МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Інтелектуальний програмний модуль виявлення аномалій на основі аналізу мережевого трафіку

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма «**Аналітика даних»**

Освітній рівень: бакалавр

Виконала: студентка 4 курсу, групи АнД-41
Кліменкова Н.А.
_____
(прізвище та ініціали)

Керівник: Іларіонов О.Є.
_____
(прізвище та ініціали)
к.т.н., доцент
_____
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 11 від 06.06.2022 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2022

Анотація

Кліменкова Ніна Анатоліївна виконала випускну кваліфікаційну роботу на тему «Інтелектуальний програмний модуль виявлення аномалій на основі аналізу мережевого трафіку» за спеціальністю 122 – «Комп'ютерні науки», освітньою програмою «Аналітика даних».

У випускній кваліфікаційній роботі досліджено питання аномалій у мережевому трафіку; створено сервіс, для виявлення зловмисних з'єднань із використання наступних класифікаційних методів: Decision Tree, KNeighbors, LinearSVC.

Ключові слова: аномалія, мережа, машинне навчання, система виявлення вторгень, Decision Tree, KNeighbors, LinearSVC.

Abstract

The bachelor's project " Intelligent software module for detecting anomalies based on network traffic analysis" has been completed by Nina Klimenkova speciality 122 – "Computer Science".

The issue of anomalies in network traffic is investigated in the final qualification work; created a service to detect malicious connections using the following classification methods: Decision Tree, KNeighbors, LinearSVC.

Keywords: anomaly, network, machine learning, intrusion detection system, Decision Tree, KNeighbors, LinearSVC.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.
_____
"___" _____2022 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

_____Кліменковій Ніні Анатоліївні_____

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи): Інтелектуальний програмний модуль виявлення аномалій на основі аналізу мережевого трафіку

затверджена протоколом засідання кафедри від «23 » грудня 2021 р. №4

2. Термін здачі студентом закінченого проекту (роботи) 31 травня 2022 року

3. Вихідні дані до проекту (роботи): Застосунок, що має можливість відрізняти зловмисні з'єднання від звичайних та класифікувати їх відповідним чином

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
   4.1 Аналітичний огляд інтелектуального програмного модулю виявлення аномалій
   4.2 Предметно – технологічні особливості реалізації інтелектуального програмного модулю виявлення аномалій
   4.3 Проектно – технічна реалізація інтелектуального програмного модулю виявлення аномалій

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)
   5.1 Аналіз актуальності, цілі та постановка завдання розробки програмного застосунку: 2-3 слайди
   5.2 Основні вимоги до застосунку: 1-2 слайди
   5.3 Проектно-технічна реалізація програмного модулю: 3-5 слайдів
   5.4 Технологічні особливості реалізації програмного модулю з виявленням аномалій: 4-5 слайдів
   5.5 Проведення тестування розробленого програмного модулю з виявленням аномалій: 4-5 слайдів

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 15 лютого 2022 року

Керівник _____ / О.Є. Іларіонов /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Н.А. Кліменкова /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1	Обговорення з керівником постановки завдання та змісту пояснювальної записки	25.01.2022 - 25.02.2022	
2	Постановка задачі, формалізація задачі, вибір методів та засобів реалізації поставленої задачі, аналіз літературних джерел	26.02.2022 - 10.03.2022	
3	Проектно-технічна реалізація інтелектуального застосунку виявлення аномалій	11.03.2022 - 20.04.2022	
4	Тестування програмного модулю виявлення аномалій	21.04.2022 - 30.04.2022	
5	Оформлення пояснювальної записки, підготовка презентації	01.05.2022 - 21.05.2022	

Студент-дипломник _____ / Н.А. Кліменкова /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / О.Є. Іларіонов /
(підпис) (ПІБ)

Зміст

Перелік умовних позначень та скорочень

ПЗ - програмне забезпечення

СВВ - система виявлення вторгнень

DDoS - Denial of Service «відмова в обслуговуванні»

IDS - система виявлення вторгнень

ОС - операційна система

СВА – система виявлення аномалій

Вступ

Найбільш поширений спосіб, за допомогою якого зловмисники можуть проникнути в інфраструктуру – це шлях через мережу. Забезпечення безпеки мережі представляє собою масштабну практичну область діяльності по захисту комп’ютерних мереж. Модель захисту мереж складна через широкий діапазон атак та потенційних загроз. Як і при захисті будь-якої складної системи, спеціалісти з кібербезпеки мають протистояти хакерам у багатьох направленнях і не покладатися на надійність якогось одного елемента системи.

Як зазначили у статті від DBIR: «В середньому, виявлення вразливості займає близько десяти днів» [1]. Але після того, як зловмисник отримав доступ до системи, шкода наноситься протягом декількох днів, або навіть швидше. Тому виявлення аномального з’єднання означає вчасне реагування на небезпеку.

Завдання розробки ефективного інструменту для виявлення аномалій у мережевому трафіку вкрай важливе, так як вчасне реагування може попередити негативні наслідки, через які страждають тисячі компаній кожного дня. Під такими наслідками слід розуміти оприлюднення приватних даних, значні економічні втрати, збій у роботі критичної інфраструктури тощо. Вони несуть у собі не тільки матеріальні збитки, а також значною мірою загрожують безпеці та життю населення.

Основна мета розробки інтелектуального модуля у даній роботі є виявлення зловмисних з’єднань у мережі за допомогою методів машинного навчання та технологій штучного інтелекту.

В процесі розробки модулю, будуть досліджені підходи до виявлення аномалій та проаналізована їх ефективність для роботи з локальною мережею. Також буде розглянуте поняття система виявлення вторгнень(СВВ), як ефективний спосіб боротьби з кібератаками різних видів та цілей.

Таким чином, дана робота буде корисна для будь-якого підприємства, дані якого передаються через мережу та яке дбає про безпеку даних та швидкість реагування на небезпеку.

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ІНТЕЛЕКТУАЛЬНОГО ПРОГРАМНОГО МОДУЛЮ ВИЯАЛЕННЯ АНОМАЛІЙ

1.1 Огляд мережевого трафіку та поняття аномалії

У 2020 році в Україні зафіксували близько 1 мільйон випадків кіберзагроз. Серед них найпоширеніші - мережеві атаки, спроби мережевого сканування, спроби WEB-атак, фішинг та DDoS-атаки, повідомляє Рада національної безпеки і оборони [2]. З роками кількість атак тільки збільшується, а рівень шкоди від них набуває нових розмахів. На жаль, така ситуація має місце у всьому світі і несе у собі небезпеку для даних мільярдів користувачів.

Вищезазначені загрози у мережі містять певні відхилення від звичайного з’єднання, тому їх можна охарактеризувати аномальними з’єднаннями у мережі. З точки зору забезпечення безпеки мережі і хосту, виявлення аномалій означає виявлення випадків непередбачуваних нападів або непомічених раніше вразливостей. Не залежно від типу і мети кібератаки, час реагування відіграє ключову роль.

Зазвичай система виявлення атак на мережу розпізнає трафік хакера за допомогою зіставлення його з шаблонами у раніше підготовленому наборі даних, така процедура має назву «перевіркою сигнатур». Дані системи використовують принцип пакету антивірусних програм. Вони намагаються встановити відповідність між пакетами, що надійшли у мережу та пакетами сигнатур відомих атак, таким чином ефективно виявляють вже відомі сигнатури. В свою чергу, СВВ на основі методу виявлення аномалій проводить дослідження пакетів для класифікації, щоб встановити несхожість між типовою і нетиповою поведінкою.

Даючи визначення мережевому трафіку можна сказати, що це складний динамічний процес і являє собою суперпозицію багатьох потоків з множинними взаємозв'язками, які генеруються різними потоками. Мережевий трафік, або

трафік даних, розбивається на пакети даних і пересилається по мережі, перш ніж повторно збирається пристроєм або комп’ютером-отримувачем. Проведення аналізу трафіку та виявлення певних відхилень є важливим етапом для розпізнавання загрози.

За результатами останніх досліджень на експериментальних даних можна сказати, що трафік сучасних комп'ютерних мереж має особливу структуру, яка проявляє ефект самоподібності. Цей ефект проявляється в тому, що статистичні характеристики трафіку масштабуються при усередненні значень взятих за різні проміжки часу. Іншими словами, під самоподібністю мається на увазі повторюваність розподілу навантаження в часі при різних масштабах (рисунок 1.1) [3].

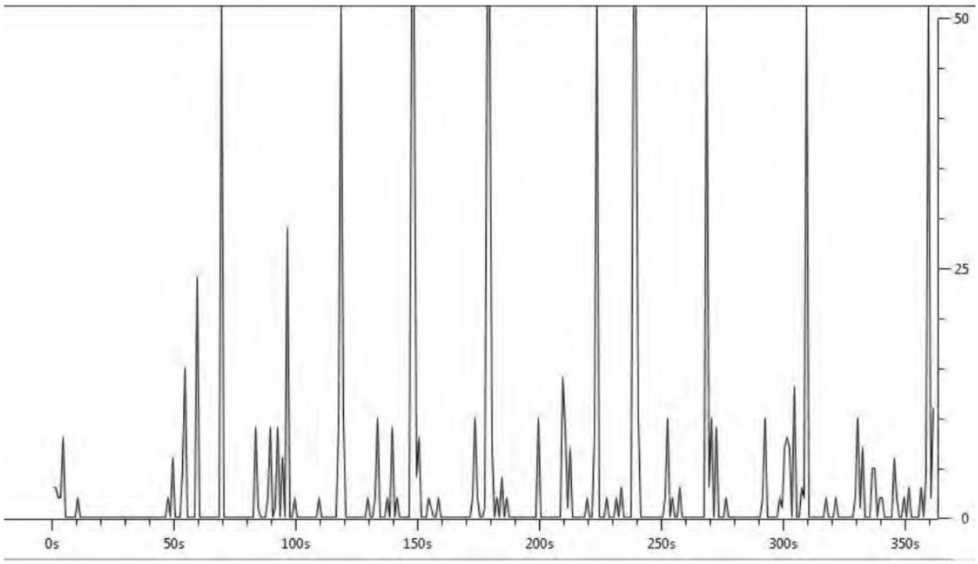

Рисунок 1.1 Звичайне навантаження трафіку при бездротовому з’єднанні [3]

Даний ефект пов'язаний з властивостями TCP/IP протоколу передачі даних, який сьогодні використовується в комп'ютерних мережах. Трафік в мережах такого виду має всплесковий характер, як видно з рисунка 1.1, що підвищує імовірність перенавантажень у вузлах мережі, які ведуть до переповнення буферів і викликають втрати або затримки. Дослідження самоподібності наголошують, що дане явище значно погіршує якість трафіку через мережу.

Важливість виявлення аномалій не обмежується сферою кібербезпеки. Виявлення аномалій можна визначити як будь-який метод пошуку подій, які не відповідають очікуванням і прогнозам. Наприклад, у системах, які вимагають високий рівень надійності, можливо використати методи виявлення аномалій для знаходження різних ознак аварійної ситуації, що дасть змогу операторам прийняти превентивні міри. Іншим прикладом сфери застосування даної системи може стати шахрайство в області фінансів. Серед надзвичайно великої кількості законних транзакцій за допомогою вивчення шаблонів звичайних подій і виявленню відхилень від них можна попередити шкоду від зловмисників [4].

У зв’язку з цим розробка ефективних способів розпізнавання зловмисних з’єднань виявлення їх шаблонів для вчасного реагування та запобігання негативним наслідкам для системи з’єднань у мережі стає однією з найактуальніших задач на сьогодні.

1.2 Огляд методів виявлення аномалій

Масові кібератаки ініціюють створення спеціальних засобів, технічних рішень та систем протидії. Для виявлення вторгнень у мережу застосовуються сучасні засоби, методи, моделі, програмне забезпечення (ПЗ), а також складні технічні рішення для систем виявлення та запобігання атак, які залишаються ефективними при появі нових або модифікованих видів кібернебезпек. Якщо ж проаналізувати практичний досвід, у випадку появи нових аномалій або вірусів, вищезазначені засоби не завжди виявляються ефективними, а також вимагають тривалих часових ресурсів для їх адаптації та реагування на небезпеку. Через це СВВ мають постійно досліджуватись і вдосконалюватись для забезпечення їх ефективного функціонування.

Таким чином, проведення аналізу технічних особливостей, спеціальних засобів та ПЗ виявлення кіберзагроз, зловживань та аномалій у інформаційній системі для їх використання при виборі і розробці СВВ, а також визначення

найбільш ефективних відповідних механізмів захисту сьогодні є актуальним завданням.

На сьогодні існує велика кількість підходів до даного питання. Одним з них стали системи виявлення вторгнень, які існують з 1986 року і широко застосовуються у системах забезпечення безпеки. Як відомо, навіть у наші дні застосування евристик, порогових значень та простих статистичних показників залишаються надійним способом виявлення аномалій. Даний метод використовує простий статистичний збір даних для того, щоб запобігти необхідності встановлення порогового значення вручну, але зберігає у собі характеристики евристичного методу і залишає необхідність прийняття рішення по випадковим параметрам. Необхідність встановлення порогового значення вручну стало поштовхом для розглядання інших підходів таких, як машинне навчання.

Машинне навчання - це процес використання хронологічних (історичних) даних для створення алгоритму прогнозування на даних, які будуть зібрані у майбутньому. Прогнозування, в свою чергу, досить логічно-зрозумілий спосіб виявлення аномалій, так як навчання відбувається на раніше добутих даних, а потім формується прогноз майбутніх подій на основі нових даних. Таким чином, аномаліями можна вважати будь-які значні розбіжності між прогнозом та спостереженнями. Також, одним із завдань машинного навчання є задача класифікації, що означає розподіл класу, до якого слід віднести нові дані, що поступили. Класифікація може бути як бінарною, так і мультикласовою. Мультикласова класифікація необхідна у випадку, якщо потрібно встановити чи є зловмисне програмне забезпечення програмою-вимагачем, кейлогером або троянською програмою. Крім того, машинне навчання можна використовувати для вирішення задачі регресійного аналізу, в якій відбувається спроба передбачити значення змінної у чисельному вигляді. Задачі регресійного аналізу, для яких вхідні дані пов’язані з інтервалом часу, іноді називають аналізом часового ряду, або динамічного ряду. Виявлення аномалій у свою чергу являє собою більш високий рівень регресійного аналізу - це задача виявлення

значимого відхилення від спостережуваного значення до очікуваної величини, яке може підставою вважати, що відбувається вторгнення. Також машинне навчання використовують для задачі кластеризації, що несе у собі розподіл схожих за ознаками елементів у групи(кластери). Варто зауважити, що задачі класифікації і регресії є прикладом навчання з учителем, у той час як задача кластеризації є прикладом навчання без вчителя.

1.3 Конструювання ознак для виявлення аномалій

За словами розробників та спеціалістів з кібербезпеки, СВВ давала уявлення про трафік в окремих частинах мережі і про всі нетипові прояви на підставі аналізу журнальних файлів на хості. Дана інформація дозволяла зрозуміти, чи піддавалася мережа атакам злочинців для отримання доступу до важливих мережевих ресурсів. Такі системи встановлювалися в ключових вузлах мережі - на рівні міжмережевих екранів, комутаторів, маршрутизаторів, серверів Web, баз даних та інших обслуговуючих пристроїв всередині підприємства.

Однак початкові СВВ видавали велику кількість повідомлень і таким чином створювали величезні обсяги інформації про проходження трафіку через мережу і через системи хоста, нерідко посилаючи помилкові сигнали тривоги у великій кількості. На жаль, багато системних адміністраторів були не в змозі систематизувати таку значну кількість даних і мінімізували функції таких систем, або зовсім відключали їх. Тому для уникнення навантаження великим об’ємом даних на систему, варто розглянути можливі аномалії у трьох просторах: хост, мережа, веб-додаток. Це необхідно для того, щоб усвідомити особливості кожного із даних просторів.

Як і в будь-якій іншій задачі машинного навчання вибір правильних ознак для аномалій має першочергову важливість. Деякі алгоритми потребують вводу у формі часового послідовного потоку даних, але на практиці частіше потребується генерація спеціально сформованих потоків даних, до яких будуть застосовуватись алгоритми виявлення аномалій. Як вже було зазначено, у рамках

дипломної роботи будуть розглянуті три області: виявлення вторгнення на хост, виявлення вторгнення на мережу та виявлення вторгнення на веб-додаток. Між даними трьома областями є вагома різниця, і до кожної з них необхідний набір специфічних умов.

1.3.1 Виявлення вторгнень на хост

При розробці агенту, який виявляє вторгнення на хости (сервер, ноутбук, настільна система), потребується генератор особистих спеціалізованих метрик. Виникає необхідність в установці зав'язків між сигналами, що збираються із різних ресурсів. Значимість різноманітних метрик може змінюватись у широких діапазонах в залежності від моделі потенційної небезпеки. Набір таких системних метрик можна формувати різноманітними способами, існує велика кількість інструментів та середовищ для задач такого виду.

Шкідливе програмне забезпечення є найбільш поширеною загрозою для хостів у більшості програмних середовищ. Припускаючи, що шкідлива програма впливає на операції системного рівня, таку програму можна виявити за допомогою збору сигналів про операції системного рівня і пошуку признаків загроз у таких даних. Нижче наведено список деяких спільних сигналів, які можна зібрати для вирішення подібних задач [4]:

- активні процеси
- активні/нові облікові записи користувачів
- модулі ядра
- мережеві з'єднання
- зміна і механізмі системного планувальника
- операції активізації
- каталоги для файлів
- розширення браузера

1.3.2 Виявлення вторгнень у мережу

Майже всі форми вторгнень на хост провокують обмін інформацією з навколишньою середою. Більша частина вразливостей використовується с метою викрадення важливих даних з цільового хоста, тому потрібно сконцентрувати увагу на виявлені вторгнень на мережеве середовище. При атаці ботнетів видалені сервера оперативного управління і контроля обмінюються інформацією із комп'ютерами-зомбі, що були зламані для передачі різноманітних інструкцій. Таким чином хакери можуть отримати віддалений доступ до комп'ютера через слабке місце – сервіс, який дозволяє використовувати командним рядком та правами доступа суперкористувача. Рекламне ПЗ потребує обміну інформацією з зовнішніми серверами для примусового завантаження рекламного контенту на цільовий комп'ютер. Шпигунське ПЗ частіше всього передає результати прихованого моніторингу мережі на зовнішній приймаючий сервер. Така програмна екосистема виявлення вторгнень у мережу пропонує велику кількість утиліт і додатків, які допомагають у зборі сигналів мережевого трафіку будь яких типів: від простих утиліт моніторингу протоколів таких як tcpdump [5] до більш складних інструментів сніфінгу, таких як Zeek [6]. Робота інструментальних засобів виявлення вторгнень у мережу заснована на концепції спостереження за трафіком, зо передається між хостами, як зазначалося раніше. Як і при виявлені вторгнень на хости, атаки можна ідентифікувати по збігу трафіка з відомою сигнатурою (зразком) шкідливого трафіка або за допомогою методу виявлення аномалій, що порівнює трафік з раніше встановленими базовими або опорними лініями. Для даної проблеми вже були створені деякі методи боротьбі такі як Snort та процес повної інспекції пакетів [7].

1.3.3 Виявлення вторгнень у веб-додаток

Інспекція HTTP-сервера може надати належний рівень інформації і являє собою більш очевидний спосіб отримання ознак, що виникають під час взаємодії користувача з веб-додатком. Стандартні веб-сервери, такі як Apache, генерують журнальні записи у спільному форматі, який також називають журналом доступу. Навіть при досить обмеженому рівні видимості вмісту запитів у стандартних файлів журналів HTTP-сервера в них все ж є деякі корисні признаки, які можна виявити [4]:

- статистичні дані про доступ на рівні IP
- спотворення строки URL
- декодування елементів URL і HTML
- незвичні шаблони посилань
- послідовні спроби доступу к кінцевому пункту

1.4 Вимоги до інтелектуального програмного модуля

Для грамотного підбору методів виявлення аномалій, необхідно визначити усі цілі оптимальної системи виявлення аномалій.

1.4.1 Низький рівень хибнопозитивних і хибнонегативних результатів

Термін «аномалія» передбачає подію, яка вирізняється на фоні інших подій. Складність полягає у тому, що аномалій виникають у такій формі, що їх легко сплутати з хибними сигналами або ж навпаки пропустити справжній сигнал про небезпеку. Хибнопозитивні випадки виникають, коли система помилково вважає звичайну подію аномальною. Хибнопозитивні результати можуть здаватися не дуже небезпечними, але кожне застереження пов’язано з витратами. Кожний хибний сигнал витрачає час аналітика, який досліджує такі випадки. Тому високий рівень хибних сигналів може призвести до порушення цілісності системи, а спеціалісти не зможуть вирізнити справжні аномалії, які потребую вчасного реагування.

1.4.2 Простота конфігурації, налаштування та супроводу

Через неправильну конфігурацію системи на основі порогових значень є ризик отримати великий відсоток хибнопозитивних показників. Також невелика кількість параметрів та простота початкової конфігурації впливає на зручність і ефективність роботи системи. Машинно навчений детектор виявлення аномалій може почати генерувати хибні сигнали з високою частотою. Оптимальний детектор аномалій має представляти чітку картину того, як зміна параметрів системи напряму призводить до зміни якості, кількості та природи вихідних сигналів.

1.4.3 Адаптованість до зміни трендів у даних

Сезонність – це властивість даних показувати постійно однакові шаблони даних, які з’являються через природні цикли активності користувачів. Алгоритми виявлення аномалій, які не мають механізму обробки сезонних коливань, будуть видавати велику кількість хибнопозитивних реагувань у періоди сезонних трендів. Розроблена система має враховувати сезонність, тобто враховувати сформовані шаблони даних, особливо у веб-трафіку, де дане явище розповсюджене.

1.4.4 Ефективна робота з наборами даних різного походження

Так як далеко не всі набори даних мають нормальний розподіл, велику роль у виявлені аномалій відіграє оцінка щільності для моделювання нормальності у даних. Система виявлення аномалій має ефективно працювати з наборами даних, які мають різні властивості та розподіли.

1.4.5 Ефективність використання ресурсів і можливість використання реального часу

Саме у сфері забезпечення безпеки системи, швидкість реакції і швидкість виявлення аномалії відіграють одну з ключових ролей, так як від цього залежить рівень шкоди, який може нанести атака. Оператори мають бути застережені про потенційні вразливості у випадку аварійних ситуацій у системі протягом декількох хвилин з моменту виникнення підозрілого сигналу. Так як на рахунку кожна хвилина, СВА повинна працювати у неперервному потоковому режимі, збирати та оброблювати дані та генерувати виводи з мінімальною затримкою.

1.4.6 Сигнали тривоги

Є можливість аналізувати сигнали тривоги на основі статистичного порогового значення. Звичайна повторна поява події за правилами дає змогу зрозуміти, які умови стали причиною сигналу тривоги. Аналіз сигналу тривоги є важливим показником для навчання детектора виявлення аномалій, так як це дає змогу знизити рівень хибнопозитивних і хибнонегативних результатів. У випадку коли сигнал тривоги необхідно аналізувати з умовою жорстких часових обмежень, наявність чітких пояснень може спростити процес прийняття рішень аналітиком, який реагує на сигнал аномалії.

1.5 Постановка задачі для розробки програмного модуля

Досліджуючи проблематику даної теми, можна сказати, що основна складність полягає у поганій структурованості даних, різноманітності та їх надзвичайних об’ємах, тому для визначення кореляції або певних шаблонів зловмисних вторгнень необхідно провести специфічну підготовку даних. Також генерація надійного і повного набору ознак надзвичайно важлива для процесу виявлення аномалій. Важливим акцентом даної задачі є врахування часових

характеристик роботи кожного методу, так як швидкість реагування на небезпеку є ключовим моментом для підприємств на сьогодні.

У якості досліджуваних даних був використаний набір даних, підготовлений MIT Lincoln Labs, які розробили імітаційну модель локальної мережі та збирали послідовності пакетів TCP протягом дев’яти тижнів. Дані використовуються здебільшого у навчальних цілях, але наближують задачу до реального практичного вирішення.

У ході аналізу існуючих методів та підходів до даної задачі, було вирішено створити інтелектуальний програмний модуль, який буде працювати на основі підходу навчання з вчителем та буде здатен розрізняти зловмисне з’єднання від звичайного та відносити його до відповідного класу шляхом моделювання правил аномальної роботи системи.

Висновки за розділом 1

Роблячи висновок можна сказати, що за останні роки технологія виявлення вторгнень зазнала швидкого розвитку, багато важливих питань досі залишаються відкритими. По-перше, системи виявлення аномалій повинні стати більш ефективними, навчившись виявляти широкий діапазон атак з мінімальною кількістю помилкових тривог та за мінімальний проміжок часу. По-друге, методи виявлення вторгнень повинні розвиватися враховуючи швидкості, зростання розміру і динамізму сучасних мереж. Навіть через те, що захист мереж стає все надійніше, виявлення вторгнень завжди залишиться невід'ємною частиною будь-якої серйозної системи безпеки.

РОЗДІЛ 2 ПРЕДМЕТНО – ТЕХНОЛОГІЧНІ ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ІНТЕЛЕКТУАЛЬНОГО ПРОГРАМНОГО МОДУЛЮ ВИЯВЛЕННЯ АНОМАЛІЙ

Для детального розгляду архітектури інтелектуального модулю та усвідомлення всіх тонкощів реалізації варто розглянути поняття система виявлення вторгнення більш глибоко. Тому у даному розділі, буде приділена увага СВВ, її особливостям, перевагам та недолікам. Також будуть описані обрані методи навчання з вчителем, які використовувались для розробки класифікатора. Буде розглянуто архітектуру та специфіку інтелектуального програмного модуля виявлення аномалій.

2.1 Опис інтелектуального підходу до виявлення аномалій у мережевому трафіку

Системи виявлення вторгнень забезпечують виявлення загрози і запобігають її подальшому розвитку. В свою чергу, система виявлення атак являє собою програмний засіб, який призначений для виявлення підтвердження несанкціонованого доступу в комп'ютерну систему та мережу.

Досить важко зустріти СВВ, в якій був би реалізований тільки один підхід до аналізу даних, так як сучасні системи використовують декілька технологій одночасно.

Використання СВВ допомагає досягнути таких цілей:

• виявити вторгнення або мережеві атаки;

• забезпечити належний контроль якості адміністрування, особливо у великих і складних мережах;

• спрогнозувати можливі майбутні атаки і виявити вразливості для запобігання їх подальшого розвитку;

• отримати корисну інформацію про проникнення, для відновлення і

налаштування конфігурації мережі;

• визначити розташування джерела атаки по відношенню до локальної мережі (зовнішні або внутрішні атаки).

Як і будь-яка система, СВВ складається з набору компонентів, пов'язаних між собою. Для кращого розуміння, за яку саме частину відповідає розроблений програмний модуль, слід розібрати основні складові частини СВВ. На рисунку 2.1 зображена архітектура системи виявлення вторгнень.

Основними компонентами СВВ є сенсорна підсистема, підсистема аналізу, сховище, консоль управління та модуль реагування.

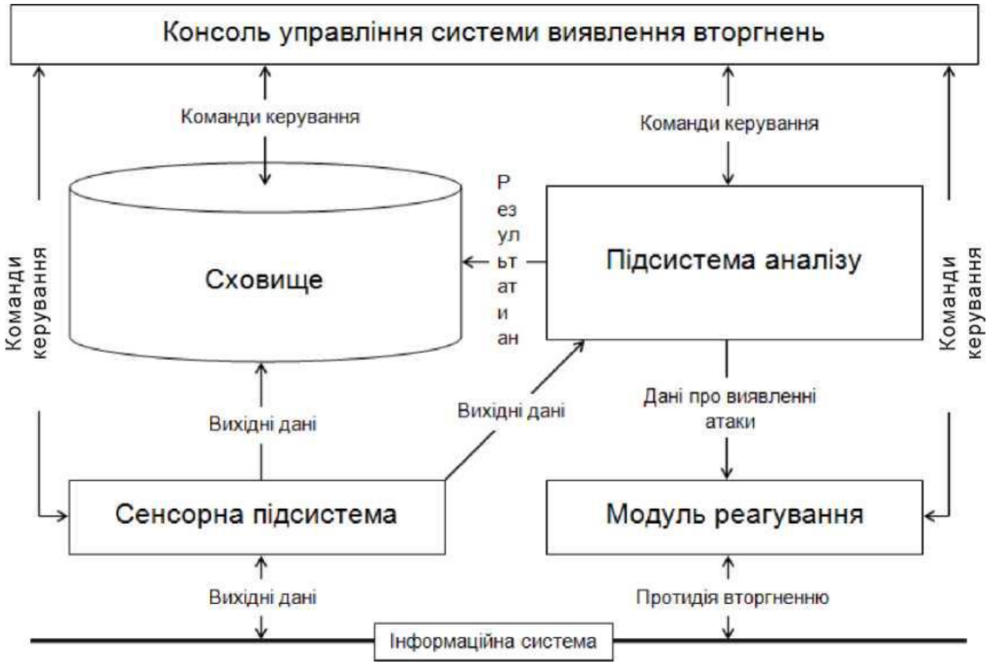

Рисунок 2.1 Архітектура системи виявлення вторгнень [8]

Варто зауважити, що дана робота сконцентрована саме на блоці «підсистема аналізу». Таким чином, наведена схема вдало ілюструє, яке місце займає розроблений програмний модуль у загальній системі виявлення вторгнень, а також наглядно демонструє взаємодію з іншими процесами, які мають місце.

Ідея методів виявлення аномалій, полягає в тому, щоб розпізнати, чи є процес, що викликав зміни в роботі системи, діями хакера. Метод виявлення аномалій або поведінковий метод базується не на моделях інформаційних атак,

а на моделях штатного функціонування (поведінки) ІС. Принцип роботи будь-якого з таких методів полягає в виявленні невідповідності між поточним режимом роботи ІС і режимом роботи, що відповідає штатної моделі даного методу. Будь-яка невідповідність розглядається як інформаційна атака.

Прикладом такої невідповідності може бути, якщо система виявлення атак зафіксує вхід співробітника компанії у мережу в суботу о 12.30, це може свідчити про те, що пароль користувача вкрадений або підібраний і зловмисник використовує його для несанкціонованого проникнення.

Перевага методів даного типу - можливість виявлення нових атак без модифікації або поновлення параметрів моделі. На жаль, створити точну модель штатного режиму функціонування ІС дуже складно.

Найбільш поширені серед поведінкових методів ті, що базуються на статистиці. Статистичні моделі визначають статистичні показники, що характеризують параметри звичайної поведінки системи. Якщо ж спостерігається певне відхилення даних параметрів від заданих значень, то фіксується факт виявлення атаки. Як правило, в якості таких параметрів можуть виступати: навантаження на канали зв'язку, рівень завантаження процесора, штатний час роботи користувачів системи та кількість звернень до мережевих ресурсів.

У свою чергу, методи виявлення аномалій спрямовані на виявлення невідомих атак і вторгнень. Для системи, що захищається на основі сукупності параметрів оцінки формується образ нормального функціонування. В сучасних СВВ виділяють головний спосіб побудови образу, а саме накопичення найбільш характерної статистичної інформації для кожного параметра.

Неважко помітити, що у виявленні значну роль відіграє безліч параметрів оцінки, саме тому вибір оптимальної безлічі параметрів оцінки є вкрай важливим. Іншим, не менш важливим завданням є визначення загального показника аномальності. Складність полягає в тому, що ця величина повинна характеризувати загальний стан «аномальності» в системі, що захищається [9].

Проаналізувавши існуючі методи та підходи, було вирішено зупинитись на трьох методах, які на маю думку вдаліше підходять для поставленої задачі.

Перший метод дерева рішень. Як описує А. Олійник у навчальному посібнику: «Дерева рішень або дерева вирішальних правил – це один з методів автоматичного аналізу даних, який задає спосіб подання правил виду «якщо – то» в ієрархічній послідовній структурі, де кожному об’єкту відповідає лише один вузол, що дає рішення» [10]. Область використання даного методу можна об'єднати в три класи:

- опис даних: застосування методу дозволяє зберігати інформацію про набір даних в зручній для обробки формі, яка містить в собі точні описи об'єктів;
- класифікація: застосування методу дерев дозволяє впоратись із задачею класифікації, тобто віднесення об'єктів до одного з зазначених класів;
- регресія: у випадку коли змінна має неточні значення, то застосування дерева дозволяє визначити залежність даної цільової змінної від незалежних (вхідних) змінних.

Також А. Олійник зазначає: «Головна перевага «дерева рішень» перед іншими методами - можливість пов'язати ставлення цілі з діями, що підлягають реалізації в сьогоденні. При побудові багаторівневого "дерева рішень" досягнення мети кожного з рівнів моделі забезпечується комплексом заходів попереднього рівня. Кожен рівень "дерева рішень" повинен займати певне місце в ієрархічній послідовності, складеної на основі дотримання причинно-наслідкових зв’язків» [10].

У ролі критерію для розбиття виступає статистичний критерій, який має назву «індекс Gini». Обчислюється за формулою:

$$Gini(c) = 1 - \sum_{j=1}^{k} p_j^2, \qquad (2.1)$$

Де c – поточний вузол, p_j - ймовірність класу j у вузлі c.

Важливим показником є точність розпізнавання даного методу. Під точністю розпізнавання слід розуміти відношення правильно класифікованих об’єктів при навчанні до загальної кількості об’єктів з множини, яка навчається, а під помилкою прогнозу – множина неправильно класифікованих.

Наступним методом є метод k-найближчих сусідів. Метод k-найближчих сусідів — це класифікаційний метод, який є непараметричним, де для класифікації об'єктів у межах простору властивостей використовуються відстані (наприклад евклідові), що рахуються до всіх інших об'єктів. Далі обираються об'єкти, до яких відстань найменша після чого, вони виділяються в окремий клас. Як йдеться у лекції К. Воронцова: «Параметр k у методі kNN часто вибирається на підставі досвіду чи знань про розв'язувану задачу класифікації. Бажано, щоб параметр k був непарним, щоб зменшити імовірність «нічиєї». Найчастіше вибираються значення k = 3 і k = 5, але використовуються і великі значення, між 50 і 100. Як альтернативу параметр k можна вибрати так, щоб він гарантував найкращі результати на відкладеній частині навчального множини» [11]. Можна також приписати ваги “голосам” k найближчих сусідів, використовуючи їх косинусну міру подібності:

$$\cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \qquad (2.2)$$

де A_i та B_i – координати вектору A та B відповідно.

Також для розробки класифікатора був використаний метод опорних векторів. Метод опорних векторів - відноситься до групи граничних методів, він визначає класи за допомогою границь областей. За допомогою даного методу вирішуються задачі бінарної класифікації, тобто тільки за двома категоріями c і $\bar{c}$ (слід взяти до уваги те, що цей підхід може бути розширений на необхідне число категорій). Також передбачається, що кожен об'єкт класифікації є

вектором у N-вимірному просторі, у той час як кожна координата вектора – це деяка ознака, кількісно більша у тих межах, в яких більша ця ознака виражена в даному об'єкті. Ідея методу полягає в тому, що потрібно знайти пряму (гіперплощина у N-мірному просторі), що відокремлює всі точки одного класу від точок іншого. У випадку коли вдається знайти таку пряму, задача класифікації зводиться до визначення взаємного розташування точки і прямої: якщо нова точка лежить з одного боку прямої (гіперплощини), то вона належить класу c , якщо з іншого боку – класу $\bar{c}$. Необхідно знайти вектор w такий, що для деякого граничного значення b і нової точки і x виконується:

$$y = \begin{cases} +1, \text{якщо } w * x_i \geq b, \\ -1, \text{якщо } w * x_i > b, \end{cases} \quad (2.3)$$

де $w * x_i$ – скалярний добуток векторів w і x_i

Рівняння 2.4 у свою чергу описує гіперплощину, що розділяє класи.

$$w * x_i = b, \quad (2.4)$$

Іншими словами, якщо скалярний добуток векторів w і x_i не менш значення b, то нова точка належить до першого класу, якщо менше – до другого. Відомо, що вектор w перпендикулярний до розділяючої прямої, яка має бути знайдена, а значення b залежить від найкоротшої відстані між поділяючою прямою і початком координат. Таким чином, якщо існує розділяюча пряма, то вона не єдина. Постає питання як обрати найкращу розділяючу пряму. Метод SVM базується на такому постулаті: пряма, що є найкращою для розділення – це та, яка знаходиться максимально далеко від найближчих до неї точок обох класів [12]. Тобто завдання методу SVM полягає в тому, щоб знайти такі вектор w і число b , щоб для деякого $\varepsilon > 0$ (половини ширини розділяючої поверхні) виконувалося рівняння (2.5):

$$\begin{cases} w * x_i \geq b + \varepsilon \rightarrow y_i = +1, \\ w * x_i \geq b - \varepsilon \rightarrow y_i = -1 \end{cases} \quad (2.5)$$

Також у ході роботи з даними були використані наступні методи для балансування даних. Алгоритм SMOTE - це стратегія заснована на ідеї створення певної кількості штучних прикладів, які були б «схожі» на наявні в міноритарною класі, але при цьому не дублювали їх. Даний підхід має недолік в тому, що «наосліп» збільшує щільність прикладами в області слабо представленого класу. У разі, якщо міноритарні приклади рівномірно розподілені серед мажоритарних і мають низьку щільність, алгоритм SMOTE тільки сильніше перемішає класи [13].

RandomUnderSampler – метод зниження вибірки мажоритарного класу (класів) шляхом випадкового відбору зразків із заміною або без неї [14].

Також для визначення точності роботи методів, була використана бібліотечна функція sklearn.metrics.accuracy_score [15]. В основі функції лежить формула (2.6):

$$accuracy(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} 1(\hat{y}_i = y_i), \quad (2.6)$$

де $\hat{y}_i$- прогнозуємо значення, $y -$ відповідне правдиве значення, n – кількість зразків.

2.2 Підготовка набору даних для навчання інтелектуального модулю

Як відомо, найбільш поширеними є локальні і мережеві СВВ. Локальна СВВ передбачає, що система виявлення встановлюється на кожному окремому комп'ютері. Мережева СВВ збирає пакети, що надходять в мережу через один пристрій і аналізують їх, перш ніж пересилати заданим вузлам. Так як мережеві СВВ вважаються менш ефективними на сьогоднішній день (тому що чим більша

кількість вузлів в мережі тим важче стає забезпечити надійність фільтрації пакетів), то задача і дані моделюють саме процеси локальної мережі, для якої розробляється програмний модуль, який слугую для роботи локальної СВВ.

Системи виявлення мережевих вторгнень і виявлення ознак комп'ютерних атак на інформаційні системи, як повідомляють [16], вже давно застосовуються як один з найсильніших інструментів оборони ІС і використовуються для виявлення окремих типів шкідливої активності, яка негативно впливає на безпеку комп'ютерної системи. До такої активності відносяться мережеві атаки, що спрямовані проти вразливих сервісів, атаки, які передбачають підвищення привілеїв, неавторизований доступ до важливих файлів, а також дії шкідливого програмного забезпечення (комп'ютерних вірусів, троянів і черв'яків).

Після усвідомлення конкретної цілі дослідження відбувся процес дослідження предметного середовище, з яким маємо справу та дослідження обраних даних, які задовольняють задачу. Як зазначалося вище, у якості набору даних були обрані дані MIT Lincoln Labs, які розробили імітаційну модель локальної мережі та збирали послідовності пакетів TCP протягом дев’яти тижнів.

Дані містять всього 4 категорії атак:

DOS: відмова в обслуговуванні або атака на певну систему, щоб вивести її з ладу і інші користувачі не могли отримати доступ до ресурсів системи, або коли цей доступ ускладнений.

R2L: визначається отриманням доступу стороннього або незареєстрованого користувача до комп'ютера;

U2R: це незаконний доступ до привілеїв локального суперкористувача, наприклад, різні атаки «переповнення буфера»;

probe: полягає у скануванні портів з метою отримання конфіденційної інформації.

П’ята категорія, яка має назву «benign» означає безпечне з’єднання.

Якщо характеризувати набір ознак, що містить обраний набір даних, то він має 41 ознаку, при чому містяться як бінарні, так і номінальні ознаки. На таблицях нижче можна ознайомитися із структурою даних та їх типами.

Таблиця 2.1. Основні особливості окремих TCP-з'єднань.

Назва функції	Опис	Тип
duration	length(number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g tcp, udp, etc.	discrete
service	network service on the destination, e.g, http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of “wrong” fragment	continuous
urgent	number of urgent packets	continuous

Таблиця 2.2. Ознаки вмісту в з’єднанні, запропоновані знаннями домену.

Назва функції	Опис	Тип
hot	number of “hot” indicators	continuous
num_feiled_logins	number of failed .login attempts	continuous

Продовження таблиці 2.2

logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of “compromised” conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if “su root” command attempted; 0 otherwise	discrete
num_root	number of “root” accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the “hot” list; 0 otherwise	discrete
is_guest_login	1 if the login is a “guest” login; 0 otherwise	discrete

Таблиця 2.3. Характеристики трафіку, що розраховуються із використанням 2-секундного тимчасового вікна.

Назва функції	Опис	Тип
count	number of connections to the same host as the	continuous

Продовження таблиці 2.3

	number of connections to the same host as the current connection in the past two seconds	
	note: the following features refer to these same-host connections.	
serror_rate	% of connections that have “syn” errors	continuous
rerror_rate	% of connections that have “rej” errors	continuous
same_srv_rate	% of connections to same services	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the two seconds	continuous
	note: The following features refer to these same-service connections.	
srv_serror_rate	% of connections that have “syn” errors	continuous
srv_rerror_rate	% of connections that have “rej” errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

Також наявний список видів атак, які мали місце. Серед яких perl, satan, neptune, imap, landmodule, back, warezmaster тощо.

2.3 Архітектура застосунку виявлення аномалій у мережевому трафіку

Інтелектуальний модуль для виявлення аномалій приймає на вхід дані, які були зібрані під час роботи мережі. На виході із системи користувач отримає результати точності по кожній моделі та час роботи кожного методу. Чинники, які можуть впливати на модуль це користувач безпосередньо, а також аналітик або ж спеціаліст з кібербезпеки. До обмеженнь, які мають місце, можна вказати конфіденційність даних та доступи до мережі, які з певних причин можуть не надаватись. Програмний модуль можна описати у вигляді «чорної скриньки» або IDEF0, яка зображена на рисунку 2.2.

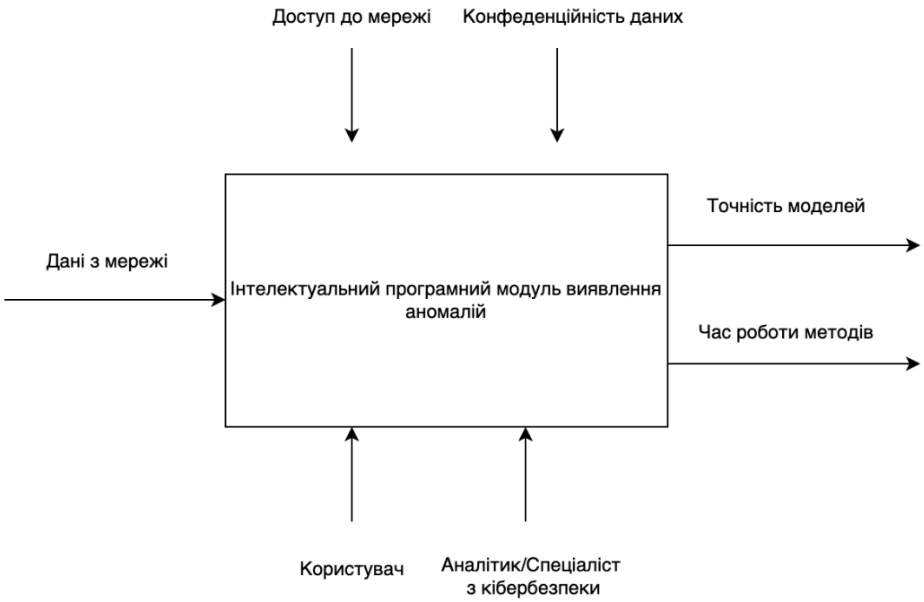

Рисунок 2.2 Опис системи в нотації IDEF0

Аналіз мережевого трафіку на наявність аномалій буде включати в себе дві основні функції:

- попередній обробці даних
- побудова класифікаційної моделі прогнозування;

Функція побудови моделі прогнозування буде включати у себе наступні підфункції:

- структуризація даних;
- очистка даних;
- нормалізація даних;
- усунення дисбалансу класів;
- налаштування параметрів моделі прогнозування;
- навчання моделі;

Аналіз впливу аномалій на трафік буде включати у себе:

- вибір класифікаційної моделі;
- прогнозування;
- виявлення найбільш впливових характеристик.

Із описаного вище функціонального аналізу побудуємо дерево функцій, яке зображене на рисунку 2.3

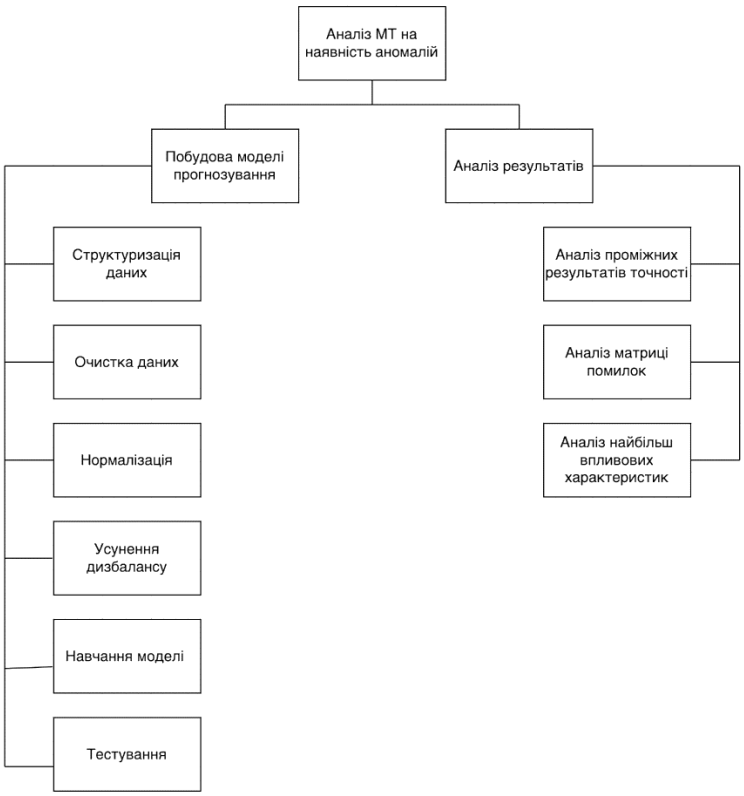

Рисунок 2.3 Дерево функцій системи

Із вищеописаного дерева функцій можна описати карту процесів. Побудова класифікаційної моделі складається із чітко визначених послідовних функцій що включає: очистку даних, нормалізації, усунення дисбалансу, навчання, тестування.

Із побудованої класифікаційної моделі ми переходимо до аналізу результатів, що складається із зв’язаних процесів: аналізу проміжних результатів точності, аналізу матриці помилок та аналізу найбільш впливових характеристик.

Для побудови узагальненої архітектури системи буде використовуватися засіб StarUML. Для визначення компонентів системи побудуємо компонентну діаграму (рисунок 2.4).

Застосунок буде мати клієнт-серверну архітектуру; відповідні вузли представлені на діаграмі як «Client» та «Server».

Клієнтський вузол буде мати наступні компоненти:

- збір даних системи (Data Collection);
- спеціаліст з кібербезпеки (Cybersecurity specialist);
- клієнтський інтерфейс (User Interface).

Серверна частина буде мати наступні компоненти:

- підготовка даних (Data Preprocessing);
- навчання (Training);
- прогнозування (Prediction).

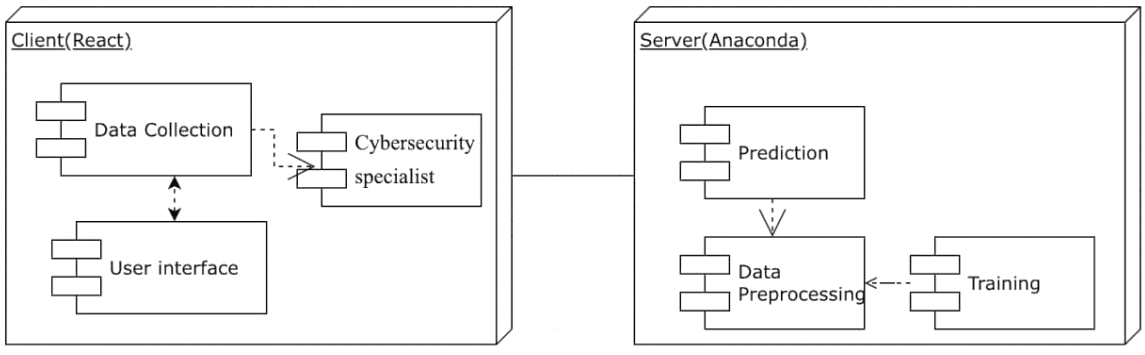

Рисунок 2.4 Компонентна архітектура застосунку

Застосунок буде складатись із двох складових: клієнта та сервера. Застосування клієнта буде відповідати за відображення та візуалізацію даних системи. Він буде базуватись у наданні користувачу основної функціональності додатку.

Користувач буде мати можливість локально переглядати інформацію про дані. Обрані дані будуть надсилатись на сервер для їх подальшої обробки.

На сервері будуть оброблятися дані системи і за відповідним запитом від клієнта буде виконуватись функція прогнозування з обраною моделлю. Ці дані будуть автоматично надсилатись до клієнта. Також сервер буде відповідати за функцію навчання моделі та виводити результуючі графіки.

Даний інтелектуальний модуль можна описати за допомогою діаграми прецедентів (рисунок 2.5), яка буде описувати можливості користувача та системи. Для користувача буде доступним обрати необхідну інформацію, отримати візуалізацію та результати по кожному з методів. Для модуля будуть доступні прогнозування, навчання моделей, а також отримання результатів і візуалізації по кожному із методів.

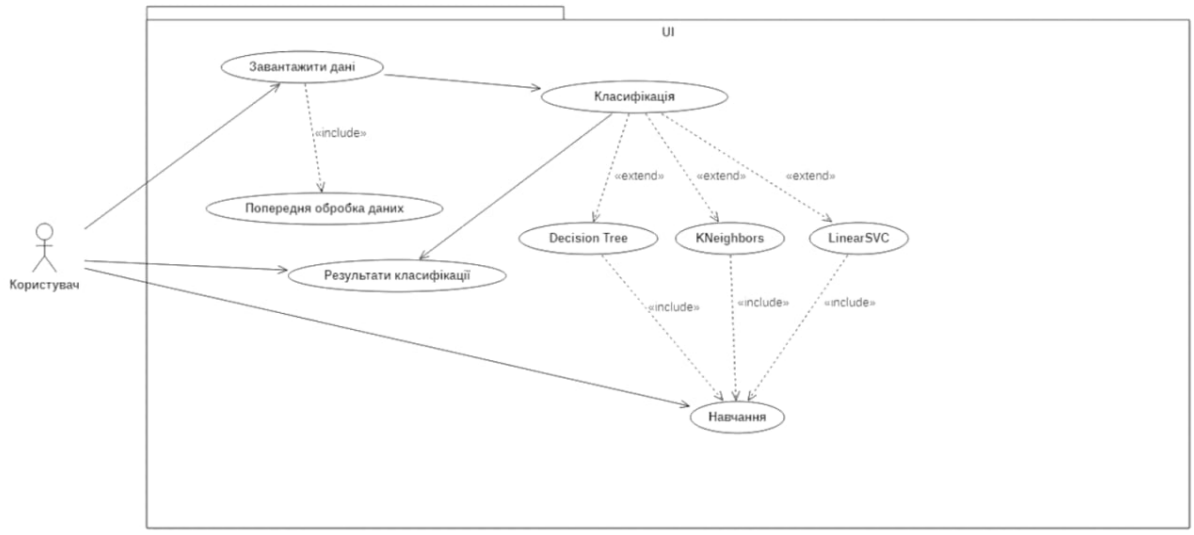

Рисунок 2.5 UML діаграма прецедентів

Також для усвідомлення послідовності процесів програмного модуля була розроблена діаграма послідовності, що зображена на рисунку 2.6. З наведеної діаграми можна зрозуміти, що процес починається із завантаження даних користувачем, які відображаються у розробленому інтерфейсі веб-додатку. Далі, дані передають безпосередньо у програмний модуль, де відбувається процес обробки. Наступним кроком дані подаються у розроблену модель класифікатора, після чого отримані результати у вигляді звіту надходять користувачу на окреме вікно веб-додатку.

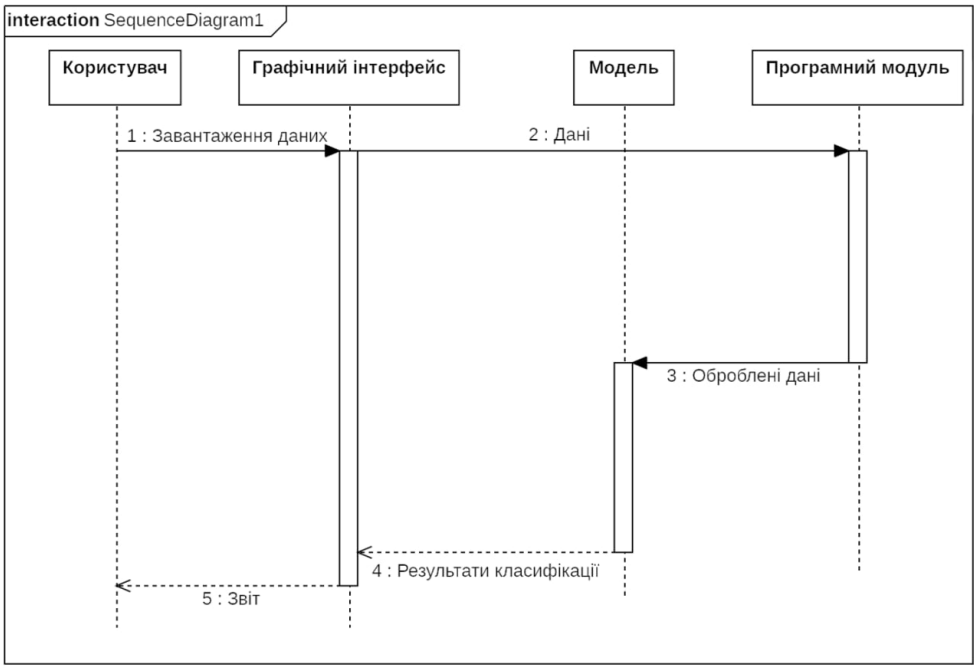

Рисунок 2.6 - UML діаграма послідовності

Також для наглядної демонстрації процесів, які передбачає інтелектуальний модуль, була розроблена діаграма діяльності, яка зображена на рисунку 2.7. Характеризуючи процеси діяльності програми, варто сказати, що процеси мають послідовний характер та досить залежні один від одного. Після запуску програми, вона готова до використання і завантаження даних. Після чого відбувається блок попередньої обробки, який включає в себе структуризацію, очистку даних, нормалізацію та усунення дисбалансу класів. Після чого відбувається процес навчання, класифікації та оцінка результатів. Далі відбувається завершення роботи програми та вихід.

Рисунок 2.7 - UML діаграма діяльності

2.4 Вибір середовища та інструментів для розробки застосунку виявлення аномалій у мережевому трафіку

Для реалізації попередньої обробки даних та алгоритмів обраних методів було вирішено використовувати програмне забезпечення «Anaconda». Anaconda - це розповсюджуваний дистрибутив різніх програмних продуктів, а саме, мов програмування Python та R. Вонаа спеціалізується на науці про дані (Data Science), наукових обчисленнях (scientific computing), застосуванні методів

машинного навчання, обробці даних ,а також у передбачувальній аналітиці тощо. Використання платформи має на меті спрощення управління пакетами та їх розгортання. Дистрибутив Anaconda використовують понад 16 мільйонів користувачів і містить більш ніж 1600 популярних пакетів наукових даних, придатних для Windows, Linux та MacOS, например, NumPy, SciPy та Ggplot2 [17].

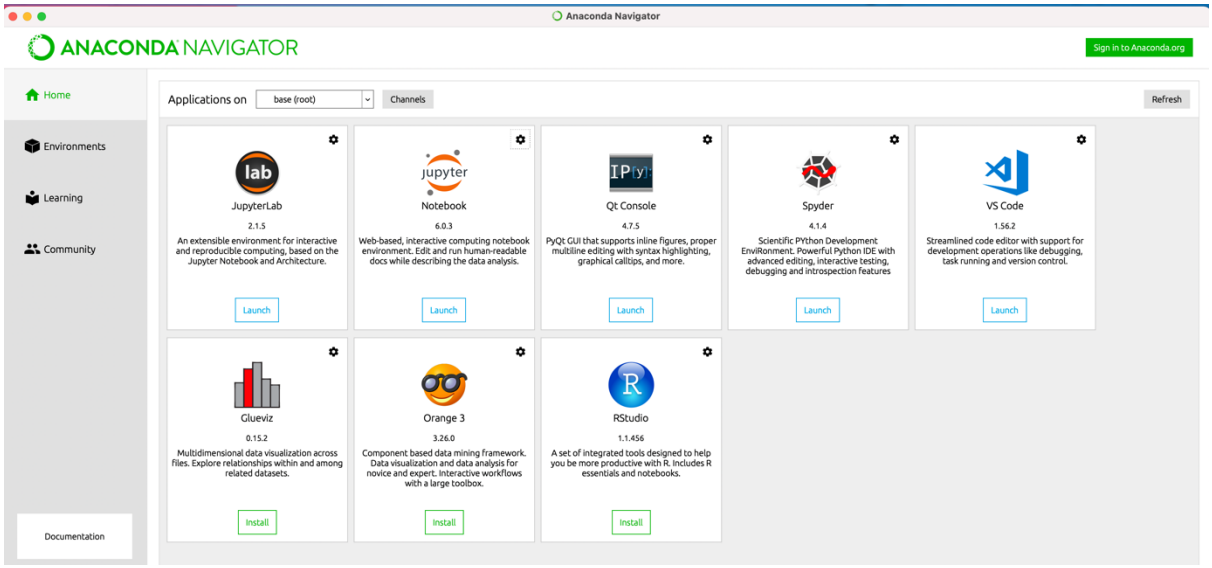

Рисунок. 2.8 Зовнішній вигляд програмного забезпечення

Найбільш поширеними інструментами в Data Science на сьогоднішній день є R і Python. У кожного інструменту є свої плюси і мінуси, проте, останнім часом по всіх параметрах виграє Python. Це стало після того, як з'явилася відмінно документована бібліотека Scikit-Learn, в якій реалізована велика кількість алгоритмів машинного навчання [18].

Переваги мови Python :

- Інтерпретатор Python реалізований практично на всіх платформах та операційних системах.
- Розвиток мови (є можливість удосконалювати мову усіма зацікавленими програмістами)
- Наявність великої кількості модулів, що підключаються до програми, які забезпечують різноманітні додаткові можливості.

Приклади модулів:

- Numerical Python - розширені математичні можливості (маніпуляція з цілими векторами та матрицями);
- Tkinter - використання графічного інтерфейсу користувача;
- OpenGL - використання великої бібліотеки графічного моделювання двох та тримірних об‘єктів.
- Простота та гнучкість мови.

Завдяки їм, Python може використовуватись користувачами (математиками, фізиками, економістами та ін.), що не є програмістами, але використовують обчислювальну техніку в своїй роботі.

Недоліки мови Python:

Єдиним недоліком мови є порівняно невисока швидкість виконання програми Python, що обумовлено її інтерпретованістю. Проте переваги значно більші у програмах не дуже критичних щодо швидкості виконання. Тому було вирішено обрати саме цю мову [19].

Середовищем для реалізації став Jupyter Notebook. Jupyter Notebook – це командна оболонка для інтерактивних обчислень. Цей інструмент може використовуватися не лише на Python, але й на інших мовах програмування: Julia, R, Haskell і Ruby. Він часто використовується для роботи з даними, статистичним моделюванням і машинним навчанням. Дану програму було вибрано через зручність у використанні під час написання коду програми, тому що код розділений на певні блоки, які можна запускати незалежно один від одного [20].

Для розробки інтерфейсу додатку було використано такі технології як React.js фреймворк, UI бібліотека Materialize та Chart.js [21].

React.js фреймворк, у свою чергу, це відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці одно сторінкових застосунків.

Висновки за розділом 2

Отже, в рамках другого розділу було розглянуто інтелектуальний підхід до розробки модуля, детально описані методи і підходи, які застосовувалися. Описано архітектуру застосунку як словесно, так і за допомогою UML діаграм. Виявлено за допомогою чого, було досягнуто розуміння всіх внутрішніх процесів, взаємодії користувача із розробленим веб-додатком, а також послідовності процесів, що відбуваються у ході роботи модулю. Також було описано інструменти, за допомогою яких відбулася реалізація інтелектуального програмного модуля.

РОЗДІЛ 3 ПРОЕКТНО – ТЕХНІЧНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОГО ПРОГРАМНОГО МОДУЛЮ ВИЯВЛЕННЯ АНОМАЛІЙ

3.1 Опис реалізації програмного застосунку виявлення аномалій у мережевому трафіку

Варто зауважити, що обраний набір даних був не структурований та важкий для експлуатації, тому велика увага приділялася саме структуризації та попередній обробці даних.

У рамках попередньої обробки даних було проведено деяке дослідження видів атак та виявимо під яку категорію підпадає кожна з них. На рисунку 3.1 можна побачити розподіл атак за їх видами.

```
In [91]: print(category)

defaultdict(<class 'list'>, {'benign': ['normal'], 'dos': ['apache2', 'back', 'mailbomb', 'processtable', 'snmpgetatt
ack', 'teardrop', 'smurf', 'land', 'neptune', 'pod', 'udpstorm'], 'u2r': ['ps', 'buffer_overflow', 'perl', 'rootkit',
'loadmodule', 'xterm', 'sqlattack', 'httptunnel'], 'r2l': ['ftp_write', 'guess_passwd', 'snmpguess', 'imap', 'spy',
'warezclient', 'warezmaster', 'multihop', 'phf', 'imap', 'named', 'sendmail', 'xlock', 'xsnoop', 'worm'], 'probe':
['nmap', 'ipsweep', 'portsweep', 'satan', 'mscan', 'saint', 'worm']})
```

Рисунок 3.1 Розподіл видів атак за категоріями

Розподіляємо весь набір даних на категорії – тестові, символьні та бінарні, так як кожний тип даних потребує дещо специфічної попередньої обробки.
У ході процесу структуризації дані були переведені із текстового формату даних .txt у формат файлу .csv для проведення більш зручних операцій над даними. Після процесу структуризації дані мають наступний вигляд(рисунок 3.2):

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_s
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	0.17	0.03	
1	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	
2	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	
3	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	
4	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	

5 rows × 43 columns

Рисунок 3.2 Вигляд даних

Наступним кроком, формуємо тренувальну та тестову вибірки. У класичному співвідношенні 80% на 20 % відповідно.

Далі був проведений етап візуалізації початкових даних. На рисунку 3.3 та рисунку 3.4 можемо побачити розподіли тренувальних даних, де видно які види атак наявні, а також їх розподіл по категоріям.

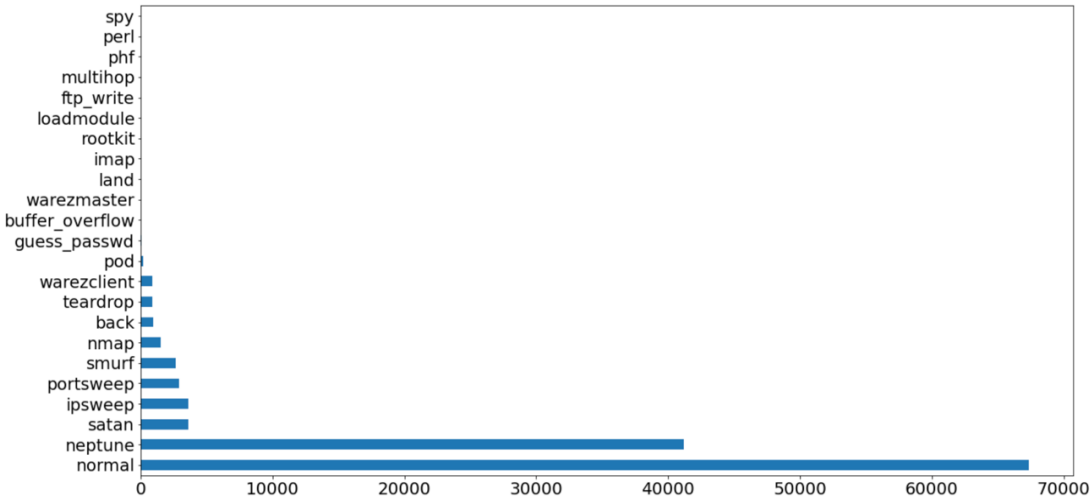

Рисунок 3.3 Розподіл тренувальних даних по 22 категоріям

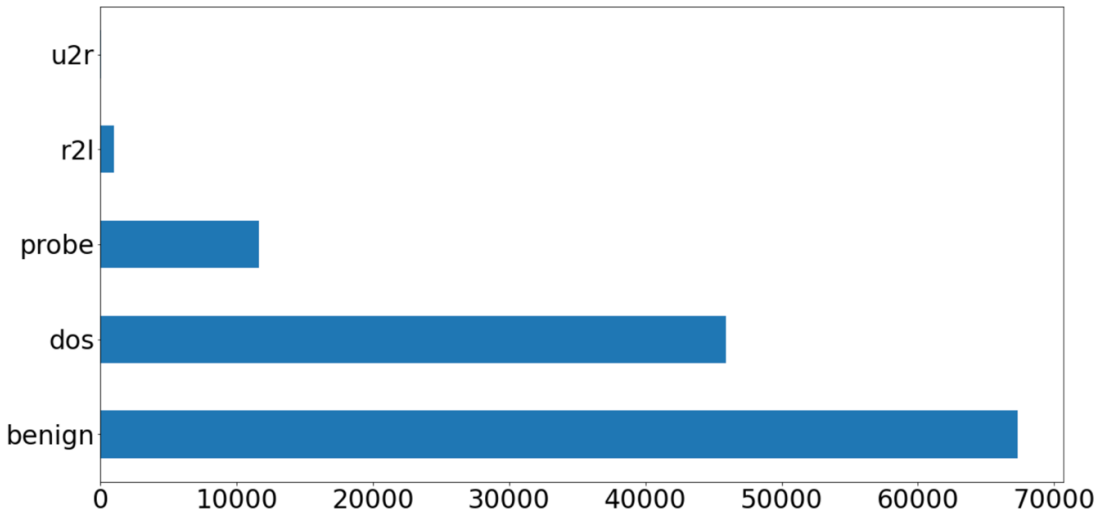

Рисунок 3.4 Розподіл тренувальних даних по 5 категоріях

Дані діаграми показали на явний дисбаланс класів, що може завадити навчанню моделей. Тому для підвищення точності розробленого класифікатора будуть застосовуватись методи боротьби із дисбалансом.

У рамках очищення даних також були видалені деякі ознаки, які не несли у собі ніякого змістового навантаження.

Важливим етапом у попередній обробці даних було кодування символьних змінних для зручної роботи з ним. Функція pd.get_dummies() генерує для кожного можливого значення символьної змінної бінарне значення. Наприклад, якщо елемент вибірки має значення flag=S2, то представлення фіктивної змінної буде виглядати так (рисунок 3.5):

```
# flag_S0, flag_S1, flag_S2, flag_S3, flag_SF, flag_SH
[    0,       0,       1,       0,       0,       0    ]
```

Рисунок 3.5 Вигляд фіктивної змінної

Також під час попередньої обробки даних була помічена небезпечна особливість – розподіл кожної ознаки має великий розкид, що може також впливати на точність прогнозування моделі, що можна побачити на рисунку 3.6. Тому була проведена нормалізація ознак різними способами

Out[46]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	r
count	125973.00000	1.259730e+05	1.259730e+05	125973.000000	125973.000000	125973.000000	125973.000000	
mean	287.14465	4.556674e+04	1.977911e+04	0.000198	0.022687	0.000111	0.204409	
std	2604.51531	5.870331e+06	4.021269e+06	0.014086	0.253530	0.014366	2.149968	

Рисунок 3.6 Характеристики ознак

Після використання декількох методів нормалізації, я зупинилася на StandardScaler, за допомогою якого, вдалося підвищити точність моделі.

Також, як раніше зазначалося, був виявлений ще один недолік в даних, а саме дисбаланс класів. Тому використаємо два методи боротьби з цим явищем. Результати зображені у таблиці 3.1

Таблиця 3.1 Результати балансування

Назва категорії	Кількість до балансування	Кількість після балансування методом SMOTE	Кількість після балансування методом RandomUnderSampler
benign	67343	67343	25194
dos	45927	67343	25194
probe	11656	67343	25194
r2l	995	67343	25194
u2r	52	67343	

З числових характеристик кожної категорій можна зробити висновок, що метод SMOTE підвищує кількість даних до верхньої межі найбільшої категорії за чисельністю, у той час як метод RandomUnderSampler балансує дані методом зниження кількості до найменшої категорії за чисельністю. Ці два протилежні підходи дали змогу порівняти і оцінити який підхід краще працює у даному випадку. Краще на точність вплинув метод SMOTE.

Варто зауважити, що перед кожним вирішенням недоліку в даних, відслідковувалась точність прогнозування обраних моделей. Як дані вдосконалення впливають на покращення точності буде показано у наступному розділі. На даному етапі приведу приклад найкращого показника точності, який був досягнутий з використанням моделі дерева рішень, що можна побачити у таблиці 3.2.

Таблиця 3.2 Матриця помилок методу «Дерева рішень»

	benign	dos	probe	r2l	u2r
benign	9365	56	289	1	0
dos	1541	5998	97	0	0
probe	677	220	1526	0	0
r2l	2278	1	14	277	4

Продовження таблиці 3.2

u2r	175	0	5	5	15

Точність, яка було досягнута сягає 77% на тестовій вибірці, що є досить достойним результатом для погано структурованих даних.

Для реалізації інтерфейсу додатку необхідно викликати відповідні модулі та бібліотеки, які передбачаються для інструменту React.js фреймворк. Такі бібліотеки як Grid, ToggleButton та ToggleButtonGroup мали місце. Варто зауважити, що програмний файл, в якому відбувається розробка методів, пов'язаний з файлом для розробки інтерфейсу.

Були розроблені три вікна для надання необхідної інформації для користувачів. У першому вікні користувач може оглянути дані, які досліджуються, а також переглянути розподіли у даних, які мають місце. Як виглядає дане вікно наведено на рисунок 3.7.

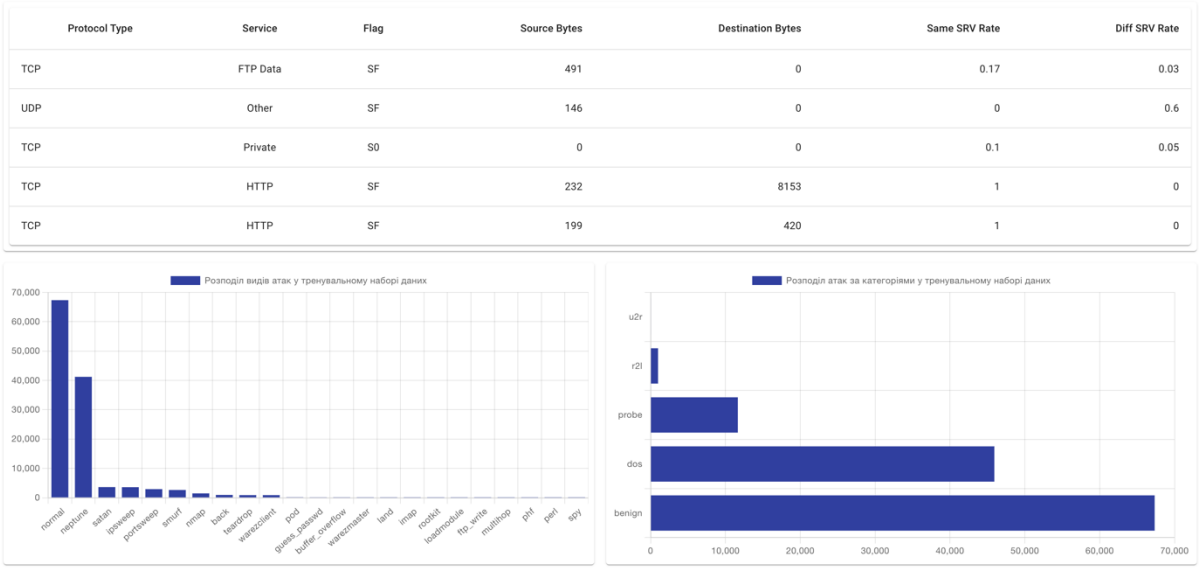

Рисунок 3.7 Вигляд першого вікна

У другому вікні користувач може переглянути результати точності за кожним методом та переглянути матрицю помилок, а також наявний блок візуалізації, який дозволяє оцінити переваги та недоліки моделей та усвідомити які ж характеристики у наборі даних найбільш впливають на результати та

порівняти ефективність методів як за точністю прогнозування, так і за часом роботи методів. Вигляд вікна можна побачити на рисунку 3.8

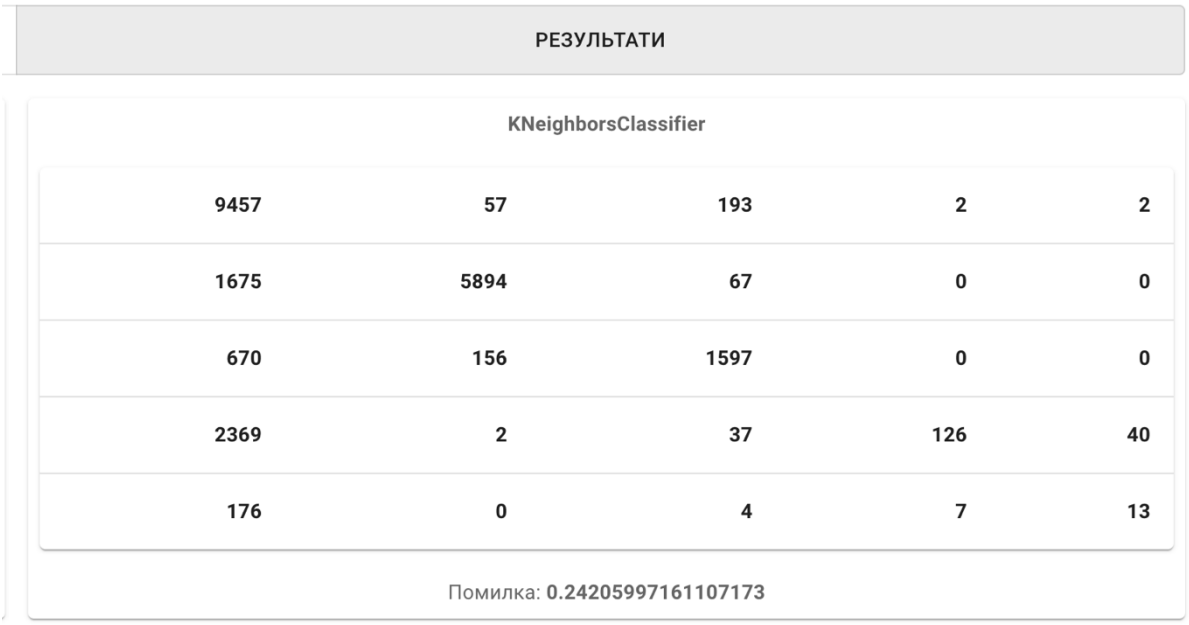

Рисунок 3.8 Вигляд результатів за методами

3.2 Тестування та аналіз результату роботи інтелектуального програмного модуля виявлення аномалій у мережевому трафіку

Як вже було зазначено, для даного курсового проекту були обрані три метода машинного навчання, такі як: метод опорних векторів, метод дерева рішень та метод найближчого сусіда. У ході роботи в наборі даних були виявлені певні недоліки, усунувши які, можемо прослідкувати як змінювалась точність обраних моделей у таблиці 3.3.

Таблиця 3.3 Точність моделей

Назва	Після препроцесінгу	Після нормалізації	Після балансування
DecisionTreeClassifier	0.7566	0.7567	0.7761
KNeighborsClassifier	0.7369	0.758	0.758

Продовження таблиці 3.3

LinearSVC	0.6857	0.7218	0.7219

Вимірявши показники швидкості розрахунків кожного методу, можна зробити висновок, що найшвидшим виявився метод дерев рішень.

А також, аналізуючи показники точності прогнозування, найбільш точним виявився метод дерев рішень з точність 77%.

У ході експериментальної верифікації було визначено найвпливовіші характеристики, від яких залежить точність прогнозування моделі. Діаграма, яка відображає дані результати наведена нижче. Варто зауважити, що для візуалізації даної діаграми у веб-додатку, були обрані характеристики, які мають найбільший вплив. Це було зроблено для того, щоб користувач мав краще усвідомлення впливів.

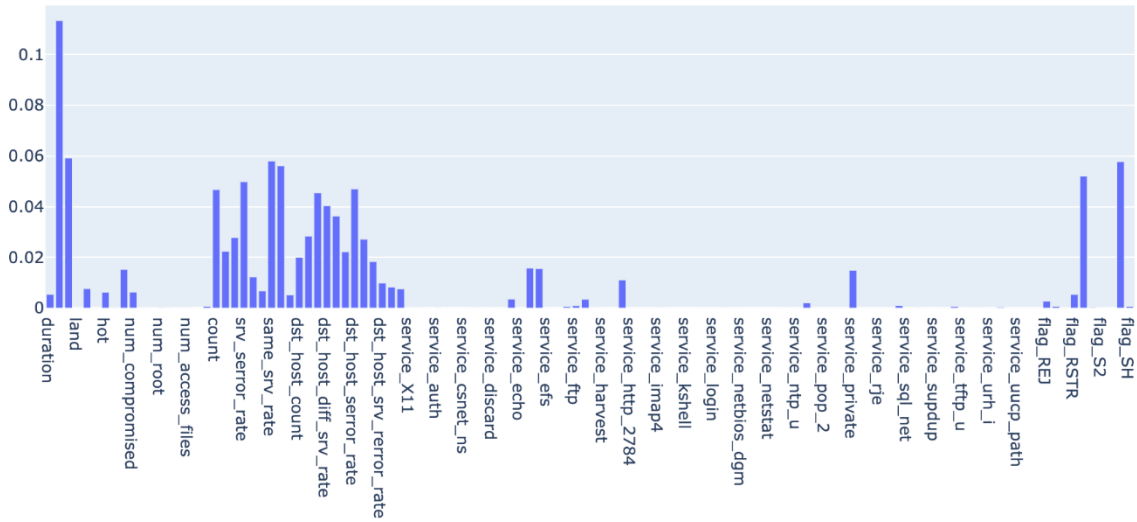

Рисунок 3.9 Візуалізація найвпливовіших характеристик

Для більш наглядного порівняння роботи кожного із методів була розроблена відповідна діаграма, яка демонструє результати точності кожного методу, а також помилку кожного методу, яка мала місце. Дана діаграма зображена на рисунку 3.10

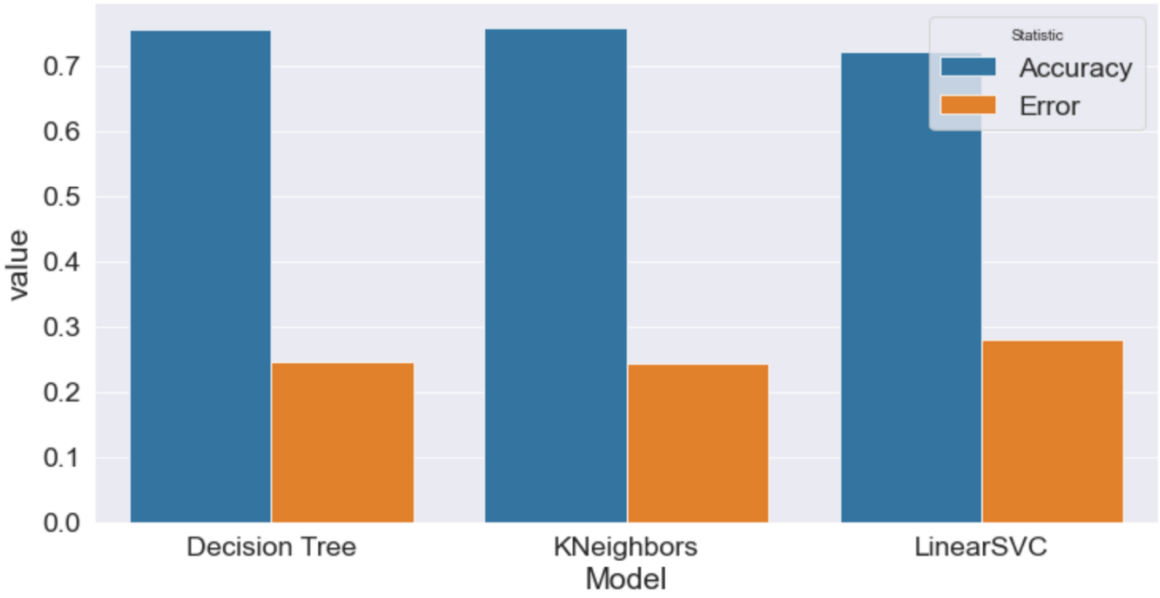

Рисунок 3.10 Діаграма порівняння точності результатів

3.3 Інструктивний матеріал користувача для роботи з інтелектуальним програмним модулем виявлення аномалій у мережевому трафіку

Для роботи з інтелектуальним застосунком необхідно заздалегідь установити Python останньої версії. Варто зауважити, що основний файл, де відбувається побудова та навчання моделі для класифікації, а також структурування даних має розширення .ipynb. Тому в обраному користувачем середовищі, варто використати відповідний інтерпретатор для даного розширення.

Наступним кроком потрібно розархівувати папку з проектом і запустити відповідний файл, куди передаються дані, як показано на рисунку 3.11.

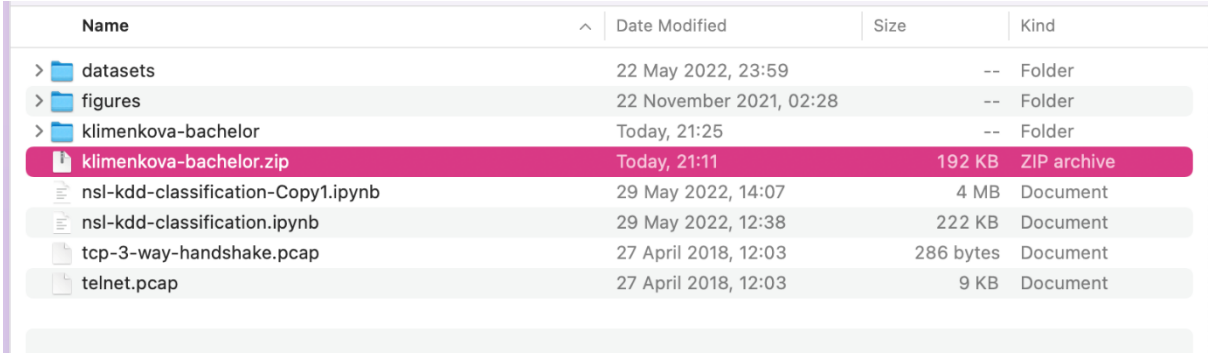

Рисунок 3.11 Архів проекту

У браузері, який встановлено за замовченням, відкриється розроблений веб-додаток. Функціонал інтерфейсу дозволяє користувачу перемикатись між вкладками, обрати метод, результати якого цікавлять, а також обирати необхідний графік для порівняння результатів.

Варто підкреслити переваги розробленого додатку, так як він є універсальним для будь-якої операційної системи, також він досить легкий у застосуванні та інтуїтивно зрозумілий для користувача.

Для зручності користувача був розроблений чіткий алгоритм по використанню.

Інструкція користувача:

1. Перевірити необхідні інструменти для роботи з додатком (Python останньої версії, програмне середовище, будь-який інтернет браузер)
2. Розархівувати папку з проектом
3. Запустити відповідний html файл або ж перейти за посиланням у текстовому файлі формату .txt, який міститься у папці

На першій сторінці Ви можете ознайомитись із виглядом даних, які містяться у наборі. Також побачити візуалізацію деяких розподілів щодо категорій та видів атак у тренувальній та тестовій виборках.

На другій сторінці містяться результати за кожним методом машинного навчання, а саме DecisionTreeClassifier, KneighborsClassifier, LinearSVC. Є можливість ознайомитись з точністю роботи кожного методу, а також зі швидкістю роботи кожного методу. Блок візуалізації у нижньому правому кутку дасть можливість ознайомитись з важливістю і впливом кожної характеристики набора даних, а також порівняти точність кожного методу.

Висновок за розділом 3

В рамках третього розділу було розглянуто опис реалізації програмного застосунку виявлення аномалій у мережевому трафіку. Описано програмний

підхід та кроки його реалізації, специфіку роботи з обраними даними, програмні блоки та методи класифікації.

Проведена експериментальна верифікація, порівняно результати роботів кожного методу. Описано блок візуалізації даних, за допомогою якого користувач має можливість наглядно порівняти отримані результати.

Також, розроблено інструкцію користувача, яка орієнтована на спрощення експлуатації веб-застосунку та більш зручного його використання.

Висновок

У результаті виконання дипломної роботи було отримано інтелектуальний веб-додаток з для виявлення аномалій у мережевому трафіку.

Проведено аналітичний огляд дипломної роботи. Проаналізовано існуючі підходи, рішення та результати, що досягнуто на даний момент часу. Проаналізовані існуючі методи та підходи щодо проблеми аналізу системи на ураження.

Проведено функціональний аналіз, визначено вимоги до застосунку. Розроблено архітектуру застосунку, визначено зв’язки між компонентами розробленої системи. Розроблено застосунок із використанням мов програмування Python та фрейморка React.js, та створено документацію для користувача.

Таким чином, даний застосунок можна буде в подальшому використовувати у компаніях, які дбають про безпеку та конфіденційність даних та прагнуть завчасно реагувати на небезпеку. З технічної точки зору даний застосунок можна покращити шляхом оптимізації коду та залученню нових методів класифікації, що в подальшому підвищить швидкість реагування на небезпеку.

Точність моделі в результаті балансування та обробки даних була підвищена не на значний відсоток, але загальна тенденція така, що усунення недоліків допомагало підвищувати точність.

Список використаних джерел

1. Verizon Data Breach Investigations Report. [Електронний ресурс]. Режим доступу: https://www.verizon.com/business/resources/reports/2022/dbir/2022-data-breach-investigations-report-dbir.pdf

2. Звіт Ради Національної безпеки України [Електронний ресурс]. Режим доступу: https://ms.detector.media/kiberbezpeka/post/25227/2020-08-07-v-ukraini-v-2020-rotsi-zafiksuvaly-1-milyon-kiberatak-rnbo/

3. Шелухин О.И. Обнаружение вторжений в компьютерные сети (сетевые аномалии): учебное пособие для вузов / О.И. Шелухин, Д.Ж. Сакалема, А.С. Филинова. – М.: Горячая линия-Телеком, 2013. – 220 с

4. Кларенс Чіо, Девід Фрімен. -М.: ДМК Пресс, 2020. «Машинное обучение и безопасность» с. 97, с. 99, с. 104-106, с. 108

5. Опис утиліти для моніторингу протоколів [Електронний ресурс]. Режим доступу: https://www.tcpdump.org/manpages/tcpdump.1.html

6. Опис утиліти Zeek для моніторингу протоколів [Електронний ресурс]. Режим доступу: https://zeek.org/

7. Опис інструменту для виявлення зловмисних з’єднань [Електронний ресурс]. Режим доступу: https://www.snort.org/

8. Скабцов Н., Аудит безопасности информационных систем. — СПб.: Питер, 2018. — 272 с.: ил. — (Серия «Библиотека программиста»).

9. А. Завада, О. Самчишин, В. Охрімчук, "Аналіз сучасних систем виявлення атак і запобігання вторгненням”, Інформаційні системи, Житомир: Збірник наукових праць ЖВІ НАУ, Т. 6, No 12, с. 97-106, 2012.

10. Інтелектуальний аналіз даних : навчальний посібник / А. О. Олійник, С. О. Субботін, О. О. Олійник. – Запоріжжя : ЗНТУ, 2012. с 90-91

11. Лекція з методу k-найближчих сусідів. [Електронний ресурс]. Режим доступу: http://om.univ.kiev.ua/users_upload/15/upload/file/pr_lecture_03.pdf

12. Опис методу опорних векторів [Електронний ресурс]. Режим доступу: https://ela.kpi.ua/bitstream/123456789/45721/1/NP_Osnovy_teorii_intelekt_analizu.pdf

13. Опис методу SMOTE [Електронний ресурс]. Режим доступу: https://loginom.ru/blog/imbalance-class

14. Опис методу RandomUnderSampler [Електронний ресурс]. Режим доступу:https://imbalancedlearn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html

15. Опис метрики вимірювання точності [Електронний ресурс]. Режим доступу: https://scikitlearn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html#sklearn.metrics.accuracy_score

16. Аналіз та класифікація методів виявлення вторгнень в інформаційну систему / В. В. Бєрковський, О. С. Безсонов // Системи управління, навігації та зв'язку. - 2017. - Вип. 3. - С. 57-62. - [Електронний ресурс] – Режим доступу: URL: http://nbuv.gov.ua/UJRN/suntz_2017_3_17

17. Опис програмного середовища «Анаконда» [Електронний ресурс]. Режим доступу: https://www.anaconda.com/

18. Опис бібліотеки Scikit-learn [Електронний ресурс]. Режим доступу: https://scikit-learn.org/stable/

19. Опис мови програмування Python [Електронний ресурс]. Режим доступу: https://disted.edu.vn.ua/courses/learn/7649

20. Опис командної оболонки Jupyter Notebook [Електронний ресурс]. Режим доступу: https://jupyter.org/

21. Опис фреймоворку React.js [Електронний ресурс]. Режим доступу: https://www.codecademy.com/learn/react

Додатки

Додаток А

Лістинг програми класифікатора

```
import os
from collections import defaultdict
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
dataset_root = 'datasets/nsl-kdd'
from collections import defaultdict
dataset_root = 'datasets/nsl-kdd'
category = defaultdict(list)
category['benign'].append('normal')
with open(dataset_root + '/training_attack_types.txt', 'r') as f:
    for line in f.readlines():
        attack, cat = line.strip().split(' ')
        category[cat].append(attack)
train_file = os.path.join(dataset_root, 'KDDTrain+.txt')
test_file = os.path.join(dataset_root, 'KDDTest+.txt')
col_names = np.array(header_names)

nominal_idx = [1, 2, 3]
binary_idx = [6, 11, 13, 14, 20, 21]
numeric_idx = list(set(range(41)).difference(nominal_idx).difference(binary_idx))
nominal_cols = col_names[nominal_idx].tolist()
binary_cols = col_names[binary_idx].tolist()
numeric_cols = col_names[numeric_idx].tolist()
category = defaultdict(list)
category['benign'].append('normal')
with open('datasets/nsl-kdd/training_attack_types.txt', 'r') as f:
    for line in f.readlines():
        attack, cat = line.strip().split(' ')
        category[cat].append(attack)
attack_mapping = dict((v,k) for k in category for v in category[k])
train_df = pd.read_csv(train_file, names=header_names)
train_df['attack_category'] = train_df['attack_type'] \
                            .map(lambda x: attack_mapping[x])
train_df.drop(['success_pred'], axis=1, inplace=True)
test_df = pd.read_csv(test_file, names=header_names)
test_df['attack_category'] = test_df['attack_type'] \
```

```
                        .map(lambda x: attack_mapping[x])
test_df.drop(['success_pred'], axis=1, inplace=True)
train_attack_types = train_df['attack_type'].value_counts()
train_attack_cats = train_df['attack_category'].value_counts()
test_attack_types = test_df['attack_type'].value_counts()
test_attack_cats = test_df['attack_category'].value_counts()
plot1=train_attack_types.plot(kind='barh', figsize=(20,10), fontsize=20)
plot2 = train_attack_cats.plot(kind='barh', figsize=(20,10), fontsize=30)
plot3=test_attack_types.plot(kind='barh', figsize=(20,10), fontsize=15)
plot4=test_attack_cats.plot(kind='barh', figsize=(20,10), fontsize=30)
info=train_df[binary_cols].describe().transpose()
train_df['su_attempted'].replace(2, 0, inplace=True)
test_df['su_attempted'].replace(2, 0, inplace=True)
train_df.groupby(['su_attempted']).size()
train_df.drop('num_outbound_cmds', axis = 1, inplace=True)
test_df.drop('num_outbound_cmds', axis = 1, inplace=True)
numeric_cols.remove('num_outbound_cmds')
train_Y = train_df['attack_category']
train_x_raw = train_df.drop(['attack_category','attack_type'], axis=1)
test_Y = test_df['attack_category']
test_x_raw = test_df.drop(['attack_category','attack_type'], axis=1)
combined_df_raw = pd.concat([train_x_raw, test_x_raw])
combined_df = pd.get_dummies(combined_df_raw, columns=nominal_cols, drop_first=True)
train_x = combined_df[:len(train_x_raw)]
test_x = combined_df[len(train_x_raw):]
# Store dummy variable feature names
dummy_variables = list(set(train_x)-set(combined_df_raw))
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, zero_one_loss, accuracy_score
classifier1 = DecisionTreeClassifier(random_state=17)
classifier1.fit(train_x, train_Y)
pred_y = classifier1.predict(test_x)
results = confusion_matrix(test_Y, pred_y)
error = zero_one_loss(test_Y, pred_y)
accur = accuracy_score(test_Y, pred_y)
print(results)
print(error)
print(accur)
from sklearn.neighbors import KNeighborsClassifier
classifier2 = KNeighborsClassifier(n_neighbors=1, n_jobs=-1)
classifier2.fit(train_x, train_Y)
pred_y = classifier2.predict(test_x)
```

```
results = confusion_matrix(test_Y, pred_y)
error = zero_one_loss(test_Y, pred_y)
accur = accuracy_score(test_Y, pred_y)
print(results)
print(error)
print(accur)
from sklearn.svm import LinearSVC
classifier3 = LinearSVC()
classifier3.fit(train_x, train_Y)
pred_y = classifier3.predict(test_x)
results = confusion_matrix(test_Y, pred_y)
error = zero_one_loss(test_Y, pred_y)
accur = accuracy_score(test_Y, pred_y)
print(results)
print(error)
print(accur)
from sklearn.preprocessing import StandardScaler
durations = train_x['duration'].values.reshape(-1, 1)
standard_scaler = StandardScaler().fit(durations)
scaled_durations = standard_scaler.transform(durations)
pd.Series(scaled_durations.flatten()).describe()
from sklearn.preprocessing import MinMaxScaler

min_max_scaler = MinMaxScaler().fit(durations)
min_max_scaled_durations = min_max_scaler.transform(durations)
pd.Series(min_max_scaled_durations.flatten()).describe()
from sklearn.preprocessing import RobustScaler
min_max_scaler = RobustScaler().fit(durations)
robust_scaled_durations = min_max_scaler.transform(durations)
pd.Series(robust_scaled_durations.flatten()).describe()
standard_scaler = StandardScaler().fit(train_x[numeric_cols])
train_x[numeric_cols] = \
   standard_scaler.transform(train_x[numeric_cols])
test_x[numeric_cols] = \
   standard_scaler.transform(test_x[numeric_cols])
train_Y_bin = train_Y.apply(lambda x: 0 if x is 'benign' else 1)
test_Y_bin = test_Y.apply(lambda x: 0 if x is 'benign' else 1)
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, zero_one_loss
classifier1 = DecisionTreeClassifier(random_state=17)
classifier1.fit(train_x, train_Y)
pred_y = classifier1.predict(test_x)
```

```
results = confusion_matrix(test_Y, pred_y)
error = zero_one_loss(test_Y, pred_y)
accur = accuracy_score(test_Y, pred_y)
print(results)
print(error)
print(accur)
from sklearn.neighbors import KNeighborsClassifier
classifier2 = KNeighborsClassifier(n_neighbors=1, n_jobs=-1)
classifier2.fit(train_x, train_Y)
pred_y = classifier2.predict(test_x)
results = confusion_matrix(test_Y, pred_y)
error = zero_one_loss(test_Y, pred_y)
accur = accuracy_score(test_Y, pred_y)
print(results)
print(error)
print(accur)
from sklearn.svm import LinearSVC
classifier3 = LinearSVC()
classifier3.fit(train_x, train_Y)
pred_y = classifier3.predict(test_x)
results = confusion_matrix(test_Y, pred_y)
error = zero_one_loss(test_Y, pred_y)
accur = accuracy_score(test_Y, pred_y)
print(results)
print(error)
print(accur)
```