	ст. викл. Мєшков В.І.			
економічний	к.е.н., доц. Пілова Д.П.	95б	відмінно	
Рецензент				
Нормоконтролер	ст. викл. Мєшков В.І.			

Дніпро
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри
безпеки інформації та телекомунікацій
_____ д.т.н., проф. Корнієнко В.І.

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавра

студенту *Дзядеку Максиму Івановичу* **академічної групи** *125-20-2*
(прізвище ім'я по-батькові) (шифр)

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

на тему ***Методи машинного навчання для виявлення аномалій мережевого трафіку інформаційно-комунікаційної системи підприємства***

затверджену наказом ректора НТУ «Дніпровська політехніка» від 23.05.2024 № 469-с

Розділ	Зміст	Термін виконання
Розділ 1	Аналіз та порівняльна характеристика існуючих методів виявлення аномалій та програмних реалізацій IDS/IPS на ринку	15.03.2024
Розділ 2	Розробка підходу для оцінки найефективнішого алгоритму машинного навчання для задач виявлення аномалій мережевого трафіку інформаційно-комунікаційної системи підприємства	10.05.2024
Розділ 3	Розрахунок капітальних та експлуатаційних витрат на впровадження проєкту, оцінка економічного ефекту та терміну окупності капітальних інвестицій на застосування запропонованого рішення	11.06.2024

Завдання видано _____ Валерій МАГРО
(підпис керівника) (ім'я, прізвище)

Дата видачі: 15.01.2024р.

Дата подання до екзаменаційної комісії: 28.06.2024р.

Прийнято до виконання _____ Максим ДЗЯДЕК
(підпис студента) (ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 102 с., 30 рис., 13 табл., 5 додатків, 34 джерел.

Об’єкт дослідження: методи машинного навчання в системах виявлення аномалій мережевого трафіку для інформаційно-комунікаційної системи підприємства.

Предмет розробки: підхід до виявлення аномалій мережевого трафіку із застосуванням методу машинного навчання.

Мета роботи: дослідження алгоритмів машинного навчання та оцінка їх ефективності для задач виявлення аномалій мережевого трафіку для інформаційно-комунікаційної системи підприємства.

У першому розділі було проаналізовано існуючі методи виявлення аномалій, які застосовуються в системах виявлення аномалій, наведено їх порівняльну характеристику, а також проаналізовано програмні реалізації IDS/IPS на ринку.

У другому розділі було розроблено підхід для оцінки найефективнішого алгоритму машинного навчання для задач виявлення аномалій мережевого трафіку для інформаційно-комунікаційної системи підприємства.

У третьому розділі було проведено розрахунки капітальних та експлуатаційних витрат на впровадження запропонованого рішення, обчислено загальний ефект та термін окупності капітальних інвестицій на впровадження проєкту.

Практична цінність розробки полягає у застосуванні алгоритмів машинного навчання в системах виявлення аномалій для підвищення рівня захищеності інформаційно-комунікаційної системи підприємства.

ЗАГРОЗИ ДЛЯ ІНФОРМАЦІЇ, КІБЕРБЕЗПЕКА, АНОМАЛІЇ МЕРЕЖЕВОГО ТРАФІКУ, СИСТЕМИ ВИЯВЛЕННЯ АНОМАЛІЙ, АЛГОРИТМИ МАШИННОГО НАВЧАННЯ, ОЦІНКА ЕФЕКТИВНОСТІ МОДЕЛІ.

ABSTRACT

Explanatory note: 102 pp., 30 pic., 13 table, 5 app, 34 sources.

Object of research: machine learning methods in network traffic anomaly detection systems for an enterprise information and communication system.

Subject of development: an approach to detecting network traffic anomalies using machine learning.

The aim of the study is to investigate machine learning algorithms and evaluate their effectiveness for detecting network traffic anomalies for an enterprise information and communication system.

The first section analyses the existing anomaly detection methods used in anomaly detection systems, provides their comparative characteristics, and analyses IDS/IPS software implementations on the market.

In the second section, an approach was developed to evaluate the most effective machine learning algorithm for network traffic anomaly detection for an enterprise's information and communication system.

In the third section, we calculated the capital and operating costs for the implementation of the proposed solution, and calculated the overall effect and payback period of the capital investment in the project.

The practical value of the development lies in the application of machine learning algorithms in anomaly detection systems to increase the level of security of the enterprise information and communication system.

THREATS TO INFORMATION, CYBERSECURITY, NETWORK TRAFFIC ANOMALIES, ANOMALY DETECTION SYSTEMS, MACHINE LEARNING ALGORITHMS, MODEL EFFICIENCY EVALUATION.

СПИСОК УМОВНИХ СКОРОЧЕНЬ

AIDS – Application-based Intrusion Detection System;

ANN – Artificial Neural Networks;

СВВ – системи виявлення вторгнень;

DoS – Denial-of-Service Attack;

GPU – Graphics Processing Unit;

HIDS – Host-based Intrusion Detection System;

IDS – Intrusion Detection System;

IPS – Intrusion Prevention System;

KNN – K-Nearest Neighbors;

NIDS – Network-based Intrusion Detection System;

PIDS – Perimeter Intrusion Detection System;

R2L – Remote to Local attack;

SIEM – Security Information and Event Management;

U2R – User to Root attack;

VMIDS – Virtual Machine-based Intrusion Detection System;

АС – автоматизована система;

ІБ – інформаційна безпека;

ІКС – інформаційно-комунікаційна система;

НД – нормативний документ;

НСД – несанкціонований доступ;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – програмний комп'ютер;

СВА – система виявлення аномалій;

ТЗІ – технічний захист інформації.

ЗМІСТ

с.

ВСТУП

Актуальність задачі виявлення аномалій мережевого трафіку на підприємстві зростає з кожним днем. Мотивація на злочинні дії зі сторони шахраїв зростає разом із цінністю інформації, яка обробляється та зберігається в інформаційно-комунікаційних системах (ІКС) підприємства. Отже, виникає потреба у впровадженні дієвого засобу захисту від загроз такого типу, а саме впровадження системи виявлення аномалій (СВА) мережевого трафіку.

Станом на сьогодні існує багато методів, на яких базуються СВА. Серед цих методів є сенс визначити та обрати для реалізації найефективніший метод, який буде демонструвати найвищу результативність. В рамках кваліфікаційної роботи запропоновано впровадження СВА мережевого трафіку для ІКС підприємства на основі методу машинного навчання.

Завдання роботи включають в себе:

1. Дослідити стан питання, проаналізувавши існуючі методи виявлення аномалій та програмних реалізацій IDS/IPS на ринку;

2. Розробити підхід для аналізу та оцінки застосування різних алгоритмів машинного навчання для задач виявлення аномалій мережевого трафіку;

3. За результатами показників ефективності застосування алгоритмів машинного навчання зробити висновок про найефективніший алгоритм в рамках поставленої задачі;

4. Розрахувати капітальні та експлуатаційні витрати на впровадження запропонованого рішення, обчислити загальний ефект і термін окупності капітальних інвестицій;

5. Базуючись на отриманих результатах розрахунку загального ефекту та терміну окупності капітальних інвестицій, зробити висновок про економічну доцільність впровадження запропонованого рішення на умовному підприємстві.

Теоретична значущість роботи полягає в поглибленні знань про застосування методів машинного навчання для задач виявлення аномалій мережевого трафіку. Проведене дослідження дає змогу краще зрозуміти принцип дії існуючих методів

виявлення аномалій мережевого трафіку, і зробити висновок про найефективніший метод для поставленої задачі.

Прикладна цінність отриманих результатів полягає у можливості використання отриманих даних про ефективність моделі на базі певного алгоритму машинного навчання для розробки СВА мережевого трафіку.

Результати кваліфікаційної роботи мають потенціал для подальшого впровадження і використання такої системи в реальних умовах, сприяючи підвищенню рівня захищеності мережі ІКС підприємства.

РОЗДІЛ 1. СТАН ПИТАННЯ. ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність проблеми

У сучасному світі інформаційні системи відіграють все більш важливу роль у житті людей та організацій. Зростання залежності від інформаційних систем, а також збільшення обсягу та цінності інформації, яка обробляється та зберігається в цих системах, робить їх привабливими мішенями для зловмисників. Атаки стають все більш досконалими та складними, що робить виявлення аномалій в мережевому трафіку для запобігання вторгнень актуальною проблемою, яка потребує ефективного рішення.

Згідно з НД ТЗІ 1.1-003-00, атаку визначають як спробу реалізації загрози. Під загрозою мають на увазі будь-які обставини або події, що можуть бути причиною порушення політики безпеки інформації і/або нанесення збитків автоматизованій системі (АС) [1]. На рис. 1.1 наведена класифікація загроз для інформації згідно з НД ТЗІ 1.4-001-00 [2].

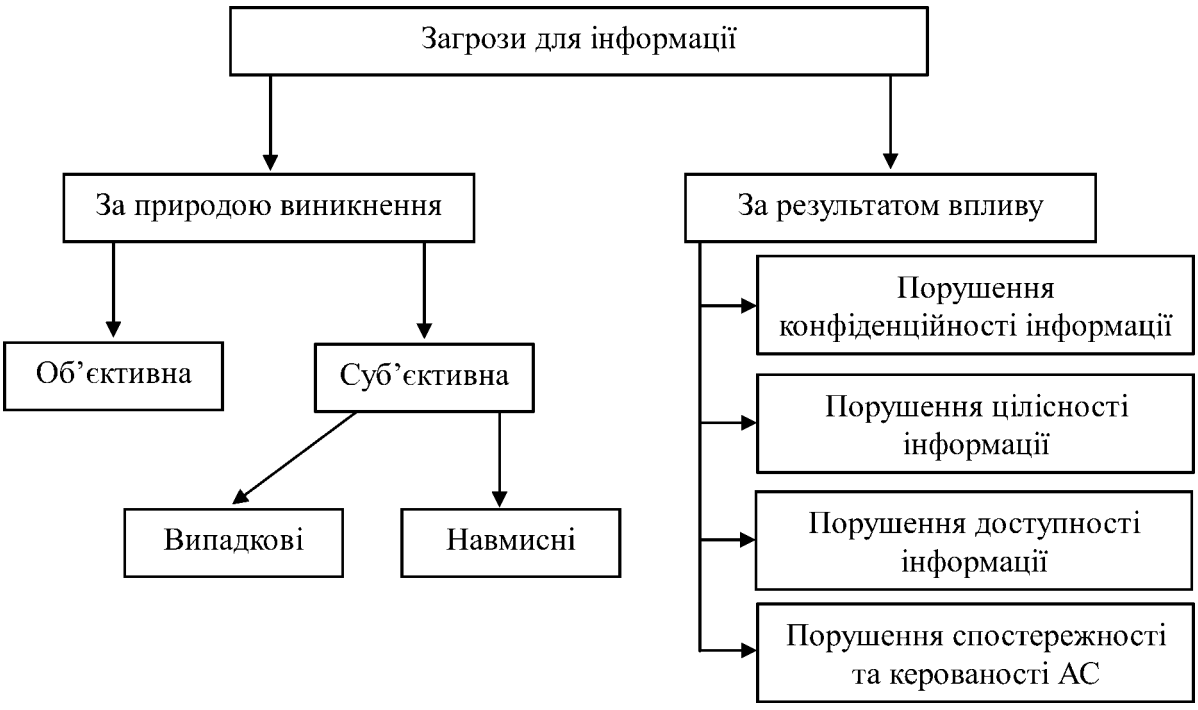

Рисунок 1.1 – Класифікація загроз для інформації

Джерело загрози – це потенційні антропогенні, техногенні або стихійні носії загрози безпеки. Всі джерела загроз безпеці інформації можна розділити на три групи:

1) Обумовлені діями суб'єкта (антропогенні джерела загроз)

2) Обумовлені технічними засобами (техногенні джерела небезпеки)

3) Обумовлені стихійними джерелами

Дивлячись через призму аномалій мережевого трафіку, основні джерела, які розглядаються в рамках цієї загрози, є антропогенні джерела загроз. Загрози безпеці інформації, які обумовленні діями суб'єкта, класифікують на два види: внутрішні та зовнішні [3]. На рис. 1.2 наведені джерела, на які поділяється антропогенне джерело загроз.

Рисунок 1.2 – Зовнішні та внутрішні антропогенні джерела загроз

Внутрішні та зовнішні джерела загроз рівноправно мають місце в причинах

порушення безпеки інформації, тому важливо впровадити універсальний метод моніторингу аномалій мережевого трафіку, який буде однаково ефективний як для зовнішніх, так і внутрішніх джерел загроз.

Наразі існуючі методи та системи виявлення аномалій мережевого трафіку демонструють недостатню ефективність у боротьбі з раніше невідомими загрозами, тому виникає необхідність використання методу, який дозволить ефективніше реагувати на загрози такого типу.

1.2 Огляд існуючих методів ідентифікації аномалій мережевого трафіку

Аномалія мережевого трафіку – це подія або умова в мережі, що характеризується статистичними відхиленням від заданої структури трафіку, отриманої на основі раніше зібраних профілів і контрольних показників [4].

На рис. 1.3 наведено розподіл аномалій в мережі на певні види.

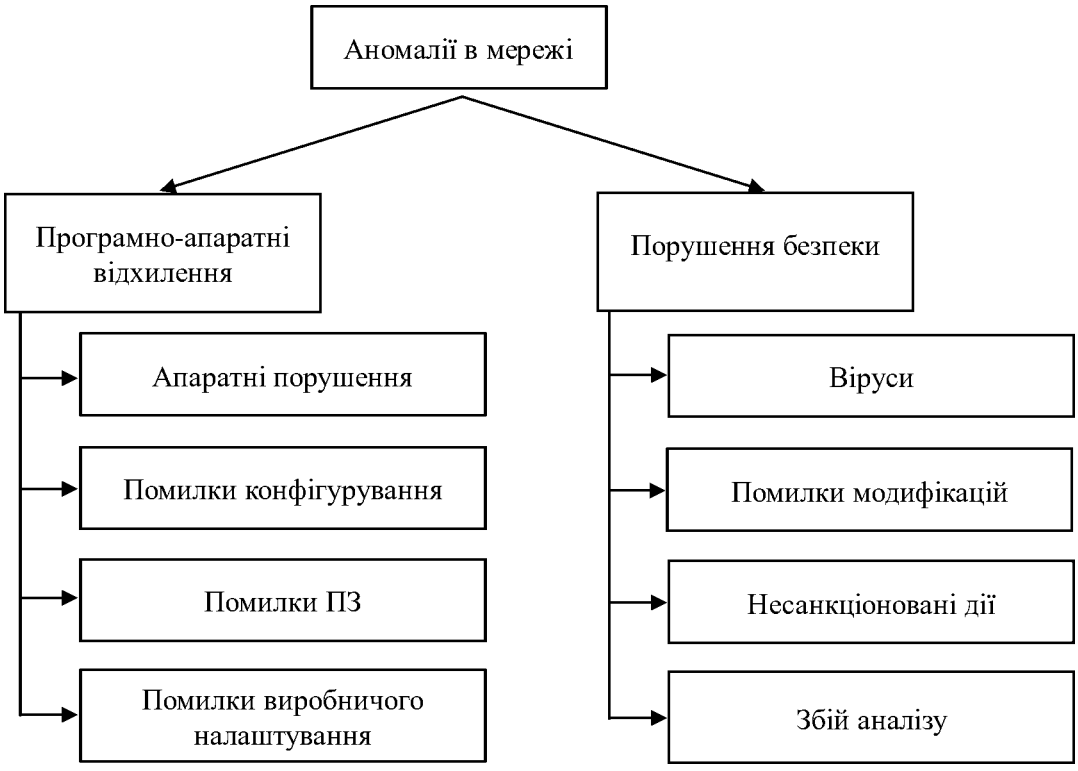

Рисунок 1.3 – Види аномалій в мережі [5]

Спираючись на дані, наведені у [6], аномалії в мережі умовно можна поділити

на три типи: точкові, контекстуальні (умовні), та колективні. Характеристика трьох типів аномалій наведена у таблиці 1.1.

Таблиця 1.1 – Типи аномалій в мережі

Тип аномалії	Характеристика	Приклад
Точкові	Окремий екземпляр даних, який, відносно до решти даних, визначений як аномальний	Велика кількість запитів до сервера з одної IP-адреси
Контекстуальні (умовні)	Окремий екземпляр даних, який був визначений як аномальний у певному контексті	Надмірна активність користувачів з певної групи
Колективні	Група даних, яка визначена аномальною по відношенню до цілого набору даних	Несподівані відхилення від звичайних патернів трафіку, зміни у шаблонах комунікації

Умовна класифікація методів виявлення аномалій представлена на рис. 1.4.

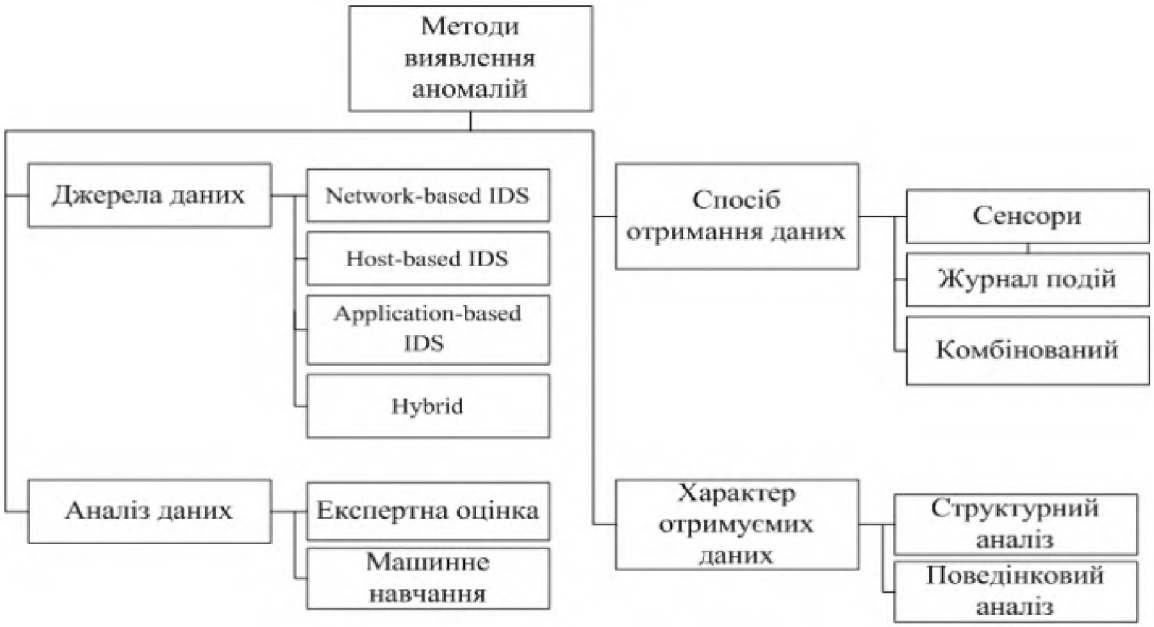

Рисунок 1.4 – Класифікація методів виявлення аномалій [7]

Критерій «Джерела даних» визначає рівень абстракції аналізованих подій в системах, що захищаються [7]. В рамках даного критерію розглядається наступні чотири рівні: рівень мережі, рівень хоста, рівень додатків та гібридний рівень. На практиці частіше використовують системи рівня мережі або рівня хоста. Після

огляду всіх критеріїв буде зроблено детальний огляд наведених вище систем.

Наступним критерієм є аналіз даних. Цей критерій визначає, яким самим способом буде виконуватися аналіз отриманих даних.

Експертна оцінка базується на основі думок експертів, і в подальшому будується адекватна модель майбутнього розвитку об'єкта прогнозування [7].

Машинне навчання навчається на вибірках, і його перевага в тому, що після навчання модель не просто вивчає приклади з вибірки для навчання, а має змогу розпізнавати закономірності і робити висновки. Саме цей метод є досліджуваним об'єктом, який буде використовуватися в рамках виконання кваліфікаційної роботи.

Критерій «Спосіб отримання даних» визначає методи та засоби, за допомогою яких отримуються дані для подальшого аналізу [7].

Сенсори являють собою пристрої, які переглядають потік даних, і створюють звіти про атаки.

Журнал подій є стандартним способом ОС для запису та зберігання інформації про важливі події безпеки.

Критерій «Характер отриманих даних» характеризує дані за виглядом. Структурний аналіз полягає в дослідженні елементів кожного об'єкта та їхніх зв'язків. Мета структурного аналізу – визначення певної послідовності елементів, як цілісний об'єкт, елементи і частини якої співвіднесені й пов'язані строгими зв'язками [7].

У свою чергу, аналіз поведінки ґрунтується на перехопленні всіх важливих системних функцій або установці міні-фільтрів, що дозволяють відстежувати всю активність там, де розгорнута система. Технологія поведінкового аналізу дозволяє оцінювати ланцюжок дій, а не лише одну дію, що підвищує ефективність протидії загрозам. Цей метод реалізований в HIDS.

Станом на зараз одним із найбільш поширених засобів, які використовуються для виявлення аномалій, є системи виявлення аномалій (СВА) (англ. Intrusion Detection System, IDS) – це програмні або апаратні системи, які автоматизують процес моніторингу та ідентифікації подій, що відбуваються в комп'ютерній системі або мережі, і аналізують їх на наявність загроз безпеки [8]. Інформацію для

аналізу подій система виявлення вторгнень отримує із різних областей всередині мережі. Процес виявлення вторгнення працює на припущенні, що дії, які виконуються зловмисником для здійснення атаки, мають суттєві відмінності від звичайної активності користувача системи, і таким чином з'являється можливість ідентифікувати аномалію.

IDS поділяються на чотири основні види: Network-based IDS (NIDS), Host-based IDS (HIDS), Application-based IDS (AIDS) та гібридні IDS. Також існують інші різновиди IDS, але вони менш популярні: Perimeter IDS (PIDS), Virtual Machine-based IDS (VMIDS). Ці види IDS відрізняються місцем встановлення. Наприклад, NIDS працює на рівні мережі, а HIDS – тільки на рівні окремо взятого хоста. Як було зазначено вище, види IDS, які наразі частіше застосовуються на практиці, є NIDS та HIDS. Розглянемо детальніше обидва види систем.

NIDS встановлюється в стратегічно важливих місцях мережі і аналізує весь вхідний та вихідний трафік всіх пристроїв в мережі. Ця система глибоко аналізує трафік, розкладаючи кожен пакет на рівні від канального до рівня додатків. Суттєва відмінність NIDS від міжмережевого екрану в тому, що NIDS здатна виявити внутрішню загрозу в мережі, у той час коли міжмережевий екран аналізує атаки, які виконуються ззовні мережі.

HIDS є альтернативою NIDS. Така система встановлюється на окремий хост всередині мережі, і контролює виключно його захист. HIDS працює за тим самим принципом, що і NIDS. Частіше використовується для встановлення на критично важливі машини в мережі, які рідко змінюють конфігурацію. В таблиці 1.2 наведено переваги та недоліки для обох систем.

Таблиця 1.2 – Переваги та недоліки NIDS та HIDS

Вид IDS	Переваги	Недоліки
NIDS	– декілька систем при оптимальному встановленні можуть прослуховувати велику мережу – розгортання не дає високого впливу на продуктивність мережі через те, що NIDS зазвичай є пасивними	– важко обробляти всі пакети в великій мережі, а отже можуть пропустити розпізнавання атаки, яка почалася під час великих обсягів трафіку – багато переваг NIDS незастосовні до більш сучасних мереж, заснованих на

Кінець таблиці 1.2

Вид IDS	Переваги	Недоліки
	пристроями, які прослуховують мережевий канал без впливу на нормальне функціонування мережі. Це є перевагою, якщо виникає необхідність модифікувати топологію мережі для розміщення системи. – можуть бути зроблені практично невразливими для атак або абсолютно невидимими для атакуючих [7]	мережевих комутаторах через те, що комутатор фрагментує мережу. Більшість комутаторів не надають універсального моніторингу портів – не можуть аналізувати зашифровану інформацію – більшість NIDS не можуть сказати, чи була атака успішною, вони можуть повідомити лише про початок атаки [7]
HIDS	– можливість стежити за подіями локально щодо хоста, що дає можливість визначити атаки, які не визначає NIDS – можуть функціонувати в оточенні, в якому мережевий трафік зашифрований – на функціонування системи не впливає наявність комутаторів – працюють з результатами аудиту ОС, маючи через це перевагу у визначенні цілісності [7]	– більш складні в управлінні через те, що інформація повинна бути налаштована і управлятися для кожного відкритого хоста – через те, що система розгорнута на тому ж хості, який теоретично є метою атаки, то, як складова частина атаки, HIDS може бути ідентифікована і заборонена – унеможливлення виявлення атаки, коли метою є вся мережа через те, що система спостерігає за мережевими пакетами, які отримує хост, на якому розгорнута HIDS – можуть бути блоковані окремими DoS-атаками – через використання результатів аудиту ОС в якості джерела інформації, обсяги інформації можуть бути великі, що потребує додаткового дискового простору для зберігання в системі – використовують обчислювальні ресурси хоста, за яким спостерігає, що впливає на продуктивність системи, що захищається [7]

Отже, з таблиці 1.2 можна зробити декілька висновків: з одного боку, NIDS забезпечують можливість моніторингу великих мереж і можуть бути ефективними в розпізнаванні атак, залишаючись при цьому невразливими для них. Проте, вони мають обмеження у розпізнаванні атак в умовах великих обсягів трафіку та в сучасних мережах, де топологія мережі включає в себе мережеві комутатори. Також, вони не можуть ефективно працювати з зашифрованою інформацією та не

повідомляють про успішну атаку, а лише про її початок.

З іншого боку, HIDS дозволяють стежити за подіями на конкретних хостах і є ефективними у виявленні атак, які в свою чергу не розпізнаються NIDS, однак вони потребують більш складного управління, та є обмеженими у виявленні атак, які спрямовані на всю мережу. Крім того, під час атаки вони можуть бути ідентифіковані та заблоковані, та потребують значних обчислювальних ресурсів, адже використання системи впливає на продуктивність хоста.

Отже, вибір між NIDS та HIDS залежить від конкретних вимог та особливостей мережі, де вони будуть використовуватися.

1.3 Огляд та аналіз існуючих програмних реалізацій IDS/IPS на ринку

Наразі найвідоміші СВВ, які реалізовані програмно, є Snort, Suricata, McAfee Network Security Platform, Zeek (Bro) та Security Onion Розглянемо кожну систему детальніше.

Snort – це ПЗ з відкритим вихідним кодом, яке було створено Мартіном Решем у 1998 році як система виявлення та запобігання вторгнень у мережу. По суті, Snort є представником NIDS, і діє за принципом аналізу та реєстрації мережевих пакетів. Система стежить за мережею відповідно до набору правил, які визначаються адміністратором. Ця система є мультиплатформною, і використовується для таких ОС, як Windows та Linux. Snort вміє вести протоколювання, аналіз та пошук за вмістом, застосовується для активного блокування або пасивного виявлення широкого спектра атак та зондувань, може визначати атаки під прикриттям (шум), коли зловмисник надсилає безліч пакетів на цільову систему, і під час цього шуму проводить атаку [9].

Система працює наступним чином: мережевий пакет, який потрапив у Snort, послідовно проходить через декодери, препроцесори, і потім потрапляє в детектор, який в свою чергу керується раніше заданими правилами. Завдання декодерів – вилучити дані мережевого і транспортного рівня (IP, TCP, UDP) з протоколів канального рівня (Ethernet, 802.11, Token Ring тощо).

Завдання препроцесорів – підготовка вилучених даних до процесу застосування правил. В результаті, перед відправкою в детектор формуються «надпакети», до яких починають застосовуватися правила. Процес виявлення загрози зводиться до пошуку у «надпакеті» визначених у правилі сигнатур [9]. Переваги системи Snort у тому, що вона безкоштовна, і легко розгортається та налаштовується. На рисунку 1.4 продемонстровано інтерфейс користувача Snort в системі PfSense.

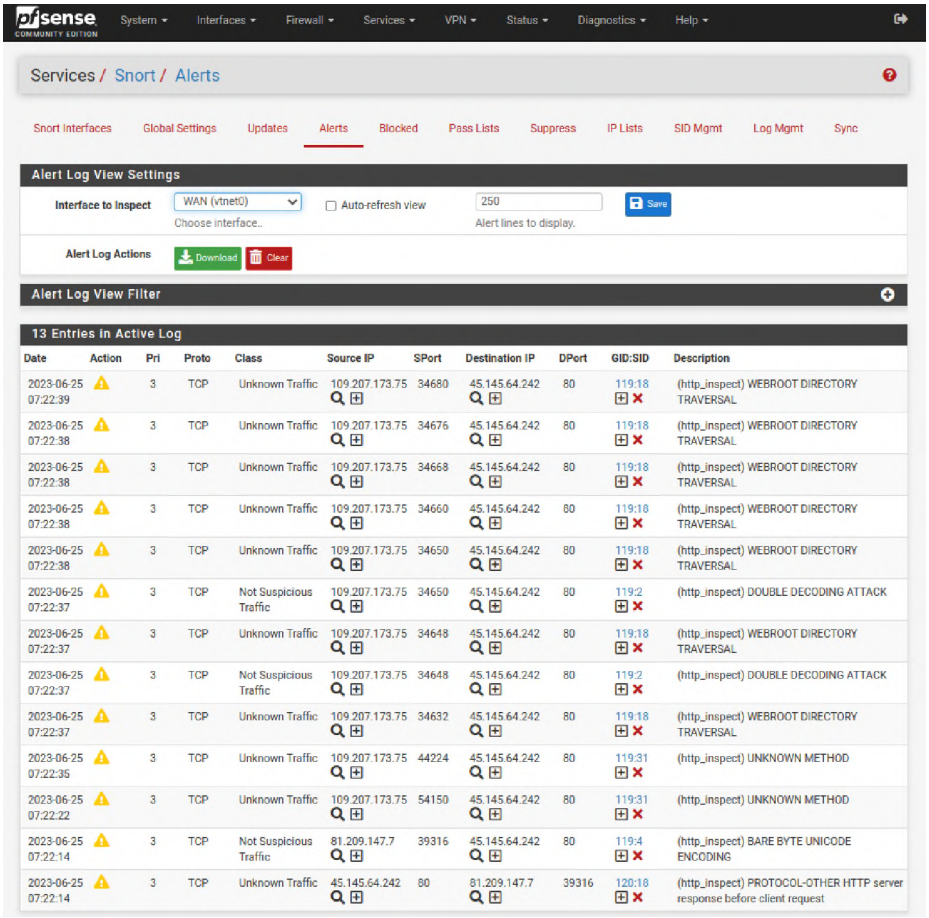

Рисунок 1.4 – Інтерфейс користувача Snort

Suricata – це ПЗ з відкритим вихідним кодом для аналізу мережі та виявлення загроз. Система може використовуватися як IDS, або як IPS (Intrusion Prevention System). Suricata була розроблена OSIF – Фондом відкритої інформаційної безпеки, і є безкоштовною. Вона використовує набір правил і мову підписів для виявлення та запобігання загроз. Система є мультиплатформною, і може використовуватися на

таких ОС, як Windows, Mac та Linux. Завдяки тому, що при розробці системи були використані новітні розробки, вона працює швидше, ніж аналогічні системи конкурентів. Крім того, Suricata сумісна із стандартними утилітами аналізу результатів, і підтримує такі самі модулі, як і Snort [9]. Suricata, як і її конкурент Snort, працює на основі методології підписів, керується правилами, які задані адміністратором. Головною перевагою цієї системи є можливість налаштування набору правил як і в Snort, а також наявність таких функцій, як багатопоточність та прискорення GPU (Graphics Processing Unit). Також, однією з переваг Suricata є зручність інтегрування цієї системи в інше ПЗ (Splunk, SIEM тощо). На рисунку 1.5 можна побачити інтерфейс користувача Suricata.

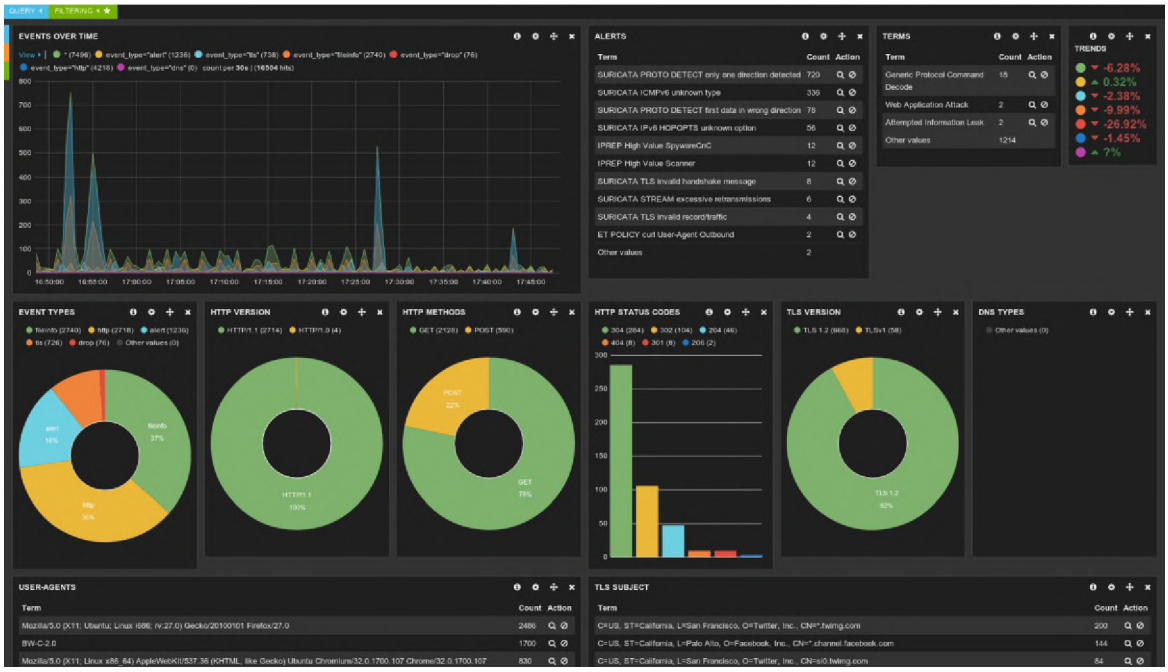

Рисунок 1.5 – Інтерфейс користувача Suricata [10]

McAfee Network Security Platform – це рішення від американської компанії, яка спеціалізується на розробці ПЗ для захисту і аналізу системи, що захищається, від шкідливого ПЗ та інших видів загроз.

Цей продукт підходить для великих компаній, які можуть собі дозволити виділити значний бюджет на захист своєї мережі, тому що ціна програмного продукту стартує від $10 000. Дане рішення блокує величезну кількість загроз,

доступ до шкідливих сайтів, попереджує та запобігає DDoS-атаки тощо. Ця система одночасно використовує два підходи до запобігання загрозам – з використанням сигнатур і без використання сигнатур. Технологія виявлення вторгнень без використання сигнатур базується на виявленні загроз через аномалії, що дозволяє блокувати раніше невідомі атаки, для яких не існує сигнатур. Одна з переваг цієї системи є можливість інформування про загрози у фізичних і віртуальних мережах в режимі реального часу. Інтеграція з McAfee Advanced Threat Defense та McAfee MOVE AntiVirus (один із компонентів рішення McAfee Cloud Workload Security) дає організаціям можливість автоматизувати складні процеси забезпечення безпеки у критично важливому центрі обробки даних [9]. На рисунку 1.6 наведений інтерфейс користувача McAfee Network Security Platform.

Рисунок 1.6 – Інтерфейс користувача McAfee Network Security Platform

Zeek (раніше – Bro) – це програмний аналізатор трафіку с відкритим вихідним кодом. Це безкоштовне ПЗ, яке має вбудовані аналізатори для багатьох мережевих протоколів. Також є можливість створення аналізатору для протоколів, які не підтримуються Zeek «з коробки». Принцип роботи цієї системи є дублювання маршрутизатору в мережі з метою отримання копії трафіку. Після отримання цієї копії, він обробляє її, аналізує, структурує дані, базуючись на протоколах.

Оброблені дані зберігаються в логах. При правильному налаштуванні, Zeek не заважає мережі, не перевантажує групи безпеки непотрібними даними, і є чудовим інструментом для моніторингу та аналізу трафіка. Оброблені дані в логах потім можуть застосовуватися в SIEM-системах (Security Information and Event Management). На рисунку 1.7 наведено візуалізацію даних Zeek в SIEM-системі Elastic Security.

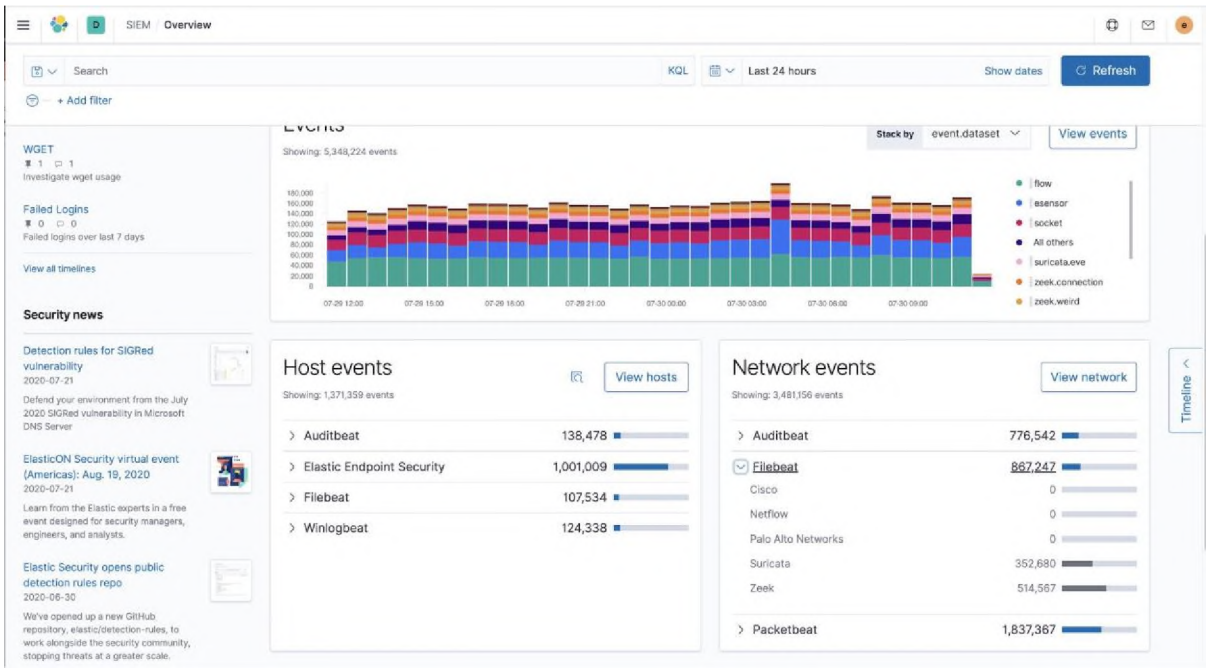

Рисунок 1.7 – Візуалізація даних Zeek в Elastic Security

Security Onion – це ПЗ з відкритим вихідним кодом, яке призначено лише для Linux-систем. Розробником є компанія Security Onion Solutions, США. В цю СВВ було інтегровано інструментарій одразу з трьох програмних реалізації IDS, які були наведені вище – Snort, Zeek та Suricata. Наразі Security Onion підтримується компанією OSSEC. Перевагами цього продукту є безкоштовне ПЗ із відкритим вихідним кодом, широкий спектр інструментів аналізу (Elasticsearch, Logstash, Wazuh, Sguil, Squert, CyberChef, NetworkMiner), висока деталізація, вбудований аналізатор пакетів, параметри відтворення трафіку, використання методів виявлення сигнатур та аномалій. Незважаючи на те, що повторне використання існуючих інструментів означає, що Security Onion отримує переваги від усталеної

репутації своїх компонентів, оновлення елементів у пакеті може бути складним [11]. На рисунку 1.8 наведено інтерфейс користувача Security Onion.

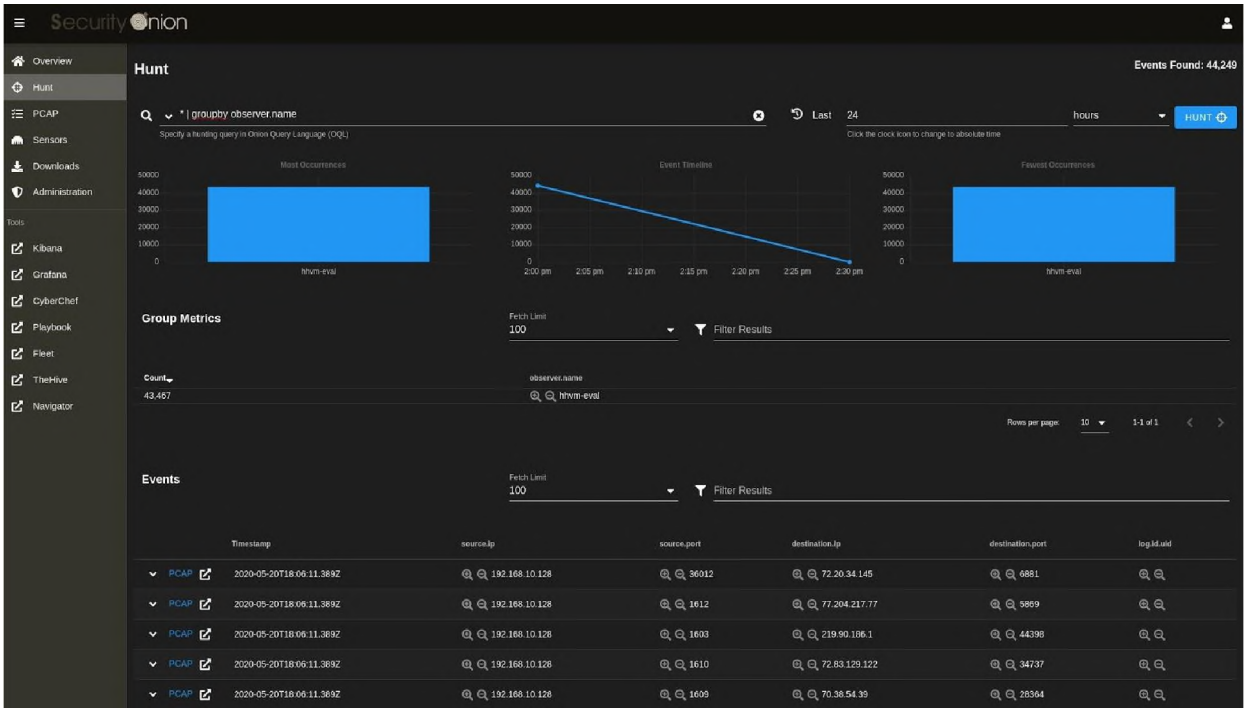

Рисунок 1.8 – Інтерфейс користувача Security Onion

1.4 Порівняльний аналіз програмних реалізацій IDS/IPS на ринку та виявлення їх недоліків

Протягом довгого часу лідером серед програмних IDS на ринку була Snort, але через неможливість повністю адаптуватися під нові умови (поява багатоядерних процесорів, використання IPv6, збільшення обсягів трафіку тощо) перевагу стали надавати іншим системам.

Suricata є альтернативою Snort. Саме вона використовується більшістю приватних та державних організацій, і впроваджується компаніями для захисту своїх активів. Основна відмінність, через яку віддають перевагу Suricata, порівнюючи із Snort – це багатопоточність. Через те, що інструмент може використовувати декілька ядер процесора одночасно, це дозволяє досягнути балансу навантаження та обробляти більше даних. Також однією з причин, чому

при розгортанні IDS віддають перевагу Suricata над Snort, є використання майже всього того, що напрацьовано у Snort. Фактично, у Suricata міститься весь той інструментарій, як і у Snort, ще й свої унікальні напрацювання та дороблення. Також, головною перевагою цієї системи є обробка сьомого рівня OSI (Open System Interconnection) – рівня додатків. Це значно підвищує здатність виявляти шкідливі програми для додатків. Мінус Suricata це складність налаштування системи саме через великий інструментарій. Користувачі скаржаться, що в деяких питаннях Suricata недостатньо виразна в своїй документації, що ускладнює налаштування та розгортання системи.

McAfee Network Security Platform є продуктом із високою вартістю, що є мінусом у порівнянні із іншими системами з відкритим вихідним кодом. Проте якщо бюджет не є проблемою, цей інструмент дозволяє забезпечити захист мережі на високому рівні. Перевагою McAfee Network Security Platform є використання двох методів виявлення атак – за допомогою сигнатур та через виявлення аномалій, що дозволяє опрацьовувати раніше невідомі загрози. Істотним недоліком цієї системи є досить помітне уповільнення роботи мережі через великий функціонал.

Zeek є чудовим інструментом для аналізу трафіку і пошуку загроз, і в основному він використовується як традиційний IDS на основі сигнатур. Через це виникає основний недолік – недостатній рівень захисту мережі від загроз, чия сигнатура була невідома раніше. Також можна виділити такий недолік, як незручність використання інструменту, оскільки Zeek був розроблений з упором на функціональну частину, а не зручність інтерфейсу. Перевагою цієї системи є можливість налаштування Zeek для зосередження аналізу на певних протоколах, що дозволяє проводити глибший аналіз, ніж у інших реалізаціях.

Security Onion містить в собі інструментарій з трьох систем – Snort, Suricata та Zeek. Використання такого широкого набору інструментів дозволяє виконувати більш гнучкий та глибокий аналіз пакетів трафіку. Основним недоліком є те, що Security Onion призначений тільки для Lunix систем, коли в реаліях сьогодення мають місце й інші операційні системи. Також з недоліків можна виділити, що із використанням продукту також виникає необхідність використання Kibana –

плагіну для візуалізації даних. Ця необхідність виникає через недостатньо оптимізований інтерфейс програми для користувача. Звісно, що використання плагіну Kibana у поєднанні з таким широким спектром інструментів, які реалізовані в Security Onion, спричиняє істотне навантаження на ресурси системи, що захищається.

На таблиці 1.3 наведено узагальнені риси усіх систем, наведених в рамках аналізу. В таблиці наведено основні переваги, недоліки та спосіб вирішення недоліків програмного продукту для підвищення продуктивності.

Таблиця 1.3 – Порівняльний аналіз програмних IDS/IPS на ринку [9]

Назва інструменту	Переваги	Недоліки	Спосіб вирішення
Snort	здатність блокування широкого спектру атак при правильному налаштуванні; можливість визначення атак під прикриттям	повільна робота через однопоточність	реалізація багатопоточності
Suricata	багатопоточність; висока продуктивність; спрощене інтегрування із сторонніми програмами; обробка рівня додатків моделі OSI	складність налаштування; недостатньо виразна документація	використання сторонніх плагінів для оптимізації інтерфейсу і налаштування системи
McAfee Network Security Platform	виявлення невідомих раніше атак через використання двох методів виявлення загроз; забезпечення високого рівня захисту	висока вартість системи; уповільнення роботи через великий функціонал	систему доцільно використовувати лише у випадках, коли необхідність захисту такого рівня виправдовує витрати на розгортання системи та незручності, які пов'язані з її використанням
Zeek	підтримка різних режимів роботи; гнучкий функціонал; створювання спеціальних аналізаторів	складність спілкування з інтерфейсом та складність налаштувань	системою повинні користуватись фахівці, що мають великий досвід роботи з подібними системами; zeek

Кінець таблиці 1.3

Назва інструменту	Переваги	Недоліки	Спосіб вирішення
	протоколів; сумісність логів для використання їх в SIEM-системах		призначена для використання в мережах, де необхідна гнучкість та велика кількість налаштувань
Security Onion	висока деталізація на рівні криміналістики; вбудований аналізатор пакетів і параметри відтворення трафіку	доступність використання лише на ос Linux; навантаження на ресурси системи через необхідність використання Kibana	реалізація багатоплатформенності для системи; оптимізація інтерфейсу користувача для більш спрощеного спілкування із системою

Після проведеного аналізу, можна виділити спільні для більшості систем недоліки, а саме:

1) уповільнення роботи системи або навантаження на мережу;

2) складність налаштування;

3) незручність інтерфейсу;

4) складність налаштування системи та правил для недосвідчених користувачів.

1.5 Висновки за розділом

Інформаційні системи стають невід'ємною частиною будь-якої діяльності, і підхід до захисту інформації в цих системах повинен бути серйозним. У рамках розділу в загальному плані було розглянуто види загроз, їх класифікацію, визначено поняття аномалії в мережі, їх види та типізацію. Також було розглянуто методи виявлення аномалій мережевого трафіку за певними критеріями, і наведено коротку характеристику для кожного з них. СВА типів NIDS та HIDS, як системи, які сьогодні частіше застосовуються на практиці, було розглянуто детальніше, наведено характеристику та побудовано таблицю переваг та недоліків для кожної з систем.

Розглянувши існуючі методи виявлення аномалій, а також існуючі і популярні

на сьогодні програмні реалізації систем IDS, у рамках кваліфікаційної роботи буде запропоновано використання СВА на основі методу машинного навчання. Цей метод обраний через значну перевагу над іншими методами в швидкості оброблення даних та отриманні результатів. Також він не потребує постійного додавання та оновлення даних через самонавчання системи, і має високу актуальність через свою високу ступінь точності ідентифікації загроз в динамічних умовах. Така система забезпечує високу швидкість функціонування, що є вкрай важливим фактором при роботі системи в реальному часі. Використання системи, яка базується на методі машинного навчання, дозволить вирішити недоліки, які були визначені для більшості програмно реалізованих IDS (таблиця 1.3).

Система, яка базується на основі машинного навчання не буде істотно навантажувати мережу чи ресурси системи, тому що вона буде спостерігати та реагувати лише на аномальні для мережі події, уникаючи аналізу всіх без виключення пакетів даних. Складність налаштування виключається на етапі розгортання системи, тому що на етапі розробки буде реалізовано навчання моделі на великих вибірках даних, що у майбутньому, під час використання системи в режимі захисту, дозволить мінімізувати участь людини в налаштуванні системи. Така система не потребує реалізації правил для мережі, що дасть змогу використовувати такий продукт навіть недосвідченим користувачам. При цьому, система такого типу буде забезпечувати достатньо високий рівень захисту. Це обумовлено тим, що система, яка базується на методі машинного навчання, реагує як на відомі атаки, так і на раніше невідомі, тому що вона не має прив'язки до сигнатур чи правил, і працює виключно на аналізі та визначені аномальних явищ в мережевому трафіку. Точність спрацювань буде залежати від якості вибірки даних, обраної для навчання моделі, а також від алгоритму машинного навчання.

РОЗДІЛ 2. СПЕЦІАЛЬНА ЧАСТИНА

2.1 Теоретичні відомості про використання машинного навчання в СВА

У першому розділі були розглянуті методи виявлення аномалій, які використовуються в IDS. У рамках другого розділу як покращення існуючих моделей буде запропонована програмна реалізація СВА на основі методу машинного навчання. Проаналізуємо роль машинного навчання в сучасних СВА.

Машинне навчання – один із способів функціонування штучного інтелекту, а саме – практичної реалізації його можливостей завдяки створенню алгоритмів для виявлення закономірностей під час аналізу великих даних, та їх подальшого використання для самонавчання [12]. Головне завдання машинного навчання не є вирішення однієї конкретної задачі, а навчання машини вирішувати інші подібні завдання [13]. Саме машинне навчання є основою сучасних СВА, і завдяки використанню цього методу підвищується точність та ефективність виявлення кібератак.

Наразі існує велика кількість алгоритмів машинного навчання, такі як дерева рішень, ансамблеві моделі, нейронні мережі тощо, і завдяки ним IDS, які базуються на методі машинного навчання, мають змогу адаптуватися до специфіки мережі та загроз різних видів.

Одна з основних переваг використання IDS для виявлення аномалій мережевого трафіку ІКС підприємства на основі методу машинного навчання є здатність такої системи до самонавчання та адаптації. Саме ці показники забезпечують необхідний рівень гнучкості на тлі кіберзагроз, які постійно змінюються та вдосконалюються. Також, використання машинного навчання у IDS значно зменшує кількість хибних спрацювань системи. Це досягається шляхом ретельного навчання моделі на великих та різноманітних вибірках даних.

На рисунку 2.1 показано, які типи машинного навчання існують.

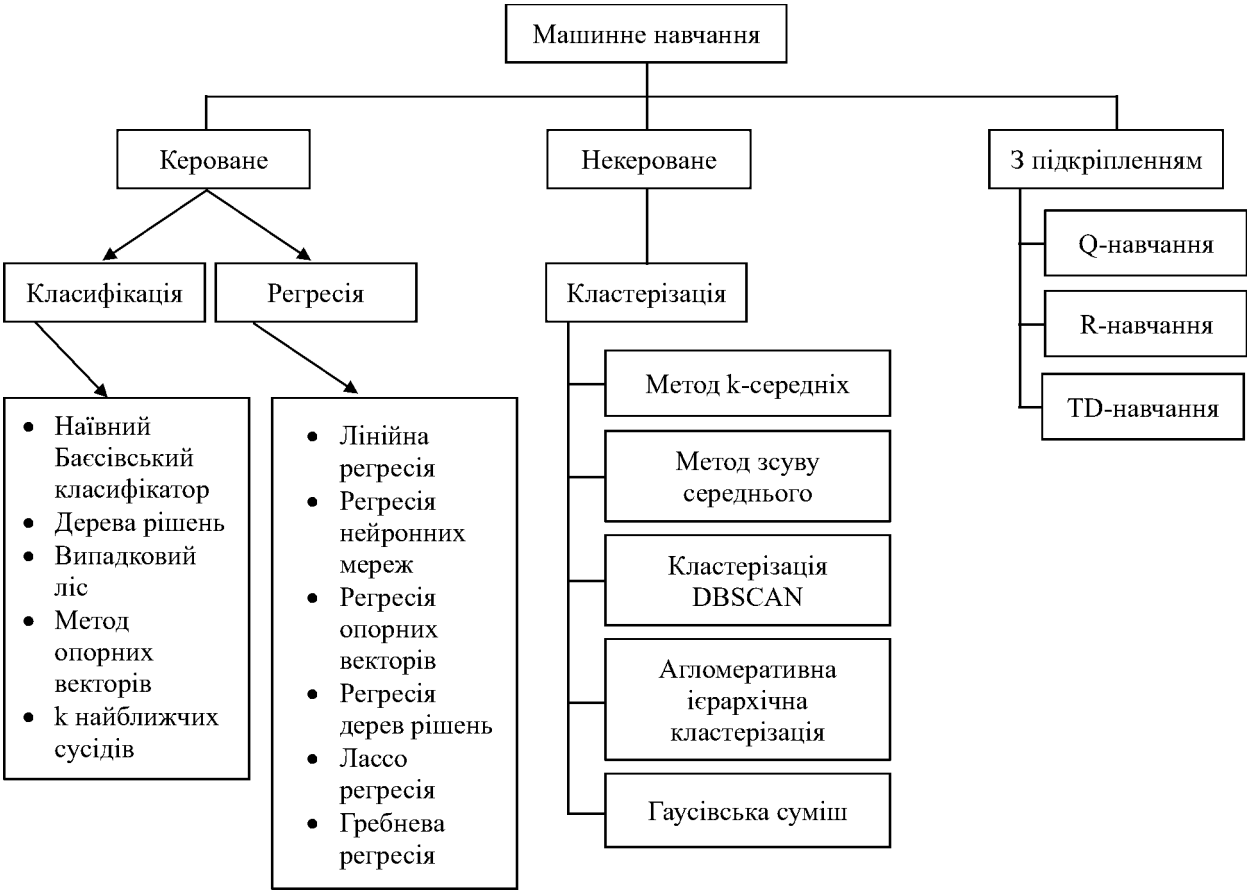

Рисунок 2.1 – Основні типи машинного навчання

Використання керованого машинного навчання в IDS надає широкий спектр інструментів для аналізу і класифікації мережевих даних. При керованому навчанні модель використовує заздалегідь підготовлену вибірку даних із мітками, які позначають нормальні дані, або дані із слідами вторгнення [14]. Дуже важливо використовувати для навчання добре підготовлені дані, які дозволяють моделі явно виділяти зловмисну та безпечну поведінку. Саме вірно підібраний набір даних відіграє вирішальну роль в точності та коректності спрацювання системи.

Некероване машинне навчання використовується з немаркованими даними у вибірці. При використанні цього типу машинного навчання модель вдосконалюється сама по собі без втручання людини, власноруч виявляючи закономірності в наданому наборі даних. Використання такого типу машинного навчання має місце при аналізу великих даних.

На рисунку 2.2 наведено три категорії некерованого машинного навчання.

Рисунок 2.2 – Категорії некерованого машинного навчання

Асоціація – метод машинного навчання, заснований на правилах та закономірностях. Має місце в IDS, оскільки модель, яка була навчена за допомогою такого методу, має можливість слідкувати за ланцюгом зловмисних дій, виявляючи при цьому атаки в різних фазах.

Зниження розмірності – процес скорочення кількості випадкових змінних шляхом отримання множини головних змінних [15]. Ця категорія машинного навчання дає змогу дещо спростити вибірку даних, яка подана для навчання моделі. Через призму СВВ, зниження розмірності полегшує процес аналізу трафіку та виявлення аномалій через виділення основних характеристик та показників трафіку.

Кластерізація – метод машинного навчання, який групує вибірку по точкам даних за спільними властивостями. В контексті використання цього методу в IDS, кластерізація може бути застосована для виявлення патернів трафіку, який є аномальним для мережі, що захищається.

Отже, некероване машинне навчання, завдяки своїм можливостям для аналізу вибірки даних, дає змогу виявити аномалії та нетипові патерни мережевого трафіку, які можуть залишитися невиявленими при використанні інших підходів до машинного навчання, але його недолік в тому, що цей метод потребує використання великої кількості даних, а отже й значних обчислювальних потужностей та витрат часу. Також, у порівнянні з іншими методами, некероване машинне навчання більш

схильне до отримання неточних результатів, що в свою чергу тягне за собою необхідність втручання людини для перевірки даних.

Машинне навчання з підкріпленням також має місце в СВВ. Цей тип дещо відрізняється від керованого та некерованого машинного навчання. Сутність цього принципу полягає в тому, що модель виконує дії в певному середовищі, і в процесі виконання цих дій самонавчається. Такий тип машинного навчання використовується для розробки IDS, яка буде ефективно виконувати свої задачі в умовах змінюваних патернів атак та змінюваної інфраструктури мережі.

В основі цього типу машинного навчання лежить дві мети – мінімізація помилок та отримання максимального результату від виконання завдання. В контексті IDS, максимальний результат являє собою швидкість виявлення вторгнення, коректність ідентифікації загрози та правильність спрацювання.

В реальних умовах застосування такого принципу дуже обмежене, тому що модель, яка навчається за таким методом, вимагає значного обсягу взаємодії із певними середовищами для підвищення ефективності навчання. Забезпечити це в достатніх обсягах важко, тому інженери частіше віддають перевагу іншим типам.

На таблиці 2.1 наведено узагальнення переваг та недоліків для кожного з розглянутих методів машинного навчання.

Таблиця 2.1 – Переваги та недоліки методів машинного навчання

Назва методу	Переваги	Недоліки
Кероване машинне навчання	Демонструє високу точність для відомих патернів	Потребує попередньо підготовлені вибірки даних великого обсягу; демонструє меншу ефективність для раніше невідомих патернів
Некероване машинне навчання	Має можливість виявлення аномалій без попереднього маркування даних	Точність навчання прямо пропорційно залежить від якості вибірки даних, яка подана для навчання; вимагає значні обчислювальні потужності
Навчання із підкріпленням	Легко адаптується до змін середовища, в якому виконує задачі захисту, виявляє достатньо високу точність до невідомих патернів	Вимагає значного обсягу взаємодії із певними середовищами для підвищення ефективності

Кінець таблиці 2.1

Назва методу	Переваги	Недоліки
		навчання, що важко забезпечити в реальних умовах

2.2 Огляд алгоритмів машинного навчання, які використовуються в СВА

Сьогодні існує велика кількість алгоритмів машинного навчання, кожен з яких знаходить застосування в залежності від задачі. На рисунку 2.3 продемонстровано концепцію роботи СВА, яка базується на методі машинного навчання.

Рисунок 2.3 – Концепція роботи СВА, яка базується на методі машинного навчання

Розглянемо детальніше алгоритми машинного навчання, які використовуються для машинного навчання в СВА.

Метод KNN (K-Nearest Neighbors) – алгоритм класифікації та регресії, який базується на гіпотезі компактності, яка припускає, що об’єкти, які розташовані близько один до одного в множині ознак мають схожі значення цільової змінної або належать до одного класу. Алгоритм KNN працює таким чином: спочатку

вираховується відстань між тестовим та навчальним зразком, далі обирається k-найближчий зразок (сусід), де значення k відомо завчасно. Підсумковим прогнозом буде мода у випадку класифікації і середнє арифметичне у випадку регресії. Ці етапи повторюються для кожного з тестових зразків. Схема роботи алгоритму KNN наведена на рисунку 2.4.

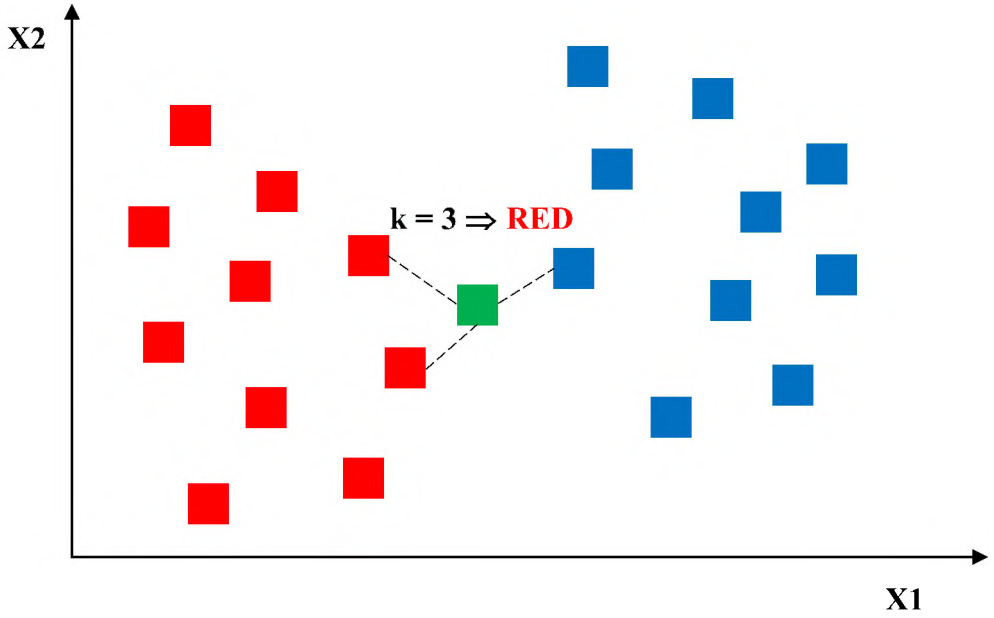

Рисунок 2.4 – Схема роботи алгоритму KNN

Алгоритм «дерева рішень» (Decision trees) є представником керованого методу машинного навчання. Він класифікує і обчислює регресію даних, використовуючи істині чи хибні відповіді на певні питання. Ця модель складається с таких компонентів, як «листя», «гілки» та вузли прийняття рішень. Вузол виконує ідентифікацію атрибуту, від якого в свою чергу залежить цільова функція. Гілка вказує на наступний вузол, базуючись на значенні атрибуту. Листок є вказівкою на клас до якого належить запис [16]. На рисунку 2.5 наведено абстрактну схему роботи алгоритму «дерева рішень».

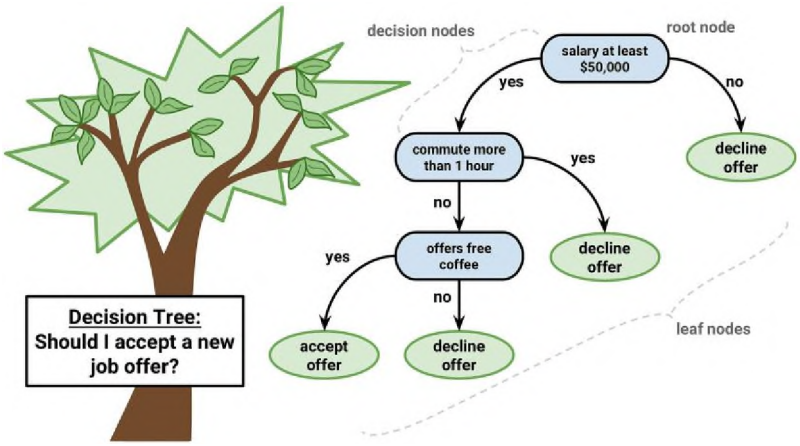

Рисунок 2.5 – Схема роботи алгоритму Decision Trees [17]

Наївний Баєсівський класифікатор (Naive Bayes) – один з найпростіших класифікаторів машинного навчання, який базується на теоремі Баєса. Цей алгоритм припускає, що всі змінні в наборі даних «наївні», тобто не корелюють один з одним. Такий класифікатор обчислює можливість приналежності об'єкта до якогось класу. Ця ймовірність обчислюється з шансу, що якась подія відбудеться, з опорою на події, що вже відбулися. Кожен параметр об'єкта, що класифікується, вважається незалежним від інших параметрів [18]. Такий алгоритм має застосування в СВВ завдяки легкості його використання при досить великій точності [19]. На рисунку 2.6 представлено схему принципу дії алгоритму Naïve Bayes.

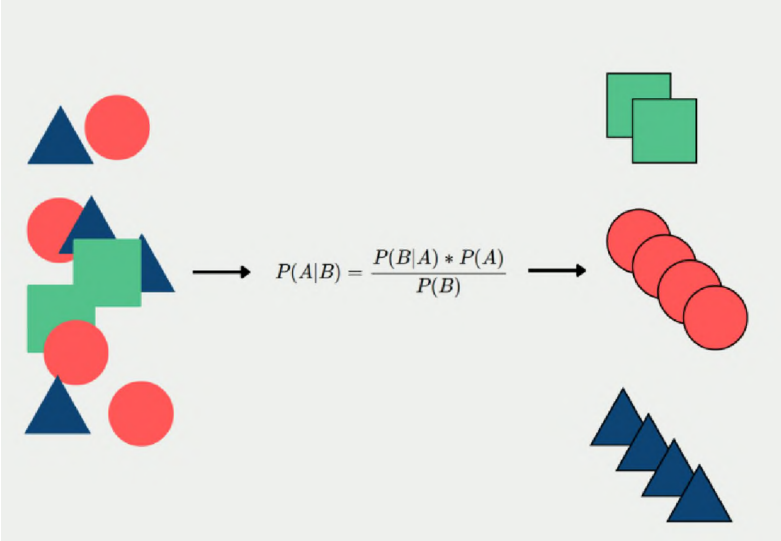

Рисунок 2.6 – Принцип дії алгоритму Naive Bayes [20]

Алгоритм Random Forest (випадковий ліс) – підхід в машинному навчанні, в якому використовуються дерева рішень. В рамках цього методу «ліс» створюється шляхом об'єднання великої кількості різних структур рішень, які формуються різними методами [21]. Випадковий ліс генерує N-дерев рішень, базуючись на навчальній вибірці. В процесі виконання, для кожного дерева випадковим чином виконується повторна вибір навчального набору даних. Таким чином створюється N-дерев рішень, кожне з яких відрізняється один від одного. Врешті решт проводиться голосування шляхом вибору нових оцінок з оцінок, які були виставлені N-деревами. Значення з найвищим рейтингом визначаються як кінцеві значення [21]. На рисунку 2.7 наведено спрощене представлення принципу дії алгоритму Random Forest.

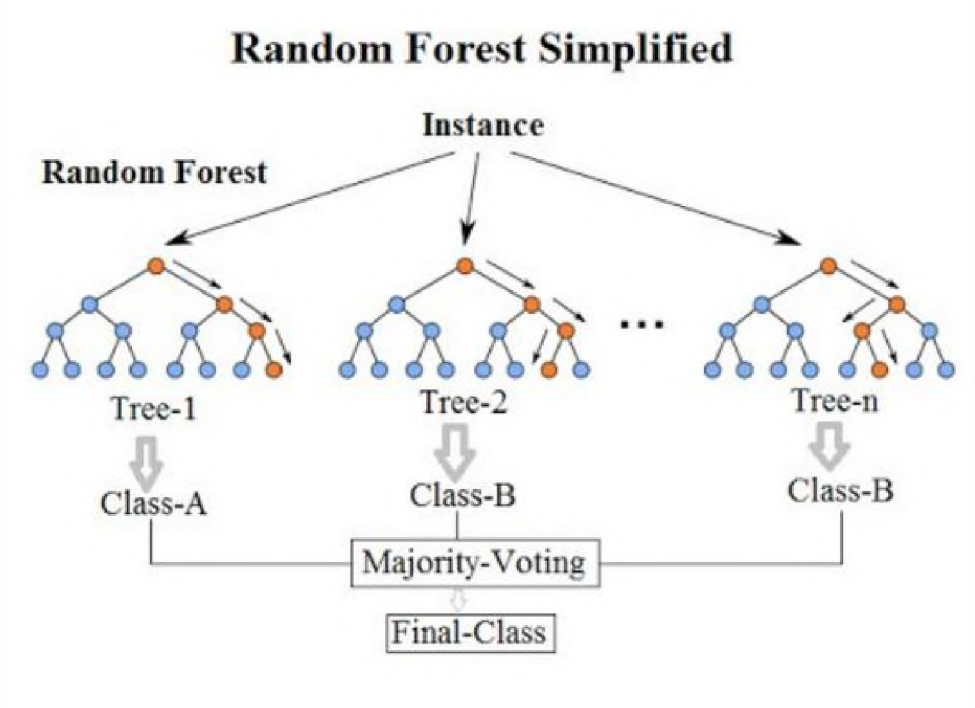

Рисунок 2.7 – Принцип дії алгоритму Random Forest [22]

Adaptive Boost (скорочено – AdaBoost) – алгоритм машинного навчання, розроблений для підвищення ефективності класифікації. Принцип дії алгоритму можна пояснити наступним чином: дані діляться на групи за приблизними правилами, і кожний раз при запуску алгоритму к цим правилам додаються нові.

Таким чином, створюється множина слабих та малоефективних правил, які мають назву «базові правила» [23]. Після багатократного відпрацювання алгоритму, базові правила об'єднуються в одне, яке є сильнішим. Під час цього процесу алгоритм привласнює ваговий коефіцієнт кожному слабкому правилу. Найбільше значення коефіцієнта привласнюється найменшій похибці. Ці вагові коефіцієнти використовуються при виборі кінцевих правил. Кінцеве правило створюється, надаючи перевагу слабким правилам з високим значенням вагового коефіцієнту [23]. На рисунку 2.8 наведено принцип дії алгоритму AdaBoost.

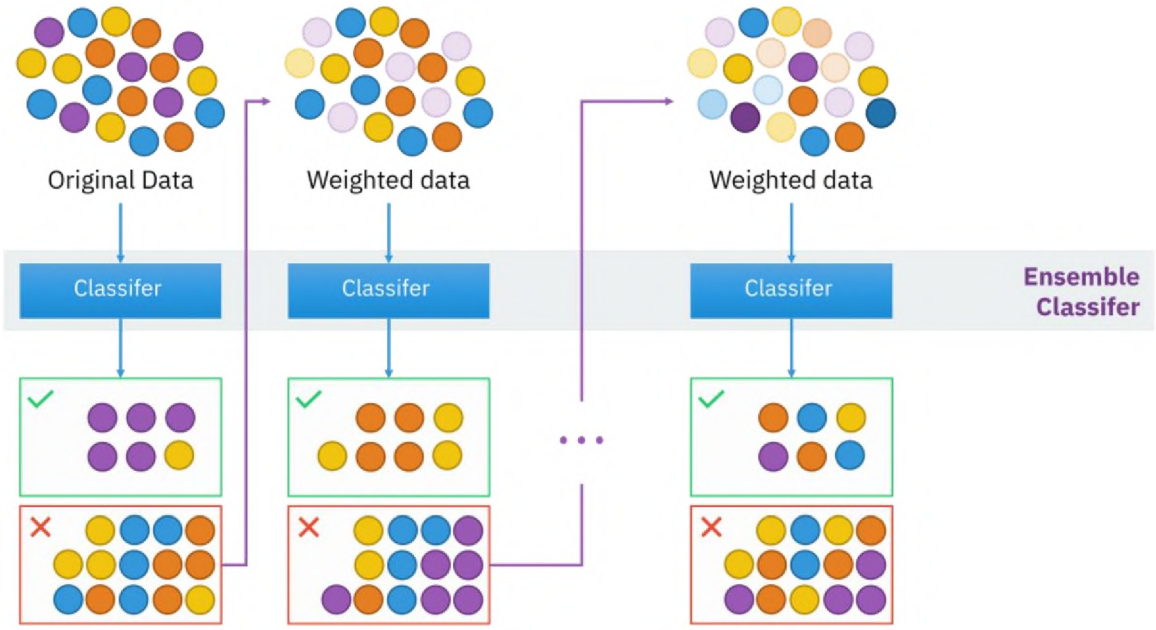

Рисунок 2.8 – Принцип дії алгоритму AdaBoost [24]

Алгоритм MLP (Multi-Layer Perceptron) – це тип штучних нейронних мереж. Штучні нейронні мережі (ANN) – це алгоритм машинного навчання, в основі якого лежить принцип дії мозку людини. Мета цього методу – імітувати властивості людського мозку, такі як навчання, прийняття рішень, отримання нової інформації. В той час, коли людський мозок складається з взаємопов'язаних клітин, які називаються нейронами, ANN складаються з взаємопов'язаних ієрархічних штучних клітин [25]. MLP складається з трьох етапів: вхідний шар, прихований шар та вихідний шар. Вхідний шар – це етап, який відповідає за отримання даних,

обробка інформації в рамках цього шару не виконується. Отримана інформація передається на наступний, прихований шар. Кожен нейрон на цьому етапі пов'язаний з нейронами в прихованому шарі [25]. В прихованому шарі дані, які були отримані від вхідного шару, обробляються і передаються на наступний шар – вихідний. В алгоритмі кількість прихованих шарів чи кількість штучних нейронів в цьому шарі може змінюватися. Зміна кількості шарів та числа нейронів прямо впливає на продуктивність алгоритму. Збільшуючи кількість шарів та нейронів, MLP може вирішувати більш складні задачі, однак це також призведе до збільшення часу, який необхідний для роботи алгоритму [25]. У вихідному шарі, який є останнім, кожна комірка пов'язана із всіма комірками прихованого шару, і результати обробки даних в прихованому шарі подаються на цьому етапі. На рисунку 2.9 продемонстровано схематичну діаграму трирівневого MPL-алгоритму.

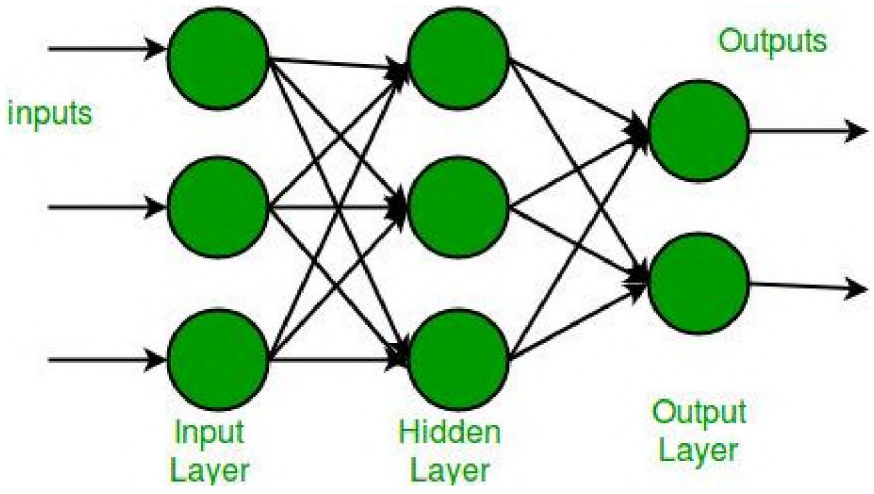

Рисунок 2.9 – Схема трирівневого MLP-алгоритму [26]

QDA (Quadratic Discriminant Analysis) – це метод дискримінантного аналізу. Дискримінантний аналіз являє собою статистичний метод відношення виміряних даних до однієї групи серед багатьох інших груп. При такому відношенні досліджувані дані повинні бути віднесені до тієї групи, якій вони належать. Якщо вони віднесені до групи, до якої вони не належать, виникає помилка. Ця помилка відома як «коефіцієнт помилок». Мета цього алгоритму полягає в тому, щоб виконати процес розподілу із мінімальним коефіцієнтом помилок. Якщо

припустити, що набір даних має нормальний розподіл і дисперсії рівні, аналіз буде мати назву лінійний дискримінантний аналіз. Але якщо припущення хибне, тоді аналіз має назву квадратичний дискримінантний аналіз. Для того щоб можна було застосувати квадратичний дискримінантний аналіз, кількість зразків повинна бути більшою, ніж кількість груп.

В таблиці 2.2 наведено узагальнення алгоритмів машинного навчання, які були оглянуті, за їх перевагами та недоліками.

Таблиця 2.2 – Переваги та недоліки певних алгоритмів машинного навчання

Назва алгоритму	Переваги	Недоліки
K-Nearest Neighbors	Проста реалізація; досить гарна продуктивність при роботі з багатомірними даними; швидкий на етапі навчання; можна адаптувати під необхідне завдання вибором метрики чи ядра [18]	Відносно повільний на етапі оцінки
Decision trees	Інтерпретованість моделі; дерева рішень можуть легко візуалізуватися як сама модель (дерево), так і прогноз для окремого взятого тестового об'єкта (шлях у дереві); швидкі процеси навчання та прогнозування; мінімальна кількість параметрів моделі; підтримка і числових та категоріальних ознак [18]	Дерева дуже чутливі до шумів у вхідних даних; роздільна межа, побудована деревом рішень, має свої обмеження і на практиці дерево рішень за якістю класифікації поступається деяким іншим методам [18]
Naive Bayes	Легко і швидко передбачає клас тестового набору даних; добре справляється із багатокласовим прогнозуванням; добре працює з категоріальними ознаками (порівняно з числовими), продуктивність наївного байєсовського класифікатора краща, ніж в інших простих методах, більш того, йому потрібно менше навчальних даних [18]	Обмеженням даного алгоритму є припущення незалежності ознак, однак у реальних завданнях цілком незалежні ознаки трапляються вкрай рідко; якщо змінна має категорію (в тестовому наборі даних), яка не спостерігалася в навчальному наборі даних, то модель надасть їй нульову ймовірність і не зможе зробити прогноз [18]

Кінець таблиці 2.2

Назва алгоритму	Переваги	Недоліки
Random Forest	Алгоритм добре працює з великими датасетами; застосовний для вирішення багатьох задач машинного навчання; підходить для роботи із відсутніми значеннями у вибірці (на відміну від Naive Bayes), замінюючи відсутні дані на власні	Структура алгоритму є досить важкою, оскільки складається з багатьох дерев рішень; функціонування алгоритму досить важке для розуміння
AdaBoost	Відсутня необхідність перетворення змінних для застосування алгоритму; може виконувати операції над великою кількістю слабких правил; рідко виникає проблема перебору; добре підходить для роботи із відсутніми значеннями у вибірці	Слабкий до шуму та екстремальних значень у вибірці даних; результативні значення мають не найбільш високі значення у порівнянні з іншими алгоритмами
Multi-Layer Perceptron	Добре впорається із важкими задачами; може працювати із відсутніми у вибірці даними; може узагальнювати результати після процесу навчання, таким чином охоплюючи більш широку область застосування як алгоритм машинного навчання	Важко побудувати структуру мережі; користувач повинен сам обирати структуру мережі, яка підходить; важкий для інтерпретації та розуміння
Quadratic Discriminant Analysis	Гнучкість алгоритму через допускання різних коваріаційних матриць для кожного класу; враховує відмінності між класами та всередині класів; більш точний при складних розподілах; ефективність для добре відокремлених класів	Потребує великих обсягів датасету для стабільної оцінки; чутливий до шуму; високий ризик перенавчання у порівнянні з іншими алгоритмами

Всі наведені вище алгоритми машинного навчання будуть використані в рамках кваліфікаційної роботи під час реалізації IDS. У процесі імплементації буде наведено проміжні результати по кожному з алгоритмів і зроблено висновок про алгоритм, який показав найкращі результати для такого роду задач по тестовій вибірці даних.

2.3. Огляд відомих вибірок даних для навчання моделі

Для тренування моделі необхідно використовувати широку вибірку даних для більш точного та коректного навчання. Правильний вибір набору даних у поєднанні з вибором алгоритму, який більш за все підходить для конкретної задачі є запорукою успішного навчання моделі. Дивлячись через призму навчання та тестування СВВ, існують такі широко відомі вибірки даних:

1) KDDcup99 – вибірка даних, яка походить від DARPA'98, і використовується для оцінки СВВ. Цей датасет налічує близько 5 мільйонів зразків. Кожен з них має 41 характеристику і класифікацію як «норма» чи «атака». В рамках цієї вибірки атаки поділяються на 4 типи: DoS (denial-of-service attack), U2R (User to Root), R2L (Remote to Local), зондування (Probe).

2) Kyoto-2006+ – вибірка даних, яка була створена під час збору даних різних заходів безпеки мережі в Кіотському університеті. Базуючись на 41 ознаках датасету KDDcup99, визначено 24-ознакові вибірки (14 статистичних та 10 додаткових). Містить в собі трафік з 2006 по 2015 рік.

3) NSL-KDD – запропонований в 2009 році датасет, в якому виправлено недоліки KDDcup99. Містить 4 категорії атак, включає в себе навчальний та тестовий набори з 21 та 37 атаками різних типів відповідно.

4) UNSW–NB15 – датасет, який був створений в Австралійському центрі кібербезпеки в 2015 році. Містить в собі поєднання звичайних патернів трафіку та атаки. Загалом налічує в собі 9 типів атак.

5) CICIDS2017 – вибірка даних, яка була створена Канадським інститутом кібербезпеки в 2017 році. Містить в собі трафік із звичайною активністю в мережі та атаками.

В таблиці 2.3 наведено узагальнену характеристику вищенаведених датасетів для навчання моделі.

Таблиця 2.3 – Характеристика відомих вибірок даних для навчання моделі

Назва вибірки	Рік випуску	Кількість типів атак	Атаки
KDDcup99	1999	4	Probe, DoS, U2R, L2R
Kyoto-2006+	2006-2015	2	Відомі та невідомі атаки
NSL-KDD	2009	4	Probe, DoS, U2R, L2R
UNSW-NB15	2015	7	Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms
CICIDS2017	2017	7	Brute Force, DoS, HeartBleed, WebAttack, Infiltration, Botnet, DDoS

Отже, оглянувши найвідоміші вибірки даних для навчання та тестування моделі, зроблено деякі висновки. Головний недолік вибірки KDDcup99 є незбалансованість, а також велика кількість надмірних записів, що перетворює тренування моделі на розпізнавання поширених патернів і пропускання рідкісних, аномальних зразків. Крім того, через великий об'єм навчальної вибірки дослідники зазвичай розділяють її на частини і використовують їх для тестування ефективності алгоритмів класифікації, що в результаті призводить до недостовірних результатів [27].

Kyoto-2006+ має досить нешироку типізацію атак (лише два типи). І через те, що дані дещо застарілі (трафік в рамках вибірки з 2006 по 2015 роки), модель не буде відповідати вимогам сучасної IDS і не буде демонструвати достатній рівень захисту в умовах сьогодення.

NSL-KDD хоча й позиціонує себе як датасет, в якому виправлено недоліки вибірки KDDcup99 (усунуто дублікати та неврівноваженість), але все ж базується на тих самих, вже досить застарілих даних. Також, у порівнянні з іншими вибірками, має дещо обмежений обсяг, що може стати причиною недостатньої ефективності навчання моделі.

Вибірку UNSW-NB15 деякі дослідники ставлять під сумніви, тому що вибірка містить штучно згенеровані дані, які можуть не повністю відображати реальні мережеві умови, з якими потім зіткнеться модель

Проаналізувавши недоліки деяких вибірок даних для навчання та тестування моделі, в рамках кваліфікаційної роботи навчання та тестування моделі системи

виявлення аномалій буде запропоновано на базі датасету CICIDS2017. Ця вибірка містить сучасні дані, у порівнянні з іншими наборами даних. Також, в рамках цього датасету представлені дані про сучасні атаки, такі як Brute Force, DoS, HeartBleed, WebAttack, Infiltration, Botnet, DDoS тощо, що робить її більш релевантною для поточних досліджень. Дані зібрані з реальних мережевих середовищ, що в майбутньому покращить застосовність моделі в реальних умовах. Також, CICIDS2017 виділяють як збалансовану та різноманітну вибірку даних. Така властивість сприяє покращенню якості моделі та її здатності розпізнавати широкий спектр загроз. Виходячи з вищенаведених переваг, вибір CICIDS2017 для навчання моделі дозволить створити більш надійну та ефективну IDS, яка зможе адекватно реагувати на загрози та забезпечувати високий рівень безпеки в реальних умовах.

2.4. Детальний огляд датасету CICIDS2017

Датасет CICIDS2017 являє собою архів розміром 271 Мб, який містить 8 файлів формату .csv. На рисунку 2.10 продемонстровані файли датасету в теці.

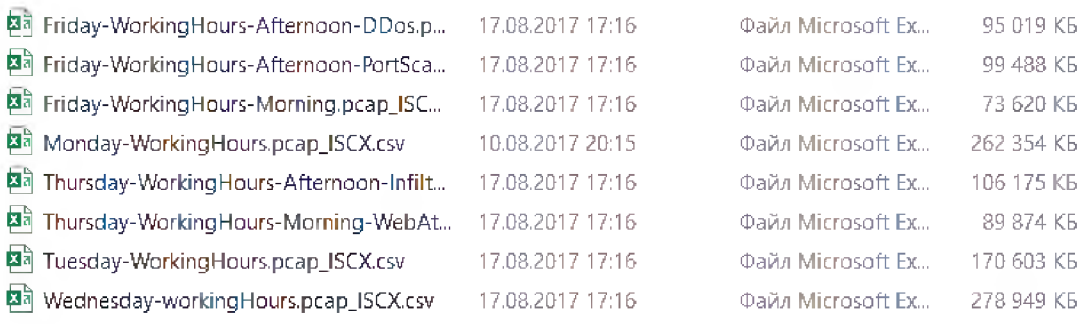

Рисунок 2.10 – Файли датасету в теці

Для подальшого опрацювання вибірки даних буде використана мова програмування Python.

2.4.1 Мова програмування Python як інструмент для обробки вибірки даних та імплементації машинного навчання

Python [28] – безкоштовна об'єктно-орієнтовна мова програмування із доволі простим синтаксисом та гнучкою структурою. Цією мовою програмування легко та зручно писати код та аналізувати його. Додатковою перевагою є велика кількість навчальних матеріалів, документації, книг, форумів, які присвячені Python. Крім того, у Python створено багато бібліотек, які полегшують вирішення певних задач. Для виконання завдань в рамках кваліфікаційної роботи використано Python версії 3.12.

Нижче буде коротко оглянуто бібліотеки, які були використані для реалізації запропонованого рішення.

Sklearn [29] – відкрита бібліотека для імплементації машинного навчання в Python. Вона забезпечує інструменти, які необхідні для створення і тренування моделі. Сьогодні є однією з найбільш застосовних бібліотек для реалізації машинного навчання на мові Python.

Pandas [30] – бібліотека, яка призначена для маніпуляції із даними та їх аналізу. В рамках бібліотеки пропонуються структури даних та методи, які призначені для роботи з наборами даних.

Matplotlib [31] – бібліотека на Python, яка призначена для візуалізації даних у вигляді двовимірної графіки. Matplotlib є гнучким інструментом для візуалізації та аналізу даних, особливо у поєднанні з бібліотеками типу NumPy.

NumPy [32] – бібліотека на Python, яка підтримує операції над багатовимірними масивами та матрицями, а також надає велику кількість математичних функцій для роботи з цими масивами даних.

В таблиці 2.4 наведено технічні характеристики комп'ютеру, на якому будуть виконуватися подальші обчислення та операції.

Таблиця 2.4 – Технічні характеристики ПК

CPU	AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz
RAM	16GB DDR4 2667 MHz
OS	Windows 10 Pro 21H2
GPU	Nvidia GeForce GTX 1650 + AMD Radeon™ Vega 8 Graphics

2.4.2. Попередня підготовка датасету до роботи

Оглянувши основні методи та засоби, за допомогою яких буде виконуватися реалізація машинного навчання в IDS, переходимо до етапу попередньої підготовки датасету. Перед тим, як оглянути та проаналізувати вміст вибірки даних, необхідно видалити невірні та неповні записи з файлів. Для цього, за допомогою бібліотеки Pandas читаємо та перезаписуємо пусті комірки у вибірці. Також, однією з умов підготовки вибірки даних є перетворення записів, які подані не в числовому форматі. Для цього використано клас LabelEncoder, який реалізований в бібліотеці Sklearn. Після очищення даних, об'єднуємо вісім файлів вибірки типу .csv (рисунок 2.10) в один, підготовлений для подальшого аналізу, файл типу .csv. На рисунку 2.11 наведено результати виконання коду програми, який наведено в додатку Д до пояснювальної записки.

```
Процедура може зайняти від 5 до 10 мінут, в залежності від потужності ПК.



Підготовка файлу "Monday-WorkingHours.pcap_ISCX" виконана.

Підготовка файлу "Tuesday-WorkingHours.pcap_ISCX" виконана.

Підготовка файлу "Wednesday-workingHours.pcap_ISCX" виконана.

Підготовка файлу "Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX" виконана.

Підготовка файлу "Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX" виконана.

Підготовка файлу "Friday-WorkingHours-Morning.pcap_ISCX" виконана.

Підготовка файлу "Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX" виконана.

Підготовка файлу "Friday-WorkingHours-Afternoon-DDos.pcap_ISCX" виконана.

Попередня підготовка виконана. Набір даних було очищено та підготовлено для роботи.
Витрачений час: =  415.9094834327698 секунд

Process finished with exit code 0
```

Рисунок 2.11 – Результат виконання коду програми для попередньої підготовки датасету

Важливо зауважити, що навчання моделі відбувається за навчальною вибіркою даних, а тестування ефективності навчання моделі відбувається за тестувальним датасетом. У випадку CICIDS2017 є лише одна вибірка даних, яка не поділена на навчальні та тестувальні дані. Щоб отримати необхідні нам два набори даних для задач навчання та тестування був використаний метод train_test_split, який реалізований в рамках бібліотеки Sklearn. Ця функція виконує розподіл однієї цілої вибірки на два набори – навчальний та тестувальний, за пропорцією 80% на 20% на користь навчальних даних.

2.4.3. Структура та частотні характеристики вибірки даних CICIDS2017

Після проведення підготовки даних датасету, виконано огляд даних у вибірці. CICIDS2017 загалом містить 3119345 записів, але після очищення вибірки даних, було видалено 288602 некоректних записи.

У вибірці даних містяться такі типи атаки, як DoS Hulk, PortScan, DDoS, DoS GoldenEye, FTP-Patator, SSH-Patator, DoS slowloris, DoS Slowhttptest, Bot, Web Attack – Brute Force, Web Attack – XSS, Infiltration, Web Attack – SQL Injection та Heartbleed. Нижче розглянемо кожну з них детальніше.

DoS Hulk – це атака типу DoS (скор. Denial-of-service), яка має за мету довести атаковану систему до відмовлення. Hulk – це абревіатура від речення «HTTP Unbearable Load King». В рамках атаки генерується велика кількість унікальних запитів з нерегулярним інтервалом від одного й того самого хосту. Таким чином, окрім здійснення атаки відмови в обслуговуванні, ще й відбувається введення захисних механізмів мережі в оману. Через це фільтрування трафіку стає важким. Загалом, атака такого типу частіше використовується як інструмент пентестинту, але також має місце і в застосуванні зловмисником для нанесення шкоди.

PortScan – це розповсюджена атака, яку використовують зловмисники для того, щоб знайти слабкі місця в мережі. Атака за допомогою сканування портів допомагає зловмисникам знайти відкриті порти і дослідити, чи приймаються або відправляються дані за допомогою цих портів. Також використовуючи цю атаку

можна розвідати, чи використовуються в організації такі засоби безпеки, як міжмережевий екран тощо. Сутність атаки полягає в відправленні запиту на порт із подальшим аналізом відповіді від порту для пошуку слабких місць мережі. Сканування портів дозволяє отримати таку інформацію, як от запущені служби, користувачі, яким належать служби, чи дозволені анонімні входи у систему, які мережеві служби потребують автентифікації тощо.

DDoS – розподілена атака відмови в обслуговуванні. Принцип здійснення такої атаки як і у DoS, окрім того, що здійснюється вона з декількох хостів, скоординувавши свої обчислювальні потужності для здійснення більш масової атаки, в той час коли DoS-атака зазвичай виконується з однієї машини.

DoS Goldeneye – це атака типу DoS, виконана за допомогою інструменту Goldeneye, який реалізований в рамках ОС Kali Linux. Цей інструмент використовується для тестування мережі на стійкість від атак відмови в обслуговуванні.

FTP-Patator – це атака на мережевий протокол FTP (скор. File Transfer Protocol), яка здійснюється за допомогою інструменту Patator, який реалізований в ОС Kali Linux. Сутність цієї атаки полягає в застосуванні методу грубої сили для зламу хешу і перехоплення даних. За допомогою цього інструменту можна виконувати велику кількість атак, зокрема й атаку SSH-Patator, яка наведена в рамках вибірки даних.

DoS slowloris – різновид атаки відмови в обслуговуванні, яка виконується за таким принципом: між ботом та атакованим сервером встановлюється сесія TCP. Після успішного створення сесії бот не відповідає ACK-пакетом для втримання сесії в відкритому стані, поки вона не закриється за таймаутом. Звісно що підтримання такої «пустої» сесії навантажує ресурси атакованого серверу, змушуючи витрачати їх на підтримання сесій із ботами. Із зростанням кількості таких сесій наступає відмова в обслуговуванні через нестачу обчислювальних ресурсів.

DoS Slowhttptest – атака відмови в обслуговуванні, яка виконується за допомогою інструменту Slowhttptest, який реалізований в ОС Kali Linux. Цей

інструмент реалізує найбільш розповсюджені види атак відмов в обслуговуванні одночасно.

Bot – атака, в рамках якої використовуються автоматизовані скрипти для порушення роботи мережі, крадіжки персональних даних, здійснення шахрайських покупок чи інших злочинних дій. Ці скрипти також можуть виконувати атаки відмови в обслуговуванні, але у порівнянні з інструментами для здійснення таких атак, вони мають більший функціонал і є гнучким інструментом для здійснення атак.

Web Attack – Brute Force – атака, яка використовує метод грубої сили для підбору будь-яких важливих даних, таких як паролю, чи зламу хешу тощо. Метод грубої сили являє собою завчасно налаштовані зловмисником дії, які виконуються багатопоточно із високою частотою за певними правилами, наприклад перебір по заданому словарю, перебор символів та інших комбінацій.

Web Attack – XSS – це різновид ін'єкцій, при якій шкідливі скрипти вбудовуються у надійні веб-сайти. Ця атака має назву «міжсайтовий скріптинг». В рамках цієї атаки зловмисник використовує веб-додаток для відправки шкідливого коду у вигляді сценарію на боці браузеру іншому кінцевому користувачу. Оскільки браузер кінцевого користувача вважає, що скрипт отриманий з надійного веб-сайту, він довіряє та виконує його. Виконання цього скрипту тягне за собою втрату файлів cookie, маркерів сеансів та іншої інформації, яка збережена у браузері кінцевого користувача. Такі сценарії мають змогу навіть переписувати HTML-код веб-сторінки.

Infiltration – атака, в рамках якої відбувається НСД в атаковану систему за допомогою різних інструментів. Це загальне позначення типу атак, які мають за мету проникнення в систему для викрадення конфіденційної інформації, порушення цілісності та/або доступності будь-якими методами.

Web Attack – SQL Injection – це атака, яка використовує вразливість бази даних. За допомогою шкідливого запиту зловмисник може отримати доступ до будь-яких даних в базі, а також змінювати та видаляти їх. Атаки такого типу можуть залишатися непомітними протягом тривалого часу, що надасть зловмиснику

постійний бекдор в ІКС організації.

Heartbleed – атака, яка використовує помилку в ПЗ OpenSSL. Через цю вразливість з'являється можливість читати пам'ять на сервері чи клієнті, в тому числі вилучати закритий ключ серверу для реалізації атаки «людина посередині». В рамках атаки відправляється змінений запит, що запитує дані з серверу. Через цю помилку валідність запиту не перевірялася, що дозволяє зловмиснику отримувати НСД до даних.

На рисунку 2.12 наведено статистику розподілу записів в датасеті, включаючи помилкові записи. Статистика містить у собі назви атак та кількість записів по кожній з них, які упорядковані від більшої кількості записів в файлі до меншої. Ці дані були отримані шляхом виведення в консоль після запуску програмного коду, наведеного в додатку Д.

Label	
BENIGN	2359289
DoS Hulk	231073
PortScan	158930
DDoS	41835
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack - Brute Force	1507
Web Attack - XSS	652
Infiltration	36
Web Attack - Sql Injection	21
Heartbleed	11

Рисунок 2.12 – Розподіл даних в датасеті

Проаналізувавши результати частотного розподілу, до рисунку 2.12 можна

зробити висновок, що у датасеті переважають доброякісні дані (BENIGN). Серед злоякісних даних у трійку за кількістю записів в датасеті входять атаки такого типу: DoS Hulk, PortScan та DDoS.

Нижче, на рисунках 2.13 – 2.15 наведено стовпчасті діаграми, які були побудовані за допомогою бібліотеки Matplotlib, які демонструють поділ даних за кількістю записів в датасеті на великі, середні та малі групи відповідно. До великої групи, як було зазначено вище, відносяться записи із доброякісними даними, та трійка злоякісних даних – атаки DoS Hulk, PortScan та DDoS. До середньої групи відносяться записи із кількістю від 652 до 10293 (рисунок 2.12). До малої групи відносяться атаки Heartbleed, SQL Injection та Infiltration. На рисунку 2.16 наведено процентний розподіл доброякісних та злоякісних даних у вибірці.

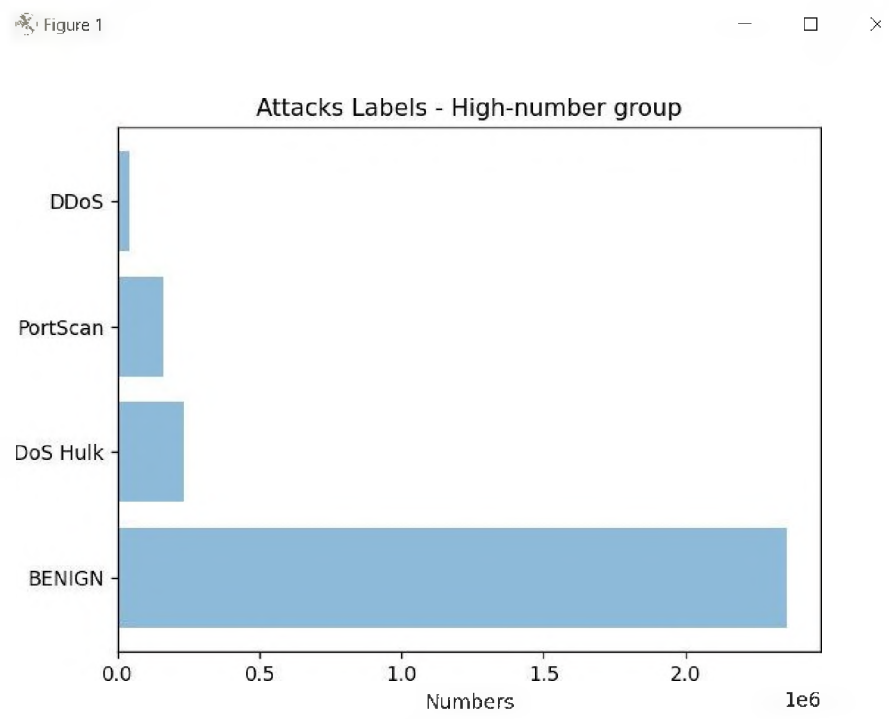

Рисунок 2.13 – Велика група даних у датасеті

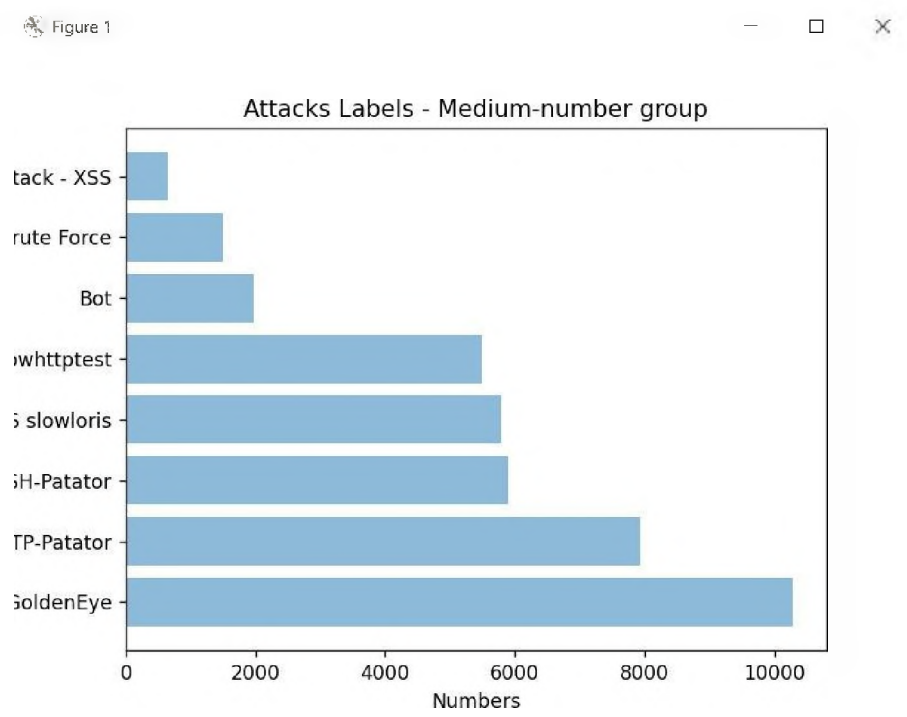

Рисунок 2.14 – Середня група даних в датасеті

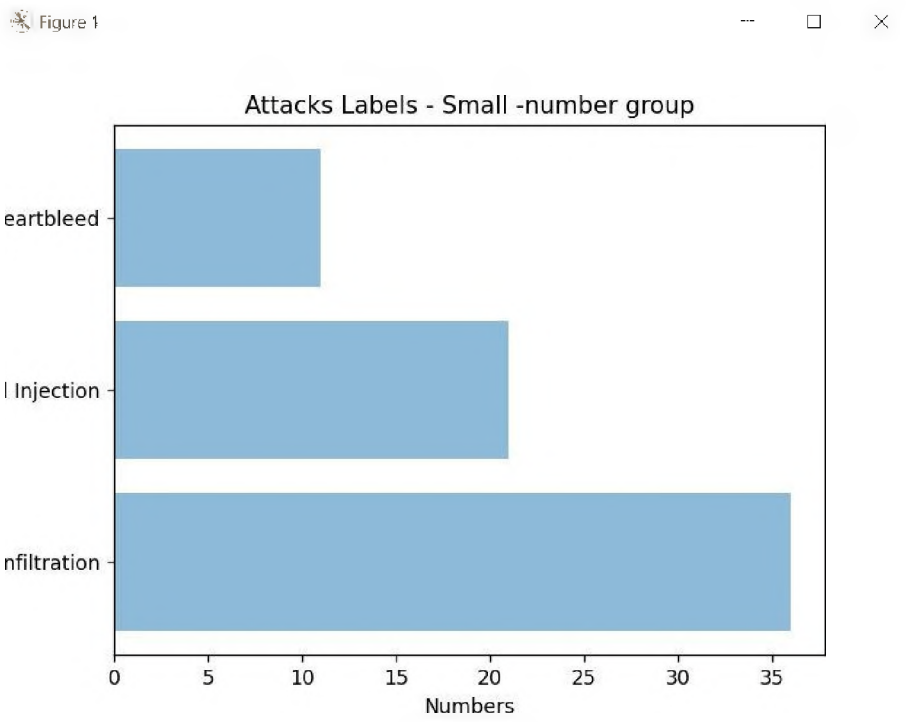

Рисунок 2.15 – Маленька група даних в датасеті

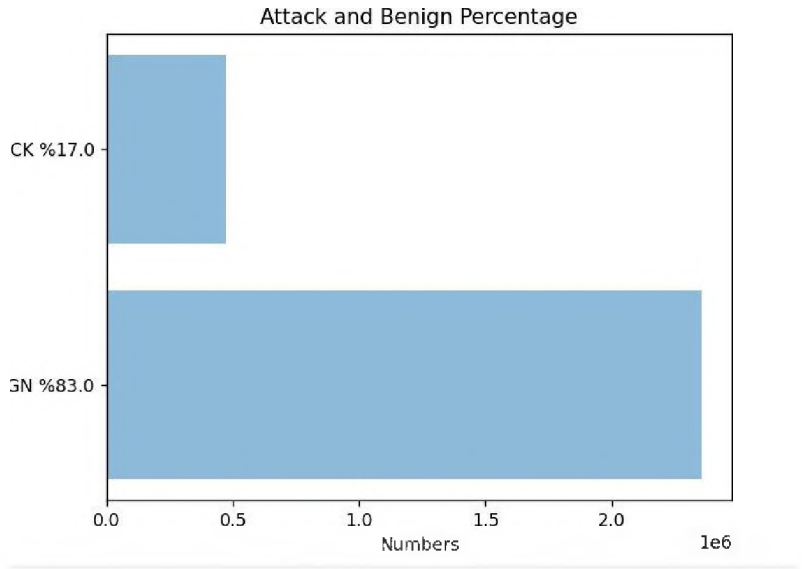

Рисунок 2.16 – Процентний розподіл доброякісних та злоякісних даних в датасеті

З рисунку 2.16 бачимо, що процентне представлення кількості записів із доброякісними даними дорівнює 83%. Для злоякісних даних процентне представлення дорівнює 17%. Далі, перед процесом імплементації машинного навчання, розглянемо методи оцінки ефективності застосування певного алгоритму.

2.5. Методи оцінки ефективності застосування алгоритмів машинного навчання

Результати, які буде демонструвати модель, оцінюватимуться за чотирма критеріями, за якими зазвичай оцінюють ефективність застосування алгоритму машинного навчання. Ці критерії мають назву accuracy, precision, f-measure та recall. Всі ці критерії приймають значення від 0 до 1, де 0 – найнижчий показник ефективності, 1 – найвищий показник ефективності.

Accuracy – показник відношення кількості успішно класифікованих атак до загальної кількості даних [33]. У формулі (2.1) наведено процес розрахунку показника.

$$\text{Accuracy} = \frac{TN+TP}{FP+TN+TP+FN} \tag{2.1}$$

Recall (Sensitivity) – показник відношення даних, які класифіковані як атака, до всіх даних атак. У формулі (2.2) наведено процес розрахунку показника.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{2.2}$$

Precision – показник відношення кількості успішно класифікованих атак до всіх класифікованих записів. [33]. У формулі (2.3) наведено процес розрахунку показника.

$$\text{Precision} = \frac{TP}{FP+TP} \tag{2.3}$$

F-measure – показник гармонічного середнього значення чутливості та точності. Цей показник використовується як результат загальної успішності моделі. У формулі (2.4) наведено процес розрахунку показника.

$$\text{F-measure} = \frac{2}{\frac{1}{Recall}+\frac{1}{Precision}} \tag{2.4}$$

У формулах (2.1), (2.2) та (2.3) використовуються такі значення:

1) TP (True Positive) – зразок, який класифікований як «атака».

2) FP (False Positive) – зразок нормальної поведінки, який класифікований як «атака»

3) FN (False Negative) – зразок атаки, який класифікований як нормальна поведінка

4) TN (True Negative) – зразок нормальної поведінки, який класифікований як нормальна поведінка.

Такий розподіл можна представити у вигляді матриці невідповідності, яка наведена на рисунку 2.17.

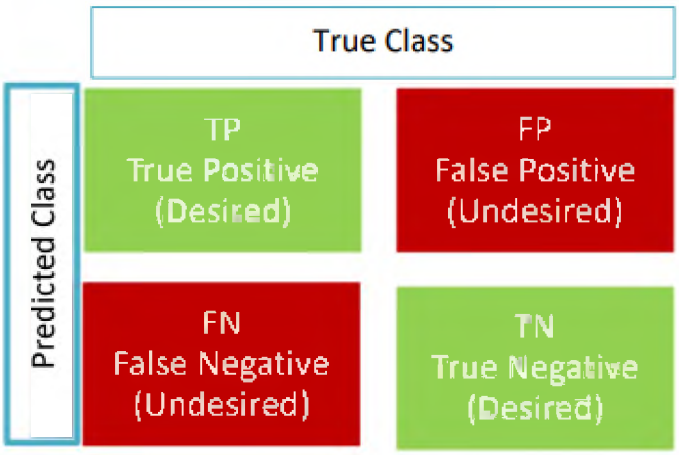

Рисунок 2.17 – Матриця невідповідності

2.6. Аналіз ознак CICIDS2017 для оптимізації роботи алгоритмів машинного навчання

Для оптимізації роботи алгоритмів машинного навчання є необхідність у виведенні найбільш важливих ознак датасету для кожного з наявних записів про атаки.

Процес обчислення показника важливості відбувався алгоритмом Random Forest за штучно створеною вибіркою даних, яка містить записи із нормальною поведінкою, та записами атак лише одного типу. Пропорції розподілу 30% на 70% на користь записів із нормальною поведінкою. Принцип обчислення базується на побудові дерева рішень, в якому для кожної з ознак надається показник значущості, спираючись на корисність ознаки при побудові дерева. Після ітерації алгоритму, ці показники порівнюються між собою та подаються в сортованому вигляді. Сума цих показників є загальною вагою всього дерева. Порівняння показника ознаки по відношенню до загальної ваги дерева демонструє важливість ознаки в рамках побудованого дерева.

Для того, щоб ефективно відібрати найважливіші ознаки, які дозволять визначити атаку, треба відмовитись від аналізу неузагальнених ознак, такі як IP-адреса, порт для проникнення в систему тощо. Це необхідно зробити через високу вірогідність того, що атакуюча сторона надасть перевагу використанню невідомого

порту для проникнення замість класичного, а також застосуванню великої кількості IP-адрес. Також, використання певного номеру порту може значно погіршити результат навчання моделі через те, що значний відсоток ПЗ використовує одні й ті самі номери портів.

На рисунку 2.18 наведено зразок формату виведення результату виконання коду програми для обчислення показників важливості в консоль на прикладі значень для типу атаки «Bot». Дані виводяться структуровано, кожна ознака з нового рядка, поруч із нею розрахований коефіцієнт важливості. Коефіцієнти при виведенні були відсортовані за спаданням.

```
Bot importance list:
Bot
                            importance
Features
Bwd Packet Length Mean        0.347262
Flow IAT Mean                 0.038789
Flow IAT Min                  0.014835
Flow Duration                 0.006453
Flow IAT Max                  0.002497
Flow IAT Std                  0.001951
Flow Packets/s                0.000861
Flow Bytes/s                  0.000720
Bwd Packet Length Max         0.000600
Total Backward Packets        0.000331
Bwd Packet Length Std         0.000297
Fwd Packet Length Mean        0.000233
Total Length of Bwd Packets   0.000185
Fwd Packet Length Max         0.000179
Fwd IAT Total                 0.000138
Total Length of Fwd Packets   0.000107
Fwd Packet Length Min         0.000055
Fwd Packet Length Std         0.000054
Total Fwd Packets             0.000048
Bwd Packet Length Min         0.000003
```

Рисунок 2.18 – Показники важливості для типу атаки «Bot»

Аналіз найважливіших ознак буде відбуватись на основі перших чотирьох ознаках, які мають найбільшу вагу. В таблиці 2.5 наведено узагальнення

найважливіших чотирьох ознак для кожного з типів атак.

Таблиця 2.5 – Узагальнення чотирьох найвищих показників важливості для типів атак

Атака та назва ознаки	Показник важливості ознаки
Bot	
Bwd Packet Length Mean	0,331250
Flow IAT Max	0,017769
Flow Duration	0,016612
Flow IAT Std	0,010918
DDoS	
Bwd Packet Length Std	0,471681
Total Backward Packets	0,094693
Fwd IAT Total	0,010499
Total Length of Fwd Packets	0,007061
DoS GoldenEye	
Flow IAT Max	0,495369
Total Backward Packets	0,047905
Bwd Packet Length Std	0,037416
Flow IAT Min	0,031267
DoS Hulk	
Bwd Packet Length Std	0,503321
Fwd Packet Length Std	0,070922
Fwd Packet Length Max	0,008432
Flow IAT Min	0,001632
DoS Slowhttpstest	
Flow IAT Mean	0,645150
Fwd Packet Length Min	0,073916
Bwd Packet Length Mean	0,024968
Fwd Packet Length Std	0,022666
DoS slowloris	
Flow IAT Mean	0,477731
Bwd Packet Length Mean	0,087704
Total Length of Bwd Packets	0,030311
Total Fwd Packets	0,020402
FTP-Patator	
Fwd Packet Length Max	0,053154
Fwd Packet Length Std	0,026250
Fwd Packet Length Mean	0,002166
Bwd Packet Length Std	0,002117
Heartbleed	
Bwd Packet Length Mean	0,064
Total Length of Bwd Packets	0,048
Bwd Packet Length Max	0,044
Total Fwd Packets	0,04

Кінець таблиці 2.5

Атака та назва ознаки	Показник важливості ознаки
Infiltration	
Total Length of Fwd Packets	0,198312
Flow Duration	0,095636
Fwd Packet Length Mean	0,083329
Fwd Packet Length Max	0,018956
PortScan	
Flow Bytes/s	0,313033
Total Length of Fwd Packets	0,304494
Flow Duration	0,000826
Flow IAT Max	0,000708
SSH-Patator	
Fwd Packet Length Max	0,000872
Flow IAT Mean	0,000671
Flow IAT Max	0,000490
Flow Duration	0,000478
Web Attack	
Total Length of Fwd Packets	0,013293
Bwd Packet Length Std	0,004364
Flow Bytes / s	0,003952
Total Fwd Packets	0,001612

Отже, виконано розрахунок показників важливості для кожного з типів атак, про які є записи у вибірці CICIDS2017. В таблиці 2.5 виділено чотири найзначущі ознаки. Під час розрахунку показника важливості вдалося дослідити, що майже для кожної з атак є одна чи декілька ознак, які зустрічаються в рейтингу і для інших атак. Також можемо визначити, що, наприклад, в атаці PortScan два найвищих показники важливості ознак, такі як кількість байт потоку даних в секунду та загальний розмір пакетів в прямому напрямку, мають дуже помітну різницю у порівнянні з іншими двома ознаками. Це викликано тим, що під час здійснення атаки типу PortScan її сутністю є відправка пакетів у надвеликій кількості без корисного навантаження, або з малою кількістю корисного навантаження. Під час цього процесу атакуюча сторона підвищує ефективність атаки, що виконується, при цьому не втрачаючи свою пропускну здатність. Тому, в рамках цієї атаки можемо зробити припущення, що ознака загальної довжини надісланих пакетів даних повинна бути помітно нижче, ніж доброякісні потоки.

Також можемо зазначити, що під час розрахунку важливості ознак для атаки

типу SSH-Patator не було виділено якісь ознаки, які мали б більшу вагу у порівнянні з іншими. Значення важливості дуже низькі та близькі за значенням один до одного. Причина цього в тому, що атака такого типу виконується за допомогою декількох етапів – сканування, застосування грубої сили та відмирання.

2.7. Розрахунок показників важливості для загального набору даних з атаками та нормальною поведінкою

Як було зазначено в розділі 2.3.2, де виконувалася попередня підготовка датасету до роботи, вісім файлів вибірки типу .csv були об'єднані в один. Зараз, в рамках цього розділу буде застосований інший підхід до визначення показників важливості ознак. Сутність цього підходу криється в застосуванні алгоритму Random Forest до загального файлу, де об'єднані всі 8 файлів вибірки. Всі дані про типи атак загально помічаються як атаки. Таким чином, у файлі містяться лише дві мітки – атаки та нормальна поведінка. Результати застосування цього підходу для розрахунку показників важливості наведені в таблиці 2.6.

Таблиця 2.6 – Рейтинг показників важливості для об'єднаної вибірки

Назва ознаки	Опис ознаки	Показник важливості ознаки
Bwd Packet Length Std	Стандартне відхилення розміру пакету в зворотному напрямку	0,246620
Flow Bytes/s	Кількість байтів потоку в секунду	0,178786
Total Length of Fwd Packets	Загальний розмір пакету в прямому напрямку	0,102427
Fwd Packet Length Std	Стандартне відхилення розміру пакету в прямому напрямку	0.063894
Flow IAT Std	Стандартне відхилення часу між двома пакетами в потоці	0,009896
Flow IAT Min	Мінімальний час між двома пакетами, надісланими в потоці	0,006940
Fwd IAT Total	Загальний час між двома пакетами, надісланими в прямому напрямку	0,005117

Кінець таблиці 2.6

Назва ознаки	Опис ознаки	Показник важливості ознаки
Flow Duration	Тривалість потоку	0,004144
Bwd Packet Length Max	Максимальний розмір пакету в зворотному напрямку	0,003996
Flow IAT Max	Максимальний час між двома пакетами, надісланими в потоці	0,003524
Flow IAT Mean	Середній час між двома пакетами, надісланими в потоці	0,003251
Total Length of Bwd Packets	Загальний розмір пакетів в прямому напрямку	0,001410
Fwd Packet Length Min	Мінімальний розмір пакету в прямому напрямку	0,000690
Bwd Packet Length Mean	Середній розмір пакету в зворотному напрямку	0,000599
Flow Packets / s	Кількість пакетів потоку в секунду	0,000542
Fwd Packet Length Mean	Середній розмір пакету в прямому напрямку	0,000502
Total Backward Packets	Загальна кількість пакетів в зворотному напрямку	0,000170
Total Fwd Packets	Загальна кількість пакетів в прямому напрямку	0.000128
Fwd Packet Length Max	Максимальний розмір пакету в прямому напрямку	0,000125
Bwd Packet Length Min	Мінімальний розмір пакету в зворотному напрямку	0,000079

На рисунку 2.19 представлено гістограму розподілу показнику важливості ознаки за даними таблиці 2.6.

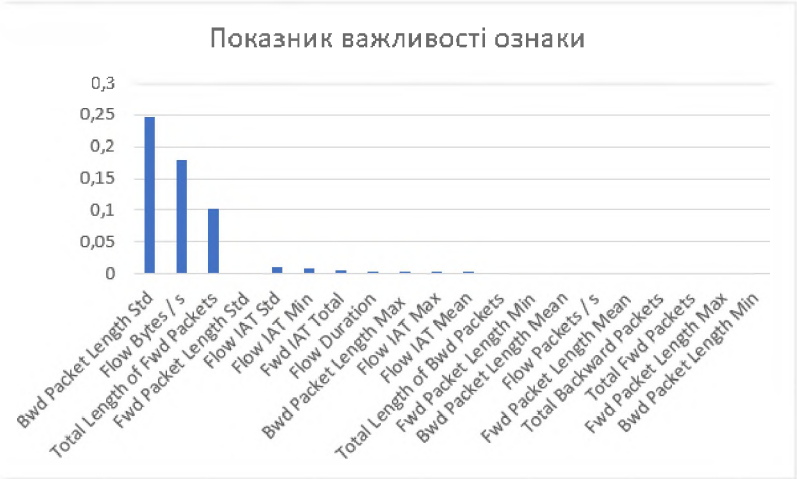

Рисунок 2.19 – Розподіл показників важливості для ознак в об’єднаній вибірці

Як бачимо з рисунку 2.19, топ-3 ознак для об'єднаної вибірки є Bwd Packet Length Std, Flow Bytes/s та Total Length of Fwd Packets.

2.8 Побудова моделей

Для моделювання в рамках кваліфікаційної роботи використано два різних підходи.

В першому випадку будуть використовуватися файли та ознаки, які були створені під час обчислення показників важливості для кожної з атак (таблиця 2.5). Ці файли, як і було зазначено в пункті 2.5, містять записи про атаки та записи із нормальною поведінкою в пропорції 30% на 70% на користь записів із атаками.

В рамках кваліфікаційної роботи моделювання виконується сімома алгоритмами машинного навчання, які було розглянуто в пункті 2.2. Принцип моделювання за цим підходом криється в застосуванні певного алгоритму до кожного з файлів рівно десять разів. Результат моделювання наводиться в форматі розрахунку показників ефективності застосування алгоритмів машинного навчання, які були розглянуті в рамках пункту 2.4, для кожного з алгоритмів після його застосування до певного файлу. Цей підхід має за мету дослідження ефективності застосування певного алгоритму машинного навчання до певних типів атак та його продуктивності, яку він демонструє під час моделювання.

В другому випадку уся вибірка даних з атаками та нормальною поведінкою використовується як один файл. Всі атаки в цьому наборі розмічаються як «атака». В результаті маємо файл із записами «атака» та нормальна поведінка.

Вибір ознак виконується шляхом об'єднання топ-4 ознак для кожної з атак в одну комбінацію ознак. Таким чином, не враховуючи повторення ознак, маємо комбінацію з вісімнадцяти ознак, ці ознаки наведено в таблиці 2.7.

Таблиця 2.7 – Комбінація вісімнадцяти ознак для типів атак

Bwd Packet Length Max F	Flow IAT Mean	Fwd Packet Length Min
Bwd Packet Length Mean	Flow IAT Min	Fwd Packet Length Std
Bwd Packet Length Std F	Flow IAT Std	Total Backward Packets

Кінець таблиці 2.7

Flow Bytes/s	Fwd IAT Total	Total Fwd Packets
Flow Duration	Fwd Packet Length Max	Total Length of Bwd Packets
Flow IAT Max	Fwd Packet Length Mean	Total Length of Fwd Packets

З іншого боку, є варіант реалізації вищенаведених підходів із використанням ознак з високою вагою важливості для об'єднаної вибірки (таблиця 2.6). Якщо встановити мінімальний відсотковий поріг частоти появи певної ознаки у вибірці даних в розмірі 0,8%, можна відібрати сім ознак замість запропонованих вісімнадцяти в таблиці 2.7. В таблиці 2.8 наведено сім ознак, які були відібрані із врахуванням мінімального відсоткового порогу. Частота появи ознаки обчислювалася як кількість записів про ознаку відносно загальної кількості записів.

Таблиця 2.8 – Комбінація семи ознак для типів атак

Назва ознаки	Показник важливості	Частота появи ознаки, %
Bwd Packet Length Std	0,246620	38,9%
Flow Bytes/s	0,178786	28,27%
Total Length of Fwd Packets	0,102427	16,19%
Fwd Packet Length Std	0.063894	10,11%
Flow IAT Std	0,009896	1,55%
Flow IAT Min	0,006940	1,09%
Fwd IAT Total	0,005117	0,7%

Отже, наразі наступає етап застосування машинного навчання. На рисунку 2.20 наведено зразок формату виведення результату імплементації алгоритмів машинного навчання до набору із вісімнадцяти ознаками. В такому ж форматі виводяться дані під час імплементації алгоритмів машинного навчання для набору із сьома ознаками. Програмний код, який був використаний для імплементації машинного навчання наведений в додатку Д.

Файл	Алгоритм ML	Accuracy	Precision	Recall	F-measure	Витрачений час
all_data	Naive Bayes	0.78	0.63	0.64	0.63	3.22
all_data	QDA	0.31	0.58	0.58	0.31	4.6581
all_data	Random Forest	0.94	0.96	0.83	0.88	22.04
all_data	ID3	0.95	0.97	0.86	0.9	29.6519
all_data	AdaBoost	0.95	0.95	0.87	0.91	378.1626
all_data	MLP	0.84	0.75	0.54	0.53	188.7249
all_data	Nearest Neighbors	0.97	0.94	0.95	0.95	2088.622

Рисунок 2.20 – Результати застосування алгоритмів машинного навчання

Узагальнення результатів застосування алгоритмів машинного навчання для набору із вісімнадцятьма ознаками наведені в таблиці 2.9.

Таблиця 2.9 – Результати застосування алгоритмів машинного навчання для вісімнадцяти ознак

Назва алгоритму машинного навчання	Значення критерію оцінки ефективності				
	F-measure	Precision	Recall	Accuracy	Час, хв
Naive Bayes	0,63	0,63	0,64	0,78	3,2
QDA	0,31	0,58	0,58	0,31	4,7
Випадковий ліс	0,88	0,96	0,83	0,94	22
Дерева рішень	0,90	0,97	0,86	0,95	29,66
Adaptive Boost	0,91	0,95	0,87	0,95	378,2
MLP	0,53	0,75	0,54	0,84	188,8
K-найближчих сусідів	0,95	0,94	0,95	0,97	2088,6

З результатів імплементації алгоритмів машинного навчання можна зробити висновок, що якщо обирати характеристикою витраченого часу на навчання моделі, то алгоритми Naive Bayes та QDA є найшвидшими, але вони демонструють досить посередні результати. Найкращий результат за чотирма показниками має алгоритм KNN, але витрати часу і, відповідно, обчислювальних ресурсів на його реалізацію надвеликі. Для того, щоб зробити більш точний висновок щодо інших алгоритмів, перейдемо до другого варіанту застосування цих алгоритмів, а саме до застосування їх до об'єднаної вибірки даних із застосуванням семи ознак, які наведені в таблиці 2.8. На рисунку 2.21 наведено результат виконання програмного коду навчання моделей для сьома ознак.

Файл	Алгоритми ML	Accuracy	Precision	Recall	F-measure	Time
all_data	Naive Bayes	0.82	0.66	0.64	0.65	1.886
all_data	QDA	0.38	0.58	0.61	0.38	2.3039
all_data	Random Forest	0.94	0.96	0.83	0.88	20.3664
all_data	ID3	0.95	0.93	0.89	0.91	12.558
all_data	AdaBoost	0.94	0.93	0.85	0.88	168.5123
all_data	MLP	0.84	0.75	0.54	0.53	164.5371
all_data	Nearest Neighbors	0.97	0.94	0.95	0.94	214.4103

Рисунок 2.21 – Виведення результатів навчання моделей в консоль

В таблиці 2.10 наведено узагальнення результатів застосування алгоритмів

машинного навчання для набору із сьома ознаками.

Таблиця 2.10 – Результати застосування алгоритмів машинного навчання для сьома ознак

Назва алгоритму машинного навчання	Значення критерію оцінки ефективності				
	F-measure	Precision	Recall	Accuracy	Час, хв
Naive Bayes	0,65	0,66	0,64	0,82	1,9
QDA	0,38	0,58	0,61	0,38	2,3
Випадковий ліс	0,88	0,96	0,83	0,94	20,37
Дерева рішень	0,91	0,93	0,89	0,95	12,56
Adaptive Boost	0,88	0,93	0,85	0,94	168,51
MLP	0,53	0,75	0,54	0,84	164,54
K-найближчих сусідів	0,94	0,94	0,95	0,97	214,41

Отже, після застосування алгоритмів машинного навчання для вибірки із сьома ознаками, можна порівняти результати із результатами їх застосування для вибірки із вісімнадцятьма ознаками. По-перше, бачимо скорочення часу, який необхідний для відпрацювання алгоритму. Це пов'язано із скороченням кількості ознак. Лідером за показниками залишився алгоритм KNN, але так само поступається за часом реалізації. В цілому, для більшості алгоритмів спостерігається або покращення критеріїв на декілька пунктів, або ті самі значення цих критеріїв. Виключенням є алгоритм Adaptive Boost, які при застосуванні для вибірки із сьома ознаками продемонстрував результат, який за всіма критеріями оцінки (окрім часу) відстає від результатів застосування його для вибірки із вісімнадцятьма ознаками на 1-3 пункти.

2.9. Висновки за розділом

В рамках спеціальної частини було досліджено відомості про використання алгоритмів машинного навчання в СВА. У дослідженні було розглянуто такі алгоритми: Naive Bayes, Quadratic discriminant analysis, Random Forest, Decision Trees, Adaptive Boost, Multilayer Perceptron, K-Nearest Neighbors.

Після огляду алгоритмів машинного навчання було стисло розглянуто

популярні вибірки даних для навчання та тестування моделі. Після їх аналізу вибір упав на вибірку CICIDS2017. Наведений датасет завдяки своїм перевагам, при застосуванні його для навчання та тестування, здатний створити надійну та ефективну IDS, яка буде мати змогу адекватно реагувати на загрози та забезпечувати високий рівень безпеки в реальних умовах.

Використовуючи мову програмування Python та реалізовані в ній інструменти та бібліотеки, було проведено попередню підготовку набору даних для роботи.

Для оцінки результатів застосування алгоритмів машинного навчання і подальшого зручного аналізу було використано чотири критерії оцінки – F-measure, Accuracy, Precision та Recall. Також алгоритми оцінювалися за часом, який був необхідний для відпрацювання алгоритму.

Проаналізувавши типи атак та їх ознаки, які наведені в рамках вибірки даних, було сформовано рейтинг найважливіших ознак. Виходячи з цього рейтингу, було обрано два підходи до обрання ознак для навчання моделі. Перший підхід базується на основі вісімнадцяти ознак, які були виведені шляхом об'єднання в одну комбінацію найважливіших чотирьох ознак для кожного з типів атак. Другий підхід реалізовується за допомогою застосування алгоритмів машинного навчання для вибірки із сьома ознаками, які були отримані шляхом обчислення важливості ознаки через частоту її зустрічі в записах, вираженої в процентному вигляді.

На базі вищенаведених двох підходів було виконано навчання моделей із застосуванням алгоритмів, які в рамках розділу були попередньо розглянуті. Нижченаведені результати являють собою середнє значення критерію F-measure (критерій, який демонструє загальний успіх застосування алгоритму), який був отриманий в результаті моделювання із застосуванням вибірки із вісімнадцяти та сьома ознаками: Naive Bayes – 0.64, QDA – 0.35, Random Forest – 0.88, Decision Trees – 0.91, Adaptive Boost – 0.90, MLP – 0.53, KNN – 0,95.

Як було зазначено вище, окрім чотирьох критеріїв оцінки, також було виконана оцінка часу, який необхідний для навчання моделі. Алгоритм KNN демонструє дуже гарні показники за чотирма критеріями оцінки алгоритмів, але потребує значно більше часу та обчислювальних ресурсів для навчання моделі.

Тож, проаналізувавши всі отриманні значення, можна зробити висновок, що алгоритм Decision Trees демонструє дуже гарні результати за метриками як для вісімнадцяти, так і для семи ознак, і також потребує адекватної витрати часу і, відповідно, обчислювальних ресурсів, які необхідні для навчання моделі. Виходячи з результатів проведеного дослідження, використання алгоритму Decision Trees в IDS для задач виявлення аномалій мережевого трафіку ІКС підприємства є найбільш оптимальним та виваженим рішенням, яке базується на його високих показниках ефективності.

РОЗДІЛ 3. ЕКОНОМІЧНИЙ РОЗДІЛ

Розробка та впровадження СВА мережевого трафіку на основі методу машинного навчання для ІКС підприємства потребує її економічного обґрунтування, тому метою цього розділу є здійснення відповідних розрахунків для встановлення економічної доцільності запропонованого рішення.

Для цього необхідно виконати такі розрахунки:

– капітальні витрати на проектування, розробку та впровадження програмного забезпечення;

– річні експлуатаційні витрати на утримання та обслуговування об'єкта проектування;

– показники економічної ефективності впровадження системи захисту на підприємстві.

3.1. Розрахунок капітальних витрат

Капітальні інвестиції – це кошти, призначені для створення і придбання основних фондів і нематеріальних активів, що підлягають амортизації.

Трудомісткість розробки СВА мережевого трафіку на основі машинного навчання для ІКС підприємства визначається як сума тривалості робочих операцій, які необхідні в рамках розроблення цієї системи. У формулі (3.1) наведено процес розрахунку трудомісткості:

$$t = t_{\text{ТЗ}} + t_{\text{о}} + t_{\text{м}} + t_{\text{р}} + t_{\text{д}}, \tag{3.1}$$

де $t_{\text{ТЗ}}$ – тривалість складання технічного завдання для проектування СВА мережевого трафіку на основі методу машинного навчання.

$t_{\text{ТЗ}}$ = 8 годин;

$t_{\text{о}}$ – тривалість ознайомлення з технічним завданням для вивчення потреб

замовника, пошук необхідної інформації. $t_о$ = 24 години;

$t_м$ – тривалість моделювання СВА на основі різних методів машинного навчання із метою виявлення найефективнішого для поданих в технічному завданні задач за допомогою аналізу результатів навчання. $t_м$ = 60 годин;

$t_р$ – тривалість розробки, навчання та тестування моделі на основі обраного алгоритму машинного навчання. $t_р$ = 20 годин.

$t_д$ = тривалість підготовки технічної документації. $t_д$ = 16 годин.

Отже, підставивши значення у формулу (3.1), отримуємо:

$$t = 8 + 24 + 60 + 20 + 16 = 128 \text{ годин}$$

Витрати на розробку політики використання СВА мережевого трафіку на основі машинного навчання для ІКС підприємства $К_{рп}$ складаються з витрат на заробітну платню фахівця з ІБ $З_{зп}$ та вартості витрат машинного часу, який необхідний для розробки цієї політики $З_{мч}$.

$$К_{рп} = З_{зп} + З_{мч}$$
$$З_{зп} = t * З_{зпг},$$

де t – трудомісткість розробки СВА мережевого трафіку на основі машинного навчання для ІКС підприємства, годин;

$З_{зпг}$ – середня заробітна платня фахівця з ІБ в годину.

Трудомісткість була визначена за формулою (3.1), і дорівнює 128 годин. Середня місячна заробітна платня фахівця з ІБ, за даними, які наведені в [34], дорівнює 25 тис. грн. Розділивши середню місячну заробітну платню фахівця з ІБ на кількість робочих годин в місяць, отримаємо значення $З_{зпг} = 156,25$ грн.

Отже:

$$З_{зп} = t * З_{зпг} = 128 * 156,25 = 20000 \text{ грн}$$

Вартість машинного часу для розробки політики використання СВА мережевого трафіку на основі машинного навчання для ІКС підприємства визначається за формулою (3.2):

$$З_{мч} = t * С_{мч}, \quad (3.2)$$

де t – трудомісткість розробки СВА мережевого трафіку на основі машинного навчання для ІКС підприємства, годин;

$С_{мч}$ – вартість 1 години машинного часу ПК, грн/година.

Вартість 1 години машинного часу ПК визначається за формулою (3.3):

$$С_{мч} = P * t_{нал} * С_е + \frac{Ф_{зал} * Н_а}{F_p} + \frac{К_{лпз} * Н_{апз}}{F_p}, \text{ грн}, \quad (3.3)$$

де Р – встановлена потужність ПК, кВт;

$t_{нал}$ – загальна кількість використаних ПК для реалізації рішення;

$С_е$ – тариф на електричну енергію, грн/кВт*година;

$Ф_{зал}$ – залишкова вартість ПК на поточний рік, грн;

$Н_а$ – річна норма амортизації на ПК, частки одиниці;

$Н_{апз}$ – річна норма амортизації на ліцензійне програмне забезпечення, частки одиниці;

$K_{лпз}$ – вартість ліцензійного програмного забезпечення, грн;

F_p – річний фонд робочого часу

Значення встановленої потужності ПК було взято як середнє значення потужності для ПК, який є оптимальним для виконання процесу машинного навчання. Значення тарифу на електричну енергію було взято за актуальним тарифом станом на момент написання цього розділу. Залишкова вартість ПК визначена як різниця між первісною вартістю ПК, який використовується для навчання моделі, і його поточною вартістю із урахуванням зносу. Значення норми амортизації на ПК розраховано лінійним методом. Значення річного фонду

робочого часу взято в розмірі 1920. Вартість ліцензійного ПЗ взята як вартість місячної підписки на ПЗ, яке необхідно для розробки системи.

Отже, підставимо значення у формулу (3.3):

$$С_{мч} = 0,8 * 3 * 4,32 + \frac{8000 * 0,5}{1920} + \frac{996 * 0,2}{1920} = 12,14 \text{ грн}$$

Тепер маємо всі необхідні значення для розрахунку вартості машинного часу для розробки політики використання СВА мережевого трафіку на основі машинного навчання для ІКС підприємства. Підставимо їх у формулу (3.2):

$$З_{мч} = 128 * 12,14 = 1553,92 \text{ грн}$$

Підставимо отримані значення у формулу розрахунку витрат на розробку політики використання СВА мережевого трафіку на основі машинного навчання для ІКС підприємства:

$$К_{рп} = З_{зп} + З_{мч} = 20000 + 1553,92 = 21553,92 \text{ грн}$$

Аналіз та оцінка ефективності застосування алгоритмів машинного навчання для вибору єдиного алгоритму, на якому буде базуватись СВА, виконувалися за допомогою бібліотек мови програмування, яка використовується для розробки системи. Через це додаткові витрати на придбання апаратного та програмного забезпечення не виникають. Витрати на залучення зовнішніх консультантів не виникають.

Виникають витрати на імплементацію системи в мережу ІКС підприємства, які складають 3000 грн.

Отже, в рамках запропонованого рішення, капітальні витрати визначаються за формулою (3.4):

$$К = К_{рп} + К_{н}, \quad (3.4)$$

де $К_{рп}$ – витрати на розробку політики використання СВА мережевого трафіку на основі машинного навчання для ІКС підприємства;

$К_{н}$ – витрати на імплементацію системи в мережу ІКС підприємства.

Для визначення розміру капітальних витрат підставимо значення до формули (3.4):

$$К = К_{рп} + К_{н} = 21553{,}92 + 3000 = 24553{,}92 \text{ грн}$$

Отже, капітальні витрати на проектування, розробку та впровадження СВА мережевого трафіку на основі машинного навчання для ІКС підприємства складають 24553,92 грн.

3.2. Розрахунок поточних (експлуатаційних) витрат

Річні поточні витрати на функціонування системи ІБ визначаються за формулою (3.5):

$$С = С_{в} + С_{к} + С_{ак}, \text{ грн}, \quad (3.5)$$

де $С_{в}$ – вартість відновлення й модернізації системи;

$С_{к}$ – витрати на керування системою ІБ в цілому;

$С_{ак}$ – витрати, викликані активністю користувачів системи ІБ.

Витрати на відновлення і модернізацію системи ($С_{в}$) визначаються вартістю планового оновлення та донавчання моделі на більш актуальній вибірці даних. Така модернізація проводиться два рази на рік спеціалістом з ІБ, і її сукупна вартість складає 6000 грн.

Витрати на керування системою ІБ в цілому ($С_{к}$) розраховуються за

формулою (3.6):

$$С_к = С_н + С_а + С_з + С_{єв} + С_{ел} + С_о + С_{тос}, \quad (3.6)$$

де $С_н$ – витрати на навчання адміністративного персоналу й кінцевих користувачів;

$С_а$ – річний фонд амортизаційних відрахувань;

$С_з$ – річний фонд заробітної плати інженерно-технічного персоналу, що обслуговує систему ІБ;

$С_{єв}$ – єдиний внесок на загальнообов'язкове державне соціальне страхування;

$С_{ел}$ – вартість електроенергії;

$С_о$ – витрати на залучення сторонніх організацій для виконання деяких видів обслуговування, навчання та сертифікацію обслуговуючого персоналу;

$С_{тос}$ – витрати на технічне й організаційне адміністрування та сервіс системи ІБ.

Витрати на навчання адміністративного персоналу й кінцевих користувачів ($С_н$) становлять 2000 грн.

Річний фонд амортизаційних відрахувань ($С_а$) при корисному терміні використання 2 роки за прямолінійним методом амортизації розраховується за формулою (3.7):

$$С_а = \frac{К}{2}, \text{ грн}, \quad (3.7)$$

де К – капітальні витрати.

Отже, річний фонд амортизаційних відрахувань складає 12277 грн.

Річний фонд заробітної плати інженерно-технічного персоналу, що обслуговує систему ІБ визначається за формулою (3.8):

$$С_з = З_{осн} + З_{дод}, \text{ грн}, \quad (3.8)$$

де $З_{осн}$ – основна заробітна плата;

$З_{дод}$ – додаткова заробітна плата.

За даними [34], основна заробітна плата фахівця з ІБ на місяць складає 25000 грн. Додаткова заробітна платня дорівнює 10% від основної заробітної плати. Виконання робіт впровадження СВА в ІКС підприємства потребує залучення фахівця з ІБ лише на 0,1 ставки. За цими даними можемо розрахувати річний фонд заробітної плати, підставивши їх у формулу (3.8):

$$С_з = З_{осн} + З_{дод} = (25000 * 12 + 25000 * 12 * 0{,}1) * 0{,}1 = 33000 \text{ грн}$$

Єдиний внесок на загальнообов'язкове державне соціальне страхування ($С_{єв}$), згідно чинної редакції Закону України «Про збір та облік єдиного внеску на загальнообов'язкове державне соціальне страхування» від 29.01.2024, становить 22% для всіх категорій платників. Тож:

$$С_{єв} = 33000 * 0{,}22 = 7260 \text{ грн}$$

Вартість електроенергії, що споживається апаратурою системи ІБ протягом року ($С_{ел}$), визначається за формулою (3.9):

$$С_{ел} = P * F_p * Ц_е, \text{ грн}, \quad (3.9)$$

де Р – встановлена потужність апаратури ІБ;

$F_р$ – річний фонд робочого часу системи ІБ;

$Ц_{ел}$ – тариф на електроенергію.

Встановлена потужність апаратури ІБ дорівнює 0,8 кВт, річний фонд робочого часу системи ІБ дорівнює 1920 год., тариф на електроенергію 4,32 грн. Отже, підставимо дані в формулу (3.9):

$$С_{ел} = P * F_p * Ц_е = 0{,}8 * 1920 * 4{,}32 = 6635{,}52 \text{ грн}$$

Витрати на залучення сторонніх організацій для виконання деяких видів обслуговування, навчання та сертифікацію обслуговуючого персоналу ($С_о$) не виникають.

Витрати на технічне й організаційне адміністрування та сервіс системи ІБ ($С_{тос}$) визначаються як 2% від вартості капітальних витрат. Отже:

$$С_{тос} = К * 0{,}02 = 24553{,}92 * 0{,}02 = 491{,}08 \text{ грн}$$

Отже, тепер маємо всі дані для розрахунку витрат на керування системою ІБ в цілому ($С_к$) за формулою (3.6):

$$С_к = 2000 + 12277 + 33000 + 7260 + 6635{,}52 + 491{,}08 = 61663{,}6 \text{ грн}$$

Розрахувавши витрати на керування системою ІБ, для визначення річних поточних витрат на функціонування системи ІБ залишилось розрахувати витрати, викликані активністю користувачів системи ІБ ($С_{ак}$). Ці витрати орієнтовно можна визначити у розмірі 10% від капітальних витрат. Тому:

$$С_{ак} = К * 0{,}1 = 24553{,}92 * 0{,}1 = 2455{,}4 \text{ грн}$$

Отже, тепер можна розрахувати річні поточні витрати на функціонування системи ІБ за формулою (3.5):

$$С = С_в + С_к + С_{ак} = 6000 + 61663{,}6 + 2455{,}4 = 70119 \text{ грн}$$

Таким чином, виконавши необхідні обчислення було визначено, що річні поточні витрати на функціонування системи ІБ складають 70119 грн.

3.3. Оцінка можливого збитку від атаки на вузол або сегмент мережі

Для того, щоб виконати розрахунок вартості збитків від атаки на вузол або сегмент мережі, можна застосувати спрощену модель оцінки для умовного підприємства. Необхідні вихідні дані для розрахунку:

$t_{\text{п}}$ – час простою вузла або сегмента внаслідок атаки, годин. ($t_{\text{п}} = 2$ години);

$t_{\text{в}}$ – час відновлення після атаки персоналом, що обслуговує корпоративну мережу, годин ($t_{\text{в}} = 3$ години);

$t_{\text{ви}}$ – час повторного введення загубленої інформації співробітниками атакованого вузла або сегмента корпоративної мережі, годин ($t_{\text{ви}} = 3$ години);

$З_{\text{о}}$ – заробітна плата обслуговуючого персоналу, грн ($З_{\text{о}} = 17000$ грн);

$З_{\text{с}}$ – заробітна плата співробітників атакованого вузла або сегмента корпоративної мережі, грн ($З_{\text{с}} = 20000$ грн);

$Ч_{\text{о}}$ – чисельність обслуговуючого персоналу, осіб ($Ч_{\text{о}} = 1$ особа);

$Ч_{\text{с}}$ – чисельність співробітників атакованого вузла або сегмента корпоративної мережі, осіб ($Ч_{\text{с}} = 3$ особи);

О – обсяг прибутку атакованого вузла або сегменту корпоративної мережі, грн (О = 200 тис. грн/рік);

$П_{\text{зч}}$ – вартість відновлення або заміни устаткування або запасних частин, грн ($П_{\text{зч}} = 12000$ грн);

І – число атакованих сегментів корпоративної мережі, одиниць (І = 1 одиниця);

N – середнє число атак на рік (N = 39).

Упущена вигода від простою атакованого вузла або сегменту корпоративної мережі визначається за формулою (3.10):

$$U = П_{\text{п}} + П_{\text{в}} + V, \tag{3.10}$$

де $П_{\text{п}}$ – оплачувані втрати робочого часу та простої співробітників атакованого вузла або сегмента корпоративної мережі, грн;

$П_в$ – вартість відновлення працездатності вузла або сегменту корпоративної мережі, грн;

V – втрати від зниження обсягу продажів за час простою атакованого вузла або сегмента корпоративної мережі, грн.

Втрати від зниження продуктивності співробітників атакованого вузла або сегменту корпоративної мережі складаються з втрат їхньої заробітної плати за час простою внаслідок атаки, і визначаються за формулою (3.11):

$$П_п = \frac{\sum З_с}{F} * t_п, \text{ грн}, \quad (3.11)$$

де $З_с$ – заробітна плата співробітників атакованого вузла або сегмента корпоративної мережі, грн;

F – місячний фонд робочого часу (при 40-годинному робочому тижні дорівнює 176 годин);

$t_п$ – час простою вузла або сегмента внаслідок атаки, годин.

Отже, підставимо дані у формулу (3.11):

$$П_п = \frac{\sum З_с}{F} * t_п = \frac{20000*3}{176} * 2 = 681{,}82 \text{ грн}$$

Витрати на відновлення працездатності вузла або сегменту корпоративної мережі обчислюються за формулою (3.12):

$$П_в = П_{ви} + П_{пв} + П_{зч}, \quad (3.12)$$

де $П_{ви}$ – витрати на повторне введення інформації, грн;

$П_{пв}$ – витрати на відновлення вузла або сегменту корпоративної мережі, грн;

$П_{зч}$ – вартість заміни устаткування або запасних частин, грн.

Витрати на повторне введення інформації ($П_{ви}$) обчислюються виходячи з розміру заробітної плати співробітників атакованого вузла або сегменту

корпоративної мережі ($З_с$), які зайняті повторним введенням втраченої інформації, з урахуванням необхідного для цієї операції часу ($t_{ви}$). Тож, розрахуємо витрати на повторне введення інформації:

$$П_{ви} = \frac{\sum З_с}{F} * t_{ви} = \frac{20000*3}{176} * 3 = 1022{,}73 \text{ грн}$$

Витрати на відновлення вузла або сегменту корпоративної мережі ($П_{пв}$) визначаються часом відновлення після атаки ($t_в$) і розміром середньогодинної заробітної плати обслуговуючого персоналу:

$$П_{пв} = \frac{\sum З_о}{F} * t_в = \frac{17000*1}{176} * 3 = 289{,}77 \text{ грн}$$

Наразі маємо всі дані для розрахунку витрат на відновлення працездатності вузла або сегменту корпоративної мережі за формулою (3.12):

$$П_в = П_{ви} + П_{пв} + П_{зч} = 1022{,}73 + 289{,}77 + 12000 = 13312{,}5 \text{ грн}$$

Втрати від зниження очікуваного обсягу прибутків за час простою атакованого вузла або сегмента корпоративної мережі визначаються виходячи із середньогодинного обсягу прибутку і сумарного часу простою сегмента корпоративної мережі, і обчислюються за формулою (3.13):

$$V = \frac{O}{F_г} * (t_п + t_в + t_{ви}), \text{ грн}, \qquad (3.13)$$

де О – обсяг прибутку атакованого вузла або сегменту корпоративної мережі, грн;

$F_г$ – річний фонд робочого часу підприємства (для умовного підприємства із 52 робочими тижнями, 5-ти денним робочим тижнем, 8-ми годинним робочим днем становить 2080 годин);

$t_{\text{п}}$ – час простою вузла або сегмента внаслідок атаки, годин;

$t_{\text{в}}$ – час відновлення після атаки персоналом, що обслуговує корпоративну мережу, годин;

$t_{\text{ви}}$ – час повторного введення загубленої інформації співробітниками атакованого вузла або сегмента корпоративної мережі, годин.

Отже, підставимо дані у формулу (3.13):

$$V = \frac{O}{F_{\text{г}}} * (t_{\text{п}} + t_{\text{в}} + t_{\text{ви}}) = \frac{200000}{2080} * (2 + 3 + 3) = 769{,}23 \text{ грн}$$

Далі підставимо необхідні дані для розрахунку упущеної вигоди від простою атакованого вузла або сегменту корпоративної мережі (U) у формулу (3.10):

$$U = \Pi_{\text{пв}} + \Pi_{\text{в}} + V = 289{,}77 + 13312{,}5 + 769{,}23 = 14371{,}5 \text{ грн}$$

Загальний збиток від атаки на вузол або сегмент корпоративної мережі (В) визначається за формулою (3.14):

$$B = \sum I \sum N\, U, \text{ грн}, \tag{3.14}$$

де I – число атакованих сегментів корпоративної мережі, одиниць;

N – середнє число атак на рік;

U – упущена вигода від простою атакованого вузла або сегменту корпоративної мережі.

Підставимо дані у формулу (3.14):

$$B = \sum I \sum N\, U = \sum 1 \sum 39\ \ 14371{,}5 = 560488{,}5 \text{ грн}$$

Таким чином, загальний збиток від атаки на вузол або сегмент корпоративної мережі становить 560488,5 грн.

3.4. Загальний ефект від впровадження системи інформаційної безпеки

Загальний ефект від впровадження системи ІБ (Е) визначається з урахуванням ризиків порушення ІБ, і визначається за формулою (3.15):

$$\text{Е} = B * R - C, \text{ грн}, \quad (3.15)$$

де В – загальний збиток від атаки у разі перехоплення інформації, грн;

R – вірогідність успішної реалізації атаки на вузол або сегмент мережі, частки одиниці;

С – щорічні витрати на експлуатацію системи ІБ.

Вірогідність успішної реалізації атаки на вузол або сегмент мережі умовного підприємства взято на рівні 22%. Тож, підставимо дані у формулу (3.15):

$$\text{Е} = B * R - C = 560488{,}5 * 0{,}22 - 70119 = 53188{,}47 \text{ грн}$$

Отже, загальний ефект від впровадження системи ІБ становить 53188,47 грн.

3.5. Визначення та аналіз показників економічної ефективності системи інформаційної безпеки

Для того, щоб визначити економічну ефективність системи ІБ, необхідно розрахувати та проаналізувати два показники: коефіцієнт повернення інвестицій (ROSI) та термін окупності капітальних інвестицій (T_o).

Коефіцієнт повернення інвестицій (ROSI) показує, скільки гривень додаткового прибутку приносить одна гривня капітальних інвестицій на впровадження системи ІБ. Цей коефіцієнт розраховується за формулою (3.16):

$$ROSI = \frac{E}{K}, \text{ частки одиниці}, \quad (3.16)$$

де Е – загальний ефект від впровадження системи ІБ, грн;

К – капітальні інвестиції за варіантами, що забезпечили цей ефект, грн.

Отже, підставимо необхідні дані у формулу (3.16):

$$ROSI = \frac{E}{K} = \frac{53188,47}{24553,92} = 2,17 \text{ частки одиниці}$$

Якщо умовне підприємство здійснює фінансування капітальних інвестицій у систему ІБ за рахунок позикових коштів (кредит), то проєкт вважається економічно доцільним, якщо розрахункове значення коефіцієнта повернення інвестицій перевищує величину банківської кредитної ставки з урахуванням інфляції:

$$ROSI > (N_{\text{кр}} + N_{\text{інф}})/100,$$

де $N_{\text{кр}}$ – банківська кредитна ставка, %;

$N_{\text{інф}}$ – річний рівень інфляції, %.

У випадку якщо умовне підприємство здійснює фінансування капітальних інвестицій за рахунок реінвестування власних коштів (частини прибутку та амортизаційних відрахувань), то проєкт визнається економічно доцільним, якщо розрахункове значення коефіцієнта повернення інвестицій перевищує величину річної депозитної ставки з урахуванням інфляції:

$$ROSI > (N_{\text{деп}} - N_{\text{інф}})/100$$

де $N_{\text{деп}}$ – річна депозитка ставка.

Припустимо, що банківська кредитна ставка ($N_{\text{кр}}$) становить 20%, річна депозитна ставка ($N_{\text{деп}}$) – 15%, річний рівень інфляції ($N_{\text{інф}}$) – 5%. Отже, у випадку якщо умовне підприємство здійснює фінансування капітальних інвестицій у

систему ІБ за рахунок позикових коштів, тоді розрахункове значення коефіцієнта повернення інвестицій становить:

$$2{,}17 > \frac{(20+5)}{100} = 2{,}17 > 0{,}25$$

У випадку, якщо умовне підприємство здійснює фінансування капітальних інвестицій за рахунок реінвестування власних коштів (частини прибутку та амортизаційних відрахувань), тоді значення розрахункового коефіцієнта повернення інвестицій дорівнює:

$$2{,}17 > \frac{15-5}{100} = 2{,}17 > 0{,}1$$

Як бачимо, в обох варіантах фінансування капітальних інвестицій проєкт вважається економічно доцільним.

Термін окупності капітальних інвестицій (T_o) показує, за скільки років капітальні інвестиції окупляться за рахунок загального ефекту від впровадження системи ІБ, і розраховується за формулою (3.17):

$$T_o = \frac{K}{E} = \frac{1}{ROSI}, \text{ років}, \qquad (3.17)$$

де К – капітальні інвестиції, грн;

Е – загальний ефект від впровадження системи ІБ, грн;

ROSI – коефіцієнт повернення інвестицій.

Отже, підставимо дані у формулу (3.17):

$$T_o = \frac{K}{E} = \frac{1}{ROSI} = \frac{1}{2{,}17} = 0{,}46 \text{ року}$$

Тож, термін окупності капітальних інвестицій (T_o) становить 0,46 року.

3.6. Висновки за розділом

Тож, підбиваючи підсумки економічного розділу можна зробити висновок, що розробка та впровадження СВА мережевого трафіку на основі машинного навчання для ІКС підприємства є економічно доцільним як при фінансуванні капітальних інвестицій за рахунок позикових коштів, так і за рахунок реінвестування власних коштів (частини прибутку та амортизаційних відрахувань).

Капітальні витрати у розмірі 24553,92 грн дозволяють отримати 53188,47 грн річного загального ефекту. Проведені розрахунки демонструють, що на 1 гривню капітальних витрат припадає 2,17 грн економічного ефекту (значення коефіцієнту ROSI становить 2,17 грн). При цьому річні поточні витрати на функціонування системи ІБ складають 70119 грн. Значення терміну окупності капітальних інвестицій становить 0,46 року.

ВИСНОВКИ

1. Роль інформаційних систем в сучасному світі зростає шаленими темпами, як і цінність інформації, яка обробляється та зберігається в цих системах. Це, звісно, стає привабливою мішенню для кіберзлочинців. Атаки, які вони здійснюють, постійно вдосконалюються, тому будь-яке підприємство потребує впровадження дієвого та ефективного підходу до задач виявлення аномалій мережевого трафіку. Сьогодні існує безліч методів, на яких базуються системи виявлення аномалій, але за своїми показниками швидкодії, участі людини у процесі виявлення аномалії та точності спрацювання виграють саме методи машинного навчання. Наразі на ринку пропонують певні системи виявлення аномалій, які базуються на різних методах, але в них є свої недоліки, починаючи від зручності використання, і завершуючи досить великою ціною придбання такої системи. Отже, задача розроблення системи виявлення аномалій, яка буде одночасно ефективною у виявлення атак, і також економічно доцільною для підприємства, залишається актуальною.

2. Ефективність системи виявлення аномалій мережевого трафіку, яка базується на методі машинного навчання, залежить, по-перше, від самого алгоритму машинного навчання, і по-друге, від якості вибірки даних, яку було обрано для навчання моделі. В рамках кваліфікаційної роботи було проаналізовано декілька популярних датасетів для навчання моделі, і обрано одну – CICIDS2017. Завдяки її перевагам, а саме тому, що вона є найбільш сучасною вибіркою даних у порівнянні з іншими, має ширший спектр атак у вибірці, а також використовує дані із реальних мережевих середовищ, навчання моделі із використанням цього датасету дозволить реалізувати ефективну модель. Для оцінки було обрано сім алгоритмів машинного навчання: Naive Bayes, Quadratic discriminant analysis, Random Forest, Decision Trees, Adaptive Boost, Multilayer Perceptron, K-Nearest Neighbors. Усі операції, такі як попередня підготовка датасету до роботи, навчання моделі, візуалізація та аналіз результатів було виконано за допомогою мови програмування Python та реалізованих у ній бібліотек. Вищенаведені алгоритми

оцінювалися за різними метриками, в результаті чого було зроблено висновок, що найбільш ефективний алгоритм машинного навчання для застосування його в системі виявлення аномалій є алгоритм Decision Trees. Саме його застосування є найбільш оптимальним та виваженим рішенням, спираючись на його високі показники ефективності та адекватні витрати машинного часу, який необхідний для навчання моделі.

3. У рамках кваліфікаційної роботи було виконано розрахунки капітальних витрат та річних експлуатаційних витрат, які необхідні для впровадження системи виявлення аномалій на запропонованому алгоритмі машинного навчання. Результати розрахунків загального ефекту, а також терміну окупності капітальних інвестицій продемонстрували, що запропоноване рішення є економічно доцільним для умовного підприємства, при чому як за позикові кошти (кредит), так і за рахунок реінвестування власних коштів (частини прибутку та амортизаційних відрахувань).

4. Отже, проведене дослідження демонструє, що застосування алгоритмів машинного навчання в системах виявлення аномалій дозволяє суттєво підвищити рівень захищеності мережі інформаційно-комунікаційної системи підприємства, і впровадження такого рішення на підприємстві є економічно доцільним.

5. Напрямком для майбутніх досліджень може бути застосування алгоритмів машинного навчання не тільки в системах виявлення, а і запобігання вторгнень. Якщо навчити модель не тільки виявляти загрозу і передавати інформацію про неї до відповідальної особи, а ще й автоматично запобігати цій загрозі, можна було б зменшити час реагування на загрозу, а відповідно і ймовірні втрати від атаки до мінімуму, а також скоротити вплив людського фактору на запобігання вторгненню.

ПЕРЕЛІК ПОСИЛАНЬ

1. НД ТЗІ 1.1-003-99. Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу, Київ 1999.

2. НД ТЗІ 1.4-001-2000. Типове положення про службу захисту інформації в автоматизованій системі, Київ 2000.

3. James P. Anderson Co. Computer Security Threat Monitoring and Surveillance, Box 42 Fort Washington, Pa. 19034, 1980.

4. Очеретний О.К. Моделювання передвісників атак на відмову в обслуговуванні комп'ютерних мереж. *Наукова думка інформаційного століття:* матеріали міжн. наук.-тех. конф., том 6, м. Дніпро, 19 червня 2017 р. Дніпро, 2017. С. 113-120.

5. Лагун І.І., Лагун А.Е. Використання дискретного малохвильового перетворення для виявлення аномалій мережевого трафіку: стаття: Національний університет «Львівська політехніка». Львів, 2011.

6. Gunes Kayacik, A. Nur Zincir-Heywood, Malcolm I. Heywood. Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99: *Third Annual Conference on Privacy, Secure and Trust:* conference, October 12-14, 2005, The Fairmont Algonquin, St. Andrews, New Brunswick, Canada, Proceedings.

7. Рубан І.В., Мартовицький В.О., Партика С.О. Класифікація методів виявлення аномалій в інформаційних системах. *Системи озброєння і військова техніка*, 2016, № 3 (47). Харківський національний університет радіоелектроніки, Харків, 2016.

8. Rebecca Bace, Peter Mell. Intrusion Detection Systems: *NIST Special Publication on Intrusion Detection System.*

9. Коробейнікова Т.І., Цар О.О. Аналіз сучасних відкритих систем виявлення та запобігання вторгнень. *Міжнародний науковий журнал «Грааль науки»*, № 27 (травень, 2023). DOI: 10.36074/grail-of-science.12.05.2023.050.

10. Скріншот: «Suricata IDS». *Flickr*: вебсайт. URL: https://www.flickr.com/photos/xmodulo/23720382520/

11. Stephen Cooper. The Best Network Intrusion Detection Systems Software & NIDS Tools. *Comparitech*: вебсайт. URL: https://www.comparitech.com/net-admin/nids-tools-software/#What_is_the_difference_between_NIDS_and_HIDS

12. Петришин В.С., Поліщук Д.О., Ніколюк П.К. Машинне навчання. *Комп'ютерні технології обробки даних:* матеріали III Всеукраїнської наук.-практ. конф. (м. Вінниця, 8 грудня 2022 р.), Вінниця, 2022.

13. Аліна Присяжнюк. Як працює Machine Learning та його застосування на практиці. *Na chasi*: вебсайт. URL: https://nachasi.com/tech/2019/01/31/yak-pratsyuye-machine-learning/

14. What is unsupervised learning? *IBM*: вебсайт. URL: https://www.ibm.com/topics/unsupervised-learning

15. Sam T. Roweis, Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. DOI: 10.1126/science.290.5500.2323

16. Leszek Rutkowsi, Maciej Jaworski, Lena Pietruczuk, Piotr Duda. Decision Trees for Mining Data Streams Based on the Gaussian Approximation. DOI: 10.1109/TKDE.2013.34

17. Decision Tree Algorithm. *Medium*: вебсайт. URL: https://blog.gopenai.com/decision-tree-algorithm-484ec33387f9

18. Вознюк М.Ю., Левківський В.Л. Аналіз моделей класифікації в Python для створення системи автоматичної категорізації публікацій блогу. Державний університет «Житомирська політехніка».

19. Xiaodong Yang, Ying Li Tian. EigenJoints-based action recognition using Naïve-Bayes-Nearest-Neighbor. *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*: conference (16-21 June 2012 Providence, RI, USA), Providence, RI, USA, 2012. DOI: 10.1109/CVPRW.2012.6239232.

20. What is the Naive Bayes Alhorithm? *Data Base Camp*: вебсайт. URL: https://databasecamp.de/en/ml/naive-bayes-algorithm

21. John Paul Mueller, Luca Massaron. Machine Learning for Dummies. 29.03.2021 p.

22. Will Koehrsen. Random Forest Simple Explanation. *Medium*: вебсайт. URL:

https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d

23. D.D. Denison, M.H. Hansen, C.C. Holmes, B. Mallick, B.Yu. Nonlinear Estimation and Classification. *The Boosting Approach to Machine Learning: An Overview:* pp 149-171.

24. AdaBoost Algorithm in Machine Learning. *AlmaBetter*: вебсайт. URL: https://www.almabetter.com/bytes/tutorials/data-science/adaboost-algorithm

25. Ethem Alpaydin. Introdution to Machine Learning, third edition. The MIT Press Cambridge, Massachusetts London, England, 2014.

26. Multi-Layer Perceptron Learning in Tensorflow. *Geekforgeeks*: вебсайт. URL: https://www.geeksforgeeks.org/multi-layer-perceptron-learning-in-tensorflow/

27. Сех І.І., Герович М.Б., Федисів Л.В., Пелещак О.А. Бази даних атак для навчання та тестування інтелектуальних систем виявлення вторгнень. *Актуальні задачі сучасних технологій*: матеріали X міжн. наук.-практ. конф. молодих учених та студентів (24-25 листопада 2021 р., Тернопіль), Тернопіль, 2021.

28. Офіційна документація мови програмування Python. URL: https://docs.python.org/3.12/

29. Офіційна документація Python-бібліотеки Scikit-learn. URL: https://scikit-learn.org/stable/

30. Офіційна документація Python-бібліотеки Pandas. URL: https://pandas.pydata.org

31. Офіційна документація Python-бібліотеки Matplotlib. URL: https://matplotlib.org

32. Офіційна документація Python-бібліотеки Numpy. URL: https://numpy.org

33. Monowar H. Bhuyan, D.K. Bhattacharyya, J.K. Kalita. Network Anomaly Detection: Methods, Systems and Tools: *IEEE Communications Surveys & Tutorials* (Volume: 16, Issue: 1, First Quarter 2014), 06.06.2013, pp. 303-336. DOI: 10.1109/SURV.2013.052213.00046

34. Спеціаліст з інформаційної безпеки: середня зарплата в Україні. *Work.ua*: вебсайт. URL: http://surl.li/nuvuct

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

№	Формат	Найменування	Кількість листів	Примітка
1	А4	Реферат	2	
2	А4	Список умовних скорочень	1	
3	А4	Зміст	2	
4	А4	Вступ	2	
5	А4	1 Розділ	17	
6	А4	2 Розділ	37	
7	А4	3 Розділ	16	
8	А4	Висновки	2	
9	А4	Перелік посилань	3	
10	А4	Додаток А	1	
11	А4	Додаток Б	1	
12	А4	Додаток В	1	
13	А4	Додаток Г	1	
14	А4	Додаток Д	14	

ДОДАТОК Б. Перелік документів на оптичному носії

Пояснювальна записка Дзядек.docx

Пояснювальна записка Дзядек.pdf

Презентація.pptx

ML_Python_Code.zip

Довідка_плагіат.pdf

ДОДАТОК В. Відгуки керівників розділів

Відгук керівника економічного розділу:

Економічний розділ виконаний відповідно до вимог, які ставляться до кваліфікаційних робіт, та заслуговує на оцінку 95 б. («відмінно»).

Керівник розділу _____ Дар'я ПІЛОВА

(підпис) (ім'я, прізвище)

ДОДАТОК Г. ВІДГУК

на кваліфікаційну роботу бакалавра на тему:
«Методи машинного навчання для виявлення аномалій мережевого трафіку інформаційно-комунікаційної системи підприємства»
студента групи 125-20-2
Дзядека Максима Івановича

Пояснювальна записка складається з титульного аркуша, завдання, реферату, списку умовних скорочень, змісту, вступу, трьох розділів, висновків, переліку посилань та додатків, розташованих на 102 сторінках та містить 30 рисунків, 13 таблиці, 34 джерела та 5 додатків.

Мета роботи: дослідження алгоритмів машинного навчання та оцінка їх ефективності для задач виявлення аномалій мережевого трафіку для інформаційно-комунікаційної системи підприємства.

У першому розділі було проаналізовано існуючі методи виявлення аномалій, які застосовуються в системах виявлення аномалій, наведено їх порівняльну характеристику, а також проаналізовано програмні реалізації IDS/IPS на ринку.

У другому розділі було розроблено підхід для оцінки ефективнішого алгоритму машинного навчання для задач виявлення аномалій мережевого трафіку для інформаційно-комунікаційної системи підприємства.

Практична цінність розробки полягає у застосуванні алгоритмів машинного навчання в системах виявлення аномалій мережевого трафіку для підвищення рівня захищеності інформаційно-комунікаційної системи підприємства.

Студент показав достатній рівень володіння теоретичними положеннями з обраної теми, показав здатність формувати власну точку зору (теоретичну позицію).

Робота оформлена та написана грамотною мовою, відповідає вимогам положення про систему запобігання та виявлення плагіату у Національному технічному університеті «Дніпровська політехніка». Містить необхідний ілюстрований матеріал. Автор добре знає проблему, уміє формулювати наукові та практичні завдання і знаходить адекватні засоби для їх вирішення.

В цілому робота задовольняє усім вимогам і може бути допущена до захисту, а його автор заслуговує на оцінку «_____».

Керівник спец. розділу
ст. викл. каф. БІТ Вадим МЄШКОВ

ДОДАТОК Д. Лістинг коду програми

1) Файл 01_Підготовка.py

```
import pandas as pd  # імпорт бібліотеки для роботи із даними
import os  # імпорт бібліотеки із стандартними функціями
from sklearn import preprocessing  # імпорт бібліотеки машинного навчання
import time  # імпорт бібліотеки, яка буде виступати лічільником часу виконання коду
from variables import CSV_FILES, MAIN_LABELS  # імпорт констант

seconds = time.time()
print("Процедура може зайняти від 5 до 10 мінут, в залежності від потужності ПК.\n\n\n")
number = "0123456789"
main_labels2 = MAIN_LABELS
MAIN_LABELS = (",".join(i for i in MAIN_LABELS))
MAIN_LABELS = MAIN_LABELS + "\n"
flag = True
for i in range(len(CSV_FILES)):
    ths = open(str(i) + ".csv", "w")
    ths.write(MAIN_LABELS)
    with open("./CSVs/" + CSV_FILES[i] + ".csv", "r") as file:
        while True:
            try:
                line = file.readline()
                if line[0] in number:
                    if " – " in str(line):
                        line = (str(line).replace(" – ", " - "))
                    line = (str(line).replace("inf", "-1"))
                    line = (str(line).replace("Infinity", "-1"))

                    line = (str(line).replace("NaN", "0"))

                    ths.write(str(line))
                else:
                    continue
            except:
                break
    ths.close()

    df = pd.read_csv(str(i) + ".csv", low_memory=False)
    df = df.fillna(0)

    string_features = ["Flow Bytes/s", "Flow Packets/s"]
    for ii in string_features:
        df[ii] = df[ii].replace('Infinity', -1)
        df[ii] = df[ii].replace('NaN', 0)
        number_or_not = []
        for iii in df[ii]:
            try:
                k = int(float(iii))
```

```
            number_or_not.append(int(k))
        except:
            number_or_not.append(iii)
    df[ii] = number_or_not

string_features = []
for j in main_labels2:
    if df[j].dtype == "object":
        string_features.append(j)
try:
    string_features.remove('Label')
except:
    print("error!")
labelencoder_X = preprocessing.LabelEncoder()

for ii in string_features:
    try:
        df[ii] = labelencoder_X.fit_transform(df[ii])
    except:
        df[ii] = df[ii].replace('Infinity', -1)
df = df.drop(main_labels2[61], axis=1)

if flag:
    df.to_csv('all_data.csv', index=False)
    flag = False
else:
    df.to_csv('all_data.csv', index=False, header=False, mode="a")
os.remove(str(i) + ".csv")
print('Підготовка файлу "' + CSV_FILES[i] + '" виконана.\n')

print("Попередня підготовка виконана. Набір даних було очищено та підготовлено для роботи.")
print("Витрачений час: = ", time.time() - seconds, "секунд")
```

2) Файл 02_Статистика.py

```
import numpy as np  # імпорт бібліотеки для роботи із масивами
import matplotlib.pyplot as plt  # імпорт бібліотеки для візуалізації результатів
import pandas as pd  # імпорт бібліотеки для роботи із даними

plt.rcdefaults()


def graph(objects, performance, x_label, y_label):  # функція для виведення графіку на екран
    y_pos = np.arange(len(objects))
    plt.barh(y_pos, performance, align='center', alpha=0.5)
    plt.yticks(y_pos, objects)
    plt.xlabel(x_label)
    plt.title(y_label)
    plt.show()
```

```
df = pd.read_csv('all_data.csv', usecols=["Label"])
print(df.iloc[:, 0].value_counts())
a = (df.iloc[:, 0].value_counts())

key = a.keys()
values = a.values
small_labels = []
small_values = []
big_labels = []
big_values = []
medium_labels = []
medium_values = []
attacak = 0
benign = 0

for i in range(0, len(values)):
    if values[i] > 11000:
        big_labels.append(str(key[i]))
        big_values.append(values[i])
    elif values[i] < 600:
        small_labels.append(str(key[i]))
        small_values.append(values[i])
    else:
        medium_labels.append(str(key[i]))
        medium_values.append(values[i])

    if str(key[i]) == "BENIGN":
        benign += values[i]
    else:
        attacak += values[i]

key = [benign, attacak]

labels = ["BENIGN %" + str(round(benign / (benign + attacak), 2) * 100),
          "ATTACK %" + str(round(attacak / (benign + attacak), 2) * 100)]
graph(big_labels, big_values, "Numbers", "Attacks Labels - High-number group")
graph(medium_labels, medium_values, "Numbers", "Attacks Labels - Medium-number group")
graph(small_labels, small_values, "Numbers", "Attacks Labels - Small -number group")
graph(labels, key, "Numbers", "Attack and Benign Percentage")
```

3) Файл 03_Фільтрація атак.py

```
import random  # імпорт бібліотеки для випадкового вибору запису із вибірки
import os  # імпорт бібліотеки із стандартними функціями системи
import pandas as pd  # імпорт бібліотеки для роботи із даними
import time  # імпорт бібліотеки, за допомогою якої буде вібуватися відлік часу на виконання коду
from variables import MAIN_LABELS, ATTACKS_DICT  # імпорт констант
from functions import folder  # імпорт функції для виклику помилки, якщо папка результатів не була створена

seconds = time.time()
```

```
print("Процес може зайняти від 3 до 8 хвилин, в залежності від потужності ПК.\n\n\n")
MAIN_LABELS = (",".join(i for i in MAIN_LABELS))
folder("./attacks/")
BENIGN = 2359289
for i in ATTACKS_DICT:
    a, b = 0, 0
    ths = open(".\\attacks\\" + i + ".csv", "w")
    ths.write(str(MAIN_LABELS) + "\n")
    benign_num = int(BENIGN / (ATTACKS_DICT[i] * (7 / 3)))
    with open("all_data.csv", "r") as file:
        while True:
            try:
                line = file.readline()
                line = line[:-1]
                k = line.split(",")
                if k[83] == "BENIGN":
                    rnd = random.randint(1, benign_num)
                    if rnd == 1:
                        ths.write(str(line) + "\n")
                        b += 1
                if k[83] == i:
                    ths.write(str(line) + "\n")
                    a += 1
                else:
                    continue
            except:
                break
    ths.close()
    print(i, "Файл перевірено.\n Атаки:%d\n Норма:%d\n\n\n " % (a, b))

webs = ["Web Attack - Brute Force", "Web Attack - XSS", "Web Attack - Sql Injection"]
flag = True
for i in webs:
    df = pd.read_csv(".\\attacks\\" + str(i) + ".csv")
    if flag:
        df.to_csv('.\\attacks\\Web Attack.csv', index=False)
        flag = False
    else:
        df.to_csv('.\\attacks\\Web Attack.csv', index=False, header=False, mode="a")
    os.remove(".\\attacks\\" + str(i) + ".csv")

print("Задачу завершено.")
print("Витрачено часу = ", time.time() - seconds, "секунд")
```

4) Файл 04_1_Вибір функції для файлів атак.ру

```
import numpy as np  # імпорт бібліотеки для роботи із масивами
import os  # імпорт бібліотеки із стандартними функціями
import pandas as pd  # імпорт бібліотеки для роботи із даними
import matplotlib.pyplot as plt  # імпорт бібліотеки для візуалізації результатів
from sklearn.ensemble import RandomForestRegressor  # імпорт бібліотеки машинного навчання
import time  # імпорт бібліотеки, за допомогою якої буде вибуватися відлік часу на виконання коду
```

```
from variables import MAIN_LABELS  # імпорт констант
from functions import folder  # імпорт функції для виклику помилки, якщо папка результатів не була створена

seconds = time.time()
csv_files = os.listdir("attacks")
ths = open("importance_list_for_attack_files.csv", "w")
folder("./feaure_pics/")
for j in csv_files:
    df = pd.read_csv(".\\attacks\\" + j, usecols=MAIN_LABELS)
    df = df.fillna(0)
    attack_or_not = []
    for i in df["Label"]:
        if i == "BENIGN":
            attack_or_not.append(1)
        else:
            attack_or_not.append(0)
    df["Label"] = attack_or_not

    y = df["Label"].values
    del df["Label"]
    X = df.values

    forest = RandomForestRegressor(n_estimators=250, random_state=0)
    forest.fit(X, y)
    importances = forest.feature_importances_
    std = np.std([tree.feature_importances_ for tree in forest.estimators_],
                 axis=0)
    indices = np.argsort(importances)[::-1]
    refclasscol = list(df.columns.values)
    impor_bars = pd.DataFrame({'Features': refclasscol[0:20], 'importance': importances[0:20]})
    impor_bars = impor_bars.sort_values('importance', ascending=False).set_index('Features')
    plt.rcParams['figure.figsize'] = (10, 5)
    impor_bars.plot.bar()
    count = 0
    fea_ture = j[0:-4] + "=["
    for i in impor_bars.index:
        fea_ture = fea_ture + "\"" + str(i) + "\","
        count += 1
        if count == 5:
            fea_ture = fea_ture[0:-1] + "]"
            break
    print(j[0:-4], "importance list:")
    print(j[0:-4], "\n", impor_bars.head(20), "\n\n\n")
    print(fea_ture)
    plt.title(j[0:-4] + " Attack - Feature Importance")
    plt.ylabel('Importance')
    plt.savefig("./feaure_pics/" + j[0:-4] + ".pdf", bbox_inches='tight', orientation='portrait', format='pdf')
    ths.write(fea_ture)
    plt.tight_layout()
    plt.show()
    print("----------------------------------------------------------------------------------------------------\n\n\n\n")
```

```
print("Задачу виконано!")
print("Витрачений час = ", time.time() - seconds, "секунд")
ths.close()
```

5) Файл 04_2_Вибір функції для all_data.py

```
import numpy as np  # імпорт бібліотеки для роботи із масивами
import pandas as pd  # імпорт бібліотеки для роботи із даними
import matplotlib.pyplot as plt  # імпорт бібліотеки для візуалізації результатів
from sklearn.ensemble import RandomForestRegressor  # імпорт бібліотеки машинного навчання
import time  # імпорт бібліотеки, за допомогою якої буде вибуватися відлік часу на виконання коду
from variables import MAIN_LABELS  # імпорт констант
from functions import folder  # імпорт функції для виклику помилки, якщо папка результатів не була створена

seconds = time.time()
CSV_FILES = ["all_data.csv"]
ths = open("importance_list_all_data.csv", "w")
folder("./feaure_pics/")
for j in CSV_FILES:
    df = pd.read_csv(j, usecols=MAIN_LABELS)
    df = df.fillna(0)
    attack_or_not = []
    for i in df["Label"]:
        if i == "BENIGN":
            attack_or_not.append(1)
        else:
            attack_or_not.append(0)
    df["Label"] = attack_or_not

    y = df["Label"].values
    del df["Label"]
    X = df.values

    forest = RandomForestRegressor(n_estimators=250, random_state=0)
    forest.fit(X, y)
    importances = forest.feature_importances_
    std = np.std([tree.feature_importances_ for tree in forest.estimators_],
                 axis=0)
    indices = np.argsort(importances)[::-1]
    refclasscol = list(df.columns.values)
    impor_bars = pd.DataFrame({'Features': refclasscol[0:20], 'importance': importances[0:20]})
    impor_bars = impor_bars.sort_values('importance', ascending=False).set_index('Features')
    plt.rcParams['figure.figsize'] = (10, 5)
    impor_bars.plot.bar()
    count = 0
    fea_ture = j[0:-4] + "=["
    for i in impor_bars.index:
        fea_ture = fea_ture + "\"" + str(i) + "\","
        count += 1
        if count == 5:
            fea_ture = fea_ture[0:-1] + "]"
```

```
        break
    print(j[0:-4], "importance list:")
    print(j[0:-4], "\n", impor_bars.head(20), "\n\n\n")
    print(fea_ture)
    plt.title(j[0:-4] + " Attack - Feature Importance")
    plt.ylabel('Importance')
    plt.savefig("./feaure_pics/" + j[0:-4] + ".pdf", bbox_inches='tight', orientation='portrait', format='pdf')
    ths.write(fea_ture)
    plt.tight_layout()
    print("----------------------------------------------------------------------------------------------\n\n\n\n")

print("Задачу виконано!")
print("Витрачений час = ", time.time() - seconds, "секунд")
ths.close()
```

6) Файл 05_1_Реалізація машинного навчання для файлів атак.ру

```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA  # імпорт необхідних функцій для ML
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
import matplotlib.pyplot as plt  # імпорт бібліотеки для візуалізації результатів
import numpy as np  # імпорт бібліотеки для роботи із масивами
import os  # імпорт бібліотеки із стандартними функціями системи
import pandas as pd  # імпорт бібліотеки для роботи із даними
import csv  # імпорт бібліотеки для роботи із .csv-файлами
import time  # імпорт бібліотеки, за допомогою якої буде вібуватися відлік часу на виконання коду
import warnings
from functions import folder  # імпорт функції для виклику помилки, якщо папка результатів не була створена
from variables import ALL_FEATURES  # імпорт константи

ML_LIST = {
    "Naive Bayes": GaussianNB(),
    "QDA": QDA(),
    "Random Forest": RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    "ID3": DecisionTreeClassifier(max_depth=5, criterion="entropy"),
    "AdaBoost": AdaBoostClassifier(),
    "MLP": MLPClassifier(hidden_layer_sizes=(13, 13, 13), max_iter=500),
    "Nearest Neighbors": KNeighborsClassifier(3)}

warnings.filterwarnings("ignore")
result = "./results/results_1.csv"
csv_files = os.listdir("attacks")
```

```
path = ".\\attacks\\"
repetition = 10
folder_name = "./results/"
folder(folder_name)
folder_name = "./results/result_graph_1/"
folder(folder_name)
seconds = time.time()
with open(result, "w", newline="", encoding="utf-8") as f:
    wrt = csv.writer(f)
    wrt.writerow(["File", "ML algorithm", "accuracy", "Precision", "Recall", "F1-score", "Time"])

for j in csv_files:
    print('%-17s %-17s  %-15s %-15s %-15s %-15s %-15s' % (
        "File", "ML algorithm", "accuracy", "Precision", "Recall", "F1-score", "Time"))  # print output
header
    a = []

    feature_list = list(ALL_FEATURES[j[0:-4]])
    df = pd.read_csv(path + j, usecols=feature_list)
    df = df.fillna(0)
    attack_or_not = []
    for i in df["Label"]:
        if i == "BENIGN":
            attack_or_not.append(1)
        else:
            attack_or_not.append(0)
    df["Label"] = attack_or_not

    y = df["Label"]
    del df["Label"]
    feature_list.remove('Label')
    X = df[feature_list]
    for ii in ML_LIST:
        precision = []
        recall = []
        f1 = []
        accuracy = []
        t_time = []
        for i in range(repetition):
            second = time.time()
            X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                test_size=0.20, random_state=repetition)
            clf = ML_LIST[ii]
            clf.fit(X_train, y_train)
            predict = clf.predict(X_test)
            f_1 = f1_score(y_test, predict, average='macro')
            pr = precision_score(y_test, predict, average='macro')
            rc = recall_score(y_test, predict, average='macro')
            precision.append(float(pr))
            recall.append(float(rc))
            f1.append(float(f_1))
            accuracy.append(clf.score(X_test, y_test))
```

```
            t_time.append(float((time.time() - second)))

        print('%-17s %-17s  %-15s %-15s %-15s %-15s %-15s' % (
            j[0:-4], ii, str(round(np.mean(accuracy), 2)), str(round(np.mean(precision), 2)),
            str(round(np.mean(recall), 2)), str(round(np.mean(f1), 2)), str(round(np.mean(t_time), 4))))
        with open(result, "a", newline="", encoding="utf-8") as f:
            wrt = csv.writer(f)
            for i in range(0, len(t_time)):
                wrt.writerow([j[0:-4], ii, accuracy[i], precision[i], recall[i], f1[i], t_time[i]])
        a.append(f1)

    ml = ["Naive Bayes", "QDA", "Random Forest", "ID3", "AdaBoost", "MLP", "Nearest Neighbors"]
    temp = 0
    fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(12, 6), sharey=True)
    for c in range(2):
        for b in range(4):
            axes[c, b].boxplot(a[temp])
            axes[c, b].set_title(str(j[0:-4]) + " - " + str(ml[temp]), fontsize=7)
            axes[c, b].set_ylabel("F measure")
            temp += 1
            if temp == 7:
                break
        if temp == 7:
            break

    plt.savefig(folder_name + j[0:-4] + ".pdf", bbox_inches='tight', orientation='portrait', format='pdf')
    plt.show()
    print(
        "\n----------------------------------------------------------------------------------------------------\n\n")

print("Задачу виконано.")
print("Витрачений час: = ", time.time() - seconds, "секунд")
```

7) Файл 05_2_Реалізація машинного навчання із 18 функціями.ру

```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA  # імпорт необхідних функцій для ML
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
import matplotlib.pyplot as plt  # імпорт бібліотеки для візуалізації результатів
import numpy as np  # імпорт бібліотеки для роботи із масивами
import os  # імпорт бібліотеки із стандартними функціями системи
import pandas as pd  # імпорт бібліотеки для роботи із даними
```

```
import csv  # імпорт бібліотеки для роботи із .csv-файлами
import time  # імпорт бібліотеки, за допомогою якої буде вібуватися відлік часу на виконання коду
import warnings
from functions import folder  # імпорт функції для виклику помилки, якщо папка результатів не була створена
from variables import FEATURES_18  # імпорт константи

warnings.filterwarnings("ignore")
result = "./results/results_2.csv"
csv_files = ["all_data.csv"]
path = ""
repetition = 10
folder_name = "./results/"
folder(folder_name)
folder_name = "./results/result_graph_2/"
folder(folder_name)
ML_LIST = {
    "Naive Bayes": GaussianNB(),
    "QDA": QDA(),
    "Random Forest": RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    "ID3": DecisionTreeClassifier(max_depth=5, criterion="entropy"),
    "AdaBoost": AdaBoostClassifier(),
    "MLP": MLPClassifier(hidden_layer_sizes=(13, 13, 13), max_iter=500),
    "Nearest Neighbors": KNeighborsClassifier(3)}

seconds = time.time()
with open(result, "w", newline="", encoding="utf-8") as f:
    wrt = csv.writer(f)
    wrt.writerow(["File", "ML algorithm", "accuracy", "Precision", "Recall", "F1-score", "Time"])
for j in csv_files:
    print('%-17s %-17s  %-15s %-15s %-15s %-15s %-15s' % (
        "Файл", "Алгоритм ML", "Accuracy", "Precision", "Recall", "F-measure", "Витрачений час"))
    feature_list = list(FEATURES_18[j[0:-4]])
    df = pd.read_csv(path + j, usecols=feature_list)
    df = df.fillna(0)
    attack_or_not = []
    for i in df["Label"]:
        if i == "BENIGN":
            attack_or_not.append(1)
        else:
            attack_or_not.append(0)
    df["Label"] = attack_or_not
    y = df["Label"]
    del df["Label"]
    feature_list.remove('Label')
    X = df[feature_list]
    for ii in ML_LIST:
        precision = []
        recall = []
        f1 = []
        accuracy = []
        t_time = []
```

```
    for i in range(repetition):
        second = time.time()
        X_train, X_test, y_train, y_test = train_test_split(X, y,
                                           test_size=0.20, random_state=repetition)
        clf = ML_LIST[ii]
        clf.fit(X_train, y_train)
        predict = clf.predict(X_test)
        f_1 = f1_score(y_test, predict, average='macro')
        pr = precision_score(y_test, predict, average='macro')
        rc = recall_score(y_test, predict, average='macro')
        precision.append(float(pr))
        recall.append(float(rc))
        f1.append(float(f_1))
        accuracy.append(clf.score(X_test, y_test))
        t_time.append(float((time.time() - second)))

    print('%-17s %-17s  %-15s %-15s %-15s %-15s %-15s' % (
        j[0:-4], ii, str(round(np.mean(accuracy), 2)), str(round(np.mean(precision), 2)),
        str(round(np.mean(recall), 2)), str(round(np.mean(f1), 2)), str(round(np.mean(t_time), 4))))
    with open(result, "a", newline="", encoding="utf-8") as f:
        wrt = csv.writer(f)
        for i in range(0, len(t_time)):
            wrt.writerow([j[0:-4], ii, accuracy[i], precision[i], recall[i], f1[i], t_time[i]])

    plt.boxplot(f1)
    plt.title("All Dataset - " + str(ii))
    plt.ylabel('F-measure')
    plt.savefig(folder_name + j[0:-4] + str(ii) + ".pdf", bbox_inches='tight', orientation='portrait',
format='pdf')
    plt.show()

print("Задачу виконано.")
print("Витрачений час = ", time.time() - seconds, "секунд")
```

8) Файл 05_3_Реалізація машинного навчання із 7 функціями.py

```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA  # імпорт
необхідних функцій для ML
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
import matplotlib.pyplot as plt  # імпорт бібліотеки для візуалізації результатів
import numpy as np  # імпорт бібліотеки для роботи із масивами
import os  # імпорт бібліотеки із стандартними функціями системи
import pandas as pd  # імпорт бібліотеки для роботи із даними
```

```
import csv  # імпорт бібліотеки для роботи із .csv-файлами
import time  # імпорт бібліотеки, за допомогою якої буде вібуватися відлік часу на виконання коду
import warnings
from functions import folder  # імпорт функції для виклику помилки, якщо папка результатів не була створена
from variables import FEATURES_7  # імпорт константи

ML_LIST = {
    "Naive Bayes": GaussianNB(),
    "QDA": QDA(),
    "Random Forest": RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    "ID3": DecisionTreeClassifier(max_depth=5, criterion="entropy"),
    "AdaBoost": AdaBoostClassifier(),
    "MLP": MLPClassifier(hidden_layer_sizes=(13, 13, 13), max_iter=500),
    "Nearest Neighbors": KNeighborsClassifier(3)}

warnings.filterwarnings("ignore")
result = "./results/results_3.csv"
csv_files = ["all_data.csv"]
path = ""
repetition = 10
folder_name = "./results/"
folder(folder_name)
folder_name = "./results/result_graph_3/"
folder(folder_name)
seconds = time.time()
with open(result, "w", newline="", encoding="utf-8") as f:
    wrt = csv.writer(f)
    wrt.writerow(["File", "ML algorithm", "accuracy", "Precision", "Recall", "F1-score", "Time"])

for j in csv_files:
    print('%-17s %-17s  %-15s %-15s %-15s %-15s %-15s' % (
        "Файл", "Алгоритми ML", "Accuracy", "Precision", "Recall", "F-measure", "Time"))  # print output header
    feature_list = list(FEATURES_7[j[0:-4]])
    df = pd.read_csv(path + j, usecols=feature_list)
    df = df.fillna(0)
    attack_or_not = []
    for i in df["Label"]:
        if i == "BENIGN":
            attack_or_not.append(1)
        else:
            attack_or_not.append(0)
    df["Label"] = attack_or_not
    y = df["Label"]
    del df["Label"]
    feature_list.remove('Label')
    X = df[feature_list]

    for ii in ML_LIST:
        precision = []
        recall = []
```

```
    f1 = []
    accuracy = []
    t_time = []
    for i in range(repetition):
        second = time.time()
        X_train, X_test, y_train, y_test = train_test_split(X, y,
                                           test_size=0.20, random_state=repetition)
        clf = ML_LIST[ii]
        clf.fit(X_train, y_train)
        predict = clf.predict(X_test)
        f_1 = f1_score(y_test, predict, average='macro')
        pr = precision_score(y_test, predict, average='macro')
        rc = recall_score(y_test, predict, average='macro')
        precision.append(float(pr))
        recall.append(float(rc))
        f1.append(float(f_1))
        accuracy.append(clf.score(X_test, y_test))
        t_time.append(float((time.time() - second)))

    print('%-17s %-17s  %-15s %-15s %-15s %-15s %-15s' % (
        j[0:-4], ii, str(round(np.mean(accuracy), 2)), str(round(np.mean(precision), 2)),
        str(round(np.mean(recall), 2)), str(round(np.mean(f1), 2)), str(round(np.mean(t_time), 4))))
    with open(result, "a", newline="", encoding="utf-8") as f:
        wrt = csv.writer(f)
        for i in range(0, len(t_time)):
            wrt.writerow([j[0:-4], ii, accuracy[i], precision[i], recall[i], f1[i], t_time[i]])

    plt.boxplot(f1)
    plt.title("All Dataset - " + str(ii))
    plt.ylabel('F-measure')
    plt.savefig(folder_name + j[0:-4] + str(ii) + ".pdf", bbox_inches='tight', orientation='portrait',
format='pdf')
    plt.show()

print("Задачу виконано.")
print("Витрачений час = ", time.time() - seconds, "секунд.")
```

9) Файл 05_4_Порівняння показників машинного навчання.py

```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA  # імпорт
необхідних функцій для ML
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import f1_score
import pandas as pd  # імпорт бібліотеки для роботи із даними
import warnings
import time  # імпорт бібліотеки, за допомогою якої буде вібуватися відлік часу на виконання коду
from variables import FEATURES

seconds = time.time()
```

```
warnings.filterwarnings("ignore")
df=pd.read_csv('all_data.csv',usecols=FEATURES)
print ('%-17s %-17s ' % ("Feature Number","Feature"))
for i in range(len(FEATURES)-1):
   print ('%-17s %-17s' % (i+1,FEATURES[i]))

print ('\n\n\n')
attack_or_not=[]
for i in df.iloc[:,-1]:
   if i =="BENIGN":
      attack_or_not.append(1)
   else:
      attack_or_not.append(0)
df.iloc[:,-1]=attack_or_not
y = df.iloc[:, -1].values
my_list=[]
least=0
ml_list={
"Naive Bayes":GaussianNB(),
"QDA":QDA(),
"MLP":MLPClassifier(hidden_layer_sizes=(13,13,13),max_iter=500)}
FEATURES.pop()
print ('%-17s %-30s %-10s  %-10s %-15s ' % ("Алгоритм ML","Ознака","F-measure","Accuracy",
"Список ознак"))
for j in ml_list:
   my_list=[]
   for i in FEATURES:
      my_list.append(i)
      X = df.loc[:, my_list].values # data

      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

      clf = ml_list[j]
      clf.fit(X_train, y_train)
      predict =clf.predict(X_test)
      f1=clf.score(X_test, y_test)
      result=f1_score(y_test, predict, average='macro')
      accuracy=round(clf.score(X_test, y_test),2)
      temp="["
      for ii in my_list:
         temp+=str(my_list.index(ii)+1)+", "
      if result>=least:
         least=result
         print ('%-17s %-30s %-10s  %-10s %-15s %-15s ' % (j,i,result,accuracy ,temp, "------> New
feature found!!!"))
      else:
         my_list.remove(my_list[len(my_list)-1])
         print ('%-17s %-30s %-10s  %-10s %-15s ' % (j,i,result,accuracy ,temp))
   print("F1=" ,least,j," The most efficient feature list =",my_list,"\n\n")

print("Задачу виконано.")
print("Витрачений час = ",time.time()- seconds ,"секунд")
```