МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: “СИСТЕМА АНАЛІЗУ ТА КЛАСИФІКАЦІЇ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ МЕТОДІВ МАШИННОГО НАВЧАННЯ”

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.

_____ Іван АФАНАСЬЄВ
(підпис) *Ім’я, ПРІЗВИЩЕ здобувача*

Виконав(ла): здобувач(ка) вищої освіти гр. УБД-41

Іван АФАНАСЬЄВ
Ім’я, ПРІЗВИЩЕ

Керівник: Ірина ЛОЗОВА
к.т.н. Ім’я, ПРІЗВИЩЕ

Рецензент: Сергій ГАХОВ
к.в.н., доцент Ім’я, ПРІЗВИЩЕ

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

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
“____”_____2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Афанасьєву Івану Олександровичу
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи “Система аналізу та класифікації шкідливого програмного забезпечення з використанням методів машинного навчання”,
керівник кваліфікаційної роботи Лозова І.Л., к.т.н.,
(ІПРІЗВИЩЕ, Ім’я,, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від “20“ лютого 2026 р. №51

2. Строк подання кваліфікаційної роботи “12” травня 2026р.

3. Вихідні дані до кваліфікаційної роботи: *шкідливе програмне забезпечення, класифікація шпз, машинне навчання, аналіз даних, кібербезпека, виявлення загроз.*

4. Перелік питань, які мають бути розроблені:
4.1. Проаналізувати існуючі підходи до виявлення ШПЗ та дослідити можливості застосування методів машинного навчання.
4.2. Обгрунтувати вибір алгоритмів для класифікації та розробити структуру системи аналізу ШПЗ.
4.3. Реалізувати систему та провести експериментальну перевірку.

5. Перелік ілюстративного матеріалу: *презентація PowerPoint.*

6. Дата видачі завдання “05“ березня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Етапи кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Визначення об’єкту, предмету, мети та завдань дослідження.	18.03.2026	
2.	Збір та аналіз літератури.	30.03.2026	
3.	Аналіз особливостей аналізу та класифікації шкідливого програмного забезпечення.	08.04.2026	
4.	Дослідження основних методів машинного навчання.	15.04.2026	
5.	Вивчення інструментів та методів аналізу та класифікації шкідливого програмного забезпечення з використанням методів машинного навчання	22.04.2026	
6.	Формулювання висновків за результатами проведеного дослідження.	29.04.2026	
7.	Оформлення роботи.	06.05.2026	
8.	Оформлення презентації.	11.05.2026	
9.	Отримання рецензії на роботу.	10.06.2026	
10.	Захист в ЕК.	__.06.2026	

Здобувач вищої освіти _____ Іван АФАНАСЬЄВ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Ірина ЛОЗОВА
(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 здобувач Афанасьєв І. 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: "Система аналізу та класифікації шкідливого програмного забезпечення з використанням методів машинного навчання"

Кваліфікаційна робота і рецензія додаються.

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

Висновок керівника кваліфікаційної роботи

Здобувач АФАНАСЬЄВ Іван у процесі виконання кваліфікаційної роботи дослідив сучасні підходи до аналізу шкідливого програмного забезпечення та особливості використання методів машинного навчання у сфері кібербезпеки. У роботі проведено аналіз існуючих методів виявлення ШПЗ, виконано обробку датасету PE-файлів та реалізовано систему класифікації шкідливого програмного забезпечення із застосуванням алгоритму Random Forest.

Під час виконання роботи АФАНАСЬЄВ Іван продемонстрував належний рівень теоретичної підготовки, навички роботи з сучасними засобами аналізу даних та вміння застосовувати методи машинного навчання для вирішення практичних задач кібербезпеки. Здобувач проявив відповідальність, самостійність та здатність до опрацювання значного обсягу науково-технічної інформації.

Кваліфікаційна робота виконана відповідно до поставлених завдань, оформлена згідно з установленими вимогами та заслуговує на оцінку "відмінно", а здобувач АФАНАСЬЄВ Іван – на присвоєння кваліфікації бакалавра з кібербезпеки за освітньою програмою Управління інформаційною та кібернетичною безпекою.

Керівник кваліфікаційної роботи _____ Ірина ЛОЗОВА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

"____" _____ 2026 року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач Афанасьєв І. 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 АФАНАСЬЄВА Івана
на тему “Система аналізу та класифікації шкідливого програмного забезпечення з використанням методів машинного навчання”

Актуальність. В сучасних умовах постійного зростання кількості кіберзагроз та розвитку шкідливого програмного забезпечення особливої актуальності набувають питання автоматизації процесів виявлення та класифікації malware. Використання методів машинного навчання дозволяє підвищити ефективність систем інформаційної безпеки та забезпечити швидкий аналіз шкідливого програмного забезпечення. Тому тема кваліфікаційної роботи є актуальною та має практичне значення.

Позитивні сторони.

1. У роботі проведено аналіз сучасних підходів до класифікації шкідливого програмного забезпечення та методів машинного навчання у сфері кібербезпеки.
2. Автором розроблено систему класифікації шкідливого програмного забезпечення із використанням алгоритму Random Forest та реалізовано її засобами мови програмування Python.
3. У роботі проведено експериментальне дослідження ефективності запропонованого підходу та виконано аналіз результатів класифікації.
4. Кваліфікаційна робота оформлена відповідно до встановлених вимог, матеріал викладено послідовно та логічно, результати дослідження подано у вигляді таблиць та рисунків.
5. Автор опрацював значну кількість наукових джерел, у тому числі англомовних публікацій.

Недоліки.

Доцільно було б приділити більше уваги використанню методів глибокого навчання та комбінуванню статичного і динамічного аналізу шкідливого програмного забезпечення.

Однак зазначені зауваження не впливають на загальну позитивну оцінку кваліфікаційної роботи.

Висновок: Кваліфікаційна робота виконана на належному науково-методичному рівні, має практичне значення та відповідає вимогам до бакалаврських кваліфікаційних робіт і заслуговує оцінки “відмінно”, а здобувач АФАНАСЬЄВ Іван заслуговує присвоєння кваліфікації бакалавра з кібербезпеки за освітньою програмою «Управління інформаційною та кібернетичною безпекою».

Рецензент: _____ Сергій ГАХОВ
к.в.н., доцент *підпис* Ім’я, ПРІЗВИЩЕ

РЕФЕРАТ

Кваліфікаційна робота присвячена темі дослідження методів аналізу і класифікації шкідливого програмного забезпечення із застосуванням технологій машинного навчання. Робота має таку структуру: вступ, три глави, в яких є 25 рисунків, висновки і список використаних джерел із 34 найменувань. Обсяг роботи становить 96 аркушів, з яких 6 аркушів, це перелік умовних скорочень і список використаних джерел.

Метою роботи є розробка та дослідження системи аналізу і класифікації шкідливого програмного забезпечення на основі методів машинного навчання.

Об'єктом дослідження є процеси аналізу та виявлення шкідливого програмного забезпечення.

Предмет дослідження **–** методи і моделі машинного навчання для класифікації шкідливого програмного забезпечення, а також підходи до підвищення ефективності їх застосування.

Методи дослідження**.** Для розв'язання поставленого наукового завдання у роботі використані методи аналізу і синтезу, класифікації, порівняльного аналізу, методи машинного навчання, статистичні методи оцінювання якості моделей, а також системний підхід до побудови інформаційних систем.

Як результат, у роботі проаналізовано сучасні підходи до виявлення шкідливого програмного забезпечення, досліджено особливості використання методів машинного навчання у задачах кібербезпеки; розроблено систему класифікації шкідливого програмного забезпечення, що базується на використанні вибраних алгоритмів машинного навчання; проведене експериментальне дослідження ефективності запропонованого підходу та виконане оцінювання якості класифікації.

Галузь застосування**.** Розроблена система може бути використана в антивірусних рішеннях, системах виявлення загроз, а також як компонент у складних системах забезпечення інформаційної безпеки.

Ключові слова: ШКІДЛИВЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ,

КЛАСИФІКАЦІЯ ШПЗ, МАШИННЕ НАВЧАННЯ, АНАЛІЗ ДАНИХ, КІБЕРБЕЗПЕКА, ВИЯВЛЕННЯ ЗАГРОЗ.

ABSTRACT

The qualification work is devoted to the study of methods for analyzing and classifying malicious software using machine learning technologies. The thesis is structured as follows: an introduction, three chapters containing 25 figures, conclusions, and a list of references comprising 34 titles. The thesis comprises 96 pages, of which 6 pages consist of a list of abbreviations and a list of references.

The purpose of the study is to develop and study a system for analyzing and classifying malicious software based on machine learning methods.

The object of the study is the processes of analyzing and detecting malicious software.

The subject of the study is machine learning methods and models for classifying malicious software, as well as approaches to improving the effectiveness of their application.

Research methods. To address the scientific problem posed, this work employs methods of analysis and synthesis, classification, comparative analysis, machine learning methods, statistical methods for evaluating model quality, as well as a systematic approach to building information systems.

As a result, this paper analyzes modern approaches to malware detection, examines the specific features of using machine learning methods in cybersecurity tasks, and develops a malware classification system based on selected machine learning algorithms; and an experimental study of the proposed approach's effectiveness was conducted, along with an evaluation of the classification quality.

Field of application. The developed system can be used in antivirus solutions, threat detection systems, and as a component in complex information security systems.

Keywords: MALWARE, MALWARE CLASSIFICATION, MACHINE LEARNING, DATA ANALYSIS, CYBERSECURITY, THREAT DETECTION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ШПЗ	Malware	Шкідливе програмне забезпечення
ML	Machine Learning	Машинне навчання
PE	Portable Executable	Формат виконуваних файлів Windows
API	Application Programming Interface	Інтерфейс прикладного програмування
SVM	Support Vector Machine	Метод опорних векторів
KNN	K-Nearest Neighbors	Метод k-найближчих сусідів
ANN	Artificial Neural Network	Штучна нейронна мережа
CNN	Convolutional Neural Network	Згорткова нейронна мережа
RNN	Recurrent Neural Network	Рекурентна нейронна мережа
LSTM	Long Short-Term Memory	Мережа довготривалої короткочасної пам’яті
PCA	Principal Component Analysis	Метод головних компонент
URL	Uniform Resource Locator	Уніфікований локатор ресурсу
DLL	Dynamic Link Library	Бібліотека динамічного компонування
RF	Random Forest	Випадковий ліс
AI	Artificial Intelligence	Штучний інтелект
CSV	Comma-Separated Values	Формат табличних даних
GPU	Graphics Processing Unit	Графічний процесор
OS	Operating System	Операційна система
AES	Advanced Encryption Standard	Стандарт симетричного шифрування
TP	True Positive	Істинно позитивний результат
TN	True Negative	Істинно негативний результат
FP	False Positive	Хибнопозитивний результат

FN	False Negative	Хибнонегативний результат
Accuracy	Accuracy	Точність класифікації
Precision	Precision	Прецизійність
Recall	Recall	Повнота
F1-score	F1-score	F1-міра якості класифікації

ВСТУП

Актуальність теми. У сучасних умовах швидкої цифровізації та широкого впровадження інформаційних технологій питання забезпечення кібербезпеки стає особливо актуальним. Кількість та складність шкідливого програмного забезпечення постійно зростають, а традиційні сигнатурні методи виявлення поступово втрачають свій потенціал, що особливо виявляється в боротьбі з новою й модифікованою загрозою.

В цьому випадку величезним допоміжним інструментом можуть стати методи машинного навчання, що дають змогу автоматизувати аналіз програм і знаходити приховані в них закономірності. Завдяки цьому ми можемо створювати системи, що адаптуються до нових загроз без необхідності постійного оновлення сигнатур.

Попри велике наукове зацікавлення цією темою, залишаються актуальними питання підвищення точності класифікації, зменшення кількості хибних спрацьовувань, а також ефективного поєднання різних підходів до аналізу шкідливого програмного забезпечення. Тому розробка системи для аналізу і класифікації ШПЗ із застосуванням технологій машинного навчання стає актуальним завданням сезону.

Мета роботи полягає у розробці та дослідженні системи аналізу і класифікації шкідливого програмного забезпечення з використанням методів машинного навчання.

Об'єкт дослідження – процеси виявлення та аналізу шкідливого програмного забезпечення.

Предмет дослідження – методи та моделі машинного навчання для класифікації шкідливого програмного забезпечення, а також підходи до підвищення їх ефективності.

Для досягнення поставленої цілі потрібно вирішити наступні **завдання**:

1. Проаналізувати існуючі підходи до виявлення ШПЗ та дослідити можливості застосування методів машинного навчання.

2. Обґрунтувати вибір алгоритмів для класифікації та розробити структуру системи аналізу ШПЗ.

3. Реалізувати систему та провести експериментальну перевірку.

Методи дослідження. В роботі використано методи теоретичного аналізу, узагальнення, класифікації, порівняння, методи машинного навчання, статистичні методи оцінки якості моделей, а також системний підхід до проєктування програмних систем.

Наукова новизна. У роботі удосконалено метод автоматизованої класифікації шкідливого програмного забезпечення на основі характеристик PE-файлів за рахунок застосування алгоритму машинного навчання Random Forest у поєднанні з комплексною попередньою обробкою даних (очищення dataset, трансформація категоріальних ознак у числові та відбір інформативних параметрів), що забезпечило підвищення ефективності автоматизованої багатокласової класифікації, автоматизацію повного циклу аналізу даних та покращення інтерпретованості результатів системи.

Практичне значення одержаних результатів. Розроблена система аналізу та класифікації шкідливого програмного забезпечення може бути використана у складі антивірусних рішень, систем виявлення вторгнень та інших засобів інформаційної безпеки. Отримані результати дають змогу підвищити ефективність виявлення шкідливого програмного забезпечення та зменшити кількість помилкових спрацьовувань.

Апробація результатів кваліфікаційної роботи відбулася на Всеукраїнській науково-практичній конференції "Стратегії кіберстійкості: управління ризиками та безперервність бізнесу" 25 лютого 2026 року, де було представлено тези доповіді " Система аналізу та класифікації шкідливого програмного забезпечення з використанням методів машинного навчання" [36].

Розділ 1 АНАЛІЗ МЕТОДІВ ВИЯВЛЕННЯ ТА КЛАСИФІКАЦІЇ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Класифікація шкідливого програмного забезпечення

Шкідливе програмне забезпечення є однією з найбільш небезпечних загроз сучасним інформаційним системам. Активний розвиток цифрових технологій, збільшення кількості користувачів мережі Інтернет та поширення хмарних сервісів сприяють постійному зростанню кількості кібератак і шкідливих програм. Сучасне ШПЗ використовується для викрадення конфіденційної інформації, несанкціонованого доступу до систем, порушення роботи програмного забезпечення та здійснення фінансових атак. У зв'язку з цим особливої актуальності набуває задача класифікації ШПЗ та дослідження його основних типів і характеристик [1].

Шкідливе програмне забезпечення (malware) – це програмний код або програмне забезпечення, створене з метою нанесення шкоди інформаційним системам, мережам або користувачам. Основними цілями ШПЗ є порушення конфіденційності, цілісності та доступності інформації, викрадення даних, отримання віддаленого доступу до системи, а також приховане спостереження за діяльністю користувачів [2].

Одним із найпоширеніших типів шкідливого програмного забезпечення є комп'ютерні віруси. Особливістю вірусів є здатність до самореплікації шляхом зараження інших файлів або програм. Після запуску зараженого файлу вірус активується та може змінювати системні дані, пошкоджувати файли або порушувати роботу операційної системи.

Іншим видом ШПЗ є мережеві черв'яки (worms), які здатні самостійно поширюватися мережею без участі користувача. Для поширення черв'яки використовують вразливості операційних систем або мережевих служб. Основною небезпекою такого типу ШПЗ є швидке зараження великої кількості пристроїв та створення значного навантаження на мережеву інфраструктуру.

Троянські програми (Trojan) маскуються під легітимне програмне забезпечення або корисні файли для введення користувача в оману. Після встановлення троянські програми можуть відкривати віддалений доступ до системи, викрадати конфіденційну інформацію або встановлювати інші шкідливі компоненти. На відміну від вірусів та черв'яків, трояни не мають механізму самостійного поширення [1].

Окрему категорію сучасних кіберзагроз становлять програми-вимагачі (ransomware). Принцип їх роботи полягає у шифруванні файлів користувача або блокуванні доступу до системи з подальшою вимогою викупу за відновлення доступу до даних. За останні роки ransomware став одним із найбільш небезпечних видів шкідливого програмного забезпечення як для окремих користувачів, так і для підприємств та державних установ.

Шпигунське програмне забезпечення (spyware) використовується для прихованого збору інформації про користувача або систему. Таке ПЗ може здійснювати моніторинг активності користувача, збирати паролі, банківські дані та іншу конфіденційну інформацію без відома користувача.

Ще одним небезпечним видом malware є rootkit – набір програмних засобів, призначених для приховування присутності шкідливого програмного забезпечення у системі. Rootkit дозволяє приховувати процеси, файли та мережеві підключення, що значно ускладнює виявлення шкідливого ПЗ традиційними засобами захисту [3].

Для узагальнення основних типів шкідливого програмного забезпечення наведемо їх порівняльну характеристику у вигляді таблиці 1.1.

Таблиця 1.1.

Основні типи шкідливого програмного забезпечення

Тип ШПЗ	**Основне призначення**	**Особливості**
Virus	Зараження файлів	Самореплікація
Worm	Поширення мережею	Автономне функціонування

Продовження таблиці 1.1.

Тип ШПЗ	**Основне призначення**	**Особливості**
Trojan	Несанкціонований доступ	Маскування під легітимне ПЗ
Ransomware	Шифрування файлів	Вимога викупу
Spyware	Прихований збір інформації	Моніторинг активності користувача
Rootkit	Приховування діяльності malware	Маскування процесів і файлів

Сучасне ШПЗ постійно еволюціонує та використовує нові методи обходу систем захисту. Значного поширення набули поліморфне шкідливе програмне забезпечення (polymorphic malware) та метаморфне шкідливе програмне забезпечення (metamorphic malware), які здатні змінювати власний програмний код для уникнення сигнатурного виявлення. Також активно застосовується безфайлове шкідливе програмне забезпечення (fileless malware), що функціонує без запису файлів на диск та використовує системні процеси й оперативну пам’ять. Подібні механізми значно ускладнюють процес виявлення шкідливого програмного забезпечення традиційними методами [2].

Таким чином, шкідливе програмне забезпечення характеризується великою різноманітністю типів, механізмів поширення та методів приховування діяльності. Постійний розвиток malware і поява нових видів кіберзагроз зумовлюють необхідність використання сучасних підходів до аналізу та класифікації шкідливого програмного забезпечення, зокрема із застосуванням методів машинного навчання.

1.2 Основні підходи до аналізу шкідливого програмного забезпечення

Аналіз ШПЗ є одним із ключових етапів забезпечення кібербезпеки та протидії сучасним кіберзагрозам. Основною метою аналізу malware є визначення принципів роботи шкідливого програмного забезпечення, виявлення його функціональних можливостей, механізмів поширення та потенційних загроз для

інформаційних систем. Отримані результати використовуються для створення ефективних засобів виявлення, класифікації та блокування шкідливого ПЗ [1].

Сучасні методи аналізу malware можна поділити на декілька основних категорій: статичний аналіз, динамічний аналіз та гібридний аналіз. Кожен із підходів має власні особливості, переваги та недоліки.

Статичний аналіз передбачає дослідження програмного коду або структури виконуваного файлу без його запуску. Основною перевагою такого підходу є безпечність, оскільки дослідження виконується без безпосереднього виконання шкідливого коду. Статичний аналіз дозволяє визначати структуру PE-файлів, досліджувати імпортовані бібліотеки, API-виклики, рядки, секції файлу та інші характеристики програмного забезпечення.

Одним із найпоширеніших методів статичного аналізу є сигнатурний аналіз, який базується на пошуку відомих шаблонів або сигнатур ШПЗ. Такий підхід активно використовується сучасними антивірусними системами. Однак сигнатурний аналіз має низку недоліків, зокрема низьку ефективність щодо нових або модифікованих загроз. Значну проблему також становлять polymorphic та metamorphic malware, які змінюють власний код для уникнення виявлення [2].

До інструментів статичного аналізу належать дизасемблери, декомпілятори та спеціалізовані засоби аналізу виконуваних файлів, такі як IDA Pro, Ghidra, PE Explorer та інші.

Динамічний аналіз передбачає запуск шкідливого програмного забезпечення у контрольованому середовищі з подальшим спостереженням за його поведінкою. Для цього використовуються віртуальні машини, sandbox-середовища та ізольовані тестові системи.

Під час динамічного аналізу досліджуються:

- поведінка програми;
- мережеві підключення;
- зміни у файловій системі;
- модифікації реєстру;

- створення нових процесів;
- взаємодія з операційною системою.

Основною перевагою динамічного аналізу є можливість виявлення реальної поведінки malware та аналізу навіть обфускованого або зашифрованого шкідливого коду. Разом із тим такий підхід потребує значних обчислювальних ресурсів та спеціалізованого середовища виконання. Крім того, деякі сучасні шкідливі програми здатні визначати запуск у sandbox-середовищі та приховувати власну активність [1].

Для реалізації динамічного аналізу широко використовуються такі системи, як Cuckoo Sandbox, Any.Run та Hybrid Analysis.

Гібридний аналіз поєднує методи статичного та динамічного аналізу з метою підвищення ефективності виявлення malware. У межах такого підходу спочатку виконується дослідження структури файлу та його ознак, після чого здійснюється аналіз поведінки програми під час виконання.

Основною перевагою гібридного аналізу є можливість отримання більшої кількості характеристик шкідливого програмного забезпечення, що позитивно впливає на точність класифікації та виявлення загроз. Саме гібридний підхід часто використовується у сучасних системах аналізу malware на основі машинного навчання [3].

Для порівняння основних підходів до аналізу шкідливого програмного забезпечення слід навести їх характеристику у вигляді таблиці 1.2.

Таблиця 1.2.

Порівняння основних підходів до аналізу ШПЗ

Підхід	**Основна особливість**	**Переваги**	**Недоліки**
Статичний аналіз	Аналіз без виконання файлу	Висока швидкість, безпечність	Низька ефективність щодо obfuscated malware

Продовження таблиці 1.2.

Підхід	**Основна особливість**	**Переваги**	**Недоліки**
Динамічний аналіз	Аналіз поведінки під час виконання	Виявлення реальної активності malware	Високі вимоги до ресурсів
Гібридний аналіз	Комбінування двох підходів	Вища точність аналізу	Складність реалізації

Останніми роками особливої актуальності набуває застосування методів машинного навчання у процесі аналізу malware. Використання алгоритмів класифікації дозволяє автоматизувати процес виявлення ШПЗ, аналізувати великі обсяги даних та виявляти нові типи загроз без необхідності створення сигнатур вручну. Методи машинного навчання можуть застосовуватися як у статичному, так і у динамічному аналізі [2].

Таким чином, сучасні підходи до аналізу шкідливого програмного забезпечення мають різні принципи функціонування та сфери застосування. Статичний аналіз забезпечує високу швидкість дослідження, динамічний дозволяє виявляти поведінкові характеристики malware, а гібридний аналіз поєднує переваги обох підходів. Використання методів машинного навчання у поєднанні з сучасними методами аналізу malware є перспективним напрямом розвитку систем кіберзахисту.

1.3 Ознаки для виявлення шкідливого програмного забезпечення

Одним із ключових етапів аналізу та класифікації ШПЗ є визначення ознак, які дозволяють виявляти потенційно небезпечні файли та відрізняти їх від легітимного програмного забезпечення. У системах аналізу malware ознаки використовуються як вхідні дані для алгоритмів класифікації та моделей машинного навчання. Від якості та інформативності обраних ознак значною мірою залежить ефективність виявлення шкідливого ПЗ [2].

Ознаки ШПЗ можна поділити на декілька основних категорій: статичні, динамічні та поведінкові. Кожен тип ознак характеризує певні особливості функціонування malware та використовується у відповідних методах аналізу.

Статичні ознаки отримуються без безпосереднього запуску програмного файлу. Аналіз таких характеристик є одним із найпоширеніших підходів у системах автоматизованого виявлення malware. Основною перевагою статичного аналізу є висока швидкість дослідження та відсутність необхідності виконання потенційно небезпечного коду [1].

До основних статичних ознак належать:

- структура PE-файлу;
- розмір файлу;
- кількість секцій;
- ентропія файлу;
- імпортовані DLL-бібліотеки;
- API-виклики;
- рядки (strings);
- сигнатури та хеш-значення.

Особливу роль у статичному аналізі відіграє дослідження Portable Executable (PE) структури виконуваних файлів операційної системи Windows. Аналіз заголовків PE-файлу, таблиць імпорту та секцій дозволяє виявити аномальні характеристики, які часто притаманні шкідливому програмному забезпеченню.

Ентропія файлу також є важливою ознакою для виявлення malware. Високі значення ентропії можуть свідчити про використання пакувальників, шифрування або обфускації коду, що часто застосовується для приховування шкідливої активності [3].

Описавши статистичні ознаки, продемонструємо приклад структури PE-файлу на рис. 1.1 [26].

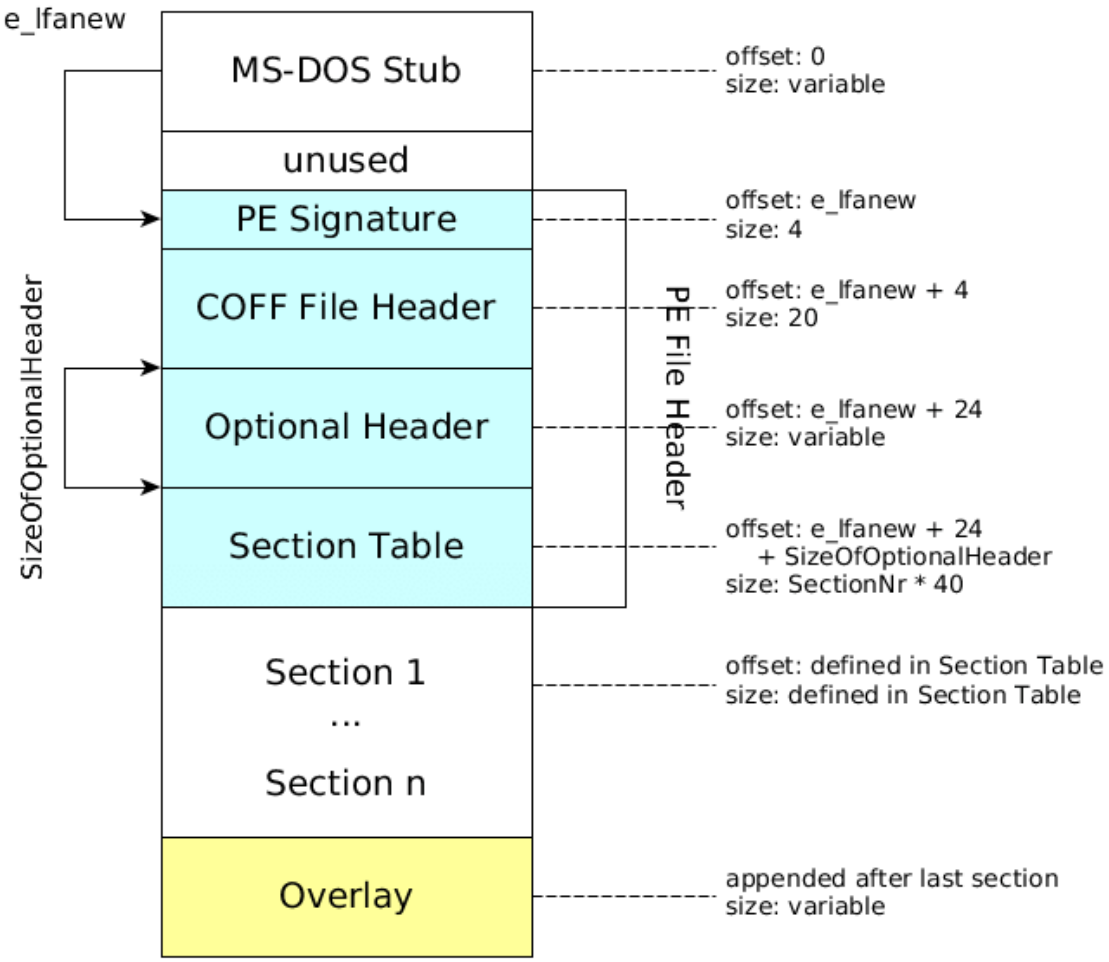

Рис. 1.1. Структура PE-файлу Windows.

Динамічні ознаки формуються під час виконання програмного забезпечення у контрольованому середовищі. На відміну від статичного аналізу, динамічний підхід дозволяє досліджувати реальну поведінку ШПЗ та виявляти приховану активність.

До основних динамічних ознак належать:

- мережеві підключення;
- створення нових процесів;
- зміни файлової системи;
- модифікація системного реєстру;
- використання системних ресурсів;
- взаємодія з операційною системою.

Під час виконання malware може здійснювати підключення до командних серверів (Command and Control), створювати приховані процеси або змінювати системні параметри. Аналіз подібних дій дозволяє визначити характер поведінки шкідливого програмного забезпечення та підвищити точність його класифікації [1].

Окрему групу становлять *поведінкові ознаки*, які описують характер взаємодії malware із системою та користувачем. Поведінковий аналіз широко використовується у сучасних системах кіберзахисту, оскільки дозволяє виявляти навіть нові або модифіковані види шкідливого програмного забезпечення.

До поведінкових характеристик належать:

- спроби приховування активності;
- несанкціоноване отримання доступу;
- перехоплення введення з клавіатури;
- масове шифрування файлів;
- підозріла мережева активність;
- спроби обходу механізмів захисту.

Поведінкові ознаки є особливо важливими для виявлення сучасного polymorphic та fileless malware, яке може змінювати власний код або працювати без запису файлів на диск [3].

Узагальнимо основні ознаки ШПЗ та наведемо їх порівняльну характеристику у вигляді таблиці 1.3.

Таблиця 1.3.

Основні ознаки для виявлення шкідливого програмного забезпечення

Тип ознак	**Приклади ознак**	**Особливості**
Статичні	PE-структура, API-виклики, ентропія	Аналіз без запуску файлу
Динамічні	Мережеві з’єднання, процеси	Аналіз під час виконання
Поведінкові	Шифрування файлів, keylogging	Аналіз дій malware

Однією з основних проблем сучасних систем виявлення malware є значна кількість ознак та необхідність їх ефективного відбору. Надлишкова кількість характеристик може призводити до зниження продуктивності моделей машинного навчання та збільшення часу аналізу. У зв’язку з цим важливим етапом побудови систем класифікації шкідливого програмного забезпечення є процес відбору найбільш інформативних ознак [2].

Таким чином, ознаки шкідливого програмного забезпечення є основою сучасних систем аналізу та класифікації malware. Використання статичних, динамічних та поведінкових характеристик дозволяє формувати ефективні моделі виявлення загроз і підвищувати точність роботи систем кіберзахисту.

1.4. Методи машинного навчання у задачах кібербезпеки

Сучасні інформаційні системи щоденно піддаються значній кількості кіберзагроз, що зумовлює необхідність використання ефективних засобів автоматизованого виявлення та аналізу атак. Традиційні методи захисту, засновані на сигнатурному аналізі, не завжди здатні виявляти нові або модифіковані загрози. У зв’язку з цим особливої актуальності набуває використання методів машинного навчання у задачах кібербезпеки [4].

Машинне навчання (Machine Learning, ML) – це напрям штучного інтелекту, який передбачає побудову математичних моделей, здатних навчатися на основі даних та приймати рішення без жорстко заданих правил. У сфері кібербезпеки методи машинного навчання використовуються для [5]:

- виявлення шкідливого програмного забезпечення;
- аналізу мережевого трафіку;
- виявлення аномальної активності;
- класифікації загроз;
- виявлення вторгнень;
- аналізу поведінки користувачів.

Основною перевагою машинного навчання є здатність виявляти нові або раніше невідомі загрози на основі аналізу ознак та закономірностей у даних. На

відміну від сигнатурних методів, ML-моделі можуть адаптуватися до нових типів атак та автоматично вдосконалювати процес виявлення кіберзагроз.

Найбільш поширеними підходами машинного навчання у задачах кібербезпеки є:

- навчання з учителем (supervised learning);
- навчання без учителя (unsupervised learning);
- напівконтрольоване навчання (semi-supervised learning);
- глибоке навчання (deep learning).

Навчання з учителем є одним із найпоширеніших підходів у системах класифікації шкідливого програмного забезпечення. У межах такого підходу система навчається на попередньо розмічених даних, де кожен зразок належить до певного класу, наприклад «malware» або «benign software».

До найбільш популярних алгоритмів навчання з учителем належать:

- Decision Tree;
- Random Forest;
- Support Vector Machine (SVM);
- K-Nearest Neighbors (KNN);
- Logistic Regression;
- Naive Bayes.

Алгоритм Random Forest широко використовується у задачах виявлення malware завдяки високій точності класифікації та здатності працювати з великою кількістю ознак. Support Vector Machine ефективно застосовується для задач бінарної класифікації та аналізу високовимірних даних [6].

Методи *навчання без учителя* використовуються у випадках, коли дані не мають попереднього маркування. Основною задачею таких алгоритмів є виявлення прихованих закономірностей, аномалій або групування об’єктів за схожими характеристиками.

У сфері кібербезпеки unsupervised learning часто використовується для:

- виявлення аномальної мережевої активності;

- аналізу поведінки користувачів;
- виявлення нових типів атак;
- кластеризації шкідливого програмного забезпечення.

До основних алгоритмів належать:

- K-Means;
- DBSCAN;
- Hierarchical Clustering.

Останніми роками значного поширення у задачах кібербезпеки набули методи *глибокого навчання* (Deep Learning). Використання нейронних мереж дозволяє автоматично виділяти складні залежності у великих обсягах даних та підвищувати точність виявлення кіберзагроз [7].

У задачах аналізу malware найчастіше використовуються:

- Artificial Neural Networks (ANN);
- Convolutional Neural Networks (CNN);
- Recurrent Neural Networks (RNN);
- Long Short-Term Memory (LSTM).

Глибокі нейронні мережі можуть використовуватися для аналізу API-викликів, мережевого трафіку, поведінкових характеристик та бінарного коду шкідливого програмного забезпечення [8].

Для узагальнення основних методів машинного навчання доцільно навести їх порівняльну характеристику у вигляді табл. 1.4.

Таблиця 1.4.

Основні методи машинного навчання у задачах кібербезпеки

Метод	**Основне призначення**	**Переваги**	**Недоліки**
Random Forest	Класифікація malware	Висока точність	Велика кількість дерев
SVM	Бінарна класифікація	Робота з високовимірними даними	Висока складність
K-Means	Кластеризація	Простота реалізації	Необхідність вибору k
CNN	Аналіз складних залежностей	Висока ефективність	Значні обчислювальні ресурси

Таким чином, методи машинного навчання є перспективним напрямом розвитку сучасних систем кібербезпеки. Використання ML-алгоритмів дозволяє автоматизувати процес виявлення та класифікації шкідливого програмного забезпечення, підвищувати точність аналізу та ефективно протидіяти сучасним кіберзагрозам.

1.5. Огляд існуючих рішень

Сучасні системи кібербезпеки використовують велику кількість програмних рішень для виявлення, аналізу та класифікації ШПЗ. Існуючі системи відрізняються принципами роботи, методами аналізу та рівнем автоматизації процесу виявлення кіберзагроз. Значна частина сучасних рішень поєднує сигнатурний аналіз, поведінковий аналіз та методи машинного навчання для підвищення точності виявлення malware [4].

Одним із найвідоміших сервісів аналізу шкідливого програмного забезпечення є VirusTotal. Дана платформа дозволяє перевіряти файли та URL-адреси за допомогою великої кількості антивірусних рушіїв і механізмів аналізу. VirusTotal підтримує сигнатурний аналіз, хешування файлів та поведінковий аналіз підозрілих об’єктів. Основною перевагою сервісу є можливість швидкого отримання результатів перевірки та використання великої бази сигнатур (Рис. 1.2) [9].

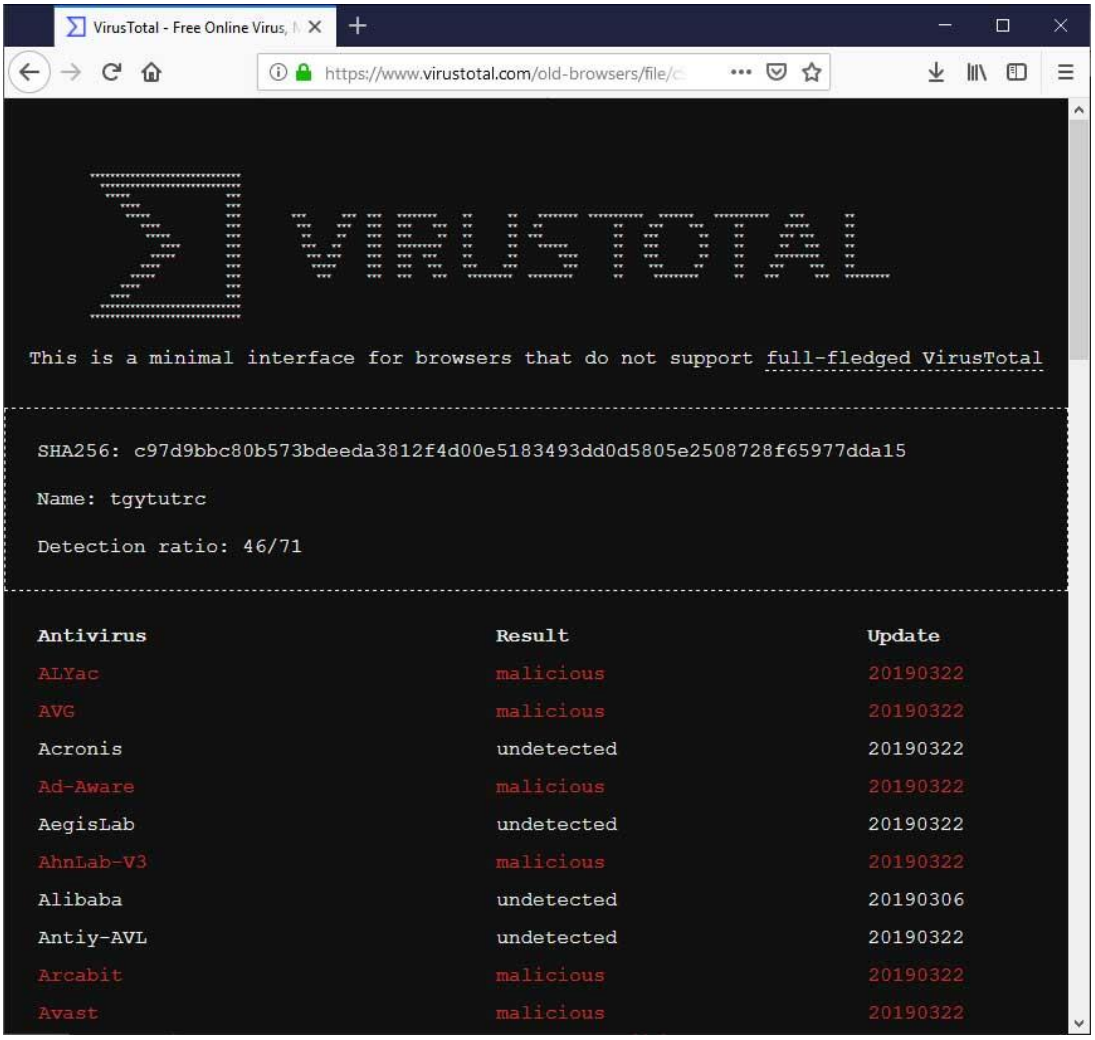

Рис. 1.2. Приклад аналізу файлу у системі VirusTotal.

Для динамічного аналізу ШПЗ широко використовуються sandbox-системи. Одним із найпоширеніших рішень є Cuckoo Sandbox – система автоматизованого аналізу malware з відкритим вихідним кодом. Платформа дозволяє запускати підозрілі файли у віртуальному середовищі та аналізувати їх

поведінку, мережеву активність, взаємодію з файловою системою та реєстром операційної системи [10].

Серед сучасних комерційних платформ динамічного аналізу значного поширення набув сервіс ANY.RUN. Особливістю системи є інтерактивне sandbox-середовище, яке дозволяє виконувати детальний аналіз поведінки шкідливого програмного забезпечення у режимі реального часу. Платформа підтримує аналіз мережевих підключень, процесів, змін у системі та взаємодію користувача із середовищем виконання [11].

Ще одним популярним рішенням є платформа Hybrid Analysis, яка поєднує статичний та динамічний аналіз malware. Система дозволяє формувати детальні звіти щодо поведінки програмного забезпечення, виявляти підозрілі API-виклики та аналізувати мережеву активність [12].

Для дослідження систем класифікації malware значну роль відіграють набори даних (datasets), що містять зразки шкідливого та легітимного програмного забезпечення. Одним із найвідоміших наборів даних є Microsoft Malware Classification Challenge Dataset, який активно використовується у дослідженнях із застосуванням машинного навчання для виявлення шкідливого ПЗ [13].

Виконаємо порівняння існуючих систем аналізу ШПЗ у вигляді табл. 1.5.

Таблиця 1.5.

Порівняння існуючих систем аналізу шкідливого програмного забезпечення

Система	**Тип аналізу**	**Основні можливості**
VirusTotal	Статичний/гібридний	Перевірка файла багатьма AV-рушіями
Cuckoo Sandbox	Динамічний	Аналіз поведінки malware
ANY.RUN	Динамічний	Інтерактивний sandbox
Hybrid Analysis	Гібридний	Автоматизований аналіз malware

В українських наукових роботах також досліджуються сучасні методи аналізу ШПЗ та підходи до побудови систем кіберзахисту. Основна увага приділяється використанню поведінкового аналізу, методів виявлення аномалій та застосуванню алгоритмів машинного навчання для автоматизації процесу виявлення загроз [14], [15].

Незважаючи на значну кількість існуючих рішень, сучасні системи аналізу malware мають певні обмеження. Частина систем характеризується високими вимогами до обчислювальних ресурсів, інші залежать від сигнатурного аналізу та не завжди здатні ефективно виявляти нові типи шкідливого програмного забезпечення. Це зумовлює необхідність подальшого розвитку систем класифікації malware із використанням методів машинного навчання та комбінованого аналізу ознак [5].

1.6. Аналіз недоліків існуючих підходів

Незважаючи на активний розвиток сучасних систем кіберзахисту та засобів аналізу шкідливого програмного забезпечення, існуючі підходи до виявлення та класифікації malware мають низку суттєвих недоліків. Постійне ускладнення механізмів приховування шкідливого коду, використання обфускації та поява нових типів кіберзагроз значно ускладнюють процес аналізу malware і знижують ефективність традиційних методів захисту [3].

Одним із найбільш поширених підходів до виявлення ШПЗ є сигнатурний аналіз. Його принцип роботи базується на порівнянні файлів із базою відомих сигнатур malware. Основною перевагою такого підходу є висока швидкість роботи та простота реалізації. Проте сигнатурний аналіз має суттєвий недолік – неможливість ефективного виявлення нових або модифікованих загроз. У випадку появи нового виду malware система не здатна виявити його без попереднього оновлення бази сигнатур [1].

Додаткову проблему становить використання сучасними шкідливими програмами методів обфускації коду, шифрування та пакування виконуваних файлів. Поліморфне (polymorphic malware) та метаморфне шкідливе програмне

забезпечення (metamorphic malware) здатне змінювати власну структуру під час поширення або виконання, що значно ускладнює застосування сигнатурних методів виявлення [2].

Статичний аналіз malware також має низку обмежень. Незважаючи на високу швидкість дослідження та відсутність необхідності запуску потенційно небезпечного коду, статичний аналіз не завжди дозволяє визначити реальну поведінку програмного забезпечення. У багатьох випадках шкідливі програми використовують механізми приховування функціональності або активують шкідливі дії лише за певних умов. Це знижує ефективність виключно статичного підходу [4].

Динамічний аналіз дозволяє досліджувати поведінку шкідливого програмного забезпечення під час виконання, однак також характеризується певними недоліками. Реалізація динамічного аналізу потребує значних обчислювальних ресурсів, використання віртуальних середовищ та спеціалізованих sandbox-систем. Крім того, деякі сучасні шкідливі програми здатні виявляти запуск у віртуальному середовищі та змінювати власну поведінку для уникнення виявлення [10].

Ще однією проблемою існуючих систем аналізу malware є велика кількість ознак, які використовуються для класифікації програмного забезпечення. Надлишкові або нерелевантні ознаки можуть негативно впливати на точність моделей машинного навчання та збільшувати час аналізу даних. У зв’язку з цим важливого значення набувають методи відбору ознак та оптимізації процесу класифікації [8].

У сучасних системах кібербезпеки активно застосовуються методи машинного навчання, проте ML-моделі також мають певні обмеження. Значна частина алгоритмів потребує великих обсягів навчальних даних та значних обчислювальних ресурсів для навчання моделей. Крім того, якість роботи моделей машинного навчання суттєво залежить від [5]:

- якості набору даних;
- правильності маркування;

- балансу класів;
- обраних ознак;
- параметрів алгоритму.

Проблемою також є можливість перенавчання моделей (overfitting), коли вона демонструє високу точність на навчальних даних, але працює менш ефективно на нових зразках. У задачах кібербезпеки це може призводити до збільшення кількості хибнопозитивних або хибнонегативних результатів класифікації [7].

Для узагальнення основних недоліків існуючих підходів доцільно навести порівняльну характеристику у вигляді табл. 1.6. [1], [8], [10].

Таблиця 1.6.

Недоліки існуючих підходів до аналізу шкідливого програмного забезпечення

Підхід	**Основні недоліки**
Сигнатурний аналіз	Неможливість виявлення нових загроз
Статичний аналіз	Низька ефективність щодо obfuscated malware
Динамічний аналіз	Значні вимоги до ресурсів
ML-моделі	Потреба у великому наборі даних
Deep Learning	Висока складність навчання

Таким чином, існуючі підходи до аналізу та виявлення ШПЗ мають низку обмежень, пов’язаних із розвитком сучасних кіберзагроз, використанням механізмів приховування коду та складністю аналізу великих обсягів даних. Це зумовлює необхідність розробки більш ефективних систем класифікації malware із використанням методів машинного навчання, комбінованого аналізу ознак та сучасних підходів до автоматизації кіберзахисту.

1.7. Постановка задачі

Проведений аналіз сучасних підходів до виявлення та класифікації шкідливого програмного забезпечення показав, що традиційні сигнатурні методи мають обмежену ефективність при виявленні нових, модифікованих та поліморфних типів ШПЗ. У зв’язку з цим актуальним є використання методів машинного навчання, які дозволяють автоматизувати процес аналізу програмного забезпечення та підвищити ефективність виявлення кіберзагроз [5], [8].

Разом із тим існуючі системи аналізу malware часто характеризуються [6], [16]:

- високою обчислювальною складністю;
- значною залежністю від набору ознак;
- потребою у великих обсягах даних;
- складністю інтеграції різних методів аналізу;
- недостатньою точністю при роботі з новими типами загроз.

В межах даної роботи основна увага приділяється розробці методу та програмної системи класифікації ШПЗ на основі методів ML та статичного аналізу PE-файлів.

Для досягнення поставленої мети у другому та третьому розділах роботи необхідно вирішити такі задачі [19], [20]:

1. Формалізувати задачу класифікації шкідливого програмного забезпечення;
2. Визначити методи обробки та відбору ознак;
3. Обґрунтувати вибір алгоритмів машинного навчання;
4. Розробити підхід до класифікації malware на основі статичного аналізу;
5. Спроєктувати архітектуру системи;
6. Розробити алгоритм роботи системи;
7. Обґрунтувати вибір технологій реалізації;
8. Реалізувати програмну систему аналізу та класифікації malware;

9. Провести експериментальне дослідження ефективності розробленого підходу;
10. Виконати аналіз результатів класифікації та оцінити ефективність системи.

В межах роботи передбачається використання методів статичного аналізу виконуваних PE-файлів, процедур preprocessing даних та алгоритмів машинного навчання для автоматизованої класифікації програмного забезпечення.

Основною задачею є побудова системи класифікації:

$F(X) \rightarrow Y$, де:

- X – вектор ознак PE-файлу;
- Y – результат класифікації програмного забезпечення.

У результаті виконання роботи планується реалізувати програмну систему, здатну автоматично аналізувати PE-файли, виділяти інформативні ознаки та визначати належність програмного забезпечення до класу malware або benign software із використанням алгоритмів машинного навчання [16], [22].

Таким чином, постановка задачі визначає основні напрями подальшого дослідження, які пов’язані з розробкою методу класифікації ШПЗ, реалізацією програмної системи та проведенням експериментального оцінювання ефективності запропонованого підходу.

Висновки до розділу 1

В першому розділі кваліфікаційної роботи було проведено дослідження сучасних підходів до аналізу та класифікації ШПЗ. Розглянуто основні типи malware, їх особливості, механізми функціонування та поширення. Встановлено, що сучасне шкідливе програмне забезпечення характеризується високим рівнем складності, використанням механізмів обфускації, шифрування та приховування активності, що значно ускладнює процес його виявлення.

У ході дослідження проаналізовано основні підходи до аналізу ШПЗ, зокрема статичний, динамічний та гібридний аналіз. Визначено їх основні переваги та недоліки. Встановлено, що статичний аналіз забезпечує високу

швидкість дослідження програмного забезпечення, проте має обмежену ефективність щодо сучасних модифікованих загроз. Динамічний аналіз дозволяє досліджувати поведінку malware під час виконання, однак потребує значних обчислювальних ресурсів і спеціалізованого середовища. Гібридний підхід поєднує переваги обох методів та дозволяє підвищити ефективність аналізу.

Також у роботі досліджено основні ознаки, які використовуються для виявлення ШПЗ. Розглянуто статичні, динамічні та поведінкові характеристики malware, що можуть застосовуватись у системах автоматизованого аналізу та класифікації. Встановлено, що ефективність систем виявлення значною мірою залежить від правильного вибору та обробки ознак програмного забезпечення.

Окрему увагу приділено застосуванню методів машинного навчання у задачах кібербезпеки. Розглянуто основні алгоритми класифікації, кластеризації та глибокого навчання, які використовуються у сучасних системах аналізу malware. Визначено, що використання ML-методів дозволяє автоматизувати процес виявлення шкідливого програмного забезпечення та підвищити точність класифікації нових загроз.

У межах розділу також виконано огляд існуючих систем аналізу malware, серед яких VirusTotal, Cuckoo Sandbox, ANY.RUN та Hybrid Analysis. Проведений аналіз показав, що сучасні рішення мають певні обмеження, пов’язані із залежністю від сигнатурних баз, високими вимогами до обчислювальних ресурсів та складністю виявлення нових типів шкідливого програмного забезпечення.

На основі проведеного дослідження обґрунтовано актуальність розробки системи аналізу та класифікації malware із використанням методів ML. Сформульовано постановку задачі та визначено основні етапи реалізації системи, що будуть розглянуті у наступних розділах роботи.

Розділ 2 РОЗРОБКА МЕТОДУ ТА ПРОЄКТУВАННЯ СИСТЕМИ КЛАСИФІКАЦІЇ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Формалізація задачі класифікації

Одним із ключових етапів розробки системи аналізу та класифікації шкідливого програмного забезпечення є формалізація задачі класифікації. Правильне визначення структури вхідних даних, класів об’єктів, ознак та критеріїв оцінювання дозволяє побудувати ефективну модель машинного навчання для автоматизованого виявлення malware.

У загальному вигляді задача класифікації полягає у віднесенні програмного забезпечення до одного з наперед визначених класів на основі аналізу певного набору ознак. У межах даної роботи розглядається задача бінарної класифікації, де програмний об’єкт може належати до одного з двох класів [4]:

- benign software – легітимне програмне забезпечення;
- malware.

Формально набір досліджуваних об’єктів можна представити у вигляді множини: $X=\{x_1,x_2,x_3,\ldots,x_n\}$, де:

- x_i – окремий програмний файл;
- n – кількість досліджуваних об’єктів.

Кожен об’єкт характеризується набором ознак: $x_i=(f_1,f_2,f_3,\ldots,f_m)$, де:

- f_j – окрема ознака програмного забезпечення;
- m – кількість ознак.

У системах аналізу malware як ознаки можуть використовуватись [2]:

- характеристики PE-файлу;
- API-виклики;
- ентропія секцій;
- розмір файлу;
- мережеві характеристики;
- поведінкові параметри;

- статистичні характеристики коду.

Задача класифікації полягає у побудові функції:

F(X)→Y, де:

- X – множина ознак;
- Y – множина класів:

Y={0,1}, де:

- 0 – легітимне програмне забезпечення;
- 1 – шкідливе програмне забезпечення.

Основною метою моделі машинного навчання є побудова такої функції класифікації, яка забезпечуватиме максимальну точність визначення належності програмного забезпечення до відповідного класу [5]. Вставимо схему процесу класифікації malware (Рис. 2.1) [27].

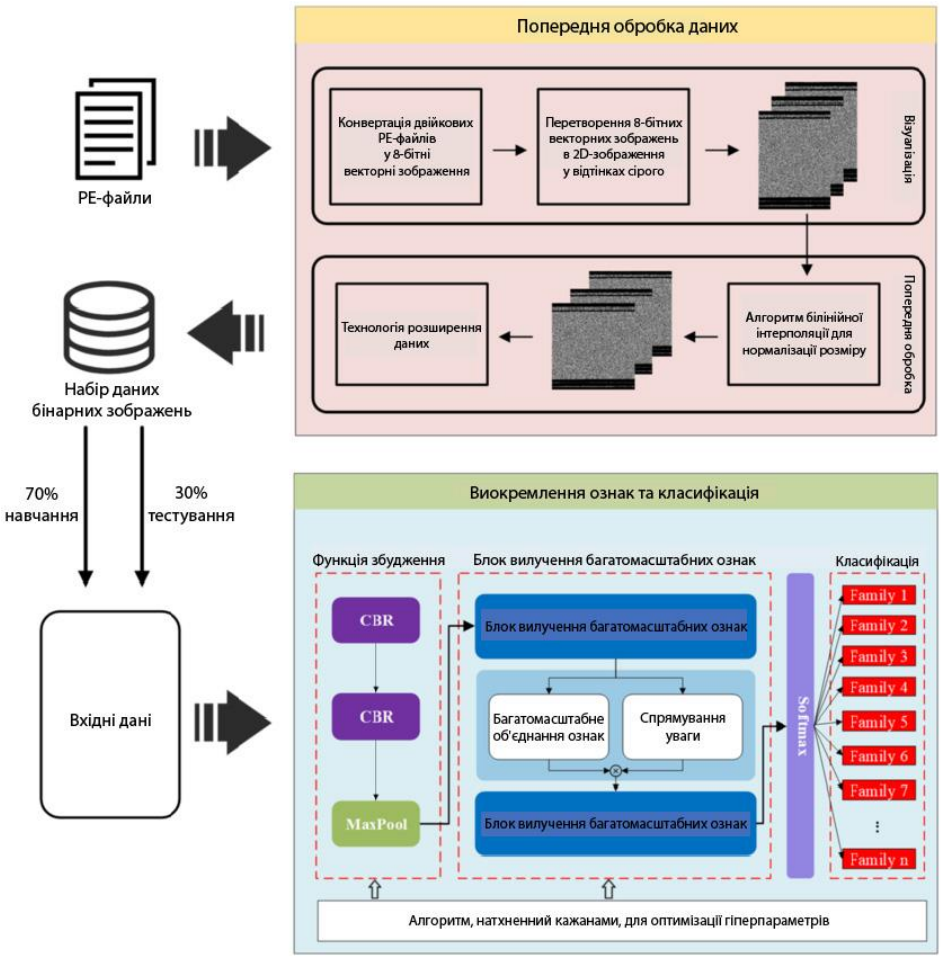

Рис. 2.1. Загальна схема процесу класифікації шкідливого програмного забезпечення.

Для побудови системи класифікації необхідно виконати декілька основних етапів:

1. Збір та підготовка набору даних;
2. Виділення та обробка ознак;
3. Поділ даних на навчальну та тестову вибірки;
4. Навчання моделі машинного навчання;
5. Оцінювання якості класифікації;
6. Тестування системи на нових даних.

Одним із важливих етапів є формування вектора ознак, який використовується як вхідні дані для алгоритму машинного навчання. У межах даної роботи передбачається використання методів статичного аналізу виконуваних файлів та формування числового представлення ознак програмного забезпечення.

Для оцінювання ефективності системи класифікації використовуються стандартні метрики ML:

- Accuracy;
- Precision;
- Recall;
- F1-score.

Точність класифікації (Accuracy) визначається співвідношенням правильно класифікованих об’єктів до загальної кількості досліджуваних зразків [7]:

Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$, де:

- TP – кількість правильно визначених шкідливих об’єктів;
- TN – кількість правильно визначених безпечних об’єктів;
- FP – кількість хибнопозитивних результатів;
- FN – кількість хибнонегативних результатів.

Крім Accuracy, важливе значення у задачах кібербезпеки мають метрики Precision та Recall, оскільки значна кількість хибних спрацювань може негативно впливати на ефективність системи захисту.

Для узагальнення основних елементів задачі класифікації доцільно навести основні елементи задачі класифікації ШПЗ в табл. 2.1. [4]–[8].

Таблиця 2.1.

Основні елементи задачі класифікації шкідливого програмного забезпечення

Елемент	**Опис**
Вхідні дані	Ознаки програмного забезпечення
Класи	Malware / Benign
Метод аналізу	Машинне навчання
Результат	Клас програмного забезпечення
Метрики оцінки	Accuracy, Precision, Recall, F1-score

Таким чином, в межах даної роботи задача класифікації формалізується як задача бінарної класифікації програмного забезпечення на основі аналізу ознак виконуваних файлів та використання методів машинного навчання. Визначення структури вхідних даних, класів та метрик оцінювання є основою для подальшого проєктування системи аналізу та класифікації ШПЗ.

2.2. Методи обробки та відбору ознак

Одним із найважливіших етапів побудови системи класифікації шкідливого програмного забезпечення є обробка та відбір ознак. Якість сформованого набору ознак безпосередньо впливає на точність роботи моделей ML, швидкість навчання алгоритмів та ефективність виявлення malware. Наявність надлишкових або нерелевантних ознак може призводити до зниження продуктивності системи та збільшення кількості помилок класифікації. [5]

В задачах аналізу ШПЗ ознаки можуть формуватись на основі:

- статичного аналізу виконуваних файлів;

- динамічного аналізу поведінки програм;
- поведінкових характеристик malware;
- статистичних параметрів коду;
- мережевої активності.

У межах даної роботи основна увага приділяється статичним ознакам виконуваних PE-файлів, оскільки такий підхід забезпечує високу швидкість аналізу та дозволяє автоматизувати процес класифікації.

До основних ознак, які можуть використовуватись у системі, належать [2]:

- розмір файлу;
- кількість секцій PE-файлу;
- ентропія секцій;
- кількість API-викликів;
- характеристики таблиці імпорту;
- кількість DLL-бібліотек;
- статистичні характеристики байтового коду;
- інформація про заголовки PE-файлу.

Продемонструємо схему процесу обробки даних (Рис. 2.2) [28].

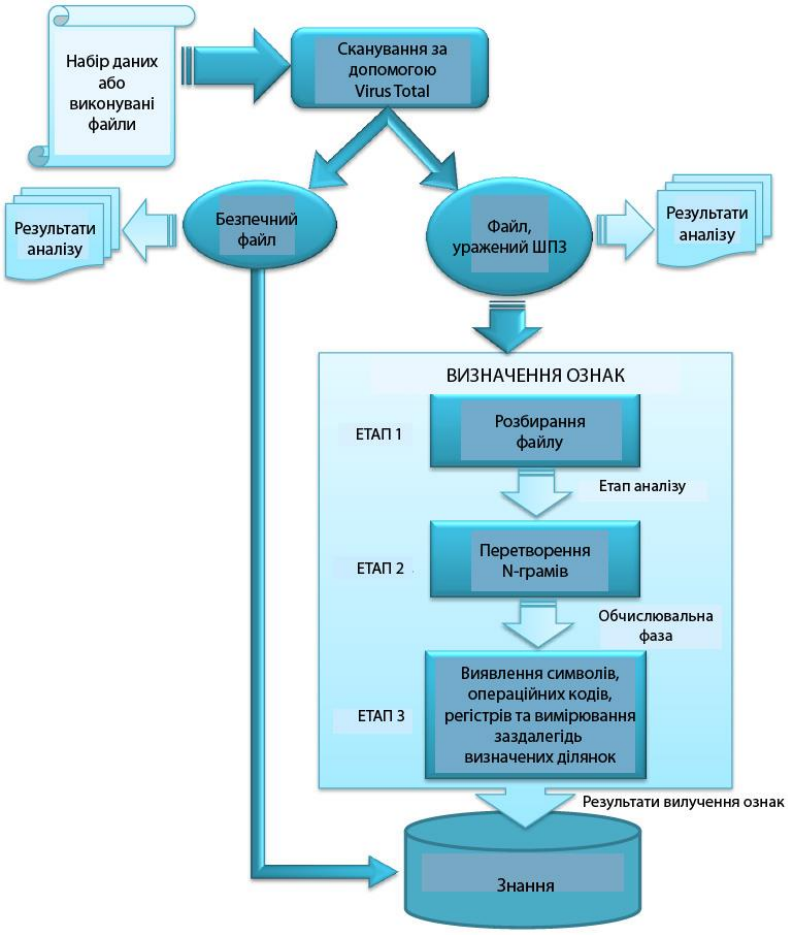

Рис. 2.2. Процес обробки та відбору ознак ШПЗ.

Перед використанням ознак у моделі машинного навчання необхідно виконати попередню обробку даних. Основними етапами preprocessing є:

- очищення даних;
- усунення пропущених значень;
- нормалізація;
- кодування категоріальних параметрів;
- масштабування ознак.

Нормалізація використовується для приведення числових значень до єдиного масштабу, що дозволяє підвищити стабільність роботи алгоритмів машинного навчання. Одним із найпоширеніших методів є Min-Max нормалізація [7]:

$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$, де:

- x – початкове значення ознаки;

- xmin – мінімальне значення ознаки;
- xmax – максимальне значення ознаки;
- x′ – нормалізоване значення.

Для зменшення розмірності даних та підвищення ефективності класифікації застосовуються методи відбору ознак (feature selection). Основною метою feature selection є визначення найбільш інформативних характеристик програмного забезпечення та усунення надлишкових параметрів.

Методи відбору ознак поділяються на три основні категорії:

- filter methods (методи фільтрації);
- wrapper methods (методи-обгортки);
- embedded methods (вбудовані методи).

Filter methods базуються на статистичному аналізі ознак без використання конкретної моделі машинного навчання. До найбільш поширених методів належать:

- Correlation Analysis;
- Chi-Square;
- Information Gain;
- Mutual Information.

Перевагою filter methods є висока швидкість роботи та простота реалізації [4].

Wrapper methods використовують результати роботи ML-моделі для оцінювання важливості ознак. У межах такого підходу виконується багаторазове навчання моделі з різними комбінаціями ознак.

До найбільш поширених методів належать:

- Forward Selection;
- Backward Elimination;
- Recursive Feature Elimination.

Основним недоліком wrapper methods є значна обчислювальна складність [5].

Embedded methods виконують відбір ознак безпосередньо під час навчання моделі машинного навчання. Прикладом такого підходу є алгоритми Random Forest та Lasso Regression, які автоматично оцінюють важливість ознак.

Використання embedded methods дозволяє поєднати процес навчання моделі та оптимізацію набору ознак [8].

Порівняємо методи відбору ознак у вигляді табл. 2.2.

Таблиця 2.2.

Порівняння методів відбору ознак

Метод	**Основна особливість**	**Переваги**	**Недоліки**
Filter	Статистичний аналіз	Висока швидкість	Менша точність
Wrapper	Оцінка через ML-модель	Висока якість відбору	Значні витрати ресурсів
Embedded	Відбір під час навчання	Баланс швидкості та точності	Залежність від моделі

Однією з важливих проблем у задачах класифікації ШПЗ є висока розмірність даних. Для зменшення кількості ознак можуть використовуватись методи reduction, зокрема Principal Component Analysis (PCA). PCA дозволяє зменшити кількість параметрів шляхом формування нових узагальнених компонент, що зберігають основну інформацію набору даних [7].

Використання методів обробки та відбору ознак дозволяє:

- підвищити точність класифікації;
- зменшити час навчання моделей;
- знизити ризик перенавчання;
- підвищити продуктивність системи.

Таким чином, процес обробки та відбору ознак є важливим етапом побудови системи аналізу та класифікації шкідливого програмного забезпечення. Використання сучасних методів preprocessing та feature selection

дозволяє підвищити ефективність моделей машинного навчання та забезпечити більш точне виявлення malware.

2.3. Вибір та обґрунтування алгоритмів

Одним із ключових етапів розробки системи класифікації ШПЗ є вибір алгоритмів ML, які забезпечуватимуть ефективне виявлення malware на основі аналізу ознак виконуваних файлів. Від правильного вибору алгоритму залежить точність класифікації, швидкість роботи системи, стійкість до шумових даних та здатність системи працювати з новими типами загроз [5].

У задачах аналізу шкідливого програмного забезпечення найчастіше використовуються алгоритми:

- Decision Tree;
- Random Forest;
- Support Vector Machine (SVM);
- K-Nearest Neighbors (KNN);
- Logistic Regression;
- нейронні мережі.

Для вибору найбільш доцільного алгоритму необхідно враховувати [7]:

- тип та розмір набору даних;
- кількість ознак;
- швидкість навчання;
- обчислювальну складність;
- точність класифікації;
- стійкість до перенавчання.

Decision Tree є одним із базових алгоритмів класифікації, який будує дерево рішень на основі набору ознак. Кожен вузол дерева відповідає певній умові, а кінцеві вузли визначають результат класифікації.

Основними перевагами Decision Tree є:

- простота реалізації;
- висока швидкість роботи;

- наочність результатів;
- можливість роботи з числовими та категоріальними даними.

Проте алгоритм має схильність до перенавчання, особливо у випадку великих наборів даних або надлишкової кількості ознак [4].

Random Forest є ансамблевим методом ML, який складається з великої кількості дерев рішень. Кожне дерево навчається на випадковій підмножині даних та ознак, а фінальний результат визначається шляхом голосування.

Основними перевагами Random Forest є:

- висока точність класифікації;
- стійкість до шумових даних;
- зменшення ризику перенавчання;
- можливість оцінювання важливості ознак.

Завдяки високій ефективності та стабільності Random Forest широко використовується у задачах класифікації malware та аналізу кіберзагроз [6], [16].

Продемонструємо принцип роботи алгоритма (Рис. 2.3) [29].

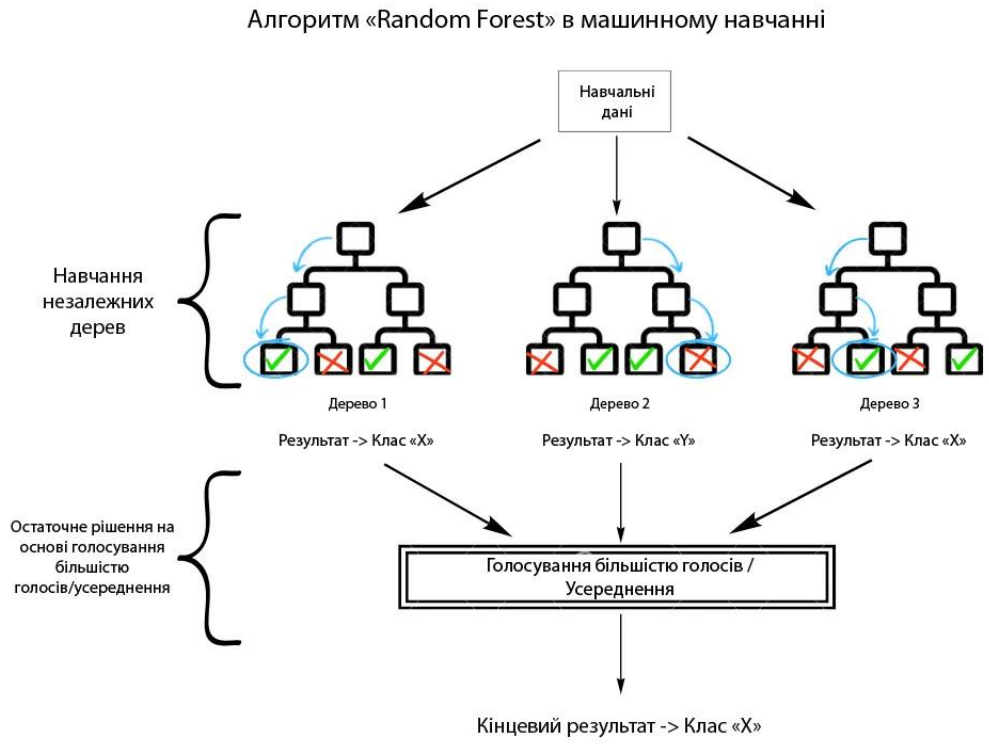

Рис. 2.3. Принцип роботи алгоритму Random Forest.

Support Vector Machine є алгоритмом бінарної класифікації, який виконує побудову оптимальної гіперплощини для розділення об’єктів різних класів. Основною перевагою SVM є ефективна робота з високовимірними даними та можливість використання різних kernel-функцій [8], [17].

До переваг алгоритму належать:

- висока точність класифікації;
- ефективність для складних наборів даних;
- стійкість до великої кількості ознак.

Недоліками SVM є:

- висока обчислювальна складність;
- значний час навчання;
- складність налаштування параметрів моделі.

Алгоритм *K-Nearest Neighbors* базується на визначенні найближчих сусідів об’єкта у просторі ознак. Класифікація виконується на основі класів найближчих об’єктів [5].

Перевагами KNN є:

- простота реалізації;
- відсутність етапу навчання;
- ефективність для невеликих наборів даних.

Основними недоліками алгоритму є:

- низька продуктивність при великих наборах даних;
- залежність від вибору параметра k;
- чутливість до шумових даних.

Серед алгоритмів машинного навчання для задач класифікації також широко використовується *Logistic Regression*. Даний алгоритм належить до методів лінійної класифікації та застосовується для прогнозування належності об’єкта до певного класу на основі набору ознак. Основними перевагами Logistic Regression є простота реалізації, висока швидкість навчання та низькі вимоги до обчислювальних ресурсів [17].

У задачах виявлення шкідливого програмного забезпечення Logistic Regression може забезпечувати достатньо високі результати класифікації при використанні якісного набору ознак. Разом із тим алгоритм має обмеження при роботі зі складними нелінійними залежностями між характеристиками malware, що може знижувати ефективність класифікації порівняно з ансамблевими методами машинного навчання [16], [17].

Останніми роками у задачах кібербезпеки активно використовуються *нейронні мережі* та методи Deep Learning. Штучні нейронні мережі здатні автоматично виявляти складні закономірності у великих обсягах даних та забезпечувати високу точність класифікації malware [7], [18].

У задачах аналізу ШПЗ використовуються:

- Artificial Neural Networks (ANN);
- Convolutional Neural Networks (CNN);
- Recurrent Neural Networks (RNN).

Основними перевагами нейронних мереж є:

- висока точність;
- автоматичне виділення ознак;
- ефективність для складних задач класифікації.

Разом із тим нейронні мережі потребують:

- значних обчислювальних ресурсів;
- великих наборів даних;
- тривалого навчання моделей.

Підсумуємо та порівняємо алгоритми ML в табл. 2.3.

Таблиця 2.3.

Порівняння алгоритмів машинного навчання

Алгоритм	**Переваги**	**Недоліки**
Decision Tree	Простота реалізації	Схильність до overfitting
Random Forest	Висока точність	Велика кількість дерев

Продовження таблиці 2.3.

Алгоритм	**Переваги**	**Недоліки**
SVM	Робота з високовимірними даними	Висока складність
KNN	Простота використання	Низька швидкість
Logistic Regression	Висока швидкість роботи, простота	Гірша робота з нелінійними залежностями
Нейронні мережі	Висока ефективність	Значні витрати ресурсів

У межах даної роботи як основний алгоритм класифікації використаємо Random Forest. Вибір даного алгоритму обумовлений:

- високою точністю класифікації;
- стійкістю до шумових даних;
- ефективною роботою з великою кількістю ознак;
- відносно невисокою складністю реалізації;
- можливістю оцінювання важливості ознак.

Крім того, Random Forest демонструє хороші результати у задачах аналізу malware та широко використовується у сучасних дослідженнях кібербезпеки [6].

Для додаткового порівняння ефективності системи у роботі також можуть використовуватись інші алгоритми машинного навчання, зокрема SVM або Logistic Regression.

Таким чином, проведений аналіз алгоритмів ML дозволив визначити найбільш доцільний підхід для побудови системи класифікації ШПЗ. Використання Random Forest у поєднанні з методами відбору ознак дозволить забезпечити ефективне виявлення malware та підвищити точність класифікації програмного забезпечення [16].

2.4. Запропонований підхід

У результаті проведеного аналізу сучасних підходів до виявлення та класифікації ШПЗ було встановлено, що найбільш перспективним напрямом є використання методів машинного навчання у поєднанні зі статичним аналізом виконуваних файлів. Запропонований у роботі підхід передбачає автоматизовану класифікацію програмного забезпечення на основі аналізу ознак PE-файлів та використання алгоритмів машинного навчання для визначення належності об’єкта до класу malware або benign software [5].

Основною ідеєю запропонованого підходу є поєднання:

- методів статичного аналізу;
- процедур обробки та відбору ознак;
- алгоритмів машинного навчання;
- автоматизованого процесу класифікації.

На відміну від традиційних сигнатурних методів, запропонований підхід дозволяє виявляти нові або модифіковані типи шкідливого програмного забезпечення шляхом аналізу закономірностей у наборі ознак [8].

У межах системи передбачається використання статичного аналізу виконуваних PE-файлів без запуску програмного забезпечення у реальному середовищі. Це дозволяє [2]:

- зменшити ризик зараження системи;
- скоротити час аналізу;
- знизити навантаження на обчислювальні ресурси;
- автоматизувати процес аналізу malware.

Вставимо загальну схему запропонованого підходу (Рис. 2.4) [30].

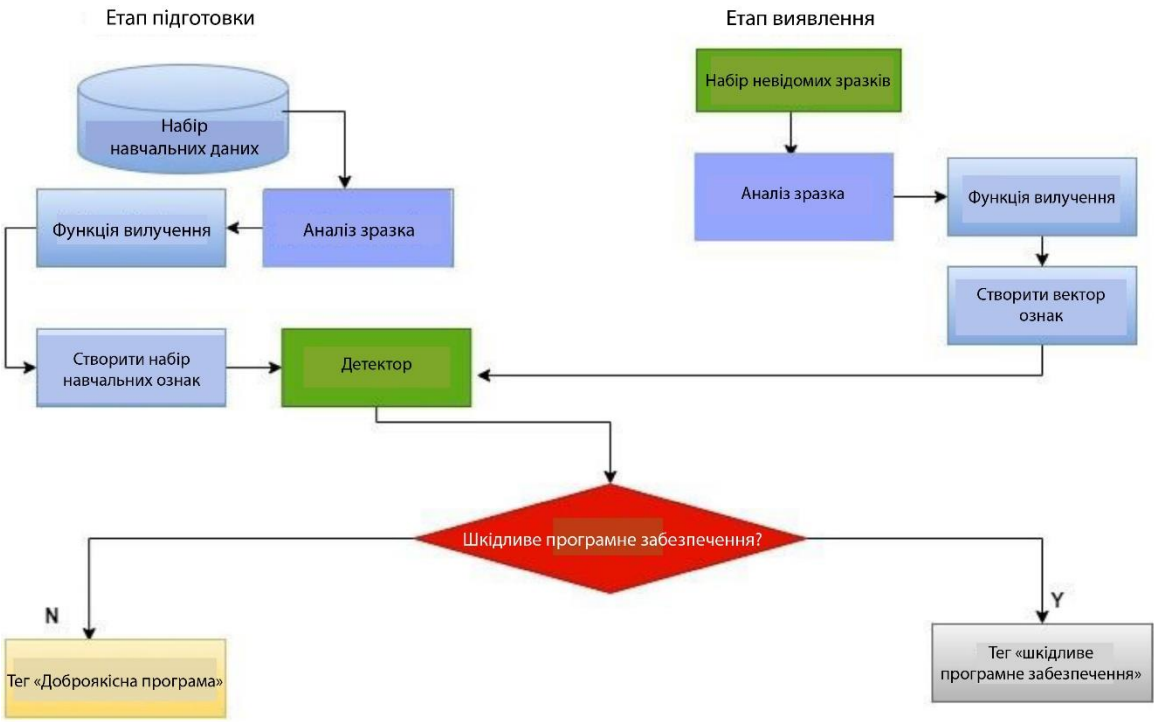

Рис. 2.4. Загальна схема запропонованого підходу до класифікації ШПЗ.

Запропонований підхід передбачає виконання декількох основних етапів:

1. Отримання набору досліджуваних PE-файлів;
2. Виділення статичних ознак програмного забезпечення;
3. Попередню обробку даних;
4. Відбір найбільш інформативних ознак;
5. Навчання моделі машинного навчання;
6. Тестування системи;
7. Класифікацію програмного забезпечення.

На етапі feature extraction із PE-файлів формуються характеристики, які використовуються як вхідні дані для моделі машинного навчання. Основними ознаками можуть бути [2], [6]:

- кількість секцій;
- ентропія секцій;
- характеристики PE-заголовків;
- API-виклики;
- статистичні параметри файлу;

- характеристики таблиці імпорту.

Після формування набору ознак виконується preprocessing даних, який включає:

- очищення даних;
- нормалізацію;
- усунення пропущених значень;
- масштабування ознак.

Для зменшення розмірності даних та підвищення точності класифікації використовується feature selection. У межах роботи передбачається застосування методів оцінювання важливості ознак на основі алгоритму Random Forest [16].

Для побудови системи класифікації у роботі пропонується використання алгоритму Random Forest. Вибір алгоритму обумовлений [6], [16]:

- високою точністю класифікації;
- стійкістю до шумових даних;
- ефективною роботою з великою кількістю ознак;
- можливістю оцінювання важливості ознак;
- низькою схильністю до перенавчання.

Процес навчання моделі передбачає формування навчальної та тестової вибірок:

$D = D_{train} \cup D_{test}$, де:

- D_{train} – навчальна вибірка;
- D_{test} – тестова вибірка.

Після навчання моделі виконується оцінювання якості класифікації за допомогою стандартних метрик [7]:

- Accuracy;
- Precision;
- Recall;
- F1-score.

Запропонуємо процес навчанні ML-моделі у вигляді Рис. 2.5 [31].

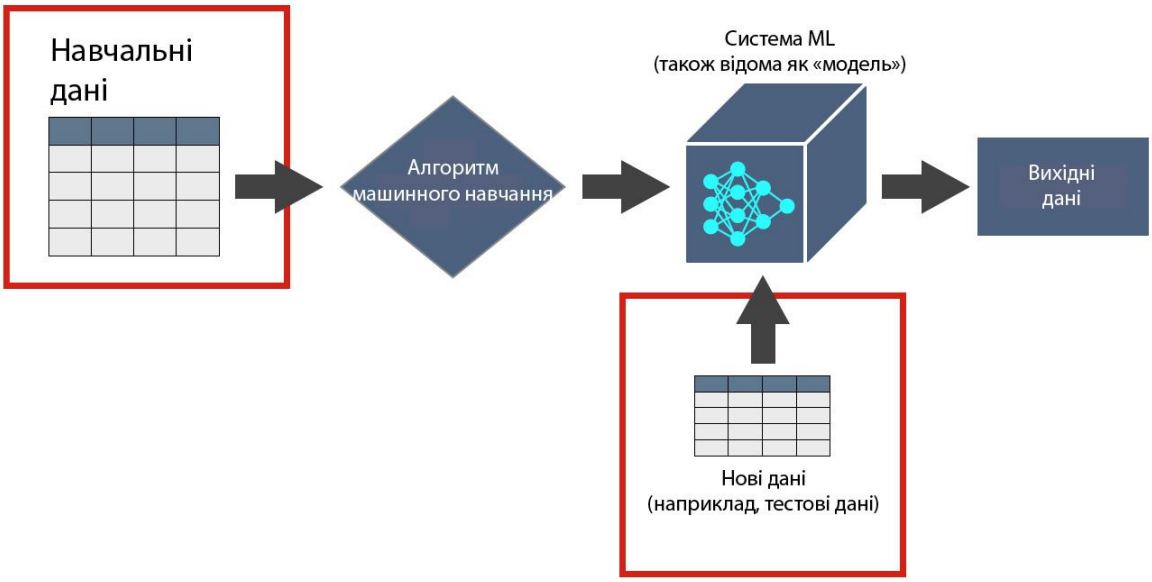

Рис. 2.5. Процес навчання моделі класифікації malware.

Для оцінювання ефективності запропонованого підходу планується проведення експериментального дослідження із використанням набору даних malware та benign software. Результати класифікації дозволять визначити точність роботи системи та порівняти її ефективність з існуючими підходами.

Для узагальнення основних етапів запропонованого підходу наведемо інформацію в табл. 2.4. [5], [16].

Таблиця 2.4.

Основні етапи запропонованого підходу

Етап	**Основний зміст**
Збір даних	Формування набору PE-файлів
Виділення ознак	Отримання характеристик malware
Обробка даних	Нормалізація та preprocessing
Feature Selection	Відбір інформативних ознак

Продовження таблиці 2.4.

Етап	**Основний зміст**
Навчання моделі	Побудова Random Forest
Класифікація	Визначення типу ПЗ

Таким чином, запропонований підхід базується на використанні методів статичного аналізу та алгоритмів машинного навчання для автоматизованої класифікації шкідливого програмного забезпечення. Використання Random Forest та процедур відбору ознак дозволяє підвищити точність класифікації, знизити вплив надлишкових даних та забезпечити ефективний аналіз сучасних кіберзагроз.

2.5. Архітектура системи

Одним із важливих етапів розробки системи аналізу та класифікації ШПЗ є проєктування її архітектури. Правильно побудована архітектура системи забезпечує ефективну взаємодію між окремими компонентами, автоматизацію процесу аналізу malware та можливість масштабування системи у майбутньому.

Запропонована система призначена для автоматизованого аналізу виконуваних PE-файлів та їх класифікації за допомогою алгоритмів машинного навчання. Архітектура системи побудована за модульним принципом, що дозволяє окремо реалізовувати, тестувати та вдосконалювати функціональні компоненти системи [5].

Основними компонентами системи є:

- модуль завантаження файлів;
- модуль аналізу PE-файлів;
- модуль виділення ознак;
- модуль обробки даних;
- модуль машинного навчання;
- модуль класифікації;
- модуль формування результатів.

Перелічивши компоненти, вставимо головну архітектурну схему (Рис. 2.6) [32].

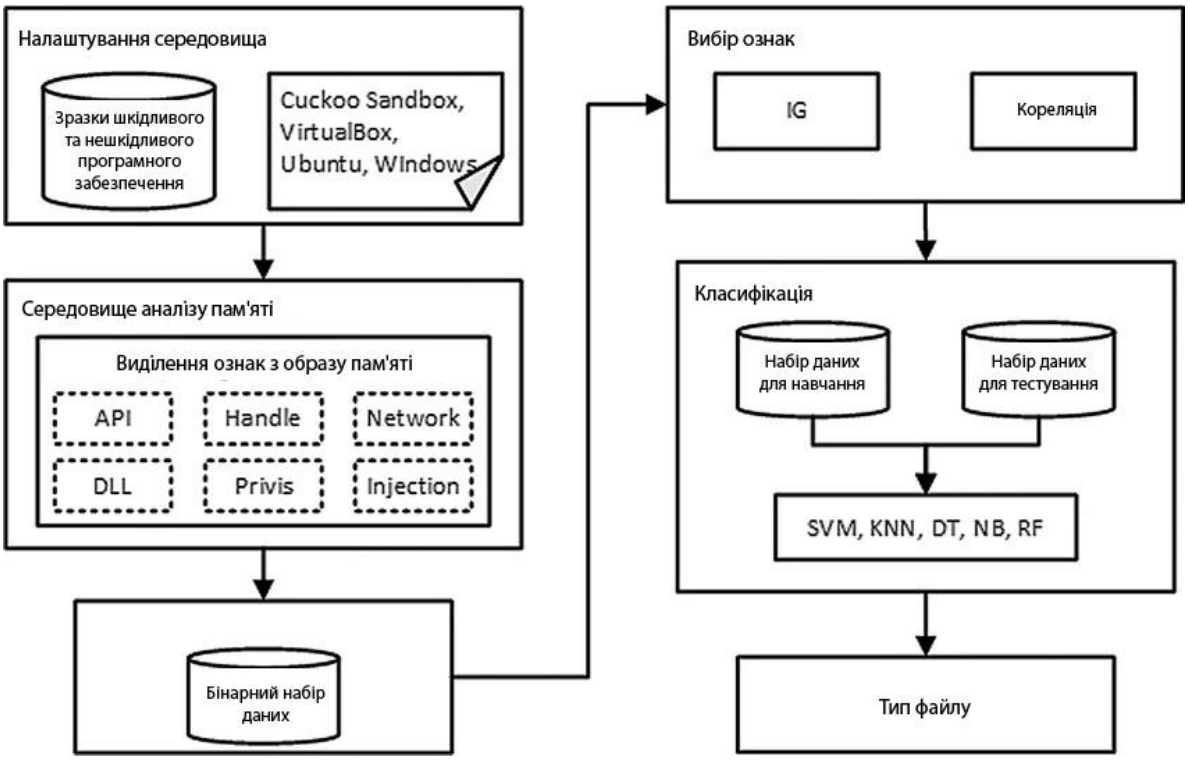

Рис. 2.6. Архітектура системи класифікації malware.

Модуль завантаження файлів відповідає за отримання виконуваних PE-файлів, які надходять до системи для подальшого аналізу. Основними функціями модуля є:

- перевірка типу файлу;
- зчитування даних;
- передача файлу до модуля аналізу.

На цьому етапі система здійснює первинну перевірку формату файлу та його коректності.

Модуль аналізу PE-файлів виконує статичний аналіз структури виконуваного файлу. Основною задачею даного компонента є отримання інформації про [2]:

- PE-заголовки;
- секції файлу;
- таблицю імпорту;

- API-виклики;
- характеристики виконуваного коду.

У межах роботи передбачається використання статичного аналізу без запуску програмного забезпечення у середовищі виконання, що дозволяє знизити ризик зараження системи та зменшити витрати обчислювальних ресурсів.

Після аналізу PE-файлу система формує *набір ознак*, які використовуються для навчання та роботи ML-моделі. Модуль feature extraction відповідає за формування числового представлення характеристик програмного забезпечення.

До основних ознак належать [6]:

- ентропія секцій;
- кількість DLL-бібліотек;
- кількість API-викликів;
- розмір секцій;
- статистичні параметри PE-файлу;
- характеристики імпортованих функцій.

Вставимо допоміжне зображення, де буде показано детальну структуру PE (Рис. 2.7) [33].

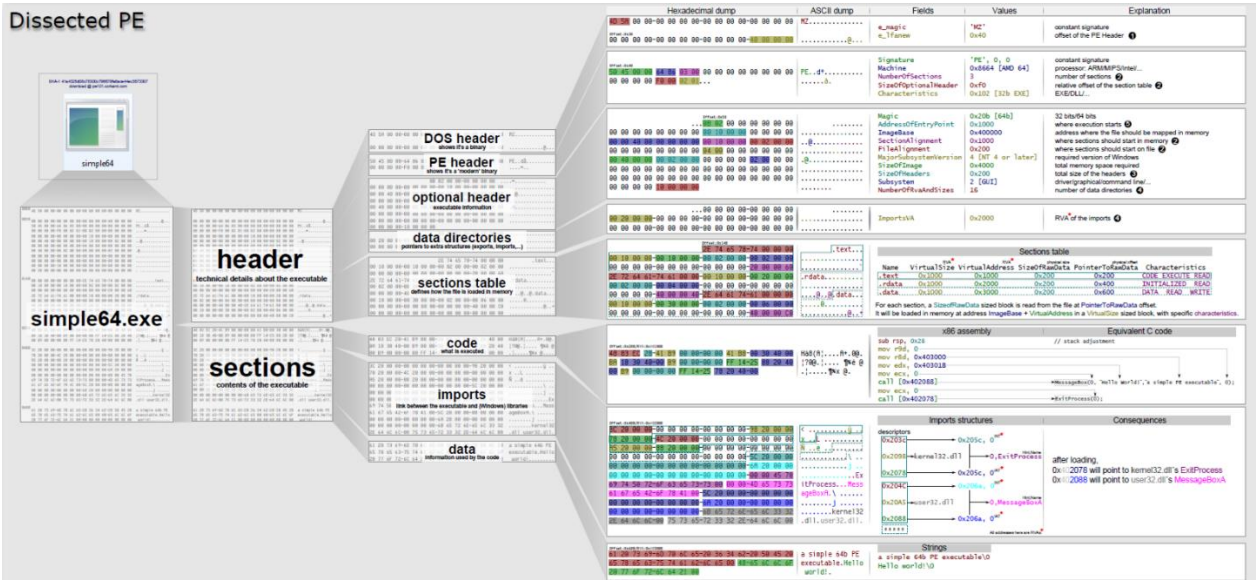

Рис. 2.7. Візуалізація детальної структури PE.

Модуль preprocessing виконує попередню обробку даних перед подачею до алгоритму машинного навчання. Основними задачами модуля є [7]:

- очищення набору даних;
- нормалізація ознак;
- масштабування;
- усунення пропущених значень;
- відбір інформативних ознак.

Для зменшення розмірності даних можуть використовуватись методи feature selection та Principal Component Analysis (PCA).

Модуль машинного навчання є основним компонентом системи класифікації malware. Саме цей модуль виконує:

- навчання моделі;
- збереження параметрів;
- прогнозування результату класифікації;
- оцінювання ефективності алгоритму.

У межах роботи передбачається використання алгоритму Random Forest, який забезпечує високу точність класифікації та стійкість до шумових даних [16].

Модель машинного навчання реалізує функцію:

$F(X) \rightarrow Y$, де:

- X – вектор ознак;
- Y – результат класифікації.

Після завершення аналізу система формує результат класифікації програмного забезпечення. *Модуль класифікації* визначає належність об’єкта до одного з класів:

- malware;
- benign software.

Результат роботи системи може містити:

- клас програмного забезпечення;
- рівень ймовірності;
- значення основних ознак;

- інформацію про модель класифікації.

Узагальнимо компоненти системи у вигляді табл. 2.5. [5], [16].

Таблиця 2.5.

Основні компоненти системи

Компонент	**Основне призначення**
Модуль завантаження	Отримання PE-файлів
Модуль аналізу	Аналіз структури файлу
Feature Extraction	Формування ознак
Preprocessing	Обробка даних
ML-модуль	Класифікація malware
Output Module	Формування результатів

Архітектура системи побудована таким чином, щоб забезпечити можливість подальшого розширення функціональності. У майбутньому система може бути доповнена:

- динамічним аналізом malware;
- інтеграцією sandbox-середовищ;
- використанням deep learning моделей;
- автоматичним оновленням наборів даних.

Таким чином, запропонована архітектура системи забезпечує ефективну організацію процесу аналізу та класифікації шкідливого програмного забезпечення. Модульний підхід дозволяє підвищити гнучкість системи, спростити процес реалізації та забезпечити можливість подальшого вдосконалення системи кіберзахисту.

2.6. Алгоритм роботи системи

Для забезпечення ефективного аналізу та класифікації ШПЗ необхідно визначити послідовність роботи основних компонентів системи. Алгоритм

роботи системи описує порядок взаємодії модулів, етапи обробки даних та процес формування результату класифікації malware.

Запропонований алгоритм функціонування системи базується на використанні методів статичного аналізу виконуваних файлів та алгоритмів машинного навчання. Основною задачею системи є автоматизоване визначення належності програмного забезпечення до класу malware або benign software на основі аналізу набору ознак [5].

У загальному вигляді алгоритм роботи системи складається з таких етапів:

1. Отримання виконуваного файлу;
2. Перевірка коректності формату;
3. Аналіз структури PE-файлу;
4. Виділення ознак;
5. Попередня обробка даних;
6. Відбір інформативних ознак;
7. Класифікація програмного забезпечення;
8. Формування результату аналізу.

Вставимо блок-схему алгоритму (Рис. 2.8) [34].

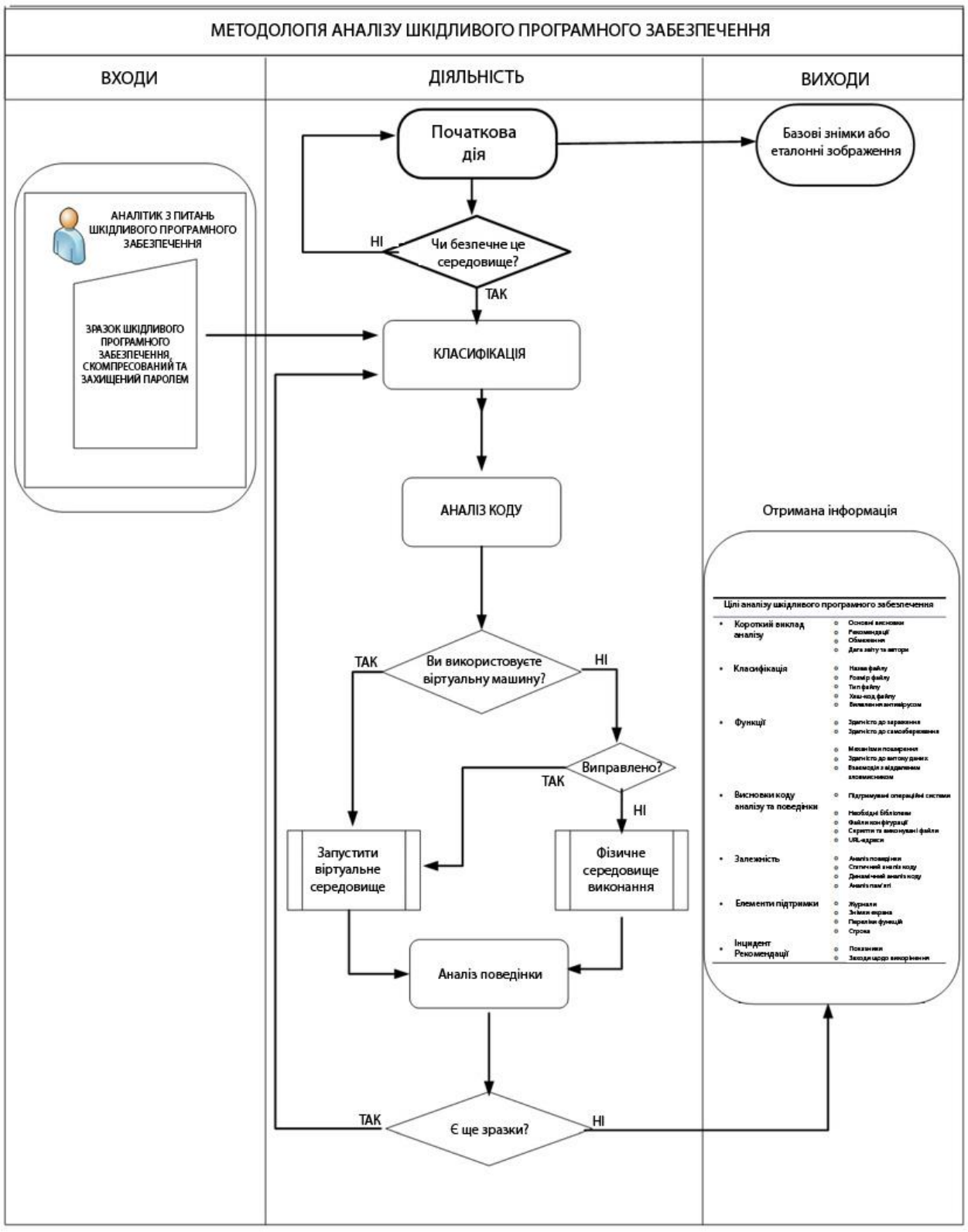

Рис. 2.8. Алгоритм роботи системи аналізу та класифікації malware.

На першому етапі система отримує виконуваний файл для подальшого аналізу. Після завантаження здійснюється перевірка:

- формату файлу;
- цілісності даних;
- наявності PE-структури.

Якщо файл не відповідає вимогам системи, подальший аналіз не виконується [2].

Після перевірки файлу система виконує статичний аналіз PE-структури. На цьому етапі здійснюється зчитування:

- PE-заголовків;
- секцій виконуваного файлу;
- таблиці імпорту;
- API-функцій;
- характеристик виконуваного коду.

Статичний аналіз дозволяє отримати основні характеристики програмного забезпечення без запуску файлу у середовищі виконання [6], [20].

На основі отриманих характеристик система формує вектор ознак:

$X=(f_1,f_2,f_3,\ldots,f_n)$, де:

- f_i – окрема ознака PE-файлу;
- n – кількість ознак.

До набору ознак можуть входити [8], [20]:

- ентропія секцій;
- кількість DLL;
- розмір секцій;
- кількість API-викликів;
- характеристики таблиці імпорту;
- статистичні параметри файлу.

Після формування ознак виконується preprocessing даних:

- нормалізація;
- масштабування;
- очищення даних;
- усунення пропущених значень.

Для підвищення ефективності системи застосовується feature selection, що дозволяє залишити лише найбільш інформативні параметри [7].

Після завершення preprocessing дані передаються до моделі ML. У межах роботи використовується алгоритм Random Forest, який виконує класифікацію програмного забезпечення на основі сформованого набору ознак [16], [19].

Функцію класифікації можна подати у вигляді:

Y=F(X), де:

- X – вектор ознак;
- Y – результат класифікації.

Результатом роботи системи є визначення одного з класів:

- malware;
- benign software.

Крім класу програмного забезпечення система може формувати рівень ймовірності класифікації.

На завершальному етапі система формує результат роботи та передає його користувачу. Результат може містити:

- тип програмного забезпечення;
- значення основних ознак;
- результат класифікації;
- рівень впевненості моделі;
- інформацію про використаний алгоритм.

Для оцінювання ефективності роботи системи використовуються метрики [5]:

- Accuracy;
- Precision;
- Recall;
- F1-score.

Узагальнимо етапи алгоритму в табл. 2.6.

Таблиця 2.6.

Основні етапи алгоритму роботи системи

Етап	**Основний зміст**
Завантаження файлу	Отримання PE-файлу
Аналіз структури	Зчитування PE-параметрів
Виділення ознак	Формування вектора ознак
Preprocessing	Обробка та нормалізація
Feature Selection	Відбір важливих ознак
Класифікація	Робота ML-моделі
Формування результату	Виведення результату

Для підвищення продуктивності системи алгоритм може бути доповнений:

- кешуванням результатів аналізу;
- автоматичним оновленням моделі;
- інтеграцією нових наборів даних;
- підтримкою динамічного аналізу malware.

Таким чином, запропонований алгоритм роботи системи забезпечує автоматизований процес аналізу та класифікації ШПЗ. Використання методів статичного аналізу, preprocessing даних та алгоритмів машинного навчання дозволяє підвищити ефективність виявлення malware та забезпечити стабільну роботу системи класифікації.

2.7. Вибір технологій реалізації

Під час розробки системи аналізу та класифікації ШПЗ важливим етапом є вибір технологій реалізації, які забезпечуватимуть ефективну обробку даних, навчання моделей машинного навчання та автоматизацію процесу аналізу malware. Вибір програмних засобів і бібліотек повинен враховувати [5]:

- продуктивність системи;
- простоту реалізації;
- підтримку алгоритмів машинного навчання;

- можливість роботи з наборами даних;
- масштабованість системи;
- підтримку аналізу PE-файлів.

У межах роботи для реалізації системи пропонується використання мови програмування Python. Даний вибір обумовлений широким використанням Python у задачах машинного навчання, аналізу даних та кібербезпеки. Мова Python має значну кількість бібліотек для реалізації ML-моделей, preprocessing даних та роботи з виконуваними файлами [21].

Основними перевагами Python є:

- простота синтаксису;
- велика кількість готових бібліотек;
- підтримка machine learning;
- кросплатформеність;
- активна спільнота розробників.

Для реалізації алгоритмів машинного навчання доцільно використати бібліотеку Scikit-learn. Дана бібліотека містить велику кількість готових алгоритмів класифікації та засобів preprocessing даних [22].

У межах роботи бібліотека Scikit-learn може використовуватись для:

- реалізації алгоритму Random Forest;
- поділу даних на train/test вибірки;
- оцінювання метрик класифікації;
- нормалізації ознак;
- feature selection.

Однією з основних переваг бібліотеки є простота інтеграції та ефективна робота з наборами даних середнього розміру.

Для роботи з наборами даних та обробки ознак доцільно використовувати бібліотеки:

- NumPy;
- Pandas.

Бібліотека NumPy забезпечує ефективну роботу з багатовимірними масивами та математичними операціями, а Pandas використовується для обробки табличних даних та формування dataset [21].

Для візуалізації результатів роботи системи можуть використовуватись:

- Matplotlib;
- Seaborn.

Дані бібліотеки дозволяють будувати:

- графіки;
- гістограми;
- confusion matrix;
- діаграми важливості ознак.

Для реалізації статичного аналізу виконуваних файлів доцільно використовувати бібліотеку pefile. Вона дозволяє виконувати аналіз структури PE-файлів та отримувати інформацію про [20]:

- заголовки PE;
- секції;
- імпортовані DLL;
- API-виклики;
- характеристики виконуваного файлу.

Використання бібліотеки pefile дозволяє автоматизувати процес виділення ознак malware та інтегрувати його у загальний pipeline системи.

Як середовище розробки доцільно використовувати:

- PyCharm;
- Visual Studio Code;
- Jupyter Notebook.

Jupyter Notebook може застосовуватись для:

- аналізу даних;
- тестування моделей;
- проведення експериментів;
- візуалізації результатів.

Для зберігання вихідного коду та контролю версій доцільно використовувати Git та платформу GitHub [23].

Для навчання моделі машинного навчання можуть використовуватись [13]:

- Microsoft Malware Classification Challenge Dataset;
- EMBER Dataset;
- власні набори PE-файлів.

Набір даних може містити:

- шкідливі зразки;
- легітимне програмне забезпечення;
- набір ознак для навчання моделі.

Підсумуємо технології реалізації в табл. 2.7.

Таблиця 2.7.

Технології реалізації системи

Технологія	**Призначення**
Python	Основна мова реалізації
Scikit-learn	Алгоритми ML
NumPy	Робота з масивами
Pandas	Обробка dataset
pefile	Аналіз PE-файлів
Matplotlib	Візуалізація
Git/GitHub	Контроль версій

Таким чином, вибір сучасних технологій реалізації дозволяє забезпечити ефективну побудову системи аналізу та класифікації ШПЗ. Використання Python, бібліотек машинного навчання та засобів аналізу PE-файлів забезпечує гнучкість системи, автоматизацію процесу аналізу malware та можливість подальшого вдосконалення функціональності системи.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи було виконано розробку методу та проєктування системи класифікації ШПЗ на основі методів ML.

В межах розділу проведено формалізацію задачі класифікації malware, визначено основні вхідні та вихідні параметри системи, а також сформовано математичну модель задачі бінарної класифікації програмного забезпечення. Встановлено, що ефективність роботи системи значною мірою залежить від якості сформованого набору ознак та вибору алгоритму машинного навчання.

Досліджено методи обробки та відбору ознак, які використовуються у задачах аналізу шкідливого програмного забезпечення. Визначено основні підходи до preprocessing даних, нормалізації та feature selection. Встановлено, що використання процедур відбору ознак дозволяє зменшити розмірність даних, підвищити точність класифікації та знизити ризик перенавчання моделі [5], [7].

У процесі аналізу алгоритмів машинного навчання було розглянуто особливості Decision Tree, Random Forest, Support Vector Machine, K-Nearest Neighbors та нейронних мереж. За результатами порівняння встановлено, що алгоритм Random Forest є найбільш доцільним для реалізації системи класифікації malware завдяки високій точності, стійкості до шумових даних та можливості оцінювання важливості ознак [6], [16].

У розділі також запропоновано підхід до побудови системи класифікації ШПЗ, який базується на поєднанні статичного аналізу PE-файлів, процедур preprocessing та алгоритмів машинного навчання. Запропонований підхід дозволяє автоматизувати процес аналізу malware та забезпечити ефективне виявлення шкідливого програмного забезпечення [19], [20].

Розроблено архітектуру системи, визначено основні функціональні модулі та описано їх взаємодію. У межах архітектури передбачено використання модульного підходу, що забезпечує гнучкість системи та можливість її подальшого розширення.

Крім того, було сформовано алгоритм роботи системи, який описує послідовність виконання основних етапів аналізу та класифікації програмного забезпечення – від отримання PE-файлу до формування результату класифікації.

У процесі проєктування системи також здійснено вибір технологій реалізації. Для побудови системи обрано мову програмування Python та бібліотеки Scikit-learn, Pandas, NumPy і pefile, які забезпечують ефективну реалізацію алгоритмів машинного навчання та аналізу виконуваних файлів [21], [22].

Отже, результати другого розділу створюють основу для практичної реалізації системи аналізу та класифікації шкідливого програмного забезпечення, а також проведення експериментальних досліджень ефективності запропонованого підходу у наступному розділі роботи.

Розділ 3 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

3.1. Реалізація системи

В межах дипломної роботи було реалізовано програмну систему класифікації ШПЗ із застосуванням методів ML. Реалізація виконувалася мовою програмування Python із використанням бібліотек pandas, NumPy, scikit-learn, matplotlib та seaborn, які забезпечують засоби для обробки даних, навчання моделей та візуалізації результатів [21–24].

Основним завданням розробленої системи є автоматизована класифікація шкідливого програмного забезпечення за сімействами malware на основі характеристик PE-заголовків виконуваних файлів. Для реалізації системи було використано відкритий dataset Windows PE Malware Classification Dataset, що містить набір ознак PE-файлів та інформацію про тип шкідливого програмного забезпечення.

Загальна структура програмної системи складається з таких основних етапів:

1. Завантаження та попередня обробка dataset;
2. Формування набору ознак та цільової змінної;
3. Поділ даних на навчальну та тестову вибірки;
4. Навчання моделі Random Forest;
5. Оцінювання якості класифікації;
6. Аналіз важливості ознак;
7. Побудова графічних результатів експерименту.

На першому етапі здійснюється завантаження dataset у форматі CSV за допомогою бібліотеки pandas. Для цього використовується функція read_csv(), яка дозволяє зчитати набір даних у вигляді таблиці DataFrame. Після завантаження виконується попередній аналіз структури dataset та перевірка назв ознак (Рис. 3.1).

```
data = pd.read_csv("malware_dataset.csv")

print("Dataset loaded successfully")

print(data.head())

print("\nColumns:")
print(data.columns)
```

Рис. 3.1. Попередній аналіз структури датасету та перевірка назв ознак.

У dataset містяться як числові характеристики PE-файлів, так і службові поля, що не мають практичної цінності для задачі машинного навчання. Зокрема, поле SHA256 використовується лише для ідентифікації файлів і не впливає на процес класифікації, тому воно було видалене (Рис 3.2).

```
data = data.drop(columns=["SHA256"], errors="ignore")
```

Рис. 3.2. Видалення поля SHA256.

Для формування цільової змінної використано поле Malware_Type, яке містить назви сімейств ШПЗ. Оскільки алгоритми ML працюють із числовими значеннями, було застосовано метод LabelEncoder, що перетворює текстові назви класів у числові мітки (Рис. 3.3) [24].

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

data["Label"] = encoder.fit_transform(data["Malware_Type"])
```

Рис. 3.3. Застосування методу LabelEncoder.

Після формування цільової змінної текстове поле Malware_Type було видалено з набору даних.

Для забезпечення коректної роботи системи також виконувалася попередня обробка dataset, що включала (Рис. 3.4):

- видалення пропущених значень;
- перетворення всіх ознак у числовий формат;
- повторне очищення dataset від некоректних записів.

```
data = data.dropna()

data = data.apply(pd.to_numeric, errors='coerce')

data = data.dropna()
```

Рис. 3.4. Попередня обробка датасету.

Після завершення обробки даних було сформовано матрицю ознак X та вектор цільових класів y (Рис. 3.5).

```
X = data.drop("Label", axis=1)

y = data["Label"]
```

Рис. 3.5. Формування матриці ознак X та вектору цільових класів y.

Для навчання та тестування системи dataset було поділено на навчальну та тестову вибірки у співвідношенні 80% до 20%. Під час поділу використовувався параметр stratify, що забезпечує однаковий розподіл класів у вибірках (Рис. 3.6).

```
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
```

Рис. 3.6. Розподіл на навчальну та тестову вибірки.

Як основний алгоритм класифікації було використано Random Forest – ансамблевий метод машинного навчання, що базується на побудові множини

дерев рішень [16]. Для підвищення стабільності та точності класифікації було використано 500 дерев рішень та додаткові параметри обмеження глибини дерева і мінімальної кількості зразків у вузлах (Рис. 3.7).

```
model = RandomForestClassifier(
    n_estimators=500,
    max_depth=20,
    min_samples_split=5,
    min_samples_leaf=2,
    random_state=42,
    n_jobs=-1
)
```

Рис. 3.7. Використання Random Forest.

Навчання моделі здійснювалося методом fit(), після чого виконувалося прогнозування класів для тестової вибірки (Рис. 3.8).

```
model.fit(X_train, y_train)

print("\nModel training completed")
```

Рис. 3.8. Навчання моделі та прогнозування класів для тестової вибірки.

Для оцінювання ефективності системи використовувалися метрики Accuracy, Precision, Recall та F1-score, які дозволяють комплексно оцінити якість класифікації [25].

Крім цього, у системі реалізовано механізм побудови матриці помилок (Confusion Matrix) та графіка важливості ознак. Для цього використовувалися бібліотеки matplotlib і seaborn. Побудовані графіки автоматично зберігаються у директорії «results», що дозволяє використовувати їх у подальшому аналізі результатів експериментів.

Таким чином, у межах роботи було реалізовано програмну систему класифікації ШПЗ, яка забезпечує автоматизовану обробку PE-файлів, навчання моделі машинного навчання та аналіз результатів класифікації.

3.2. Реалізація обробки даних та моделей

Одним із ключових етапів розробки системи класифікації ШПЗ є попередня обробка даних та реалізація моделей машинного навчання. Якість підготовки dataset безпосередньо впливає на точність класифікації, стабільність роботи системи та коректність результатів експерименту [16], [21].

В межах реалізованої системи обробка даних виконувалася засобами бібліотеки pandas, що забезпечує ефективну роботу з табличними наборами даних. На початковому етапі dataset було завантажено у форматі DataFrame, після чого виконано аналіз структури даних, типів ознак та наявності пропущених значень.

Для задачі класифікації використовувалися ознаки PE-заголовків виконуваних файлів Windows, серед яких:

- TimeDateStamp;
- SizeOfImage;
- AddressOfEntryPoint;
- SizeOfCode;
- SizeOfInitializedData;
- NumberOfSections;
- Characteristics;
- DllCharacteristics та інші.

Загалом dataset містив понад 50 ознак, що характеризують структуру PE-файлів та можуть використовуватись для виявлення шкідливого програмного забезпечення.

На етапі попередньої обробки було видалено поле SHA256, оскільки воно використовується лише як унікальний ідентифікатор файлу та не має аналітичної цінності для машинного навчання (Рис. 3.2).

Для формування цільової змінної використовувалося поле Malware_Type, яке містило назви сімейств шкідливого програмного забезпечення. Оскільки алгоритми машинного навчання працюють із числовими значеннями, текстові

назви класів було перетворено у числові мітки за допомогою LabelEncoder (Рис. 3.3).

Після створення цільової змінної поле Malware_Type було видалено з набору даних (Рис 3.9).

```
data = data.drop(columns=["Malware_Type"], errors="ignore")
```

Рис 3.9. Видалення поля Malware_Type з набору даних.

В процесі підготовки даних також було виконано очищення dataset від пропущених та некоректних значень. Для цього використовувалися методи dropna() та to_numeric(), що забезпечують коректне перетворення всіх ознак у числовий формат (Рис. 3.4).

Після завершення обробки даних було сформовано матрицю ознак X та вектор цільових класів у (Рис. 3.5).

Для навчання та тестування моделі dataset було поділено на навчальну та тестову вибірки. У роботі використовувалося співвідношення 80% до 20%, що є одним із найбільш поширених підходів у задачах машинного навчання [24].

Крім цього, для забезпечення рівномірного розподілу класів використовувався параметр stratify, який дозволяє уникнути дисбалансу класів у вибірках (Рис. 3.6).

В межах дипломної роботи як основну модель класифікації було використано алгоритм Random Forest. Вибір даного алгоритму обумовлений його здатністю ефективно працювати з великою кількістю ознак, високою стійкістю до перенавчання та можливістю оцінювання важливості ознак [16], [22].

Для підвищення якості класифікації було налаштовано основні параметри моделі (Рис. 3.7):

- n_estimators=500 – кількість дерев рішень;
- max_depth=20 – максимальна глибина дерева;
- min_samples_split=5 – мінімальна кількість зразків для розділення вузла;

- min_samples_leaf=2 – мінімальна кількість зразків у листі дерева.

Навчання моделі здійснювалося методом fit(), після чого виконувалося прогнозування результатів класифікації для тестової вибірки (Рис. 3.8).

Для оцінювання якості класифікації використовувалися такі метрики:

- Accuracy;
- Precision;
- Recall;
- F1-score.

Використання декількох метрик дозволяє більш об’єктивно оцінити ефективність системи в умовах багатокласової класифікації malware (Рис. 3.10) [25].

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
```

Рис. 3.10. Використання декількох метрик.

Окрім оцінювання метрик класифікації, у системі було реалізовано механізм визначення важливості ознак. Це дозволяє встановити, які характеристики PE-файлів мають найбільший вплив на процес класифікації ШПЗ (Рис. 3.11).

```
feature_importance = model.feature_importances_
```

Рис. 3.11. Механізм визначення важливості ознак.

Для візуального аналізу результатів використовувалися бібліотеки matplotlib та seaborn, за допомогою яких будувалися:

- графік важливості ознак;
- матриця помилок (Confusion Matrix).

Отримані результати автоматично зберігалися у директорії «results», що дозволяє використовувати їх під час подальшого аналізу експериментів та оформлення результатів дипломної роботи.

Таким чином, в межах роботи було реалізовано повний цикл обробки даних та навчання моделі машинного навчання для задачі класифікації ШПЗ. Реалізований підхід забезпечує автоматизовану обробку dataset, навчання моделі та оцінювання результатів класифікації.

Фрагменти програмної реалізації, наведені у роботі, демонструють основні етапи функціонування системи, тоді як повний текст програмного коду подано у додатку А.

3.3. Опис датасету

Для навчання та тестування системи класифікації шкідливого програмного забезпечення у межах дипломної роботи було використано відкритий датасет Malware PE Dataset for Machine Learning Research, розміщений у репозиторії Mendeley Data [35]. Dataset містить набір характеристик PE-файлів операційної системи Windows та інформацію про класи ШПЗ, що дозволяє використовувати його для задач багатокласової класифікації malware.

Dataset сформовано на основі аналізу Portable Executable (PE) файлів, що є стандартним форматом виконуваних програм у середовищі Windows. У наборі даних містяться ознаки PE-заголовків, які широко застосовуються у задачах статичного аналізу шкідливого програмного забезпечення [5], [9], [16].

Основною перевагою використаного dataset є наявність структурованих числових ознак, що дозволяють застосовувати методи машинного навчання без необхідності додаткового аналізу бінарного коду виконуваних файлів.

Dataset містить:

- унікальний ідентифікатор файлу (SHA256);
- тип шкідливого програмного забезпечення (Malware_Type);
- набір характеристик PE-заголовків;
- параметри структури виконуваного файлу;
- інформацію про секції та службові характеристики PE-файлу.

У процесі реалізації системи використовувалися ознаки, що характеризують структуру виконуваного файлу, зокрема:

- TimeDateStamp;
- SizeOfImage;
- AddressOfEntryPoint;
- SizeOfCode;
- SizeOfInitializedData;
- NumberOfSections;
- Characteristics;
- DllCharacteristics;
- Subsystem;
- ImageBase та інші.

Загалом dataset містив 54 стовпці, серед яких:

- 1 службове поле (SHA256);
- 1 поле типу malware (Malware_Type);
- 52 числові ознаки PE-файлів.

У межах роботи поле SHA256 не використовувалося для навчання моделі, оскільки воно виконує лише функцію ідентифікації файлу та не впливає на процес класифікації.

Для реалізації задачі багатокласової класифікації використовувалися такі класи ШПЗ:

- Spyware;
- Ransomware;
- Trojan;
- Downloader;
- Generic Malware.

Текстові назви класів було автоматично перетворено у числові мітки за допомогою алгоритму LabelEncoder.

На Рис. 3.12 наведено фрагмент dataset після завантаження у середовище Python.

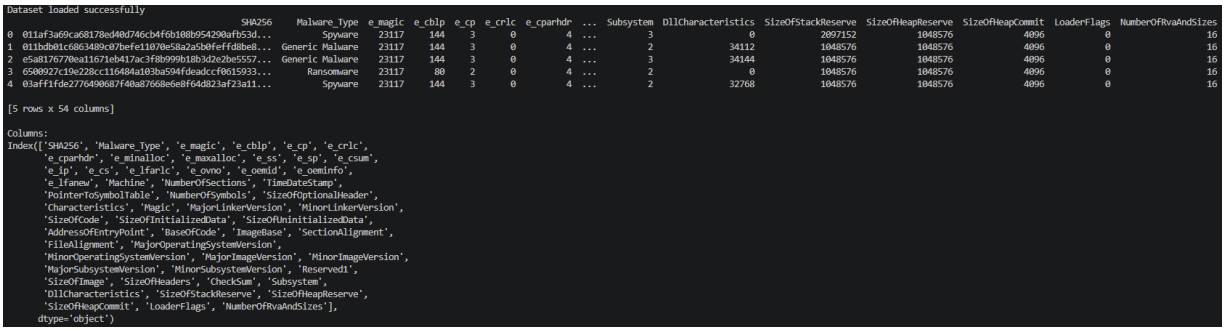

Рис 3.12. Фрагмент датасету.

В табл. 3.1 наведено приклади основних ознак dataset.

Таблиця 3.1.

Основні ознаки dataset

Ознака	**Опис**
TimeDateStamp	Часова мітка компіляції PE-файлу
SizeOfImage	Загальний розмір PE-файлу у пам’яті
AddressOfEntryPoint	Адреса точки входу програми
SizeOfCode	Розмір програмного коду
NumberOfSections	Кількість секцій PE-файлу
Characteristics	Характеристики PE-заголовка
DllCharacteristics	Атрибути DLL та механізми захисту
Subsystem	Тип підсистеми Windows
ImageBase	Базова адреса завантаження файлу

Під час попередньої обробки dataset було виконано:

- видалення службових полів;
- обробку пропущених значень;
- перетворення ознак у числовий формат;
- формування цільової змінної.

Для оцінювання ефективності системи dataset було поділено на навчальну та тестову вибірки у співвідношенні 80% до 20%. Розподіл виконувався із

застосуванням параметра stratify, що дозволило зберегти співвідношення класів у вибірках.

Використаний dataset є придатним для задач класифікації шкідливого програмного забезпечення, оскільки містить достатню кількість ознак PE-файлів та дозволяє реалізувати багатокласову класифікацію malware із застосуванням методів машинного навчання.

3.4. Методика проведення експерименту

Для перевірки ефективності розробленої системи класифікації шкідливого програмного забезпечення було проведено експериментальне дослідження із використанням методів машинного навчання. Основною метою експерименту є оцінювання здатності моделі Random Forest виконувати багатокласову класифікацію malware на основі характеристик PE-файлів.

Експеримент проводився у середовищі Python із використанням бібліотек pandas, scikit-learn, NumPy, matplotlib та seaborn. Для навчання та тестування моделі використовувався dataset Windows PE Malware Classification Dataset, який містить набір ознак PE-заголовків виконуваних файлів та інформацію про тип ШПЗ [16], [21].

На першому етапі експерименту було виконано завантаження dataset та попередню обробку даних. Обробка включала:

- видалення службових полів;
- очищення dataset від пропущених значень;
- перетворення ознак у числовий формат;
- формування цільової змінної;
- кодування класів malware у числові значення.

Для забезпечення коректного навчання моделі набір даних було поділено на навчальну та тестову вибірки. У роботі використовувалося співвідношення:

- 80% – навчальна вибірка;
- 20% – тестова вибірка.

Поділ dataset здійснювався за допомогою функції train_test_split() із параметром stratify, що забезпечує збереження співвідношення класів у вибірках (Рис 3.6).

Для реалізації класифікації було використано алгоритм Random Forest, який належить до ансамблевих методів машинного навчання та базується на побудові множини дерев рішень [16], [22]. Даний алгоритм було обрано через його високу стійкість до перенавчання, можливість роботи з великою кількістю ознак та здатність оцінювати важливість характеристик dataset.

У процесі експерименту використовувалися такі параметри моделі (Рис 3.7):

- n_estimators=500 – кількість дерев рішень;
- max_depth=20 – максимальна глибина дерева;
- min_samples_split=5 – мінімальна кількість зразків для поділу вузла;
- min_samples_leaf=2 – мінімальна кількість зразків у листі дерева.

Навчання моделі здійснювалося методом fit(), після чого виконувалося прогнозування результатів класифікації для тестової вибірки (Рис. 3.8).

Для оцінювання якості класифікації використовувалися такі метрики:

- Accuracy;
- Precision;
- Recall;
- F1-score.

Використання декількох метрик дозволяє більш об’єктивно оцінити ефективність системи в умовах багатокласової класифікації malware (Рис. 3.10) [25].

Окрім оцінювання метрик класифікації, у системі було реалізовано механізм визначення важливості ознак. Це дозволяє встановити, які характеристики PE-файлів мають найбільший вплив на процес класифікації ШПЗ (Рис. 3.11).

Для детального аналізу результатів класифікації також використовувалася матриця помилок (Confusion Matrix), яка дозволяє оцінити кількість правильних та помилкових класифікацій для кожного класу malware (Рис. 3.13).

```
cm = confusion_matrix(y_test, y_pred)
```

Рис 3.13. Матриця помилок.

Для візуального аналізу результатів використовувалися бібліотеки matplotlib та seaborn, за допомогою яких будувалися:

- графік важливості ознак;
- матриця помилок (Confusion Matrix).

Побудова графіків здійснювалася за допомогою бібліотек matplotlib та seaborn, а результати автоматично зберігалися у директорії «results».

3.5. Результати експериментів

В ході експериментального дослідження було проведено навчання та тестування моделі Random Forest для задачі багатокласової класифікації шкідливого програмного забезпечення на основі характеристик PE-файлів. Експеримент виконувався із використанням dataset Windows PE Malware Classification Dataset та реалізованої програмної системи, описаної у попередніх підрозділах.

Після завершення навчання моделі було виконано оцінювання якості класифікації за допомогою метрик Accuracy, Precision, Recall та F1-score. Отримані результати наведено у табл. 3.2.

Таблиця 3.2.

Результати класифікації malware за допомогою Random Forest

Метрика	**Значення**
Accuracy	0.396
Precision	0.367
Recall	0.396
F1-score	0.352

Отримані результати демонструють, що система здатна виконувати багатокласову класифікацію ШПЗ за характеристиками PE-файлів. Значення Accuracy на рівні 39,6% свідчить про те, що система коректно визначає значну частину класів malware, однак складність задачі багатокласової класифікації та схожість ознак різних сімейств шкідливого програмного забезпечення негативно впливають на загальну точність системи.

Для детальнішого аналізу результатів класифікації було побудовано матрицю помилок (Confusion Matrix) (Рис. 3.14).

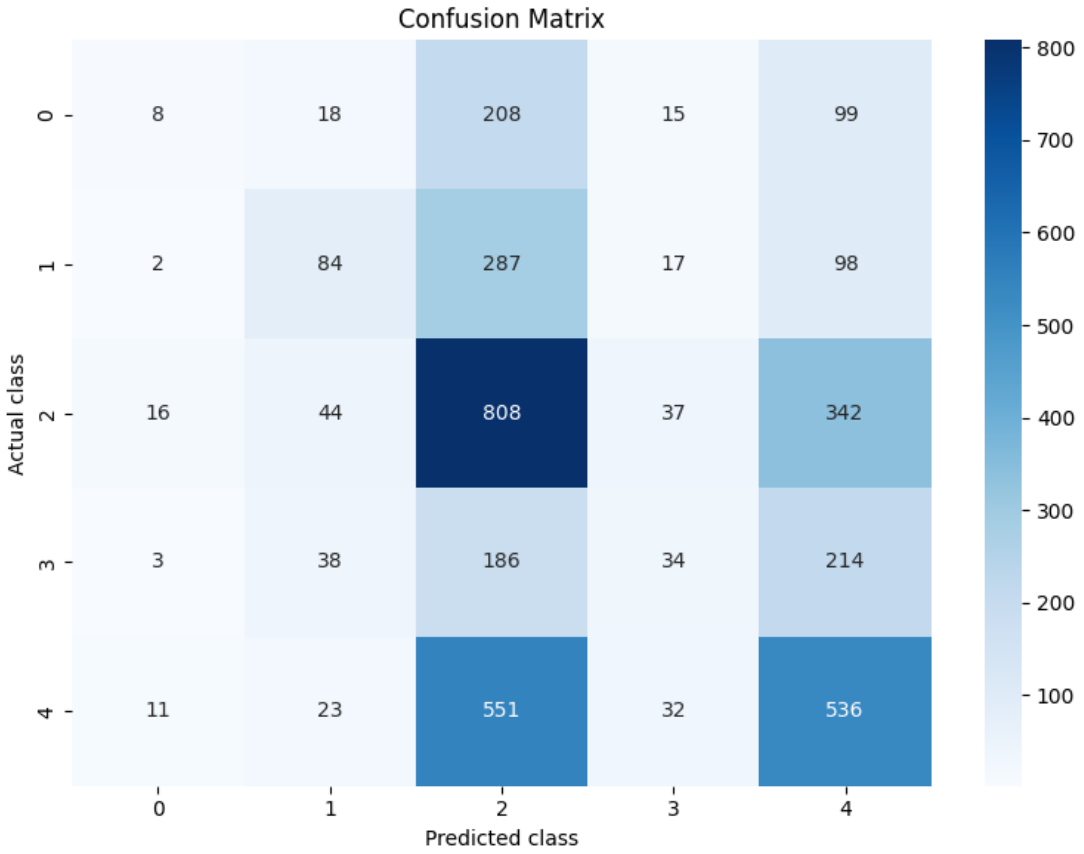

Рис 3.14. Матриця помилок.

У матриці помилок відображено кількість правильних та помилкових класифікацій для кожного класу malware. Аналіз отриманих результатів показує, що модель демонструє кращу точність для класів, які мають більшу кількість навчальних прикладів у dataset. Водночас між окремими типами ШПЗ спостерігається значна кількість помилкових класифікацій, що пояснюється схожістю структурних характеристик PE-файлів.

Під час експерименту також було виконано оцінювання важливості ознак моделі Random Forest (Рис. 3.15).

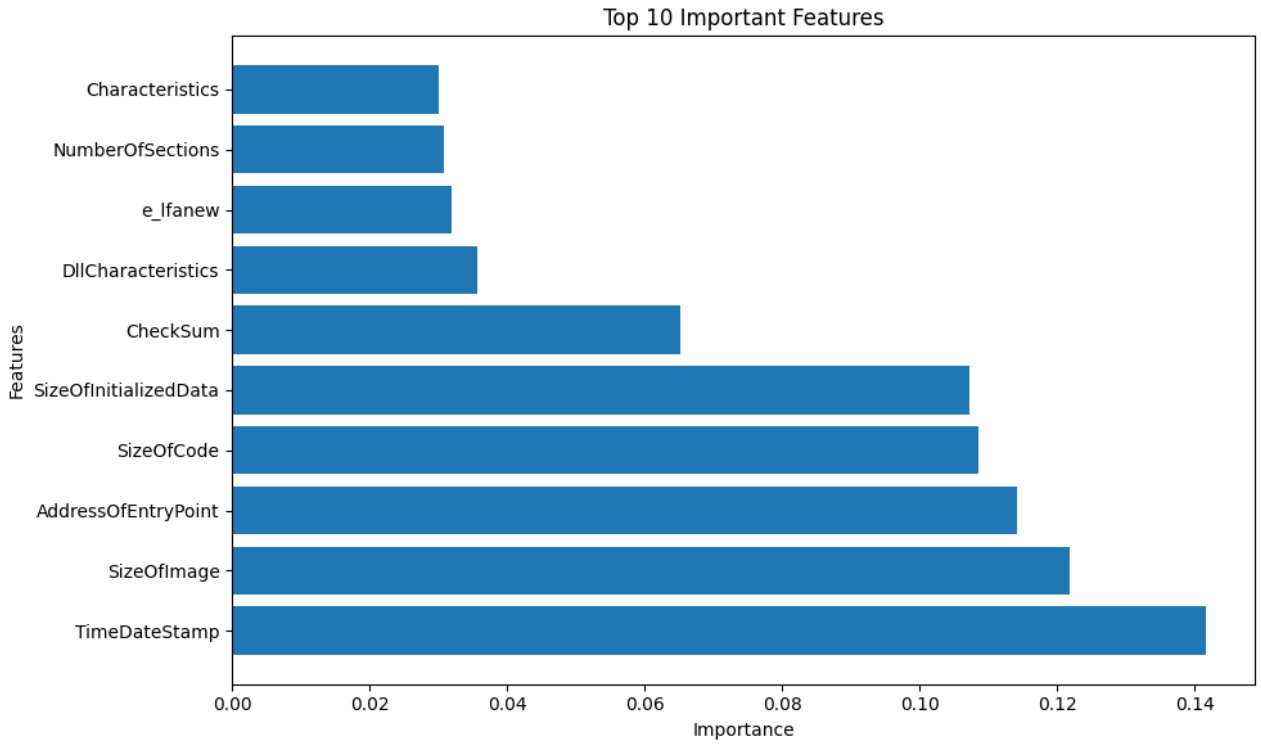

Рис. 3.15. Оцінювання важливості ознак моделі.

За результатами аналізу встановлено, що найбільший вплив на процес класифікації мають такі ознаки:

- TimeDateStamp;
- SizeOfImage;
- AddressOfEntryPoint;
- SizeOfCode;
- SizeOfInitializedData.

Наведені характеристики описують структуру виконуваного файлу та параметри його PE-заголовка, що підтверджує доцільність використання методів статичного аналізу для задач виявлення шкідливого програмного забезпечення.

На Рис. 3.15 показано, що ознака TimeDateStamp має найбільший коефіцієнт важливості серед усіх характеристик dataset. Це може пояснюватися

тим, що різні сімейства malware часто мають характерні шаблони компіляції та структури виконуваних файлів.

Крім цього, високий рівень важливості демонструють ознаки SizeOfImage та AddressOfEntryPoint, які характеризують структуру PE-файлу та особливості організації програмного коду. Отримані результати узгоджуються з сучасними дослідженнями у сфері статичного аналізу шкідливого програмного забезпечення [5], [16].

У процесі експерименту також було встановлено, що використання ансамблевого алгоритму Random Forest дозволяє ефективно працювати з великою кількістю ознак та забезпечує стабільність результатів класифікації навіть за наявності складної багатокласової структури dataset.

Результати експериментального дослідження підтверджують працездатність розробленої системи та можливість застосування методів машинного навчання для автоматизованої класифікації ШПЗ.

3.6. Порівняння з існуючими підходами

Для оцінювання ефективності розробленої системи було виконано порівняння отриманих результатів із сучасними підходами до класифікації шкідливого програмного забезпечення, що використовують методи машинного навчання та статичного аналізу PE-файлів.

У сучасних дослідженнях для задач виявлення та класифікації malware найчастіше застосовуються [16], [18], [22]:

- Decision Tree;
- Random Forest;
- Support Vector Machine (SVM);
- Logistic Regression;
- Gradient Boosting;
- нейронні мережі та методи Deep Learning.

Більшість сучасних систем використовують статичний аналіз PE-файлів, оскільки даний підхід дозволяє виконувати класифікацію без запуску потенційно

небезпечного програмного забезпечення [5], [9]. Основними ознаками для класифікації виступають характеристики PE-заголовків, секцій виконуваного файлу, імпортованих бібліотек та параметри структури програмного коду.

В межах дипломної роботи для класифікації malware було використано алгоритм Random Forest, який продемонстрував стабільні результати в умовах багатокласової класифікації та великої кількості ознак dataset.

Для порівняння результатів у табл. 3.3 наведено характеристики найбільш поширених алгоритмів машинного навчання, що використовуються у задачах класифікації ШПЗ.

Таблиця 3.3.

Порівняння підходів до класифікації malware

Алгоритм	**Переваги**	**Недоліки**
Logistic Regression	Простота реалізації, швидке навчання	Низька ефективність для складних нелінійних залежностей
Decision Tree	Інтерпретованість результатів	Схильність до перенавчання
Random Forest	Висока стабільність, робота з великою кількістю ознак, оцінка важливості ознак	Значні обчислювальні витрати
Support Vector Machine	Висока точність для окремих задач	Висока складність налаштування
Neural Networks	Можливість роботи зі складними залежностями	Потребують великих обсягів даних та ресурсів

Проведений аналіз показує, що алгоритм Random Forest є одним із найбільш збалансованих підходів для задач класифікації ШПЗ. Даний алгоритм

забезпечує достатню точність класифікації, стійкість до шуму та можливість роботи з великою кількістю ознак [16], [22].

На відміну від Logistic Regression, яка ефективно працює переважно для лінійно роздільних даних, Random Forest дозволяє моделювати складні нелінійні залежності між ознаками PE-файлів та типами malware. Це є особливо важливим у задачах аналізу шкідливого програмного забезпечення, де характеристики різних сімейств malware можуть суттєво перетинатися.

У порівнянні з нейронними мережами та методами Deep Learning запропонований підхід має менші вимоги до обчислювальних ресурсів та простіший процес навчання моделі. Крім цього, Random Forest дозволяє виконувати оцінювання важливості ознак, що є важливим для аналізу структури PE-файлів та пояснення результатів класифікації.

Водночас сучасні підходи на основі Deep Learning здатні забезпечувати вищу точність класифікації, особливо при використанні великих dataset та глибоких нейронних мереж [18]. Однак реалізація таких систем потребує значних обчислювальних ресурсів, використання GPU та складнішої процедури навчання моделей.

У межах проведеного дослідження отримані результати підтверджують, що використання Random Forest разом із ознаками PE-заголовків дозволяє реалізувати працездатну систему класифікації malware із помірними вимогами до ресурсів та прийнятною якістю класифікації.

Таким чином, запропонований підхід є доцільним для реалізації систем автоматизованого аналізу шкідливого програмного забезпечення та може бути використаний як основа для подальшого вдосконалення систем виявлення malware.

3.7. Аналіз результатів та помилок

Проведене експериментальне дослідження дозволило оцінити ефективність розробленої системи класифікації ШПЗ та проаналізувати основні фактори, що впливають на якість роботи моделі ML.

За результатами експерименту модель Random Forest продемонструвала такі показники:

- Accuracy – 39,6%;
- Precision – 36,7%;
- Recall – 39,6%;
- F1-score – 35,2%.

Отримані результати свідчать про те, що система здатна виконувати багатокласову класифікацію шкідливого програмного забезпечення на основі ознак PE-файлів. Водночас рівень точності моделі залишається помірним, що пояснюється складністю задачі класифікації malware та особливостями використаного dataset.

Однією з основних причин зниження точності є схожість характеристик різних сімейств ШПЗ. Багато сучасних malware мають подібну структуру PE-заголовків, використовують однакові механізми упаковки коду та мають схожі параметри виконуваних файлів [5], [16]. Через це система може помилково відносити зразки одного класу до іншого.

Аналіз матриці помилок показав, що найбільша кількість помилок виникає між класами, які мають близькі структурні характеристики PE-файлів. Це свідчить про те, що використання лише статичних ознак PE-заголовків не завжди дозволяє повністю відокремити окремі типи malware.

Крім цього, на якість класифікації впливає дисбаланс dataset. Деякі класи шкідливого програмного забезпечення містять значно більшу кількість зразків, ніж інші. У результаті модель краще навчається на більш представлених класах та демонструє нижчу точність для менш поширених типів malware.

Під час аналізу важливості ознак було встановлено, що найбільший вплив на класифікацію мають:

- TimeDateStamp;
- SizeOfImage;
- AddressOfEntryPoint;
- SizeOfCode;
- SizeOfInitializedData.

Це свідчить про те, що характеристики структури PE-файлу дійсно містять корисну інформацію для задач класифікації ШПЗ. Водночас використання лише PE-ознак обмежує можливості системи, оскільки сучасне malware часто застосовує механізми обфускації та модифікації структури виконуваних файлів [9].

Незважаючи на виявлені обмеження, використання алгоритму Random Forest дозволило забезпечити:

- стабільність результатів класифікації;
- стійкість до перенавчання;
- можливість роботи з великою кількістю ознак;
- оцінювання важливості характеристик dataset.

Перевагою реалізованої системи також є відносно невисокі вимоги до обчислювальних ресурсів у порівнянні з методами Deep Learning. Це дозволяє використовувати запропонований підхід у системах автоматизованого аналізу malware без необхідності застосування високопродуктивного апаратного забезпечення.

Для підвищення точності класифікації у подальших дослідженнях доцільно:

- використовувати більші та більш збалансовані dataset;
- комбінувати методи статичного та динамічного аналізу;
- застосовувати методи глибокого навчання;
- виконувати додатковий відбір найбільш інформативних ознак;
- використовувати методи балансування класів dataset.

Таким чином, результати проведеного дослідження підтверджують працездатність запропонованої системи класифікації шкідливого програмного

забезпечення та демонструють можливість застосування методів машинного навчання для автоматизованого аналізу malware. Отримані результати також показують перспективність подальшого розвитку систем виявлення шкідливого програмного забезпечення на основі комбінування різних підходів машинного навчання та аналізу виконуваних файлів.

Висновки до розділу 3

В третьому розділі дипломної роботи було реалізовано програмну систему класифікації ШПЗ із застосуванням методів машинного навчання та проведено експериментальне дослідження ефективності запропонованого підходу.

У межах роботи реалізовано повний цикл обробки даних, що включає завантаження dataset, попередню обробку ознак, формування навчальної та тестової вибірок, навчання моделі ML та оцінювання результатів класифікації. Реалізація системи виконувалася мовою програмування Python із використанням бібліотек pandas, scikit-learn, NumPy, matplotlib та seaborn.

Для проведення експериментів було використано dataset Windows PE Malware Classification Dataset, що містить характеристики PE-файлів та інформацію про типи шкідливого програмного забезпечення. У процесі попередньої обробки dataset було виконано очищення даних, видалення службових полів та перетворення категоріальних ознак у числовий формат.

Як основний алгоритм класифікації було використано Random Forest, який продемонстрував стабільність роботи та можливість ефективної обробки великої кількості ознак. Проведене експериментальне дослідження показало, що система здатна виконувати багатокласову класифікацію malware на основі характеристик PE-файлів.

За результатами тестування отримано такі показники ефективності:

- Accuracy – 39,6%;
- Precision – 36,7%;
- Recall – 39,6%;
- F1-score – 35,2%.

В ході аналізу результатів було встановлено, що найбільший вплив на процес класифікації мають ознаки TimeDateStamp, SizeOfImage, AddressOfEntryPoint, SizeOfCode та SizeOfInitializedData. Це підтверджує ефективність використання характеристик PE-заголовків для задач аналізу ШПЗ.

Також було проведено аналіз матриці помилок та визначено основні причини зниження точності класифікації, серед яких:

- схожість характеристик різних сімейств malware;
- дисбаланс dataset;
- обмеження статичного аналізу PE-файлів;
- використання лише структурних ознак виконуваних файлів.

Порівняння запропонованого підходу з існуючими методами показало, що використання алгоритму Random Forest забезпечує прийнятну якість класифікації при відносно невисоких вимогах до обчислювальних ресурсів.

Отримані результати підтверджують працездатність розробленої системи та можливість застосування методів машинного навчання для автоматизованої класифікації шкідливого програмного забезпечення. Проведене дослідження також показало перспективність подальшого вдосконалення систем виявлення malware шляхом комбінування методів статичного та динамічного аналізу, використання більш збалансованих dataset та застосування сучасних методів глибокого навчання.

ВИСНОВКИ

У результаті проведеного дослідження було реалізовано систему аналізу та класифікації шкідливого програмного забезпечення на основі методів машинного навчання, здатну автоматично виконувати обробку характеристик PE-файлів та визначати типи malware. Основною розв'язаною інженерно-технічною задачею стало поєднання методів статичного аналізу виконуваних файлів із алгоритмами машинного навчання в межах єдиної програмної системи.

У ході дослідження було проаналізовано сучасний стан проблеми поширення ШПЗ, розглянуто основні типи malware, методи їх аналізу та підходи до автоматизованої класифікації. Проведений аналіз дозволив обґрунтувати доцільність використання методів машинного навчання для задач виявлення та класифікації шкідливого програмного забезпечення.

У межах аналітичного етапу було досліджено dataset Windows PE Malware Classification Dataset, виконано попередню обробку даних, очищення dataset, перетворення категоріальних ознак та формування навчальної і тестової вибірок. Також було визначено найбільш інформативні характеристики PE-файлів, які впливають на якість класифікації malware.

В роботі проведено порівняння сучасних алгоритмів ML, що використовуються у задачах класифікації ШПЗ. За результатами аналізу для реалізації системи було обрано алгоритм Random Forest, який забезпечує стабільність класифікації, можливість роботи з великою кількістю ознак та стійкість до перенавчання.

Розроблена система реалізує повний цикл обробки даних: завантаження dataset, підготовку ознак, навчання моделі, прогнозування результатів класифікації, оцінювання ефективності та візуалізацію результатів. Реалізацію системи виконано мовою програмування Python із використанням бібліотек pandas, NumPy, scikit-learn, matplotlib та seaborn.

За результатами експериментального дослідження отримано такі показники ефективності системи: Accuracy = 0,396; Precision = 0,367; Recall =

0,396; F1-score = 0,352. Аналіз матриці помилок показав, що система здатна виконувати багатокласову класифікацію malware, однак складність задачі та схожість характеристик різних сімейств ШПЗ впливають на загальну точність класифікації.

У процесі дослідження було встановлено, що найбільший вплив на результати класифікації мають ознаки TimeDateStamp, SizeOfImage, AddressOfEntryPoint, SizeOfCode та SizeOfInitializedData, які характеризують структуру PE-файлів та особливості організації програмного коду.

Отримані результати підтверджують можливість практичного застосування методів машинного навчання для автоматизованого аналізу та класифікації шкідливого програмного забезпечення. Розроблена система може бути використана як основа для створення програмних засобів моніторингу інформаційної безпеки та систем автоматизованого виявлення malware.

Перспективами подальшого розвитку роботи є використання методів глибокого навчання, комбінування статичного та динамічного аналізу, застосування більш масштабних dataset та вдосконалення методів відбору ознак для підвищення точності класифікації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/328760930_A_Survey_on_Malware_Analysis_Techniques_Static_Dynamic_Hybrid_and_Memory_Analysis

2. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1808.01201

3. Machine learning and malware detection : a systematic literature review / S. Mahdavifar та ін. // Journal of Big Data. 2018. – Режим доступу: https://link.springer.com/article/10.1186/s13673-018-0125-x

4. Survey of Machine Learning Techniques in Malware Analysis [Електронний ресурс]. – Режим доступу: https://ouci.dntb.gov.ua/en/works/7nO3ZvP4/

5. Machine Learning and Deep Learning Methods for Cybersecurity // Computers & Security. 2020. – Режим доступу: https://www.sciencedirect.com/science/article/pii/S0167404820302707

6. Malware Detection Using Machine Learning Algorithms // IEEE Xplore [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/8851294

7. A Survey of Cyber Security Using Machine Learning // Applied Sciences. 2020. – Режим доступу: https://www.mdpi.com/2076-3417/10/19/7012

8. Machine Learning-Based Malware Detection Techniques // Computer Networks. 2021. – Режим доступу: https://www.sciencedirect.com/science/article/pii/S1389128621000960

9. VirusTotal [Електронний ресурс]. – Режим доступу: https://www.virustotal.com/gui/home/upload

10. Cuckoo Sandbox Documentation [Електронний ресурс]. – Режим доступу: https://cuckoosandbox.org/

11. ANY.RUN Interactive Malware Sandbox [Електронний ресурс]. – Режим доступу: https://any.run/

12. Hybrid Analysis [Електронний ресурс]. – Режим доступу: https://www.hybrid-analysis.com/

13. Microsoft Malware Classification Challenge Dataset [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/c/malware-classification

14. Методи виявлення шкідливого програмного забезпечення [Електронний ресурс]. – Режим доступу: https://ela.kpi.ua/server/api/core/bitstreams/0f1d7b8a-5b7d-4b2d-9f37-08c5a5cf6a6e/content

15. Сучасні методи аналізу шкідливого програмного забезпечення [Електронний ресурс]. – Режим доступу: https://er.nau.edu.ua/handle/NAU/54872

16. Breiman L. Random Forests // Machine Learning. 2001. Vol. 45, № 1. P. 5–32. – Режим доступу: https://link.springer.com/article/10.1023/A:1010933404324

17. Support Vector Machines and Applications [Електронний ресурс]. – Режим доступу: https://link.springer.com/book/10.1007/978-3-319-02300-7

18. Deep Learning for Cybersecurity Applications // Procedia Computer Science. 2020. – Режим доступу: https://www.sciencedirect.com/science/article/pii/S1877050920304865

19. Malware Detection Using Static Analysis and Machine Learning Techniques // IEEE Xplore [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/9312600

20. PE File Analysis for Malware Detection [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/357255382_PE_File_Analysis_for_Malware_Detection

21. McKinney W. Python for Data Analysis. Sebastopol : O’Reilly Media, 2022. – Режим доступу: https://wesmckinney.com/book/

22. Scikit-learn Documentation [Електронний ресурс]. – Режим доступу: https://scikit-learn.org/stable/

23. GitHub Documentation [Електронний ресурс]. – Режим доступу: https://docs.github.com/en

24. Python Malware Analysis [Електронний ресурс]. – Режим доступу: https://www.packtpub.com/en-us/product/python-malware-analysis-9781801814623

25. Feature Engineering for Machine Learning / Alice Zheng, Amanda Casari. Sebastopol : O'Reilly Media, 2018. – Режим доступу: https://www.oreilly.com/library/view/feature-engineering-for/9781491953235/

26. Structure of a PE file [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/figure/Structure-of-a-PE-file_fig8_350722779

27. Overall structure of the proposed malware classification framework [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/figure/Overall-structure-of-the-proposed-malware-classification-framework_fig1_363851325

28. Proposed Architecture for Feature Extraction of Malware [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/figure/Proposed-Architecture-for-Feature-Extraction-of-malware_fig1_316621142

29. Understanding Random Forest: A Comprehensive Guide [Електронний ресурс]. – Режим доступу: https://medium.com/@lomashbhuva/understanding-random-forest-a-comprehensive-guide-ab83fbc998e3

30. Malware Detection Based on Machine Learning Techniques // Symmetry. 2022. – Режим доступу: https://www.mdpi.com/2073-8994/14/11/2304

31. Train Test Split in Machine Learning [Електронний ресурс]. – Режим доступу: https://sharpsight.ai/blog/scikit-train_test_split/

32. Architecture of Malware Detection Classification Approach [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/figure/Architecture-of-malware-detection-classification-approach_fig1_348977684

33. PE Format Documentation [Електронний ресурс]. – Режим доступу: https://onlyf8.com/pe-format

34. Diagram of the Malware Analysis Methodology [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/figure/Diagram-of-the-malware-analysis-methodology_fig3_339342077

35. Malware PE Dataset for Machine Learning Research [Електронний ресурс] // Mendeley Data. – Режим доступу: https://data.mendeley.com/datasets/vnj7sxkt53/1

36. Система аналізу та класифікації шкідливого програмного забезпечення з використанням методів машинного навчання Лозова І. Л., Афанасьєв І. О. - - Тези конференції 25.02.2026 Стратегії кіберстійкості: управління ризиками та безперервність бізнесу: матеріали всеукр. наук.-практ. конф., м. Київ, 25 лютого 2026 р. С. 139-141 - Україна

ДОДАТКИ

ДОДАТОК А

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    confusion_matrix,
    classification_report
)



data = pd.read_csv("malware_dataset.csv")

print("Dataset loaded successfully")

print(data.head())

print("\nColumns:")
print(data.columns)
```

```
data = data.drop(columns=["SHA256"], errors="ignore")


from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

data["Label"] = encoder.fit_transform(data["Malware_Type"])


data = data.drop(columns=["Malware_Type"], errors="ignore")


data = data.dropna()

data = data.apply(pd.to_numeric, errors='coerce')

data = data.dropna()


X = data.drop("Label", axis=1)

y = data["Label"]
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42,
    stratify=y
)



model = RandomForestClassifier(
    n_estimators=500,
    max_depth=20,
    min_samples_split=5,
    min_samples_leaf=2,
    random_state=42,
    n_jobs=-1
)

model.fit(X_train, y_train)

print("\nModel training completed")



y_pred = model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print("\nModel Evaluation:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)



cm = confusion_matrix(y_test, y_pred)

print("\nConfusion Matrix:")
print(cm)



feature_importance = model.feature_importances_

importance_df = pd.DataFrame({
    "Feature": X.columns,
    "Importance": feature_importance
})

importance_df = importance_df.sort_values(
    by="Importance",
```

```
    ascending=False
)

print("\nTop Features:")
print(importance_df.head(10))



plt.figure(figsize=(10, 6))

plt.barh(
    importance_df["Feature"][:10],
    importance_df["Importance"][:10]
)

plt.xlabel("Importance")
plt.ylabel("Features")
plt.title("Top 10 Important Features")

plt.tight_layout()

plt.savefig("results/feature_importance.png")

plt.show()

print("\nGraph saved successfully")

import seaborn as sns

plt.figure(figsize=(8, 6))
```

```
sns.heatmap(
    cm,
    annot=True,
    fmt='d',
    cmap='Blues'
)

plt.xlabel("Predicted class")
plt.ylabel("Actual class")
plt.title("Confusion Matrix")

plt.tight_layout()

plt.savefig("results/confusion_matrix.png")

plt.show()
```