ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти бакалавр

на тему: «Метод класифікації мережевого трафіку з використанням машинного навчання»

Виконав: здобувач вищої освіти
за освітньо-професійною програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
освітнього ступеня бакалавр
групи 126ІСТ_бд_2022[1](стн)
Постолакі М. О.
Керівник: Протас Н. М.
Рецензент: Муравльов В. В.

Полтава – 2024 року

ВСТУП

Актуальність. У світі стрімко йде зростання кількості засобів телекомунікації (Інтернет, ТБ, телефонія тощо.). Величезна кількість сервісів та програм використовують різні ресурси мережі [1]. Виникають ситуації, коли в єдиній системі є кілька типів трафіку, наприклад, пошуковий, платний або прямий, і виникає завдання, пов'язане з розробкою методів класифікації інтернет-трафіку. Дані методи допомагають швидше виявляти шкідливий трафік і запобігати DDoS-атаки, які можуть спричинити непоправну шкоду обчислювальній системі [2].

Згідно зі статистикою, кількість DDoS-атак у четвертому кварталі 2023 року зростала з кожним днем. Виражений пік припав на кінець грудня (31 грудня) і складав приблизно 1655 атак [3]. Лідером серед типів DDoS-атак є SYN-флуд. Його частка складає 78,28%. На другому місці знаходяться UDP - атаки (15,17%). Атаки через протоколи TCP та GRE мають 5,47% та 0,69% відповідно. На останньому місці в четвертому кварталі знаходиться HTTP із часткою 0,39%. Виходячи зі статистики, можна зробити висновок, що перша половина четвертого кварталу 2023 року пройшла неспокійно. Далі йшло бурхливе зростання DDoS -атак і в останній день кварталу досягло свого піку. Найпопулярнішим типом DDoS -атак залишається SYN -флуд [4].

Метою кваліфікаційної роботи є розробка методу класифікації мережевого трафіку з використанням машинного навчання з учителем. Для досягнення поставленої мети необхідно вирішити такі завдання:

1) провести огляд наукової літератури у предметній галузі;

2) підготувати навчальну та тестову вибірки;

3) застосувати методи машинного навчання з учителем, провести їх навчання та тестування;

4) розробити програму для класифікації трафіку;

5) протестувати роботу програми.

Об'єкт дослідження – процеси функціонування систем протидії мережевим атакам та аналізу мережевого трафіку.

Предмет дослідження – проєктування методів машинного навчання для класифікації мережевого трафіку з метою виявлення атак.

Методи досліджень – проведені в роботі дослідження базуються методах системного аналізу, проєктування та прототипування методів і класів програмних додатків, порівняння різних методів машинного навчання за їхньою ефективністю та точністю.

Інформаційна база – Інтернет-ресурси та друковані видання, що місять інформацію про методи машинного навчання та про класифікатори мережевого трафіку, мережеві стандарти, довідкова інформація про використані програмні компоненти архітектури.

Практична значущість – на основі запропонованого підходу розроблено ефективну та надійну систему класифікації мережевого трафіку, що відповідає вимогам користувачів та забезпечує високу продуктивність і зручність використання.

Результати роботи апробовані в рамках наукової конференції здобувачів вищої освіти за результатами науково-дослідної роботи у 2023-2024 роках.

Структура кваліфікаційної роботи логічно пов'язана з задачами досліджень і містить перелік умовних позначень, вступ, три розділи основної частини, висновки, список використаних джерел. Загальний обсяг текстової частини кваліфікаційної роботи складає 48 сторінок формату А4. Вона містить 19 рисунків і 19 таблиць.

РОЗДІЛ 1
АНАЛІЗ ЗАДАЧ ПРОТИДІЇ АТАКАМ ТА ВИМОГ ДО МЕРЕЖЕВОГО ТРАФІКУ

1.1 Аналіз задач класифікації мережевого трафіку

Ідентифікація та класифікація мережевого трафіку використовуються для вирішення таких завдань, як визначення пріоритетів при формуванні смуги пропускання для окремих видів трафіку, встановлення правил управління мережею, забезпечення безпеки мережі та діагностичний моніторинг. У загальному вигляді завдання класифікації мережевого трафіку можна сформулювати наступним чином: одержання на вхід деяких характеристик мережевого трафіку з визначенням на виході класу, до якого цей вид трафіку належить.

На даний момент актуальність цього завдання значно зросла у зв'язку з розширенням сфери його застосування, що зараз включає як системи застосування політик, так і сферу інформаційної безпеки. Практично будь-яка система аналізу трафіку включає компонент класифікації. Перед тим, як класифікувати трафік мережі, необхідно ідентифікувати його характеристики. Ці характеристики можуть бути визначені в результаті аналізу властивостей, що характеризують інтернет-трафік. До них можуть належати різні особливості загального мережевого трафіку.

Для вирішення відомих проблем класифікації широкого поширення набули технології машинного навчання, які виявилися найефективнішими. Такі методи дозволяють розроблюваній системі легко адаптуватися до природи Інтернет-ресурсів, що постійно змінюється, і враховувати специфіку аналізу мережевого трафіку. Завдяки машинному навчанню, системи можуть автоматично оновлювати свої моделі та підвищувати точність класифікації. Це забезпечує більш ефективне управління мережею та підвищує рівень її захисту від загроз. Впровадження таких технологій є критично важливим для сучасних мережевих інфраструктур.

1.2 Аналіз методів класифікації при використанні машинного навчання з учителем

Розглянемо множину об'єктів (ситуацій) та множину можливих відповідей (відгуків, реакцій). Існує певна залежність між відповідями та об'єктами, яка невідома. Відомо лише кінцеву сукупність прецедентів – пару «об'єкт, відповідь», яка називається навчальною вибіркою. На основі цих даних потрібно відновити залежність, тобто побудувати алгоритм, здатний для будь-якого об'єкта видавати досить точну відповідь. Для вимірювання точності відповідей вводиться функціонал якості. Під учителем розуміється або сама навчальна вибірка, або той, хто визначив правильні відповіді для заданих об'єктів.

В алгоритмі K-Nearest Neighbors (k-NN) кожній точці даних (об'єкту) призначається клас, який є найпоширенішим серед сусідів даного об'єкта, класи яких вже відомі. Як функцію відстані між двома точками в просторі найчастіше використовують евклідову відстань. Нехай є дві точки в n-вимірному просторі ознак: $x_1 = (x_{11}, ..., x_{1n})$ і $x_2 = (x_{21}, ..., x_{2n})$. Евклідова відстань між цими точками визначається наступним чином:

$$d(x_1, x_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}. \quad (1.1)$$

Число найближчих сусідів k та метрика є ключовими компонентами алгоритму k-NN.

Наївний байєсовський класифікатор (Naive Bayes) є імовірнісним класифікатором, заснованим на теоремі Байєса, з припущенням про незалежність ознак об'єкта. Нехай $\{c_1, ..., c_k\}$ – фіксована безліч класів. Потрібно знайти клас c_j, для якого його ймовірність для об'єкта максимальна max $p(c_j|y)$. Імовірність приналежності $p(c_j|y)$ об'єкта до класу обчислюється за допомогою формули Байєса для умовних ймовірностей:

$$p(c_j|y) = \frac{p(y|c_j) \cdot p(c_j)}{p(y)}, \quad (1.2)$$

де $p(c_j|y)$ – ймовірність отримання класу c_j незалежно від спостережуваних даних;

$p(y|c_j)$ – ймовірність зустріти об'єкт серед об'єктів класу c_j.

Завдання зводиться до оцінки розподілу об'єктів для класів $p(c_j)$ та розподілів $p(y|c_j)$ на основі тренувальних даних.

Метод опорних векторів (Support Vector Machine, SVM) розділяє точки даних гіперплощиною з розмірністю n−1. Мета полягає в тому, щоб розділити точки даних гіперплощиною, яка має максимальну відстань до найближчих точок з кожного боку. Гіперплощина може бути представлена як множина точок X, що задовольняють умові $\omega \cdot x+b=0$, де ω – нормальний вектор гіперплощини, а b – зміщення. Лінійний SVM вирішується шляхом формулювання задачі квадратичної оптимізації.

Дерево прийняття рішень (Decision Tree) є класифікатором, побудованим на основі правил виду «якщо...то...», упорядкованих в деревоподібну ієрархічну структуру. Процес рекурсивного розбиття вихідної множини об'єктів на підмножини здійснюється за допомогою вирішальних правил, які перевіряють значення атрибутів. На кожному кроці алгоритм обирає атрибут, для якого приріст інформації максимальний.

Логістична регресія (Logistic Regression) вивчає зв'язок між категоріально залежною змінною та набором незалежних змінних. Вона використовується, коли залежна змінна має лише два значення, такі як «0 і 1» або «Так і Ні». Логістична регресія передбачає, що незалежні змінні розподілені нормально, як і в дискримінантному аналізі. Логітне перетворення записується як:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right), \quad (1.3)$$

де p – ймовірність позитивної реакції.

Логістичне перетворення є зворотним до логіту і записується як:

$$p = \frac{1}{1+e^{-\text{logit}(p)}}. \tag{1.4}$$

Аналіз методів класифікації вказує на їх різноманітність та можливість застосування до різних задач. Вибір конкретного методу залежить від характеристик даних та вимог до точності класифікації. Використання комбінованих підходів може значно покращити якість класифікації та забезпечити більш стійкі результати.

1.3 Порівняння існуючих систем виявлення атак, які використовують методи машинного навчання

Кількість IoT-пристроїв щороку зростає. Проте багато з них мають значні вразливості, такі як незашифрована передача даних та застаріла прошивка Linux. У жовтні 2016 року відбулася масштабна DDoS-атака «Mirai», в результаті якої було заражено понад сто тисяч пристроїв. Багато популярних вебсайтів, таких як Amazon, Netflix, Twitter, CNN та PayPal, були недоступні протягом кількох годин. Ця атака стала приводом для розробки нових методів виявлення та блокування аномального трафіку. Нещодавні дослідження показали, що машинне навчання є досить перспективним у цьому напрямі. У роботі [6] автори представили своє програмне забезпечення для виявлення мережевих атак, засноване на машинному навчанні.

Архітектура ПЗ, представлена на рис. 1.1, складається з чотирьох блоків, перелічених нижче.

1. Захоплення трафіку. Процес захоплення трафіку записує такі дані: вихідна IP-адреса, вихідний порт, кінцева IP-адреса, кінцевий порт, розмір пакета, тимчасова мітка всіх IP-адрес, що відправляються з пристроїв.

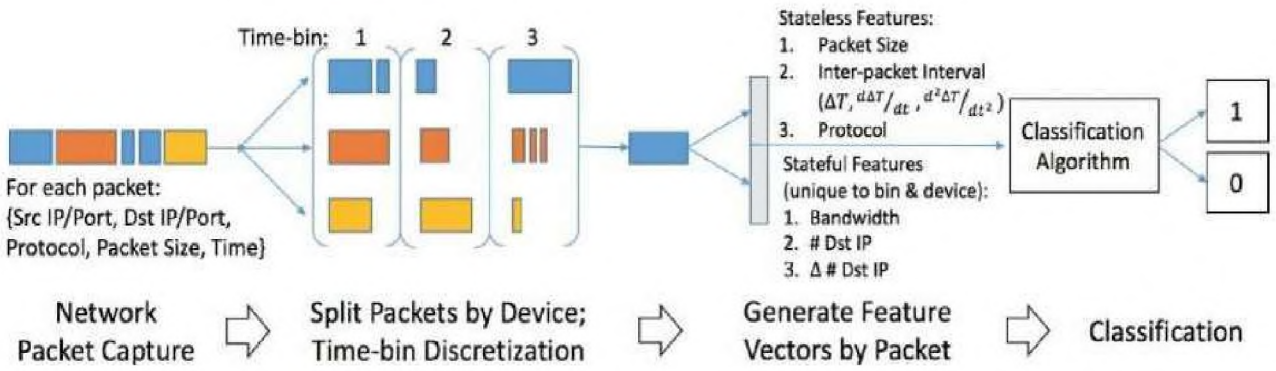

Рисунок 1.1 – Типова архітектура ПЗ для виявлення мережевих атак [6]

2. Групування пакетів за пристроями та часом. Пакети від кожного пристрою відокремлюються IP-адресою. Далі пакети діляться за часовими мітками, записаними в проміжний блок.

3. Вилучення ознак. Відстежувані та невідстежувані ознаки генеруються для кожного пакета на основі знань домену про поведінку IoT-пристроїв.

4. Бінарна класифікація. Такі класичні алгоритми машинного навчання, як K-Nearest Neighbors, Random Forest, Decision Tree та Support Vector Machine, можуть відрізнити звичайний трафік від аномального з дуже високою точністю.

Дослідження вчених [7] показали, що неефективність виявлення та пом'якшення шкоди від DDoS-атак безпосередньо пов'язана з постійними помилками конфігурації, марно витраченим часом, а також через відсутність інструментів, які стежать за динамікою мережі без постійного втручання людини. Це призвело до використання автономних рішень, здатних працювати на основі поведінкових та характеристичних аналізів трафіку. У цьому контексті прийняття рішень з використанням методів машинного навчання вирізнялося високою гнучкістю процесу класифікації, що покращувало виявлення шкідливого трафіку.

Система з розробленим підходом збирає зразки мережевого трафіку та класифікує їх. Повідомлення про атаки передаються через хмарну платформу для зручного керування трафіком систем безпеки. Весь процес продемонстровано на рис. 1.2.

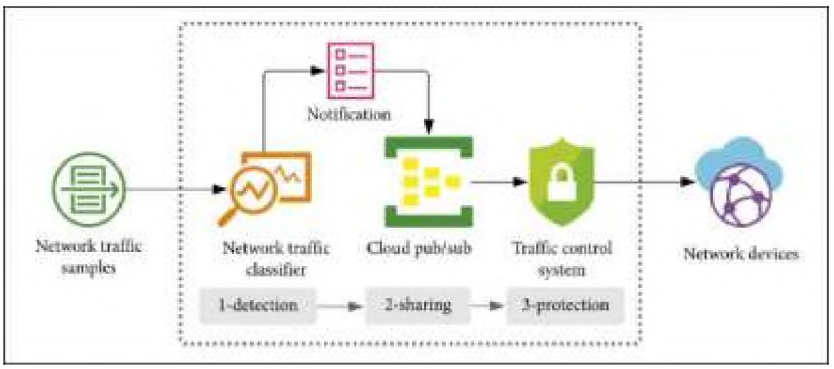

Рисунок 1.2 – Сценарій роботи системи інтелектуального виявлення мережевих атак

Ядро системи виявлення складається із набору сигнатурних даних (SDS, Signature Dataset) та алгоритму машинного навчання (MLA, Machine Learning Algorithm). На рис. 1.3 показано найважливіші етапи роботи системи виявлення.

Звичайний трафік та сигнатури DDoS-атак були вилучені, позначені та збережені в базі даних. Потім було створено набір сигнатурних даних із використанням методів вибору об'єктів. Після цього був відібраний найточніший алгоритм машинного навчання, навчений та завантажений у систему.

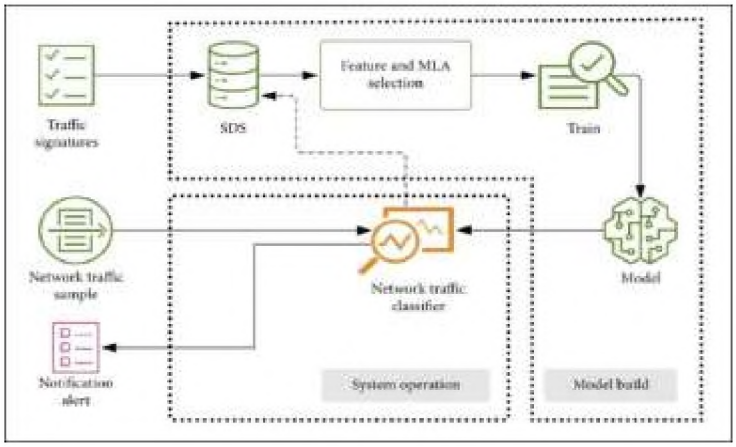

Рисунок 1.3 – Компоненти системи виявлення мережевих атак [7]

Архітектура системи була розроблена для роботи із зразками мережевого трафіку, що надаються стандартними промисловими протоколами, зібраними з мережевих пристроїв. Немарковані зразки приймаються, групуються в потокових таблицях та записуються в буфер приймача. Таким чином, коли довжина таблиці досягає або перевищує еталонне значення, вони передаються

класифікатору для маркування, як показано на рис. 1.4. Якщо термін дії таблиці потоків спливає, вона може бути оброблена повторно. Маленькі таблиці потоків зустрічаються частіше при нижчих частотах дискретизації або деяких типах DDoS-атак, наприклад, атаках SYN flood.

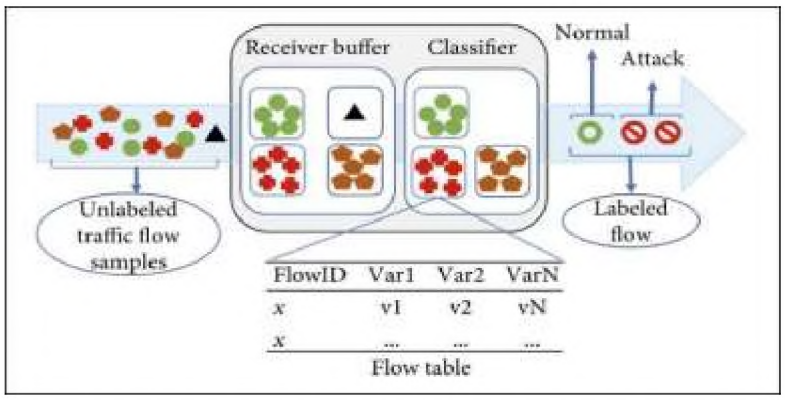

Рисунок 1.4 – Робота схеми класифікації трафіку

Повний алгоритм роботи системи виявлення представлений на рис. 1.5. Під час кожного циклу процесу виявлення зразки трафіку приймаються і зберігаються в таблиці потоків.

Для кожного нового потоку обчислюється унікальний ідентифікатор (FlowID), який враховує 5 атрибутів (src_IP, dst_IP, src_port, dst_port і transport_protocol) на кроках 1 і 2. Якщо це новий потік, тобто не існує іншої таблиці потоку з тим самим FlowID, таблиця потоку записується в буфер загальної пам'яті. В іншому випадку, якщо існує таблиця потоків з тим самим FlowID, дані нового потоку об'єднуються з даними в існуючій таблиці потоків на кроках 3 і 4. Після операції злиття, якщо довжина таблиці перевищує або дорівнює еталонному значенню, таблиця потоків класифікується, і якщо вона виявлена як атака, видається повідомлення. В іншому випадку вона знову записується в буфер загальної пам'яті.

На кроці 7 відбувається пошук прострочених таблиць потоків у загальному буфері, тобто таблиць потоків, які перевищують встановлений час дії системи. Для кожної таблиці потоку з простроченим терміном дії система перевіряє її довжину. Якщо довжина таблиці потоків менша або дорівнює мінімальному

еталонному значенню, ця таблиця потоків обробляється на кроці 8. Новий FlowID обчислюється за допомогою 3 атрибутів (src_IP, dst_IP і transport_protocol), після чого таблиця потоків повертається до кроків 3 і 4.

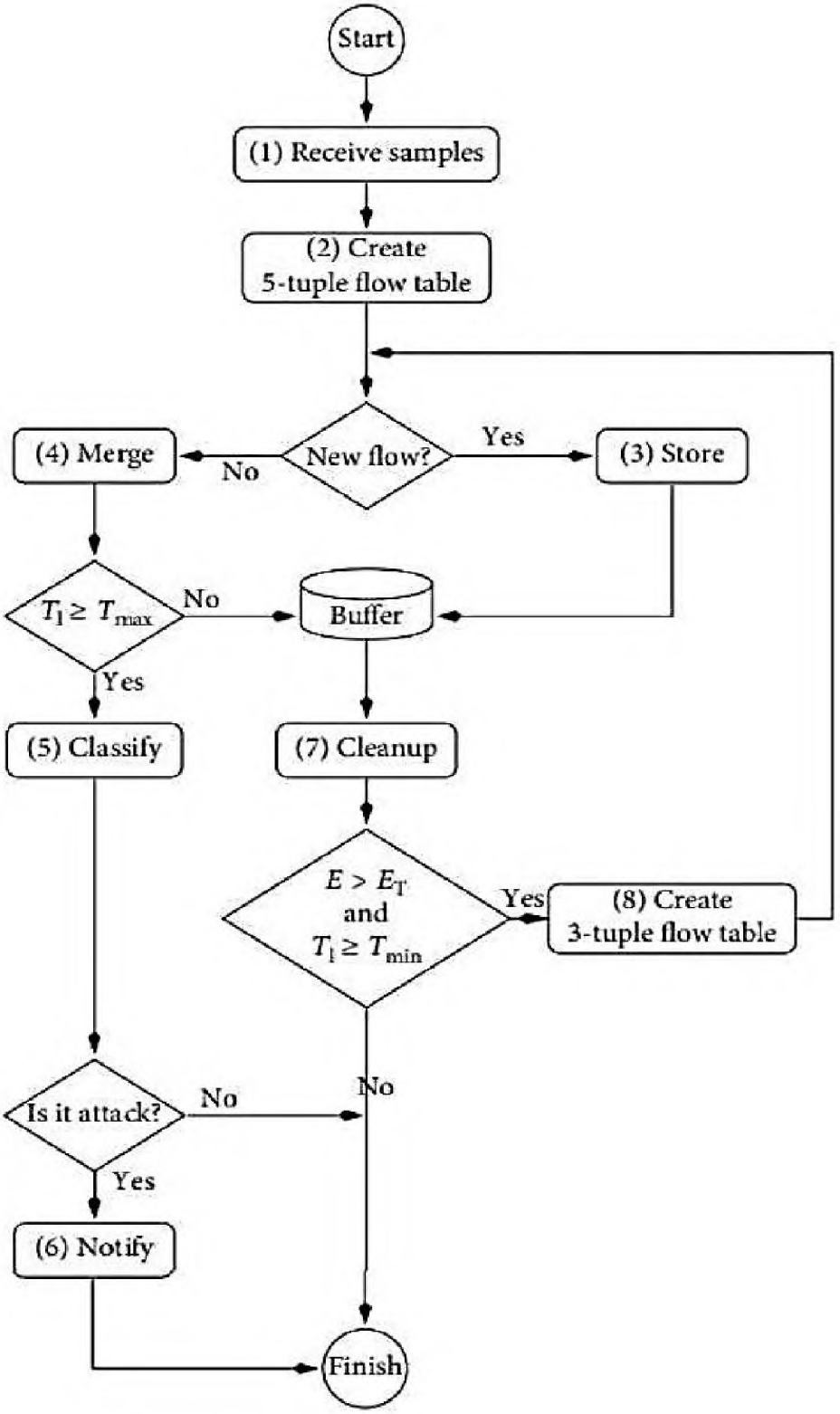

Рисунок 1.5 – Алгоритм системи виявлення мережевих атак [7]

Дослідження «Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models» [9] фокусується на DDoS-атаках в архітектурі SDN (Software-Defined Networking) і пропонує моделі машинного навчання, які підтримуються методами вибору ознак для виявлення атак. Метою є розробка системи виявлення DDoS-атак, що базується на машинному навчанні з високою ефективністю для архітектури SDN. Виявлення DDoS-атак на контролер SDN, розташований у площині даних, є дуже важливою вимогою для забезпечення безперервності роботи мережі. Якщо на контролері виявлено атакуючий трафік, йому буде простіше створити нові правила для таблиці потоків перемикачів у площині даних, щоб запобігти атаці. Це надає значну перевагу в запобіганні атакам. Автори пропонують використовувати методи вибору ознак за допомогою моделей машинного навчання для виявлення DDoS-атак. Вони вважають, що їхній підхід зробить значний внесок у ефективне виявлення DDoS-атак у SDN.

На рис. 1.6 показано етапи процесу застосування методів вибору ознак та моделей машинного навчання. Об'єкти в наборі даних, використовуваному в дослідженні, були отримані з основних метрик, важливих для безперервності архітектури SDN. Дані про атаки, отримані в результаті DDoS-атак, були класифіковані за допомогою методів машинного навчання. Об'єкти в наборі даних були вибрані з використанням методів вибору ознак та характеристик самого набору даних. Було досліджено класифікатори отриманого набору ознак.

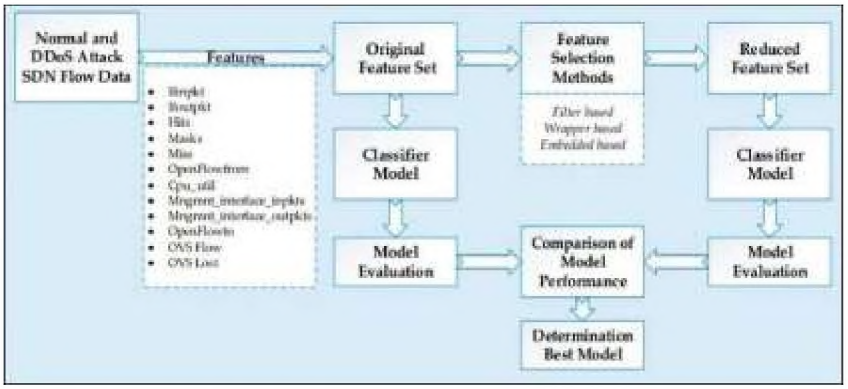

Рисунок 1.6 – Етапи процесу застосування методів вибору функцій та моделей машинного навчання

Широкий спектр атак загрожує промисловим, науковим та урядовим організаціям. Ці атаки часто призводять до втрати важливої інформації та впливають як на споживачів, так і на підприємства. На сьогоднішній день більшість підприємств впроваджують численні засоби контролю безпеки, а також застосовують кращі практики, такі як використання служб оперативної інформації про загрози та сканування кінцевих точок для захисту від кіберзагроз. Інструменти моніторингу використовуються у більшості організацій або на мережевому рівні (наприклад, системи виявлення мережевих вторгнень, вебпроксі, брандмауери), або на кінцевих вузлах (наприклад, антивірусне програмне забезпечення).

З появою журналів безпеки, зібраних великими підприємствами, машинне навчання стало важливим оборонним інструментом для виявлення дедалі витонченіших кібератак. Воно має великий потенціал у прискоренні виявлення шкідливого програмного забезпечення, але ці алгоритми також мають певні недоліки.

Зокрема, Sommer та Paxson у роботі «On Designing Machine Learning Models for Malicious Network Traffic Classification» [10] звернули увагу на труднощі використання машинного навчання в оперативних налаштуваннях для кібербезпеки. Основні обмеження, які вони виявили, перераховані нижче:

1. Машинне навчання ефективне для задач з маркованими даними, тоді як у сфері кібербезпеки більшість даних не має маркування.

2. Помилки машинного навчання (особливо помилкові спрацьовування) мають високу вартість, оскільки сигнали тривоги мають бути розслідувані аналітиками безпеки.

3. Мережевий трафік характеризується великою різноманітністю за нормальних умов.

4. Проведення обґрунтованих оцінок зазвичай утруднено через відсутність стандартів.

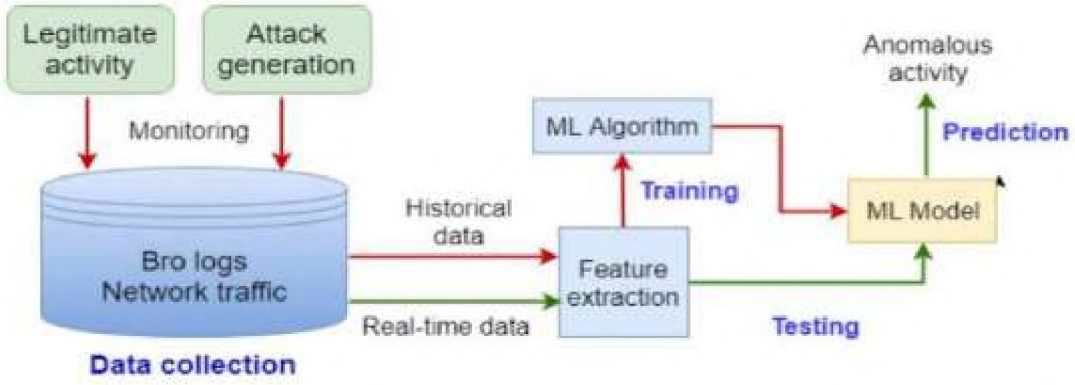

Рисунок 1.7 – Система класифікації трафіку, запропонована в [10]

Автори представляють розроблену системну архітектуру, представлену на рис. 1.7. Їх система обробляє мережеві журнали, зібрані під час атаки. Після збору даних їх готують до навчання. У цій системі використовується ряд алгоритмів класифікації для навчання класифікатора та оптимізації за стандартними метриками, такими як точність, точність (precision), відгук (recall), F1 та AUC.

1.4 Порівняння існуючих класифікаторів, які використовуються у методах машинного навчання

Для збору нормального трафіку автори [6] підключили три IoT-пристрої і протягом деякого часу записували PCAP (Packet Capture) файли, що реєструють всі пакети, відправлені за цей період. Аномальним трафіком були три види DDoS-атак: TCP SYN flood, UDP flood та HTTP GET flood.

У підсумку автори [6] протестували п‘ять алгоритмів машинного навчання для класифікації трафіку: K-Nearest Neighbors, Random Forest, Decision Tree, Support Vector Machine та чотиришарову нейронну мережу. Усі моделі машинного навчання були реалізовані з використанням бібліотеки Python – scikit-learn, за винятком нейронної мережі, яка була реалізована з використанням бібліотеки Keras. Точність класифікаторів варіювалася від 0,91 до 0,99, причому

атакуючих пакетів було більше, ніж звичайних (таблиця 1.1). Найкраще себе показав алгоритм Random Forest, найгіршим став алгоритм Support Vector Machine. Нейронна мережа показала непогані результати, але їх можна покращити шляхом збільшення кількості даних для навчання.

Таблиця 1.1 – Результати оцінювання точності класифікаторів за [6]

	KN	LSVM	DT	RF	NN
Precision (Normal)	.998	.992	.996	.999	.983
Precision (Attack)	.999	.991	.999	.999	.999
Recall (Normal)	.993	.870	.993	.998	.989
Recall (Attack)	.999	.999	.999	.999	.998
Fl (Normal)	.995	.927	.994	.998	.986
Fl (Attack)	.999	.995	.996	.999	.999
Accuracy	.999	.991	.999	.999	.999

Дані, використані в дослідженні [7], є вторинними, тобто зібраними іншими дослідниками. Є багато онлайн-банків, які надають мережевий трафік для безпосереднього використання. Вибір ознак є важливим кроком у процесі розпізнавання образів і визначенні найменшого набору змінних, здатного ефективно описувати набір класів. Декілька методів вибору змінних доступні в літературі та реалізовані в програмних бібліотеках, таких як scikit-learn та інших. У цій роботі відбір змінних проводився у два етапи. По-перше, рекурсивне усунення ознак з перехресною валідацією було використано з деякими алгоритмами машинного навчання, які широко використовуються в науковій літературі, наприклад, Random Forest, Logistic Regression, AdaBoost, Stochastic Gradient Descent, Decision Tree та Perceptron. Random Forest показав вищу точність, використовуючи 28 параметрів, тоді як AdaBoost використав сім параметрів, але показав нижчу точність, як зазначено в табл. 1.2.

Таблиця 1.2 – Результати оцінювання точності класифікаторів за [7]

#	MLA	No. of features	Accuracy
1	RF	28	0.996010
2	DTree	25	0.994182
3	LR	26	0.972327
4	SGD	16	0.969474
5	Perceptron	28	0.937256
6	AdaBoost	7	0.931131

У дослідженні «A study for DDOS attack classification method» [8] автори порівнюють ефективність трьох алгоритмів машинного навчання для класифікації DDoS-атак: Decision Tree, Naive Bayes та штучної нейронної мережі. Вибір цих трьох алгоритмів обумовлений не тим, що вони кращі за інші алгоритми, а через схожість характеристик. Це дослідження показало, який з алгоритмів найбільш підходить для класифікації та виявлення DDoS-атак.

У таблиці 1.3 показано порівняння результатів трьох методів машинного навчання. Результати порівняння також включають дані інших дослідників. Видно, що метод класифікації, заснований на штучних нейронних мережах, має кращі показники ефективності порівняно з іншими алгоритмами.

Таблиця 1.3 – Результати оцінювання точності класифікаторів за [8]

Performance Metrics	Decision Tree (%)	Naive Bayes (%)	Artificial Neural Network (%)
True Positive (TP)	84.0	76.6	84.5
False Positive (FP)	22.2	35.3	22.8
Precision	83.9	78.9	84.6
Recall	84.0	76.6	84.5
F-Measure	83.9	74.8	84.2
ROC/AUC	84.3	81.3	86.8

Справжнє позитивне значення (TP, True Positive) становить 84,5%, хибне позитивне значення (FP, False Positive) складає 22,8%, точність досягає 84,6%, відгук становить 84,5%, f-міра дорівнює 84,2%, а площа під кривою ROC (AUC) становить 86,8% (рис. 1.8).

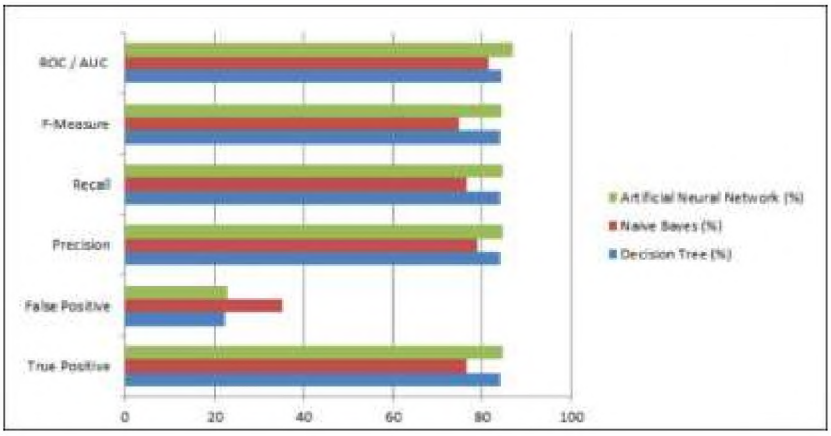

Рисунок 1.8 – Порівняння ефективності алгоритмів виявлення мережевих атак

Це показує, що штучна нейронна мережа може бути ефективно використана для класифікації DDoS-атак з кількох причин: по-перше, шаблон DDoS-атаки непередбачуваний, постійно змінюється і не має чітких параметрів.

По-друге, зі збільшенням кількості пакетів даних зростає різницева швидкість. У цьому випадку два інші методи також справляються з цією задачею, але оптимальним вибором є використання штучної нейронної мережі. Це не означає, що інші два методи неефективні, але для класифікації DDoS-атак, виходячи з результатів порівняння, метод штучних нейронних мереж є найкращим за точністю.

У результаті проведених досліджень [9] було проведено порівняння ефективності алгоритмів із застосуванням методу вибору ознак і без нього (табл. 1.4).

Таблиця 1.4 – Порівняльна таблиця точності класифікаторів із застосуванням методу вибору функцій і без нього за [9]

Classifier	Method	No. of Features	Accuracy	Sensitivity	Specificity	Precision	Fl_Score
SVM	-	12	92.11%	88.71%	96.93%	91.42%	89.91%
	Filter	10	92.46%	89.13%	97.02%	92.02%	90.43%
	Wrapper	8	92.15%	90.20%	97.26%	90.23%	90.21%
	Embedded	10	92.46%	90.73%	97.41%	90.49%	90.60%
KNN	-	12	95.67%	93.87%	98.01%	97.05%	95.30%
	Filter	6	97.15%	95.88%	98.68%	98.10%	96.92%
	Wrapper	6	98.30%	97.73%	99.45%	97.72%	97.70%
	Embedded	8	96.30%	94.95%	98.85%	95.09%	94.80%
ANN	-	12	91.07%	87.27%	96.58%	89.89%	88.45%
	Filter	6	92.28%	89.02%	96.99%	91.62%	90.20%
	Wrapper	10	91.44%	87.82%	97.31%	88.11%	87.89%
	Embedded	6	92.09%	88.91%	97.42%	89.22%	89.06%
	-	12	94.48%	91.77%	98.29%	92.94%	91.79%
	Filter	8	95.70%	93.49%	98.65%	95.07%	93.60%
	Wrapper	10	94.87%	92.05%	98.43%	93.29%	92.01%
	Embedded	10	95.09%	93.34%	98.45%	93.44%	93.18%

На основі проведеного у [9] дослідження можна зробити висновок, що використання методів вибору ознак значно підвищує ефективність класифікаторів для виявлення DDoS-атак у архітектурі SDN. Зокрема,

алгоритми, що використовують методи вибору ознак, демонструють кращі показники точності, чутливості та специфічності порівняно з тими, які використовуються без вибору ознак. Це підтверджує важливість оптимізації вибору ознак для покращення продуктивності системи виявлення атак.

У статті [10] автори описують деякі конкретні керівні принципи та рекомендації щодо використання контрольованого машинного навчання у кібербезпеці. Як приклад розглянуто проблему виявлення бот-мережі за даними мережевого трафіку. Вони використовують загальнодоступний набір даних (CTU-13), який включає мережевий трафік, зібраний з атак на Чеський університет CTU у 2011 році. Набір даних містить 13 сценаріїв, кожен з яких включає легітимний трафік, а також різні атаки, такі як спам, сканування портів та DDoS.

Результати за трьома класифікаторами наведені в табл. 1.5, а криві precision-recall наведені на рис. 1.9. Обидва ансамблеві методи працюють краще, ніж модель логістичної регресії, при цьому метрика F1 за всіма сценаріями досягає від 0.94 до 0.98. Різниця між випадковим лісом та градієнтним бустингом мінімальна, вони обидві є ефективними класифікаційними моделями.

Таблиця 1.5 – Результати оцінювання точності класифікаторів за [10]

Model	Training Scenarios	Testing Scenario	Prec.	Recall	Fl	AUC
Logistic	2,9	1	0.90	0.90	0.90	0.94
Regression	1,9	2	0.98	0.95	0.97	0.99
	1,2	9	0.97	0.87	0.92	0.96
Random	2,9	1	0.99	0.97	0.98	0.99
Forest	1,9	2	0.95	0.96	0.95	0.98
	1,2	9	1	0.90	0.94	0.96
Gradient	2,9	1	1	0.97	0.98	0.99
boosting	1,9	2	1	0.92	0.96	0.99
	1,2	9	1	0.87	0.93	0.95

У наші дні мережева безпека відіграє надзвичайно важливу роль у системах передачі даних та мереж. Кількість шкідливих користувачів та зловмисників постійно зростає, особливо у сфері бізнесу та освіти. Традиційні апаратні брандмауери можуть бути заблоковані постачальником і є дорогими. У

системі, описаній у «Performance Analysis of Traffic Classification with Machine Learning» [11], використовується програмний брандмауер з відкритим вихідним кодом, який зменшує складність, час, є адаптивним у конфігурації та більш економічним. Однак при встановленні правил на брандмауері можуть виникати помилки конфігурації, які створюють вразливості. Основними факторами системи захисту є надійність та стійкість. Ця система фокусується на системі виявлення вторгнень (IDS), а не на брандмауері.

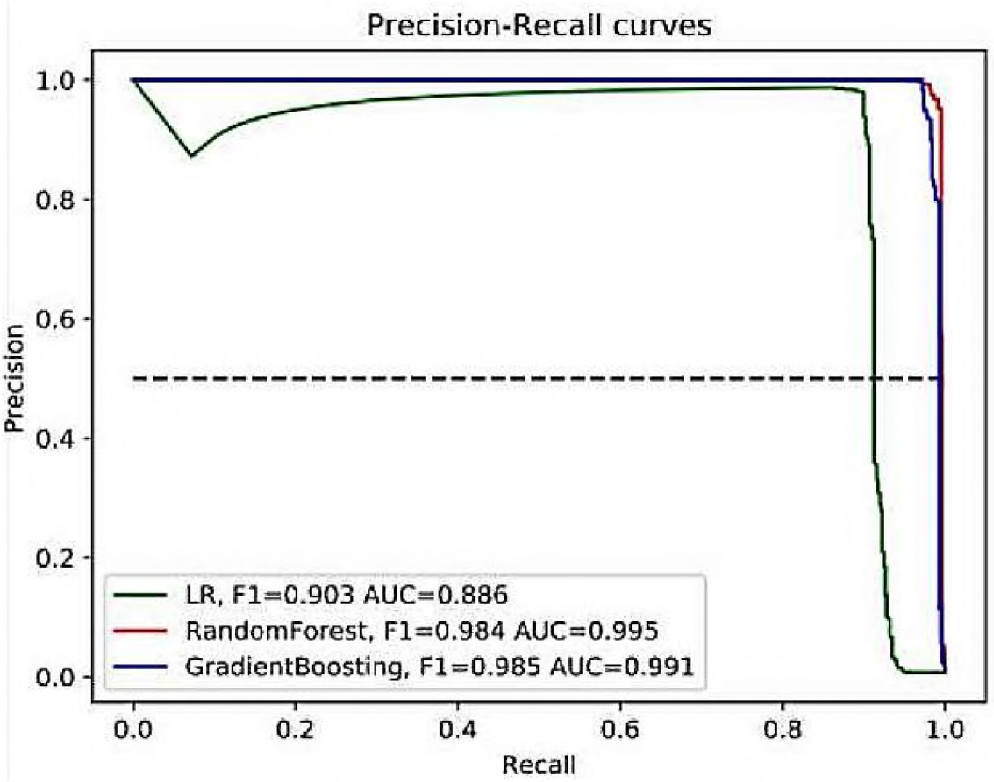

Рисунок 1.9 – Крива precision-recall

IDS збирає різноманітний вхідний трафік даних та аналізує, які дані належать до яких видів атак. Система виявлення вторгнень має два основні типи: сигнатурний, який виявляє шкідливі атаки за певними байтовими шаблонами, та на основі аномалій, який використовує статистичний моніторинг мережевого трафіку. У дослідженні використовується відкрите джерело Snort-IDS для аналізу протоколів і виявлення вмісту, що збігається. Виявлення вторгнень необхідне як додатковий бар'єр для систем захисту і надає важливі дані для вжиття контрзаходів.

Основними напрямками досліджень у цій роботі є:

1. Створення правил міжмережевого екрану на програмних інтерфейсах міжмережевого екрану, таких як вихідний трафік, доступ до IPCop, внутрішній трафік, переадресація портів та доступ до зовнішнього IPCop з урахуванням сервісів.

2. Забезпечення політики IDS на основі сигнатур та доказів за допомогою машинного навчання.

3. Реалізація запропонованого набору даних для покращення продуктивності системи.

Набір даних CICIDS-2017, що використовується в даному дослідженні, містить велику кількість трафіку і 78 ознак для виявлення аномалій. Він складається з двох видів трафіку – нормального (легітимного) та атак, які є складними типами та покращують продуктивність IDS на цьому наборі даних. CICIDS-2017 включає 7 типів атак, таких як Brute force, PortScan, Botnet, Dos, DDoS, Web, Infiltration. У цьому експерименті було використано 26 параметрів набору даних, які найефективніше допомагають класифікувати трафік (табл. 1.6).

При атаці DoS, алгоритм Random Tree має найвищу точність 99,8%, за ним слідують Logistic та JRiP із 99,6%. При атаці PortScan класифікатори J48 і Random Tree мають найнижчу точність 84,8% і 91,57% відповідно (табл. 1.7).

Таблиця 1.6 – Список відібраних параметрів набору даних CICIDS2017

No.	Feature Name Extracted from CICIDS2017		
1	Destination Port	14	Bwd Packet Length Std
2	Flow Duration	15	Bwd Header Length
3	Total Fwd Packets	16	Fwd Packetss
4	Total Backward Packets	17	Bwd Packet's
5	Total Length of Fwd Packets	18	Min Packet Length
6	Total Length of Bwd Packets	19	Max Packet Length
7	Fwd Packet Length Max	20	Packet Length Mean
8	Fwd Packet Length Min	21	Packet Length Std
9	Fwd Packet Length Mean	22	Packet Length Variance
10	Fwd Packet Length Std	23	Average Packet Size
11	Bwd Packet Length Max	24	Avg Fwd Segment Size
12	Bwd Packet Length Min	25	Avg Bwd Segment Size
13	Bwd Packet Length Mean	26	Fwd Header Length

Таблиця 1.7 – Результати оцінювання точності класифікаторів за для типів атак DoS та PortScan [6]

Detection Classifier	Accuracy %	
	DoS	PortScan
Logistic	99.598	99.89
SVM	99.198	100
J48	99.1	84.766
JRiP	99.599	100
Random Tree	99.799	91.57
Multiclass Classifier	99.499	99.899

Використання програмного брандмауера з відкритим вихідним кодом та системи виявлення вторгнень забезпечує ефективний захист мережі від різних типів атак. Використання алгоритмів машинного навчання та аналізу мережевого трафіку значно підвищує точність виявлення атак. У майбутніх дослідженнях варто зосередитися на подальшій оптимізації алгоритмів та розширенні наборів даних для покращення захисту від кіберзагроз.

РОЗДІЛ 2
ПРОЄКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ КЛАСИФІКАЦІЇ МЕРЕЖЕВОГО ТРАФІКУ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

2.1 Особливості бібліотек для створення алгоритмів на основі машинного навчання

Бібліотека scikit-learn [12] є найпоширенішим вибором для вирішення завдань класичного машинного навчання. Однією з основних переваг цієї бібліотеки є те, що вона працює на основі декількох поширених математичних бібліотек, таких як NumPy і SciPy, і легко інтегрується з ними. Scikit-learn широко використовується для промислових систем, у яких застосовуються алгоритми класичного машинного навчання, для досліджень, а також для новачків, які роблять перші кроки у галузі машинного навчання. Ця бібліотека спеціалізується на алгоритмах машинного навчання для вирішення завдань навчання з учителем: класифікація (прогнозування ознаки, множина допустимих значень якої обмежена) та регресія (прогнозування ознаки з числовими значеннями).

Pandas [13] – це високорівнева бібліотека для аналізу даних на Python. В екосистемі Python, pandas є найпросунутішою та швидко розвивається бібліотекою для обробки та аналізу даних. Ключові особливості pandas:

1. Швидкий та ефективний об'єкт DataFrame з індивідуальною індексацією за замовчуванням.

2. Інструменти для завантаження даних у об'єкти даних у пам'яті з різних форматів файлів.

3. Вирівнювання даних та інтегрована обробка відсутніх даних.

4. Зміна форми та поворот наборів даних.

5. Мітка нарізки, індексація та підмножина великих наборів даних.

5. Стовпці із структури даних можуть бути вилучені або вставлені.

6. Угруповання даних для агрегації та перетворень.

7. Високопродуктивне злиття та об'єднання даних.

8. Функціональність часових рядів.

TensorFlow – це ще одна популярна бібліотека для машинного навчання, розроблена Google [14]. Вона підтримує як навчання з учителем, так і без учителя, а також дозволяє легко будувати нейронні мережі. TensorFlow відома своєю масштабованістю та використанням у виробничих системах. Основні переваги TensorFlow:

1. Підтримка розподілених обчислень для прискорення тренувань моделей.

2. Гнучкість у побудові нейронних мереж різної складності.

3. Інструменти для візуалізації моделі та результатів.

4. Підтримка різних мов програмування, таких як Python, C++, та JavaScript.

Scikit-learn, pandas, та TensorFlow є потужними інструментами для створення алгоритмів машинного навчання. Scikit-learn підходить для класичного машинного навчання, pandas – для обробки та аналізу даних, а TensorFlow – для побудови нейронних мереж та розподілених обчислень. Використання цих бібліотек дозволяє розробляти ефективні та масштабовані рішення для різноманітних задач машинного навчання.

2.2 Вибір наборів даних для обробки в системі класифікації

Набори даних для навчання моделей класифікації та регресії було взято із сайту kaggle.com [22].

Набір даних після попередньої обробки для регресії має 414153 рядків і 77 колонок з параметрами, а також цільову ознаку («Label»). Набір даних після передобробки для класифікації має 276952 рядків і 77 колонок з параметрами, а також цільову ознаку («Label»).

Розподіл цільової ознаки в наборах даних реалізований за допомогою методу pyplot з бібліотеки matplotlib і представлено на рис. 2.1 та рис. 2.2.

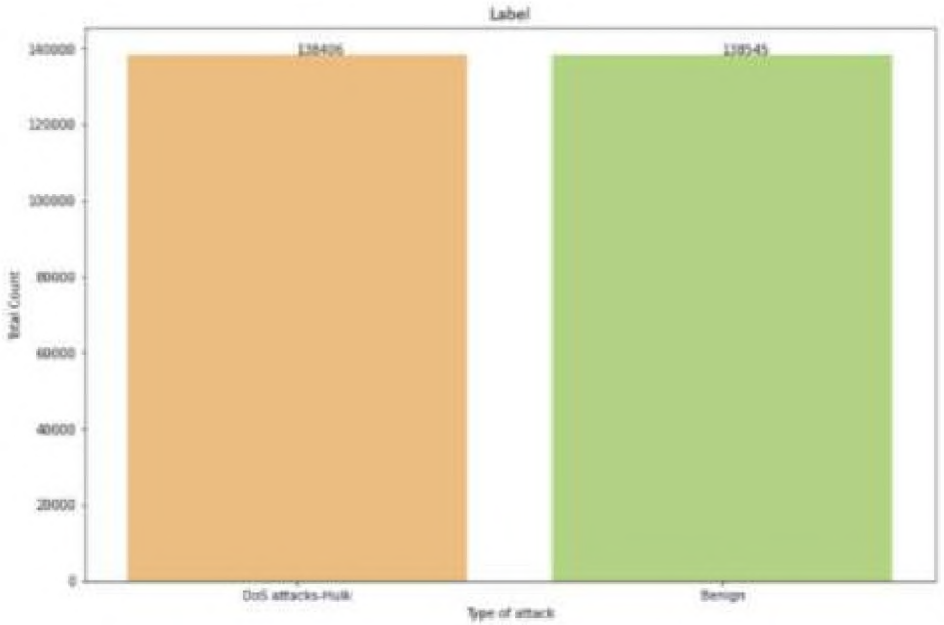

Рисунок 2.1 – Розподіл цільової ознаки набору даних для класифікації

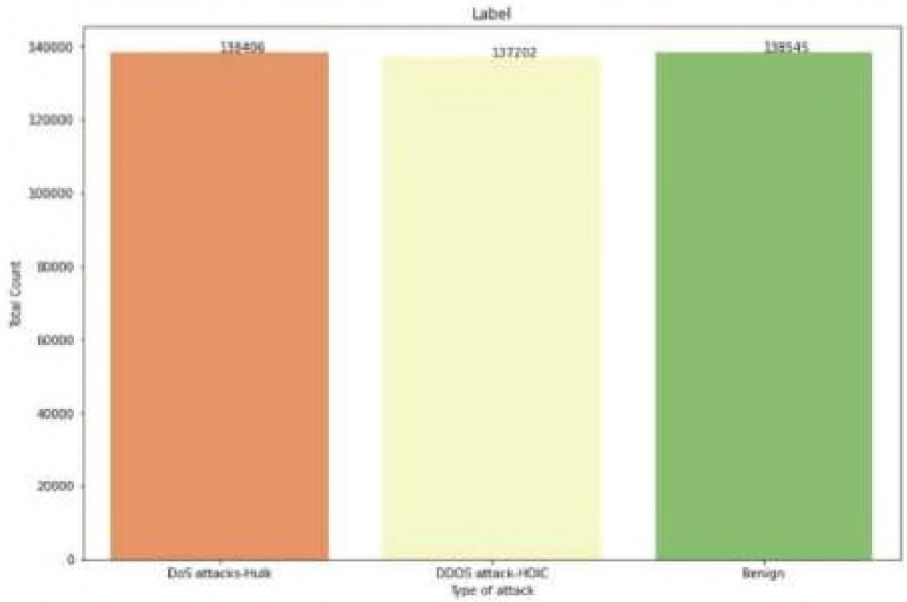

Рисунок 2.2 – Розподіл цільової ознаки набору даних для регресії

Як видно з діаграм, кількість даних з усіх видів атак збалансована, це означає, що аналізовані набори даних підійдуть для ефективної оцінки алгоритмів машинного навчання.

2.3 Функціональні та нефункціональні вимоги до системи класифікації мережевого трафіку

Функціональні вимоги описують поведінку системи, тобто її дії (обчислення, перетворення, перевірки, обробку тощо) [23]. Система, що розробляється, повинна задовольняти наступним функціональним вимогам:

1. Користувач повинен мати можливість завантажити дані у вигляді CSV-файлу.

2. Система повинна дозволяти користувачеві класифікувати мережевий трафік за допомогою готових моделей.

3. Користувач повинен мати можливість вибрати алгоритм машинного навчання та набір даних для класифікації трафіку.

4. Система повинна повідомляти про некоректність введених даних.

Нефункціональні вимоги описують властивості системи (зручність використання, безпека, надійність, розширюваність тощо), які вона повинна мати під час реалізації своєї поведінки [23]. Оскільки існує велика кількість готових бібліотек та надбудов для створення методів машинного навчання з учителем, система повинна задовольняти такі нефункціональні вимоги:

1. Система повинна бути написана мовою Python з використанням мікрофреймворку Flask.

2. Система повинна використовувати методи машинного навчання із учителем для класифікації трафіку.

3. Система повинна класифікувати дані, отримані з CSV-файлу, про вхідний трафік на три типи: DoS-Hulk, DDoS-HOIC і легітимний трафік.

4. Система повинна видавати результат класифікації з точністю щонайменше 90% для бінарної класифікації та середньою абсолютною помилкою трохи більше 10% для багатокласової класифікації.

Ретельне визначення функціональних та нефункціональних вимог забезпечить створення ефективної та надійної системи для класифікації мережевого трафіку. Дотримання цих вимог сприятиме точному виявленню

різних типів трафіку та підвищить безпеку мережі. Використання Python та Flask забезпечить гнучкість і легкість у розробці та подальшому вдосконаленні системи.

2.4 Проєктування варіантів використання системи класифікації мережевого трафіку

Діаграма варіантів використання (рис. 2.3) описує процес роботи додатку для класифікації мережевого трафіку з використанням методів машинного навчання з учителем [24]. Користувач може запустити навчання моделі, використовуючи дані мережевого трафіку, а потім використовувати цю модель для класифікації нового мережевого трафіку. Додаток також надає інформацію про алгоритми машинного навчання, які використовуються в процесі класифікації.

Основний актор, що взаємодіє з системою – користувач. Користувач має три варіанти використання системи: переглянути інформацію про алгоритми машинного навчання, запустити класифікацію мережного трафіку та запустити навчання моделі.

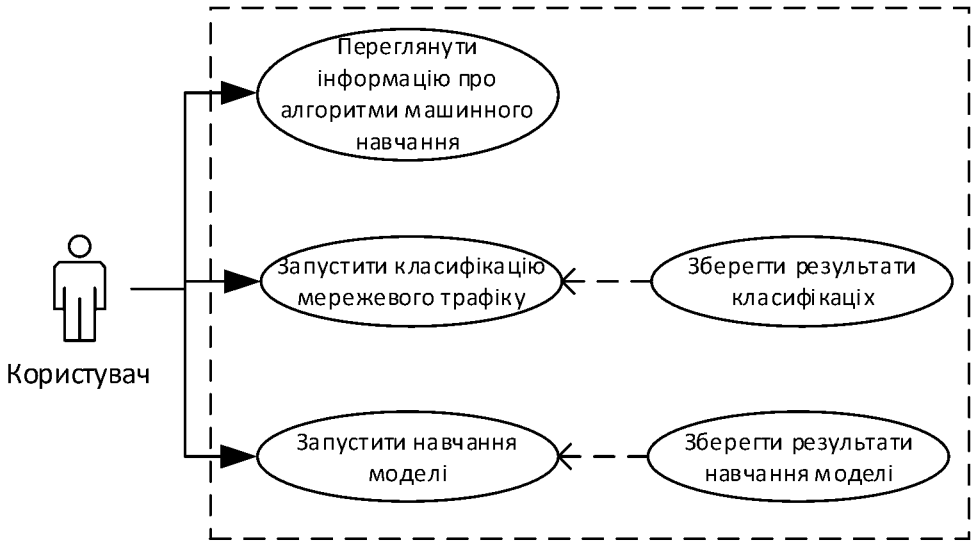

Рисунок 2.3 – Діаграма варіантів використання

Специфікація варіанта використання «Переглянути інформацію про алгоритми машинного навчання» наступна:

Прецедент : Переглянути інформацію про алгоритми машинного навчання.

ID : 1.

Інструкція: Перегляд інформації про алгоритми машинного навчання.

Головні актори: Користувач.

Другі актори: Ні.

Передумови: Ні.

Основний потік:

1. Потік ініціюється, коли користувач натискає на головній сторінці в розділі «Алгоритми» кнопку «Дізнатися більше».

2. Система відкриває сторінку «Алгоритми».

3. Користувач переглядає інформацію.

Наслідки: Ні.

Альтернативні потоки: Ні.

Специфікація варіанта використання «Запустити класифікацію мережевого трафіку» наступна:

Прецедент : Запустити класифікацію мережевого трафіку.

ID : 2.

Анотація: Запустіть процес класифікації мережного трафіку після вибору користувачем готової моделі.

Головні актори: Користувач.

Другі актори: Ні.

Передумови:

1. Користувач перейшов на сторінку «Класифікація».

2. Користувач вибрав готову навчену модель.

3. Користувач завантажив набір даних.

Основний потік:

1. Потік ініціюється, коли користувач натискає кнопку «Класифікувати мережевий трафік».

2. Система запускає процес класифікації.

3. Система виводить результати класифікації на екран.

Наслідки: Ні.

Альтернативні потоки:

1. Потік ініціюється, коли система запускає класифікаційний процес.

2. Система виявляє помилку.

3. Система викликає виняток.

4. Система виводить повідомлення про помилку.

Специфікація варіанта використання «Запустити навчання моделі» наступна:

Прецедент : Запустити навчання моделі.

ID : 3.

Анотація: Запуск процесу навчання моделі після вибору параметрів алгоритму.

Головні актори: Користувач.

Другі актори: Ні.

Передумови:

1. Користувач перейшов на сторінку «Навчання».

2. Користувач вибрав алгоритм та його параметри

3. Користувач вибрав набір даних.

Основний потік:

1. Потік ініціюється, коли користувач натискає кнопку «Навчити модель».

2. Система запускає процес навчання.

3. Система виводить інформацію про закінчення навчання на екрані.

4. Користувач переглядає інформацію.

Наслідки: Ні.

Альтернативні потоки:

1. Потік ініціюється, коли система запускає навчання.

2. Система виявляє помилку.

3. Система викликає виняток.

4. Система виводить повідомлення про помилку користувачеві.

Проєктування варіантів використання системи класифікації мережевого трафіку демонструє чітку організацію процесів взаємодії користувача з додатком. Завдяки можливостям перегляду інформації про алгоритми, запуску класифікації трафіку та навчання моделей, система стає інтуїтивно зрозумілою та зручною у використанні. Це дозволяє користувачам ефективно працювати з даними мережевого трафіку, використовуючи сучасні методи машинного навчання для підвищення безпеки та продуктивності мережі.

2.5 Компоненти архітектури системи класифікації мережевого трафіку

Архітектура системи передбачає використання мікрофреймворку Flask для побудови вебдодатку [25]. Основний програмний модуль (app.py) відповідає за маршрутизацію запитів та обробку введених даних. Користувач може завантажувати дані у вигляді CSV-файлів, які потім обробляються модулями classification.py та fitting.py. Модуль classification.py відповідає за класифікацію мережевого трафіку, використовуючи навчені моделі, тоді як модуль fitting.py здійснює навчання моделей на основі нових даних. Модуль forms.py забезпечує створення та валідацію форм для сторінок, що дозволяє користувачу легко взаємодіяти з системою. На рис. 2.4 зображено діаграму компонентів системи класифікації мережевого трафіку з використанням методів машинного навчання з учителем. Система складається з наступних артефактів: classification.py, app.py, fitting.py та forms.py:

1. app.py – це основний програмний модуль, в котрому реалізовано програмну логіку. Він відповідає за взаємодію з користувачем, обробку запитів, та забезпечення зв'язку між різними компонентами системи.

2. classification.py – програмний модуль, який відповідає за процес класифікації мережевого трафіку. Він використовує навчені моделі для прогнозування типу трафіку на основі вхідних даних.

3. fitting.py – програмний модуль, який відповідає за процес навчання моделі на наборі даних. Цей модуль містить алгоритми машинного навчання, які тренуються на вхідних даних, щоб створити моделі для класифікації.

4. forms.py – програмний модуль, який відповідає за створення форм для сторінок learning.html та classification.html. Ці форми забезпечують зручний інтерфейс для введення та завантаження даних користувачем.

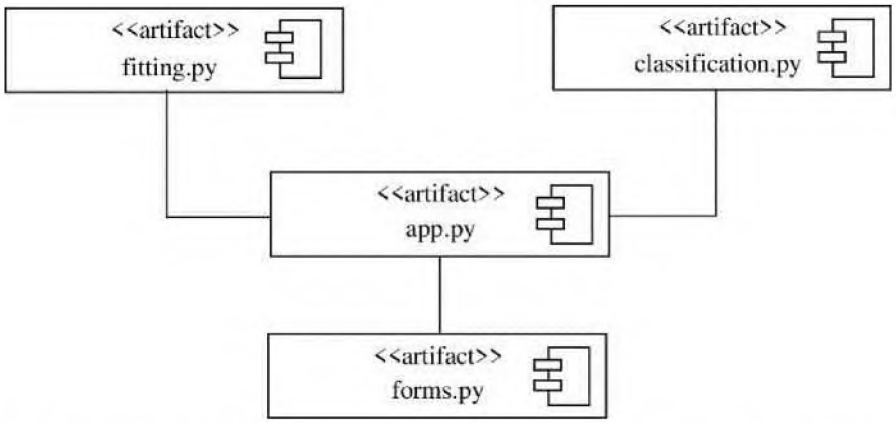

Рисунок 2.4 – Діаграма компонентів системи класифікації мережевого трафіку

Архітектура системи класифікації мережевого трафіку з використанням методів машинного навчання з учителем є добре організованою та модульною. Такий підхід забезпечує легкість у підтримці та розширенні системи, а також зручність для користувачів. Завдяки використанню мікрофреймворку Flask і чітко визначеним компонентам, система може ефективно виконувати класифікацію мережевого трафіку з високою точністю, що є важливим для підтримання безпеки та продуктивності мережі.

2.6 Розроблення діаграми діяльності системи класифікації мережевого трафіку

На рис. 2.5 представлено діаграму діяльності. У представленій діаграмі користувач знаходиться у головному вікні програми. Користувач вибирає файл

та навчену модель для класифікації і натискає на кнопку «Класифікація». Після цього система запускає класифікацію вибраного файлу та виводить результати класифікації на екран. Користувач переглядає результати класифікації і може повторно запустити класифікацію з тим самим файлом та моделлю або вибрати інші.

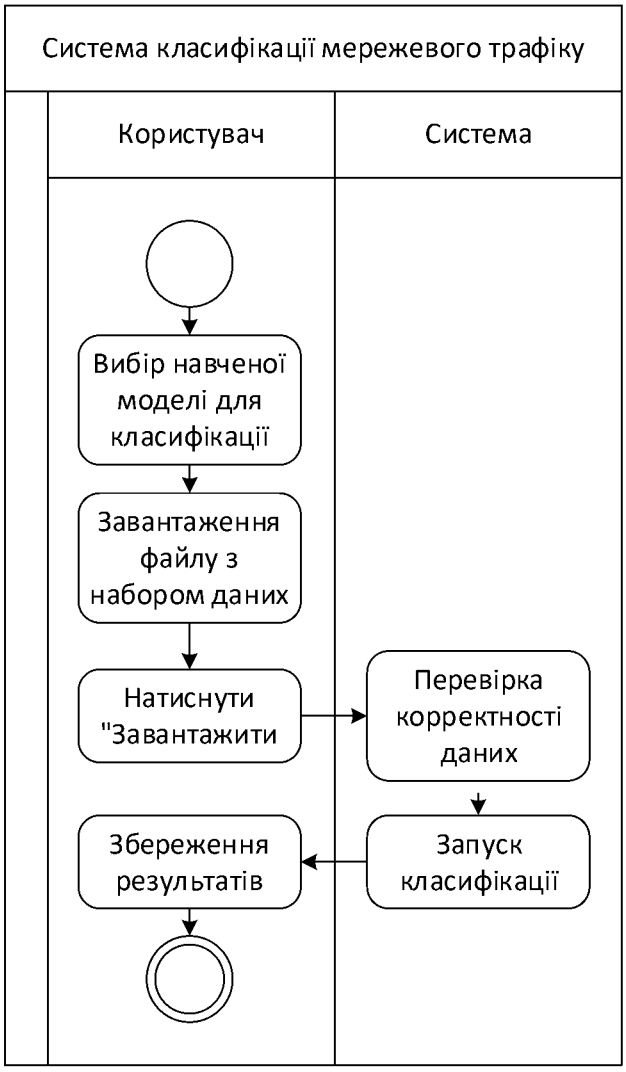

Рисунок 2.5 – Діаграма діяльності системи класифікації мережевого трафіку

Такий підхід дозволяє користувачам легко й інтуїтивно використовувати систему для класифікації мережевого трафіку. Крім того, можливість повторного запуску класифікації з різними моделями або файлами підвищує гнучкість і ефективність роботи з програмою. Ця функціональність робить систему зручною та практичною для широкого кола користувачів, від новачків до досвідчених спеціалістів.

РОЗДІЛ 3
РОЗРОБКА ФУНКЦІЙ ПРОГРАМНОГО ЗАСТОСУНКУ СИСТЕМИ КЛАСИФІКАЦІЇ МЕРЕЖЕВОГО ТРАФІКУ

3.1 Програмні засоби реалізації компонент системи класифікації мережевого трафіку

Для розробки програмної частини системи було обрано високорівневу мову Python 3.7.8 [27]. Розробка велася у редакторі Visual Studio Code 1.56.2 [28]. Інтерфейс користувача був створений за допомогою мови розмітки HTML5, каскадних таблиць стилів CSS3 і фреймворку Bootstrap 4.6.0 [29]. Підключення Bootstrap до Flask здійснювалося за допомогою бібліотеки Bootstrap-Flask 1.0.4. Для навігації на сторінці використовувалася бібліотека JavaScript – jQuery 2.1.3 [30].

Для реалізації серверної частини був використаний мікрофреймворк Flask 1.1.2. Компоненти були реалізовані за допомогою наступних бібліотек для мови Python: pandas 1.1.5, pickle, numpy 1.19.4, WTForms 2.3.3, Werkzeug 2.0.0. Методи машинного навчання з учителем були реалізовані за допомогою бібліотеки scikit-learn 0.24.2. Для відображення математичних формул використовувалася кросбраузерна бібліотека JavaScript – MathJax 3.0.1.

Використання цих технологій дозволило створити ефективну та надійну систему класифікації мережевого трафіку. Такий вибір інструментів забезпечує високу продуктивність та зручність використання, роблячи систему придатною для широкого кола користувачів.

3.2 Прототипування функцій передобробки наборів даних

Перед початком навчання моделей машинного навчання необхідно відповідно обробити набори даних, щоб отримати більш ефективні результати. По-перше, слід перевірити наявність пропусків у даних. Для цього була

спроєктована функція підрахунку пропусків, яка надає інформацію про кількість та відсоток пропущених значень у кожному стовпці. UML-діаграма цієї функції показана на рис. 3.1

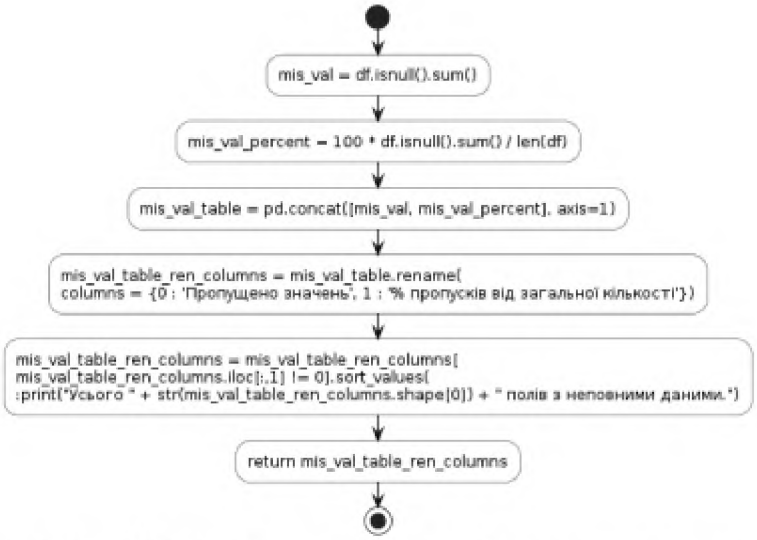

Рисунок 3.1 – UML-діаграма функції підрахунку пропусків

З отриманими даними про пропуски використовують методи fillna() бібліотеки pandas для їх заповнення. Пропуски в числових стовпцях заповнюються медіанним значенням, а в рядкових – значенням, яке найчастіше зустрічається.

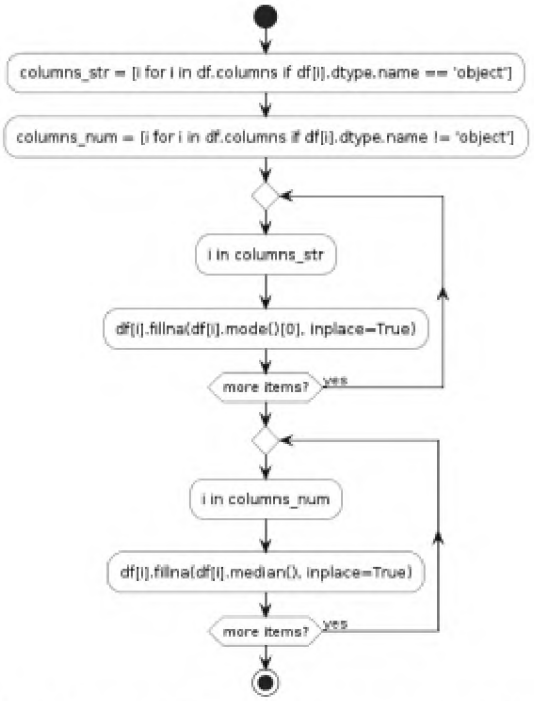

Рисунок 3.2 – UML-діаграма функції заповнення пропущених даних

Для коректної роботи алгоритмів машинного навчання категоріальні дані потрібно перетворити у числові. Ця операція виконується за допомогою функції, UML-діаграма якої представлена на рис. 3.3.

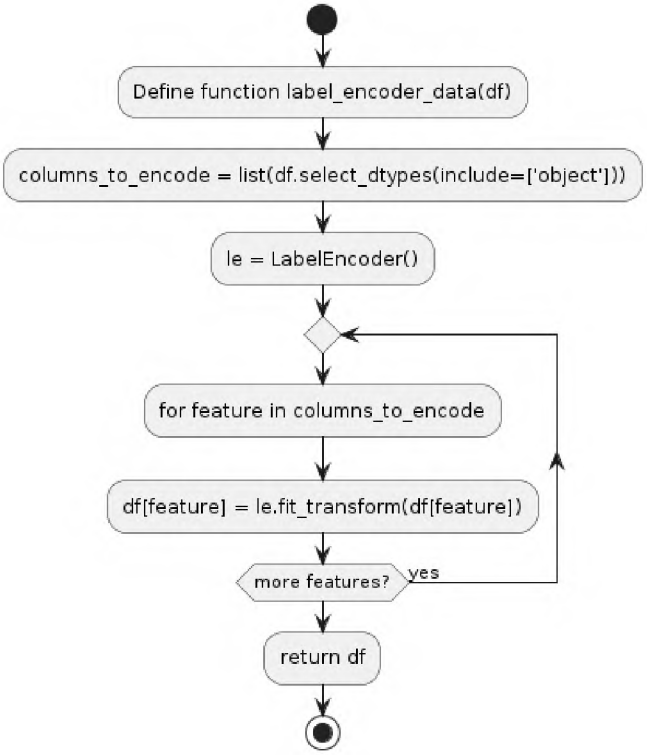

Рисунок 3.3 – UML-діаграма функції перетворення категоріальних даних на числові

Після обробки даних була побудована матриця кореляції, на основі якої видалено стовпці з низькою кореляцією з цільовою ознакою «Label». В результаті цих операцій кількість стовпців зменшилася до 53, включаючи цільову ознаку.

3.3 Підключення методів машинного навчання з учителем

У цьому підрозділі описується навчання моделей машинного навчання, які дозволяють користувачеві запускати процес класифікації на різних наборах

даних. Експериментально були визначені параметри моделей машинного навчання, наведені в таблиці 3.1.

Таблиця 3.1 – Набір параметрів для моделей машинного навчання

Модель	**Набір параметрів**
Decision Tree Regressor	criterion = «mae», max_depth = 5
KNeighbors Regressor	n_neighbors = 5, algorithm = «auto»
Support Vector Machine Regressor	kernel = «rbf», gamma = «auto», C = 1.0, max_iter = 20000
Decision Tree Classifier	criterion = «gini», max_depth = 5
KNeighbors Classifier	n_neighbors = 5, algorithm = «auto»
Support Vector Machine Classifier	kernel = «rbf», gamma = «auto», C = 1.0, max_iter = 1000
Gaussian Naive Bayes	var_smoothing = 1e-15
Logistic Regression	penalty = «l2», solver = «newton-cg»

Навчання алгоритмів Decision Tree, k-Nearest Neighbors, Support Vector Machine, Naive Bayes, Logistic Regression для класифікації представлено на рис. 3.4.

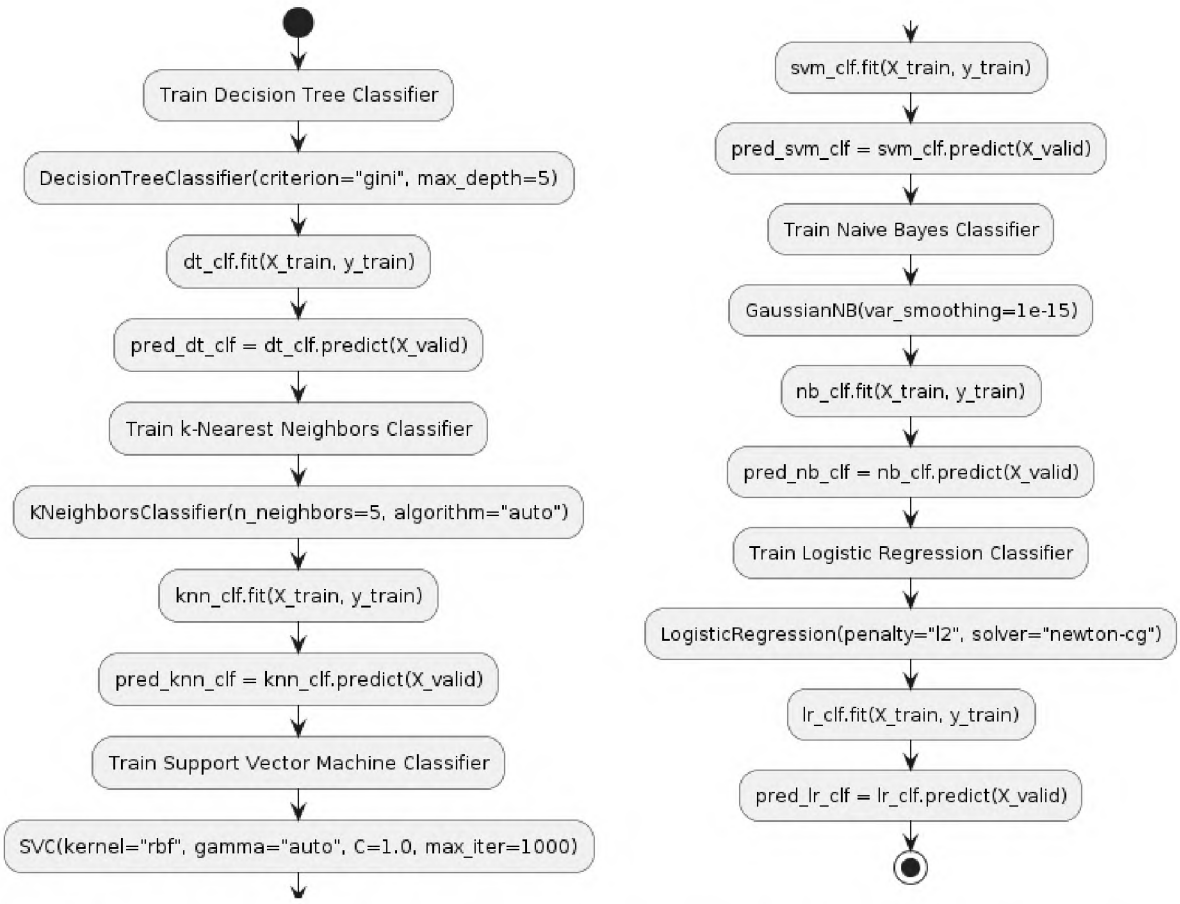

Рисунок 3.4 – UML-діаграма функцій навчання алгоритмів для класифікації

Навчання алгоритмів Decision Tree, k-Nearest Neighbors, Support Vector Machine для регресії представлено на рис. 3.5.

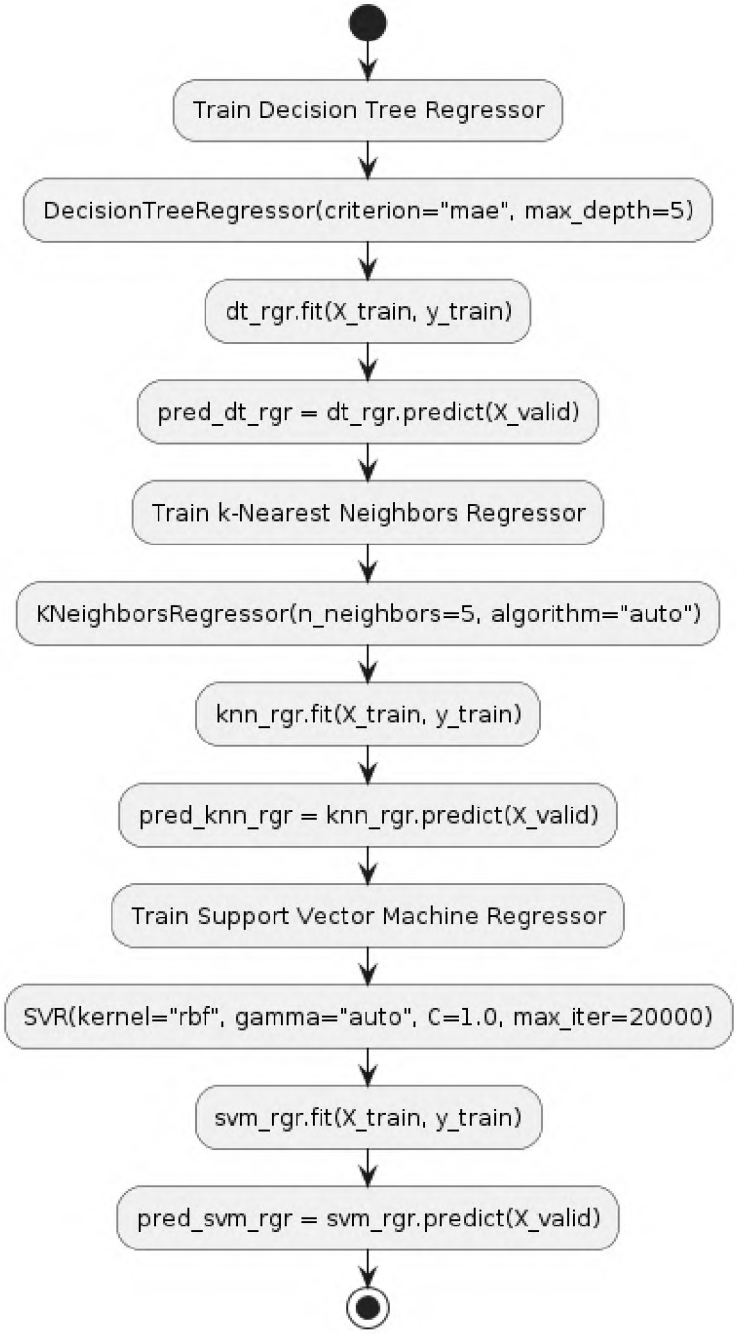

Рисунок 3.5 – UML-діаграма функцій навчання алгоритмів для регресії

Як видно з рис. 3.4 і рис. 3.5, моделі машинного навчання підготовлені для використання у класифікації та регресії. Якість моделей машинного навчання з учителем представлена в таблиці 3.2.

Таблиця 3.2 – Якість моделей

Модель	Метрика	Значення
Decision Tree Regressor	mean absolute error (середня абсолютна помилка)	0,0001
KNeighbors Regressor	mean absolute error (середня абсолютна помилка)	0,0083
Support Vector Machine Regressor	mean absolute error (середня абсолютна помилка)	0,03
Decision Tree Classifier	accuracy (точність)	0,9991
KNeighbors Classifier	accuracy (точність)	0,9923
Gaussian Naive Bayes	accuracy (точність)	0,9329
Logistic Regression	accuracy (точність)	0,9652

Як видно з таблиці 3.2, запропоновані моделі машинного навчання демонструють високу ефективність у розв'язанні поставлених завдань.

3.4 Прототипування функцій і класів компонентів системи класифікації

Компонент classification.py відповідає за процес класифікації мережевого трафіку. Розглянемо функції, які належать до цього компонента.

Функція load_model() використовується для коректного завантаження моделі машинного навчання. У цій функції застосовується метод load з бібліотеки pickle. Функція повертає завантажену модель, яку можна використовувати для класифікації мережевого трафіку.

Функція predict() використовується для передбачення типу трафіку на основі завантаженої навченої моделі. У цій функції присутній метод predict з бібліотеки pickle, а також метод around з бібліотеки numpy для точнішого передбачення моделі. Функція повертає набір даних та масив із передбаченнями.

Функція getfiles() використовується для отримання певного типу файлів із зазначеної директорії. У цій функції застосовується метод listdir з бібліотеки os. Функція повертає відібрані файли з директорії.

Компонент fitting.py відповідає за процес навчання моделі. Розглянемо функції, які належать до цього компонента. Функція MLA() використовується для навчання моделі на основі вибраних параметрів.

Компонент app.py відповідає за програмну логіку системи. Розглянемо функції, які належать до цього компонента.

Функція classification() використовується для коректної обробки дій користувача на сторінці classification.html та для запуску процесу класифікації.

Функція learning() використовується для коректної обробки дій користувача на сторінці learning.html та для запуску процесу навчання.

Компонент forms.py відповідає за створення форм для вибору набору даних та моделі. Розглянемо класи, які належать до цього компонента.

Клас ClassificationForm призначений для створення форм для вибору набору даних та моделі.

Клас LearningForm призначений для створення форм для вибору алгоритму, його параметрів та набору даних для навчання. У класі реалізовано створення форми для вибору набору даних і моделі.

3.5 Проєктування інтерфейсу системи класифікації

Інтерфейс користувача включає чотири вебсторінки. Розглянемо кожну з них.

1. classification.html. На цій вебсторінці представлений інтерфейс, який дозволяє завантажувати та аналізувати мережевий трафік на предмет шкідливих атак, а також завантажувати результати класифікації у форматі CSV.

2. main_page.html. Ця вебсторінка є стартовою та містить короткий опис проєкту. З неї можна перейти до розділів «Алгоритми», «Класифікація», «Навчання» або на сторінки соціальних мереж розробника.

3. learning.html. Ця вебсторінка дозволяє користувачеві навчити модель, вибравши алгоритм машинного навчання, його параметри та набір даних, а також завантажити навчену модель.

4. algorithms.html. На цій вебсторінці представлений інтерфейс, який дозволяє користувачеві детальніше ознайомитися з алгоритмами машинного навчання з учителем. Інформація про алгоритми включає опис принципу роботи алгоритму з наданням формул та зображень, а також переваги та недоліки даного алгоритму.

3.6 Результати тестування моделей машинного навчання

Далі розглянуто тестування моделей машинного навчання (табл. 3.3 –3.10).

Таблиця 3.3 – Тестування моделі Decision Tree Regressor

Набір параметрів	mean absolute error (середня абсолютна помилка)
criterion = «mae», max_depth = 1	0,2178
criterion = «mae», max_depth = 3	0,1453
criterion = «mae», max_depth = 5	0,0001

Висновок: Модель Decision Tree Regressor показала найменшу середню абсолютну помилку при max_depth = 5.

Таблиця 3.4 – Тестування моделі KNeighbors Regressor

Набір параметрів	mean absolute error (середня абсолютна помилка)
n_neighbors = 5, algorithm = «kd_tree»	0,1037
n_neighbors = 5, algorithm = «ball_tree»	0,0763
n_neighbors = 5, algorithm = «brute»	0,0083

Висновок: Модель KNeighbors Regressor показала найкращі результати при використанні алгоритму «brute».

Таблиця 3.5 – Тестування моделі Support Vector Machine Regressor

Набір параметрів	mean absolute error (середня абсолютна помилка)
kernel = «linear», C = 1.0, max_iter = 100	0,2284
kernel = «rbf», C = 1.0, max_iter = 100	0,1136
kernel = «rbf», C = 1.0, max_iter = 1000	0,03

Висновок: Модель Support Vector Machine Regressor показала найменшу помилку при використанні ядра «rbf» та max_iter = 1000.

Таблиця 3.6 – Тестування моделі Decision Tree Classifier

Набір параметрів	accuracy (точність)
criterion = «gini», max_depth = 1	0,9712
criterion = «gini», max_depth = 3	0,9838
criterion = «gini», max_depth = 5	0,9991

Висновок: Модель Decision Tree Classifier досягла найвищої точності при max_depth = 5.

Таблиця 3.7 – Тестування моделі Kneighbors Classifier

Набір параметрів	accuracy (точність)
n_neighbors = 5, algorithm = «kd_tree»	0,7289
n_neighbors = 3, algorithm = «brute»	0,9145
n_neighbors = 5, algorithm = «brute»	0,9923

Висновок: Модель KNeighbors Classifier показала найкращі результати при n_neighbors = 5 та алгоритмі «brute».

Таблиця 3.8 – Тестування моделі Support Vector Machine Classifier

Набір параметрів	accuracy (точність)
kernel = «linear», C = 1.0, max_iter = 1000	0,6982
kernel = «rbf», C = 1.0, max_iter = 100	0,8923
kernel = «rbf», C = 1.0, max_iter = 1000	0,9734

Висновок: Модель Support Vector Machine Classifier досягла найвищої точності при використанні ядра «rbf» та max_iter = 1000.

Таблиця 3.9 – Тестування моделі Naive Bayes

Набір параметрів	accuracy (точність)
var-smoothing = 1e-3	0,5634
var-smoothing = 1e-7	0,7543
var-smoothing = 1e-15	0,9329

Висновок: Модель Naive Bayes показала найкращі результати при var-smoothing = 1e-15.

Таблиця 3.10 – Тестування моделі Logistic Regression

Набір параметрів	accuracy (точність)
penalty = «l1», solver = «liblinear»	0,5648
penalty = «l2», solver = «saga»	0,7129
penalty = «l2», solver = «newton-cg»	0,9652

Модель Logistic Regression досягла найвищої точності при використанні параметрів penalty = «l2» та solver = «newton-cg».

У кожній з моделей було обрано найкращий набір параметрів, що дозволило досягти високої точності та мінімальних помилок. Запропоновані моделі машинного навчання демонструють ефективність для задач класифікації та регресії.

3.7 Розрахунок витрат забезпечення функціонування системи класифікації

Витратна частина створення серверної компоненти системи класифікації складається з таких витрат, як: витрати на електроенергію, витрати на розміщення в мережі Інтернет (хостинг), заробітна плата програмістів та інші витрати на канцелярські товари, витратні матеріали для комп'ютерів, оренду приміщення, амортизацію комп'ютерів і оргтехніки тощо.

Розрахунок електроенергії виконується за тарифом для підприємств за 1 кВт/г = 4,32 грн. Тоді щомісячна вартість оплати електроенергії підприємством складе 1263,6 грн.

Сервер системи класифікації планується розмістити на ресурсах провайдера міста Полтава, що забезпечить зручне обслуговування. Розрахунок щомісячних витрат на підтримку серверної компоненти системи класифікації наведено в таблиці 3.11, а щомісячних матеріальних витрат – у таблиці 3.12. Витрати на період розробки програмного продукту складають Зпр = 847,1 грн.

Розрахуємо собівартість програмного продукту за формулою:

$$С_{ст} = З_{пр} + ЗП_{пр} + ЕСВ + А\ . \qquad (3.1)$$

Тоді собівартість розробки сторінки вебсайту складе:

$$Сст = 847,1 + 1818 + 422,65 + 108,5 = 3196,25 \text{ грн.}$$

Таблиця 3.11 – Розрахунок щомісячних витрат на підтримку серверної компоненти системи класифікації

Найменування	Сума, грн	ЄСВ, грн
Зарплата програміста	14000	422,65
Зарплата кур'єра	2000	422,65
Транспортні витрати кур'єра	350	×
Електроенергія	1263,6	×
Хостинг	200	×
Інтернет	100	×
Інші витрати	300	×
Разом:	8213,6	845,3
Всього витрат:	19058,9	

Таблиця 3.12 – Розрахунок щомісячних матеріальних витрат

Найменування	Сума, грн/міс.
Електроенергія	1263,6
Хостинг	200
Інтернет	100
Інші витрати	300
Разом:	1863,6

Таким чином, щомісячні витрати на підтримку серверної компоненти системи класифікації складають 19058,9 грн, а щомісячні матеріальні витрати – 1863,6 грн. Собівартість розробки вебсайту становить 3196,25 грн. Загальні витрати показують економічну доцільність та ефективність обраного підходу до реалізації системи.

ВИСНОВКИ

В результаті виконання роботи була досягнута її початкова мета та вирішені поставлені завдання. Було проаналізовано основну наукову, методичну та нормативну літературу з теми класифікації мережевого трафіку для протидії кібератакам.

Проведено детальний аналіз задач класифікації мережевого трафіку, методів машинного навчання, що використовуються для цих цілей, а також порівняння існуючих систем виявлення атак. Аналіз показав, що ідентифікація та класифікація мережевого трафіку є критично важливими для забезпечення ефективного управління мережею та її безпеки. Сучасні технології машинного навчання, такі як k-NN, наївний байєсовський класифікатор, метод опорних векторів, дерева рішень та логістична регресія, виявилися дуже ефективними в завданнях класифікації.

Порівняння існуючих систем виявлення атак показало, що використання машинного навчання значно підвищує точність виявлення атак. Кращими алгоритмами виявилися Random Forest та нейронні мережі, що показали високу точність класифікації. Крім того, виявлення DDoS-атак в архітектурі SDN з використанням методів вибору ознак та моделей машинного навчання є перспективним напрямом для забезпечення безперервності роботи мережі.

Проєктування архітектури системи класифікації мережевого трафіку з використанням машинного навчання передбачає використання декількох потужних бібліотек, таких як scikit-learn, pandas та TensorFlow, для різних етапів обробки та аналізу даних. Вибір відповідних наборів даних з сайту kaggle.com забезпечує збалансованість та придатність для оцінки алгоритмів машинного навчання. Чітко визначені функціональні та нефункціональні вимоги гарантують ефективність та надійність системи, що дозволяє класифікувати мережевий трафік з високою точністю. Розроблені варіанти використання та компоненти архітектури системи забезпечують інтуїтивно зрозумілий інтерфейс для користувачів та легкість у подальшій підтримці та розширенні. Архітектура

системи, побудована на основі мікрофреймворку Flask, забезпечує модульність та гнучкість, необхідні для ефективного виконання класифікації мережевого трафіку, підвищуючи безпеку та продуктивність мережі.

У роботі було розглянуто процес розробки функцій програмного застосунку системи класифікації мережевого трафіку. Для розробки програмної частини системи було обрано високорівневу мову програмування Python, редактор Visual Studio Code, а також HTML5, CSS3, фреймворк Bootstrap та бібліотеку Bootstrap-Flask для створення інтерфейсу користувача. Було розроблено прототипи функцій для підрахунку та заповнення пропущених значень у наборах даних, а також для перетворення категоріальних даних на числові. Це дозволило забезпечити коректну роботу алгоритмів машинного навчання та підвищити якість обробки даних.

Розроблено та протестовано декілька моделей машинного навчання з учителем, включаючи Decision Tree, KNeighbors, Support Vector Machine, Naive Bayes та Logistic Regression. Було визначено оптимальні параметри для кожної з моделей, що забезпечило високу точність класифікації та мінімальну середню абсолютну помилку для регресії. Проведено тестування моделей машинного навчання, що підтвердило високу ефективність запропонованих алгоритмів для задач класифікації та регресії. Моделі показали високі значення точності та низькі значення середньої абсолютної помилки.

Таким чином, поставлені задачі розв‘язано у повному обсязі. Напрямком подальших досліджень є впровадження методів глибинного навчання (Deep Learning) для підвищення точності класифікації та виявлення більш складних патернів у мережевому трафіку, а також оптимізація існуючих алгоритмів та програмного забезпечення для покращення продуктивності системи.