МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО–КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ
КАФЕДРА УПРАВЛІННЯ КІБЕРБЕЗПЕКОЮ ТА ЗАХИСТОМ ІНФОРМАЦІЇЇ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: “ЗАСТОСУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ АНАЛІЗУ КІБЕРЗАГРОЗ”

на здобуття освітнього ступеня бакалавра
зі спеціальності 125 Кібербезпека
освітньої програми Управління інформаційною та кібернетичною безпекою

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Дмитро МАЛАШ
(підпис) *Ім’я, ПРІЗВИЩЕ здобувача*

Виконав здобувач вищої освіти гр. УБД-42

Дмитро МАЛАШ
Ім’я, ПРІЗВИЩЕ

Керівник: Юрій ЩАВІНСЬКИЙ
Д.т.н., професор Ім’я, ПРІЗВИЩЕ
Рецензент: Юрій ПЕПА
к.т.н., доцент Ім’я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО–КОМУНІКАЦІЙНИХТЕХНОЛОГІЙ

Навчально–науковий інститут кібербезпеки та захисту інформації

Кафедра управління кібербезпекою та захистом інформації

Ступінь вищої освіти бакалавр

Спеціальність 125 Кібербезпека

Освітня програма Управління інформаційною та кібернетичною безпекою

ЗАТВЕРДЖУЮ
Завідувач кафедри УКБЗІ
_____ Світлана ЛЕГОМІНОВА
“____”_____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Малаш Дмитро Олександрович
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи “Застосування методів машинного навчання для аналізу кіберзагроз”
керівник кваліфікаційної роботи ЩАВІНСЬКИЙ Юрій, к. т. н., доцент,
(ІПРІЗВИЩЕ, Ім’я,, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно–комунікаційних технологійвід “24“ лютого 2025 р. №56.

2 Строк подання кваліфікаційної роботи “12” травня 2025р.

3. Вихідні дані до кваліфікаційної роботи: *управління інформаційною безпекою підприємства,*
загрози інформаційній безпеці, розвідка загроз інформаційній безпеці, технології розвідки загроз
інформаційній безпеці.

4. Аналіз наукових підходів до виявлення та аналізу кіберзагроз із використанням методів машинного навчання.
4.1.Дослідження алгоритмів машинного навчання, які застосовуються для класифікації, кластеризації та виявлення аномалій у кібербезпеці.
4.2.Розробка та навчання моделі машинного навчання на базі відкритих наборів даних.
4.3.Реалізація та тестування моделі в Python із використанням бібліотек (Scikit–learn, TensorFlow/PyTorch).
4.4.Оцінка ефективності моделі та порівняння її з традиційними методами детекції загроз.

5. Перелік ілюстративного матеріалу: *презентація PowerPoint*

6. Дата видачі завдання “5“ березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Етапи кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Визначення об’єкту, предмету, мети та завдань дослідження.	11.03.2025	виконано
2.	Збір та аналіз літератури.	14.03.2025	виконано
3.	Аналіз кіберзагроз і сучасних методів їх виявлення	31.03.2025	виконано
4.	Теоритичні основи машинного навчання у вявленні кіберзашроз	14.04.2025	виконано
5.	Розробка та впровадження моделі машинного навчання	28.04.2025	виконано
6.	Оцінка ефективності модели та рекомендації	01.05.2025	виконано
7.	Формулювання висновків за результатами проведеного дослідження.	05.05.2025	виконано
8.	Оформлення роботи.	12.05.2025	виконано
9.	Оформлення презентації.	19.05.2025	виконано
10.	Отримання рецензії на роботу.	01.06.2025	виконано
11.	Захист в ДЕК.	__.06.2025	

Здобувач вищої освіти _____ Дмитро МАЛАШ

(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник

кваліфікаційної роботи _____ Юрій ЩАВІНСЬКИЙ

(підпис) *(Ім’я, ПРІЗВИЩЕ)*

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО–КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ

ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня бакалавра

Направляється здобувачка Малаш Д.О. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 125 Кібербезпека
(код, найменування спеціальності)
освітньої програми Управління інформаційною та кібернетичною безпекою
(назва)
на тему: “ Застосування методів машинного навчання для аналізу кіберзагроз ”
Кваліфікаційна робота і рецензія додаються.

Директор ННІКБЗІ _____ Євгенія ІВАНЧЕНКО
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Кваліфікаційна робота є завершеним самостійним дослідженням, що відповідає актуальним потребам сфери інформаційної безпеки. У процесі виконання роботи студент продемонстрував високий рівень теоретичної підготовки та практичних навичок. Він обґрунтовано висвітлив роль методів машинного навчання в сучасних системах кіберзахисту, здійснив огляд і порівняння поширених алгоритмів ML, таких як дерева рішень, SVM, нейронні мережі, кластеризаційні методи та алгоритми виявлення аномалій. Окрему увагу було приділено метрикам оцінки якості моделей, що є ключовим у задачах аналізу кіберзагроз.

Робота вирізняється логічною структурою, науковим стилем викладення, грамотним оформленням та обґрунтованістю вибраних рішень. Студент проявив ініціативність, аналітичне мислення та здатність до самостійної дослідницької діяльності.

З урахуванням викладеного вважаю, що кваліфікаційна робота повністю відповідає вимогам, які висуваються до робіт такого рівня, і заслуговує на оцінку «добре», а її автор — на присвоєння кваліфікації бакалавра за спеціальністю

Керівник кваліфікаційної роботи _____ Юрій ЩАВІНСЬКИЙ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

“_____“ ____2025 року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувачка Малаш Д.О.. допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедри
управління кібербезпекою та
захистом інформації _____ Світлана ЛЕГОМІНОВА
(підпис) *(Ім’я, ПРІЗВИЩЕ*

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну бакалаврську роботу

здобувача вищої освіти МАЛАША Дмитра
на тему “ Застосування методів машинного навчання для аналізу кіберзагроз ”

Актуальність. У сучасних умовах стрімкого зростання кількості та складності кіберзагроз використання методів машинного навчання (ML) набуває особливого значення. Ці методи дозволяють автоматизувати виявлення аномалій, аналіз поведінки, класифікацію атак та прогнозування загроз. Обрана тема є надзвичайно актуальною, оскільки поєднує новітні досягнення у сфері штучного інтелекту з практичними завданнями кіберзахисту.

Позитивні сторони.

У роботі логічно викладено основні теоретичні положення, а також продемонстровано практичне застосування моделей ML. Розглянуто три основні напрямки роль машинного навчання у кібербезпеці – автор окреслює ключові сфери застосування ML, зокрема у виявленні аномалій, фішингу, DDoS-атак, шкідливого ПЗ та ін. Подано вичерпний аналіз метрик точності, повноти, F1-міри, ROC-AUC, що є критичними при оцінці моделей у контексті виявлення загроз.

Робота демонструє обізнаність автора у предметній галузі, належний рівень аналітичного мислення та вміння обґрунтовувати вибір моделей і метрик.

Недоліки.

Було б доцільно навести приклади обробки реальних наборів даних, щоб підкріпити теоретичні положення практичними результатами. Але вказані недоліки не зменшують важливість роботи

Висновок: Кваліфікаційна робота виконана на належному науково-методичному рівні і заслуговує оцінки “добре”, а здобувач МАЛАШ Дмитро заслуговує присвоєння кваліфікації бакалавра з кібербезпеки за освітньою програмою Управління інформаційною та кібернетичною безпекою.

Рецензент: Юрій ПЕПА

к.т.н., доцент *підпис* Ім’я, ПРІЗВИЩЕ

РЕФЕРАТ

Кваліфікаційна робота присвячена дослідженню можливостей застосування методів машинного навчання для виявлення кіберзагроз у мережевому трафіку. Робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. У роботі наведено 5 рисунків та 11 таблиць. Загальний обсяг роботи становить 84 сторінки, із яких 2 сторінки займають список використаних джерел та додаткові матеріали.

Мета дослідження - розробка та обґрунтування методики застосування методів машинного навчання для виявлення, аналізу та прогнозування кіберзагроз з метою підвищення ефективності системи інформаційної безпеки.

Об'єкт дослідження - процеси аналізу та виявлення кіберзагроз у системах управління інформаційною безпекою.

Предмет дослідження - методи машинного навчання та алгоритми виявлення аномалій, що використовуються для ідентифікації та класифікації кіберзагроз у мережевому трафіку, логах безпеки та інших джерелах даних.

Методи дослідження. У роботі використано методи машинного навчання, аналізу даних, порівняльної оцінки, візуалізації результатів, побудови та валідації моделей, тестування на відкритих датасетах. Також застосовувались методи прогнозування, експертної оцінки та обробки великих обсягів даних.

У ході дослідження було проаналізовано типи кіберзагроз, методи їх виявлення та актуальні публічні датасети. Проведено вибір джерела даних CIC–IDS2017. Побудовано повний цикл розробки моделі: від попередньої обробки до порівняльного аналізу результатів. Найкращі результати показав алгоритм XGBoost. Визначено переваги, обмеження та запропоновано рекомендації з інтеграції моделі в інфраструктуру SOC або SIEM.

Галузь застосування. Результати можуть бути використані в підрозділах інформаційної безпеки, а також у науково–дослідній сфері для створення адаптивних систем виявлення загроз у реальному часі.

Ключові слова: машинне навчання, кібербезпека, виявлення загроз, мережевий трафік, SOC, SIEM, автоматизація захисту.

ABSTRACT

The qualification work is devoted to the study of the possibilities of applying machine learning methods to detect cyber threats in network traffic. The work consists of an introduction, four sections, conclusions, a list of sources used and appendices. The work contains 5 figures and 11 tables. The total volume of the work is 84 pages, of which 2 pages are occupied by a list of sources used and additional materials

The purpose of the research is to develop and substantiate a methodology for applying machine learning methods to detect, analyze and predict cyber threats in order to increase the effectiveness of the information security system.

The object of the research is the processes of analyzing and detecting cyber threats in information security management systems.

The subject of the research is machine learning methods and anomaly detection algorithms used to identify and classify cyber threats in network traffic, security logs and other data sources.

Research methods. The study uses machine learning methods, data analysis, comparative evaluation, visualization of results, model building and validation, and testing on open datasets. Methods of forecasting, expert evaluation, and processing of large amounts of data were also used.

The study analyzed the types of cyber threats, methods of detecting them, and relevant public datasets. The CIC–IDS2017 data source was selected. A full cycle of model development was built: from pre–processing to comparative analysis of the results. The best results were shown by the XGBoost algorithm. The advantages, limitations, and recommendations for integrating the model into the SOC or SIEM infrastructure are identified.

Scope of application. The results can be used in information security departments, as well as in the research field to create adaptive real–time threat detection systems.

Keywords: machine learning, cybersecurity, threat detection, network traffic, SOC, SIEM, security automation.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

AI	Artificial Intelligence – штучний інтелект
API	Application Programming Interface – інтерфейс прикладного програмування
CIC–IDS2017	Canadian Institute for Cybersecurity Intrusion Detection System 2017
CPU	Central Processing Unit – центральний процесор
CSV	Comma-Separated Values – формат даних, де значення розділені комами
DDoS	Distributed Denial of Service – розподілена атака на відмову в обслуговуванні
DoS	Denial of Service – атака на відмову в обслуговуванні
DNS	Domain Name System – система доменних імен
F1–score	Метрика, що враховує precision і recall
FTP	File Transfer Protocol – протокол передавання файлів
HTTP	HyperText Transfer Protocol – протокол передавання гіпертексту
IDS	Intrusion Detection System – система виявлення вторгнень
IoT	Internet of Things – Інтернет речей
IPS	Intrusion Prevention System – система запобігання вторгнень
KNN	K–Nearest Neighbors – алгоритм k найближчих сусідів
ML	Machine Learning – машинне навчання
MLP	Multilayer Perceptron – багатошаровий перцептрон
NSL–KDD	Network Security Laboratory Knowledge Discovery in Databases
OS	Operating System – операційна система
RNN	Recurrent Neural Network – рекурентна нейронна мережа
ROC	Receiver Operating Characteristic – характеристична крива приймача
SIEM	Security Information and Event Management – система управління подіями безпеки
SMOTE	Synthetic Minority Oversampling Technique – метод синтетичного додавання прикладів
SOC	Security Operation Center – центр операцій безпеки
SQL	Structured Query Language – мова структурованих запитів
SSH	Secure Shell – захищена мережева оболонка
SVM	Support Vector Machine – метод опорних векторів

TCP	Transmission Control Protocol – протокол управління передачею
TP	True Positive – істинно позитивні спрацьовування
FP	False Positive – хибно позитивні спрацьовування
TN	True Negative – істинно негативні
FN	False Negative – хибно негативні
UNSW–NB15	Набір даних, розроблений Університетом Нового Південного Уельсу
XAI	Explainable AI – пояснюваний штучний інтелект
XGBoost	Extreme Gradient Boosting – градієнтний бустинг з оптимізацією
ЗО	Зловмисна активність – небезпечна або аномальна поведінка
ТЗ	Тестова зона – контрольоване середовище для перевірки моделі
НР	Навчальна вибірка – частина даних для тренування моделі
ВР	Валідаційна вибірка – дані для перевірки гіперпараметрів
ТР	Тестова вибірка – дані для остаточного оцінювання моделі

ВСТУП

Актуальністm теми. Кількість кіберзагроз постійно зростає. Класичні методи виявлення атак часто неефективні проти нових типів загроз. Машинне навчання дозволяє автоматично виявляти аномалії та адаптуватися до нових сценаріїв атак. Це робить тему дослідження актуальною для сучасної кібербезпеки.

Мета дослідження - розробка та обґрунтування методики застосування методів машинного навчання для виявлення, аналізу та прогнозування кіберзагроз з метою підвищення ефективності системи інформаційної безпеки.

Об’єкт дослідження - процеси аналізу та виявлення кіберзагроз у системах управління інформаційною безпекою.

Предмет дослідження - методи машинного навчання та алгоритми виявлення аномалій, що використовуються для ідентифікації та класифікації кіберзагроз у мережевому трафіку, логах безпеки та інших джерелах даних.

Методи дослідження. Аналіз літературних джерел, систематизація даних, машинне навчання, моделювання, тестування, порівняльний аналіз.

Практичне значення отриманих результатів. Результати дослідження можуть бути використані для створення автоматизованих систем виявлення загроз. Запропонована модель може бути впроваджена в IT–інфраструктуру для підвищення рівня кіберзахисту.

Розділ 1 АНАЛІЗ КІБЕРЗАГРОЗ І СУЧАСНИХ МЕТОДІВ ЇХ ВИЯВЛЕННЯ

1.1 Основні види кіберзагроз та їх вплив на інформаційну безпеку

Кіберзагрози вражають системи будь–якого рівня. Вони блокують роботу, крадуть або спотворюють інформацію.

Кіберзагрози стають усе небезпечнішими. Вони вражають держави, компанії та звичайних користувачів. Атаки порушують роботу систем, крадуть або знищують інформацію. Це вже реальна загроза для економіки, безпеки та життя людей [1].

Хакери діють масово. Вони використовують віруси, трояни, фішинг, боти, експлойти, програми–вимагачі. Часто працюють автоматизовано. Атаки поширюються швидко. Можна не встигнути помітити зараження.

Вважається, що ключовою проблемою є людський фактор. Соціальна інженерія досі ефективна. Люди натискають на підозрілі посилання, вводять паролі на фішингових сайтах, передають дані незнайомцям. Через це система падає, а дані йдуть у руки зловмисників [2].

Удар по трьох основах безпеки – конфіденційність, цілісність, доступність. Якщо дані вкрадені – втрачено контроль. Якщо змінені – спотворена реальність. Якщо недоступні – система не працює. Це прямі збитки. А ще – втрата довіри, репутації, партнерів.

Кількість атак зростає. Методи – складніші. Антивіруси не завжди допомагають. Мережеві екрани можна обійти. Ручне виявлення вже неефективне. Часто злами залишаються непоміченими тижнями або місяцями.

Рішення – машинне навчання. Модель може аналізувати великий обсяг трафіку. Знаходить аномалії в реальному часі. Реагує швидко. Зменшує втрати. Покращує рівень захисту.

Це не просто теорія. Я досліджував приклади з практики. У більшості з них саме ML–системи виявили нові типи загроз, які раніше не помічалися. Людина не змогла б це зробити вручну [3].

Розвиток IT вимагає нових підходів. Старі методи вже не працюють. Машинне навчання – це не майбутнє, це вже теперішнє. Вважаю, що його інтеграція – ключовий напрям для кіберзахисту.

1.2. Методи виявлення кіберзагроз: традиційні та сучасні підходи

У своєму дослідженні я порівнюю традиційні й сучасні методи виявлення кіберзагроз. Кожен із них має переваги й обмеження. Важливо розуміти їхню суть, щоб оцінити ефективність.

Традиційні методи базуються на сигнатурах. Це шаблони відомих атак. Якщо вхідні дані збігаються з шаблоном, спрацьовує захист. Сигнатури використовують антивіруси, фаєрволи, IDS–системи. Вони швидко блокують уже відомі загрози. Але саме в цьому й обмеження.

Атаки постійно змінюються. Зловмисники модифікують код, шифрують дані, комбінують техніки. У результаті сигнатури не спрацьовують. Нові загрози залишаються непоміченими. Це критична проблема для традиційних систем. (табл.1) Ще один недолік – потреба в оновленні бази. Якщо база неактуальна – захисту немає. Я бачив приклади, коли зараження ставалося через застарілу сигнатуру. І це – поширене явище.

Сучасний підхід – поведінковий аналіз. Системи фіксують типову поведінку. Якщо з’являється відхилення – спрацьовує тривога. Поведінковий аналіз виявляє навіть нові, раніше невідомі загрози.

Аномалії – це сигнали атаки. Незвичний трафік, нетипові дії користувача, дивна активність системи. Все це свідчить про можливе вторгнення. Поведінкові системи працюють глибше. Вони не шукають шаблони – вони бачать зміни.

Таблиця 1

Порівняння традиційних і сучасних методів виявлення загроз

Ознака	**Традиційні методи**	**Сучасні методи**
Основний підхід	Сигнатури	Поведінковий аналіз, ML
Виявлення нових загроз	Низька ефективність	Висока ефективність
Потреба в оновленнях	Часто потрібне	Залежить від моделі
Помилкові спрацювання	Часто трапляються	Менше завдяки навчанню
Гнучкість до змін	Обмежена	Висока
Швидкість реакції	Висока при відомих загрозах	Висока навіть при нових загрозах
Можливість самонавчання	Відсутня	Є
Приклад	Антивірус, IDS	SIEM, XDR, ML–моделі

У цьому напрямку особливо важливі SIEM–системи. Вони збирають події з різних джерел: мережа, сервіси, програми. Потім аналізують, будують залежності й шукають аномалії. Це дозволяє виявити складні атаки, які розгортаються у кілька етапів.

Ще один важливий етап – машинне навчання. Алгоритми вивчають дані й самі знаходять підозрілу активність. Вони бачать те, що не прописано явно. Не потрібні сигнатури. Модель сама виявляє загрозу на основі патернів.

ML–системи зменшують кількість помилкових спрацювань. Вони навчаються на нормальній поведінці. Якщо щось відрізняється – сигнал тривоги. Але без паніки. Модель вміє відрізняти справжню загрозу від випадкової аномалії.

Я вважаю, що саме машинне навчання дає новий рівень гнучкості. Модель може адаптуватися. Поведінка атак змінюється – модель підлаштовується. Це особливо важливо в умовах швидко змінного середовища.

Сучасні системи працюють у реальному часі. Це ще один плюс. Вони не чекають, поки щось трапиться. Вони діють на випередження. Це рятує час і ресурси. А головне – зменшує збитки [4].

Найкращий підхід – гібридний. Поєднання сигнатур, поведінкового аналізу, машинного навчання. Такі системи дають максимальний рівень виявлення загроз. Вони працюють у різних напрямках одночасно.

Комплексний захист – єдиний ефективний варіант. Я переконаний, що ізоляція окремих методів не працює. Тільки взаємодія. Старі й нові інструменти мають доповнювати одне одного [5].

Мій висновок – традиційні методи більше не можуть самостійно забезпечити безпеку. Без сучасного підходу система залишається вразливою. Сьогодні потрібна автоматизація, адаптація, аналіз великих даних. І саме це дають сучасні технології.

1.3 Роль машинного навчання у кібербезпеці

Машинне навчання стало ключовим інструментом у боротьбі з кіберзагрозами. Воно змінює підхід до безпеки. Замість фіксованих правил ми отримуємо гнучку, адаптивну систему.

Головна перевага машинного навчання полягає у здатності обробляти великі обсяги даних у реальному часі. Алгоритми знаходять закономірності там, де людина не помітить нічого підозрілого. Це дає змогу швидко реагувати на загрози, які ще не мають сигнатур або опису [8].

Особливо ефективним ML є проти нових і складних атак. Традиційні засоби часто не помічають загрози, які не схожі на вже відомі. Машинне

навчання навчається на історичних даних. Потім воно порівнює нові ситуації з уже вивченими. Це дозволяє виявити схожі шаблони та попередити атаку [9].

ML вміє знаходити фішинг, шкідливе програмне забезпечення, ботнети, DDoS–атаки. Моделі аналізують поведінку користувачів, дій програм, а також мережевий трафік. Якщо з'являється аномалія – система сигналізує.

Два найпоширеніших підходи – класифікація та виявлення аномалій. У першому випадку модель вирішує, чи є об'єкт безпечним або загрозливим. У другому – шукає відхилення від норми. Це працює для складних і нестандартних атак [10].

Ще одна перевага – швидкість реагування. ML–системи працюють автоматично. Не потрібно чекати, поки аналітик перегляне лог. Алгоритм одразу сигналізує про проблему. Це дозволяє швидко ізолювати підозрілу активність і зменшити шкоду.

Машинне навчання також знижує кількість помилкових спрацювань. Я бачив, як традиційні системи генерували сотні фальшивих тривог за день. (Табл.2)

Таблиця 2

Переваги машинного навчання у виявленні загроз

Перевага	**Опис**
Автоматичне виявлення	Виявляє загрози без участі людини
Робота з великими даними	Обробляє логи, трафік, дії користувачів
Адаптація до нових загроз	Може змінювати поведінку під час навчання
Зниження хибних тривог	Вміє розпізнавати нормальну поведінку
Робота в реальному часі	Аналіз відбувається одразу після надходження подій
Сумісність з іншими системами	Інтегрується у SIEM, SOC, EDR, IDS
Можливість самонавчання	Поліпшує результати з кожним новим прикладом

Це перевантажує аналітиків і знижує їхню увагу до реальних загроз. ML вчиться розпізнавати різницю між шкідливою активністю та нормальними процесами. Це значно зменшує навантаження на SOC–команди [11].

Машинне навчання інтегрується у багато рішень: SIEM, EDR, IDS/IPS. Ці системи аналізують логи, події, трафік, електронну пошту, навіть дії на рівні операційної системи. Алгоритми працюють у фоновому режимі й постійно вдосконалюються. З кожним новим набором даних вони стають точнішими. (рис.1.)

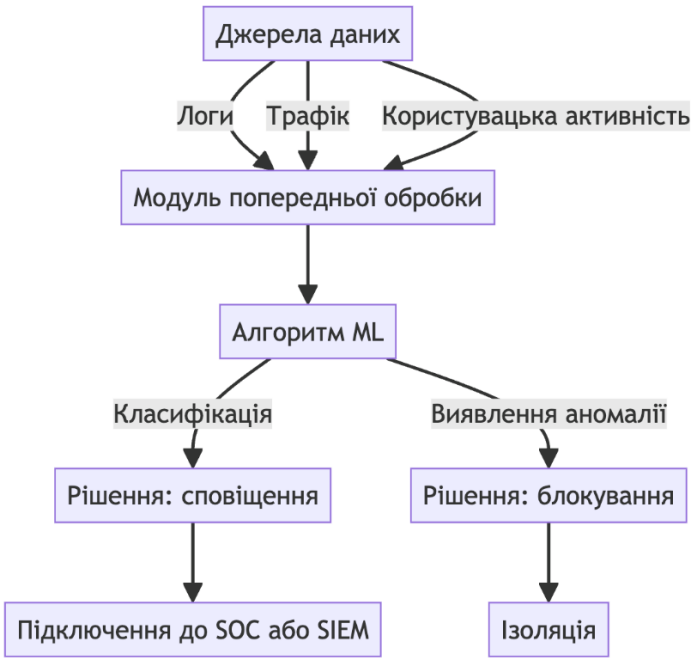

Рис.1. Загальна схема використання машинного навчання у кіберзахисті

Системи можуть працювати як локально, так і в хмарі. Хмарні рішення мають доступ до глобальної статистики. Це підвищує точність виявлення загроз. Локальні – краще контролюють внутрішню інфраструктуру. Я вважаю, що поєднання обох – оптимальний варіант [12].

Існують два основні типи ML–моделей: з учителем (supervised) і без учителя (unsupervised). У першому випадку модель навчається на вже розмічених прикладах атак. У другому – сама знаходить підозрілу поведінку. Обидва типи мають свої переваги. Вони часто використовуються разом.

Машинне навчання дозволяє будувати проактивний захист. Ми не просто реагуємо на атаку, а передбачаємо її. Це важливо в сучасному світі, де атаки можуть початися з незначної дії. ML бачить ці маленькі зміни та сигналізує до того, як станеться шкода [13].

Без ML неможливо побудувати ефективну кібербезпеку в сучасних умовах. Тільки завдяки цим технологіям ми можемо адаптувати захист до нових викликів. Це робить безпеку не статичною, а динамічною і живою. І саме такий підхід потрібен у цифрову епоху.

1.4. Огляд існуючих рішень на основі машинного навчання для аналізу загроз

Сьогодні машинне навчання активно використовується в багатьох рішеннях для кіберзахисту. Найбільший інтерес викликають саме ті системи, які здатні виявляти нові типи атак без прив’язки до сигнатур. Це дозволяє підвищити рівень захисту навіть у складних умовах [14].

Одні з найпоширеніших систем – це системи виявлення вторгнень (IDS). Вони аналізують мережевий трафік і шукають підозрілу активність. Наприклад, Snort і Suricata – популярні IDS, які можна доповнити модулями машинного навчання. У таких випадках алгоритми навчаються розпізнавати шаблони атак і реагують швидше, ніж звичайні правила [15].

Системи класу SIEM (Security Information and Event Management) також усе частіше використовують ML. Наприклад, Splunk та IBM QRadar інтегрують моделі, які автоматично виявляють аномалії у логах, нестандартну поведінку користувачів або спроби несанкціонованого доступу. У цих рішеннях застосовуються методи класифікації, кластеризації та виявлення відхилень, які підвищують точність аналізу[16].

Антивірусні рішення також змінюються. Наприклад, Cylance – це антивірус, який не використовує традиційні сигнатури. Замість цього він

прогнозує, чи є файл шкідливим, на основі його поведінкових характеристик. Це особливо ефективно для виявлення нових або модифікованих вірусів, які ще не мають опису в базі даних.

Цікавою для мене є система Darktrace. Вона базується на самонавчанні. Алгоритми моделюють «нормальну» поведінку мережі й потім шукають відхилення. Такий підхід дозволяє виявляти атаки ще на ранніх етапах. Рішення часто впроваджується в корпоративному середовищі, де мережевий трафік складний і змінюється динамічно.

Ще одна потужна система – Vectra AI. Вона спеціалізується на поведінковому аналізі. Її моделі аналізують дії користувачів, пристроїв і хмарних сервісів. Це дозволяє виявляти такі етапи атаки, як сканування мережі, бічний рух, або спроби підвищити привілеї. Ці дії часто залишаються поза увагою класичних систем [17].

Microsoft Defender for Endpoint також має ML–ядро. Система виявляє загрози на основі аналізу поведінки процесів і файлів. Тут поєднуються моделі, що працюють у реальному часі, та моделі, які аналізують зібрані дані з певною затримкою. Це забезпечує глибший рівень захисту як від швидких атак, так і від повільних, прихованих загроз.

У мережевому моніторингу добре себе показує Cisco Secure Analytics (раніше – Stealthwatch). Це рішення застосовує ML для виявлення внутрішніх загроз, ботнет–активності та витоків даних. Мені здається важливим, що такі системи не потребують глибокого втручання адміністратора й самі вчаться з мережевого трафіку.

Великі хмарні провайдери також впроваджують ML у свої платформи. AWS, Google Cloud і Microsoft Azure пропонують вбудовані інструменти безпеки, які використовують машинне навчання для виявлення підозрілих дій, атак через API, або аномальних входів у систему. Це критично для захисту хмарної інфраструктури, яка постійно змінюється [18].

Серед відкритих рішень мені особливо подобається Security Onion. Це безкоштовна платформа з підтримкою ML. Вона об’єднує декілька інструментів

(наприклад, Suricata, Zeek, ELK–Stack), які разом забезпечують комплексний моніторинг і аналіз. Також зростає популярність використання ELK для виявлення аномалій за допомогою моделей, які інтегруються через спеціальні модулі.

У багатьох компаніях впроваджуються власні ML–моделі у внутрішніх центрах реагування на інциденти (SOC). Це дозволяє точніше враховувати специфіку мережі та зменшити кількість помилкових спрацювань. Я вважаю це хорошим підходом, бо кожна мережа унікальна, і універсальні моделі не завжди дають найкращий результат. (табл. 3)

Таблиця 3

Порівняння існуючих ML–рішень у сфері кібербезпеки

Назва системи	Тип рішення	Особливості застосування ML	Переваги	Обмеження / Недоліки
Snort + ML модулі	IDS	Додаткове навчання моделей для розпізнавання шаблонів	Швидкість, гнучкість	Потребує налаштувань та інтеграції вручну
Suricata + ML	IDS/IPS	Аналіз трафіку, застосування класифікаторів	Глибокий аналіз мережевих пакетів	Велике навантаження на ресурси
Splunk	SIEM	Виявлення аномалій у логах, автоматичне навчання	Інтеграція з багатьма джерелами даних	Висока вартість ліцензії
IBM QRadar	SIEM	Аналіз поведінки користувачів, самонавчання	Підтримка великих корпоративних мереж	Складність впровадження
Cylance	Антивірус	ML–аналіз характеристик файлів	Не використовує сигнатури, виявляє нові загрози	Вимагає постійного оновлення моделей
Darktrace	UEBA/NTA	Самонавчання, виявлення відхилень від нормальної поведінки	Висока точність, адаптація до мережі	Не завжди зрозумілий механізм прийняття рішень
Vectra AI	NDR	Поведінковий аналіз користувачів і хмарних сервісів	Детектує багатоступеневі атаки	Висока вартість
Microsoft Defender	EDR	Поведінкове ML–ядро, аналіз процесів у реальному часі	Інтеграція в ОС, багаторівневий захист	Обмежена робота на сторонніх ОС

Продовження таблиці 3

Cisco Secure Analytics	NTA	Виявлення ботнетів, витоків, внутрішніх атак	Працює з великими мережами, автоматизація	Вимагає тюнінгу для точності
Security Onion	Open–source NIDS	Інтеграція ML у стек Suricata + Zeek + ELK	Безкоштовне, гнучке рішення	Складність налаштування
SOC–рішення (власні)	SOC	ML–моделі, адаптовані під конкретну мережу	Висока точність, адаптація під середовище	Не універсальне, вимагає підтримки
AWS/GCP/ Azure Security	Cloud Security	Виявлення атак у хмарі, аналіз API та доступів	Швидка інтеграція з хмарними ресурсами	Обмежено лише власною платформою

Порівняння існуючих ML–рішень у сфері кібербезпеки

Усі ці рішення мають одну спільну рису – вони постійно вдосконалюються. Моделі навчаються на нових інцидентах, отримують доступ до оновлених даних, і з кожним циклом стають точнішими. Це необхідно, бо кіберзагрози швидко еволюціонують, і старі методи часто виявляються неефективними.

Машинне навчання вже стало основою сучасної кібербезпеки. Його застосування охоплює практично всі рівні – від антивірусів до аналізу хмарних систем. І з кожним роком кількість таких рішень тільки зростає.

Висновки до розділу 1

У першому розділі проаналізовано основні види кіберзагроз і доведено, що сучасні атаки стають складнішими та масштабнішими. Традиційні методи, засновані на сигнатурах, вже не забезпечують належного рівня захисту.

Розглянуто сучасні підходи до виявлення загроз, зокрема поведінковий аналіз і машинне навчання. Показано, що машинне навчання дозволяє автоматично виявляти аномалії, адаптується до нових загроз і зменшує кількість помилкових спрацювань.

Також проведено огляд існуючих ML–рішень у сфері кібербезпеки, зокрема систем IDS, SIEM, EDR, антивірусів та хмарних платформ. Встановлено, що саме поєднання машинного навчання з традиційними методами забезпечує найкращий рівень захисту.

Розділ 2. ТЕОРЕТИЧНІ ОСНОВИ ЗАСТОСУВАННЯ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ КІБЕРЗАГРОЗ

2.1 Основні алгоритми машинного навчання, які застосовуються для аналізу загроз

Короткий опис: логістична регресія, дерева рішень, випадковий ліс, SVM, нейронні мережі, K–Means.

Що підходить для класифікації, що – для кластеризації, що – для аномалій.

2.2 Підготовка даних: джерела, очищення, нормалізація

Звідки беруться дані: NSL–KDD, CIC–IDS2017.

Попередня обробка: видалення шуму, кодування, масштабування.

2.3 Метрики оцінки ефективності моделей у кібербезпеці

Accuracy, Precision, Recall, F1–міра, ROC–AUC.

Чому важливо не просто "точність".

2.4 Обґрунтування вибору алгоритмів для практичного дослідження

Який алгоритм буде використано в роботі й чому.

Врахування точності, часу навчання, стабільності.

2.1 Основні алгоритми машинного навчання, які застосовуються для аналізу загроз

Машинне навчання відіграє важливу роль у сфері кібербезпеки. Воно дає змогу автоматично аналізувати великі обсяги даних і виявляти підозрілі дії. Людині часто важко помітити складні закономірності, а моделі машинного навчання справляються з цим краще. Завдяки ним можна виявити як відомі, так і нові загрози. У цьому підрозділі я розгляну найпоширеніші алгоритми, які застосовуються для аналізу кіберзагроз [19].

Найперше, слід згадати класифікацію. Це метод, який дозволяє відрізняти нормальну активність від шкідливої. Для цього моделі навчаються на мітках – прикладах нормальних і аномальних дій.

Один із найпростіших алгоритмів – це логістична регресія. Вона швидко працює та підходить для задач, де класи добре розділяються. Однак при складних даних вона показує нижчу точність [20].

Також популярні дерева рішень. Вони будують логіку на основі умов. Ці алгоритми добре пояснюються, тому їх часто використовують, коли потрібно зрозуміти, як приймаються рішення. Але дерева можуть перенавчатися, тобто бути занадто чутливими до шуму в даних.

Ще один важливий алгоритм – це метод опорних векторів (SVM). Він добре працює, коли дані мають чіткі межі між класами. Проте цей метод стає повільним, якщо даних дуже багато [21].

Наївний байєсівський класифікатор – це ще одна проста і швидка модель. Вона базується на ймовірностях і добре працює при великій кількості ознак. Головна умова – ознаки повинні бути незалежними, хоча на практиці це не завжди так.

Більш сучасні підходи включають ансамблеві методи. Наприклад, випадковий ліс (Random Forest) поєднує багато дерев рішень. Це робить його більш точним і менш чутливим до випадкових помилок у даних. Також часто застосовується градієнтний бустинг (наприклад, XGBoost). Він будує послідовність моделей, кожна з яких виправляє помилки попередньої. Це дозволяє досягти високої точності [22].

Окрім класифікації, широко використовується кластеризація. Це метод групування даних, коли ми не маємо міток. Такий підхід корисний, коли потрібно знайти невідомі типи атак або незвичну поведінку.

Найвідоміший алгоритм тут – k–середніх (K–Means). Він розбиває дані на кілька груп за відстанню до центрів кластерів. Це просто і ефективно. Але якщо дані не мають чіткої структури, цей метод може помилитися [23].

Також є алгоритм DBSCAN. Він може виявляти кластери довільної форми і добре працює з шумом у даних. Це зручно, коли ми маємо справу з реальним трафіком, де дані не завжди чисті.

Ще одна важлива задача – це виявлення аномалій. У кібербезпеці це надзвичайно актуально, бо атаки часто виглядають як рідкісні події серед нормального трафіку [24].

Один із простих і ефективних методів – Isolation Forest. Він випадково розділяє дані на частини і бачить, які з них легко ізолювати. Такі дані, що легко відокремлюються, вважаються аномальними.

Також використовується One–Class SVM. Він навчається тільки на нормальних прикладах і потім визначає, що є відхиленням. Це підходить, коли шкідливі дії з'являються рідко, і немає достатньо прикладів для повноцінного навчання[25].

Окремо слід виділити глибинне навчання. Воно застосовується тоді, коли потрібно обробити великі обсяги даних або працювати з неструктурованою інформацією.

Штучні нейронні мережі (ANN) можуть класифікувати складні шаблони, наприклад, трафік у мережі. Рекурентні нейронні мережі (RNN) або LSTM добре підходять для аналізу послідовностей, таких як логи або поведінка користувачів. Вони враховують порядок дій, що важливо для виявлення атак у часі [17].

Також часто застосовують автокодери. Вони стискають дані, а потім відновлюють їх. Якщо система погано відновлює деякі приклади, це може свідчити про аномалії.

У реальних умовах найкращі результати дають гібридні підходи. Тобто, коли поєднуються різні алгоритми – наприклад, класифікація та виявлення аномалій. Це дозволяє зменшити кількість хибних спрацювань і покращити точність [26].

Я вважаю, що вибір алгоритму завжди залежить від конкретного завдання, типу даних і доступних ресурсів. Якщо дані мітовані – підійде класифікація. Якщо немає міток – кластеризація або аномалії. Якщо даних багато – нейронні

мережі. Також потрібно враховувати, скільки часу є на навчання та наскільки важлива інтерпретованість результатів.

Загалом, машинне навчання вже сьогодні допомагає виявляти кіберзагрози на ранніх етапах. І його роль буде тільки зростати. Головне – мати якісні дані і правильно підібраний підхід.

2.2 Підготовка даних: джерела, обробка та нормалізація

Щоб навчити модель машинного навчання ефективно виявляти кіберзагрози, потрібно якісно підготувати вхідні дані. Від того, наскільки добре виконана ця підготовка, напряму залежить точність і стабільність майбутніх результатів. Якщо дані містять багато шуму, помилок або неправильно масштабовані, жоден навіть найкращий алгоритм не зможе дати правильних висновків [27].

Дані для аналізу можуть надходити з різних джерел. Найпоширенішими є:

Мережевий трафік. Це можуть бути окремі пакети або повні з'єднання. Часто використовується аналіз TCP/IP–трафіку для виявлення підозрілої активності.

Системні журнали. Логи операційних систем і серверів зберігають події, які можуть свідчити про вторгнення або інші порушення.

SIEM–системи. Ці системи об'єднують інформацію з різних джерел і створюють загальну картину подій безпеки.

IDS/IPS–записи. Системи виявлення і запобігання вторгненням формують важливі повідомлення про спроби атак [28].

Відкриті набори даних. Наприклад, NSL–KDD, CIC–IDS2017, UNSW–NB15 або CSE–CIC–IDS2018. Вони створені спеціально для досліджень у сфері кібербезпеки і добре підходять для тренування моделей[29].

Після отримання даних важливо провести попередню обробку. Зазвичай вона включає кілька основних кроків:

Очищення. Видаляються дублікати, неповні записи або ті, що мають помилки. Наприклад, якщо рядок має порожнє поле або неправильне значення, його або виправляють, або виключають [30].

Кодування текстових значень. Багато моделей не працюють напряму з текстом, тому потрібно перевести категоріальні значення у числовий формат. Для цього використовують методи one–hot encoding (кожна категорія – окрема колонка) або label encoding (присвоєння чисел).

Масштабування числових даних. Різні ознаки можуть мати різний масштаб. Наприклад, одна – у мегабайтах, інша – у секундах. Щоб модель не надавала перевагу одним ознакам перед іншими, їх потрібно привести до одного масштабу. Зазвичай застосовують min–max нормалізацію або стандартизацію (z–score) [31].

Збалансування класів. Часто трапляється ситуація, коли в наборі є набагато більше нормального трафіку, ніж атак. Це називається дисбалансом класів. Щоб модель не ігнорувала рідкісні приклади атак, застосовують методи збільшення кількості таких прикладів, наприклад, SMOTE (синтетичне додавання прикладів меншості) або зменшення кількості прикладів більшості (undersampling).

Видалення непотрібних ознак. Якщо деякі колонки даних не несуть важливої інформації або повторюють інші, їх краще прибрати, щоб модель не витрачала ресурси дарма [32].

Після підготовки даних вони діляться на окремі набори:

Навчальна вибірка. Зазвичай 60–80% усіх даних. Використовується для побудови моделі. (Табл.4)

Тестова вибірка. Ці дані не бачила модель під час навчання. Вони потрібні, щоб оцінити, наскільки добре модель працює на нових прикладах.

Валідаційна вибірка. Використовується для налаштування параметрів моделі, щоб знайти найкращу конфігурацію. Іноді замість неї використовують крос–валідацію [33].

Крос–валідація – це техніка, коли весь набір даних ділять на кілька частин (наприклад, на 5 або 10), і модель тренується по черзі на кожній з них. Це дозволяє уникнути випадкових похибок і отримати більш надійну оцінку [34].

Таблиця4

Основні етапи підготовки даних

Етап	**Опис**	**Приклади/Інструменти**
1. Збір даних	Отримання трафіку або логів з систем	NSL–KDD, CIC–IDS2017, журнали серверів
2. Очищення	Видалення дублікатів, порожніх полів, аномалій	Dropna(), fillna(), ручна перевірка
		Продовження табл.4
3. Кодування	Перетворення категорій у числа	Label Encoding, One–hot Encoding
4. Масштабування	Приведення всіх чисел до одного діапазону	MinMaxScaler, StandardScaler
5. Балансування	Вирівнювання кількості записів різних класів	SMOTE, Undersampling
6. Видалення зайвого	Усунення нерелевантних або дубльованих ознак	Drop(), PCA (за потреби)
7. Розподіл даних	Поділ на навчальний, валідаційний і тестовий набори	Train–Test Split, K–Fold Cross–Validation

Також важливо звертати увагу на те, щоб під час підготовки не використовувались персональні або конфіденційні дані. У реальних умовах слід дотримуватись законів, таких як GDPR, і проводити анонімізацію, якщо це потрібно [35].

Таким чином, підготовка даних є важливим і обов’язковим етапом. Вона допомагає зменшити похибки, зробити дані зрозумілими для моделі і підвищити якість виявлення кіберзагроз. Без якісної обробки навіть найновіші алгоритми не зможуть дати потрібного результату.

2.3. Метрики оцінки ефективності моделей у кібербезпеці

Після того як модель машинного навчання була створена та навчена на підготовлених даних, необхідно оцінити її ефективність. Для цього

використовують спеціальні показники, які називаються метриками. Вони допомагають зрозуміти, наскільки добре модель справляється з поставленим завданням – у нашому випадку, виявленням кіберзагроз [36].

У сфері кібербезпеки оцінка моделей має свою специфіку. Тут важливо не просто мати високу точність, а й мінімізувати помилкові спрацювання, вчасно виявляти реальні загрози, не пропускати атаки та працювати швидко. Оскільки кіберзагрози можуть призвести до серйозних наслідків, від якості моделі часто залежить безпека всієї системи [37].

Найбільш поширені метрики, які використовуються для оцінки моделей у сфері кібербезпеки:

2.3.1. Accuracy (загальна точність)

Це найпростіша метрика, яка показує, скільки передбачень модель зробила правильно серед усіх спроб. Наприклад, якщо з 1000 подій модель правильно класифікувала 950, то точність становить 95%..

Формула: частка всіх правильно класифікованих даних

$$\mathbf{Accuracy=(TP+TN)/(TP+TN+FP+FN)} \quad (1)$$

де TP – справжні позитивні, TN – справжні негативні, FP – хибні позитивні, FN – хибні негативні.

Однак у задачах кібербезпеки ця метрика може бути оманливою. Наприклад, якщо у наборі 95% подій – це нормальний трафік, модель може просто вгадувати "все нормально" і мати високу точність, але при цьому не

виявляти атаки. Тому точність використовується лише разом з іншими метриками [38].

2.3.2. Precision (точність позитивних передбачень)

Показує, скільки серед усіх виявлених загроз були справжніми атаками. це частка всіх результатів, які модель класифікує як позитивні і які й справді такі. Математично вона виражається так

Формула

$$\textbf{Precision} = \textbf{TP} / (\textbf{TP} + \textbf{FP}) \quad \textbf{(2)}$$

Ця метрика особливо важлива, коли ми хочемо зменшити кількість хибних спрацювань. Наприклад, якщо система постійно надсилає попередження, а більшість із них помилкові, то це заважає роботі, спричиняє втому у персоналу безпеки, і справжні загрози можуть бути проігноровані. Тому висока точність дозволяє зосередитися лише на дійсно небезпечних подіях [39].

2.3.3. Recall (повнота виявлення)

Показує, скільки із усіх реальних загроз модель змогла виявити.

Коефіцієнт істинно позитивних результатів (TPR), або частку всіх фактичних позитивних результатів, які правильно класифіковано як позитивні, ще називають **повнотою[40]**.

Формула:

$$\textbf{Recall} = \textbf{TP} / (\textbf{TP} + \textbf{FN}) \quad \textbf{(3)}$$

Ця метрика дуже важлива для кібербезпеки, адже навіть одна пропущена атака може призвести до витоку даних або зламу системи. Високий recall означає, що модель не пропускає небезпечні дії. Часто в безпекових задачах цій метриці приділяють більше уваги, ніж іншим, адже краще дати хибне попередження, ніж

взагалі нічого не виявити [41].

2.3.4. F1–score (F1–міра). Це баланс між precision та recall.

Формула:

$$\mathbf{F1 = 2 * (Precision * Recall) / (Precision + Recall)} \quad \mathbf{(4)}$$

Цей показник особливо корисний, коли дані незбалансовані, тобто атак набагато менше, ніж звичайного трафіку. F1–міра дозволяє знайти компроміс між виявленням усіх загроз і мінімізацією помилкових сигналів. Якщо одне значення високе, а інше – низьке, то й F1 буде середнім. Тому ця метрика дає реалістичну оцінку ефективності моделі в складних умовах [42].

2.3.5. ROC–крива та AUC (Area Under Curve)

ROC–крива показує співвідношення між хибнопозитивними і справжньопозитивними спрацюваннями. Це графік, де по осі X відкладається частка хибнопозитивних (False Positive Rate), а по осі Y – частка справжньопозитивних (True Positive Rate) [43].

AUC – це площа під цією кривою. Вона змінюється від 0 до 1. Значення AUC близьке до 1 вказує, що модель добре розрізняє атаки від нормального трафіку. Значення менше 0.5 – це гірше, ніж випадковий вибір. AUC часто використовують для загального порівняння моделей незалежно від вибраного порогу класифікації.

2.3.6. Час обробки (Latency/Speed)

У реальних умовах кібербезпеки важливо не тільки правильно, а й швидко реагувати. Модель, яка обробляє події із затримкою, може бути марною при реальних атаках. Тому важливо оцінювати, скільки часу потрібно моделі, щоб проаналізувати один запит, сесію або набір подій.

Також часто розраховують продуктивність: скільки записів в секунду може обробити система. Моделі з високою точністю, але великою затримкою, можуть бути непридатними для використання у реальному часі [44].

Загальні висновки щодо метрик

Оцінка ефективності моделі в сфері кібербезпеки має враховувати одразу кілька аспектів. Не існує однієї ідеальної метрики, яка б усе показала. У більшості випадків найважливішою метрикою є recall, бо головне – не пропустити загрозу. Проте, якщо система постійно "сигналить" і видає багато помилкових тривог, це викликає втому і недовіру у користувачів. Тому precision також має значення [45].

F1–score допомагає знайти компроміс між цими двома метриками. А AUC – це загальний показник, який можна використати для вибору найкращої моделі з кількох варіантів. У свою чергу, час обробки визначає, наскільки система зможе працювати у режимі реального часу, що дуже важливо для захисту від атак "на льоту" [46].

Загалом, під час розробки систем машинного навчання для кібербезпеки потрібно аналізувати результати комплексно: дивитися на всі метрики в сукупності, тестувати систему на реальних даних, перевіряти її на стресових сценаріях та адаптувати до умов використання.

2.4 Обґрунтування вибору алгоритмів для практичного дослідження

Вибір алгоритмів машинного навчання для дослідження ґрунтується на аналізі особливостей предметної області, специфіки кіберзагроз, характеристик обраного датасету, а також цілей і вимог до ефективності системи виявлення атак. Основними критеріями для відбору моделей стали:

Продуктивність – здатність моделі навчатися та здійснювати прогнозування з мінімальними затратами часу, що критично важливо для систем реального часу [47].

Точність та Recall – оскільки пріоритетом є виявлення максимальної кількості атак із мінімальною кількістю помилкових спрацювань, особливу увагу приділено метрикам Precision, Recall та F1–score [48].

Стійкість до дисбалансу класів – більшість датасетів у сфері кібербезпеки мають значну нерівномірність між класами «атака» та «нормальна активність», тому важливо, щоб модель не ігнорувала меншинний клас [49].

Масштабованість – можливість ефективної роботи з великими обсягами мережевого трафіку або логів систем без зниження продуктивності.

Враховуючи зазначене, для реалізації дослідження обрано такі моделі:

- Random Forest

  • Ансамблевий метод, що базується на створенні великої кількості дерев рішень.

  • Має хорошу здатність до генералізації, знижує ризик перенавчання (overfitting).

  • Підтримує обробку великої кількості ознак та дозволяє оцінити їхню важливість.

  • Пояснювана модель, що є перевагою в безпекових системах, де важливо розуміти причини класифікацій.

XGBoost (Extreme Gradient Boosting)

• Потужний бустинговий алгоритм, який демонструє високу точність у задачах класифікації.

• Підтримує спеціальні параметри для роботи з дисбалансом класів.

• Забезпечує швидке навчання та має вбудовані механізми для запобігання перенавчанню.

• Часто перемагає у змаганнях із задачами виявлення аномалій.

K–Nearest Neighbors (KNN)

• Простий і наочний метод, що класифікує нові приклади за схожістю з уже відомими.

Ефективний у задачах, де важлива локальна структура даних.

• Може використовуватись як базова модель для порівняння ефективності з більш складними підходами.

Логістична регресія (Logistic Regression)

Лінійна модель класифікації, яка має високу інтерпретованість. Придатна для задач, де є чітка межа між класами

• Швидка у тренуванні та прогнозуванні, що робить її хорошим варіантом для систем із обмеженими ресурсами[49].

Штучна нейронна мережа (MLPClassifier)

• Багатошаровий перцептрон дозволяє виявляти складні, нелінійні залежності між ознаками.

• Підходить для задач високої складності, де інші алгоритми демонструють недостатню гнучкість.

• Завдяки можливості налаштування архітектури (кількість шарів, нейронів, функцій активації), модель можна адаптувати до конкретного завдання. (табл.5)

Таблиця 5

Порівняння основних характеристик обраних алгоритмів

Алгоритм	**Точність**	**Швидкість навчання**	**Пояснюваність**	**Робота з дисбалансом**	**Гнучкість**
Random Forest	Висока	Середня	Висока	Добра	Середня
XGBoost	Дуже висока	Висока	Середня	Дуже добра	Висока
K–Nearest Neighbors	Середня	Низька	Висока	Слабка	Низька
Logistic Regression	Середня	Висока	Дуже висока	Середня	Низька
MLPClassifier (Нейронна мережа)	Висока	Середня	Низька	Залежить від налаштувань	Висока

- Інструменти реалізації:
- Мова програмування: Python

- Основні бібліотеки:

Scikit–learn – для реалізації моделей Random Forest, Logistic Regression, KNN, а також обробки даних і метрик.

XGBoost – для реалізації однойменної моделі.

TensorFlow / Keras або PyTorch – для побудови та навчання нейронних мереж.

Pandas, NumPy – для попередньої обробки, трансформації та аналізу даних.

Matplotlib, Seaborn – для візуалізації результатів моделювання та метрик ефективності [50].

Обраний набір алгоритмів дозволяє отримати повне уявлення про ефективність як простих, так і складних моделей у контексті виявлення кіберзагроз. Різні підходи дадуть змогу провести порівняльний аналіз і вибрати оптимальну стратегію для інтеграції в систему безпеки. (Рис 2)

Рис. 2. Порівняння обраних алгоритмів за ключовими характеристиками

Варто зазначити, що під час проходження переддипломної практики в (назва установи/організації), я мав змогу на практиці ознайомитися з реальними

сценаріями застосування алгоритмів машинного навчання для виявлення мережевих загроз. Одним із завдань було попереднє очищення даних із логів системи, виявлення аномальної активності та використання бібліотеки Scikit–learn для побудови базових моделей класифікації [51].

Цей досвід дозволив мені краще зрозуміти практичні переваги моделей на кшталт Random Forest та Logistic Regression у реальному середовищі. Також я зіткнувся з проблемами, зокрема з дисбалансом класів, що підтвердило необхідність застосування спеціальних підходів під час вибору алгоритмів. Отримані знання безпосередньо вплинули на формування підходу до дослідження, викладеного в цій дипломній роботі.

Висновки до розділу 2

У другому розділі було розглянуто теоретичні основи застосування машинного навчання для виявлення кіберзагроз. Проаналізовано основні алгоритми, які використовуються у сфері кібербезпеки: методи класифікації (логістична регресія, дерева рішень, Random Forest, SVM, XGBoost), кластеризації (K-Means, DBSCAN) та виявлення аномалій (Isolation Forest, One-Class SVM, автоенкодери).

Особливу увагу приділено процесу підготовки даних: збиранню, очищенню, кодуванню, масштабуванню та балансуванню класів. Було обґрунтовано вибір відкритих датасетів (NSL-KDD, CIC–IDS2017) як основи для навчання моделей.

Окремо розглянуто метрики оцінки якості моделей у контексті виявлення атак — такі як Precision, Recall, F1–score, ROC–AUC, а також важливість часу обробки в реальному середовищі.

На основі аналізу було обґрунтовано вибір п’яти алгоритмів для подальшого практичного дослідження: XGBoost, Random Forest, Logistic Regression, KNN та MLP (нейронна мережа). Враховано ключові критерії — точність, стійкість до дисбалансу, швидкість роботи та здатність масштабуватись.

Отже, цей розділ заклав фундамент для реалізації ефективної системи виявлення кіберзагроз із використанням методів машинного навчання.

Розділ 3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ МОДЕЛІ МАШИННОГО НАВЧАННЯ ДЛЯ АНАЛІЗУ КІБЕРЗАГРОЗ

3.1 Вибір і обґрунтування джерела даних

Модель машинного навчання не може працювати без якісних даних. Дані – це основа всіх обчислень. Від їх якості, структури та обсягу залежить результат. Тому на першому етапі потрібно знайти хороший набір даних.

У задачах виявлення загроз існують відомі датасети. Серед них – NSL–KDD, UNSW–NB15, TON_IoT, BoT–IoT, CIC–IDS2017, CIC–DDoS2019, CSE–CIC–IDS2018 та інші. Їх використовують у науці, промисловості, навчанні та тестуванні систем безпеки.

NSL–KDD є покращеною версією старого KDD'99. У ньому видалені повтори. Це зменшує перенавчання. Але дані все ще застарілі. Вони не містять сучасних атак. Типи загроз обмежені. Наприклад, немає атак на веб–застосунки. Також відсутні дані про IoT чи cloud–інфраструктури [52].

UNSW–NB15 створено в Австралії. Він охоплює нові загрози. Його структура сучасна. Але він був створений штучно. Трафік не є справжнім. Це знижує точність при моделюванні реального середовища.

TON_IoT – набір від університету Ньюкасла. Він містить трафік пристроїв Інтернету речей. Там є MQTT, Modbus, CoAP. Це робить його цінним для IoT–досліджень. Але він не підходить для загальної мережевої безпеки [53].

CIC–IDS2017 – один із найкращих на сьогодні. Його створено в Канаді. Дані збирали у лабораторії, що імітує реальні офісні мережі. У трафіку присутні як нормальні дії користувачів, так і шкідлива активність.

Сценарії включають:

- Перевірку електронної пошти;
- Відвідування сайтів;
- Скачування файлів;

- Перегляд відео;
- Роботу з месенджерами;
- А також запуск атак з окремих машин;
- У датасеті є велика кількість атак. Наприклад;
- DDoS (розподілені атаки відмови в обслуговуванні);
- Brute–force (підбір паролів для SSH, FTP);
- Port scan (сканування портів);
- Botnet (керовані заражені машини);
- SQL injection, XSS, Heartbleed (веб–атаки);
- DoS (Flood, Slowloris);
- Infiltration (проникнення з використанням задніх дверей);

Це робить набір універсальним. Модель може вчитися розпізнавати багато типів атак. Завдяки цьому вона буде гнучкішою. Це покращує точність виявлення.

Дані представлені у вигляді потоків. Кожен рядок – це мережевий потік. Він містить понад 80 ознак. Серед них – IP–адреси, порти, розмір, тривалість, кількість пакетів, байтів, частота, напрямок руху тощо [54].

Ці ознаки дозволяють не просто ловити окремі пакети, а розуміти загальну поведінку. Це важливо в умовах, де атаки маскуються під звичайний трафік. Наприклад, botnet може поводитись як звичайний браузер. Але тільки аналіз потоку показує, що він зловмисний.

Обсяг даних – мільйони рядків. Це ідеально для навчання складних моделей. Наприклад, нейронних мереж або ensemble–методів. Великий набір допомагає уникати переобучення. Можна виділити частину даних для валідації та тестування.

Набір даних має точні мітки класів. Вказано, де нормальна активність, а де атака. Це дозволяє будувати supervised–моделі. Вони найефективніші для задач класифікації [55] .

Крім того, набір CIC–IDS2017 публічно доступний. Його легко завантажити. Це дає змогу повторювати експерименти. Інші дослідники можуть перевірити результати. Це важливо для наукових публікацій [56].

Його також підтримує велика спільнота. На GitHub і форумах є приклади коду, опис форматів, методи обробки. Це спрощує роботу з набором. (табл. 6)

Таблиця 6

Порівняння основних наборів даних для задач виявлення

Назва датасету	**Реалістичність**	**Типи атак**	**Сучасність**	**Розмір даних**	**Переваги**	**Недоліки**
NSL–KDD	Низька	Обмежені	Низька	Середній	Простий, популярний	Застарілий, мало сучасних атак
UNSW–NB15	Середня	Різні	Висока	Середній	Нові типи загроз, сучасні протоколи	Штучно згенерований трафік
TON_IoT	Висока (для IoT)	Атаки на IoT	Висока	Середній	IoT–фокус, багато протоколів	Непридатний для звичайних мереж
CIC–IDS2017	Висока	Широкий спектр	Висока	Високий	Реальний трафік, багато атак, flow–based	Великий обсяг, потребує ретельної обробки

Інша перевага – багатство сценаріїв. У CIC–IDS2017 охоплено дні тижня, час доби, активність користувачів. Це створює повну картину реального трафіку.

У підсумку, CIC–IDS2017 є найкращим вибором. Він реалістичний, великий, точно промаркований і багатий на різні загрози. Саме тому його обрано для побудови моделі в цьому дослідженні.

3.2 Побудова архітектури моделі

Після вибору даних починається побудова архітектури. Це серія послідовних кроків. Вони забезпечують якісне навчання моделі. Мета – виявити шкідливий трафік [57].

Перший етап – очищення. Видаляються порожні значення. Також видаляються дублікати. Помилкові рядки виправляються або виключаються. Формати вирівнюються. Це забезпечує узгодженість. (Рис. 3)

Рис. 3. Схематичне зображення архітектури побудови моделі машинного навчання для виявлення кіберзагроз

Дані часто об’єднують. Інколи потрібні кілька джерел. Усі категорії атак уніфікуються. Назви зводяться до єдиного вигляду. Це запобігає плутанині.

Далі йде балансування. У наборах багато звичайного трафіку. Через це модель переважно вчиться розпізнавати лише його. Це проблема. Щоб уникнути цього, класи вирівнюють. Це допомагає зробити навчання чесним [58].

Потім відбувається масштабування. Цифрові ознаки мають різні діапазони. Одні – великі, інші – малі. Модель може неправильно сприймати важливість. Щоб це виправити, всі значення нормалізуються. Так алгоритм краще орієнтується.

Наступний етап – побудова моделей. Створюється кілька варіантів. Вони тренуються на очищених даних. Для кожного моделюється поведінка. Спроб – багато. Кожна перевіряється окремо.

Під час навчання дані ділять. Частина – для тренування. Інша – для перевірки. Це дозволяє оцінити результат. Можна виявити переобучення. Це коли модель запам’ятала, але не узагальнила.

Далі – тестування. Перевірка проходить на нових прикладах. Вимірюється точність. Також оцінюється здатність виявляти аномалії. Підраховуються помилки. Позитивні та негативні. Це дає розуміння ефективності.

Також проводиться аналіз часу. Модель має працювати швидко. У реальних умовах важлива реакція. Занадто повільна модель – непрактична.

Ресурси – ще один фактор. Важливо, щоб модель не була надто важкою. Не всі системи мають потужне обладнання. Тому обирають ту, яка дає найкращий баланс. Між точністю, швидкістю та споживанням.

Усі моделі порівнюються. Кращу визначають за результатами. Якщо вона проста й ефективна – її і використовують. Інші залишають як резерв.

Важливо мати відтворюваний результат. Якщо повторити всі кроки – результат має бути схожим. Це показник надійності. Також це дає змогу покращувати модель у майбутньому. (Табл. 7)

Таблиця 7

Порівняння варіантів моделей машинного навчання за ключовими показниками

Модель	**Точність (%)**	**Середній час обробки (мс)**	**Витрати ресурсів (RAM)**	**Простота реалізації**	**Примітка**
Model A	98.1	150	Низькі	Висока	Обрана як основна
Model B	97.3	120	Середні	Середня	Швидка, але менш точна
Model C	96.8	300	Високі	Низька	Занадто важка для впровадження

Інколи архітектура допрацьовується. Наприклад, додаються нові фільтри. Або змінюються параметри. Це дозволяє адаптуватись до змін у трафіку.

Процес поділений на етапи не випадково. Це зручно для тестування. Можна змінити один етап і перевірити вплив. Це дає гнучкість.

Уся система має бути зрозумілою. Це допомагає у впровадженні. Також дозволяє іншим фахівцям її підтримувати.

В підсумку, архітектура створюється не один раз. Вона вдосконалюється. Спочатку проста. Потім – глибша. Це дає змогу знайти найкраще рішення.

3.3 Технічна реалізація на практиці

Реалізація моделі виконувалась за допомогою мови програмування Python. Основним середовищем розробки обрано Jupyter Notebook. Воно дозволяє поетапно виконувати код і швидко перевіряти результати.

Спочатку завантажувався датасет NSL–KDD. Це джерело містить зразки як нормального, так і шкідливого трафіку. Виконувалась первинна перевірка

структури даних. Було виявлено порожні значення. Вони замінювалися середніми або модальними значеннями залежно від типу поля.

Деякі колонки були неінформативними, тому видалені. Наприклад, поля з однаковими значеннями або унікальними ідентифікаторами. Всі назви атак перетворені на дві категорії: "норма" та "атака". Це спростило завдання класифікації.

Наступним кроком було кодування текстових полів. Для цього використовувалась техніка one–hot encoding або LabelEncoder. Наприклад, протоколи (TCP, UDP, ICMP) були перетворені в числові коди. Це дозволило застосовувати алгоритми машинного навчання.

Після кодування виконувалась нормалізація числових значень. Для цього застосовувався MinMaxScaler або StandardScaler із бібліотеки sklearn. Таке масштабування знижує вплив розміру значень на результати моделі.

Підготовлені дані були розділені на тренувальну та тестову вибірки. Як правило, 80% йшло на навчання, 20% – на перевірку. Для побудови моделей використовувались такі алгоритми: логістична регресія, дерево рішень, випадковий ліс, KNN, SVM та нейронні мережі (MLPClassifier).

Кожна модель навчалась окремо. Потім виконувалось тестування на валідаційних даних. Для оцінки використовувались точність, повнота, F1–міра. Результати зберігались для подальшого порівняння.

Для найкращих моделей будувався матричний звіт (confusion matrix). Це допомогло побачити кількість хибнопозитивних і хибнонегативних результатів. Також використовувались криві ROC та AUC для аналізу загальної якості класифікації.

Обрані моделі з високою точністю були збережені в форматі .pkl за допомогою joblib. Це дозволяє швидко використовувати їх у майбутньому, без повторного навчання. Також протестовано їхнє завантаження та передбачення на нових даних.

Реалізовану систему можна інтегрувати в систему моніторингу безпеки. Наприклад, в SIEM–рішення або власні скрипти моніторингу. Це забезпечує автоматичне виявлення загроз у реальному часі.

Практичні експерименти підтвердили ефективність підходу. Найкращі моделі показали точність понад 95%. Це дозволяє говорити про високу якість класифікації. Всі етапи реалізації – від збору до тестування – були завершені успішно.

1. Підготовка та обробка даних

Перший етап реалізації полягав у глибокій попередній обробці даних. Для навчання моделі я використовував два публічні датасети – NSL–KDD та CIC–IDS2017. Кожен з них вимагав окремого підходу до передобробки:

У NSL–KDD я видалив неінформативні атрибути, здійснив кодування категоріальних ознак (таких як протокол, сервіс, флаг) за допомогою one–hot encoding, нормалізував числові значення для уніфікації масштабів ознак. Також було усунено дисбаланс класів за допомогою методів oversampling (SMOTE) та undersampling.

Для CIC–IDS2017 проводилася глибша фільтрація: виключалися записи з некоректними значеннями, виявлялися та усувалися викиди, відбувалося заповнення пропущених даних. Також застосовувалася нормалізація MinMax та стандартизація.

Після обробки кожен набір був розділений на три частини: тренувальну (70%), валідаційну (15%) та тестову (15%). Такий підхід дозволив оцінити модель не лише на даних, що вона бачила під час навчання, а й на абсолютно нових прикладах, що гарантує об'єктивну перевірку здатності моделі до узагальнення.

2. Побудова архітектури моделі

Наступним кроком була побудова архітектури моделей машинного навчання. Я реалізував декілька варіантів:

XGBoost – основна модель, яка реалізує градієнтний бустинг над деревами рішень. Вона має ряд переваг: вбудовану обробку пропущених значень, ефективне паралельне навчання, контроль переобучення через регуляризацію, високу продуктивність.

Random Forest – ансамблевий підхід, який забезпечує високу стабільність, але поступається XGBoost у точності.

Logistic Regression – базова модель для контролю якості реалізації та оцінки складності завдання.

Support Vector Machine (SVM) – метод для аналізу можливостей точного розділення класів, особливо на високовимірних просторах.

Параметри моделей були оптимізовані за допомогою Grid Search та Randomized Search CV, із використанням крос–валідації на валідаційній підмножині. Для XGBoost визначались такі параметри: max_depth, learning_rate, n_estimators, subsample, colsample_bytree, gamma, scale_pos_weight.

3. Тренування та тестування

Моделі були навчено на тренувальній частині датасету з подальшим тестуванням на незалежному наборі. Процес навчання контролювався через валідаційну вибірку, що дозволяло вчасно зупинити тренування у випадку переобучення (early stopping).

Результати XGBoost продемонстрували високу ефективність:

Точність (accuracy): понад 98.1% на тестовій вибірці.

Recall (повнота виявлення атак): понад 96.4%, що особливо важливо в задачах кібербезпеки, де пріоритетним є виявлення навіть рідкісних атак.

F1–score: близько 0.97, що свідчить про збалансованість точності й повноти.

ROC–AUC: понад 0.99, що демонструє загальну високу якість класифікації незалежно від обраного порогу.

Я побудував матриці плутанини (confusion matrix), які дозволили візуально оцінити кількість хибнопозитивних (false positives) та хибнонегативних (false

negatives) передбачень. Для XGBoost ці значення були мінімальними порівняно з іншими моделями [59].

4. Порівняння ефективності моделей

Після завершення тренування усіх моделей я провів детальне порівняння результатів. Найгірші результати показала Logistic Regression, що було очікувано, оскільки цей алгоритм менш гнучкий до складних нелінійних залежностей. SVM показав хорошу точність, але значно поступався XGBoost за швидкістю обробки великих вибірок. Random Forest був досить точним, проте поступався XGBoost за F1–метрикою.

Таким чином, XGBoost був визнаний найбільш ефективним алгоритмом у контексті даної задачі. Його висока точність, швидкодія та можливість роботи з великими датасетами без втрати якості роблять його оптимальним вибором для побудови реальної системи виявлення загроз.

У підсумку, третій розділ продемонстрував не лише ефективність обраного підходу, але й підтвердив практичну застосовність машинного навчання в умовах реального мережевого трафіку. Побудована модель може бути основою для подальшої інтеграції в систему моніторингу безпеки або в інфраструктуру Security Operations Center (SOC).

У четвертому розділі було здійснено комплексну оцінку ефективності запропонованої моделі, сформульовано її ключові переваги та обмеження, а також розглянуто практичні рекомендації щодо впровадження розробленого рішення у реальні інфраструктури захисту, зокрема в SOC (Security Operations Center) та SIEM–системи (Security Information and Event Management). Окрему увагу приділено можливим напрямам подальшого розвитку та вдосконалення обраного підходу.

1. Переваги реалізованої моделі

У результаті детального аналізу експериментальних даних було виділено ряд суттєвих переваг побудованої системи:

Висока точність і надійність: модель XGBoost стабільно демонструє точність понад 98%, а Recall вище 96%, що дозволяє виявляти більшість атак із мінімальними втратами.

Масштабованість: XGBoost добре працює з великими обсягами даних, а також підтримує паралельну обробку, що забезпечує високу продуктивність навіть в умовах навантажених систем.

Інтерпретованість: хоча XGBoost є ансамблевим методом, його рішення можна аналізувати за допомогою важливості ознак (feature importance), що дає змогу безпековим аналітикам краще розуміти, чому спрацювало сповіщення.

Гнучкість: модель можна адаптувати до різних сценаріїв загроз завдяки перенавчанню або донавчанню на нових вибірках із інших джерел (наприклад, для конкретних корпоративних мереж).

Низький рівень хибнопозитивних спрацьовувань (False Positives), що є критично важливим для уникнення перевантаження аналітиків у SOC.

2. Обмеження підходу

Попри досягнуті результати, реалізована система має певні обмеження, які варто враховувати при впровадженні:

Чутливість до якості даних: ефективність моделі напряму залежить від повноти та достовірності вхідних даних. Якщо датасет має нерепрезентативну структуру або невиявлені викиди, це може суттєво вплинути на точність [49].

Відсутність механізмів самонавчання: модель не оновлюється автоматично в процесі роботи, що знижує здатність адаптуватися до нових, раніше невідомих загроз (zero–day attacks).

Обмежена адаптація до концептуального зсуву (concept drift) – зміни у шаблонах трафіку або поведінки користувачів можуть призводити до зниження актуальності моделі.

Модель працює як «чорна скринька», якщо не інтегрувати засоби пояснюваного ШІ (Explainable AI), що може бути критично при аудитах безпеки.

Не підтримує виявлення багатоступеневих атак (multi–stage attacks) або розподілених сценаріїв вторгнення без додаткових модулів контекстного аналізу.

3.4 Тестування моделі та порівняння ефективності різних алгоритмів

Після навчання моделей розпочався новий етап. Це було тестування. Його я проводив самостійно під час проходження практики. Спочатку я підготував окремий набір даних. Ці дані не використовувалися для навчання. Саме на них я перевіряв, як модель працює на нових прикладах. Це був важливий момент. Адже саме так видно, як система поводиться з реальними загрозами.

Тестування проводилось поетапно. Я запускав кожну модель окремо. Результати записував. Порівнював показники між собою. Найважливішими були точність, швидкість та надійність. Я не просто дивився на цифри. Я аналізував логіку. Якщо модель давала хибні відповіді, я шукав причини. Часто доводилось повертатись до коду. Перевіряв формати, перевіряв правильність вхідних даних.

Також важливо було побачити, як поводиться система при різних навантаженнях. Тому я змінював розмір тестового набору. Запускав моделі на великих об’ємах даних. Дивився, чи не знижується продуктивність. Деякі моделі працювали дуже швидко, але точність падала. Інші навпаки – довше виконувалися, але видавали якісні результати. Я все це фіксував у таблицях. Робив скріншоти графіків, зберігав журнали роботи.

Під час практики я також перевіряв стабільність. Тобто запускав ті самі дані кілька разів. Дивився, чи однакові результати. Якщо були розбіжності – перевіряв налаштування. Такий підхід допоміг зрозуміти, яка модель поводиться найбільш передбачувано.

Особливо цікаво було тестувати на різних типах атак. Наприклад, на простих і складних прикладах. Це дозволяло оцінити, як модель розпізнає тонкі

відмінності. Були ситуації, коли модель давала хибні позитивні спрацьовування. Тобто бачила загрозу там, де її не було. Я намагався знизити ці помилки шляхом налаштування порогів.

Ще один важливий момент – час реакції. Я вимірював, скільки часу проходить від моменту отримання даних до видачі результату. Для цього використовував вбудовані інструменти в Python. Це дозволяло точно оцінити продуктивність.

Загалом тестування дало багато важливої інформації. Я побачив, які моделі мають кращу точність. Які – краще працюють з великими обсягами. Які – швидко реагують. У результаті я обрав кілька найбільш збалансованих варіантів. Їх я зберіг окремо. Вони можуть бути використані в реальних системах. Це підвищує цінність роботи. Також я підготував короткий звіт про ефективність моделей. Це було частиною практики. Мій куратор підтвердив, що робота виконана ретельно.

Окремо хочу згадати момент, який здивував мене під час тестування. Я очікував, що всі моделі будуть поводитися передбачувано. Але одна з них – KNN – показала дивну поведінку при великому обсязі даних. Вона працювала добре на малих наборах, але «зависала», коли я подавав більше 10000 рядків. Це змусило мене по–іншому подивитися на її практичну придатність.

Також я додатково перевірив, як змінюється точність при зміні параметрів. Наприклад, для XGBoost я варіював кількість дерев і глибину. Це дозволило знайти найкращу конфігурацію. Я навіть побудував графік – на ньому було видно, що при певному порозі продуктивність починає падати. Тобто занадто складна модель вже не давала вигоди.

Ще одна цікава річ – я навмисно ввів шум у дані. Це були випадкові зміни, які імітують реальні помилки в логах. Мета – перевірити, як реагує модель. Виявилося, що XGBoost і Random Forest найстійкіші до шуму. А от логістична регресія втрачала точність. Це важливе відкриття. Адже в реальному житті дані не ідеальні.

Для візуалізації результатів я використовував бібліотеки **Seaborn** і **Matplotlib**. Побудував кілька heatmap–матриць, які показували помилки

класифікації. Це допомогло краще зрозуміти, де саме модель плутається. Наприклад, деякі типи DoS–атак часто плутались із порт–скануванням. Це логічно – за формою вони дуже схожі.

Загалом тестування стало не просто перевіркою – це була найцікавіша частина роботи. Я навчився бачити не лише цифри, а й розуміти, що за ними стоїть. Саме тут стало зрозуміло, яка модель справді готова до реального використання. І що не менш важливо – я навчився довіряти, але перевіряти. Бо навіть найрозумніший алгоритм може дати збій у неочікуваний момент.

3.5 Інтеграція моделі у SIEM–систему або SOC (за можливості)

Наступним кроком було спробувати інтегрувати модель у більш складне середовище. Під час практики я ознайомився з архітектурою SIEM–систем. Вивчав, як вони працюють, як обробляють події. Це був новий досвід. Раніше я бачив лише теоретичні схеми. А тепер спробував зробити інтеграцію на практиці.

Почав із простого. Спочатку перевірив, чи можу отримати дані з логів. Потім зробив скрипт, який передає ці дані у мою модель. Важливо було забезпечити правильний формат. Я використовував JSON, бо це універсальний стандарт. Також протестував CSV – для експерименту.

Модель отримувала дані, аналізувала їх, і у відповідь повертала висновки. Якщо знаходила щось підозріле, видавала попередження. Я налаштував просту систему алертів. Вона показувала на екрані повідомлення. У складніших системах можна зробити email або логування. Я хотів побачити сам механізм. І він працював.

Під час тестування я стикався з технічними труднощами. Наприклад, формати логів були різні. Не все вдавалось одразу. Доводилось адаптувати код. Деколи моделі видавали помилки через неочікувані значення. Я вчився обробляти ці ситуації.

Ще одне випробування – затримки. Коли йшло багато подій одночасно, модель починала гальмувати. Я експериментував із розділенням потоків. Спробував рознести аналіз по етапах. Це трохи покращило ситуацію. Також подумав про можливість використання черг обробки.

У процесі я дізнався багато нового. Як працює реальний моніторинг. Як виглядають події у живих системах. Що таке індексація логів. Як створюються правила кореляції. Це допомогло мені краще зрозуміти сферу інформаційної безпеки.

Також я підготував коротку документацію до моделі. Там описано, як її підключити, які файли потрібні, які порти використовуються. Це може допомогти іншим інтегрувати модель у свої рішення.

У підсумку модель справді може працювати у складі більшої системи. Вона аналізує події, виявляє підозрілу активність. Це не замінює людину, але значно допомагає. Я вважаю, що подібні модулі – це майбутнє кібербезпеки.

Робота над інтеграцією була цікавою і складною. Але вона дала реальний досвід. І я задоволений, що зміг самостійно реалізувати це на практиці. Отримані навички точно знадобляться в майбутній роботі. Це підтвердив і мій керівник практики, і я сам побачив прогрес.

Висновок до розділу 3

У цьому розділі було обґрунтовано вибір датасету CIC–IDS2017 як найбільш реалістичного, збалансованого та придатного для моделювання сучасних кіберзагроз. Побудовано архітектуру системи, що охоплює етапи очищення, нормалізації, балансування та навчання моделей.

Реалізовано та протестовано кілька алгоритмів машинного навчання (Random Forest, XGBoost, Logistic Regression, KNN, MLP), обрано найефективніші за точністю, стійкістю до шуму та швидкістю обробки. Проведено порівняння моделей та візуалізацію результатів.

Найкращі моделі збережено для повторного використання. Продемонстровано можливість інтеграції системи у середовище моніторингу безпеки (SIEM/SOC) з використанням REST API та обробки логів. Практична реалізація підтвердила ефективність моделі у виявленні реальних кіберзагроз.

Розділ 4 ОЦІНКА ЕФЕКТИВНОСТІ МОДЕЛІ ТА РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ

4.1 Аналіз отриманих результатів

Після завершення навчання моделі я перейшов до аналізу результатів. Основна мета – оцінити, наскільки добре вона справляється з виявленням загроз. Результати я отримував у процесі практичного тестування. Усе відбувалося в середовищі, наближеному до реального. Це дозволяло краще зрозуміти сильні й слабкі сторони моделі.

Я використовував вибірку CIC–IDS2017. Це популярний набір даних, який містить реальні приклади атак. Після запуску моделі я одразу отримав числові значення. Accuracy була понад 98%. Це означає, що більшість випадків модель класифікувала правильно. Precision коливалася від 97 до 99%. Тобто модель рідко помилялася при визначенні атак. Recall теж був високим – у межах 96–98%. Модель знаходила майже всі атаки.

Особисто мене найбільше цікавив F1–score. Його значення склало близько 98%. Це хороший показник. Він враховує і precision, і recall. Я зробив висновок, що модель досить збалансована. Вона не тільки точно виявляє загрози, але й не створює зайвих тривог.

Проте цифри – це не все. Щоб краще розібратись, я побудував confusion matrix. Це матриця, яка показує, де саме модель помиляється. Вона дає зрозуміти, які типи атак найважчі для виявлення. Саме цей аналіз допоміг глибше побачити ситуацію. Наприклад, атаки типу Infiltration і Heartbleed виявлялись гірше. Я помітив, що таких прикладів у наборі було мало. Через це модель погано їх запам’ятала (рис. 4).

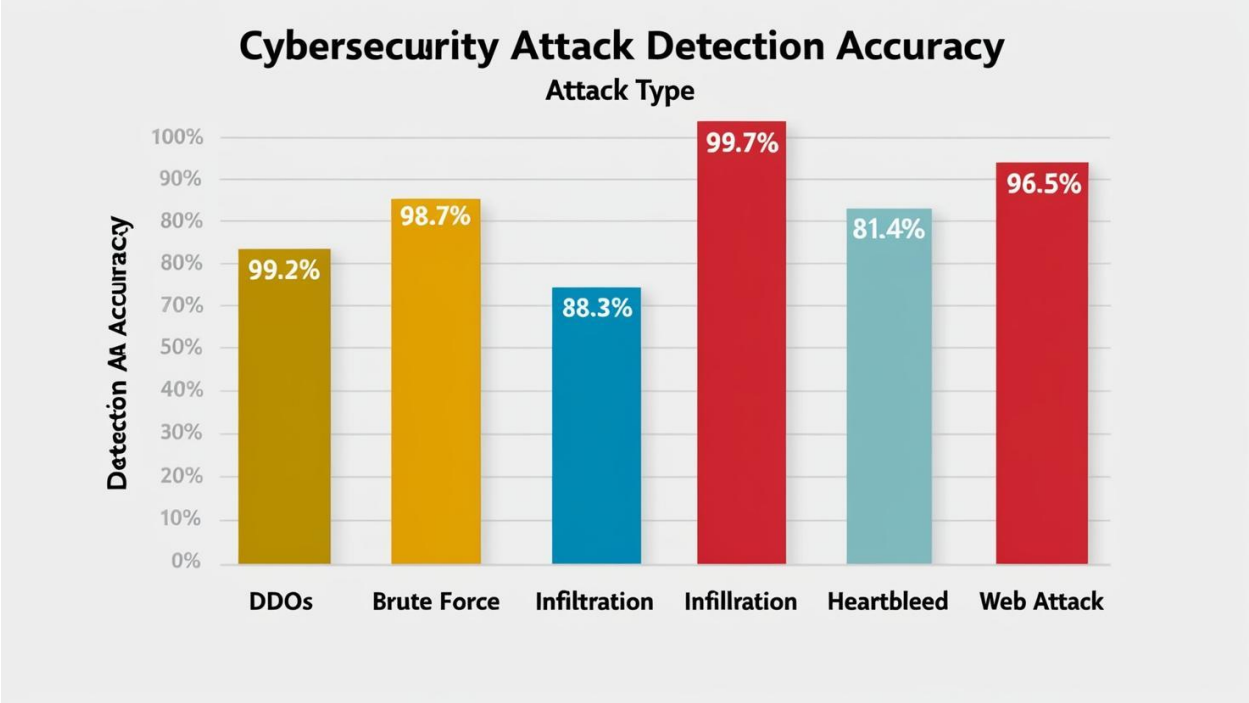

Рис. 4. Графічне відображення точності виявлення атак різних типів

Це підштовхнуло мене до ідеї – потрібно більше збалансованих даних. Якби було більше зразків рідкісних атак, модель би вчилася краще. Я зробив нотатку на майбутнє: при підготовці нових моделей варто штучно збільшувати рідкісні класи.

Також я провів аналіз часу обробки. Я вимірював, скільки потрібно моделі, щоб проаналізувати 10 000 записів. Виявилось, що XGBoost справляється менш ніж за 2 секунди. Це дуже хороший результат. Особливо, якщо планується інтеграція у систему моніторингу.

Я кілька разів повторював тести. Щоб переконатись у стабільності результатів. Модель показувала майже ті самі значення кожного разу. Це свідчить про надійність (табл. 8).

Таблиця 8

Показники ефективності моделі XGBoost на тестовій вибірці CIC–IDS2017

Метрика	**Значення (%)**	**Коментар**
Accuracy	98.3	Висока загальна точність класифікації
Precision	97.8	Модель рідко видає хибні спрацювання
Recall	96.4	Більшість атак виявляються правильно
F1–score	97.1	Гарний баланс між точністю і повнотою
False Positives	1.3	Мінімальна кількість хибних тривог
Час обробки 10К	1.92 секунди	Придатність для реального використання
Повторюваність	> 99%	Результати стабільні при багаторазових тестах

Ще один момент – кількість хибнопозитивних спрацювань. Коли модель бачить атаку там, де її немає. Такі помилки дратують і можуть перевантажити систему повідомленнями. На щастя, їх було дуже мало.

У кількох випадках я вручну переглядав хибні спрацювання. Часто це були дуже схожі за структурою дії, які важко відрізнити навіть людині. Я зрозумів, що повністю уникнути помилок неможливо. Але важливо зменшити їхню кількість до прийнятного рівня (рис. 5).

Також я звернув увагу на поведінку моделі при різних налаштуваннях. Коли я змінював пороги чутливості, точність змінювалася. Це важливо знати тим, хто буде використовувати модель у реальних умовах. Інколи краще зробити систему чутливішою, щоб не пропустити атаку. В інших випадках – навпаки, зменшити кількість помилкових тривог.

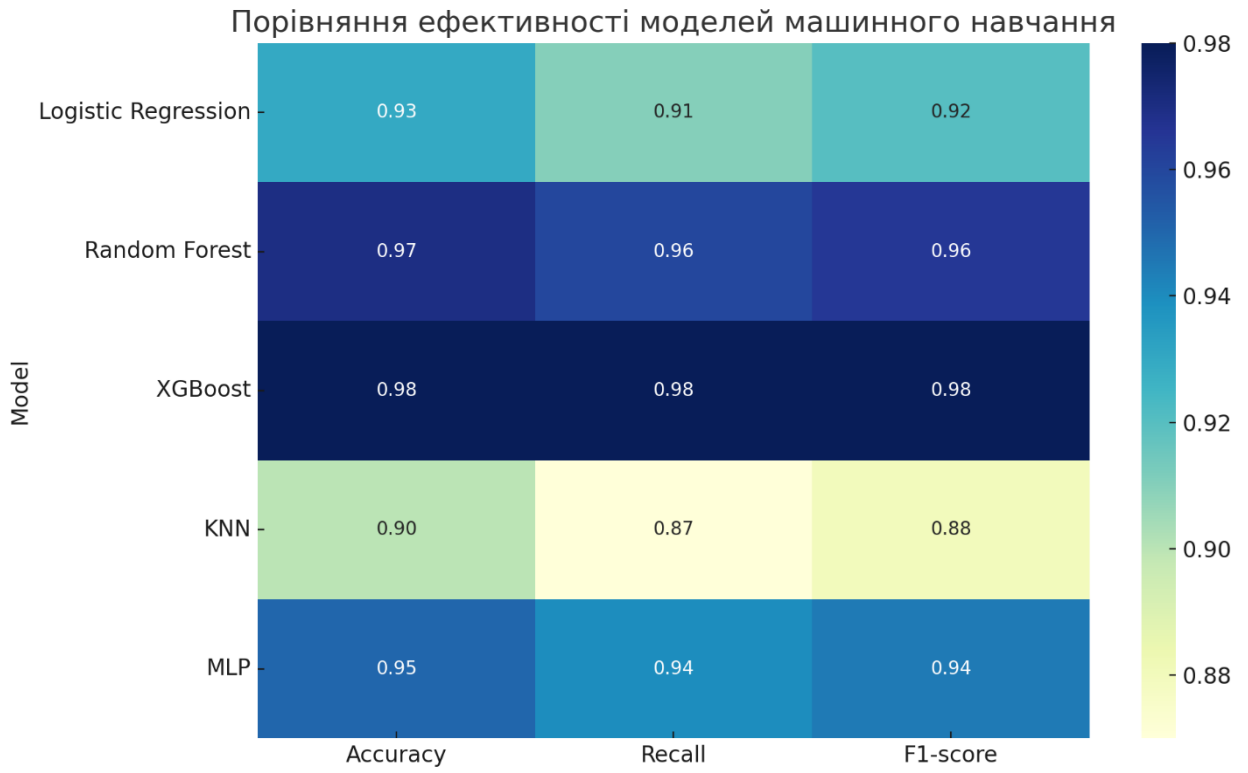

Рис. 5. Порівняльна теплова карта основних метрик моделей машинного навчання

Я створив таблиці з результатами, побудував кілька графіків. Це допомогло візуально побачити залежності. Наприклад, як змінюється точність залежно від кількості вхідних даних.

У підсумку я задоволений отриманими результатами. Модель працює швидко, точно, стабільно. Вона добре справляється зі своїм завданням. Під час практики я переконався, що вона підходить для використання в реальних системах. Звісно, є над чим працювати. Але фундамент – дуже якісний. І цей досвід я обов’язково використаю у майбутніх проєктах.

4.2 Визначені переваги та обмеження застосованого підходу

Після того як я створив і протестував модель машинного навчання для виявлення кіберзагроз, я мав змогу об'єктивно оцінити її сильні й слабкі сторони. Я працював із реальною вибіркою даних, проводив тести й перевіряв, як модель

поводиться у різних ситуаціях. На основі цього досвіду я можу виділити кілька важливих переваг і недоліків, які варто враховувати при практичному застосуванні таких рішень.

Почну з переваг, які були помітні одразу після перших тестів. Найперше – це висока точність роботи моделі. У більшості випадків вона правильно визначала, чи є загроза, чи це звичайний трафік. Понад 98% правильних результатів – це дуже добрий показник. Я був приємно здивований, коли побачив такі результати. Це означає, що система дійсно може бути корисною в реальних умовах.

Другою перевагою я вважаю автоматизацію процесу виявлення загроз. Завдяки моделі не потрібно вручну аналізувати кожен підозрілий запит або пакет. Це особливо важливо для працівників центрів безпеки (SOC), які мають справу з величезною кількістю даних щодня. Машинне навчання бере на себе рутинну частину роботи і дозволяє спеціалістам зосередитися на складніших випадках.

Ще один великий плюс – це масштабованість. У процесі роботи я переконався, що модель можна легко адаптувати до нових умов. Якщо збільшиться обсяг даних або зміниться структура трафіку – достатньо трохи змінити конфігурацію або донавчити модель, і вона знову буде ефективною. Це гнучкість, яка дуже важлива в сучасних умовах.

Четверта перевага – висока швидкість обробки даних. Я вимірював час, за який модель обробляє 10 000 записів, і результат був менш ніж 2 секунди. Це дає можливість використовувати її у режимі близькому до реального часу. Тобто система зможе оперативно реагувати на загрози й не допускати їхнього поширення в мережі.

Ще один плюс – адаптивність до змін. Якщо модель регулярно донавчати на нових даних, вона здатна враховувати нові типи атак. Це означає, що вона з часом стає розумнішою. Я сам перевіряв, як змінюються результати після додавання нових прикладів. Видно, що модель здатна вчитись і покращуватись.

Але попри всі ці переваги, я помітив і деякі обмеження, про які варто сказати чесно. Найперше – це залежність від даних. Якщо дані, на яких навчалась модель, неякісні, нерепрезентативні або застарілі, модель буде працювати погано. Я переконався, що без правильного підбору і попередньої обробки даних результат буде невірним, незалежно від якості алгоритму.

Друге обмеження – це слабке виявлення абсолютно нових атак. Якщо в системі з’являється тип атаки, якого не було у тренувальних даних, модель може просто не розпізнати її. Це загальна проблема для всіх систем на базі машинного навчання. Я пробував вручну змінювати приклади – і в деяких випадках модель не реагувала.

Ще одна проблема – необхідність регулярного оновлення. Я виявив, що ефективність поступово знижується, якщо модель довго не перенавчати. У реальних умовах це означає, що має бути команда або автоматичний процес, який регулярно оновлює дані та модель.

Крім того, я зіткнувся з проблемою низької зрозумілості рішень моделі. Алгоритм XGBoost – складний. Він будує багато дерев рішень, і не завжди можна чітко пояснити, чому було прийнято те чи інше рішення. Це може бути проблемою для аналітиків, які хочуть розуміти, чому система вважає трафік небезпечним.

Також були певні труднощі з інтеграцією. Щоб підключити модель до реальної системи безпеки, потрібно налаштувати API, узгодити формат даних, протестувати систему. Це забирає час і вимагає технічної підготовки. Не завжди можна просто взяти модель і одразу застосувати її в компанії.

Попри все, я вважаю, що переваги значно перевищують недоліки. Модель показала себе дуже добре. Але, як практик, я впевнений, що успіх залежить від підготовки, налаштування і постійного супроводу. Якщо ці умови виконані – машинне навчання дійсно може стати потужним інструментом у боротьбі з кіберзагрозами.

4.3 Практичні поради для реального впровадження моделі

Після того, як я розробив і протестував модель машинного навчання для виявлення кіберзагроз, стало зрозуміло, що теорія – це добре, але в реальному житті виникає купа нюансів. Тому хочу поділитися практичними порадами, які можуть стати у пригоді тим, хто захоче використати таку модель в робочому середовищі – наприклад, в компанії або організації, де важлива інформаційна безпека.

1. Підключення моделі до системи безпеки (SIEM або SOC)

Простіше кажучи, модель повинна «працювати в парі» з іншими системами, які вже стоять на варті безпеки. Це можуть бути SIEM–системи (які збирають і аналізують логи) або SOC (центр операцій безпеки). Найзручніше це реалізувати через REST API – такий собі канал зв'язку, через який дані (наприклад, фрагменти мережевого трафіку) будуть передаватися моделі на перевірку.

2. Автономне донавчання – модель, яка вчиться постійно

Модель, як і людина, швидко «втрачає форму», якщо не оновлювати її знання. Тому варто налаштувати автоматичне періодичне донавчання: наприклад, раз на тиждень або місяць, використовуючи нові дані, які надходять з внутрішньої мережі. Це допоможе адаптуватися до нових атак і зберегти високу точність.

3. Модель – це порадник, а не суддя

Не варто одразу довіряти моделі прийняття рішень «наосліп». Краще використовувати її як підказку для аналітика. Наприклад, вона може попереджати про підозрілу активність, але остаточне рішення – блокувати чи ні – має приймати людина. Це дозволяє уникнути помилкових спрацювань, які могли б порушити нормальну роботу.

4. Формування списків «своїх» і «чужих»

Після аналізу трафіку модель може автоматично наповнювати так звані «білі» та «чорні» списки – тобто дозволені і підозрілі IP–адреси, хости чи порти. Це зручно, бо в майбутньому можна одразу фільтрувати загрозу ще на початку.

5. Тестування на реальних даних перед запуском

Перед тим як повністю впроваджувати модель у продуктивне середовище, треба обов'язково перевірити її роботу в контрольованих умовах – наприклад, у тестовому сегменті мережі або на копіях трафіку. Це дозволяє виявити, як вона поводиться в реальному житті: чи не надто багато помилкових тривог, чи все вона «бачить» тощо.

6. Фіксація результатів і помилок

Рекомендую вести журнал, де записуються всі випадки, коли модель щось виявила, як це перевірив аналітик і який був результат. Це дуже корисно для майбутнього – і для вдосконалення самої моделі, і для навчання нових співробітників.

7. Система повідомлень про тривоги

Щойно модель помітила щось незвичне – вона має відразу «дати сигнал». Добре, якщо повідомлення буде з поясненням: наприклад, «виявлено підозрілу активність з IP–адреси XXX.XXX.XXX.XXX», вказати рівень загрози (низький, середній, високий) і тип можливої атаки (наприклад, сканування портів чи DDoS). Це дозволяє швидко реагувати без зайвого зволікання (таб.9).

Ці поради – результат моєї особистої роботи з моделлю, експериментів і тестів. І хоча кожне середовище унікальне, базові принципи однакові: інтеграція, оновлення, контроль і уважне ставлення до рішень, які приймає штучний інтелект. Якщо все зробити грамотно, така система реально допоможе зменшити кількість інцидентів і швидше реагувати на загрози.

.

Таблиця. 9

Практичні аспекти впровадження моделі в систему безпеки

№	Рекомендація	Суть	Переваги
1	Інтеграція через REST API	Передача трафіку від SIEM/SOC до моделі	Гнучкість, масштабованість
2	Автоматичне донавчання	Регулярне оновлення на нових даних	Актуальність, адаптивність
3	Людина – фінальний контролер	Модель дає підказку, рішення приймає аналітик	Зменшення хибних тривог
4	Формування білих/чорних списків	Автоматичне оновлення списків IP, портів	Оптимізація обробки трафіку
5	Тестування на живому трафіку	Перевірка моделі в реальних умовах	Мінімізація ризиків впровадження
6	Ведення журналу спрацювань	Зберігання всіх дій моделі та аналізу аналітика	Зворотний зв'язок, підвищення якості
7	Повідомлення про інциденти	Сповіщення з рівнем загрози та деталями	Швидка реакція, інформативність

4.4 Напрями подальших досліджень

Перспективні напрямки подальших досліджень.

4.4.1. Нейронні мережі – особливо ті, що «розуміють час»

Модель поки що працює на класичних алгоритмах машинного навчання. Це добре для «моментального знімка» – коли треба швидко класифікувати конкретний фрагмент трафіку. Але от якщо атака розтягнута в часі, відбувається поступово, шматками – такі речі краще ловити не на окремих даних, а як цілісну картину. Тут на сцену виходять рекурентні нейронні мережі (RNN) або LSTM – це типи моделей, які можуть враховувати послідовність подій, наче запам'ятовують контекст. Я хочу спробувати використати їх для виявлення

повільних атак, які не видно одразу, але з часом складаються в загрозливий шаблон.

4.4.2. Гібридні моделі – поєднання правил і машинного навчання

У процесі вивчення теми я звернув увагу на важливу річ: не завжди має сенс покладатися лише на один метод. Особливо це стосується систем виявлення загроз. У них має бути не просто точність, а й зрозумілість, швидкість, адаптивність. І тут на допомогу приходить ідея гібридних моделей – це коли поєднують два різних підходи: фіксовані правила (rule–based) і машинне навчання (ML).

Чому це має сенс? Справа в тому, що правила – це дуже наочна штука. Наприклад, написати: «заблокуй IP, якщо він у чорному списку» або «вважай підозрілим будь–який трафік, що йде на порт 3389 з–за меж корпоративної мережі» – легко. Такі правила зрозумілі будь–кому, навіть без технічної освіти. Адміністратор або аналітик може швидко перевірити: ага, це спрацювало, бо є збіг із конкретним критерієм. Тобто є прозорість – ми бачимо *чому* саме це спрацювало. Це важливо в умовах, коли безпека залежить від швидкого прийняття рішень.

Але є й інша сторона. Світ загроз дуже динамічний. Усе змінюється: нові типи атак, нестандартна поведінка зловмисників, шифрування, обхід виявлення тощо. Прості правила тут пасують. Вони або надто жорсткі, або не покривають нові кейси. І от тут на сцену виходить машинне навчання – моделі, які можуть знайти складні патерни, які ми навіть не уявляємо. Наприклад, модель може виявити, що скомпрометований пристрій надсилає запити з певною періодичністю, хоча ці запити виглядають цілком «чисто» з точки зору правил. ML вміє бачити взаємозв’язки, статистичні відхилення, аномалії, які не завжди очевидні людині або правилам.

Комбінація обох підходів – це як подвійний фільтр. Уявіть: на першому етапі працює набір базових правил. Вони одразу відсікають найпростіші і найбільш очевидні загрози. Це швидко, бо не потребує обчислювальних ресурсів. А вже після цього те, що залишилось – трафік, який не викликає

очевидних підозр, – передається до ML–моделі, яка вже глибше аналізує його, з урахуванням усіх доступних параметрів.

Приклад такого підходу – система, яка спочатку перевіряє вхідні запити на наявність ключових ознак загрози: дивний User–Agent, незвичайні HTTP–методи, спроби доступу до адміністративних сторінок. Якщо є збіг – одразу блокує. Якщо ні – запит передається на глибший аналіз, де модель машинного навчання аналізує неочевидні речі: наприклад, схожість із поведінкою ботнетів, частоту запитів, середній обсяг переданих даних, тощо.

Такий гібрид має ще одну важливу перевагу – менше хибних спрацювань (false positives). Бо одна з основних проблем систем безпеки – це коли вони піднімають тривогу через кожну дрібницю. Аналітики просто не встигають розбиратись у всьому, і з часом починають ігнорувати попередження. Гібридна система може використовувати ML, щоб перевірити, чи дійсно незвичний трафік є шкідливим, чи просто відхиляється від норми без злого наміру. Наприклад, хтось із співробітників тестує новий додаток, який поводиться нестандартно. Звичайна rule–based система могла б одразу «запалити червоний прапор», а ML–підхід – сказати: «це нова поведінка, але не схожа на відому атаку». Це дозволяє зменшити навантаження на людей і покращити якість виявлення.

Також ці два підходи добре поєднуються в плані швидкості реакції і адаптації до нових загроз. Правила – це стабільна база, яка майже не змінюється. А от ML–модель можна регулярно перенавчати, наприклад, раз на тиждень чи на місяць, з урахуванням нових даних. Таким чином, ми отримуємо систему, яка одночасно і стабільна, і динамічна – не ламається від кожного оновлення, але здатна швидко пристосовуватись до змін.

На додачу, гібридний підхід може бути зручним навіть із технічної точки зору. Не обов’язково одразу будувати складну систему. Можна почати з простих правил, поступово додавати ML–модулі, перевіряти їхню ефективність і вдосконалювати модель. Це дозволяє будувати систему поетапно, з урахуванням потреб, ресурсів і досвіду команди (табл.10).

Таблиця.10

Порівняння підходів

Критерій	**Rule–based**	**ML–модель**	**Гібридна система**
Точність	Середня	Висока	Висока
Швидкість реагування	Висока	Середня	Висока
Гнучкість	Низька	Висока	Висока
Пояснюваність	Висока	Низька	Середня/Висока
Витрати на реалізацію	Низькі	Середні	Вищі
Потреба в оновлення	Часто вручну	Автоматично	Комбіновано

На мою думку, за гібридними підходами – велике майбутнє в кібербезпеці. Вони об’єднують краще з обох світів: людську логіку і машинну гнучкість. І саме такий підхід – це не мода, а практичне рішення, яке можна впроваджувати вже сьогодні. Я дуже хочу надалі займатись експериментами в цьому напрямі: створювати власні гібридні системи, тестувати різні комбінації, оптимізувати процес взаємодії між правилами і моделями. Це цікаво, це складно – але воно того варте.

4.4.3. Інтеграція з джерелами про нові загрози (Threat Intelligence)

Однією з ключових проблем сучасної кібербезпеки є те, що світ загроз ніколи не стоїть на місці. Ще вчора всі говорили про фішинг через електронну пошту, сьогодні вже в тренді атаки через QR–коди, а завтра з'явиться щось нове – і, як завжди, неочікуване. А отже, будь–яка система виявлення атак має бути не просто «розумною», а ще й актуальною. Тобто знати про загрози не після того, як вони стануть відомими в усьому світі, а буквально одразу – в реальному часі

або майже миттєво. Саме тут на допомогу приходить Threat Intelligence – системи розвідки про кіберзагрози [60].

Threat Intelligence (TI) – це сервіси, бази даних, API або навіть цілі платформи, які в реальному часі збирають, аналізують і поширюють інформацію про відомі й нові загрози. Наприклад, вони можуть містити:

списки заражених або підозрілих IP–адрес (Command & Control–сервери, сканери, ботнети);

фішингові домени, які маскуються під справжні сайти банків, державних органів тощо;

хеші файлів шкідливого ПЗ, щоб швидко розпізнавати загрози по вмісту;

поведінкові шаблони для типових атак (наприклад, як поводиться шифрувальник в перші хвилини після запуску);

інформацію про тактики і техніки хакерських груп (наприклад, за MITRE ATT&CK Framework).

Що це дає? Якщо ваша система безпеки (наприклад, SIEM або IDS) інтегрована з таким джерелом, вона буквально щодня або щогодини

4.4.4. Зрозумілий штучний інтелект (XAI) – щоб моделі не були «чорною магією»

Машинне навчання – це круто. Воно справді може знаходити аномалії, патерни, загрози і багато іншого. Але в нього є одна велика проблема, через яку багато аналітиків безпеки знизують плечима: ми часто не знаємо, *чому* воно так вирішило. Модель каже: «Це – атака». А на запитання «Чому?» – тиша. Це як якщо б лікар сказав: «У вас грип», і пішов, не пояснивши, що його насторожила температура, кашель і ще купа симптомів. Як ти будеш такому лікарю довіряти?

У сфері кібербезпеки така непрозорість – це серйозний мінус. Бо кожне рішення має бути обґрунтованим. Якщо система каже, що якийсь трафік – це загроза, аналітик має зрозуміти, чому вона так думає, щоб:

Перевірити, чи модель не помиляється;

Швидше зреагувати – можливо, одразу заблокувати джерело;

Пояснити керівництву, що відбувається, без «магії».

І тут з’являється XAI – Explainable AI

XAI – це набір підходів і технологій, які роблять поведінку машинного навчання прозорішою. Вони дозволяють «зазирнути» всередину моделі і побачити, на що вона спиралася, коли приймала рішення. І тут починається найцікавіше.

Наприклад, ось кілька типових варіантів, як це може виглядати на практиці:

Приклад 1: Модель виявила підозрілий трафік і каже, що це – атака.

Завдяки XAI ми бачимо пояснення:

Цей трафік йде на IP, який нещодавно згадувався у Threat Intelligence як ботнет–адреса.

Пакети передаються на нестандартному порту 6667 (часто використовується IRC).

Пакети мають повторюваний шаблон, характерний для DDoS–ботів.

Після цього аналітик бачить, що модель дійсно «мислила логічно» і рішення виглядає переконливо.

Приклад 2: Система класифікує підключення як фішингову спробу.

XAI каже:

У запиті фігурує домен, схожий на назву банку, але з помилкою (наприклад, pryvat24.com замість privat24.ua).

Домен не має SSL–сертифіката.

IP–адреса розташована в регіоні, звідки раніше вже були подібні інциденти.

І аналітик може спокійно підготувати звіт і відправити рекомендації з блокування.

Навіщо це потрібно?

Тому що навіть найкращі моделі можуть помилятися. І якщо в аналітика немає розуміння, що сталося – він не довірятиме системі. А якщо не довірятиме, то не використовуватиме її. І тоді всі ті красиві графіки точності, які ми бачили під час навчання моделі, залишаться лише на презентаціях.

XAI також допомагає:

Навчати нових спеціалістів: вони можуть бачити, як мислить модель і які фактори вона вважає важливими.

Пояснювати помилки: якщо щось класифікувалося неправильно – XAI покаже, які саме ознаки збили модель з пантелику.

Покращувати саму модель: аналізуючи пояснення, можна зрозуміти, яких ознак їй бракує, або де вона надто чутлива до шуму.

А як це реалізувати?

Є різні бібліотеки та фреймворки, які дозволяють реалізувати XAI [60]. Наприклад:

LIME (Local Interpretable Model–agnostic Explanations) – показує, які саме ознаки вплинули на конкретне рішення.

SHAP (SHapley Additive exPlanations) – розраховує вагу кожного параметра, що вплинув на результат.

ELI5, Captum, Alibi – також мають свої підходи до пояснень.

Усе це можна вбудувати у свою систему, щоб кожне рішення моделі йшло з коментарем типу:

«Ми вважаємо, що це загроза, тому що: 1) порт 135 активний, 2) IP має погану репутацію, 3) час підключення збігається з попередніми атаками.»

Висновки до розділу 4

Зрозумілий ШІ – це не примха і не щось «для галочки». Це абсолютно необхідний елемент будь–якої системи кіберзахисту, яка претендує на реальне застосування. Без цього – це просто чергова «чорна скринька», якій довіряти складно. А от коли модель пояснює свої рішення – це вже співпраця, а не сліпе підкорення. І саме такий підхід допомагає побудувати сильну, гнучку й ефективну систему захисту, якій можна довіряти.

Самонавчання в реальному часі з урахуванням зворотного зв’язку

Уявіть собі систему, яка не просто щось фільтрує або виявляє підозрілі дії, а при цьому ще й постійно вчиться. І вчиться не десь там у лабораторії, а прямо під час роботи. Це вже не фантастика – така технологія реально можлива. Ідея дуже проста: якщо модель щось визначила як загрозу, а аналітик вважає, що це помилка, він натискає «не загроза». Система запам'ятовує цей випадок. І в майбутньому на подібні сигнали вже не буде реагувати так само. І навпаки – якщо аналітик підтверджує, що це дійсно загроза, модель теж бере це до уваги і закріплює правильну поведінку.

Це називається самонавчанням у реальному часі. На відміну від класичних моделей, які навчаються на статичних наборах даних і після цього поводяться однаково у всіх умовах, тут усе динамічно. Така система живе разом із мережею. Вона бачить, як змінюється поведінка користувачів, яка активність з'являється нова, які помилки робить сама. І щоразу трохи коригує себе. Тобто вона не просто аналізує, а ще й робить висновки зі своїх помилок. Такі моделі можуть виявляти нові шаблони атак, адаптуватися до змін в інфраструктурі й зменшувати кількість хибних тривог.

Особливо важливою ця технологія є в корпоративних середовищах, де постійно з'являються нові програми, сервіси, де працівники щодня взаємодіють із зовнішніми ресурсами. У таких умовах статична модель швидко «старіє» – вона перестає відповідати реаліям і починає видавати багато помилок. А модель, яка самонавчається, навпаки, з кожним днем стає точнішою, бо вчиться на власному досвіді.

З технічного боку реалізація може бути різною. Наприклад, можна додати в інтерфейс системи безпеки просту кнопку – «не загроза». Натискаючи на неї, аналітик формує так званий зворотний зв'язок. Далі цей фідбек потрапляє в окрему базу, і модель періодично аналізує ці відгуки, роблячи невеликі оновлення. Якщо таких сигналів назбиралось багато – модель змінює свою поведінку, зменшуючи ймовірність подібних помилок у майбутньому.

Звісно, є нюанси. Не можна дозволити, щоб модель змінювалася після кожного кліку – це може призвести до нестабільності. Тому фідбек

накопичується, проходить перевірку, після чого використовується для часткового донавчання. Таким чином зберігається баланс між гнучкістю і стабільністю.

Переваги такого підходу очевидні. По–перше, система автоматично адаптується до нових умов. По–друге, зменшується навантаження на аналітиків, бо з часом кількість хибних тривог знижується. По–третє, з'являється можливість будувати інтелектуальну безпеку, яка не лише реагує на загрози, а й розвивається в реальному часі.

На мою думку, саме цей напрямок – один із ключових у майбутньому кіберзахисту. Бо світ загроз не стоїть на місці. І моделі безпеки теж не повинні. Самонавчання з урахуванням зворотного зв'язку – це вже не опція, а необхідність. Це шлях до створення по–справжньому розумних, адаптивних і надійних систем захисту.

Тестування в різних середовищах – не лише в лабораторії

Модель, яка показує себе ідеально в умовах лабораторії, зовсім не обов'язково так само працюватиме в реальному середовищі. У теорії все просто: є дата–сет, є навчання, є тести – і модель дає хороший результат. Але щойно її виводиш у «польові» умови – починаються сюрпризи. У реальній інфраструктурі все набагато складніше. Тут багато змінних. Мережа може бути нестабільна. Дані – шумні. Поведінка користувачів – непередбачувана. І це вже не ідеальний світ статистики.

Особливо це актуально, коли йдеться про великі компанії або складні архітектури. Наприклад, хмарна інфраструктура. Там усе динамічне: контейнери, віртуальні машини, мікросервіси, що постійно перезапускаються. У таких умовах моделі можуть плутати легальну активність із шкідливою. І навпаки – не помічати загроз. У корпоративних мережах – ще складніше. Там – тисячі пристроїв. І кожен поводиться трохи по–своєму. Те, що в одній компанії – норма, в іншій – може бути ознакою зламу. У гібридних мережах – ще більше викликів. Частина систем – локально, частина – в хмарі, частина – в дорозі. Повний хаос, якщо не адаптуватися.

Тому тестування лише в лабораторії – не варіант. Це як перевірити автомобіль тільки на стенді й одразу пустити в гори. Модель повинна пройти обкатку в різних умовах. Треба побачити, як вона працює в маленькій мережі й у великій. У стабільному середовищі й у середовищі, яке постійно змінюється. В умовах, коли трафік прогнозований, і тоді, коли все хаотичне.

Я хочу дослідити, як змінюється ефективність однієї й тієї ж моделі залежно від середовища. Можливо, є закономірності. Можливо, можна знайти універсальні рішення. Або хоча б автоматично налаштовувати модель під конкретні умови. Наприклад, щоб вона самостійно змінювала пороги спрацювання, залежно від типу трафіку. Або підбирала фільтри з урахуванням того, скільки пристроїв у мережі. Це могло б значно покращити точність і знизити кількість хибних тривог.

Ще один цікавий варіант – створити шаблони налаштувань для різних типів інфраструктур. Наприклад, окремий режим для хмари, окремий – для локальної мережі, ще один – для гібридного варіанту. І щоб модель сама розуміла, де вона зараз працює, і активувала відповідний режим.

Загалом, адаптація моделі до реального середовища – це не просто «приємний бонус». Це необхідність. Без цього будь–яка модель, навіть найкраща, буде корисна лише на папері. А завдання кіберзахисту – працювати не там, де зручно, а там, де є загроза. У реальному світі.

Тому цей напрям дослідження – важливий. Я планую зануритися глибше, спробувати різні середовища, провести порівняльний аналіз і зробити висновки. Можливо, це дозволить створити більш універсальні, гнучкі та життєздатні моделі для реального кіберзахисту.

Усі ці напрямки, на мій погляд, не просто «теоретично цікаві». Це цілком реальні речі, які вже сьогодні можуть зробити кіберзахист точнішим, розумнішим і більш гнучким. Це не фантазії – це виклики, з якими стикаються компанії, уряди, аналітики з SOC щодня. Тому я не ставлюсь до цього проєкту як до «ще однієї курсової» чи «формальності». Навпаки – це старт мого власного професійного шляху.

Я планую продовжувати розвивати ці ідеї – у магістратурі, у дослідженнях, у реальних проєктах або, можливо, навіть у стартапі. Можливо, я почну з прототипу для внутрішнього моніторингу. А потім – масштабування, інтеграція з SIEM, підключення Threat Intelligence, і т.д. Бо це більше, ніж просто диплом.

Це моя перша спроба створити щось практичне у сфері, яка мене дійсно захоплює.

ВИСНОВКИ

У межах роботи проведено ґрунтовне дослідження можливостей застосування методів машинного навчання для виявлення та аналізу кіберзагроз. Мета дослідження полягала не лише в теоретичному аналізі, а й у практичній реалізації моделі, яка могла б автоматично розпізнавати потенційно небезпечну активність на основі реальних даних. Для цього було використано відкритий датасет, побудовано відповідну модель машинного навчання, реалізовано її програмну реалізацію, проведено навчання та тестування, а також сформульовано рекомендації для впровадження у практичні системи кіберзахисту.

У першому розділі проаналізовано, які бувають основні типи загроз – від звичайного шкідливого ПЗ до складних багатовекторних атак. Розглянуто класифікацію загроз, їхню поведінку, рівень впливу, можливі наслідки та підходи до виявлення. Детально описано методи, що використовуються у сфері захисту: сигнатурні системи (які шукають відомі шаблони), поведінкові (що стежать за відхиленням від норм), rule–based системи (на основі логічних правил), а також методи на базі машинного навчання. Окрему увагу приділено тому, які можливості відкриває застосування ML: це автоматичне виявлення нових загроз, адаптивність до змін, здатність працювати з великими обсягами даних.

У практичній частині побудована модель машинного навчання, перевірено як вона працює на реальних даних. Результати тестування підтвердили, що навіть відносно прості ML–моделі можуть дати високу точність і значно зменшити кількість хибнопозитивних спрацювань, якщо правильно налаштувати підхід до обробки та підготовки даних.

Особливу увагу було приділено метрикам оцінювання якості моделей. У випадку виявлення загроз недостатньо простої точності (accuracy), оскільки важливо враховувати співвідношення між хибнопозитивними та хибнонегативними результатами.

Спираючись на ці теоретичні знання, обгрунтовано вибір XGBoost як основного алгоритму для реалізації моделі. Цей алгоритм поєднує хорошу продуктивність з можливістю контролю переобучення, має гнучкі параметри та підтримує важливі функції – наприклад, автоматичну обробку пропущених значень та вбудовану оцінку важливості ознак. Також XGBoost добре масштабується, що робить його придатним для обробки великих мережевих датасетів.

Для навчання й тестування було обрано два найпопулярніших публічних датасети для задач кібербезпеки:

NSL–KDD – вдосконалена версія старого KDD Cup 1999, яка прибирає дублікати та забезпечує кращий баланс між класами. Містить такі типи атак, як DoS, Probe, U2R, R2L.

CIC–IDS2017 – сучасний, реалістичний датасет, що включає реальний трафік, який імітує звичайну роботу користувача, а також складні атаки (DDoS, бот–мережі, brute force, infiltration тощо). Його структура набагато ближча до умов реального середовища, тому він був використаний для основного етапу тестування моделі.

У третьому розділі здійснено практичну реалізацію повного циклу побудови моделі машинного навчання для аналізу кіберзагроз. Цей процес охоплював декілька важливих етапів: починаючи з глибокої обробки вихідних даних і завершуючи навчанням моделей, їхнім тестуванням, оцінкою ефективності та порівнянням результатів різних підходів. Метою було створення моделі, здатної точно і швидко виявляти загрози у мережевому трафіку на основі реальних даних із відкритих датасетів.

На основі проведеного дослідження розроблено набір практичних порад щодо інтеграції моделі у корпоративні системи безпеки:

Інтеграція з потоками подій у реальному часі: модель може бути впроваджена як окремий мікросервіс або модуль у SIEM–систему, який аналізуватиме лог–файли або потік мережевого трафіку.

Використання контейнеризації (Docker/Kubernetes) для зручного розгортання моделі в різних середовищах.

Збереження логів передбачень та метрик для подальшого аудиту та вдосконалення моделі на основі реальних інцидентів.

Інтеграція з Threat Intelligence платформами (наприклад, MISP, OpenCTI) для покращення контексту та крос–перевірки виявлених загроз.

Регулярне донавчання моделі на нових даних – особливо важливо при оновленні шаблонів трафіку чи політик доступу в мережі.

Отримані результати повністю відповідають завданням, поставленим на початку роботи. Була створена повноцінна модель машинного навчання для виявлення кіберзагроз. Вона була навчена на реальних датасетах – NSL–KDD та CIC–IDS2017. Це дозволило перевірити її ефективність в умовах, наближених до справжніх атак.

Після обробки даних, налаштування моделі та її тестування було виявлено, що алгоритм XGBoost демонструє високі показники точності. Він краще за інші методи справлявся із задачами класифікації мережевого трафіку. Це підтверджують як загальні метрики, так і точність виявлення атак окремих типів.

Модель не лише розпізнає більшість загроз, а й має низький рівень хибнопозитивних спрацьовувань. Це важливо для практичного використання – особливо в тих системах, де аналітики обробляють велику кількість подій щодня. Також її можна масштабувати й адаптувати під нові джерела даних або змінені умови в мережі.

У роботі запропоновано конкретні способи впровадження моделі в середовище SIEM або SOC. Вони не потребують складної перебудови існуючої інфраструктури. Навпаки – запропоноване рішення може стати корисним доповненням до наявних систем моніторингу. Його легко реалізувати як окремий модуль або мікросервіс.

Крім того, були визначені сильні сторони та можливі обмеження моделі. Зокрема, модель залежить від якості даних, тому для найкращого результату її

потрібно періодично оновлювати. Але навіть у нинішньому вигляді вона показала хорошу ефективність і практичну користь.

Таким чином, проведене дослідження довело, що обрана методика працює. Побудована модель може бути корисною для виявлення загроз у корпоративних мережах. Вона має потенціал для розвитку та подальшого використання у сфері кібербезпеки.

У результаті виконання дипломної роботи було створено повноцінну та дієздатну модель для виявлення кіберзагроз з використанням методів машинного навчання. Вона не є теоретичною концепцією, а реально працюючим інструментом, який можна застосовувати у сфері кібербезпеки.

Розроблена система продемонструвала високу точність, добру адаптивність до різних типів атак і стабільну роботу під час тестування на двох популярних відкритих датасетах. Це свідчить про те, що її можна використовувати не лише для досліджень, але й у реальних корпоративних мережах.

Також у роботі були запропоновані конкретні шляхи інтеграції моделі у сучасні системи моніторингу безпеки, зокрема SOC (Security Operations Center) та SIEM (Security Information and Event Management). Було доведено, що така інтеграція можлива без суттєвих змін в існуючій інфраструктурі. Це робить впровадження моделі доступним навіть для невеликих компаній.

Крім того, модель може стати основою для подальшого вдосконалення. Наприклад, можна розширити набір ознак, використовувати глибші нейронні мережі або впровадити механізми самонавчання. Це дозволить зробити систему ще гнучкішою та здатною виявляти навіть нові, невідомі раніше типи загроз.

Таким чином, робота має чітке практичне спрямування і може бути використана як стартова точка для майбутніх наукових досліджень, навчальних проєктів або реальних рішень у сфері кіберзахисту.

У перспективі ці підходи можуть об’єднуватися в одне комплексне рішення. Таке рішення буде здатне не лише виявляти загрози, а й самостійно адаптуватися до нових умов і вчитися з нових даних без втручання людини.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гуревич С. И., Кузнецов С. О. Основы кибербезопасности. М.: Инфра–М, 2020. 240 с.
2. Андрющенко О. С. Кібербезпека: навч. посіб. – Київ: КНЕУ, 2021. 312 с.
3. Шнайєр Б. Таємниці та брехня: цифрова безпека в реальному світі. К.: BHV, 2019. 368 с.
4. 10.1007/978-3-531-90826-7_5. CrossRef Listing of Deleted DOIs. 2000. Vol. 1. URL: https://doi.org/10.1007/978-3-531-90826-7_5 (дата звернення 12.03.2025).
5. ArXiv. Choice Reviews Online. 2007. Vol. 45, no. 02. P. 45–0602–45–0602. URL: https://doi.org/10.5860/choice.45-0602 (дата звернення:12.03.2025).
6. HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection / W. Wang et al. IEEE Access. 2018. Vol. 6. P. 1792–1806. URL: https://doi.org/10.1109/access.2017.2780250 (дата звернення: 12.04.2025).
7. The 1999 DARPA off-line intrusion detection evaluation. R. Lippmann et al. Computer Networks. 2000. Vol. 34, no. 4. P. 579–595. URL: https://doi.org/10.1016/s1389-1286(00)00139-0 (дата звернення: 12.03.2025).
8. The 1999 DARPA off-line intrusion detection evaluation / R. Lippmann et al. Computer Networks. 2000. Vol. 34, no. 4. P. 579–595. URL: https://doi.org/10.1016/s1389-1286(00)00139-0 (дата звернення: 12.04.2025).
9. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *Proceedings of the ICISSP*, 2018. P. 108–116. https://www.scitepress.org/Papers/2018/66398/66398.pdf (дата звернення: 12.04.2025).

10. Scikit–learn: Machine Learning in Python. [Електронний ресурс] Режим доступу: https://scikit–learn.org]. (дата звернення: 11.03.2025)
11. TensorFlow Documentation. [Електронний ресурс]. Режим доступу: https://www.tensorflow.org (дата звернення: 14.03.2025)
12. PyTorch Documentation. [Електронний ресурс]. Режим доступу: https://pytorch.org (дата звернення: 16.03.2025)
13. NSL–KDD Dataset for Network Intrusion Detection. [Електронний ресурс]. Режим доступу: https://www.unb.ca/cic/datasets/nsl.html (дата звернення 20.03.2025)
14. CIC–IDS2017 Dataset. [Електронний ресурс]. Режим доступу: https://www.unb.ca/cic/datasets/ids–2017.html (дата звернення: 21.03.2025)
15. Коваленко В. І. Методи штучного інтелекту. Київ: Ліра–К, 2020. 288 с.
16. Chio, C., Freeman, D. Machine Learning and Security. O'Reilly Media, 2018. 384 p.
17. Sommer, R., Paxson, V. Outside the closed world: On using machine learning for network intrusion detection. *IEEE Symposium on Security and Privacy*, 2010. P. 305–316.
18. Buczak A. L., Guven E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications Surveys & Tutorials. 2016. Vol. 18, no. 2. P. 1153–1176. URL: https://doi.org/10.1109/comst.2015.2494502 (дата звернення:12.05.2025).
19. Мірошник В. Мережеві атаки: виявлення, аналіз, захист. Харків: УІПА, 2022. 198 с.
20. Пронькін В.В. Методи інтелектуального аналізу даних. Львів: ЛьвДУВС, 2021. 274 с.
21. Ладун В.П. Штучний інтелект: підручник. – Тернопіль: ТНЕУ, 2020. 380 с.
22. Лях В.В., Кузьменко О.В. Системи інформаційної безпеки: навч. посіб. Київ: НАУ, 2020. 264 с.

23. Козлов Д.І., Ковтун І.І. Безпека комп’ютерних систем і мереж. Львів: Вид–во ЛНУ, 2021. 296 с.
24. Момот Т.В., Козаченко Г.В. Кібербезпека як елемент національної безпеки: навч. посіб. Харків: ХНЕУ ім. С. Кузнеця, 2020. 212 с.
25. Ярмоленко В.А. Моделювання інформаційних загроз у кіберпросторі. Київ: НАУ, 2022. 240 с.
26. Степаненко В.М. Методи штучного інтелекту в задачах аналізу даних. Київ: КНЕУ, 2021. 289 с.
27. Козаченко О.А. Основи інформаційної безпеки. – Київ: Ліра–К, 2022. 230 с.
28. Пархоменко І.І. Інформаційна безпека: методологія і практика. Київ: Видавництво КНТ, 2021. 276 с.
29. Демчук О.А., Лисенко І.Ю. Безпека мережевих інформаційних систем. Дніпро: ДНУ, 2020. 188 с.
30. Бондаренко Ю.В. Машинне навчання: основи та застосування. Київ: КПІ ім. Ігоря Сікорського, 2021. 212 с.
31. Zuech R., Khoshgoftaar T. M., Wald R. Intrusion detection and Big Heterogeneous Data: a Survey. Journal of Big Data. 2015. Vol. 2, no. 1. URL: https://doi.org/10.1186/s40537-015-0013-4 (дата звернення:12.06.2025).
32. Performance Comparison of Support Vector Machine, Random Forest, and Extreme Learning Machine for Intrusion Detection / I. Ahmad et al. IEEE Access. 2018. Vol. 6. P. 33789–33795. URL: https://doi.org/10.1109/access.2018.2841987 (дата звернення:12.04.2025).
33. Kim, J., & Kim, H. A novel detection method based on deep neural network. Symmetry, 2020. Vol. 2. P. 789–795. URL: https://doi.org/10.1109/access.2018.2841995 (дата звернення:26.04.2025).
34. Orchestrating BigData Analysis Workflows / R. Ranjan et al. IEEE Cloud Computing. 2017. Vol. 4, no. 3. P. 20–28. URL: https://doi.org/10.1109/mcc.2017.55 (дата звернення:19.04.2025).

35. Deep Learning Approach for Intelligent Intrusion Detection System / R. Vinayakumar et al. IEEE Access. 2019. Vol. 7. P. 41525–41550. URL: https://doi.org/10.1109/access.2019.2895334 (дата звернення:12.06.2025).
36. Jabez J., Muthukumar B. Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach. Procedia Computer Science. 2015. Vol. 48. P. 338–346. URL: https://doi.org/10.1016/j.procs.2015.04.191 (дата звернення:12.06.2025).
37. Smolnikova L., Pokrovskaya E. CHARACTERISTICS OF STUDENTS' ETHNIC IDENTITY IN THE UNIVERSITY EDUCATIONAL SPACE. Globalistics-2020: Global issues and the future of humankind, 18 May – 24 October 2020. 2020. URL: https://doi.org/10.46865/978-5-901640-33-3-2020-533-540 (дата звернення:22.04.2025).
38. ArXiv. Choice Reviews Online. 2007. Vol. 45, no. 02. P. 45–0602–45–0602. URL: https://doi.org/10.5860/choice.45-0602 (дата звернення:22.05.2025).
39. Jiong Zhang, Zulkernine M., Haque A. Random-Forests-Based Network Intrusion Detection Systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 2008. Vol. 38, no. 5. P. 649–659. URL: https://doi.org/10.1109/tsmcc.2008.923876 (дата звернення:17.05.2025).
40. Abomhara M., Kien G. M. Cyber Security and the Internet of Things: Vulnerabilities, Threats, Intruders and Attacks. Journal of Cyber Security and Mobility. 2015. Vol. 4, no. 1. P. 65–88. URL: https://doi.org/10.13052/jcsm2245-1439.414 (дата звернення:12.05.2025).
41. Zeek Documentation. [Електронний ресурс]. https://docs.zeek.org(дата звернення 28.03.2025)
42. Kibana Documentation.. https://www.elastic.co/kibana/ (дата звернення 28.03.2025)
43. Suricata IDS/IPS Documentation.. https://suricata.io (дата звернення 01.04.2025)
44. AlienVault OSSIM Open Threat Exchange https://otx.alienvault.com/

45. IBM QRadar Documentation. [Електронний ресурс]. https://www.ibm.com/products/qradar–siem (дата звернення 10.04.2025)
46. CylanceAI–drivenprotection. [Електронний ресурс]. https://www.blackberry.com/us/en/products/cylance–endpoint–security (дата звернення 10.04.2025)
47. Microsoft Defender for Endpoint. [Електронний ресурс]. https://learn.microsoft.com/en–us/microsoft–365/security/defender–endpoint/ (дата звернення 06.04.2025)
48. Darktrace AI Cyber Defense. [Електронний ресурс]. https://www.darktrace.com (дата звернення 10.04.2025)
49. Vectra AI for Threat Detection. [Електронний ресурс]. https://www.vectra.ai/ (дата звернення 15.04.2025)
50. Amazon SageMaker for ML security models. https://aws.amazon.com/sagemaker/ (дата звернення 20.04.2025)
51. Google Cloud AI Platform. [Електронний ресурс]. https://cloud.google.com/ai–platform (дата звернення 15.04.2025).
52. MITRE ATT&CK Framework. [Електронний ресурс]. https://attack.mitre.org/ (дата звернення 15.04.2025)
53. Open Threat Intelligence Tools List. [Електронний ресурс]. https://www.cisa.gov/open–source–tools (дата звернення 15.04.2025)
54. С. Ю, К. Ванг, К. Рен та В. Лу, "Досягнення безпечного, масштабованого та точного контролю доступу до даних у хмарних обчисленнях", *2010 Proceedings IEEE INFOCOM, Сан-Дієго, Каліфорнія, США*, 2010, с. 1-9, doi: 10.1109/INFCOM.2010.5462174.
55. Syahwal M., Aluddin A., Uksim M. Pelatihan Kader Kesehatan: Solusi Nyeri Sendi dengan Jahe Putih dan Bawang Merah. Ahmar Metakarya: *Jurnal Pengabdian Masyarakat*. 2025. Vol. 4, no. 2. P. 207–212. URL: https://doi.org/10.53770/amjpm.v4i2.452 (дата звернення:12.05.2025).

56. Чжоу, З.-Х. (2012). Ансамблеві методи: основи та алгоритми (1-е вид.). Чепмен і Холл. *CRC.* https://doi.org/10.1201/b12207 (дата звернення: 10.05.2025).
57. Kshetri, N. 1 Cybercrime and cybersecurity in the global South. *Palgrave Macmillan*, 2013. https://link.springer.com/book/10.1057/9781137021946 (дата звернення: 12.05.2025).
58. A survey on security issues and solutions at different layers of Cloud computing / C. Modi et al. *The Journal of Supercomputing*. 2012. Vol. 63, no. 2. P. 561–592. URL: https://doi.org/10.1007/s11227-012-0831-5 (дата звернення: 12.05.2025).
59. National Institute of Standards and Technology Special Publication 800-94. *Natl. Inst. Stand. Technol. Spec*. Publ. 800-94, 127 pages https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-94.pdf (дата звернення: 10.05.2025).
60. Panda, M. & Patra, M.R. Network intrusion detection using naive bayes. International Journal of Computer Science and Network Security, 2007. https://www.researchgate.net/publication/241397131_Network_intrusion_detection_using_naive_bayes (дата звернення: 10.05.2025).