МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Кваліфікаційна наукова
праця на правах рукопису

МЄШКОВ ВАДИМ ІГОРОВИЧ

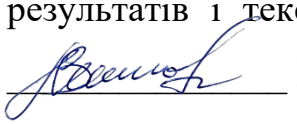
УДК 004.77:004.056

ДИСЕРТАЦІЯ
ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ ТРАФІКУ КОМП'ЮТЕРНОЇ МЕРЕЖІ ДЛЯ СИСТЕМ ВИЯВЛЕННЯ АТАК

122 Комп'ютерні науки
12 Інформаційні технології

Подається на здобуття

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

 В.І. Мєшков

Науковий керівник: Корнієнко Валерій Іванович, доктор технічних наук, професор

Дніпро – 2026

АНОТАЦІЯ

Мєшков В.І. Інформаційна технологія інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття ступеня доктора філософії за спеціальністю 122 Комп'ютерні науки галузі знань 12 Інформаційні технології. – Національний технічний університет «Дніпровська політехніка», Дніпро, 2026.

Метою дисертаційної роботи є розв'язання актуальної науково-прикладної задачі підвищення ефективності виявлення атак і розпізнавання їх типів у комп'ютерних мережах шляхом розроблення інформаційної технології інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак (СВА), що базується на методах машинного навчання та забезпечує функціонування в бінарному і багатокласовому режимах аналізу.

Актуальність теми зумовлена зростанням кількості та складності кіберзагроз, збільшенням обсягів мережевого трафіку, динамічністю його характеристик, а також обмеженістю традиційних сигнатурних методів під час виявлення нових, модифікованих і малопоширених атак. У зв'язку з цим важливого значення набуває застосування методів машинного навчання, які дають змогу виявляти статистичні закономірності в мережевому трафіку та формувати рішення щодо наявності атакувальної активності і типу виявленої загрози.

У вступі обґрунтовано актуальність теми дисертаційного дослідження, сформульовано мету і завдання роботи, визначено об'єкт, предмет і методи дослідження, розкрито наукову новизну та практичне значення одержаних результатів.

У першому розділі проаналізовано сучасні методи та засоби моніторингу мережевого трафіку й виявлення атак, розглянуто класифікацію систем IDS/IPS, особливості сигнатурних, аномальних і гібридних методів виявлення загроз. Узагальнено особливості застосування методів машинного навчання в задачах аналізу мережевого трафіку, охарактеризовано сучасні набори даних для

дослідження систем виявлення атак (СВА) та обґрунтовано вибір набору CIC-IDS2017 як репрезентативної основи для проведення відтворюваного експериментального дослідження. Показано, що ефективність інтелектуального виявлення атак визначається не лише вибором моделі машинного навчання, а й якістю підготовки даних, коректністю формування ознакового простору та обґрунтованістю системи оцінювання результатів.

У другому розділі сформульовано постановку задачі інтелектуального виявлення атак у мережевому трафіку та визначено методичні засади її розв'язання в межах задач машинного навчання. Задачу дослідження розглянуто у двох взаємопов'язаних постановках: бінарній класифікації, спрямованій на відокремлення нормального мережевого трафіку від атакувального, та багатокласовій класифікації, орієнтованій на розпізнавання конкретних типів атак. Охарактеризовано набір даних CIC-IDS2017 як експериментальну основу дослідження, визначено його структуру, особливості класів трафіку та підходи до формування вибірок для подальшого навчання й тестування моделей. Розроблено метод підготовки даних, класифікації та оцінювання мережевого трафіку для систем виявлення атак. У межах методу передбачено послідовне виконання процедур очищення, уніфікації та стандартизації даних, аналізу ознак і зниження розмірності простору даних, формування та балансування вибірок для бінарної і багатокласової класифікації. Обґрунтовано вибір моделей машинного навчання для задач виявлення атак і розпізнавання їх типів, а також визначено систему метрик оцінювання якості моделей і порядок організації експериментального дослідження. Запропоновано узагальнений алгоритм методу, який формалізує послідовність етапів обробки мережевого трафіку, побудови класифікаційних моделей та оцінювання їх ефективності, що створює методичну основу для подальшої розробки інформаційної технології інтелектуального моніторингу трафіку комп'ютерної мережі для СВА.

У третьому розділі розроблено інформаційну технологію інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак та представлено її програмну реалізацію. Визначено архітектуру інформаційної

технології, описано конвеєр обробки мережевого трафіку та обґрунтовано двоетапну схему класифікації, яка передбачає послідовне розв’язання задач бінарного виявлення атак і багатокласового розпізнавання їх типів. Сформульовано мету й завдання програмної реалізації, обґрунтовано вибір засобів програмування, бібліотек машинного навчання та середовища розроблення, а також визначено структуру програмного модуля і послідовність обробки трафіку комп’ютерної мережі. У межах програмної реалізації виконано завантаження, інтеграцію та первинний аналіз окремих частин набору даних CIC-IDS2017, реалізовано процедури попередньої обробки даних, зокрема видалення дублікатів, обробку пропущених і нескінченних значень, уніфікацію структури ознак, формування узгодженого ознакового простору, стандартизацію числових характеристик і підготовку даних до подальшого аналізу. Окремо реалізовано формування вибірки для бінарного виявлення атак, а також формування та балансування вибірки для багатокласового розпізнавання типів атак. Розроблено програмну реалізацію моделей машинного навчання для бінарного та багатокласового аналізу мережевого трафіку, організовано процедури оцінювання якості класифікації та візуалізації отриманих результатів. Особливістю розробленої інформаційної технології є поєднання в єдиному програмному модулі всіх основних етапів інтелектуального моніторингу: від завантаження й попередньої обробки мережевого трафіку до навчання моделей, оцінювання результатів і підготовки технології до інтеграції в систему виявлення атак. Запропонована реалізація забезпечує відтворюваність експериментального дослідження, підтримує два взаємопов’язані режими класифікації та може бути використана як основа для побудови інтелектуальних компонентів IDS у комп’ютерних мережах.

У четвертому розділі наведено результати експериментального дослідження ефективності розробленої інформаційної технології. Експерименти виконано на основі набору даних CIC-IDS2017 у двох постановках задачі: бінарному виявленні атак і багатокласовому розпізнаванні їх типів. Для оцінювання моделей використано фіксований поділ вибірок на навчальну та тестову частини, п’ятикратну перехресну перевірку та систему взаємодоповнювальних метрик,

зокрема загальна точність (accuracy), precision, повнота (recall), F1-score, ROC-AUC, Precision–Recall-криві та матриці помилок. У результаті експериментального дослідження встановлено, що в бінарній постановці найефективнішою є конфігурація методу опорних векторів, яка забезпечує найкращі показники виявлення атакувального трафіку. У багатокласовій постановці найвищу точність продемонструвала модель k-найближчих сусідів (k-nearest neighbors, k-NN), тоді як найкращу узагальнювальну здатність показала модель випадкового лісу. Отримані результати підтверджують ефективність запропонованої інформаційної технології та її придатність для використання в системах виявлення атак у комп'ютерних мережах.

Наукова новизна одержаних результатів полягає в тому, що вперше розроблено інформаційну технологію інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак, яка, на відміну від існуючих науково-технічних рішень, забезпечує в межах єдиного програмного модуля повний цикл підготовки даних мережевого трафіку, формування ознакового простору, побудови вибірок, балансування, навчання моделей машинного навчання та комплексного оцінювання результатів для двох взаємопов'язаних постановок задачі – бінарного виявлення атак і багатокласового розпізнавання їх типів; удосконалено метод підготовки даних, класифікації та оцінювання мережевого трафіку для систем виявлення атак, який відрізняється інтегрованим поєднанням процедур попередньої обробки даних, аналізу та зниження розмірності ознакового простору, формування й балансування вибірок, навчання моделей машинного навчання та оцінювання їх якості, що забезпечує підвищення ефективності підготовки даних і розв'язання задач бінарного виявлення атак та багатокласового розпізнавання їх типів; набули подальшого розвитку методичні положення комплексного оцінювання ефективності моделей машинного навчання в задачах моніторингу мережевого трафіку, які відрізняються уніфікованим поєднанням покласових метрик якості класифікації, ROC-AUC, Precision–Recall-кривих, матриць помилок і п'ятикратної перехресної перевірки та дають змогу визначати ефективні конфігурації моделей для двох режимів функціонування інформаційної

технології: первинного бінарного виявлення атак і подальшого багатокласового розпізнавання їх типів.

Практичне значення одержаних результатів полягає в розробці програмної реалізації інформаційної технології інтелектуального моніторингу трафіку, що може бути використана як основа для побудови інтелектуальних компонентів сучасних систем виявлення атак у комп'ютерних мережах. Її застосування забезпечує автоматизований аналіз мережевого трафіку, своєчасне виявлення атакувальної активності та подальшу деталізацію інцидентів за типами атак, що підвищує ефективність підтримки прийняття рішень у сфері кібербезпеки.

Ключові слова: кібербезпека, машинне навчання, система виявлення атак, мережевий трафік, мережеві загрози, вразливість, протокол передачі даних, Python, програмне забезпечення, інформаційна система, великі дані, нейронні мережі, прогнозування, гарантоздатність, чисельна оцінка.

ABSTRACT

Mieshkov, V.I. Information Technology for Intelligent Monitoring of Computer Network Traffic for Intrusion Detection Systems. – Qualifying scientific thesis in manuscript form.

Dissertation for the degree of Doctor of Philosophy in the specialty 122 Computer Science, field of knowledge 12 Information Technology. – Dnipro University of Technology, Dnipro, 2026.

The aim of this dissertation is to address a pressing scientific-applied problem of improving the efficiency of attack detection and classification in computer networks by developing an information technology for intelligent monitoring of computer network traffic for attack detection systems (ADS), based on machine learning methods and capable of operating in both binary and multi-class analysis modes.

The relevance of this topic stems from the growing number and complexity of cyber threats, the increase in network traffic volumes, the dynamic nature of its characteristics, and the limitations of traditional signature-based methods in detecting new, modified, and rare attacks. In this regard, the use of machine learning methods becomes crucial, as they enable the identification of statistical patterns in network traffic and the formulation of decisions regarding the presence of attack activity and the type of detected threat.

The introduction justifies the relevance of the dissertation topic, formulates the purpose and objectives of the study, defines the object, subject, and methods of the research, and highlights the scientific novelty and practical significance of the findings.

The first chapter analyzes modern methods and tools for monitoring network traffic and detecting attacks, and examines the classification of IDS/IPS systems, as well as the characteristics of signature-based, anomaly-based, and hybrid threat detection methods. It summarizes the features of applying machine learning methods to network traffic analysis tasks, describes modern datasets for studying intrusion detection systems (IDS), and justifies the selection of the CIC-IDS2017 dataset as a representative basis for conducting a reproducible experimental study. It is shown that the effectiveness of

intelligent attack detection is determined not only by the choice of a machine learning model, but also by the quality of data preparation, the correctness of feature space formation, and the soundness of the result evaluation system.

The second chapter formulates the problem of intelligent attack detection in network traffic and defines the methodological foundations for solving it within the framework of machine learning. The research problem is considered in two interrelated formulations: binary classification, aimed at distinguishing normal network traffic from attack traffic, and multi-class classification, focused on recognizing specific types of attacks. The CIC-IDS2017 dataset is characterized as the experimental basis of the study; its structure, the characteristics of traffic classes, and approaches to forming samples for subsequent training and testing of models are defined. A method for data preparation, classification, and evaluation of network traffic for attack detection systems has been developed. The method involves the sequential execution of procedures for data cleaning, unification, and standardization; feature analysis and dimensionality reduction of the data space; and the formation and balancing of samples for binary and multi-class classification. The selection of machine learning models for attack detection and type recognition tasks is justified, and a system of metrics for evaluating model quality and the procedure for organizing experimental research are defined. A generalized algorithm for the method is proposed, which formalizes the sequence of steps involved in processing network traffic, building classification models, and evaluating their effectiveness, thereby establishing a methodological foundation for the further development of information technology for intelligent monitoring of computer network traffic for IDS.

The third chapter describes an information technology for intelligent monitoring of computer network traffic for intrusion detection systems and presents its software implementation. The architecture of the information technology is defined, the network traffic processing pipeline is described, and a two-stage classification scheme is justified, which involves the sequential solution of binary attack detection tasks and multi-class recognition of their types. The goals and objectives of the software implementation are formulated, the choice of programming tools, machine learning libraries, and development environment is justified, and the structure of the software module and the

sequence of computer network traffic processing are defined. Within the scope of the software implementation, the loading, integration, and initial analysis of individual parts of the CIC-IDS2017 dataset were performed; data preprocessing procedures were implemented, including the removal of duplicates, the handling of missing and infinite values, the unification of feature structures, the formation of a harmonized feature space, standardization of numerical characteristics, and preparation of data for further analysis. Separately, a sample was formed for binary attack detection, as well as the formation and balancing of a sample for multi-class attack type recognition. A software implementation of machine learning models for binary and multi-class analysis of network traffic has been developed, and procedures for evaluating classification quality and visualizing the results have been established. A distinctive feature of the developed information technology is the combination of all the main stages of intelligent monitoring into a single software module: from loading and preprocessing network traffic to training models, evaluating results, and preparing the technology for integration into an intrusion detection system. The proposed implementation ensures the reproducibility of the experimental study, supports two interrelated classification modes, and can be used as a basis for building intelligent IDS components in computer networks.

The fourth chapter presents the results of an experimental study on the effectiveness of the developed information technology. The experiments were conducted using the CIC-IDS2017 dataset for two problem formulations: binary attack detection and multi-class classification of attack types. To evaluate the models, a fixed division of the samples into training and test sets, five-fold cross-validation, and a system of complementary metrics were used, including accuracy, precision, recall, F1-score, ROC-AUC, Precision–Recall curves, and confusion matrices. The experimental study found that in the binary setting, the most effective configuration is the support vector machine method, which provides the best performance in detecting malicious traffic. In the multi-class setting, the k-nearest neighbors (k-NN) model demonstrated the highest accuracy, while the random forest model showed the best generalization ability. The results confirm the effectiveness of the proposed information technology and its suitability for use in attack detection systems in computer networks.

The scientific novelty of the obtained results lies in the fact that, for the first time, an information technology for intelligent monitoring of computer network traffic has been developed for attack detection systems; unlike existing scientific and technical solutions, this technology provides, within a single software module, a complete cycle of network traffic data preparation, feature space formation, sample construction, balancing, training of machine learning models, and comprehensive evaluation of results for two interrelated problem formulations-binary attack detection and multi-class recognition of attack types; The method for data preparation, classification, and evaluation of network traffic for attack detection systems has been improved; it is distinguished by an integrated combination of procedures for data preprocessing, analysis, and feature space dimensionality reduction, sample formation and balancing, training of machine learning models, and evaluation of their quality, which ensures improved efficiency in data preparation and the solution of problems related to binary attack detection and multi-class recognition of attack types; the methodological principles for comprehensively evaluating the performance of machine learning models in network traffic monitoring tasks have been further developed; these tasks are characterized by a unified combination of per-class classification quality metrics, ROC-AUC, Precision–Recall curves, error matrices, and five-fold cross-validation, and enable the identification of effective model configurations for two modes of information technology operation: initial binary attack detection and subsequent multi-class recognition of their types.

The practical significance of the results obtained lies in the development of a software implementation of intelligent traffic monitoring technology, which can serve as a basis for building intelligent components of modern attack detection systems in computer networks. Its application provides automated analysis of network traffic, timely detection of attack activity, and further categorization of incidents by attack type, which enhances the effectiveness of decision-making support in the field of cybersecurity.

Keywords: cybersecurity, machine learning, intrusion detection system, network traffic, network threats, vulnerability, data transmission protocol, Python, software, information system, Big Data, neural network, forecasting, reliability, numerical estimation.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковані основні наукові результати дисертації.

Публікації у фахових виданнях України:

1. Мєшков, В. (2025). Методи машинного навчання для бінарної та багатокласової класифікації мережевого трафіку у системах виявлення атак. Таврійський науковий вісник. Серія: Технічні науки. 2025, Т. 1, № 4. С. 182-196. DOI: https://doi.org/10.32782/tnv-tech.2025.4.1.20, URL: https://journals.ksauniv.ks.ua/index.php/tech/article/view/1064/977 (Фаховий (категорія Б)).

2. Мєшков, В. (2025). Метод попередньої обробки даних для створення інформаційних моделей в інтелектуальних системах виявлення атак. Information Technology: Computer Science, Software Engineering and Cyber Security, 2025, №2. С. 108-114, DOI: https://doi.org/10.32782/IT/2025-2-11, URL: https://journals.politehnica.dp.ua/index.php/it/article/view/825/742 (Фаховий (категорія Б)).

3. Сафаров, О., Корнієнко, В., Горєв, В., Мєшков, В. (2024). Підвищення кібербезпеки електронних комунікаційних систем медичного призначення. // Information Technology: Computer Science, Software Engineering and Cyber Security, 2024, № 2, С. 128-133, DOI: https://doi.org/10.32782/IT/2024-2-16, URL: https://journals.politehnica.dp.ua/index.php/it/article/view/595/527 (Фаховий (категорія Б)).

4. Мєшков, В. (2023). Аналіз систем інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак. // Information Technology: Computer Science, Software Engineering and Cyber Security, 2023, № 1, С. 85-92, DOI: https://doi.org/10.32782/IT/2023-1-11, URL: https://journals.politehnica.dp.ua/index.php/it/article/view/262/233 (Фаховий (категорія Б)).

Наукові праці, які засвідчують апробацію матеріалів дисертації.

Матеріали конференцій та тези доповідей:

1. Мєшков В. І. Архітектура інформаційної технології інтелектуального моніторингу мережевого трафіку. // ITSec: Безпека інформаційних технологій: матеріали XIV Міжнар. наук.-техн. конф., м. Тернопіль, 22-24 трав. 2025 р. Тернопіль-Київ: ЗУНУ-ДУІКТ, 2025. 243с.

2. Мєшков В. І. Аналіз кореляційної матриці для показників набору даних CSE-CIC-IDS2017. // XII Міжнародна науково-технічна конференція студентів, аспірантів та молодих вчених «Молодь: наука та інновації», Дніпро, 13-15 листопада 2024 року (у 3-х томах) / Національний технічний університет «Дніпровська політехніка» – Дніпро: НТУ «ДП», 2024. Том 2. 291с.

3. V.I. Mieshkov, I. Mamuzić. Promising directions of computer network traffic monitoring for intrusion detection systems // 17th International Symposium of Croatian Metallurgical Society „Materials and Metallurgy“, SHMD ´2024. Materials And Metallurgy. Book Of Abstracts, Zagreb, Croatia: 2024, April, 18-19, С. 319.

4. Мєшков В. І. Аналіз наборів даних мережевого трафіку для систем виявлення атак // I (VII) міжнародна науково-практична конференція здобувачів вищої освіти і молодих учених «Інформаційні технології: теорія і практика», Дніпро, 20-22 березня 2024 року / Національний технічний університет «Дніпровська політехніка» – Дніпро: НТУ «ДП», 2024, С. 281-285.

5. Мєшков В. І. Сучасні системи моніторингу трафіку в комп’ютерній мережі // XIII Міжнародна науково-технічна конференція студентів, аспірантів та молодих вчених «Наукова весна», Дніпро, 1-3 березня 2023 року / Національний технічний університет «Дніпровська політехніка» – Дніпро: НТУ «ДП», 2023. С. 183-186.

6. Мєшков В. І., Корнієнко В. І. Ідентифікація та прогнозування трафіку комп’ютерної мережі для систем виявлення атак // XIII Всеукраїнська науково-практична конференція здобувачів вищої освіти і молодих учених «Молоді вчені 2023 - від теорії до практики»: Матеріали. Електронне видання. – Дніпро, Журфонд, 2023. С. 164-169.

7. Мєшков В. І., Корнієнко В. І. Розробка інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак // XIV Всеукраїнська науково-практична конференція «Актуальні проблеми управляння інформаційною безпекою держави», – Київ, 30 березня 2023, / Національна академія Служби безпеки України, Матеріали конференції, 2023. С. 294-298.

ЗМІСТ

с.

ВСТУП

Обґрунтування вибору теми дослідження. Сучасні комп'ютерні мережі є критично важливою складовою інформаційної інфраструктури підприємств, установ і організацій, а їх функціонування супроводжується постійним зростанням кількості та складності кіберзагроз. Умови цифровізації, розширення мережевих сервісів, збільшення обсягів передаваних даних і підвищення інтенсивності мережевих взаємодій зумовлюють необхідність удосконалення методів і засобів моніторингу мережевого середовища та своєчасного виявлення ознак атак. У таких умовах мережевий трафік виступає одним із найбільш інформативних джерел даних для виявлення атакувальної активності, оскільки відображає взаємодії між компонентами мережі та дає змогу фіксувати як явні, так і приховані ознаки кіберзагроз.

Традиційні системи виявлення атак, що базуються переважно на сигнатурних механізмах, залишаються ефективними для розпізнавання відомих шаблонів загроз, проте виявляються недостатньо гнучкими у випадках нових, модифікованих або малопоширених атак. Це знижує їхню результативність у сучасному мережевому середовищі, де атакувальна активність постійно еволюціонує, а характеристики трафіку змінюються динамічно. Саме тому одним із перспективних напрямів розвитку систем виявлення атак є застосування інтелектуальних методів аналізу даних, заснованих на машинному навчанні, які дають змогу виявляти статистичні закономірності у високовимірному просторі ознак мережевого трафіку та формувати класифікаційні рішення щодо наявності атакувальної активності.

Разом із тим ефективність інтелектуального виявлення атак визначається не лише вибором конкретної моделі машинного навчання, а й якістю підготовки даних, коректністю формування ознакового простору, урахуванням дисбалансу класів, зменшенням надлишковості ознак і дотриманням методично коректної процедури оцінювання результатів. Для мережевого трафіку характерні пропущені та некоректні значення, дубльовані записи, істотні відмінності у масштабах числових характеристик, висока корельованість ознак і нерівномірність

представлення класів, що безпосередньо впливає на здатність моделей до узагальнення. У зв'язку з цим постає потреба не лише в побудові окремих класифікаційних моделей, а в розробленні цілісної інформаційної технології, яка інтегрує всі етапи обробки мережевого трафіку: від формування підготовлених вибірок до навчання моделей і комплексного оцінювання їх ефективності.

Особливої актуальності набуває побудова такої інформаційної технології для двох взаємопов'язаних постановок задачі: бінарного виявлення атак, коли необхідно швидко відокремити нормальний трафік від атакувального, та багатокласового розпізнавання, коли важливо визначити конкретний тип атаки для подальшого реагування на інцидент. Реалізація двох режимів аналізу в межах єдиної інформаційної технології дозволяє поєднати функції первинного виявлення атакувальної активності з її детальнішою інтерпретацією, що підвищує практичну цінність результатів дослідження для систем виявлення атак у комп'ютерних мережах.

Для проведення дослідження доцільним є використання сучасного репрезентативного набору даних, придатного для відтворюваного експериментального аналізу. У роботі таким набором обрано CIC-IDS2017, який містить описи мережевих потоків нормального та атакувального трафіку, охоплює кілька категорій атак і дозволяє досліджувати задачу виявлення атак як у бінарній, так і в багатокласовій постановках. Використання цього набору даних створює основу для методично коректного порівняння моделей машинного навчання та побудови відтворюваного експериментального конвеєра.

Відтак, вибір теми дисертаційного дослідження зумовлений практичною потребою у підвищенні ефективності виявлення атак у комп'ютерних мережах та науковою необхідністю розроблення інформаційної технології інтелектуального моніторингу мережевого трафіку, яка забезпечує методично коректну підготовку даних, підтримує бінарний і багатокласовий режими класифікації, а також дозволяє підвищити достовірність і відтворюваність результатів функціонування систем виявлення атак.

Зв’язок роботи з науковими програмами, планами, темами. Дисертаційну роботу виконано в Національному технічному університеті «Дніпровська політехніка» відповідно до плану науково-дослідних робіт кафедри безпеки інформації та телекомунікацій у межах науково-дослідної роботи «Інтелектуальні методи моделювання процесів в інформаційно-телекомунікаційних системах критичної інфраструктури» (державний обліковий номер 0225U004731, дата реєстрації 17.12.2025, термін виконання етапу: 01.2024–12.2025). У межах зазначеної науково-дослідної роботи автором виконано дослідження, пов’язані з розробленням інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак, підготовкою даних мережевого трафіку, побудовою моделей машинного навчання та експериментальним оцінюванням їх ефективності.

Мета і завдання дослідження. Метою роботи є розв’язання актуальної науково-прикладної задачі підвищення ефективності виявлення атак і розпізнавання їх типів у комп’ютерних мережах шляхом розроблення інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак (СВА), що базується на методах машинного навчання та забезпечує функціонування в бінарному і багатокласовому режимах аналізу.

Для досягнення поставленої мети в роботі сформульовано такі завдання:

– проаналізувати сучасні методи та засоби моніторингу мережевого трафіку та виявлення атак, класифікацію систем IDS/IPS, а також методи машинного навчання, що застосовуються в задачах інтелектуального аналізу мережевого трафіку;

– обґрунтувати вибір набору даних для проведення дослідження та визначити вимоги до підготовки даних мережевого трафіку для задач інтелектуального виявлення атак;

– розробити метод підготовки даних, класифікації та оцінювання мережевого трафіку для систем виявлення атак;

– сформувати вибірки для бінарного виявлення атак і багатокласового

розпізнавання їх типів, а також реалізувати процедури балансування даних для підвищення коректності навчання моделей;

– розробити архітектуру інформаційної технології інтелектуального моніторингу трафіку комп'ютерної мережі для СВА, яка передбачає послідовну обробку мережевого трафіку, формування ознакового простору, навчання моделей машинного навчання та оцінювання результатів;

– розробити програмну реалізацію інформаційної технології інтелектуального моніторингу трафіку комп'ютерної мережі, що об'єднує етапи підготовки даних, формування ознакового простору, навчання моделей машинного навчання та оцінювання результатів;

– реалізувати та дослідити моделі машинного навчання для бінарної класифікації мережевого трафіку і багатокласового розпізнавання типів атак;

– провести експериментальне дослідження ефективності розробленої інформаційної технології на основі набору даних CIC-IDS2017 із використанням системи взаємодоповнювальних метрик оцінювання;

– оцінити придатність розробленої інформаційної технології для використання як основи побудови інтелектуальних компонентів систем виявлення атак у комп'ютерних мережах.

Об'єкт дослідження – моніторинг трафіку комп'ютерної мережі в системах виявлення атак.

Предмет дослідження – методи, моделі та засоби побудови інформаційної технології інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак.

Методи дослідження.

Теоретичний аналіз – вивчення наукової літератури, публікацій і сучасних науково-технічних рішень у сфері моніторингу мережевого трафіку, виявлення атак, класифікації систем IDS/IPS та застосування методів машинного навчання в задачах кібербезпеки; системний аналіз – використано для формування структури інформаційної технології інтелектуального моніторингу трафіку комп'ютерної мережі, визначення взаємозв'язків між етапами підготовки даних, побудови

моделей та оцінювання результатів; статистичний аналіз – застосовано для дослідження набору даних CIC-IDS2017, виявлення пропущених, нескінченних і дубльованих значень, аналізу розподілів ознак, кореляційних залежностей та оцінювання репрезентативності вибірок; методи попередньої обробки даних – використано для очищення даних, уніфікації структури ознак, заповнення пропущених значень, стандартизації числових характеристик, зменшення надлишковості ознак і зниження розмірності ознакового простору; методи машинного навчання – застосовано для аналізу та класифікації мережевого трафіку. Для бінарного виявлення атак використано логістичну регресію та метод опорних векторів, а для багатокласового розпізнавання типів атак – випадковий ліс, дерево рішень і метод k-найближчих сусідів; методи балансування даних – використано для усунення дисбалансу класів у вибірках, зокрема випадкове зменшення класу більшості для бінарної класифікації та метод SMOTE (Synthetic Minority Over-sampling Technique) для багатокласового розпізнавання типів атак, що дало змогу підвищити коректність навчання моделей машинного навчання; експериментальний метод – застосовано для перевірки ефективності розробленої інформаційної технології в умовах бінарної та багатокласової класифікації мережевого трафіку; методи оцінювання якості моделей – використано для аналізу результатів класифікації за допомогою показників accuracy, precision, recall, F1-score, ROC-AUC, Precision–Recall-кривих, матриць помилок і перехресної перевірки; методи візуалізації даних – застосовано для графічного подання результатів аналізу даних, кореляційної структури ознак, матриць помилок і порівняння ефективності моделей.

Наукова новизна одержаних результатів

вперше розроблено інформаційну технологію інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак, яка, на відміну від існуючих науково-технічних рішень, забезпечує в межах єдиного програмного модуля повний цикл підготовки даних мережевого трафіку, формування ознакового простору, побудови вибірок, балансування, навчання моделей машинного навчання та комплексного оцінювання результатів для двох

взаємопов’язаних постановок задачі – бінарного виявлення атак і багатокласового розпізнавання їх типів;

удосконалено метод підготовки даних, класифікації та оцінювання мережевого трафіку для систем виявлення атак, який відрізняється інтегрованим поєднанням процедур попередньої обробки даних, аналізу та зниження розмірності ознакового простору, формування й балансування вибірок, навчання моделей машинного навчання та оцінювання їх якості, що забезпечує підвищення ефективності підготовки даних і розв’язання задач бінарного виявлення атак та багатокласового розпізнавання їх типів;

набули подальшого розвитку методичні положення комплексного оцінювання ефективності моделей машинного навчання в задачах моніторингу мережевого трафіку, які відрізняються уніфікованим поєднанням покласових метрик якості класифікації, ROC-AUC, Precision–Recall-кривих, матриць помилок і п’ятикратної перехресної перевірки та дають змогу визначати ефективні конфігурації моделей для двох режимів функціонування інформаційної технології: первинного бінарного виявлення атак і подальшого багатокласового розпізнавання їх типів.

Практичне значення одержаних результатів.

Практичне значення одержаних результатів полягає у розробці програмної реалізації інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі, яка може бути використане як основа для побудови інтелектуальних компонентів сучасних систем виявлення атак. Запропонована інформаційна технологія забезпечує автоматизований аналіз мережевого трафіку, підготовку вхідних даних до класифікації, виявлення атакувальної активності в бінарному режимі та розпізнавання типів атак у багатокласовому режимі, що підвищує ефективність підтримки прийняття рішень у сфері кібербезпеки.

Практична цінність результатів дослідження полягає також у можливості використання розробленого програмного модуля для проведення експериментальних досліджень у задачах інтелектуального моніторингу мережевого трафіку, порівняльного аналізу моделей машинного навчання та

оцінювання їх ефективності на основі набору даних CIC-IDS2017. Реалізована інформаційна технологія дозволяє відтворювати повний цикл обробки даних мережевого трафіку – від їх інтеграції, очищення, стандартизації та формування ознакового простору до навчання моделей і комплексного оцінювання результатів.

Одержані результати можуть бути використані в діяльності фахівців з кібербезпеки, під час розроблення та вдосконалення систем виявлення атак, а також у науково-дослідній та освітній діяльності закладів вищої освіти для підготовки здобувачів за спеціальностями, пов’язаними з комп’ютерними науками та кібербезпекою.

Впровадження одержаних результатів.

Результати дисертаційної роботи впроваджено в діяльність ТОВ «Центум-Д» для підвищення рівня кібербезпеки корпоративної мережі та інформаційних сервісів підприємства. У межах упровадження використано розроблені методичні положення, алгоритмічні та програмні рішення для інтелектуального моніторингу мережевого трафіку, виявлення аномальної активності, первинної класифікації інцидентів і підтримки прийняття рішень щодо реагування на кіберзагрози. Зазначені рішення інтегровано з наявними засобами захисту, журналювання та моніторингу. Факт упровадження підтверджується актом, наведеним у додатках до дисертації.

Результати дисертаційного дослідження впроваджено в навчальний процес Національного технічного університету «Дніпровська політехніка» та використано під час викладання дисциплін «Кіберзахист» для підготовки здобувачів вищої освіти за спеціальністю 125 Кібербезпека (2022 рік вступу), 125 Кібербезпека та захист інформації (2023 рік вступу). Факт упровадження підтверджується актом, наведеним у додатках до дисертації.

Особистий внесок здобувача.

Дисертаційна робота є самостійно виконаним науковим дослідженням. Усі основні наукові положення, результати, висновки та рекомендації, що виносяться на захист, отримані здобувачем особисто.

У наукових працях (фахові статті категорії Б), опублікованих одноосібно, автору належать: аналіз сучасних систем інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак; розроблення методу попередньої обробки даних для створення інформаційних моделей в інтелектуальних системах виявлення атак; дослідження методів машинного навчання для бінарної та багатокласової класифікації мережевого трафіку у системах виявлення атак; обґрунтування архітектури інформаційної технології інтелектуального моніторингу мережевого трафіку; аналіз кореляційної матриці показників набору даних CIC-IDS2017; аналіз наборів даних мережевого трафіку для систем виявлення атак; дослідження сучасних систем моніторингу трафіку в комп'ютерній мережі.

У фаховій статті, опублікованій у співавторстві з О. Сафаровим, В. Корнієнком та В. Горєвим, автору належать аналіз вразливостей електронних комунікаційних систем медичного призначення, узагальнення загроз для персональної медичної інформації та участь в обґрунтуванні програмних методів забезпечення кібербезпеки таких систем. Результати цієї публікації використано для поглиблення теоретичних положень дисертаційної роботи щодо кіберзагроз, вразливостей інформаційних систем і комплексного захисту даних.

У наукових працях (тези доповіді на конференціях), опублікованих у співавторстві, особистий внесок здобувача полягає в такому: у роботі, виконаній спільно з Корнієнком В. І., здобувачеві належать аналіз методів і засобів ідентифікації та прогнозування трафіку комп'ютерної мережі для систем виявлення атак, постановка задачі дослідження та узагальнення отриманих результатів; у праці, присвяченій розробці інформаційної технології інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак, здобувачеві належать формування основних положень дослідження, обґрунтування структури інформаційної технології та підготовка матеріалів до публікації; у роботі, опублікованій у співавторстві з І. Mamuzić, здобувачеві належать аналіз перспективних напрямів моніторингу трафіку комп'ютерної мережі для систем

виявлення атак, узагальнення методів, моделей і засобів інтелектуального виявлення загроз та підготовка основної частини матеріалів.

Матеріали й результати дисертаційного дослідження пройшли апробацію на міжнародних і всеукраїнських науково-технічних та науково-практичних конференціях. Ідеї, положення та результати, що належать співавторам опублікованих праць, у дисертації використано лише з відповідними посиланнями.

Апробація результатів роботи.

Основні положення, результати та висновки дисертаційної роботи доповідалися, обговорювалися та пройшли апробацію на міжнародних і всеукраїнських науково-технічних та науково-практичних конференціях, а саме:

– XIV Міжнародній науково-технічній конференції «ITSec: Безпека інформаційних технологій» (Тернопіль, 22-24 травня 2025 р.) / Західноукраїнський національний університет – Державний університет інформаційно-комунікаційних технологій;

– XII Міжнародній науково-технічній конференції студентів, аспірантів та молодих вчених «Молодь: наука та інновації» (Дніпро, 13-15 листопада 2024 р.) / Національний технічний університет «Дніпровська політехніка»;

– 17th International Symposium of Croatian Metallurgical Society “Materials and Metallurgy” (SHMD 2024) (Загреб, Хорватія, 18-19 квітня 2024 р.) / Zagreb: Hrvatsko metalurško društvo / ISSN 0543-5846;

– I (VII) Міжнародній науково-практичній конференції здобувачів вищої освіти і молодих учених «Інформаційні технології: теорія і практика» (Дніпро, 20-22 березня 2024 р.) / Національний технічний університет «Дніпровська політехніка»;

– XIII Всеукраїнській науково-практичній конференції здобувачів вищої освіти і молодих учених «Молоді вчені 2023 – від теорії до практики» (Дніпро, 23 березня 2023 р.) / Український державний університет науки і технологій;

– XIII Міжнародній науково-технічній конференції студентів, аспірантів та молодих вчених «Наукова весна» (Дніпро, 1-3 березня 2023 р.) / Національний технічний університет «Дніпровська політехніка»;

– XIV Всеукраїнській науково-практичній конференції «Актуальні проблеми управління інформаційною безпекою держави» (Київ, 30 березня 2023 р.) / Національна академія служби безпеки України.

Публікації. За матеріалами дисертації опубліковано 11 робіт, з яких 4 статті включено до переліку фахових видань (категорія Б), затверджених МОН України за спеціальністю дисертації, та 7 – публікації у матеріалах конференцій/симпозіумів (у тому числі 5 міжнародних).

Структура та обсяг дисертації. Повний обсяг дисертації становить 230 сторінок, з яких 190 сторінок основного тексту. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел (150 найменувань) та додатків. Робота містить 47 рисунків, 13 таблиць та 5 додатків.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ МЕРЕЖЕВОГО ТРАФІКУ ТА ВИЯВЛЕННЯ АТАК

1.1 Сучасні методи та засоби моніторингу мережевого трафіку і виявлення атак

Мережевий трафік є одним із найбільш інформативних джерел даних для виявлення кіберзагроз у корпоративних, державних і критично важливих інфраструктурах. На відміну від суто хостових подій, трафік відображає взаємодії між компонентами мережі, що дає змогу фіксувати як явні, так і приховані ознаки атак: сканування, спроби експлуатації вразливостей, аномальні шаблони обміну, командно-керувальні канали (C2), ексфільтрацію даних та атаки відмови в обслуговуванні [37-40, 54]. Тому моніторинг трафіку розглядають як базовий компонент систем виявлення атак та систем запобігання вторгненням (IDS/IPS), а також як джерело подій для систем кореляції та реагування [1, 52, 53]. У керівництві NIST SP 800-94 зазначається, що технології IDPS доцільно розрізняти за типами подій та способом розгортання (мережеві, безпровідні, аналіз поведінки мережі, хостові), оскільки це визначає як можливості детектування, так і обмеження щодо продуктивності та експлуатації [1, 148].

Традиційні системи виявлення атак історично ґрунтувалися на сигнатурному методі детектування, за якого рішення про наявність атаки формується на основі зіставлення мережевої події або характеристик трафіку з відомими шаблонами з бази сигнатур. Сильними сторонами є інтерпретованість та відносна простота супроводу для відомих загроз. Однак сигнатурні методи суттєво обмежені при появі нових, модифікованих або спеціально замаскованих атак, що знижує їхню результативність у середовищах зі швидкою еволюцією інструментарію зловмисників [1, 24, 28, 60]. Натомість аномалійні, або поведінкові, методи детектування передбачають формування моделі «нормальної» активності та виявлення відхилень від неї, що потенційно дає змогу ідентифікувати невідомі атаки. Водночас практична проблема полягає у підвищенні частоти хибних

спрацювань через природну мінливість трафіку, зміни конфігурацій сервісів, сезонність і непередбачуваність поведінки користувачів. З цієї причини сучасні рішення все частіше реалізують гібридну логіку: поєднання сигнатурного контролю з моделями інтелектуального аналізу та контекстною пріоритизацією інцидентів [1, 24, 28, 52, 60].

Окремий напрям еволюції мережевого моніторингу пов’язаний із переходом від низькорівневого аналізу пакетів до формування подій вищого рівня, зручних для подальшого аналізу та автоматизації. Показовим прикладом є концепція пасивного мережевого моніторингу, реалізована в Bro/Zeek: система орієнтована на високошвидкісне спостереження мережевого каналу, чіткий поділ механізмів і політик, та генерування детальних журналів (логів) протокольних транзакцій і артефактів, що є критично важливим для розслідування інцидентів і побудови моделей поведінки [2-4, 33, 53]. У межах дослідження такі інструменти можна розглядати як «шар отримання даних та ознак», який забезпечує основу для інтелектуального аналізу.

Подальший розвиток інтелектуального моніторингу пов’язаний із широким застосуванням методів машинного навчання (ML), що дають змогу автоматично виявляти закономірності у високовимірних просторах ознак та працювати з нелінійними залежностями. У задачах IDS використовують як бінарну класифікацію (нормальний трафік/атака), так і багатокласову (визначення конкретного типу атаки) [8, 9, 24, 57, 60]. Для бінарного розділення часто застосовують лінійні моделі як базові, а також SVM, що добре працює у задачах із нелінійними межами класів за рахунок ядерних перетворень [12]. Для багатокласової класифікації в IDS-практиці широко використовуються ансамблеві методи, зокрема Random Forest, які поєднують керованість, відносну стійкість до шуму та хорошу узагальнювальну здатність [10], а також сучасні алгоритми градієнтного бустингу (наприклад, XGBoost), що демонструють високу ефективність на табличних даних та дозволяють враховувати розрідженість і масштабованість обчислень [13]. Також актуальним залишається метод k-найближчих сусідів, який може використовуватися як інтерпретована базова

модель для порівняння результатів класифікації та перевірки якості сформованого простору ознак, хоча його обчислювальна складність може бути критичною під час обробки великих потоків даних [11]. Водночас оглядові дослідження з кібербезпекового машинного навчання підкреслюють, що реальна результативність ML-IDS визначається не лише вибором моделі, а й даними, процедурою підготовки та коректністю експериментального протоколу [8, 9, 57, 58, 61, 140, 146].

Ключовою умовою розвитку інтелектуального виявлення атак стала поява більш реалістичних наборів даних і методик формування ознак. Класичні набори даних, зокрема KDD Cup 1999, історично використовувалися як бенчмарки для оцінювання систем виявлення атак, однак у літературі детально описані їхні недоліки, серед яких значна надмірність записів і наявність зміщень. Це може призводити до завищених оцінок якості та некоректного порівняння методів, моделей і алгоритмів виявлення атак [6]. Це сприяло переходу до сучасніших наборів даних, зокрема UNSW-NB15, який пропонується як більш складний і наближений до сучасних умов, та сімейства CIC-наборів даних, орієнтованих на різноманіття сценаріїв атак та відтворюваність експериментів [6, 7, 15, 16, 55, 56, 59]. У контексті роботи особливу роль відіграє CIC-IDS2017, для якого описано принципи генерації трафіку, класи атак і характеристику ознак, що дозволяє будувати відтворювані експерименти та зіставляти результати різних методів [5, 45, 55, 59]. Подібні принципи побудови експериментального набору даних використано і в CSE-CIC-IDS2018, який містить кілька сценаріїв атак та супровідні дані, а також застосовується в дослідженнях як масштабний набір для оцінювання методів і моделей систем виявлення атак [15]. Аналітичні огляди щодо CSE-CIC-IDS2018 узагальнюють практику використання цього набору та типові проблеми його застосування, зокрема дисбаланс класів, що є корисним для обґрунтування методології дисертаційного дослідження [16, 147].

Практичні виклики інтелектуального моніторингу трафіку пов’язані передусім із якістю даних та процедурою їхньої підготовки [8, 9, 55, 57–59, 61]. У реальних мережах домінує нормальний трафік, тоді як атаки є рідкісними подіями,

що породжує дисбаланс класів. За таких умов показник accuracy може бути недостатньо інформативним; доцільно застосовувати ROC-аналіз та AUC, а також Precision-Recall представлення, яке є більш показовим саме для дисбалансних задач [18-19]. Для коректного навчання моделей часто використовують методи балансування, зокрема SMOTE, що створює синтетичні приклади міноритарного класу, підвищуючи чутливість детектора до атак; при цьому важливо виконувати балансування в межах навчальної вибірки (без «витоку даних» у тестування) [14]. Окрім дисбалансу, типовою проблемою є висока розмірність, корельованість та надлишковість ознак мережевого трафіку, що ускладнює навчання і може погіршувати узагальнювальну здатність моделей. Для зниження розмірності та усунення корельованості широко застосовують PCA – від класичних робіт до фундаментальної монографії Джолліффа, яка є стандартним джерелом для обґрунтування застосування методу головних компонент [17]. На етапі нормалізації та стандартизації числових ознак широко застосовується процедура приведення даних до нульового середнього та одиничної дисперсії, що є важливим для коректної роботи низки алгоритмів машинного навчання, зокрема SVM та k-NN. Відповідні засоби стандартизації реалізовані у практичних інструментах обробки даних, зокрема StandardScaler у бібліотеці scikit-learn [20]. Для реалізації SMOTE та споріднених методів балансування в прикладних дослідженнях широко застосовується бібліотека imbalanced-learn, документація якої фіксує параметри та коректні сценарії використання оверсемплінгу [21, 141, 145].

Таким чином, сучасні методи, моделі та засоби моніторингу мережевого трафіку та виявлення атак пройшли еволюцію від сигнатурних механізмів до інтелектуальних технологій, які поєднують засоби пасивного мережевого моніторингу та журналювання подій [2-4], рекомендації щодо побудови та експлуатації IDPS [1], сучасні набори даних для відтворюваного порівняння методів [5-7, 15-16], а також методично обґрунтовані процедури підготовки даних, зниження розмірності та балансування класів [14, 17, 20-21]. У сукупності це формує підґрунтя для розроблення інформаційної технології інтелектуального моніторингу трафіку, яка повинна забезпечувати: отримання та формування

релевантних ознак, методично коректну попередню обробку даних, вибір та навчання моделей для бінарної й багатокласової класифікації, коректне оцінювання якості з урахуванням дисбалансу та практичних вимог до систем виявлення атак [8-9, 18-19].

1.2 Класифікація систем моніторингу мережевого трафіку та виявлення атак

Системи виявлення атак (Intrusion Detection Systems, IDS) та системи запобігання атакам (Intrusion Prevention Systems, IPS) є базовими засобами забезпечення кібербезпеки, що реалізують функції безперервного спостереження за подіями та мережевим трафіком, аналізу отриманих даних і формування рішень щодо наявності ознак порушень безпеки, а також (у випадку IPS/IDPS) – ініціювання реакцій у режимі реального часу [1]. Методи та моделі виявлення вторгнень концептуально пов’язані з моделлю аналізу подій і поведінкових профілів, сформульованою Д. Деннінг, яка заклала основу для сучасних принципів побудови систем детектування та інтерпретації безпекових подій [22]. У практичній площині питання проєктування та експлуатації IDPS-рішень регламентуються методичними рекомендаціями, зокрема NIST SP 800-94, де визначено основні типи IDPS, особливості їх розгортання та вимоги до подальшого супроводу [1].

Класифікація IDS/IPS є необхідною для обґрунтування архітектури й методів інтелектуального моніторингу, оскільки реальні системи поєднують різні джерела телеметрії, алгоритмічні механізми (сигнатурні, аномалійні, гібридні), архітектурні схеми та сценарії застосування. У науковій літературі сформовано таксономії IDS, які описують систему як сукупність компонентів збору подій, аналізу, баз знань і модулів реагування, а також враховують середовище застосування і цілі детектування [23]. Узагальнювальні огляди підкреслюють, що класифікацію доцільно здійснювати одночасно за принципом детектування (misuse/сигнатури та anomaly/відхилення) і за операційними аспектами (типи даних, архітектура,

реакція, часові вимоги) [24]. При цьому для практичної придатності критично важливо враховувати «проблему базової частоти» (base-rate fallacy): навіть висока чутливість детектора може супроводжуватися неприйнятним рівнем хибнопозитивних спрацювань у середовищі, де реальні атаки є рідкісними подіями [25]. Отже, класифікаційні ознаки мають визначати не лише «що аналізує система», але й «як приймає рішення» та «якими є ризики помилок».

За місцем розгортання і об’єктом спостереження IDS/IPS поділяють на мережеві (NIDS/NIPS), хостові (HIDS/HIPS), бездротові (WIDS/WIPS), прикладні (AIDS), а також гібридні (комбіновані) рішення [1], [24]. Мережеві IDS аналізують трафік у межах мережевих сегментів або на периметрі, формуючи сповіщення на основі пакетних або потокових ознак. Типовими прикладами мережевих систем сигнатурного детектування є Snort та Suricata, які підтримують використання правил виявлення атак і можуть функціонувати як у режимі IDS, так і в режимі IPS, зокрема з inline-фільтрацією мережевого трафіку [29-31].

Хостові IDS виконують контроль подій на кінцевих вузлах (журнали ОС/додатків, контроль цілісності, поведінкові ознаки тощо); як приклад платформи, що поєднує агентний збір телеметрії та централізовану аналітику, можна навести Wazuh [32]. У корпоративних практиках функціонування центрів моніторингу безпеки поширеною є гібридна модель організації моніторингу, за якої дані мережевого спостереження доповнюються телеметрією хостів, а кореляція подій безпеки здійснюється на рівні SIEM-систем та аналітичних засобів.

За типом даних та рівнем представлення мережевої активності розрізняють пакетно-орієнтований, потоково-орієнтований і подієво-журнальний моніторинг. Пакетний аналіз забезпечує найвищу деталізацію, включно з протокольною семантикою та (за відсутності шифрування) корисним навантаженням, однак має суттєві обмеження масштабованості для високошвидкісних каналів і погіршується через повсюдне використання TLS. Потоковий моніторинг NetFlow/IPFIX передбачає агрегацію мережевого трафіку у потоки та статистичні характеристики, що зменшує обсяг оброблюваних даних і забезпечує кращу масштабованість. Водночас таке подання містить менше семантичних ознак для детектування

складних атак; відповідні механізми формалізовані стандартами IETF, зокрема IPFIX і NetFlow v9 [26, 27]. Подієво-журнальний моніторинг передбачає роботу з логами, стандартизованими, зокрема, протоколом syslog, і є важливою складовою кореляції безпекових подій [34]. Методичні рекомендації з керування журналами, а також планування лог-менеджменту подано в NIST SP 800-92 і актуалізованому проєкті SP 800-92r1 [35, 36]. Для інтелектуального моніторингу трафіку найбільш практичним є комбінування рівнів: потоки – для масштабного профілювання й первинної фільтрації, події/логи – для контексту та кореляції, пакети – для підтвердження та розслідування інцидентів.

За принципом детектування системи поділяють на сигнатурні (misuse), аномалійні (anomaly), специфікаційні (specification-based) та гібридні [1], [24]. Сигнатурний метод детектування є ефективним для виявлення відомих атак і характеризується високою інтерпретованістю результатів, однак потребує постійного оновлення бази правил та має обмеження щодо розпізнавання нових або модифікованих загроз [1]. Аномалійний метод детектування ґрунтується на виявленні відхилень від моделі «нормальної» поведінки та потенційно придатний для ідентифікації невідомих атак, проте його результативність істотно залежить від якості побудови моделі норми, наявності дрейфу даних і може супроводжуватися підвищеною часткою хибнопозитивних спрацювань [24]. У класичному огляді аномалійного NIDS систематизовано техніки (статистичні, кластеризаційні, класифікаційні, знання-орієнтовані) та підкреслено проблеми адаптивності й оцінювання якості [28]. Специфікаційні методи детектування передбачають формалізацію допустимої поведінки протоколів передачі даних і сервісів та виявлення порушень заданих специфікацій, що є корисним для контролю критичних сервісів, однак потребує трудомісткого супроводу відповідних правил і моделей поведінки. З урахуванням переваг і обмежень окремих методів сучасною тенденцією є гібридизація засобів виявлення атак, що передбачає поєднання сигнатурного детектування з інтелектуальними моделями та подальшою кореляцією подій у SIEM/NDR-системах.

За архітектурою IDS/IPS поділяють на централізовані, розподілені та ієрархічні. Централізовані схеми спрощують керування політиками та кореляцію, але можуть створювати вузькі місця продуктивності. Розподілені архітектури, що поєднують сенсори, колектори та центральну аналітику, є домінуючими в корпоративних мережах, оскільки забезпечують масштабованість і стійкість систем моніторингу. Важливою складовою мережевого моніторингу як джерела подій є інструменти Network Security Monitoring (NSM), зокрема Zeek (раніше Bro), який реалізує високошвидкісне пасивне спостереження, формує протокольно-орієнтовані журнали й забезпечує основу для подальшого інтелектуального аналізу [2-4], [33]. Така модель (телеметрія → події/ознаки → аналітика) природно відповідає задачі інтелектуального моніторингу трафіку як інформаційної технології.

За характером реакції розрізняють пасивні IDS (сповіщення/журналювання) та активні IPS/IDPS (блокування, скидання сесій, ізоляція, керування політиками доступу). Вибір між IDS і IPS визначається критичністю сервісів, ризиками помилкового блокування, а також наявністю процесів супроводу правил і реагування [1]. У прикладних системах, наприклад Suricata, IPS-режим реалізується як inline-фільтрація, де правила можуть застосовуватися для відкидання або відхилення небажаного трафіку [31]. Для інтелектуальних (ML-орієнтованих) детекторів активна реакція потребує додаткових запобіжників (порогові стратегії, багаторівневе підтвердження, контроль FN/FP), інакше ризик хибних блокувань стає неприйнятним, що узгоджується з проблемою базової частоти [25].

За часовими вимогами системи розділяють на онлайн (реальний час), near-real-time та офлайн (постаналіз). Реальний час критичний для запобігання розвитку інцидентів, однак вимагає суворих обмежень на затримки й ресурси; офлайн-аналіз корисний для глибокої аналітики, побудови моделей і ретроспективного виявлення інцидентів. Для методів машинного навчання типовою є двоконтурна схема: навчання та валідація моделей виконуються офлайн, а застосування моделей і

адаптація порогів – онлайн, з можливістю переходу в «безпечний режим», наприклад лише сповіщення без блокування, у разі нестабільності.

Окремий вимір класифікації пов’язаний із підтримкою інтелектуальних методів та оцінюванням якості. Для порівняння IDS застосовують актуальні набори даних і узгоджені метрики. Зокрема, CIC-IDS2017 використовується як референтний набір для задач детектування та класифікації мережевих атак і має опис методології формування трафіку та класів [5]. Водночас результативність моделей залежить від репрезентативності даних, дисбалансу класів і дрейфу розподілів, тому в методології досліджень необхідно визначати джерела телеметрії, протокол препроцесингу, процедури крос-валідації та метрики, які адекватно відображають помилки, критичні для IDS-сценаріїв [18, 19]. У прикладному аспекті бази знань про тактики/техніки атак (наприклад, MITRE ATT&CK) можуть використовуватися для інтерпретації детектів і узгодження аналітики з практичними сценаріями реагування [37]. Таким чином, інтелектуальний моніторинг трафіку доцільно реалізовувати як інтегровану технологію, що комбінує джерела телеметрії (пакети/потоки/логи), алгоритмічні механізми (сигнатури + інтелектуальні моделі) та коректні процедури оцінювання якості з урахуванням дисбалансу і практичних вимог до детектування атак [1, 18, 19, 25].

1.3 Методи інтелектуального аналізу трафіку в системах виявлення атак

Інтелектуальний аналіз мережевого трафіку в системах виявлення атак (IDS) розглядається як еволюційний перехід від ручного опису ознак атак у вигляді правил до автоматизованого прийняття рішень на основі статистичних закономірностей та моделей машинного навчання. У межах класичної концепції детектування вторгнень акцент робиться на спостереженні за подіями, формуванні профілів поведінки та виявленні відхилень, що було систематизовано в фундаментальній моделі D. Denning [22]. У прикладному вимірі методи інтелектуального аналізу трафіку функціонують як частина більшої екосистеми IDPS, де поєднуються механізми збору телеметрії, аналітичні модулі, засоби

журналювання та процедури реагування, відповідно до рекомендацій NIST щодо проєктування та експлуатації систем IDS/IPS [1]. Відтак сучасний інтелектуальний аналіз трафіку в IDS слід інтерпретувати не як «окремий алгоритм», а як цілісну технологічну схему «дані → ознаки → моделі → метрики → рішення → інтеграція у процеси реагування».

Практична результативність інтелектуальних методів виявлення атак значною мірою визначається типом даних, що використовуються для аналізу, та рівнем їх подання. Залежно від архітектури IDS і вимог до продуктивності можуть застосовуватися пакетний аналіз, потокові представлення NetFlow/IPFIX, а також подієво-журнальна модель формування даних, що передбачає використання протокольних журналів, системних журналів і логів безпекових подій [1, 26, 27]. Пакетний рівень забезпечує найбільшу деталізацію та потенціал виявлення складних атак, однак погано масштабується у високошвидкісних мережах і суттєво втрачає інформативність за умов повсюдного шифрування. Потоковий рівень (IPFIX/NetFlow) агрегує інформацію, зменшуючи обсяг даних і підвищуючи масштабованість, але може втрачати контекст, необхідний для детектування атак на прикладному рівні [26, 27]. Подієво-журнальна модель подання даних передбачає використання журналів подій, зокрема syslog, що забезпечує зручність кореляції безпекових подій та проведення ретроспективного аналізу. Методичні аспекти керування журналами, збирання, зберігання, аналізу та планування лог-менеджменту описані у NIST SP 800-92 та його актуалізованих матеріалах [34–36]. Для задачі інтелектуального моніторингу трафіку найбільш ефективним зазвичай є комбінування цих рівнів: потоки забезпечують масштабне профілювання, протокольні події – семантику та контекст, пакети – верифікацію та експертний аналіз інцидентів.

Суттєвий внесок у розвиток інтелектуального аналізу трафіку зробили платформи мережевого моніторингу, що перетворюють сирий трафік на події/ознаки вищого рівня. Концепція Bro/Zeek, запропонована як система реального часу для виявлення мережевих вторгнень, базується на високошвидкісному пасивному спостереженні, формуванні детальних

протокольних подій та поділі «механізмів» і «політик» аналізу [2]. У сучасній практиці Zeek використовується як компонент Network Security Monitoring (NSM), що генерує багатий набір журналів (HTTP, DNS, SSL/TLS, conn тощо) і створює основу для подальших інтелектуальних методів аналізу [3, 4, 33]. Інтелектуальний аналіз трафіку має розглядатися як надбудова над механізмами збору та структуризації даних: якість і стабільність ознак, одержаних із NSM/потоків/логів, часто визначають межу досяжної точності ML-моделей.

З точки зору методів обробки даних та прийняття рішень, інтелектуальний аналіз трафіку в IDS зазвичай реалізується у формі задач класифікації або детектування аномалій. Для класифікації характерні дві постановки: бінарна (нормальний трафік / атака) і багатокласова (визначення типу атаки). Бінарна постановка часто застосовується як первинний «фільтр», що відокремлює підозрілі події від масиву нормальної активності, тоді як багатокласова – для уточнення типу мережевих загрози та підвищення цінності результату для реагування (triage). Такий поділ узгоджується з практичними вимогами IDPS щодо пріоритизації інцидентів і зниження навантаження на аналітиків [1]. Водночас вибір постановки залежить від доступності розмітки даних, умов експлуатації (оновлюваність типів атак) та ризиків помилок, що особливо важливо з урахуванням проблеми базової частоти [25].

У супервізованому (керованому) навчанні для IDS найбільш поширеними є алгоритми, які добре працюють із табличними ознаками (потоки, статистики, агрегати) та забезпечують конкурентну якість при відносній простоті впровадження. До таких належать ансамблеві методи, зокрема Random Forest, що поєднує множину дерев рішень і демонструє високу точність та стійкість до шуму/частини нерелевантних ознак [10]. Важливим класом методів є SVM, який ефективний для задач із нелінійними межами класів завдяки ядерним перетворенням і широко застосовується в задачах бінарної класифікації трафіку [12]. Для підвищення якості та масштабованості на великих даних (наборах ознак) використовуються алгоритми градієнтного бустингу, зокрема XGBoost, орієнтований на ефективність обчислень і високу результативність на

структурованих даних [13]. Як порівняльні базові методи часто застосовують k-NN, який концептуально є простим і корисним для аналізу роздільності ознакового простору, хоча може бути обмеженим за швидкодією на великих вибірках [11]. Важливо підкреслити, що для IDS ці методи мають оцінюватися не лише за «середньою точністю», але й за здатністю утримувати прийнятний баланс помилок у режимі реальної експлуатації (особливо щодо FN/FP), що визначається специфікою домену та високою ціною пропущених атак [25].

Паралельно з керованою класифікацією значну роль відіграють аномалійні, несупервізовані або напівсупервізовані методи, які спрямовані на виявлення відхилень від профілю нормальної поведінки за відсутності повної розмітки або в умовах швидкої зміни сценаріїв атак. Класичний огляд аномалійних NIDS систематизує основні напрями досліджень: статистичні методи, кластеризацію, класифікаційні моделі, знання-орієнтовані методи, а також проблеми адаптації та оцінювання [28]. Практичний компроміс полягає в тому, що аномалійні методи можуть підвищувати здатність до виявлення нових атак, але часто збільшують частку хибнопозитивних тривог, особливо в середовищі зі змінною поведінкою користувачів і сервісів [25, 28]. Саме тому в реальних SOC/NDR-сценаріях доцільними є гібридні конфігурації, які поєднують сигнатурний бар’єр (для відомих атак) із аномалійним/ML-аналізом для виявлення нових відхилень та подальшою кореляцією [1, 23, 24].

Окремим напрямом інтелектуального аналізу трафіку є методи глибокого навчання, що націлені на автоматичне виділення представлень і здатні моделювати складні нелінійні залежності. Оглядові роботи вказують на активне застосування нейромережевих архітектур, зокрема автоенкодерів, рекурентних мереж і CNN-моделей для аналізу послідовностей, у задачах IDS, однак підкреслюють і типові проблеми: вимоги до даних, чутливість до дрейфу, складність пояснення рішень, а також ризики завищення якості через некоректні експериментальні протоколи [8]. Також зазначається, що порівняння методів і моделей має враховувати не лише абсолютні значення метрик, а й відтворюваність, переносимість у нові мережеві умови та придатність для інтеграції в реальні процеси реагування [8, 9, 109].

Незалежно від класу моделей (ML чи DL), критично важливою складовою інтелектуального аналізу трафіку є формування ознакового простору та попередня обробка даних. Мережеві дані часто містять надлишкові та корельовані ознаки, неоднорідність масштабів, пропуски, шум, а також суттєвий дисбаланс класів (переважання нормального трафіку над атаками). Для моделей, чутливих до масштабу (SVM, k-NN та ін.), необхідною є стандартизація ознак (наприклад, приведення до нульового середнього та одиничної дисперсії), що відповідає загальноприйнятим практикам обробки даних та реалізовано у стандартних інструментах, зокрема StandardScaler [20]. Для зниження розмірності та декореляції ознак поширено застосовують PCA, теоретичні основи та практичні аспекти якого систематизовано у монографії I. Jolliffe [17]. PCA дозволяє зменшити складність моделі, знизити ризик перенавчання та інколи підвищити стабільність класифікації, що особливо актуально при високовимірних описах трафіку.

Дисбаланс класів є однією з найбільш критичних проблем IDS, оскільки атаки в реальних мережах є рідкісними, і модель, оптимізована під «загальну точність», може демонструвати високу accuracy, але фактично ігнорувати атаки. Для балансування класів застосовують методи оверсемплінгу/андерсемплінгу, зокрема SMOTE, який генерує синтетичні приклади міноритарного класу [14]. Однак застосування SMOTE має виконуватися методично коректно (в межах навчальної вибірки), інакше виникає ризик «витоку інформації» та завищення показників якості; практичні аспекти використання SMOTE формалізовано у відповідній документації інструментальних засобів [21]. У межах інтелектуального аналізу трафіку важливо розглядати балансування не як «технічний прийом», а як елемент методології дослідження: процедура балансування повинна бути інтегрована у pipeline навчання/валідації та чітко задокументована.

Оцінювання інтелектуальних методів у IDS має враховувати як специфіку даних, так і практичні вимоги до системи детектування. З огляду на дисбаланс класів та різну «вартість» помилок, інформативність показника accuracy є обмеженою, тому рекомендовано використовувати метрики, що відображають компроміс між чутливістю та специфічністю, а також порогові режими роботи. Для

цього традиційно застосовують ROC-аналіз та площу під ROC-кривою (AUC), теоретично та прикладно описані в роботах із аналізу ROC [18]. Водночас для дисбалансних задач аналіз на основі Precision–Recall-кривих часто є більш інформативним, оскільки дає змогу коректніше інтерпретувати якість класифікації рідкісних подій, зокрема атакувальної активності, що обґрунтовано у спеціалізованих дослідженнях [19]. Додатково, практичні сценарії експлуатації IDS вимагають акценту на показниках, пов'язаних із хибнонегативними рішеннями (FN) і хибнопозитивними тривогами (FP), оскільки перші призводять до пропуску атак, а другі – до перевантаження процесів реагування, що прямо пов'язано з проблемою базової частоти [25]. Саме тому під час методичного обґрунтування інтелектуальних методів необхідно задавати критерії вибору порогів та оцінювати стабільність рішень у різних умовах.

Важливим елементом методології інтелектуального аналізу трафіку є вибір наборів даних та забезпечення відтворюваності експериментів. У літературі зазначається, що використання застарілих наборів даних може призводити до некоректних висновків щодо сучасних атак і реальних умов експлуатації, а також до завищення показників якості. Зокрема, для KDD Cup 1999 детально описано проблеми надмірності та зміщення, що ускладнює об'єктивне порівняння методів і моделей виявлення атак [7]. У зв'язку з цим у сучасних дослідженнях активно застосовуються більш реалістичні набори даних, серед яких особливу роль відіграє CIC-IDS2017: у базовій публікації описано методику генерації трафіку, класи атак і характеристики даних, що дає змогу виконувати відтворювані експерименти [5]. Також важливими є набори CSE-CIC-IDS2018 і супровідні ресурси, що забезпечують доступність даних для досліджень та порівнянь [15, 16]. Узагальнювальні огляди на основі цих наборів даних акцентують на типових проблемах (дисбаланс, вибір ознак, протоколи валідації) та підкреслюють необхідність коректного експериментального дизайну [9].

З позиції практичного впровадження інтелектуальні методи мають інтегруватися в архітектуру IDS/IPS та процеси моніторингу. Сигнатурні системи (Snort, Suricata) залишаються важливими як ефективний механізм детектування

відомих атак на периметрі та в мережевих сегментах; вони підтримують правила та можуть працювати в IDS/IPS режимах [29-31]. Хостові компоненти (наприклад, Wazuh) доповнюють картину подіями з кінцевих вузлів, що покращує кореляцію інцидентів [32]. Інструменти NSM (Zeek) формують протокольно-орієнтовану телеметрію та журнали, що значно підвищує якість ознак для інтелектуального аналізу [2-4], [33]. У цьому контексті інтелектуальні ML-моделі можуть функціонувати як аналітичний шар, який отримує структуровані ознаки/події та видає класифікаційне рішення, а подальше реагування визначається політиками IDPS і ризиками помилок, що узгоджується з методичними рекомендаціями NIST щодо розгортання та супроводу таких систем [1].

Окрему групу методів інтелектуального аналізу мережевого трафіку становлять нейронні мережі, які здатні виявляти складні нелінійні залежності між ознаками трафіку та класами атак. У задачах виявлення атак нейронні мережі можуть застосовуватися для бінарної класифікації трафіку, багатокласового розпізнавання типів атак, а також для виявлення аномальних шаблонів поведінки в мережі. Їх перевагою є здатність до автоматизованого узагальнення складних закономірностей у великих наборах даних (Big Data), однак практичне використання таких моделей потребує значних обчислювальних ресурсів, ретельного налаштування архітектури та контролю перенавчання.

Узагальнюючи, методи інтелектуального аналізу трафіку в IDS охоплюють широкий спектр моделей і алгоритмів – від класичних супервізованих класифікаторів, зокрема SVM, Random Forest, градієнтного бустингу та k-NN [10–13], до аномалійних і глибоких моделей [8, 28]. Однак їх практична ефективність визначається не лише вибором алгоритму, а насамперед: якістю та рівнем представлення телеметрії (пакети/потоки/логи) [26, 27, 34-36], коректністю ознакового інжинірингу та препроцесингу (стандартизація, зниження розмірності, балансування) [14, 17, 20, 21], методично коректним протоколом експериментальної валідації та метриками, чутливими до дисбалансу і практичних вимог (ROC/AUC, Precision–Recall) [18, 19], урахуванням операційних ризиків,

зокрема проблеми базової частоти, що визначає прийнятність системи у реальному SOC-середовищі [25].

1.4 Типові кіберзагрози та їх прояви в мережевому трафіку як об'єкт інтелектуального моніторингу

Ефективність інтелектуального моніторингу трафіку для систем виявлення атак визначається тим, наскільки повно модель «бачить» прояви загроз у телеметрії та наскільки коректно ці прояви перетворено на ознаки. У практиці IDPS моніторинг має охоплювати як подієві індикатори (сповіщення, журнали, правила), так і поведінкові відхилення (аномалії у потоках/сеансах/протокольних транзакціях), що узгоджується з рекомендаціями щодо виявлення та аналізу інцидентів у NIST SP 800-94 [1]. У межах моделі виявлення вторгнень, що ґрунтується на аналізі подій і профілів поведінки, базовим є принцип «норма → відхилення → інтерпретація», який концептуально обґрунтовує застосування методів машинного навчання для ідентифікації атак за сукупністю слабких ознак [22].

Сучасний ландшафт загроз (на рівні організацій/галузей) характеризується поєднанням масових кампаній та цільових атак, а також стійкою присутністю класичних векторів: компрометація облікових записів, експлуатація вразливостей у вебзастосунках, шкідливе ПЗ, DDoS, а також приховані канали керування та ексфільтрації. Узагальнені звіти щодо інцидентів і порушень безпеки (наприклад, DBIR) демонструють, що реальні атаки нерідко поєднують кілька етапів – від первинного доступу до розгортання дій у мережі та витоку даних, що підсилює актуальність багаторівневого моніторингу (потоки, протокольні журнали (логи) та події хостів) [39]. Аналітичні огляди загроз на рівні ЄС також підкреслюють динамічність і комбінований характер сучасних кіберзагроз, що ускладнює їх виявлення лише сигнатурними механізмами та обґрунтовує доцільність застосування інтелектуального моніторингу як більш гнучкого засобу детектування атак [38, 142].

Для задач роботи доцільно групувати загрози за їхнім мережевим проявом (тобто за тим, як вони відображаються у трафіку та журналах):

– розвідка та підготовчі дії (reconnaissance).

Сканування портів/служб, перебір ресурсів, виявлення топології та сервісів часто проявляються в потоці як аномально висока частота коротких з'єднань, послідовні спроби підключення до багатьох портів або адрес, нетипова кількість SYN-пакетів та помилки встановлення сесій. Такі дії є типовими для етапу підготовки атак і добре фіксуються як на пакетному рівні, так і у потокових даних (NetFlow/IPFIX) [26, 27]. Методичні положення щодо тестування та оцінювання мережевої безпеки, які охоплюють активні перевірки, зокрема сканування, систематизовані у NIST SP 800-115. Цей документ є корисним як еталонний опис типових технік тестування та їхніх наслідків для мережевих артефактів [41].

– компрометація облікових записів і brute force.

Спроби підбору паролів (SSH/RDP/вебавтентифікація) проявляються повторюваними невдалими входами, серіями коротких сесій, а також піками в логах автентифікації. На рівні трафіку це можуть бути часті однотипні запити до сервісу входу, а на рівні журналів – послідовність "failed login" з одних джерел або розподілених джерел. У практиці IDPS такі сценарії потребують кореляції: мережевий трафік, системні журнали (логи) (syslog/події ОС) та контекст доступу [34-36].

– експлуатація вразливостей у вебзастосунках та прикладні атаки.

Атаки на вебрівні (ін'єкції, некоректний контроль доступу, SSRF тощо) не завжди мають виразний «пакетний» підпис, але часто відображаються у протокольних журналах (HTTP-методи, коди відповіді, аномальні URI/параметри, повторюваність запитів) та поведінкових шаблонах. Як загальновизнаний орієнтир для класифікації критичних ризиків веббезпеки використовується OWASP Top 10, який дозволяє формалізувати групи загроз і пов'язати їх із типами телеметрії (запити, журнали (логи) застосунку, WAF/IDS події) [42].

– DDoS та інші атаки відмови в обслуговуванні.

DDoS має найбільш виразні прояви у статистиці потоків і навантаженні: аномальні обсяги трафіку, різке зростання PPS/BPS, велика кількість однотипних запитів, SYN-flood, UDP-flood тощо. Для цієї групи загроз потоковий рівень (NetFlow/IPFIX) особливо цінний завдяки масштабованості, а пакетний рівень – для атрибуції типу атаки та верифікації. Практичні рекомендації щодо розуміння типів DDoS та першочергових заходів протидії систематизовані у матеріалах CISA, що також корисно для формування класів атак і індикаторів у дослідженні [40].

– приховані канали керування та ексфільтрації (C2, tunneling).

Сучасні атаки часто використовують стандартні протоколи (DNS/HTTP(S)/TLS) як транспорт для прихованого керування або витоку даних. Через поширення шифрування дедалі важливішими стають непідписні (non-payload) ознаки: статистика довжин/частоти запитів, ентропія доменів, нетипові TTL/частота NXDOMAIN, аномальні шаблони DNS-запитів, повторюваність та часові патерни. Для виявлення DNS-тунелювання в літературі виокремлюють методи детектування на основі аналізу корисного навантаження DNS-запитів, аналізу характеристик мережевого трафіку, а також методи машинного навчання, що використовують статистичні та поведінкові ознаки [43]. Додаткові сучасні узагальнення підкреслюють, що DNS-канали часто є частиною APT-сценаріїв і потребують багаторівневої кореляції (DNS + HTTP(S) + події хоста) [44].

– багатостадійні атаки та зіставлення із TTPs.

У реальних інцидентах атаки рідко зводяться до одного мережевого патерну. Натомість спостерігається послідовність дій: розвідка → первинний доступ → закріплення → переміщення мережею → ексфільтрація або вплив. Для інтелектуального моніторингу це означає необхідність переходу від локальних індикаторів до виявлення ланцюжків подій та їх узгодження з тактиками/техніками противника [37].

З огляду на наведене, інтелектуальний моніторинг трафіку для IDS доцільно розглядати як технологію, що (а) охоплює різні рівні даних (потоки, протокольні журнали (логи), системні журнали (логи)) [26, 27, 34-36]; (б) підтримує детектування як масових атак (DDoS, brute force), так і прихованих каналів

(DNS/HTTP(S) tunneling) [40, 43, 44]; (в) використовує формалізовану класифікацію загроз і сценаріїв реагування (OWASP Top 10) [37, 42]. У підсумку саме якісне узгодження «загроза → прояв у трафіку → ознаки → модель → метрики» створює методологічну основу для розроблення авторської інформаційної технології інтелектуального моніторингу трафіку [1, 18, 19, 25, 143, 149].

1.5 Аналіз недоліків наявних методів і засобів та постановка задачі

Аналіз сучасного стану систем моніторингу мережевого трафіку та виявлення атак показує, що, попри значний розвиток IDPS-рішень, у практиці кіберзахисту зберігається низка системних обмежень, які не дозволяють забезпечити стабільно високу якість детектування в реальних умовах експлуатації. Згідно з рекомендаціями щодо проєктування та супроводу IDPS, ефективність систем виявлення визначається не лише механізмами детектування, а й джерелами телеметрії, стратегіями реакції, якістю журналювання та процедурою аналізу подій [1]. Водночас класична концепція виявлення вторгнень як аналізу подій і поведінкових профілів підкреслює, що будь-яке рішення має спиратися на адекватну модель «норми» та механізми інтерпретації відхилень [22]. У сучасних мережах, де поведінка користувачів і сервісів швидко змінюється, підтримання стабільної «норми» стає складним, що безпосередньо впливає на достовірність інтелектуальних рішень.

Першою групою обмежень є алгоритмічні та методологічні недоліки сигнатурного й аномального детектування. Сигнатурні IDS/IPS (зокрема класу Snort/Suricata) демонструють високу результативність щодо відомих атак і добре інтерпретуються, але принципово обмежені щодо нових або модифікованих загроз та технік обходу [1, 29-31]. Аномалійні методи детектування, навпаки, потенційно здатні виявляти невідомі атаки, однак характеризуються підвищеним рівнем хибнопозитивних спрацювань, що зумовлено мінливістю мережевого середовища, зміною конфігурацій, сезонністю трафіку та дрейфом розподілів даних [24, 28].

Унаслідок цього практична цінність аномалійних детекторів часто знижується через необхідність ручної верифікації великого обсягу сповіщень.

Другою, критичною для практики, є проблема «базової частоти» (base-rate fallacy): у реальних мережах частка атак є малою порівняно з обсягом нормального трафіку, тому навіть невеликий відсоток хибнопозитивних спрацювань може формувати величезний потік помилкових тривог і перевантажувати процеси реагування [25]. Це означає, що системи виявлення атак мають оптимізуватися не лише за загальною точністю, а за компромісом між пропущеними атаками (FN) і хибними спрацюваннями (FP), а також працювати в коректно налаштованих порогових режимах. Саме тому для об’єктивного оцінювання інтелектуальних моделей доцільно застосовувати ROC/AUC-аналіз та Precision–Recall-подання, які є більш інформативними для дисбалансних задач [18, 19].

Третьою групою проблем є обмеження даних та їх підготовки, що безпосередньо визначає результативність ML/DL-моделей. Дослідження з кібербезпекового ML підкреслюють, що багато невдач інтелектуальних IDS зумовлені не вибором моделі, а якістю даних, некоректним протоколом експериментів і недостатньо обґрунтованими процедурами попередньої обробки [8, 9]. Для мережевих наборів даних характерні дисбаланс класів, корельованість та надлишковість ознак, неоднорідність масштабів, пропуски, шум і потенційні дублікати. Історично це демонструє приклад KDD’99, для якого показано наявність суттєвої надмірності й зміщень, що здатні призводити до завищення оцінок якості та некоректних порівнянь методів [6]. Тому перехід до сучасних наборів даних (зокрема CIC-IDS2017 та CSE-CIC-IDS2018) є кроком уперед, однак навіть вони потребують методично коректного препроцесингу та протоколів валідації [5, 15, 16].

Четверта група обмежень стосується високої розмірності та структури ознакового простору. Мережеві ознаки часто є взаємокорельованими, а їхня кількість може бути значною, що підвищує ризик перенавчання і знижує узагальнювальну здатність моделей. Для вирішення цієї проблеми використовують методи зниження розмірності та декореляції, зокрема PCA, теоретично

обґрунтований у фундаментальних джерелах [17]. Однак на практиці ефект PCA визначається вибором кількості компонент і узгодженістю з pipeline навчання, а відсутність стандартизації або некоректне застосування перетворень може погіршити результат. Для багатьох алгоритмів (SVM, k-NN) коректна стандартизація ознак є необхідною умовою стабільного навчання; у прикладних системах це реалізується через стандартизовані інструменти обробки даних (наприклад, StandardScaler) [20].

П’ята група проблем – дисбаланс класів і коректність балансування. Для задач IDS атаки є рідкісними, тому моделі часто «зміщуються» в бік домінуючого класу. Одним із поширених методів є SMOTE, що синтезує приклади міноритарного класу і може підвищувати чутливість детектора [14]. Водночас балансування має виконуватися виключно в межах навчальної вибірки (у складі коректного pipeline) – інакше виникає витік інформації та завищення метрик, що робить результати непридатними для реальної експлуатації; методичні аспекти реалізації таких процедур підтримуються у спеціалізованих інструментальних засобах [21]. Таким чином, балансування в IDS є не «опцією», а методологічно важливою частиною інформаційної технології.

Шоста група обмежень пов’язана з архітектурними та експлуатаційними вимогами. Реальні системи моніторингу працюють у середовищі багаторівневих джерел телеметрії: пакети, потоки NetFlow/IPFIX, протокольні журнали NSM-рішень (наприклад, Zeek), системні журнали (логи) (syslog), агентна телеметрія хостів (наприклад, Wazuh) [26, 27, 32-34]. У таких умовах важливим стає узгодження форматів, стійкість до пропусків, масштабованість і затримки аналізу. Механізми лог-менеджменту та планування журналювання є ключовими для кореляції й доказовості інцидентів, що підкреслюється рекомендаціями NIST щодо керування журналами [35, 36]. Крім того, поширення шифрування зменшує доступність корисного навантаження, підвищуючи значення поведінкових та статистичних ознак, що вимагає адаптації методів інтелектуального аналізу.

Сьома група проблем стосується практичної інтерпретації результатів і прив’язки детектів до сценаріїв реагування. У реальних інцидентах атаки є

багатостадійними, тому ізольоване сповіщення без контексту має обмежену цінність. Використання баз знань про тактики й техніки противника (MITRE ATT&CK) дає змогу структурувати детекти за сценаріями (TTPs) і формувати правила кореляції для підвищення операційної корисності результатів [37]. Так само сучасні огляди загроз і статистика інцидентів підкреслюють комбінований характер атак та необхідність системного моніторингу, який поєднує дані різних рівнів і забезпечує відтворювані процедури аналізу [38, 39].

Висновки до розділу 1

Розглянуто сучасний стан розвитку систем моніторингу мережевого трафіку та виявлення атак, а також аналіз основних моделей формування телеметрії у комп’ютерних мережах, зокрема пакетний режим, потокове подання NetFlow/IPFIX та подієво-журнальні дані. Показано, що вибір типу телеметрії суттєво впливає на масштабованість системи, інформативність ознак і можливості кореляції інцидентів у багаторівневих системах моніторингу типу SOC/SIEM/NDR.

Проведено порівняльний аналіз методів детектування атак у IDS, зокрема сигнатурних, які забезпечують високу ефективність для відомих загроз, але мають обмеження щодо виявлення нових і модифікованих атак, та аномалійних, які здатні виявляти невідомі мережеві загрози, проте є чутливими до стабільності моделі «норми» і схильними до підвищеного рівня хибнопозитивних спрацювань.

Узагальнено наявні класифікації IDS/IPS за архітектурою, типом реакції та часовими вимогами. Показано, що централізована або розподілена організація системи, режим функціонування IDS/IPS та вимоги до часу реагування впливають на експлуатаційну надійність, масштабованість і ризики помилкових спрацювань.

На основі аналізу сучасних методів інтелектуальної обробки мережевого трафіку встановлено, що ефективність систем виявлення атак визначається не лише вибором алгоритму машинного навчання, а й узгодженістю всіх етапів обробки даних: формування ознак, стандартизації, зменшення розмірності, зокрема PCA, балансування класів, зокрема SMOTE, та вибору відповідних метрик оцінювання

моделей. Показано, що accuracy є недостатньо інформативною метрикою в умовах дисбалансу даних, тоді як ROC/AUC-аналіз та Precision–Recall-криві забезпечують коректніше відображення співвідношення між хибнопозитивними спрацюваннями та пропущеними атаками.

Також розглянуто проблему базової частоти, згідно з якою за низької частоти атак навіть незначна частка хибнопозитивних спрацювань може створювати надмірне навантаження на процеси реагування. Визначено необхідність використання порогових стратегій, пріоритизації та сценарної інтерпретації результатів детектування. Узагальнено основні класи загроз, зокрема розвідку, brute force-атаки, вебатаки, DDoS, C2-комунікації та ексфільтрацію даних, а також їх характерні ознаки у потоковій та журнальній телеметрії.

Таким чином, невирішеною актуальною науково-прикладною задачею є підвищення ефективності виявлення атак і розпізнавання їх типів у комп'ютерних мережах шляхом розроблення інформаційної технології інтелектуального моніторингу трафіку комп'ютерної мережі для СВА.

Для розв'язання цієї задачі необхідно вирішити такі завдання:

– обґрунтувати вибір набору даних для проведення дослідження та визначити вимоги до підготовки даних мережевого трафіку для задач інтелектуального виявлення атак;

– розробити метод підготовки даних, класифікації та оцінювання мережевого трафіку для систем виявлення атак;

– розробити архітектуру інформаційної технології інтелектуального моніторингу трафіку комп'ютерної мережі для СВА, яка передбачає послідовну обробку мережевого трафіку, формування ознакового простору, навчання моделей машинного навчання та оцінювання результатів;

– розробити програмну реалізацію інформаційної технології інтелектуального моніторингу трафіку комп'ютерної мережі, що об'єднує етапи підготовки даних, формування ознакового простору, навчання моделей машинного навчання та оцінювання результатів;

– реалізувати та дослідити моделі машинного навчання для бінарної класифікації мережевого трафіку і багатокласового розпізнавання типів атак;

– провести експериментальне дослідження ефективності розробленої інформаційної технології на основі набору даних CIC-IDS2017 із використанням системи взаємодоповнювальних метрик оцінювання;

– оцінити придатність розробленої інформаційної технології для використання як основи побудови інтелектуальних компонентів систем виявлення атак у комп’ютерних мережах.

РОЗДІЛ 2. МЕТОД ПІДГОТОВКИ ДАНИХ, КЛАСИФІКАЦІЇ ТА ОЦІНЮВАННЯ МЕРЕЖЕВОГО ТРАФІКУ ДЛЯ СИСТЕМ ВИЯВЛЕННЯ АТАК

2.1 Постановка задачі інтелектуального виявлення атак у мережевому трафіку

Сучасні комп'ютерні мережі є критично важливою складовою інформаційної інфраструктури підприємств, установ і організацій, а їх функціонування супроводжується постійним зростанням кількості та складності кіберзагроз. Розширення спектра атак, поява нових сценаріїв зловмисної активності, збільшення обсягів мережевого трафіку та підвищення динамічності його характеристик зумовлюють необхідність удосконалення методів, моделей і засобів моніторингу мережевого середовища та своєчасного виявлення ознак атак. У таких умовах традиційні сигнатурні системи виявлення атак, незважаючи на їхню ефективність у випадках відомих шаблонів загроз, виявляються недостатньо гнучкими для розпізнавання нових, модифікованих або малопоширених типів атак. Це обумовлює доцільність використання інтелектуальних методів аналізу мережевого трафіку, здатних виявляти закономірності у великих даних та приймати рішення на основі вивчених статистичних залежностей [1, 8, 9, 24, 28].

У межах дисертаційного дослідження задача інтелектуального виявлення атак розглядається як задача автоматизованої класифікації мережевого трафіку на основі методів машинного навчання [8, 9]. Її суть полягає у побудові таких інформаційних моделей, які за набором ознак мережевого потоку дозволяють визначити, чи є даний трафік нормальним, чи він містить ознаки атакувальної активності. Крім базової задачі розмежування трафіку на нормальний та атакувальний, у роботі розглядається також більш складна постановка, пов'язана з багатокласовим розпізнаванням типів атак. Реалізація двоетапної схеми аналізу є практично значущою, оскільки в реальних системах кіберзахисту недостатньо лише встановити факт наявності загрози; необхідно також визначити її характер,

що дає змогу обрати адекватну стратегію реагування, локалізації та подальшого аналізу інциденту.

Отже, у роботі досліджуються дві взаємопов’язані постановки задачі. Перша з них – бінарна класифікація, у межах якої всі спостереження поділяються на два класи: нормальний трафік і атакувальний трафік. Така постановка відповідає сценарію первинного фільтра системи виявлення атак, коли головною метою є швидке відокремлення потенційно небезпечної активності від фонової мережевої поведінки. Друга постановка – багатокласова класифікація, у межах якої виконується деталізація конкретного типу атаки. Її використання дозволяє перейти від простого факту виявлення мережевої загрози до більш глибокого аналізу природи інциденту, що підвищує прикладну цінність інформаційної технології інтелектуального моніторингу мережевого трафіку.

Розв’язання поставлених задач передбачає використання експериментальних великих даних, який містить описи мережевих потоків, представлених набором числових ознак та відповідними мітками класів. При цьому якість побудованих моделей машинного навчання визначається не лише вибором конкретного алгоритму класифікації, а значною мірою – властивостями вихідних даних і коректністю їх підготовки. В задачах інтелектуального виявлення атак дані виступають не просто джерелом навчальних прикладів, а основою формування знань моделі про структуру мережевої поведінки, характер відмінностей між нормальним і шкідливим трафіком, а також особливості окремих типів атак.

На відміну від сигнатурних методів, де ознаки атак визначаються явно через набір правил, шаблонів або евристик, моделі машинного навчання формують рішення на основі статистичних закономірностей у просторі ознак. Це означає, що будь-які дефекти даних, викривлення розподілів, надлишкові залежності або некоректні процедури підготовки безпосередньо впливають на якість навчання і здатність моделі до узагальнення. У контексті систем виявлення атак така залежність є особливо критичною, оскільки помилки класифікації можуть мати суттєві наслідки для безпеки інформаційної системи. Найнебезпечнішим типом помилки для IDS є хибнонегативне рішення, за якого атакувальна активність

класифікується як нормальна [25]. У роботі особливу увагу приділено забезпеченню таких умов підготовки даних, за яких моделі будуть не лише демонструвати високу загальну точність, а й зберігати чутливість до атак, зокрема до малопредставлених і складних для виявлення класів.

Однією з ключових вимог до даних у задачах інтелектуального виявлення атак є коректність і узгодженість вихідного набору спостережень. Реальні або наближені до реальних набори мережевого трафіку можуть містити пропущені значення, некоректні числові представлення, дублікати записів, а також неоднорідність форматів окремих атрибутів. Подібні дефекти не можна розглядати як другорядні технічні похибки, оскільки вони здатні істотно спотворювати структуру простору ознак, змінювати статистичні характеристики змінних і формувати хибні закономірності, які модель буде помилково інтерпретувати як інформативні. Наявність дублікатів, зокрема, може призводити до штучного посилення окремих патернів трафіку й завищення якості моделі під час оцінювання, тоді як пропущені або нескінченні значення унеможливлюють коректне застосування багатьох алгоритмів машинного навчання та процедур попередньої обробки. У зв’язку з цим очищення даних, їх уніфікація та приведення до коректного числового формату розглядаються як обов’язкова передумова подальших етапів дослідження.

Не менш важливою є вимога узгодженості масштабів числових ознак. У наборах мережевого трафіку окремі характеристики можуть суттєво відрізнятися за величинами: одні ознаки можуть набувати порівняно невеликих значень, тоді як інші – змінюватися на порядки. За відсутності масштабування це призводить до домінування окремих змінних у процесі навчання, що є особливо критичним для моделей, чутливих до відстаней між об’єктами або до величин коефіцієнтів оптимізації. До таких моделей належать, зокрема, метод опорних векторів, метод k-найближчих сусідів та логістична регресія. Крім того, неузгоджені масштаби ускладнюють застосування методів зниження розмірності, оскільки без попередньої стандартизації простір ознак може бути зміщений у бік змінних із більшою дисперсією. Тому стандартизація ознак є важливою вимогою для

забезпечення стабільності навчання, коректності геометрії простору даних та відтворюваності експериментальних результатів.

Ще однією принциповою вимогою до даних є контроль надлишковості ознак і мультиколінеарності. Для мережевого трафіку характерною є наявність великої кількості статистичних характеристик потоків, які нерідко виявляються сильно корельованими між собою. Це пояснюється тим, що багато ознак формуються на основі спільних або близьких параметрів мережевої активності, наприклад часових інтервалів, інтенсивності потоків, обсягів пакетів чи кількості з'єднань. Надлишковість ознак ускладнює навчання моделей, підвищує ризик перенавчання та може сприяти нестійкості рішень, коли зміни у вибірці або випадкове розбиття даних помітно впливають на результат класифікації. У таких умовах постає необхідність не лише аналізу інформативності ознак, а й застосування методів зниження розмірності, що дозволяють побудувати компактніше та стійкіше представлення даних без суттєвих втрат корисної інформації.

Для задач виявлення атак особливого значення набуває також проблема дисбалансу класів. У реальному мережевому середовищі нормальний трафік, як правило, суттєво переважає шкідливий, а окремі типи атак можуть бути представлені незначною кількістю прикладів. Це створює проблему для моделей машинного навчання, оскільки вони схильні орієнтуватися на клас більшості, мінімізуючи загальну помилку, але водночас погіршуючи розпізнавання рідкісних класів. У результаті модель може демонструвати високу загальну точність, однак залишатися недостатньо чутливою до атак, особливо до тих, що трапляються рідко. З огляду на це, у дослідженні питання балансування вибірки розглядається як один із центральних етапів підготовки даних. Для бінарної постановки це означає вирівнювання кількості прикладів нормального і атакувального трафіку, а для багатокласової – забезпечення більш рівномірного представлення різних типів атак, у тому числі шляхом використання методів синтетичного збільшення вибірки.

Окремою методологічною вимогою є запобігання витоку інформації під час побудови та оцінювання моделей. У задачах аналізу мережевого трафіку експериментальний дизайн має бути побудований таким чином, щоб жодне з

перетворень, пов’язаних з підготовкою даних, не використовувало інформацію з тестової або валідаційної підмножини на етапі налаштування параметрів. Якщо параметри стандартизації, зниження розмірності чи балансування визначаються одночасно на всій вибірці, це призводить до прихованого “підглядання” у тестові дані й, як наслідок, до завищення метрик якості. У межах коректної експериментальної процедури всі параметри перетворень мають визначатися виключно на тренувальних даних і лише після цього застосовуватися до тестових або валідаційних підмножин. Дотримання цього принципу є необхідною умовою достовірності висновків, відтворюваності результатів і їх прикладної значущості для реального впровадження в системах виявлення атак.

Постановка задачі дослідження у роботі передбачає розроблення та експериментальне обґрунтування методу та процедур інтелектуального виявлення атак у мережевому трафіку на основі методів машинного навчання в бінарній і багатокласовій постановках. Досягнення поставленої мети потребує методично узгодженої організації формування навчальних даних, що передбачає забезпечення їхньої коректності, структурної узгодженості, стандартизації числових ознак, зменшення надлишковості ознакового простору, балансування класів та недопущення витоку інформації. Сформульовані постановки задачі та вимоги до даних визначають вихідні умови для побудови методу підготовки даних і формування інформаційних моделей мережевого трафіку, структура якого розглядається в подальших підрозділах.

2.2 Характеристика набору даних CIC-IDS2017 та формування вибірок для дослідження

Для проведення експериментального дослідження в роботі використано набір даних CIC-IDS2017, який є одним із найбільш поширених і репрезентативних наборів для задач побудови систем виявлення атак на основі методів машинного навчання [5, 45]. Використання саме цього набору даних зумовлено тим, що він орієнтований на моделювання реалістичних сценаріїв мережевої активності та

містить як нормальний трафік, так і різні категорії атак, характерні для сучасного мережевого середовища. Така структура даних робить його придатним для розв'язання як задачі бінарної класифікації, у якій необхідно відокремити атакувальний трафік від нормального, так і задачі багатокласового розпізнавання, де важливо визначити конкретний тип атаки.

Набір CIC-IDS2017 сформовано на основі мережевих потоків, отриманих у контрольованому середовищі, яке відтворює типові умови функціонування корпоративної мережі. Кожне спостереження в наборі являє собою опис окремого потоку трафіку, представлений множиною кількісних ознак (рис. 2.1), що характеризують інтенсивність обміну, часові параметри, статистику пакетів, ознаки активності у прямому та зворотному напрямках, а також інші узагальнені характеристики мережевої взаємодії. Окрім векторів ознак, для кожного запису задано мітку класу [5, 45], яка вказує на належність потоку до нормального трафіку або до одного з типів атак. Такий формат подання є зручним для побудови моделей машинного навчання, оскільки дозволяє формувати матрицю ознак і цільовий вектор для подальшого навчання класифікаторів та оцінювання їхньої якості.

```
[10]: data.columns

[10]: Index(['Destination Port', 'Flow Duration', 'Total Fwd Packets',
       'Total Backward Packets', 'Total Length of Fwd Packets',
       'Total Length of Bwd Packets', 'Fwd Packet Length Max',
       'Fwd Packet Length Min', 'Fwd Packet Length Mean',
       'Fwd Packet Length Std', 'Bwd Packet Length Max',
       'Bwd Packet Length Min', 'Bwd Packet Length Mean',
       'Bwd Packet Length Std', 'Flow Bytes/s', 'Flow Packets/s',
       'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max', 'Flow IAT Min',
       'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max',
       'Fwd IAT Min', 'Bwd IAT Total', 'Bwd IAT Mean', 'Bwd IAT Std',
       'Bwd IAT Max', 'Bwd IAT Min', 'Fwd PSH Flags', 'Bwd PSH Flags',
       'Fwd URG Flags', 'Bwd URG Flags', 'Fwd Header Length',
       'Bwd Header Length', 'Fwd Packets/s', 'Bwd Packets/s',
       'Min Packet Length', 'Max Packet Length', 'Packet Length Mean',
       'Packet Length Std', 'Packet Length Variance', 'FIN Flag Count',
       'SYN Flag Count', 'RST Flag Count', 'PSH Flag Count', 'ACK Flag Count',
       'URG Flag Count', 'CWE Flag Count', 'ECE Flag Count', 'Down/Up Ratio',
       'Average Packet Size', 'Avg Fwd Segment Size', 'Avg Bwd Segment Size',
       'Fwd Header Length.1', 'Fwd Avg Bytes/Bulk', 'Fwd Avg Packets/Bulk',
       'Fwd Avg Bulk Rate', 'Bwd Avg Bytes/Bulk', 'Bwd Avg Packets/Bulk',
       'Bwd Avg Bulk Rate', 'Subflow Fwd Packets', 'Subflow Fwd Bytes',
       'Subflow Bwd Packets', 'Subflow Bwd Bytes', 'Init_Win_bytes_forward',
       'Init_Win_bytes_backward', 'act_data_pkt_fwd', 'min_seg_size_forward',
       'Active Mean', 'Active Std', 'Active Max', 'Active Min', 'Idle Mean',
       'Idle Std', 'Idle Max', 'Idle Min', 'Label'],
      dtype='object')
```

Рисунок 2.1 – Перелік ознак набору даних CIC-IDS2017

Суттєвою перевагою набору CIC-IDS2017 є те, що він охоплює декілька категорій атак, які відрізняються механізмом реалізації, інтенсивністю, тривалістю та характером впливу на мережеву інфраструктуру. Це дає змогу досліджувати

поведінку моделей машинного навчання не лише в умовах простого розділення трафіку на класи «норма» та «атака» (рис. 2.2), але й у значно складнішій постановці, коли потрібно розпізнати окремі типи атакувальної активності. У контексті практичного застосування систем виявлення атак це має принципове значення, оскільки різні типи атак потребують відмінних сценаріїв реагування, локалізації та подальшого аналізу інцидентів.

Idle Std	Idle Max	Idle Min	Label
0	58100000	58100000	BENIGN
0	0	0	PortScan
376216.1629	58800000	58300000	BENIGN
0	0	0	BENIGN
0	0	0	BENIGN
0	0	0	BENIGN
0	0	0	PortScan
0	0	0	BENIGN
0	0	0	PortScan
0	0	0	PortScan
0	0	0	BENIGN
0	0	0	BENIGN
0	0	0	BENIGN
0	0	0	BENIGN
830.2040111	10000000	9999260	BENIGN

Рисунок 2.2 – Label «BENING / PortScan»

Разом із тим використання CIC-IDS2017 пов’язане з рядом особливостей, які безпосередньо впливають на організацію дослідження. Насамперед цей набір даних характеризується вираженим дисбалансом класів [5, 14, 19]. Нормальний трафік у ньому представлений значно більшою кількістю записів, ніж окремі категорії атак, а деякі типи атакувальної активності зустрічаються вкрай рідко. Така властивість є типовою для даних мережевого трафіку і певною мірою відображає реальні умови функціонування комп’ютерних мереж, де більшість потоків належить до легітимної активності. Проте з точки зору машинного навчання подібна незбалансованість створює ризик того, що модель буде орієнтуватися переважно на клас більшості, що призводить до погіршення розпізнавання малопредставлених атак. Під час роботи з CIC-IDS2017 необхідно не лише враховувати початковий розподіл класів, а й застосовувати спеціальні процедури формування вибірок для навчання.

Ще однією важливою особливістю набору є те, що його ознаки не є незалежними між собою. Оскільки значна частина характеристик мережевого потоку обчислюється на основі близьких статистичних параметрів, між окремими змінними можуть виникати сильні кореляційні зв’язки. Це призводить до надлишковості ознак, ускладнює навчання моделей і зумовлює потребу в додатковому аналізі структури даних та застосуванні методів зниження розмірності [17]. При практичному використанні CIC-IDS2017 можливі технічні труднощі, пов’язані з наявністю пропущених, нескінченних або некоректно представлених значень, що потребує попереднього очищення й уніфікації даних перед навчанням моделей.

З урахуванням цілей дослідження у роботі сформовано дві основні постановки задачі на основі CIC-IDS2017: бінарну та багатокласову. Запропонована організація експериментального дослідження дає змогу оцінити ефективність моделей машинного навчання на різних рівнях складності задачі виявлення атак і водночас відтворює логіку практичного функціонування інтелектуальних систем мережевого моніторингу. У бінарній постановці система виконує базове розмежування між безпечним і шкідливим трафіком, тоді як у багатокласовій – здійснює деталізацію виявленої мережевої загрози за типами атак.

Для бінарної класифікації всі записи з міткою «BENIGN» розглядаються як нормальний трафік, а всі інші записи – як атакувальний трафік (рис. 2.2). Така схема дозволяє сформувати узагальнену задачу виявлення вторгнень, у якій не враховується внутрішня структура атакувальної активності, а увага зосереджується на факті її наявності. З практичної точки зору це відповідає роботі первинного модуля системи виявлення атак, який повинен максимально надійно відокремлювати потенційно небезпечні мережеві потоки від легітимних. Оскільки у вихідному наборі даних кількість записів нормального трафіку істотно переважає кількість атак, для дослідження формується збалансована вибірка, у якій кількість об’єктів обох класів є порівнянною. Це створює коректні умови для навчання моделей та дозволяє отримати більш об’єктивну оцінку їхньої здатності виявляти шкідливу активність.

Для багатокласової класифікації застосовується більш деталізована схема подання цільової змінної, за якої зберігається інформація про конкретний тип атаки. У цій постановці дослідження орієнтоване на класи з достатньою кількістю спостережень для побудови статистично обґрунтованих моделей. Це пояснюється тим, що деякі категорії атак у наборі CIC-IDS2017 представлені одиничними або дуже нечисленними прикладами, що унеможливлює коректне навчання і порівняння алгоритмів. У зв’язку з цим до фінальної багатокласової вибірки включаються найбільш репрезентативні класи мережевого трафіку, серед яких нормальний трафік та кілька типових категорій атак, для яких наявна достатня кількість записів.

У межах проведеного дослідження для багатокласової класифікації було відібрано такі класи: BENIGN, DoS, DDoS, Port Scan, Brute Force, Web Attack, Bot, Infiltration, Heartbleed (рис. 2.3). Такий вибір обумовлений тим, що зазначені категорії були включені до багатокласової постановки з урахуванням їх наявності у вихідному наборі даних та можливості подальшого застосування процедур балансування класів. Натомість класи з критично малою кількістю спостережень потребують окремого контролю під час формування вибірки, оскільки їх використання без балансування може призвести до нестабільного навчання моделей. Після відбору класів здійснюється формування більш збалансованого набору даних, що дозволяє зменшити зміщення моделей у бік домінуючих категорій і створює основу для подальшого застосування методів балансування класів.

Важливо наголосити, що формування вибірок для дослідження виконується не лише з метою технічної підготовки даних, а й як окремий етап організації експерименту. Від того, які саме класи включено до аналізу, яким є їхній розподіл, скільки спостережень використовується для навчання і тестування, безпосередньо залежить інтерпретація отриманих результатів. Наприклад, висока точність моделі в умовах домінування одного класу ще не означає її реальної придатності для IDS, якщо при цьому вона не забезпечує належного рівня виявлення атак. Саме тому у

роботі формування вибірок здійснюється з урахуванням не лише доступності даних, але й методологічної коректності подальшого оцінювання моделей.

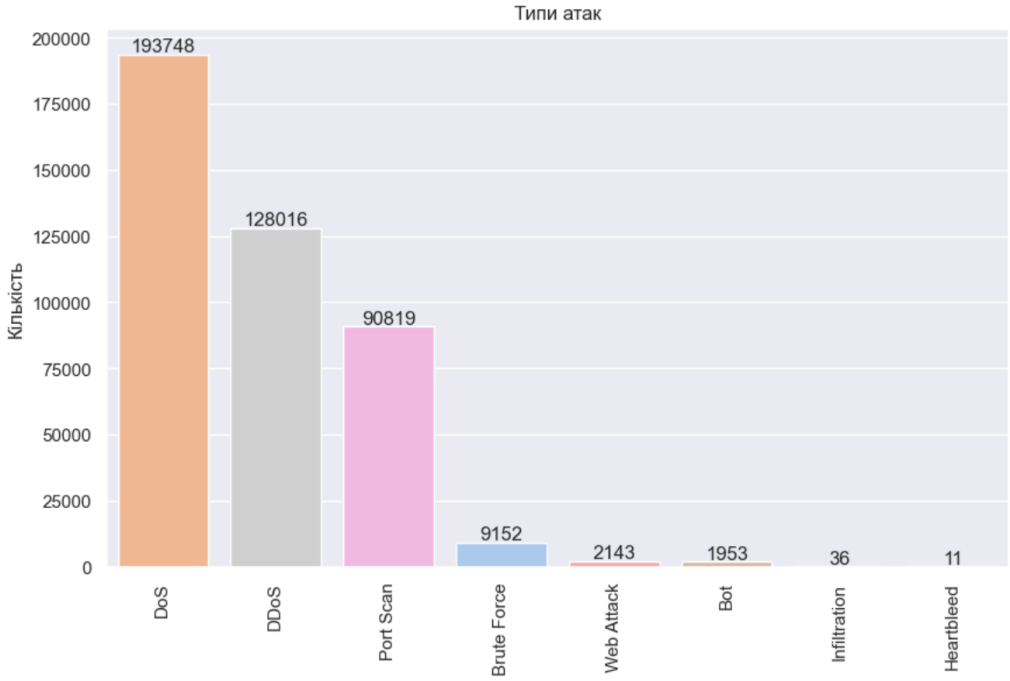


Рисунок 2.3 – Типи атак

Окрему увагу при роботі з CIC-IDS2017 приділено забезпеченню відтворюваності експериментів. Для цього під час формування підвибірок, їх перемішування, поділу на навчальну та тестову частини, а також виконання процедур балансування використовуються фіксовані параметри випадковості. Забезпечення фіксованості параметрів випадковості дає змогу відтворити експеримент за однакових умов і отримати узгоджені результати, що є важливою вимогою до наукового дослідження.

Таким чином, набір даних CIC-IDS2017 є репрезентативною основою для дослідження методів інтелектуального виявлення атак у мережевому трафіку. Його використання дозволяє реалізувати як бінарну, так і багатокласову постановку задачі, що забезпечує всебічний аналіз ефективності моделей машинного навчання. Разом із тим характерні особливості цього набору – дисбаланс класів, надлишковість ознак, наявність технічних дефектів і нерівномірна представленість окремих типів атак – вимагають застосування спеціальних процедур формування вибірок та попередньої обробки даних.

2.3 Структура методу

Запропонований метод призначений для формування підготовленого інформаційного подання мережевого трафіку, придатного для розв’язання двох взаємопов’язаних задач: бінарного виявлення атак і багатокласового розпізнавання їх типів. У бінарній постановці задача полягає у відокремленні нормального трафіку від атакувального, тоді як у багатокласовій постановці – у визначенні конкретного типу атаки на основі сукупності ознак мережевого потоку.

Вхідними даними методу є записи мережевих потоків набору CIC-IDS2017, що містять числові характеристики трафіку та відповідні мітки класів. Вихідними результатами є підготовлені ознакові матриці, цільові змінні для бінарної та багатокласової класифікації, збалансовані навчальні вибірки, навчені моделі машинного навчання та показники оцінювання їх ефективності.

Загальна структура методу включає п’ять функціональних блоків: блок попередньої обробки даних, блок формування ознакового простору, блок формування та балансування вибірок, блок побудови моделей машинного навчання та блок оцінювання результатів класифікації [8, 14, 17, 20, 21]. Зазначена структура забезпечує методичну узгодженість усіх етапів дослідження і дозволяє уникнути розгляду процедур очищення, зниження розмірності, балансування, навчання та оцінювання.

Перший блок методу охоплює процедури очищення, уніфікації та стандартизації даних мережевого трафіку. Його призначення полягає в усуненні технічних дефектів вихідного набору даних, приведенні ознак до узгодженої структури, обробці пропущених і нескінченних значень, а також масштабуванні числових характеристик. Виконання цього блоку створює основу для подальшого коректного застосування методів аналізу ознак і машинного навчання.

Другий блок пов’язаний з аналізом ознакового простору та зниженням його розмірності. На цьому етапі досліджуються взаємозв’язки між ознаками мережевого трафіку, виявляються надлишкові або сильно корельовані

характеристики, а також формується компактніше подання даних. Це дає змогу зменшити обчислювальну складність подальшого моделювання та підвищити стійкість моделей до надлишкової інформації.

Третій блок передбачає формування вибірок для бінарної та багатокласової класифікації, а також балансування класів. У межах цього блоку вихідні мітки класів перетворюються відповідно до постановки задачі: для бінарного виявлення атак формується поділ на нормальний та атакувальний трафік, а для багатокласового розпізнавання зберігається інформація про конкретні типи атак. Балансування навчальних вибірок використовується для зменшення впливу нерівномірного представлення класів на результати навчання моделей.

Четвертий блок методу охоплює побудову моделей машинного навчання. Для бінарного виявлення атак застосовуються моделі, орієнтовані на відокремлення атакувального трафіку від нормального, а для багатокласового розпізнавання – моделі, здатні класифікувати мережеві потоки за типами атак. Використання єдиної схеми підготовки даних для всіх моделей забезпечує коректність їх подальшого порівняння.

П’ятий блок пов’язаний з оцінюванням якості моделей і організацією експериментального дослідження. Для цього використовуються тестові вибірки, перехресна перевірка та система взаємодоповнювальних метрик, що дозволяють оцінити не лише загальну точність класифікації, а й здатність моделей виявляти атакувальний трафік, розпізнавати окремі типи атак і зберігати стабільність результатів на різних підмножинах даних.

2.3.1 Процедури очищення, уніфікації та стандартизації даних мережевого трафіку

Однією з визначальних передумов побудови ефективних моделей машинного навчання для виявлення атак у мережевому трафіку є коректна попередня обробка вихідних даних. Незалежно від обраного алгоритму класифікації, якість моделі значною мірою залежить від того, наскільки узгодженими, повними та придатними

до математичного аналізу є дані, що використовуються для навчання. У задачах інтелектуального моніторингу мережевого трафіку ця вимога має велике значення, оскільки набори даних часто формуються з кількох джерел, містять велику кількість статистичних ознак, а також можуть включати технічні дефекти, які спотворюють результати моделювання. Першим етапом дослідження є очищення, уніфікація та стандартизація даних, спрямовані на формування коректного та відтворюваного представлення мережевого трафіку для подальшого аналізу.

У межах запропонованого методу процедури очищення, уніфікації та стандартизації даних розглядаються як початковий етап приведення даних мережевого трафіку до узгодженого числового вигляду, придатного для подальшого застосування методів зниження розмірності, балансування класів та навчання класифікаційних моделей. Вхідними даними для цього етапу є табличне представлення мережевих потоків, сформоване на основі набору CIC-IDS2017, яке містить набір ознак і цільову мітку класу. Вихідним результатом є очищена матриця ознак та узгоджений вектор міток, що забезпечують стабільність подальших етапів дослідження [141].

Першим кроком цього етапу є інтеграція та уніфікація піднаборів даних. Оскільки CIC-IDS2017 може використовуватися як сукупність кількох окремих піднаборів, отриманих у різні часові проміжки та за різних сценаріїв мережевої активності, перед початком аналізу необхідно об’єднати їх у єдину структуру. Таке об’єднання не є суто технічною операцією, оскільки різні піднабори можуть відрізнятися порядком ознак, способами позначення атрибутів або особливостями подання окремих значень (рис. 2.4). Тому на етапі уніфікації виконується приведення назв ознак до єдиного формату, перевірка їх послідовності, а також забезпечення однакового типу представлення атрибутів у всіх частинах набору. Уніфікація є необхідною умовою для того, щоб подальші процедури обробки та аналізу виконувалися над цілісним і логічно узгодженим масивом даних.

```
# Завантаження набору даних
data1 = pd.read_csv('dataset/Monday-WorkingHours.pcap_ISCX.csv')
data2 = pd.read_csv('dataset/Tuesday-WorkingHours.pcap_ISCX.csv')
data3 = pd.read_csv('dataset/Wednesday-workingHours.pcap_ISCX.csv')
data4 = pd.read_csv('dataset/Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv')
data5 = pd.read_csv('dataset/Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv')
data6 = pd.read_csv('dataset/Friday-WorkingHours-Morning.pcap_ISCX.csv')
data7 = pd.read_csv('dataset/Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv')
data8 = pd.read_csv('dataset/Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv')
```

```
data_list = [data1, data2, data3, data4, data5, data6, data7, data8]

print('Data dimensions: ')
for i, data in enumerate(data_list, start = 1):
  rows, cols = data.shape
  print(f'Data{i} -> {rows} rows, {cols} columns')
```

```
Data dimensions:
Data1 -> 529918 rows, 79 columns
Data2 -> 445909 rows, 79 columns
Data3 -> 692703 rows, 79 columns
Data4 -> 170366 rows, 79 columns
Data5 -> 288602 rows, 79 columns
Data6 -> 191033 rows, 79 columns
Data7 -> 286467 rows, 79 columns
Data8 -> 225745 rows, 79 columns
```

Рисунок 2.4 – Поєднання окремих піднаборів даних в єдиний набір

Наступним кроком є перевірка коректності значень ознак і виявлення технічних дефектів. У великих наборах мережевого трафіку можуть зустрічатися пропущені значення, нескінченні значення, що виникають під час обчислення похідних статистичних характеристик, а також некоректні або нечислові представлення атрибутів (рис. 2.5). Наявність таких дефектів унеможливлює коректне застосування математичних методів аналізу, оскільки більшість алгоритмів машинного навчання, а також процедури стандартизації та зниження розмірності вимагають повністю числового і скінченного представлення даних [20, 46]. У зв’язку з цим процедура передбачає приведення ознак до числового формату, заміну нескінченних значень на пропущені та подальшу обробку таких пропусків відповідно до обраної стратегії імпутації. Основною метою цього кроку є забезпечення числової коректності матриці ознак і збереження максимально можливої кількості корисної інформації без внесення істотних спотворень у структуру даних.

Ідентифікація відсутніх значень

```
missing_val = data.isna().sum()
print(missing_val.loc[missing_val > 0])
```

```
Flow Bytes/s    353
dtype: int64
```

```
# Перевірка на відсутність значень
numeric_cols = data.select_dtypes(include = np.number).columns
inf_count = np.isinf(data[numeric_cols]).sum()
print(inf_count[inf_count > 0])
```

```
Flow Bytes/s      1211
Flow Packets/s    1564
dtype: int64
```

```
# Заміна будь-яких нескінченних значень (додатних чи від'ємних) на NaN (не число)
print(f'Initial missing values: {data.isna().sum().sum()}')

data.replace([np.inf, -np.inf], np.nan, inplace = True)

print(f'Missing values after processing infinite values: {data.isna().sum().sum()}')
```

```
Initial missing values: 353
Missing values after processing infinite values: 3128
```

```
missing = data.isna().sum()
print(missing.loc[missing > 0])
```

```
Flow Bytes/s      1564
Flow Packets/s    1564
dtype: int64
```

Рисунок 2.5 – Ідентифікація пропущених і нескінчених значень

Важливою складовою очищення є видалення дубльованих записів (рис. 2.6). Для задач аналізу мережевого трафіку дублікати становлять окрему проблему, оскільки вони збільшують статистичну вагу окремих патернів і можуть створювати ілюзію кращого узагальнення моделі, ніж це є насправді. Якщо однакові записи багаторазово присутні у вибірці, модель може «запам’ятовувати» ці спостереження замість виявлення реальних закономірностей, що призводить до підвищення ризику перенавчання. Крім того, наявність дублікатів може викривлювати результати крос-валідації та тестування, особливо якщо подібні або тотожні записи потрапляють як у навчальну, так і в контрольну частину вибірки. Саме тому усунення дубльованих рядків є обов’язковим елементом формування репрезентативного набору даних для подальшого дослідження.

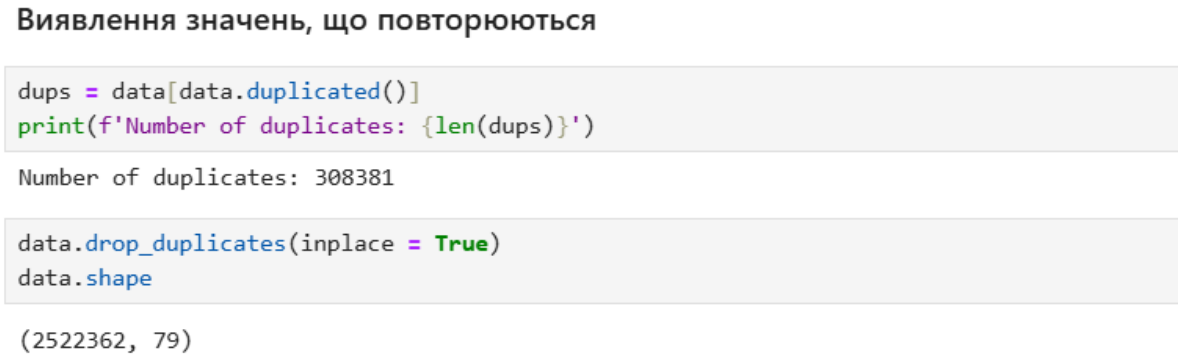


Рисунок 2.6 – Робота з дублікатами

Окрім цього, на етапі очищення доцільно контролювати наявність ознак із нульовою або майже нульовою дисперсією. Такі ознаки практично не змінюються в межах вибірки, а отже не несуть істотної інформації для розрізнення класів. Їх збереження збільшує розмірність простору ознак, ускладнює подальшу обробку та може посилювати вплив шуму. Вилучення малозмінних атрибутів дозволяє зменшити надлишковість даних і сформувати більш компактний та інформативний опис мережевого трафіку. Для задач інтелектуального виявлення атак це важливо, оскільки надмірна кількість слабкоінформативних характеристик може маскувати дійсно значущі закономірності в даних.

Паралельно з очищенням виконується формування цільової змінної залежно від обраної постановки задачі. Для бінарної класифікації всі записи нормального трафіку відносяться до одного класу, тоді як всі записи, що відповідають атакам, об’єднуються в інший клас. Для багатокласової постановки цільова змінна зберігає інформацію про конкретний тип атаки, що дозволяє досліджувати не лише факт наявності загрози, але й її категорію. На цьому етапі важливо забезпечити однозначність і стабільність кодування міток у межах усіх експериментів, оскільки коректність цільової змінної безпосередньо впливає на достовірність оцінювання моделей і на інтерпретацію отриманих результатів.

Після завершення процедур очищення та уніфікації здійснюється стандартизація числових ознак, яка є необхідною умовою для коректного застосування частини алгоритмів машинного навчання і методів подальшого аналізу. Ознаки мережевого трафіку можуть мати дуже різні масштаби: деякі з них

вимірюються в одиницях або десятках, інші – у тисячах чи навіть мільйонах. Якщо не виконати попереднього масштабування, ознаки з більшими числовими значеннями матимуть непропорційно великий вплив на результати моделювання, особливо у тих методах, де рішення залежать від геометрії простору ознак, евклідових відстаней або числової оптимізації параметрів. До таких алгоритмів, зокрема, належать логістична регресія, метод опорних векторів і метод k-найближчих сусідів.

У межах дослідження використовується стандартизація типу z-score, за якої кожна ознака перетворюється таким чином, щоб її середнє значення дорівнювало нулю, а стандартне відхилення – одиниці [20]. Таке перетворення дозволяє вирівняти внесок окремих атрибутів у процес навчання, зменшити ризик домінування окремих змінних та забезпечити більш стійке функціонування алгоритмів класифікації. Крім того, стандартизація є обов’язковою передумовою для застосування методу головних компонент, оскільки без попереднього масштабування головні компоненти можуть формуватися переважно під впливом ознак із великою дисперсією, що призведе до викривлення структури простору даних і зменшить ефективність редукції розмірності.

Особливу увагу на етапі стандартизації слід приділяти коректності експериментального дизайну, оскільки некоректне визначення параметрів масштабування може призводити до витоку інформації між навчальною та контрольною частинами вибірки. Для забезпечення достовірної оцінки узагальнювальної здатності моделей параметри стандартизації доцільно визначати на навчальних даних із подальшим застосуванням до контрольних підмножин без повторного налаштування [20, 46].

Таким чином, процедури очищення, уніфікації та стандартизації даних мережевого трафіку є базовим етапом запропонованого методу підготовки даних і формування інформаційних моделей. Їх реалізація забезпечує формування узгодженої матриці ознак, у якій усунено технічні дефекти, видалено дублікати й малоінформативні ознаки, а числові атрибути приведено до порівнюваного масштабу. Отримане представлення даних є основою для подальших етапів

дослідження, пов’язаних з аналізом ознак, зниженням розмірності, формуванням вибірок і балансуванням класів.

2.3.2 Процедура аналізу ознак і зниження розмірності простору даних

Після етапу очищення, уніфікації та стандартизації даних наступним важливим кроком дослідження є аналіз структури простору ознак мережевого трафіку та зниження його розмірності. Необхідність цього етапу зумовлена тим, що набори даних для задач виявлення атак зазвичай містять велику кількість статистичних характеристик мережевих потоків, частина з яких є взаємопов’язаними, дублює подібні властивості трафіку або має обмежену додаткову інформативність. За таких умов використання повного набору ознак не завжди підвищує якість моделювання, а навпаки, може ускладнювати навчання, збільшувати обчислювальні витрати та сприяти перенавчанню [17]. Саме тому в роботі застосовується поєднання кореляційного аналізу та методу головних компонент, що дозволяє дослідити залежності між ознаками та сформувати компактніше представлення даних без суттєвої втрати корисної інформації [145].

У задачах інтелектуального моніторингу мережевого трафіку ознаки часто формуються як похідні характеристики одних і тих самих процесів обміну даними: інтенсивності потоків, кількості пакетів, часових інтервалів, співвідношення прямих і зворотних пакетів, статистичних параметрів довжин пакетів тощо. Унаслідок цього між окремими атрибутами виникають сильні лінійні залежності, що формують явище мультиколінеарності. Для моделей машинного навчання така ситуація є небажаною, оскільки надлишкові ознаки не збільшують інформативність пропорційно до зростання розмірності, але можуть посилювати вплив шуму, ускладнювати інтерпретацію результатів і робити моделі менш стійкими до змін у вибірці. Особливо це стосується тих методів, які чутливі до геометрії простору ознак або до структури дисперсії даних.

Першим кроком цього етапу є аналіз інформативності та взаємозв’язків між ознаками мережевого трафіку. Його метою є виявлення тих характеристик, які або

несуть подібну інформацію, або формують стійкі групи взаємозалежних змінних. Для цього використовується кореляційний аналіз, який дозволяє кількісно оцінити силу лінійного зв’язку між парами ознак. У межах дослідження такий аналіз є важливим не лише як описовий інструмент, а і як засіб методичного обґрунтування подальшого застосування редукції розмірності. Якщо значна частина ознак виявляється сильно корельованою, це свідчить про наявність надлишковості в даних і створює підстави для переходу до компактнішого ортогонального представлення.

Для оцінювання сили взаємозв’язку між ознаками використовується коефіцієнт кореляції Пірсона, який відображає ступінь лінійної залежності між двома числовими змінними. На його основі формується кореляційна матриця (рис. 2.7), що дає змогу дослідити загальну структуру взаємозв’язків у просторі ознак. Візуалізація кореляційної матриці дозволяє виявити групи атрибутів із високими додатними або від’ємними кореляціями, а також визначити ознаки, для яких характерна майже повна лінійна залежність. У контексті дослідження мережевого трафіку це є важливим, оскільки наявність великої кількості таких пар підтверджує, що початковий простір ознак містить значну частку дубльованої або близької за змістом інформації.

За результатами проведеного аналізу встановлено, що в наборі даних існують численні пари ознак із високим рівнем кореляції, зокрема з коефіцієнтом кореляції 0,85 і вище. Це свідчить про виражену мультиколінеарність і підтверджує доцільність застосування методів зниження розмірності.

Виявлені кореляційні зв’язки не слід розглядати як випадкове явище, оскільки вони обумовлені природою формування характеристик мережевого трафіку: багато ознак описують одні й ті самі аспекти поведінки потоку з різних боків або в різних масштабах. Відтак використання повного набору таких атрибутів у незмінному вигляді може призводити до перевантаження моделі надлишковою інформацією та зниження її узагальнювальної здатності.

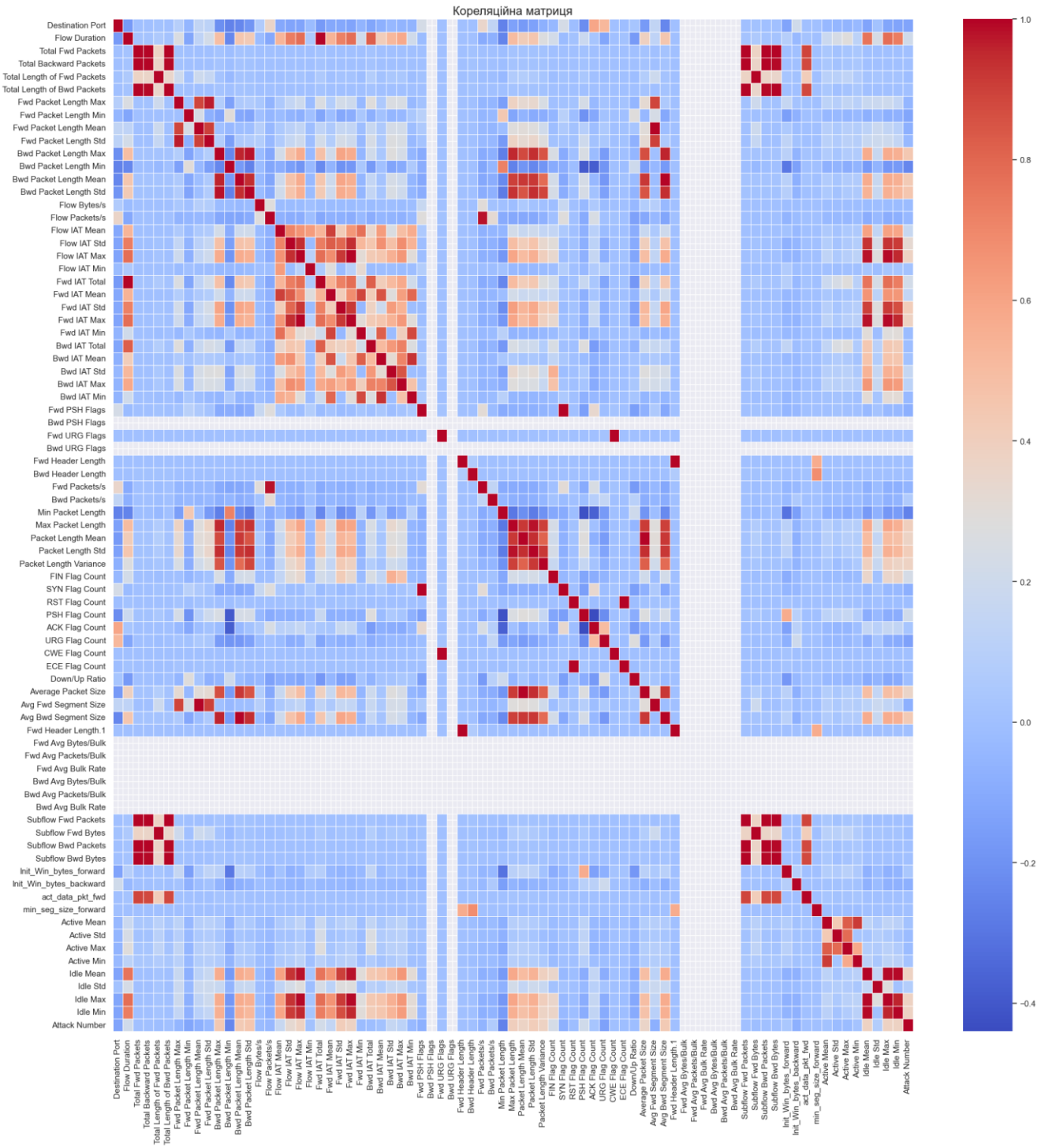


Рисунок 2.7 – Кореляційна матриця

Після виявлення надлишковості ознак у роботі застосовується метод аналізу головних компонент (Principal Component Analysis, PCA), який дозволяє перейти від початкового корельованого простору ознак до нового простору взаємно ортогональних компонент [17]. PCA є одним із найбільш поширених методів лінійного зниження розмірності і базується на перетворенні початкових змінних у

новий набір лінійних комбінацій, упорядкованих за спаданням пояснюваної дисперсії. Кожна головна компонента є новою віссю у просторі даних, що відображає один із напрямів найбільшої варіативності. У результаті перші компоненти зберігають основну інформацію про структуру вибірки, тоді як наступні компоненти мають дедалі менший внесок у загальну дисперсію і можуть містити переважно шумові або слабкоінформативні складові.

У межах дослідження PCA розглядається не як формальна процедура механічного скорочення кількості ознак, а як метод стабілізації навчання моделей та усунення мультиколінеарності. Перехід до ортогонального простору компонент дає змогу усунути лінійні залежності між змінними, зменшити вплив надлишкових ознак і сформувати компактніше представлення мережевого трафіку. Це важливо для подальшого застосування алгоритмів машинного навчання, а також для процедур балансування класів, оскільки в компактнішому просторі даних зменшується ризик посилення шумових залежностей і підвищується числова стабільність обробки.

Вхідними даними для застосування PCA є стандартизована матриця ознак, отримана на попередньому етапі обробки. Стандартизація вирівнює внесок початкових змінних і забезпечує коректність побудови компонент. У результаті застосування PCA формується нова матриця ознак, у якій кожний об’єкт мережевого трафіку представлений координатами у просторі головних компонент.

Кількість компонент у роботі обирається за критерієм кумулятивної пояснюваної дисперсії, що дозволяє знайти компроміс між компактністю представлення та збереженням інформаційної повноти даних. Практична мета полягає в тому, щоб залишити лише ті компоненти, які сумарно пояснюють переважну частку варіативності вихідного простору ознак, і водночас відкинути компоненти з мінімальним внеском, які з великою ймовірністю відображають шум або незначні флуктуації.

```python
# Стандартизація набору даних
from sklearn.preprocessing import StandardScaler

features = data.drop('Attack Type', axis = 1)
attacks = data['Attack Type']

scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

```python
from sklearn.decomposition import IncrementalPCA

size = len(features.columns) // 2
ipca = IncrementalPCA(n_components = size, batch_size = 500)
for batch in np.array_split(scaled_features, len(features) // 500):
    ipca.partial_fit(batch)

print(f'збережено інформацію: {sum(ipca.explained_variance_ratio_):.2%}')
```

```
збережено інформацію: 99.30%
```

Рисунок 2.8 – Вибір кількості головних компонент за допомогою PCA

За результатами аналізу кумулятивної пояснюваної дисперсії визначено, що 35 головних компонент забезпечують збереження 99,30 % дисперсії вихідного простору ознак (рис. 2.8). Це свідчить про наявність надлишковості у початковому наборі ознак і підтверджує доцільність переходу до компактного простору головних компонент (рис. 2.9).

З прикладної точки зору застосування PCA в дослідженні має декілька важливих переваг. По-перше, зменшується розмірність простору даних, що спрощує подальше навчання моделей і знижує обчислювальні витрати. По-друге, усувається мультиколінеарність, завдяки чому моделі працюють у більш стійкому та менш надлишковому просторі ознак. По-третє, відсіювання компонент із малим внеском у пояснену дисперсію дозволяє частково зменшити вплив шумових складових, що позитивно впливає на узагальнювальну здатність моделей. По-четверте, компактніше представлення даних створює сприятливі умови для подальшого балансування класів, оскільки генерація нових синтетичних прикладів у просторі меншої розмірності є більш стабільною, ніж у вихідному просторі великої кількості корельованих ознак.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15	PC16	PC17	PC18
0	-2.358	-0.055	0.577	0.734	3.730	0.235	-0.016	-0.216	-0.279	1.087	0.033	0.068	1.638	0.371	-0.795	0.004	0.069	0.615
1	-2.884	-0.070	0.911	1.763	8.846	0.620	-0.057	1.106	1.907	-2.756	-0.951	-0.843	6.053	1.569	-4.732	-0.121	1.314	1.949
2	-2.417	-0.057	0.615	0.851	4.304	0.276	-0.020	-0.071	-0.035	0.664	-0.076	-0.033	2.132	0.504	-1.230	-0.010	0.207	0.764
3	-2.885	-0.070	0.912	1.765	8.852	0.619	-0.057	1.105	1.907	-2.753	-0.950	-0.843	6.056	1.568	-4.730	-0.121	1.312	1.950
4	-1.505	0.081	-0.504	0.290	-0.539	0.746	0.100	0.733	-1.146	-0.563	-0.044	-0.208	-0.434	0.033	-0.609	0.018	-0.249	-0.475
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2522357	-2.271	-0.048	0.480	0.461	2.164	-0.145	-0.016	-0.778	-0.908	2.711	0.414	0.406	0.990	-0.084	0.911	0.011	-0.030	0.339
2522358	-2.268	-0.048	0.479	0.458	2.144	-0.147	-0.016	-0.781	-0.911	2.718	0.416	0.409	0.957	-0.091	0.929	0.013	-0.045	0.327
2522359	-2.267	-0.048	0.478	0.457	2.140	-0.147	-0.016	-0.781	-0.912	2.719	0.417	0.409	0.950	-0.093	0.933	0.013	-0.048	0.324
2522360	-2.102	-0.042	0.362	0.269	1.796	0.285	0.007	-0.296	-0.631	0.967	0.082	0.111	-0.169	0.134	-0.234	0.076	-0.642	-0.128
2522361	-2.382	-0.018	0.397	0.860	2.465	0.613	0.069	-0.407	-2.359	3.919	0.654	0.503	1.382	-0.127	1.307	0.042	-0.312	0.020

	PC19	PC20	PC21	PC22	PC23	PC24	PC25	PC26	PC27	PC28	PC29	PC30	PC31	PC32	PC33	PC34	PC35	Attack Type
0	-0.658	-0.597	-0.351	-0.714	-1.722	-0.070	-0.420	0.905	-0.148	-0.334	1.488	-0.164	-0.219	-0.005	0.003	0.034	-0.016	BENIGN
1	-0.497	-0.592	-0.200	-0.567	-5.607	0.296	1.603	0.594	0.283	-2.347	2.239	-0.641	-0.367	0.011	0.001	0.126	-0.191	BENIGN
2	-0.642	-0.598	-0.335	-0.697	-2.156	-0.032	-0.197	0.868	-0.102	-0.546	1.572	-0.214	-0.235	-0.003	0.002	0.044	-0.035	BENIGN
3	-0.500	-0.594	-0.201	-0.566	-5.609	0.293	1.602	0.592	0.281	-2.335	2.240	-0.638	-0.366	0.010	0.001	0.125	-0.191	BENIGN
4	-0.140	0.114	-0.114	0.096	0.292	0.386	-0.256	-0.690	-0.599	0.116	0.795	-0.211	0.042	-0.014	0.001	-0.052	0.001	BENIGN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2522357	-0.328	-0.308	-0.436	-1.017	-0.319	-0.281	-0.369	0.763	-0.043	0.788	0.786	0.248	0.036	-0.017	0.002	0.006	-0.006	BENIGN
2522358	-0.348	-0.341	-0.425	-1.035	-0.346	-0.275	-0.360	0.758	-0.041	0.789	0.787	0.249	0.038	-0.018	0.002	0.005	-0.006	BENIGN
2522359	-0.352	-0.348	-0.423	-1.040	-0.352	-0.274	-0.358	0.757	-0.041	0.789	0.787	0.250	0.038	-0.018	0.002	0.005	-0.006	BENIGN
2522360	-0.838	-0.493	-0.516	-0.085	-0.588	0.123	-0.999	0.313	-0.373	1.793	-0.925	0.297	0.129	-0.006	0.002	-0.078	0.033	BENIGN
2522361	1.495	-1.995	3.823	3.437	-0.634	-0.781	0.580	0.556	0.295	-0.035	-0.032	0.248	-0.156	0.010	0.000	-0.069	-0.001	BENIGN

Рисунок 2.9 – Формування нового простору ознак на основі головних компонент

Методологічно важливою умовою застосування кореляційного аналізу та PCA є дотримання принципу відсутності витоку інформації. Кореляційна структура даних може аналізуватися як на повному масиві для загальної характеристики ознак, так і в межах тренувальної вибірки при побудові експериментального конвеєра. Однак саме параметри PCA, зокрема напрямки головних компонент і пояснювана дисперсія, повинні визначатися лише на тренувальних даних [17, 47]. Після цього отримане перетворення застосовується до тестової або валідаційної частини без повторного налаштування. У разі використання крос-валідації така процедура має виконуватися окремо в межах кожного фолду. Дотримання цього принципу забезпечує достовірність оцінювання якості моделей та зберігає прикладну цінність отриманих результатів [112-118].

Таким чином, процедура аналізу ознак і зниження розмірності простору даних є складовою запропонованого методу підготовки даних і формування інформаційних моделей мережевого трафіку. Її застосування дозволяє виявити

структуру взаємозв’язків між ознаками, підтвердити наявність мультиколінеарності та надлишковості атрибутів, а також сформувати компактний ортогональний простір головних компонент, придатний для подальшого формування вибірок, балансування класів і навчання моделей машинного навчання. Отримане подання даних створює основу для наступної процедури, пов’язаної з формуванням і балансуванням вибірок у бінарній та багатокласовій постановках задачі виявлення атак.

2.3.3 Процедура формування та балансування вибірок для бінарної і багатокласової класифікації

Після очищення, стандартизації та зниження розмірності простору ознак наступним етапом запропонованого методу є формування вибірок для бінарної та багатокласової класифікації. На цьому етапі виконується побудова цільових змінних відповідно до обраної постановки задачі. Для бінарної класифікації всі записи нормального трафіку відносяться до класу BENIGN, а записи, що відповідають атакам, об’єднуються в узагальнений клас Attack. Для багатокласової класифікації зберігається деталізована інформація про тип атаки, що дозволяє досліджувати не лише факт наявності атакувальної активності, а й її конкретну категорію.

Як було показано у підрозділі 2.2, сформовані на основі набору даних CIC-IDS2017 вибірки для бінарної та багатокласової класифікації характеризуються нерівномірним розподілом спостережень між класами. Така властивість є типовою для задач аналізу мережевого трафіку, однак у контексті машинного навчання вона створює суттєві методичні труднощі. За наявності вираженого дисбалансу класифікаційні моделі схильні орієнтуватися переважно на класи більшості, внаслідок чого знижується якість виявлення рідкісних, але критично важливих типів атак [14, 19, 25]. У зв’язку з цим у межах запропонованої процедури обґрунтовано диференційовану організацію формування та балансування вибірок

навчальних даних, що забезпечує методично коректні умови для подальшого навчання, оцінювання та порівняння моделей машинного навчання.

У задачі бінарної класифікації балансування реалізується шляхом вирівнювання кількості спостережень двох узагальнених класів – нормального трафіку та атак. Початковий розподіл даних у такій постановці характеризується переважанням записів нормальної мережевої активності, що може зумовити статистичне зміщення моделі в бік класу більшості. За цих умов навіть формально високе значення загальної точності не гарантує належної здатності системи розпізнавати шкідливу активність, оскільки помилки щодо атак можуть маскуватися великою кількістю правильно класифікованих нормальних спостережень. Для бінарної класифікації доцільним є формування збалансованої навчальної вибірки, у якій обидва класи мають зіставне представлення [14, 19].

Практична реалізація такої процедури у роботі ґрунтується на випадковому зменшенні кількості спостережень класу більшості. З цією метою із сукупності записів нормального трафіку відбирається підмножина, обсяг якої відповідає кількості записів класу атак або є максимально наближеним до неї. У результаті формується бінарна вибірка, у якій статистична вага обох класів є вирівняною. Використання цієї процедури пояснюється її методичною простотою, прозорістю та придатністю для оцінювання базової здатності моделей відокремлювати безпечний трафік від шкідливого без впливу вираженого дисбалансу. Крім того, зменшення класу більшості у даному випадку не призводить до критичної втрати інформативності, оскільки кількість спостережень нормального трафіку у вихідному наборі є достатньо великою.

У задачі багатокласової класифікації проблема дисбалансу має складніший характер, оскільки нерівномірність спостерігається не лише між нормальним трафіком і атаками в цілому, а й між окремими типами атакувальної активності. За таких умов простого зменшення класу більшості недостатньо, оскільки це не усуває дефіциту прикладів для класів меншості. Якщо навчання моделі виконувати на такій незбалансованій вибірці без додаткових процедур вирівнювання, вона буде орієнтуватися на найбільш масові класи, тоді як рідкісні атаки розпізнаватимуться

нестабільно або взагалі ігноруватимуться. Для інтелектуальних систем виявлення атак така ситуація є неприйнятною, оскільки навіть малопоширені типи атакувальної активності можуть становити суттєву загрозу для мережевої інфраструктури.

З метою усунення зазначених обмежень у багатокласовій постановці застосовано метод SMOTE (Synthetic Minority Over-sampling Technique), який належить до методів синтетичного оверсемплінгу [14, 21]. Його сутність полягає у генерації нових штучних прикладів для класів меншості на основі вже наявних спостережень. На відміну від простого дублювання рідкісних записів, SMOTE формує нові об’єкти в просторі ознак шляхом інтерполяції між сусідніми спостереженнями того самого класу. У результаті менш представлені класи не лише кількісно збільшуються, а й отримують більш повне та безперервне представлення у просторі ознак, що створює кращі умови для навчання класифікатора.

Перевага використання SMOTE у задачах виявлення атак полягає в тому, що цей метод дозволяє підвищити чутливість моделі до класів меншості без механічного копіювання їхніх представників. Це важливо для багатокласового аналізу мережевого трафіку, де структура окремих типів атак може бути складною та неоднорідною. Формування синтетичних прикладів на основі локального оточення реальних спостережень дає змогу частково відтворити внутрішню геометрію відповідного класу та зменшити ризик того, що модель буде налаштована лише на найбільш чисельні категорії. У такий спосіб балансування сприяє більш коректному вивченню відмінностей між окремими типами атак і підвищує якість багатокласової класифікації.

У межах проведеного дослідження балансування класів розглядається не ізольовано, а як складова цілісного конвеєра попередньої обробки даних. Перед застосуванням процедур ресемплінгу виконуються очищення даних, усунення некоректних значень, стандартизація ознак та зниження розмірності простору ознак. Така послідовність є методично обґрунтованою, оскільки генерація синтетичних спостережень у попередньо нормалізованому та компактному

просторі ознак забезпечує більш стабільний результат. Зокрема, зниження розмірності дозволяє послабити вплив мультиколінеарності, зменшити надлишковість ознак і зосередити подальше балансування на найбільш інформативних компонентах представлення даних. У результаті синтетично сформовані приклади краще відображають реальну структуру класів, ніж у випадку прямого застосування оверсемплінгу до сирого багатовимірного простору.

Окрему увагу під час реалізації балансування необхідно приділяти недопущенню витоку інформації між навчальною та тестовою частинами вибірки. З методичної точки зору процедури ресемплінгу доцільно виконувати після поділу даних на тренувальну та тестову підмножини з їх застосуванням лише до навчальної частини [21, 46]. У випадку використання крос-валідації балансування також доцільно виконувати окремо в межах кожного тренувального фолду.

Таким чином, процедура формування та балансування вибірок є складовою запропонованого методу підготовки даних і формування інформаційних моделей мережевого трафіку. У межах цієї процедури для бінарної класифікації формується вибірка з двома узагальненими класами – нормальний та атакувальний трафік, після чого виконується вирівнювання їх представлення шляхом випадкового зменшення класу більшості. Для багатокласової класифікації зберігається деталізована структура типів атак і застосовується SMOTE для синтетичного збільшення класів меншості. Застосування такої схеми балансування дає змогу зменшити статистичне зміщення моделей у бік найбільш представлених класів, підвищити коректність навчання та сформувати методично обґрунтовану основу для подальшого порівняльного аналізу алгоритмів машинного навчання в задачах інтелектуального виявлення атак.

2.3.4 Моделі машинного навчання для бінарного виявлення атак і багатокласового розпізнавання їх типів

Після формування підготовлених і збалансованих вибірок наступним етапом дослідження є вибір та обґрунтування методів машинного навчання, які будуть

застосовані для класифікації мережевого трафіку. У задачах інтелектуального виявлення атак вибір алгоритму має принципове значення, оскільки різні моделі по-різному реагують на структуру простору ознак, наявність шуму, розмірність даних, дисбаланс класів і складність меж розділення між нормальним та шкідливим трафіком. У зв'язку з цим у роботі використано сукупність методів, що належать до різних класів алгоритмів машинного навчання та відрізняються за способом побудови рішень, здатністю до узагальнення й інтерпретованістю результатів.

Застосування кількох моделей машинного навчання та відповідних конфігурацій моделей у межах одного дослідження є доцільним з двох причин. По-перше, це дозволяє виконати порівняльний аналіз ефективності моделей у бінарній і багатокласовій постановках задачі. По-друге, використання алгоритмів різної природи дає змогу оцінити, як саме особливості підготовлених даних – зокрема стандартизація, зниження розмірності та балансування класів – впливають на якість класифікації. У межах дослідження для бінарного виявлення атак обрано логістичну регресію та метод опорних векторів, а для багатокласового розпізнавання типів атак – випадковий ліс, дерево рішень і метод k-найближчих сусідів. Такий розподіл моделей дозволяє врахувати специфіку кожної постановки задачі та забезпечити порівняльний аналіз алгоритмів у межах відповідного режиму класифікації [140].

Першим із розглянутих методів є логістична регресія – цей алгоритм належить до базових методів керованого навчання для задач бінарної класифікації [48] та використовується для оцінювання ймовірності належності об'єкта до одного з двох класів. У контексті аналізу мережевого трафіку логістична регресія є корисною насамперед як порівняльна базова модель, що дозволяє оцінити, наскільки добре лінійна межа розділення описує відмінності між нормальним та шкідливим трафіком. Перевагами цього методу є відносна простота, швидкість навчання, стійкість до великої кількості спостережень та можливість інтерпретації вагових коефіцієнтів. Разом із тим логістична регресія обмежена припущенням про лінійний характер межі класифікації в просторі ознак, тому її ефективність може знижуватися у випадках, коли структура даних є суттєво нелінійною. Незважаючи

на це, використання логістичної регресії у роботі є виправданим, оскільки вона дає змогу отримати базовий орієнтир для порівняння більш складних моделей [92, 107, 108, 110, 111, 115].

Другим методом є метод опорних векторів (Support Vector Machine, SVM) [12], який належить до потужних алгоритмів класифікації і широко застосовується в задачах розпізнавання образів, аналізу аномалій та кібербезпеки. Основна ідея цього методу полягає у побудові такої гіперплощини, яка максимально розділяє об’єкти різних класів у просторі ознак. Однією з ключових переваг SVM є можливість використання різних функцій ядра, що дозволяє будувати як лінійні, так і нелінійні межі класифікації. Для задач виявлення атак це є важливим, оскільки мережевий трафік часто має складну структуру, яку не можна коректно описати простою лінійною моделлю. Метод опорних векторів є чутливим до масштабів ознак, тому його застосування логічно поєднується з попередньою стандартизацією даних. Сильними сторонами SVM є висока здатність до узагальнення та ефективність у випадках складного розділення класів, однак обчислювальна складність цього методу може зростати для великих вибірок. У цьому дослідженні SVM використовується як один із ключових нелінійних класифікаторів для оцінювання ефективності побудованого конвеєра підготовки даних.

Наступним методом є дерево рішень, яке являє собою алгоритм, що виконує класифікацію шляхом послідовного розбиття простору ознак на підобласті на основі правил виду «якщо – то» [49]. Модель дерева рішень є доцільною для застосування в задачах виявлення атак завдяки наочності та інтерпретованості результатів: її функціонування можна подати у вигляді ієрархічної структури рішень, у якій кожен вузол відповідає перевірці певної ознаки, а кожна гілка – одному з можливих варіантів переходу. Перевагами дерева рішень є здатність працювати з нелінійними залежностями, відносно невисокі вимоги до попередньої обробки та зручність інтерпретації правил класифікації. Водночас одним із головних недоліків цього алгоритму є схильність до перенавчання, особливо при побудові занадто глибоких дерев. У зв’язку з цим у роботі дерево рішень використовується не лише як самостійний класифікатор, а й як важливий

проміжний етап для порівняння з ансамблевими методами, зокрема з випадковим лісом.

Четвертим методом є випадковий ліс (Random Forest), який належить до ансамблевих алгоритмів машинного навчання [10]. Його ідея полягає у побудові множини дерев рішень на різних підмножинах даних та ознак із подальшим агрегуванням їхніх рішень. Завдяки цьому зменшується варіативність окремих дерев і підвищується стійкість моделі до перенавчання. Для задач виявлення атак випадковий ліс є особливо перспективним, оскільки він здатний ефективно працювати з великим числом ознак, ураховувати складні нелінійні взаємозв'язки між змінними та забезпечувати високу якість класифікації навіть у випадках складної структури даних. Крім того, випадковий ліс дозволяє оцінювати важливість ознак, що робить його корисним не лише як інструмент класифікації, а й як засіб аналізу інформативності даних. У межах дослідження цей алгоритм розглядається як один із найбільш практично значущих методів для задач багатокласового розпізнавання атак у мережевому трафіку.

Останнім із розглянутих методів є метод k-найближчих сусідів (K-Nearest Neighbors, KNN) [11] – цей алгоритм базується на припущенні, що об'єкти, близькі один до одного у просторі ознак, з великою ймовірністю належать до одного класу. Класифікація нового об'єкта виконується на основі аналізу міток його найближчих сусідів у навчальній вибірці. Для задач аналізу мережевого трафіку KNN є цікавим з точки зору дослідження локальної структури простору ознак та здатності виявляти подібність між мережевими потоками. Його перевагою є простота реалізації та відсутність явного етапу параметричного навчання, однак ефективність KNN суттєво залежить від масштабу ознак, вибору кількості сусідів і структури простору даних. Саме тому використання цього алгоритму є виправданим лише після стандартизації ознак і бажано після зниження розмірності, що зменшує вплив «прокляття розмірності». У цьому дослідженні KNN застосовується як представник методів класифікації на основі відстані та дає змогу оцінити, наскільки добре підготовлений простір ознак зберігає локальну структуру, корисну для розрізнення класів.

Сукупність обраних методів дозволяє охопити різні принципи класифікації мережевого трафіку. Логістична регресія представляє лінійні моделі, SVM – нелінійні моделі з можливістю гнучкого налаштування межі розділення, дерево рішень і випадковий ліс – ієрархічні та ансамблеві моделі, а KNN – методи, що ґрунтуються на локальній близькості об’єктів у просторі ознак. Така комбінація є доцільною для всебічного дослідження задачі інтелектуального моніторингу трафіку та виявлення атак, оскільки дозволяє порівняти моделі різної природи за єдиних умов підготовки даних і оцінювання якості.

Важливо зазначити, що у роботі вибір методів машинного навчання зумовлений не лише їхньою популярністю, а й відповідністю поставленим дослідницьким завданням. По-перше, обрані алгоритми охоплюють обидві постановки задачі дослідження: логістична регресія та SVM використовуються для бінарного виявлення атак, тоді як дерево рішень, випадковий ліс і KNN застосовуються для багатокласового розпізнавання їх типів. По-друге, вони дозволяють оцінити ефективність різних методів класифікації на основі одного й того самого набору даних після попередньої обробки. По-третє, ці методи забезпечують різний рівень складності моделей, що дає змогу простежити, як збільшення гнучкості алгоритму впливає на якість виявлення атак і на узагальнювальну здатність. Саме тому їх використання у межах дослідження створює достатньо широку методичну основу для подальшого експериментального порівняння.

Таким чином, у підрозділі обґрунтовано вибір моделей машинного навчання, що застосовуються в межах запропонованого методу для бінарного виявлення атак і багатокласового розпізнавання їх типів. Логістична регресія та метод опорних векторів використовуються для відокремлення нормального трафіку від атакувального, тоді як дерево рішень, випадковий ліс і метод k-найближчих сусідів застосовуються для розпізнавання конкретних типів атак. Такий набір моделей дозволяє порівняти лінійні, нелінійні, ієрархічні, ансамблеві та метричні моделі класифікації мережевого трафіку за єдиних умов підготовки даних [10-12, 48, 49]. Подальший аналіз якості цих моделей потребує формалізації критеріїв оцінювання

та організації коректного експериментального дизайну, що розглядається в наступному підрозділі [107-120].

2.3.5 Метрики оцінювання якості моделей та організація експериментального дослідження

Після вибору моделей машинного навчання важливим етапом дослідження є визначення засобів оцінювання якості побудованих моделей і формування коректної схеми проведення експериментів. У задачах інтелектуального моніторингу трафіку та виявлення атак таке оцінювання недостатньо зводити лише до обчислення загальної точності класифікації, оскільки мережевий трафік характеризується складною структурою, нерівномірним розподілом класів і різною «вартістю» помилок. Для систем виявлення атак особливо небезпечними є хибнонегативні рішення, за яких атака класифікується як нормальний трафік, оскільки саме такі помилки означають фактичний пропуск інциденту безпеки. Тому організація експериментального дослідження в роботі ґрунтується на використанні кількох взаємодоповнювальних метрик та дотриманні методично коректної процедури поділу даних, навчання моделей і перевірки їхньої узагальнювальної здатності.

Експериментальне дослідження в роботі організовано окремо для бінарної та багатокласової класифікації. Така схема організації експериментального дослідження дає змогу оцінити поведінку моделей на різних рівнях складності класифікації та зіставити їхню ефективність як у задачі загального розмежування нормального й атакувального трафіку, так і в задачі деталізації типів атак. Для кожної постановки задачі формується відповідна вибірка даних, після чого виконується її поділ на навчальну та тестову частини [46, 51]. Навчальна вибірка використовується для побудови моделей і налаштування елементів конвеєра попередньої обробки, тоді як тестова призначена для контролю здатності моделей узагальнювати знання на нові дані. Такий поділ є базовою вимогою коректного

експериментального дизайну, оскільки дозволяє уникнути оцінювання моделі на тих самих даних, на яких вона навчалась [121, 122, 124].

У межах дослідження застосовується розбиття даних на навчальну та тестову підмножини з фіксованими параметрами випадковості, що забезпечує відтворюваність результатів. Використання фіксованого значення `random_state` у процедурах поділу вибірки дає змогу зберегти сталі умови проведення експерименту та порівнювати моделі в однаковому середовищі [51]. З методологічної точки зору це є важливим, оскільки зміна складу тренувальної та тестової частин може впливати на значення метрик, особливо у випадках складної структури даних або наявності рідкісних класів. Таким чином, контроль випадковості є невід'ємною частиною організації достовірного дослідження.

Однак одноразове розбиття вибірки на train/test не завжди дає достатньо надійну оцінку якості моделі, оскільки результат може залежати від конкретного складу підвибірок. Для зменшення цього ефекту в роботі застосовується перехресна перевірка (cross-validation), яка дозволяє отримати більш стійкі й статистично обґрунтовані оцінки якості класифікації [50, 51]. Суть цієї процедури полягає у багаторазовому розподілі навчальної вибірки на декілька частин – фолдів, коли модель послідовно навчається на частині даних і перевіряється на решті. У дослідженні використовується 5-fold cross-validation, за якої навчальна множина поділяється на п'ять фрагментів, і на кожній ітерації один із них використовується для валідації, а решта – для навчання. Після завершення всіх ітерацій обчислюється середнє значення метрики, що дає узагальнену оцінку якості моделі [91, 121, 122].

Застосування крос-валідації в задачах виявлення атак має особливе значення, оскільки дозволяє оцінити стійкість моделі до варіацій у вибірці та зменшити ризик випадково завищених або занижених результатів. Якщо модель демонструє близькі значення метрик на різних фолдах, це свідчить про її кращу узагальнювальну здатність і стабільність рішень. Водночас велика варіативність результатів між фолдами може вказувати на нестійкість моделі, перенавчання або високу чутливість до структури підвибірки. У контексті даного дослідження саме середні

значення метрик за фолдами розглядаються як одна з основних підстав для порівняння ефективності різних алгоритмів машинного навчання.

Для кількісного оцінювання моделей у роботі використовується набір стандартних метрик класифікації, які взаємно доповнюють одна одну [46, 51]. Однією з базових характеристик є accuracy, тобто частка правильно класифікованих об’єктів серед усіх спостережень. Ця метрика є зручною для загальної оцінки моделі, однак у задачах аналізу мережевого трафіку її недостатньо використовувати як єдиний критерій, оскільки за наявності дисбалансу класів висока accuracy може досягатися навіть при незадовільному виявленні атак. Наприклад, якщо нормальний трафік істотно домінує у вибірці, модель може демонструвати високу точність, просто відносячи більшість об’єктів до класу «BENIGN», але при цьому пропускати значну частину атакувальної активності. Тому accuracy у роботі розглядається лише як одна з допоміжних узагальнених характеристик [97-103, 121-124].

Більш інформативними для задач виявлення атак є метрики precision, recall та F1-score. Метрика precision характеризує частку об’єктів, які дійсно належать до певного класу, серед усіх тих, що були віднесені моделлю до цього класу. У контексті IDS вона дозволяє оцінити, наскільки часто система помиляється, сигналізуючи про атаку там, де її насправді немає. Високе значення precision означає, що модель формує менше хибнопозитивних рішень, а отже зменшує кількість необґрунтованих спрацювань. Водночас надмірна орієнтація лише на precision може призвести до того, що система стане «обережною» і почне пропускати частину реальних атак.

Метрика recall, або повнота, відображає частку правильно виявлених об’єктів певного класу серед усіх об’єктів цього класу, що реально присутні у вибірці. Для задач виявлення атак саме recall має особливу вагу, оскільки безпосередньо характеризує здатність системи не пропускати шкідливу активність. Низьке значення recall для класу атак означає високу частку хибнонегативних рішень, що є критично небажаним для IDS. Тому у межах дослідження recall розглядається як

одна з найважливіших метрик, особливо для бінарної постановки, де головною метою є надійне відокремлення небезпечного трафіку від нормального.

Для узгодження precision і recall використовується F1-score, що являє собою гармонійне середнє між цими двома метриками. Значення F1-score є високим лише у випадку, коли модель одночасно демонструє добру точність і добру повноту. У дослідженні ця метрика є важливою для комплексної оцінки моделей, оскільки дозволяє уникнути однобічного трактування результатів. Для задач із незбалансованими класами F1-score є більш інформативним критерієм, ніж accuracy, оскільки враховує якість виявлення менш представлених класів і відображає загальну збалансованість рішень моделі.

У межах дисертаційного дослідження чисельна оцінка ефективності моделей машинного навчання розглядається як кількісне подання результатів класифікації мережевого трафіку за допомогою системи взаємодоповнювальних метрик. Така оцінка дає змогу формалізовано порівнювати моделі між собою, визначати їх здатність до виявлення атакувального трафіку, аналізувати якість розпізнавання окремих типів атак і обґрунтовувати вибір найбільш ефективної конфігурації інформаційної технології.

Для бінарної класифікації додатково використовується ROC-AUC – площа під ROC-кривою, яка характеризує здатність моделі розрізняти два класи при різних значеннях порога прийняття рішення [18]. Ця метрика є корисною тим, що відображає якість моделі не лише в одній фіксованій точці, а в широкому діапазоні співвідношень між чутливістю і часткою хибнопозитивних рішень. Високе значення ROC-AUC свідчить про хорошу здатність моделі відокремлювати нормальний трафік від атак незалежно від конкретного порога класифікації. Для систем виявлення атак це є важливою характеристикою, оскільки в реальних умовах поріг спрацювання може бути адаптований залежно від політики безпеки, допустимого рівня хибних спрацювань і вимог до чутливості системи.

Для задач із нерівномірним розподілом класів додатково доцільним є використання Precision–Recall-кривих, які відображають співвідношення між точністю та повнотою моделі при зміні порога класифікації. На відміну від ROC-

подання, Precision–Recall-представлення є особливо інформативним у випадках, коли один із класів є менш представленим, оскільки безпосередньо фокусує увагу на якості виявлення позитивного класу [19]. У задачі виявлення атак це дозволяє точніше оцінити компроміс між кількістю хибних спрацювань і здатністю моделі виявляти атакувальний трафік [102, 104].

Окреме місце в оцінюванні результатів займає матриця помилок (confusion matrix), яка дає змогу детально проаналізувати розподіл правильних і помилкових рішень моделі за класами [51]. Для бінарної задачі матриця помилок дозволяє побачити кількість істиннопозитивних, істиннонегативних, хибнопозитивних і хибнонегативних рішень. Для багатокласової постановки вона відображає, між якими саме класами виникає найбільша плутанина. Це важливо для аналізу атак, оскільки дає змогу визначити, які типи атакувальної активності моделі схильні сплутувати між собою та які класи є найбільш проблемними для розпізнавання. Таким чином, матриця помилок у роботі використовується не лише як інструмент оцінювання, а й як засіб інтерпретації поведінки моделей [98, 103].

Організація експериментального дослідження передбачає, що всі етапи попередньої обробки даних мають бути включені до єдиного коректного конвеєра, де параметри кожного перетворення визначаються лише на навчальних даних. Це стосується стандартизації, зниження розмірності та балансування класів. Така послідовність виконання процедур обробки даних дає змогу запобігти витоку інформації з тестової або валідаційної частини у процес навчання. У разі використання крос-валідації ця вимога реалізується окремо для кожного фолду: кожне перетворення налаштовується лише на поточній тренувальній підмножині, після чого застосовується до відповідної валідаційної частини. Дотримання цього принципу є критично важливим для коректного порівняння моделей та достовірності отриманих результатів.

Для забезпечення відтворюваності експериментів у роботі використовуються фіксовані параметри випадковості у процедурах поділу даних, балансування класів і навчання тих моделей, для яких випадковість є складовою внутрішньої роботи алгоритму. Це дозволяє повторити експеримент за однакових умов та отримати

узгоджені значення метрик, що відповідає вимогам наукового дослідження. Крім того, використання єдиної схеми підготовки даних і оцінювання для всіх розглянутих алгоритмів забезпечує справедливість порівняння та дозволяє пов’язати відмінності у результатах саме з властивостями моделей, а не з різними умовами експерименту.

Таким чином, у межах дослідження визначено основні принципи оцінювання якості моделей та організації експериментального дослідження. Використання поділу даних на навчальну й тестову частини, п’ятикратної перехресної перевірки та системи взаємодоповнювальних метрик – accuracy, precision, recall, F1-score, ROC-AUC, Precision–Recall-кривих і матриць помилок – дозволяє комплексно оцінити ефективність алгоритмів машинного навчання в задачах бінарного виявлення атак і багатокласового розпізнавання їх типів. Запропонована схема оцінювання забезпечує методично обґрунтовану основу для порівняльного аналізу моделей і подальшого формування узагальненого алгоритму методу підготовки даних і класифікації мережевого трафіку.

2.4 Узагальнений алгоритм методу підготовки даних, класифікації та оцінювання мережевого трафіку для СВА

У попередніх підрозділах другого розділу було послідовно розглянуто постановку задачі дослідження, характеристику набору даних, процедури очищення та стандартизації, процедури аналізу ознак і зниження розмірності, процедури формування та балансування вибірок, вибір моделей машинного навчання та способи оцінювання їхньої ефективності. Оскільки зазначені етапи не є ізольованими, а утворюють єдину послідовність взаємопов’язаних процедур, у межах дослідження доцільно подати їх у вигляді узагальненого алгоритму. Такий алгоритм відображає логіку переходу від вихідних даних мережевого трафіку до побудови й оцінювання моделей машинного навчання та визначає методично коректну послідовність виконання експерименту.

Узагальнений алгоритм дослідження (рис. 2.10) та попередньої обробки даних розглядається як цілісний експериментальний конвеєр, у межах якого кожен наступний етап спирається на результати попереднього [8, 14, 17, 20, 21, 46, 50, 51]. Його використання забезпечує узгодженість усіх процедур обробки даних, відтворюваність експериментів і дотримання принципу відсутності витоку інформації між навчальними та контрольними підмножинами. Вхідними даними алгоритму є окремі піднабори мережевих потоків CIC-IDS2017 із відповідними ознаками та мітками класів. Вихідними результатами є підготовлені вибірки для бінарної та багатокласової класифікації, навчені моделі машинного навчання і результати оцінювання їхньої ефективності [87-91, 97-102, 112-118].

Першим етапом алгоритму є об’єднання та первинна підготовка даних. На цій стадії окремі піднабори мережевого трафіку інтегруються в єдину структуру, узгоджуються назви ознак, формати їх подання та загальна логіка представлення атрибутів. У результаті формується цілісний набір даних, придатний для подальшої аналітичної обробки. Значення цього етапу полягає в тому, що він створює єдину вхідну основу для всіх наступних перетворень і усуває неузгодженості, які могли б вплинути на достовірність результатів.

Другий етап пов’язаний з очищенням даних і приведенням їх до коректного числового вигляду. На цьому кроці виявляються й усуваються пропущені, нескінченні та некоректні значення, видаляються дублікати, а також контролюється наявність малоінформативних ознак, що не роблять суттєвого внеску у розрізнення класів. Після завершення цього етапу формується очищена матриця ознак, придатна для подальшого математичного аналізу. Саме цей крок забезпечує числову коректність даних і знижує ризик викривлення результатів навчання моделей через технічні дефекти вихідного набору.

На третьому етапі здійснюється формування цільової змінної відповідно до постановки задачі. Для бінарної класифікації всі записи нормального трафіку об’єднуються в один клас, а всі записи, що відповідають атакувальній активності, – в інший. Для багатокласової класифікації зберігається поділ за окремими типами атак, що включені до дослідження. На цьому ж етапі визначається склад

дослідницьких вибірок для кожного сценарію класифікації, що забезпечує узгодженість подальших процедур навчання й оцінювання моделей.

Четвертим етапом є поділ даних на навчальну та тестову підмножини. Цей крок має принципове методологічне значення, оскільки саме після нього всі наступні перетворення повинні виконуватися за схемою, коли параметри кожної процедури визначаються лише на навчальних даних. Дотримання такої послідовності процедур дає змогу запобігти витоку інформації з тестової вибірки у процес налаштування моделі та забезпечити об’єктивність оцінювання її узагальнювальної здатності.

Узагальнений алгоритм методу наведено на рис. 2.10.

П’ятий етап передбачає стандартизацію ознак. На цьому кроці числові характеристики мережевого трафіку приводяться до узгодженого масштабу, що є необхідним для коректного застосування алгоритмів, чутливих до величин ознак, а також для подальшого виконання зниження розмірності. Параметри стандартизації визначаються лише на навчальній частині вибірки, після чого відповідне перетворення застосовується до тестових даних без повторного налаштування. У результаті усувається непропорційний вплив ознак із більшими числовими масштабами на процес побудови моделей.

Шостий етап пов’язаний з аналізом структури простору ознак і зниженням його розмірності. На цій стадії досліджуються взаємозв’язки між характеристиками мережевого трафіку та формується компактніше представлення даних, яке зберігає основну частину корисної інформації. Використання PCA дозволяє перейти від початкового корельованого простору ознак до нового простору головних компонент, у якому зменшено мультиколінеарність і надлишковість атрибутів. Це створює більш стійкі умови для подальшого навчання моделей і спрощує обробку даних у задачах класифікації.

Сьомим етапом є балансування класів, яке здійснюється лише на навчальній частині вибірки. Спосіб балансування визначається постановкою задачі. У бінарному випадку формується зіставне представлення нормального та атакувального трафіку, що усуває зміщення моделей у бік класу більшості. У

багатокласовій постановці забезпечується більш рівномірне представлення окремих типів атак, зокрема шляхом застосування методів синтетичного збільшення класів меншості. Реалізація цього етапу створює кращі умови для навчання моделей і підвищує їх здатність розпізнавати рідкісні або слабо представлені класи.

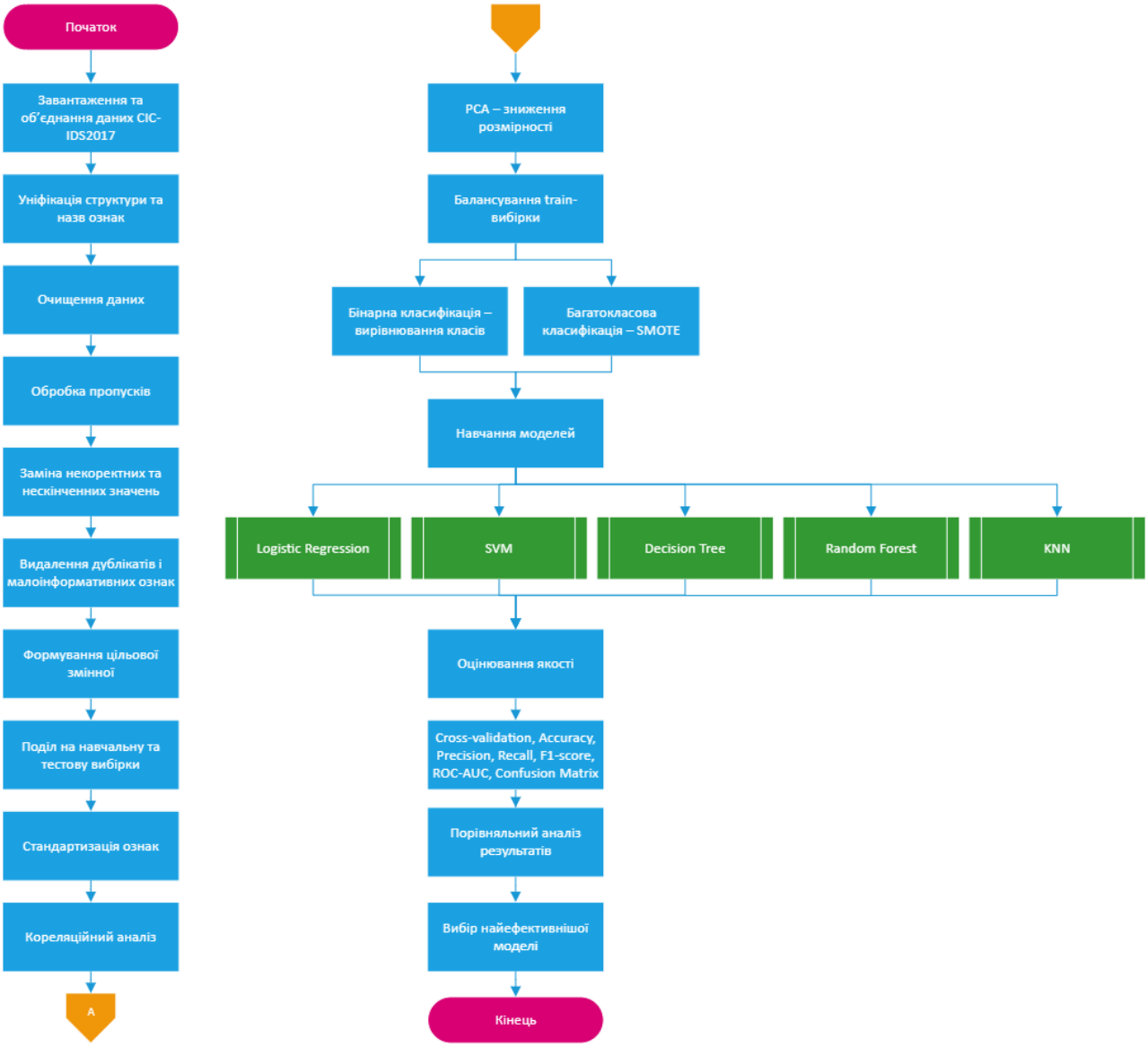


Рисунок 2.10 – Узагальнений алгоритм методу підготовки даних, класифікації та оцінювання мережевого трафіку для СВА

Восьмий етап охоплює навчання моделей машинного навчання на підготовлених вибірках. На цьому кроці до даних, що пройшли очищення,

стандартизацію, зниження розмірності та балансування, послідовно застосовуються обрані алгоритми класифікації. Використання єдиної схеми попередньої обробки для всіх моделей забезпечує коректність їх подальшого порівняння та дозволяє пов’язувати відмінності в результатах саме з властивостями алгоритмів, а не з різними умовами підготовки даних.

Завершальним, дев’ятим етапом алгоритму є оцінювання ефективності побудованих моделей. Для цього використовуються тестова вибірка, крос-валідація та система взаємодоповнювальних метрик якості класифікації. Застосування такої системи оцінювання дає змогу проаналізувати не лише загальну точність моделей, а й їхню здатність виявляти атаки, розрізняти окремі типи атакувальної активності та зберігати стабільність результатів на різних підвибірках даних. Отримані показники формують основу для подальшого порівняльного аналізу моделей у наступних розділах дисертаційної роботи.

Отже, узагальнений алгоритм об’єднує процедури очищення, уніфікації, стандартизації, аналізу ознак, зниження розмірності, формування та балансування вибірок, навчання моделей машинного навчання й оцінювання їхньої якості в єдину методично завершену схему. Реалізація такого алгоритму забезпечує ефективну організацію експериментального дослідження, створює основу для об’єктивного порівняння моделей машинного навчання та формує методичне підґрунтя для подальшого аналізу результатів, наведених у наступних розділах дисертації.

Висновки до розділу 2

У другому розділі сформовано методичну основу дослідження інтелектуального виявлення атак у мережевому трафіку на основі моделей машинного навчання. Обґрунтовано постановку задачі у бінарній та багатокласовій формах, визначено вимоги до даних, обґрунтовано вибір експериментальної основи дослідження, розроблено метод підготовки даних, класифікації та оцінювання мережевого трафіку для СВА.

Встановлено, що для задач виявлення атак доцільним є використання двох взаємопов’язаних постановок: бінарної, орієнтованої на відокремлення нормального трафіку від атакувального, та багатокласової, спрямованої на розпізнавання конкретних типів атак. Така організація дослідження забезпечує можливість одночасного аналізу як факту наявності атакувальної активності, так і її деталізації за типами атак, що підвищує прикладну значущість отриманих результатів.

Обґрунтовано використання набору даних CIC-IDS2017 як репрезентативної експериментальної бази. Показано, що цей набір даних дозволяє реалізувати обидві постановки задачі, проте характеризується дисбалансом класів, нерівномірною представленістю атак, надлишковістю ознак, наявністю кореляційних залежностей і технічних дефектів, що зумовлює необхідність спеціальної організації процедур підготовки даних, формування вибірок і балансування класів.

Розроблено процедури очищення, уніфікації та стандартизації даних мережевого трафіку, які включають інтеграцію піднаборів, усунення пропущених, нескінченних і некоректних значень, видалення дублікатів, контроль малоінформативних ознак і приведення числових характеристик до узгодженого масштабу. Показано, що ці процедури є необхідною передумовою коректного формування матриці ознак, подальшого навчання моделей машинного навчання та достовірного оцінювання результатів класифікації.

На основі кореляційного аналізу підтверджено наявність мультиколінеарності та надлишковості у просторі ознак мережевого трафіку. У зв’язку з цим обґрунтовано застосування методу головних компонент, який дозволяє зменшити розмірність даних, перейти до компактнішого ортогонального представлення ознак і зберегти основну частину інформативності вихідного простору. Це створює більш стійкі умови для подальшого формування вибірок, балансування класів і навчання моделей машинного навчання.

Обґрунтовано процедуру формування та балансування вибірок для бінарної і багатокласової класифікації. Для бінарної постановки сформовано цільову змінну, що забезпечує поділ трафіку на нормальний та атакувальний, а для багатокласової

– збережено деталізовану інформацію про конкретні типи атак. Для зменшення статистичного зміщення моделей у бік класів більшості використано диференційовану схему балансування: для бінарної класифікації – випадкове зменшення класу більшості, для багатокласової – метод SMOTE, що забезпечує синтетичне збільшення класів меншості та підвищує чутливість моделей до рідкісних типів атак.

Обґрунтовано вибір моделей машинного навчання для подальшого дослідження. Для бінарного виявлення атак визначено логістичну регресію та метод опорних векторів, а для багатокласового розпізнавання типів атак – дерево рішень, випадковий ліс і метод k-найближчих сусідів. Використання моделей різної природи забезпечує методичну основу для їх подальшого порівняння за єдиних умов підготовки даних, навчання та оцінювання.

Сформовано систему оцінювання якості моделей, яка базується на поділі вибірки на навчальну й тестову частини, застосуванні п’ятикратної перехресної перевірки та використанні взаємодоповнювальних метрик accuracy, precision, recall, F1-score, ROC-AUC, Precision–Recall-кривих і матриць помилок. Показано, що така схема оцінювання дозволяє комплексно аналізувати результати класифікації з урахуванням специфіки задач виявлення атак, де критичне значення має мінімізація хибнонегативних рішень і забезпечення стабільного розпізнавання атакувального трафіку.

Розроблено узагальнений алгоритм методу, який інтегрує основні етапи дослідження: інтеграцію та очищення даних, стандартизацію ознак, аналіз кореляційної структури, зниження розмірності, формування та балансування вибірок, навчання моделей машинного навчання й оцінювання їх ефективності. Запропонований алгоритм забезпечує цілісність дослідження, відтворюваність експерименту, недопущення витоку інформації між навчальними та контрольними підмножинами, а також створює основу для подальшої програмної реалізації та експериментальної перевірки ефективності інформаційної технології в наступних розділах дисертації.

РОЗДІЛ 3. ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ ТРАФІКУ КОМП'ЮТЕРНОЇ МЕРЕЖІ

3.1 Загальна архітектура інформаційної технології інтелектуального моніторингу трафіку

Розроблена інформаційна технологія інтелектуального моніторингу трафіку комп'ютерної мережі для СВА призначена для автоматизованої обробки даних мережевих потоків, формування ознакового простору, побудови інформаційних моделей трафіку та виявлення атак із використанням методів машинного навчання. Її архітектура орієнтована на забезпечення повного циклу роботи з мережевими даними: від отримання та попередньої підготовки вхідної інформації до класифікації трафіку, оцінювання результатів і подальшого використання отриманих рішень у системах виявлення атак [144, 150].

Необхідність побудови саме цілісної інформаційної технології зумовлена тим, що ефективність інтелектуального моніторингу мережевого трафіку визначається не лише вибором окремої моделі машинного навчання, а й узгодженістю всіх етапів обробки даних. До таких етапів належать очищення та уніфікація вхідних даних, усунення некоректних і пропущених значень, стандартизація числових ознак, зменшення надлишковості ознакового простору, формування навчальних і тестових вибірок, балансування класів, навчання моделей, їх валідація та комплексне оцінювання результатів класифікації. Тому запропонована архітектура розглядається як сукупність взаємопов'язаних функціональних блоків, кожен з яких виконує окрему задачу в загальному процесі інтелектуального аналізу мережевого трафіку [141].

Загальна архітектура інформаційної технології включає такі основні функціональні компоненти:

– блок отримання та інтеграції даних мережевого трафіку;

– блок первинного аналізу структури даних;

– блок очищення, уніфікації та попередньої обробки;

– блок формування ознакового простору;

– блок зниження розмірності та підготовки інформаційного подання трафіку;

– блок формування вибірок для бінарної та багатокласової класифікації;

– блок балансування класів у сформованих навчальних вибірках;

– блок навчання моделей машинного навчання;

– блок валідації, оцінювання та візуалізації результатів;

– блок інтерпретації результатів класифікації та формування рішення щодо належності трафіку до відповідного класу.

Вхідною інформацією для інформаційної технології є дані мережевого трафіку, подані у вигляді набору ознак, що характеризують параметри мережевих потоків. У межах дисертаційного дослідження як експериментальну основу використано набір даних CIC-IDS2017, що дає змогу досліджувати задачу виявлення атак як у бінарній, так і в багатокласовій постановках. На початковому етапі окремі частини набору даних завантажуються, інтегруються в єдину структуру та підлягають первинному аналізу для визначення складу ознак, типів даних, наявності дубльованих записів, пропущених значень, нескінченних величин і нерівномірності розподілу класів.

Блок попередньої обробки забезпечує перетворення вхідних даних у придатну для машинного навчання форму. На цьому етапі виконуються очищення даних, усунення технічних дефектів, приведення назв і структури ознак до узгодженого вигляду, видалення або обробка некоректних значень, стандартизація числових характеристик і підготовка цільових змінних. У результаті формується впорядкований ознаковий простір, який відображає характеристики мережевого трафіку та може бути використаний для побудови класифікаційних моделей.

Наступним компонентом архітектури є блок формування інформаційних моделей мережевого трафіку. Під інформаційною моделлю в межах цієї роботи розуміється підготовлене подання мережевого трафіку у вигляді сукупності ознак, цільових міток, навчальних і тестових вибірок, що забезпечують можливість подальшого навчання та оцінювання моделей машинного навчання. Виконання такої послідовності процедур дає змогу перетворити частково підготовлений набір

мережевих даних у формалізоване подання, придатне для подальшої автоматизованої класифікації.

Особливістю запропонованої інформаційної технології є підтримка двох взаємопов’язаних режимів аналізу мережевого трафіку. Перший режим призначений для бінарного виявлення атак і передбачає класифікацію трафіку на два класи: нормальний трафік і атакувальна активність. Другий режим орієнтований на багатокласове розпізнавання типів атак і забезпечує деталізацію атакувального трафіку за окремими класами загроз у межах окремої постановки задачі класифікації. Така архітектурна побудова дає змогу поєднати задачу первинного виявлення факту атаки із задачею подальшої ідентифікації її типу.

Блок навчання моделей машинного навчання реалізує побудову класифікаторів для кожного з режимів функціонування інформаційної технології. Для бінарного аналізу використовуються моделі, орієнтовані на відокремлення нормального мережевого трафіку від атакувального. Для багатокласового аналізу застосовуються моделі, які забезпечують розпізнавання конкретних типів атак. Перед навчанням моделей здійснюється поділ даних на навчальні й тестові вибірки, а для навчальних вибірок із вираженим дисбалансом класів застосовуються процедури балансування, що дозволяє зменшити вплив нерівномірного представлення різних категорій трафіку.

Блок оцінювання та візуалізації результатів забезпечує комплексну перевірку ефективності побудованих моделей. Для цього використовуються кількісні метрики якості класифікації, матриці помилок, графічне подання результатів і порівняння моделей між собою. Наявність цього блоку є важливою складовою архітектури, оскільки дозволяє не лише отримати класифікаційне рішення, а й оцінити його достовірність, стійкість і практичну придатність для задач виявлення атак.

Вихідними результатами функціонування інформаційної технології є підготовлені інформаційні моделі мережевого трафіку, навчені моделі машинного навчання, оцінки їх ефективності та класифікаційні рішення щодо належності мережевого трафіку до нормального або атакувального класу, а також щодо типу

виявленої атаки. Отримані результати можуть бути використані як основа для побудови інтелектуальних компонентів систем виявлення атак, що забезпечують автоматизований аналіз мережевого трафіку та підтримку прийняття рішень у сфері кібербезпеки.

Загальна архітектура запропонованої інформаційної технології базується на послідовному поєднанні етапів підготовки даних, формування ознакового простору, побудови інформаційних моделей, навчання класифікаторів і оцінювання результатів.

3.2 Конвеєр обробки мережевого трафіку

Конвеєр обробки мережевого трафіку є ключовим функціональним елементом розробленої інформаційної технології, оскільки забезпечує послідовне перетворення вхідних даних у структуроване подання, придатне для подальшого навчання, валідації та оцінювання моделей машинного навчання. У межах дисертаційного дослідження конвеєр розглядається як впорядкована сукупність взаємопов’язаних етапів, що охоплюють інтеграцію даних, їх очищення, попередню обробку, формування ознакового простору, побудову цільових міток, формування вибірок і підготовку інформаційних моделей для задач виявлення атак.

На відміну від ізольованого використання окремих процедур попередньої обробки, конвеєрна схема обробки даних забезпечує цілісність, послідовність і відтворюваність процесу аналізу мережевого трафіку. Це є особливо важливим для задач інтелектуального виявлення атак, оскільки якість кінцевих результатів визначається не лише вибором алгоритму машинного навчання, а й коректністю підготовки даних на всіх попередніх етапах. Загальну схему конвеєра обробки мережевого трафіку та формування інформаційних моделей доцільно подати на рис. 3.1.

Узагальнено конвеєр обробки мережевого трафіку включає такі послідовні етапи: завантаження та інтеграцію даних, первинний аналіз структури набору даних, видалення дублікатів і технічних дефектів, очищення та уніфікацію ознак,

формування ознакового простору, стандартизацію числових характеристик, зниження розмірності ознакового простору, формування цільових міток класів, поділ даних на навчальну й тестову вибірки, балансування навчальних даних та побудову інформаційних моделей для подальшої класифікації.

Рисунок 3.1 – Конвеєр обробки мережевого трафіку та формування інформаційних моделей

Першим етапом конвеєра є завантаження та інтеграція даних мережевого трафіку. У роботі для цього використано набір даних CIC-IDS2017, який містить записи мережевих потоків, отримані для різних сценаріїв нормальної та атакувальної активності. Оскільки набір подано у вигляді окремих частин, на цьому етапі виконується їх об’єднання в єдину структуру даних.

Після інтеграції даних здійснюється первинний аналіз їх структури. Його метою є визначення складу ознак, типів даних, кількості записів, наявності цільових міток, а також виявлення потенційних проблем у наборі даних. На цьому етапі встановлюється наявність дубльованих записів, пропущених значень, нескінченних величин, некоректних типів даних і нерівномірного розподілу класів. Виконання такого аналізу дозволяє оцінити якість вихідних даних і визначити

необхідні процедури попередньої обробки.

Наступним етапом є видалення дублікатів і технічних дефектів, а також очищення й уніфікація даних. Цей етап передбачає усунення повторюваних записів, обробку пропущених і нескінченних значень, приведення назв ознак до узгодженого формату, а також виключення полів, що не мають суттєвого значення для подальшого аналізу. Реалізація цього етапу є необхідною умовою для побудови коректної інформаційної моделі, оскільки наявність технічних дефектів у наборі даних може спотворювати результати навчання моделей і знижувати достовірність їх оцінювання.

Після очищення виконується формування ознакового простору мережевого трафіку. На цьому етапі виділяється сукупність числових характеристик мережевих потоків, які використовуються як вхідні ознаки для моделей машинного навчання. Саме ознаковий простір є основою подальшого представлення мережевого трафіку, оскільки він концентрує інформацію про часові, статистичні та об’ємні параметри потоків, що можуть бути використані для виявлення закономірностей нормальної та аномальної активності.

Важливою складовою конвеєра є стандартизація числових ознак. Необхідність її застосування зумовлена тим, що характеристики мережевого трафіку можуть мати різні масштаби та діапазони значень. Без приведення ознак до узгодженого масштабу окремі параметри можуть непропорційно впливати на процес навчання моделей машинного навчання. Стандартизація забезпечує коректніше подання даних і підвищує ефективність подальшої класифікації.

У межах реалізованого конвеєра передбачено також етап зниження розмірності ознакового простору. Його призначення полягає у зменшенні надлишковості даних, зниженні впливу взаємозалежних характеристик та формуванні компактнішого подання мережевого трафіку. Це дозволяє зменшити обчислювальну складність подальшого аналізу і створити більш зручну основу для побудови інформаційних моделей.

Після формування підготовленого ознакового простору здійснюється побудова цільових міток класів для двох постановок задачі. Для задачі бінарного

виявлення атак формуються мітки, що дозволяють віднести запис до одного з двох класів: «нормальний трафік» або «атака». Для задачі багатокласового розпізнавання формуються мітки, що відображають конкретний тип атакувальної активності. Таким чином, на основі підготовленого масиву мережевих даних формуються окремі інформаційні моделі для бінарної та багатокласової постановок задачі.

Після цього виконується поділ даних на навчальну й тестову вибірки. Навчальна вибірка використовується для побудови моделей машинного навчання, тоді як тестова вибірка призначена для незалежного оцінювання їхньої якості.

Для навчальних вибірок із вираженим дисбалансом класів у конвеєрі передбачено етап балансування даних. Його застосування спрямоване на зменшення впливу нерівномірного представлення класів на процес навчання моделей машинного навчання. Балансування виконується лише для навчальних даних, що забезпечує коректність подальшого тестування та не спотворює оцінку якості моделей на незалежній вибірці.

Результатом виконання конвеєра є формування інформаційних моделей мережевого трафіку. У межах цієї роботи під інформаційною моделлю розуміється структуроване подання даних, що включає підготовлений ознаковий простір, цільові мітки класів, навчальні та тестові вибірки, а також параметри попередньої обробки, необхідні для відтворюваного навчання й оцінювання моделей машинного навчання.

У результаті функціонування конвеєра формуються дві інформаційні моделі. Перша інформаційна модель орієнтована на бінарне виявлення атак і відображає розподіл трафіку за класами «нормальний трафік» та «атака». Друга інформаційна модель орієнтована на багатокласове розпізнавання типів атак і забезпечує подання мережевого трафіку за окремими категоріями атакувальної активності. Наявність двох інформаційних моделей дозволяє реалізувати два взаємопов’язані режими функціонування інформаційної технології: первинне виявлення факту атаки та подальшу деталізацію типу загрози.

Конвеєр обробки мережевого трафіку забезпечує послідовне, узгоджене й

відтворюване перетворення вхідних даних у структуровані інформаційні моделі, придатні для застосування методів машинного навчання. Його реалізація дозволяє зменшити вплив технічних дефектів і дисбалансу класів, забезпечити узгодженість бінарної та багатокласової постановок задачі та створити основу для подальшої програмної реалізації навчання, валідації й оцінювання моделей класифікації мережевого трафіку.

3.3 Двоетапна схема класифікації мережевого трафіку у системах виявлення атак

Однією з особливостей розробленої інформаційної технології є підтримка двох взаємопов’язаних постановок задачі класифікації: бінарного виявлення атак і багатокласового розпізнавання їх типів. Запропонована організація аналізу дає змогу поєднати первинне визначення факту наявності атакувальної активності з подальшою деталізацією характеру виявленої загрози. У межах дисертаційного дослідження така логіка реалізується як двоетапна схема інтелектуального аналізу мережевого трафіку [140].

Перший етап призначений для відокремлення нормальної мережевої активності від атакувальної. На цьому етапі записи трафіку класифікуються за двома узагальненими класами: «нормальний трафік» та «атака». Основне завдання такого аналізу полягає у формуванні первинного рішення щодо наявності або відсутності ознак шкідливої поведінки. Така постановка є важливою для систем виявлення атак, оскільки дозволяє оперативно ідентифікувати потенційно небезпечні мережеві потоки без необхідності попереднього визначення конкретного типу загрози.

Для реалізації першого етапу формується бінарне подання даних. Цільові мітки класів приводяться до двох категорій, що дозволяє навчати класифікатори, орієнтовані на виявлення факту атакувальної активності. У дисертаційній роботі для цієї постановки використовуються моделі машинного навчання, призначені для бінарної класифікації мережевого трафіку.

Другий етап спрямований на деталізацію виявленої атакувальної активності. На цьому рівні формується багатокласове подання даних, у якому цільові мітки відповідають окремим типам атак або категоріям мережевої активності. Завданням такого аналізу є визначення конкретного класу загрози, що дозволяє не лише зафіксувати факт атаки, а й отримати додаткову інформацію про її характер для подальшого реагування на інцидент.

На відміну від бінарного подання, багатокласова інформаційна модель зберігає деталізацію за типами атак. Вона використовується для навчання класифікаторів, здатних розрізняти окремі категорії атакувальної активності. У результаті багатокласова класифікація забезпечує більш змістовну інтерпретацію результатів моніторингу та підвищує практичну цінність отриманих рішень для систем виявлення атак.

Важливо зазначити, що двоетапна схема в межах цієї роботи не обов’язково означає жорстку послідовність програмного запуску двох моделей. Вона відображає логіку побудови інформаційної технології для двох взаємопов’язаних задач: первинного виявлення факту атаки та подальшого розпізнавання її типу. Така організація функціонування підвищує функціональну повноту розробленої технології та робить її придатною як для загального моніторингу безпеки мережі, так і для підтримки прийняття рішень під час аналізу інцидентів.

Реалізація цієї схеми потребує формування окремих вибірок і цільових міток для кожного режиму аналізу. У бінарній постановці дані подаються у вигляді двох класів, що дозволяє оцінити здатність моделей відокремлювати нормальну активність від шкідливої. У багатокласовій постановці формується детальніший простір класів, який дає змогу дослідити здатність моделей розрізняти окремі типи атак. Відповідно, для кожного режиму виконуються власні процедури підготовки вибірок, навчання моделей і оцінювання результатів.

Застосування двоетапної схеми також підвищує інтерпретованість класифікаційних рішень. Результат першого етапу дає відповідь на питання, чи містить трафік ознаки атакувальної активності. Результат другого етапу уточнює, до якого типу атаки або категорії мережевої активності належить відповідний

запис. Унаслідок цього інформаційна технологія забезпечує не лише виявлення факту атаки, а й формування більш змістовного результату для подальшого аналізу.

У практичному контексті така схема може бути використана як основа для побудови інтелектуального компонента системи виявлення атак. На першому рівні система може здійснювати виявлення підозрілої активності, а на другому – деталізувати її тип для подальшого реагування. Це дає змогу підвищити ефективність аналізу мережевого трафіку, зменшити навантаження на фахівця з кібербезпеки та забезпечити більш обґрунтовану підтримку прийняття рішень.

Узагальнюючи наведене, двоетапна схема класифікації мережевого трафіку може розглядатися як важлива складова запропонованої інформаційної технології. Вона забезпечує поєднання первинного бінарного виявлення атак із подальшим багатокласовим розпізнаванням їх типів, що підвищує функціональну повноту процесу інтелектуального моніторингу та створює основу для подальшої програмної реалізації відповідних моделей машинного навчання.

3.4 Мета та завдання програмної реалізації інформаційної технології

У межах дисертаційного дослідження, присвяченого розробленню інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак, важливим етапом є її практична програмна реалізація. У попередніх розділах розглянуто теоретичні засади побудови систем виявлення атак, проаналізовано сучасні методи, моделі та засоби інтелектуального аналізу мережевого трафіку, а також обґрунтовано методичні рішення щодо підготовки даних. У цьому розділі увагу зосереджено на безпосередньому програмному втіленні запропонованої інформаційної технології.

Метою програмної реалізації є створення програмного модуля, який забезпечує функціонування основних складових інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі та дозволяє використовувати її в задачах автоматизованого виявлення атак. Такий модуль має реалізовувати повний цикл обробки даних мережевого трафіку: від завантаження

та інтеграції вхідних наборів даних до формування ознакового простору, побудови вибірок для різних постановок задачі, навчання моделей машинного навчання, оцінювання результатів їх функціонування та підготовки аналітичних матеріалів для подальшого експериментального дослідження. Програмна реалізація в цій роботі розглядається не як допоміжний інструмент виконання окремих обчислень, а як практична форма представлення розробленої інформаційної технології.

Особливістю цієї реалізації є її орієнтація на інтелектуальний моніторинг трафіку комп'ютерної мережі, тобто на автоматизований аналіз характеристик мережевих потоків із застосуванням методів машинного навчання. У межах запропонованої інформаційної технології рішення щодо наявності ознак атаки або щодо типу виявленої загрози формуються на основі закономірностей, виявлених у даних мережевого трафіку. У зв'язку з цим програмний модуль має забезпечувати не лише технічну обробку набору даних, а й реалізацію повної послідовності процедур інтелектуального аналізу, починаючи від підготовки вхідних характеристик і завершуючи формуванням класифікаційних рішень.

З урахуванням поставленої мети до основних завдань програмної реалізації інформаційної технології належать:

– завантаження та інтеграція даних мережевого трафіку з окремих джерел у єдину структуру;

– виконання первинного аналізу та контролю якості вхідних даних;

– реалізація процедур попередньої обробки, очищення та підготовки характеристик мережевих потоків;

– формування ознакового простору, придатного для використання моделями машинного навчання;

– побудова вибірки для бінарного виявлення атак у трафіку комп'ютерної мережі;

– побудова та балансування вибірки для багатокласового розпізнавання типів атак;

– реалізація моделей машинного навчання для двох режимів аналізу мережевого трафіку;

– реалізація процедур оцінювання та візуалізації результатів функціонування моделей.

Важливим аспектом програмної реалізації є те, що вона повинна відображати фактичну логіку функціонування розробленої інформаційної технології. Це означає, що структура програмного модуля має бути побудована як послідовність взаємопов’язаних етапів, кожен з яких виконує чітко визначену функцію в загальному процесі інтелектуального моніторингу трафіку комп’ютерної мережі. До таких етапів належать завантаження та інтеграція даних, первинний аналіз, попередня обробка, формування ознакового простору, підготовка бінарної та багатокласової вибірок, балансування даних, навчання моделей машинного навчання, а також оцінювання та візуалізація результатів. Така організація дозволяє забезпечити цілісність програмного конвеєра та узгодженість між усіма його складовими.

Суттєвою вимогою до програмної реалізації є її модульність. У межах цієї роботи модульність реалізується не як сукупність ізольованих програмних засобів, а як логічна організація єдиного програмного модуля у вигляді окремих функціональних блоків: завантаження та інтеграції даних, первинного аналізу, попередньої обробки, формування ознакового простору, побудови бінарної та багатокласової вибірок, балансування, навчання моделей, оцінювання та візуалізації результатів. Така логічна організація програмного модуля забезпечує поетапний контроль результатів, узгодженість обробки даних, зручність модифікації окремих складових і можливість подальшого розширення програмного модуля без порушення цілісності всієї інформаційної технології.

Окремого значення набуває підтримка у межах одного програмного модуля двох режимів функціонування інформаційної технології. Перший режим пов’язаний із бінарним виявленням атак, коли результатом аналізу є віднесення мережевого потоку до нормального або атакувального трафіку. Другий режим орієнтований на багатокласове розпізнавання типів атак, що дозволяє деталізувати результати аналізу та розширити прикладні можливості використання технології в системах виявлення атак. Реалізація обох режимів у межах єдиного програмного

середовища дозволяє забезпечити цілісність дослідження та створює умови для подальшого порівняння різних моделей машинного навчання за однакових умов підготовки даних.

Ще однією важливою вимогою до програмної реалізації є її відтворюваність. У дисертаційному дослідженні це означає, що всі етапи обробки мережевого трафіку, формування вибірок, навчання моделей та оцінювання їх ефективності повинні бути організовані у вигляді впорядкованої послідовності процедур, яку можна повторити за однакових умов. Відтворюваність є необхідною умовою наукової обґрунтованості результатів, оскільки саме вона дозволяє перевірити коректність функціонування інформаційної технології, порівняти різні конфігурації моделей і надалі використовувати реалізований програмний модуль як основу для експериментального дослідження.

У цьому підрозділі визначено призначення, основні завдання та ключові вимоги до програмної реалізації інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак. Реалізація має забезпечувати цілісний, модульно організований і відтворюваний цикл обробки даних, побудови моделей машинного навчання та оцінювання результатів у двох режимах функціонування – бінарному і багатокласовому. Далі розглядаються засоби програмної реалізації, структура програмного модуля та особливості виконання окремих етапів розробленої інформаційної технології.

3.5 Вибір засобів програмної реалізації та організація середовища розроблення

Реалізація інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак потребує такого програмного середовища, яке забезпечує повний цикл обробки даних мережевого трафіку, побудови моделей машинного навчання, оцінювання результатів їх функціонування та подання одержаних результатів у наочній формі. З огляду на це як основний засіб програмної реалізації обрано мову програмування Python, яка є

однією з найбільш поширених у задачах аналізу даних, машинного навчання та наукових обчислень. Її використання зумовлене поєднанням кількох важливих переваг: наявністю розвиненої екосистеми бібліотек, зручністю роботи з великими масивами табличних даних, підтримкою сучасних алгоритмів інтелектуального аналізу та широкими можливостями для графічного представлення результатів [62-65].

Доцільність вибору Python у межах роботи визначається також тим, що розроблювана інформаційна технологія має інтегрувати кілька різних за призначенням етапів: завантаження та інтеграцію даних мережевого трафіку, первинний аналіз і попередню обробку, формування ознакового простору, побудову вибірок для бінарного та багатокласового аналізу, балансування даних, навчання моделей машинного навчання, а також оцінювання й візуалізацію результатів. Реалізація всіх зазначених процедур у межах одного програмного середовища забезпечує узгодженість виконання етапів обробки, знижує ймовірність технічних помилок під час передавання даних між окремими програмними компонентами та сприяє відтворюваності експериментального процесу. Тому використання Python є обґрунтованим не лише з інструментальної, а й з методичної точки зору.

Для реалізації операцій із табличними даними мережевого трафіку в роботі використано бібліотеки pandas та NumPy. Їх застосування є доцільним у зв’язку з тим, що трафік комп’ютерної мережі в межах дослідження подається у вигляді структурованих наборів даних, які містять велику кількість числових характеристик мережевих потоків. Бібліотека pandas забезпечує засоби зчитування, об’єднання, фільтрації, трансформації та аналізу табличних структур, що дає змогу реалізувати завантаження окремих піднаборів трафіку, їх інтеграцію в єдиний масив, а також подальше очищення й підготовку до машинного навчання. Бібліотека NumPy, своєю чергою, забезпечує ефективну роботу з числовими масивами та використовується для обчислювальних процедур у межах програмного модуля [66, 67, 70-74].

Ключову роль у програмній реалізації методів інтелектуального моніторингу трафіку комп'ютерної мережі відіграє бібліотека scikit-learn. Вона забезпечує основний інструментарій для побудови моделей машинного навчання, а також для виконання супровідних процедур, необхідних для коректного функціонування інформаційної технології. У межах розробленого програмного модуля засоби scikit-learn використовуються для стандартизації ознак, зниження розмірності ознакового простору, поділу набору даних на навчальні та тестові підмножини, навчання класифікаторів, перехресної перевірки та розрахунку показників якості. Використання єдиної бібліотеки для реалізації перелічених етапів дозволяє зберігати методичну цілісність програмного конвеєра та забезпечує порівнюваність результатів функціонування різних моделей виявлення атак.

Оскільки однією з характерних особливостей даних мережевого трафіку для систем виявлення атак є дисбаланс між окремими класами, важливою складовою середовища розроблення є бібліотека imbalanced-learn (пакет imblearn). Її застосування у межах роботи пов'язане з необхідністю реалізації процедур балансування даних, зокрема синтетичного збільшення кількості прикладів класів меншості. Інтеграція засобів imbalanced-learn у програмний модуль дозволяє підготувати більш репрезентативні навчальні вибірки для багатокласового розпізнавання атак і тим самим підвищити стійкість моделей машинного навчання до нерівномірності розподілу спостережень між класами [82, 83].

Для візуального аналізу даних мережевого трафіку та подання результатів функціонування інформаційної технології використано бібліотеки Matplotlib і Seaborn. Їх застосування забезпечує побудову графіків, діаграм і теплових карт, що використовуються для наочного подання результатів проміжної діагностики даних і підсумкового оцінювання моделей. У роботи зазначені засоби є важливими не лише як інструменти графічного оформлення, а як складова аналітичного супроводу інформаційної технології, оскільки дозволяють виявляти особливості структури даних мережевого трафіку, аналізувати якість класифікації та порівнювати результати роботи різних моделей машинного навчання. Додатково використано бібліотеку missingno, призначену для візуалізації пропущених значень

у табличних структурах, що є доцільним на етапі первинного аналізу та попередньої обробки даних. Її застосування дає змогу швидко оцінити характер і розподіл пропусків у наборі даних та приймати обґрунтовані рішення щодо подальшого очищення й підготовки ознакового простору [75-81].

Організація середовища розроблення у межах роботи побудована за принципом послідовного виконання взаємопов’язаних процедур у межах єдиного програмного сценарію. Це означає, що всі етапи функціонування інформаційної технології – від завантаження вхідних даних мережевого трафіку до формування підсумкових оцінок якості виявлення атак – реалізуються в узгодженому програмному просторі без потреби переходу між різними платформами або несумісними програмними засобами. Така організація є важливою з точки зору дослідницької коректності, оскільки забезпечує цілісність даних, стабільність обчислювального процесу та зручність повторного відтворення експериментів.

Суттєвою вимогою до середовища розроблення в межах дисертаційного дослідження є логічно-модульна організація та можливість поетапного контролю результатів. Обране програмне середовище дає змогу реалізувати цю вимогу шляхом поєднання окремих функціональних блоків у межах єдиного програмного модуля: завантаження та інтеграції даних, первинного аналізу, попередньої обробки, формування ознакового простору, побудови бінарної та багатокласової вибірок, балансування, навчання моделей, оцінювання та візуалізації результатів. Така структура програмної реалізації забезпечує узгоджене функціонування всієї інформаційної технології та створює умови для подальшого вдосконалення окремих її складових без порушення загальної логіки роботи.

Вибір засобів програмної реалізації та організація середовища розроблення в роботі підпорядковані вимогам повної, узгодженої та відтворюваної реалізації інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак. Використання Python і спеціалізованих бібліотек для обробки даних, машинного навчання, балансування та візуалізації створює практичне підґрунтя для розроблення програмного модуля, який реалізує повний цикл інтелектуального аналізу мережевого трафіку. Далі розглядаються

структура програмного модуля та послідовність обробки трафіку комп’ютерної мережі в межах розробленої інформаційної технології [62-83, 87-91].

3.6 Структура програмного модуля та послідовність обробки трафіку комп’ютерної мережі

Програмна реалізація інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак побудована як єдиний програмний модуль із логічно виокремленими функціональними блоками, кожен з яких відповідає окремому етапу обробки та аналізу даних мережевого трафіку.

З погляду загальної організації програмний модуль має логічно-модульну структуру, яка охоплює блоки завантаження та інтеграції даних, первинного аналізу, попередньої обробки, формування ознакового простору, побудови вибірок для бінарного та багатокласового аналізу, балансування даних, навчання моделей машинного навчання, а також оцінювання та візуалізації результатів. Незважаючи на реалізацію у вигляді єдиного програмного сценарію, така структура забезпечує чітке розмежування функцій між окремими етапами та дозволяє розглядати програмний модуль як впорядковану систему взаємопов’язаних процедур.

Початковим етапом функціонування програмного модуля є завантаження окремих файлів набору CIC-IDS2017 та їх інтеграція в єдиний масив даних мережевого трафіку. На цьому етапі здійснюється зчитування окремих CSV-файлів, які містять характеристики мережевих потоків за різні часові проміжки та сценарії мережевої активності, після чого виконується їх об’єднання в спільну табличну структуру. Далі реалізується первинний аналіз одержаного масиву: перевірка розмірності, структури, типів ознак, наявності пропущених і дубльованих значень, а також загальний статистичний огляд даних. Цей етап формує вхідну інформаційну основу для всіх подальших процедур інтелектуального аналізу.

Наступний блок програмного модуля пов’язаний із попередньою обробкою даних і формуванням ознакового простору. У його межах виконуються очищення та уніфікація структури даних, видалення дублікатів, обробка пропущених і нескінченних значень, перетворення вихідних міток у форму, придатну для подальшого аналізу, дослідження кореляцій між ознаками, стандартизація числових характеристик і зниження розмірності ознакового простору. Результатом цього етапу є підготовлене подання мережевого трафіку, придатне для побудови вибірок і навчання моделей машинного навчання.

Після формування ознакового простору в програмному модулі реалізовано дві взаємопов’язані гілки подальшої обробки даних. Перша гілка орієнтована на побудову вибірки для бінарного виявлення атак, у межах якої мережевий трафік поділяється на нормальний та атакувальний. На цьому етапі формується двокласова структура даних, що використовується для подальшого навчання моделей, призначених для первинного виявлення факту атакувальної активності. Друга гілка орієнтована на багатокласове розпізнавання типів атак. У межах цієї гілки виконується формування робочої вибірки з урахуванням поділу атак за типами, а також балансування класів для підготовки даних до навчання моделей багатокласової класифікації. Така організація забезпечує підтримку двох режимів функціонування розробленої інформаційної технології в межах єдиного програмного модуля.

Окремий блок програмного модуля становить реалізація моделей машинного навчання. Для бінарного режиму використовуються моделі логістичної регресії та методу опорних векторів, а для багатокласового режиму – моделі випадкового лісу, дерева рішень і методу k найближчих сусідів. Обраний набір алгоритмів дає змогу дослідити різні алгоритмічні принципи класифікації мережевого трафіку та порівняти ефективність моделей за однакових умов підготовки даних. Після етапу навчання реалізується блок оцінювання, у якому розраховуються показники якості класифікації, виконується перехресна перевірка, будуються матриці помилок, ROC-криві, Precision–Recall-криві та інші засоби аналітичного подання результатів.

Таким чином, послідовність обробки трафіку комп’ютерної мережі в межах розробленого програмного модуля має вигляд впорядкованого програмного конвеєра: завантаження та інтеграція даних → первинний аналіз → попередня обробка → формування ознакового простору → побудова бінарної або багатокласової вибірки → балансування даних → навчання моделей машинного навчання → оцінювання та візуалізація результатів. Така структура забезпечує цілісність реалізації інформаційної технології, відтворюваність експериментів і можливість подальшого вдосконалення окремих функціональних блоків без порушення загальної логіки роботи програмного модуля.

3.7 Програмна реалізація завантаження, інтеграції та первинного аналізу даних мережевого трафіку

Початковим етапом функціонування програмного модуля інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак є завантаження, інтеграція та первинний аналіз вхідних даних мережевого трафіку. На цьому етапі формується базова інформаційна структура, яка надалі використовується для попередньої обробки, побудови ознакового простору, формування вибірок і навчання моделей машинного навчання. Тому програмна реалізація цього етапу має забезпечувати не лише коректне зчитування вхідних даних, а й їх структурну узгодженість, цілісність та придатність до подальшої аналітичної обробки.

У розробленому програмному модулі завантаження набору даних CIC-IDS2017 реалізовано шляхом послідовного зчитування восьми окремих CSV-файлів за допомогою функції pd.read_csv() (табл. 3.1). Кожен файл перетворюється на окремий датафрейм data1-data8, що відповідає певному фрагменту мережевого трафіку. Після цього всі сформовані датафрейми об’єднуються у список data_list, який використовується як проміжна програмна структура для подальшої інтеграції даних. Використання такої проміжної структури даних дає змогу зберігати контроль над кожним завантаженим фрагментом трафіку та забезпечує

впорядковану реалізацію початкового етапу обробки даних у межах єдиного програмного сценарію [71].

На етапі завантаження для кожного файла окремо виводиться його розмірність у форматі «кількість рядків – кількість стовпців», що реалізовано через ітерацію по об’єктах списку data_list. Це дає змогу ще до інтеграції перевірити повноту завантаження набору даних, виявити можливі збої зчитування та проконтролювати узгодженість структури окремих частин трафіку.

Після завершення завантаження у програмному модулі реалізовано інтеграцію всіх частин набору даних у єдину робочу структуру. Для цього використовується операція pd.concat(data_list), результатом якої є інтегрований датафрейм data [72].

Таблиця 3.1 – Вхідні файли набору CIC-IDS2017, що використовуються на етапі завантаження даних

№	Ім’я програмного об’єкта	Назва файла	Характеристика фрагмента трафіку
1	data1	Monday-WorkingHours.pcap_ISCX.csv	Містить лише нормальний (benign) трафік
2	data2	Tuesday-WorkingHours.pcap_ISCX.csv	Містить атаки типу Brute Force FTP та SSH
3	data3	Wednesday-workingHours.pcap_ISCX.csv	Містить атаки типу DoS (Denial of Service)
4	data4	Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Містить веб-атаки: Web Attack – Brute Force, Web Attack – XSS, Web Attack – Sql Injection
5	data5	Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	Містить атаки типу Infiltration
6	data6	Friday-WorkingHours-Morning.pcap_ISCX.csv	Містить трафік типу Botnet
7	data7	Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	Містить атаки типу PortScan
8	data8	Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	Містить атаки типу DDoS

Цей об’єкт у подальшому виступає центральною програмною структурою, на якій виконуються всі операції попередньої обробки, формування ознак, побудови вибірок і навчання моделей. Після інтеграції в коді додатково обчислюються загальна кількість рядків, стовпців і комірок, що дозволяє оцінити фактичний

масштаб сформованого набору даних і проконтролювати коректність операції об’єднання. Окремо реалізовано звільнення пам’яті шляхом видалення проміжних кадрів даних із data_list, що є доцільним з огляду на значний обсяг інтегрованого набору мережевого трафіку.

Після інтеграції виконується уніфікація назв стовпців. У коді для цього формується словник col_names = {col: col.strip() for col in data.columns}, після чого застосовується перейменування через data.rename(columns=col_names, inplace=True). Така програмна операція усуває початкові та кінцеві пробіли в іменах полів і забезпечує однозначний доступ до характеристик мережевих потоків на наступних етапах обробки. Незважаючи на технічний характер цієї процедури, вона є важливою складовою розроблення програмного модуля, оскільки дозволяє уникнути помилок адресації ознак, що можуть виникати під час формування ознакового простору та реалізації алгоритмів машинного навчання.

Наступним кроком є первинний структурний аналіз інтегрованого датафрейму data. Для цього в програмному модулі використовуються виклики data.info() та data.describe().transpose(). Перший із них надає відомості про типи змінних, кількість ненульових значень і загальну структуру датафрейму, а другий – формує описові статистичні характеристики числових ознак. Додатково виводяться результати data.isnull().sum(), що дозволяє виявити наявність пропущених значень у вхідних даних. У сукупності ці операції забезпечують початкову діагностику набору даних і створюють аналітичне підґрунтя для подальшої попередньої обробки [70, 74].

На етапі завантаження, інтеграції та первинного аналізу в програмному модулі формується набір програмних структур даних, що забезпечують зберігання вхідних піднаборів мережевого трафіку, їх інтеграцію, структурний контроль та первинну діагностику якості даних (табл. 3.2).

Окремою процедурою первинного аналізу є виявлення та вилучення дубльованих записів. У коді для цього формується об’єкт dups = data[data.duplicated()], після чого виводиться кількість дубльованих спостережень, а самі повторювані записи видаляються операцією

data.drop_duplicates(inplace=True). Після вилучення дублікатів повторно контролюється розмірність датафрейму через data.shape. Така процедура має важливе значення для коректності подальшого аналізу, оскільки дублікати можуть призводити до викривлення статистичних характеристик набору даних, посилення окремих шаблонів мережевого трафіку та завищення оцінок якості моделей машинного навчання [73].

Таблиця 3.2 – Основні програмні структури даних етапу завантаження, інтеграції та первинного аналізу

Назва об'єкта	**Тип структури**	**Фрагмент вихідного коду**	**Призначення у програмному модулі**
data1–data8	DataFrame	data1 = pd.read_csv(...) … data8 = pd.read_csv(...)	Зберігання окремих піднаборів мережевого трафіку, зчитаних із CSV-файлів CIC-IDS2017
data_list	list	data_list = [data1, data2, data3, data4, data5, data6, data7, data8]	Формування проміжного контейнера для інтеграції окремих датафреймів
data	DataFrame	data = pd.concat(data_list)	Основний інтегрований датафрейм, що використовується на всіх наступних етапах обробки
rows, cols	int	rows, cols = data.shape	Зберігання розмірності окремих та інтегрованого наборів даних
col_names	dict	col_names = {col: col.strip() for col in data.columns}	Уніфікація назв стовпців шляхом видалення початкових і кінцевих пробілів
dups	DataFrame	dups = data[data.duplicated()]	Тимчасове зберігання дубльованих записів для подальшого вилучення
missing_val	Series	missing_val = data.isna().sum()	Початкове визначення кількості пропущених значень у стовпцях інтегрованого датафрейму
numeric_cols	Index	numeric_cols = data.select_dtypes(include = np.number).columns	Формування переліку числових ознак для перевірки на наявність нескінченних значень
inf_count	Series	inf_count = np.isinf(data[numeric_cols]).sum()	Підрахунок кількості нескінченних значень у числових стовпцях
missing	Series	missing = data.isna().sum()	Повторне визначення кількості пропущених значень після перетворення ±inf у NaN

Продовження таблиці 3.2

Назва об’єкта	Тип структури	Фрагмент вихідного коду	Призначення у програмному модулі
mis_per	Series	mis_per = (missing / len(data)) * 100	Обчислення частки пропущених значень відносно загального обсягу даних
mis_table	DataFrame	mis_table = pd.concat([missing, mis_per.round(2)], axis = 1)	Формування узагальненої таблиці кількості та відсотка пропущених значень
missing_vals	list	missing_vals = [col for col in data.columns if data[col].isna().any()]	Формування списку ознак із пропущеними значеннями для подальшої візуалізації
med_flow_bytes	float	med_flow_bytes = data['Flow Bytes/s'].median()	Збереження медіанного значення ознаки Flow Bytes/s для локального заповнення пропусків
med_flow_packets	float	med_flow_packets = data['Flow Packets/s'].median()	Збереження медіанного значення ознаки Flow Packets/s для локального заповнення пропусків

Важливим блоком програмної реалізації є аналіз відсутніх і технічно некоректних числових значень. Для цього в коді формується множина числових ознак numeric_cols = data.select_dtypes(include=np.number).columns, після чого виконується перевірка на наявність нескінченних значень за допомогою np.isinf(data[numeric_cols]).sum(). Далі значення np.inf та -np.inf у датафреймі замінюються на NaN через операцію data.replace([np.inf, -np.inf], np.nan, inplace=True). Виконання цієї процедури забезпечує переведення технічно некоректних числових значень у єдиний формат пропущених даних, придатний для подальшого аналізу та корекції. Після цього повторно оцінюється кількість пропущених значень і формується таблиця mis_table, яка містить абсолютну кількість та відсоткову частку відсутніх значень для проблемних ознак [68, 69, 74].

Для візуальної діагностики структури пропущених значень у програмному модулі використано бібліотеку missingno. У коді формується список missing_vals = [col for col in data.columns if data[col].isna().any()], після чого за допомогою msno.bar(data[missing_vals], ...) будується діаграма, що відображає розподіл ненульових значень у проблемних стовпцях. Використання такого графічного

представлення дозволяє швидко ідентифікувати ознаки з найбільш вираженою неповнотою даних і прийняти обґрунтовані рішення щодо способів їх подальшої обробки. У контексті розроблення програмного модуля це означає поєднання числової й графічної діагностики як взаємодоповнювальних інструментів первинного контролю якості даних [75].

У межах цього ж етапу виконано додатковий аналіз окремих інтенсивнісних ознак Flow Bytes/s та Flow Packets/s, які виявилися найбільш проблемними з погляду наявності некоректних значень. Для цих ознак будуються boxplot-діаграми та гістограми, обчислюються медіанні значення med_flow_bytes і med_flow_packets, після чого відсутні значення заповнюються за допомогою fillna(). Така локальна програмна процедура виконує роль первинної корекції ключових числових характеристик мережевого трафіку, які надалі використовуються при формуванні ознакового простору. Після виконання зазначених операцій у коді додатково здійснюється скидання індексу датафрейму через data = data.reset_index(drop=True), що забезпечує коректну послідовну нумерацію записів після видалення дублікатів і внесення структурних змін до набору даних [70, 74, 76, 77, 79, 81].

Послідовність програмних процедур, реалізованих на етапі завантаження, інтеграції та первинного аналізу даних мережевого трафіку, наведено в табл. 3.3.

Таблиця 3.3 – Основні програмні процедури етапу завантаження, інтеграції та первинного аналізу даних

№	Програмна процедура	Зміст реалізації	Результат виконання
1	Завантаження вхідних піднаборів	Послідовне зчитування восьми CSV-файлів набору CIC-IDS2017 засобами pd.read_csv()	Формування початкового набору табличних структур, що містять окремі фрагменти мережевого трафіку
2	Контроль коректності завантаження	Визначення розмірності кожного зчитаного піднабору	Перевірка повноти й коректності завантаження всіх частин набору даних
3	Інтеграція піднаборів у єдину структуру	Вертикальне об’єднання всіх завантажених частин засобами pd.concat()	Формування інтегрованого датафрейму, що використовується як основна робоча структура
4	Уніфікація назв ознак	Очищення назв стовпців від початкових і кінцевих пробілів та їх нормалізація	Забезпечення однозначної адресації ознак на наступних етапах обробки

Продовження таблиці 3.3

№	Програмна процедура	Зміст реалізації	Результат виконання
5	Первинний структурний аналіз	Отримання відомостей про типи даних, кількість ненульових значень і базові статистичні характеристики через info() та describe()	Формування початкової діагностики структури інтегрованого набору даних
6	Аналіз пропущених значень	Визначення кількості пропусків у стовпцях та обчислення їх частки у структурі набору	Виявлення проблемних ознак, що потребують подальшої обробки
7	Виявлення і вилучення дублікатів	Пошук повторюваних записів і їх видалення	Усунення надлишкових спостережень, що можуть спотворювати статистичні характеристики вибірки
8	Аналіз некоректних числових значень	Виявлення нескінченних значень у числових ознаках та їх перетворення у формат NaN	Приведення числових характеристик до узгодженого стану, придатного для подальшої обробки
9	Візуальна діагностика структури пропусків	Побудова діаграми розподілу пропущених значень засобами missingno	Наочне подання проблемних ознак і оцінювання характеру неповноти даних
10	Локальна корекція ключових інтенсивнісних ознак	Аналіз розподілів Flow Bytes/s і Flow Packets/s, обчислення медіан та заповнення пропусків	Усунення пропусків у найбільш проблемних числових характеристиках мережевого трафіку
11	Відновлення узгодженої індексації	Скидання індексу після структурних змін датафрейму	Формування коректної послідовної нумерації записів для подальших етапів обробки

Розглянута послідовність програмних дій формує завершений початковий блок функціонування розробленого програмного модуля. Його результатом є інтегрований, структурно узгоджений і первинно проаналізований датафрейм мережевого трафіку, придатний для переходу до наступного етапу – попередньої обробки даних і формування ознакового простору.

3.8 Програмна реалізація попередньої обробки даних і формування ознакового простору

Після завершення етапу завантаження, інтеграції та первинного аналізу даних мережевого трафіку в програмному модулі реалізується етап попередньої

обробки та формування ознакового простору, придатного для подальшого використання в моделях машинного навчання. На вхід цього етапу надходить інтегрований датафрейм data, у якому вже виконано уніфікацію назв стовпців, видалення дубльованих записів, перетворення нескінченних значень у формат NaN, локальну корекцію найбільш проблемних числових ознак та відновлення послідовної індексації. На цьому етапі початковий масив характеристик мережевих потоків перетворюється на впорядковане, компактне та аналітично придатне подання, яке може бути безпосередньо використане для побудови бінарних і багатокласових моделей виявлення атак.

Першою змістовною процедурою цього етапу є формування цільового подання класів атак. У програмному коді для цього створюється словник attack_map, який використовується для приведення вихідних значень стовпця Label до узагальнених категорій атак. На його основі формується новий стовпець Attack Type, що надалі використовується як основна цільова змінна для задач класифікації. Далі вихідний стовпець Label вилучається з робочого датафрейму, а для допоміжного аналітичного аналізу створюється числове кодування класів через об’єкт LabelEncoder і стовпець Attack Number. Така програмна реалізація забезпечує перехід від первинних позначень атак до уніфікованого подання цільових міток, що є необхідною умовою для подальшого формування ознакового простору та дослідження зв’язків між характеристиками мережевого трафіку і класами атак [89].

Наступною процедурою є відбір інформативних характеристик і вилучення неваріантних ознак. У програмному модулі для цього використовується підрахунок кількості унікальних значень у кожному стовпці за допомогою data.nunique(). На основі отриманих результатів формуються множини one_variable та not_one_variable, після чого з робочого набору даних вилучаються ознаки, що мають лише одне унікальне значення. Результати такого відбору фіксуються в об’єкті dropped_cols, а сам датафрейм data оновлюється таким чином, щоб містити лише характеристики, які можуть бути корисними для розмежування класів мережевого трафіку. З позиції розроблення програмного модуля ця процедура є

важливою, оскільки дозволяє зменшити надлишковість вхідного опису, спростити структуру даних та підвищити ефективність наступних процедур стандартизації й зниження розмірності.

У межах попередньої обробки в програмному модулі реалізовано також допоміжний аналітичний блок, що використовується для уточнення структури даних перед побудовою ознакового простору. До нього належать побудова візуалізацій варіативності ознак, аналіз кількості унікальних значень, дослідження розподілів вибраних стовпців, аналіз викидів у розрізі типів атак на основі міжквартильного розмаху, а також побудова графіків розподілу класів. Ці процедури не змінюють безпосередньо структуру датафрейму, однак забезпечують аналітичний супровід етапу попередньої обробки та дозволяють обґрунтувати доцільність подальших перетворень ознакового простору [76-81, 117].

Після завершення відбору ознак у програмному коді формується матриця вхідних характеристик і вектор цільових міток. Для цього використано конструкції features = data.drop('Attack Type', axis=1) та attacks = data['Attack Type']. У результаті всі числові характеристики мережевих потоків переходять до матриці features, тоді як інформація про належність кожного запису до відповідного класу трафіку зберігається окремо у векторі attacks. Така форма подання є базовою для подальшого застосування процедур стандартизації, зниження розмірності й побудови вибірок у задачах виявлення атак.

Наступним кроком у програмній реалізації є стандартизація ознак мережевого трафіку. Для цього використовується клас StandardScaler із бібліотеки scikit-learn. Після створення об’єкта scaler виконується операція scaled_features = scaler.fit_transform(features), у результаті якої формується стандартизована матриця ознак. Така операція є необхідною, оскільки окремі характеристики мережевих потоків мають різні масштаби вимірювання, а їх безпосереднє використання в моделях машинного навчання може призводити до домінування окремих змінних і викривлення результатів класифікації. У контексті програмного модуля стандартизація є обов’язковою підготовчою процедурою перед застосуванням

алгоритмів, чутливих до масштабу вхідних параметрів, зокрема SVM, k-NN та PCA [88].

Після стандартизації у програмному модулі реалізується зниження розмірності ознакового простору за допомогою методу головних компонент. У коді для цього використовується клас IncrementalPCA, що є доцільним для роботи з великими масивами даних мережевого трафіку. Кількість компонент задається як size = len(features.columns) // 2, після чого створюється об’єкт ipca = IncrementalPCA(n_components=size, batch_size=500). Навчання моделі головних компонент реалізовано поетапно: стандартизована матриця scaled_features розбивається на пакети засобами np.array_split, і для кожного пакета викликається ipca.partial_fit(batch). Після завершення навчання виконується перетворення всієї стандартизованої вибірки через transformed_features = ipca.transform(scaled_features). Така реалізація дає змогу побудувати компактніше подання ознак без втрати основних закономірностей, важливих для виявлення атак [66, 67, 88].

Результатом застосування IncrementalPCA є формування нового датафрейму new_data, який містить головні компоненти PC1, PC2, …, PCn та стовпець Attack Type. Для цього в коді використовується конструкція new_data = pd.DataFrame(transformed_features, columns=[f'PC{i+1}' for i in range(size)]), після чого до нього додається цільова змінна через new_data['Attack Type'] = attacks.values. Саме об’єкт new_data у програмному модулі виступає базою для подальшого формування вибірок у задачах бінарного виявлення атак і багатокласового розпізнавання їх типів. У такий спосіб початковий високорозмірний опис мережевого трафіку перетворюється на компактний ознаковий простір, придатний для інтелектуального моніторингу та побудови моделей машинного навчання.

Основні програмні структури даних, що формуються та використовуються на етапі попередньої обробки й формування ознакового простору, наведено в табл. 3.4.

Таблиця 3.4 – Основні програмні структури даних етапу попередньої обробки та формування ознакового простору

Назва об’єкта	Тип структури	Призначення у програмному модулі
attack_map	dict	Відображення початкових значень Label в узагальнені категорії атак
le	LabelEncoder	Числове кодування значень Attack Type для допоміжного аналітичного аналізу
one_variable	list / Index	Зберігання переліку ознак, що мають лише одне унікальне значення
not_one_variable	list / Index	Зберігання переліку ознак, що мають більше одного унікального значення
dropped_cols	list	Фіксація переліку вилучених невариантних ознак
features	DataFrame	Матриця вхідних ознак для подальшої стандартизації та зниження розмірності
attacks	Series	Вектор цільових міток, що містить значення Attack Type
scaler	StandardScaler	Об’єкт стандартизації числових ознак
scaled_features	ndarray	Стандартизована матриця вхідних ознак
size	int	Кількість головних компонент, що використовуються в IncrementalPCA
ipca	IncrementalPCA	Об’єкт моделі головних компонент для поетапного навчання на великих масивах даних
transformed_features	ndarray	Матриця ознак після зниження розмірності методом головних компонент
new_data	DataFrame	Новий датафрейм, що містить головні компоненти та стовпець Attack Type і використовується як основа для подальшого формування вибірок

Послідовність програмних процедур, реалізованих на етапі попередньої обробки даних і формування ознакового простору, наведено в табл. 3.5.

Таблиця 3.5 – Основні програмні процедури етапу попередньої обробки та формування ознакового простору

№	Програмна процедура	Зміст реалізації	Результат виконання
1	Формування узагальнених класів атак	Створення словника attack_map та побудова стовпця Attack Type на основі Label	Перехід від первинних міток атак до уніфікованого подання цільових класів
2	Числове кодування класів	Застосування LabelEncoder для формування стовпця Attack Number	Отримання числового подання класів для допоміжного аналітичного аналізу
3	Вилучення невариантних ознак	Обчислення data.nunique() та виключення стовпців з одним унікальним значенням	Зменшення надлишковості ознакового опису та спрощення структури даних

Продовження таблиці 3.5

№	Програмна процедура	Зміст реалізації	Результат виконання
4	Допоміжний аналітичний аналіз ознак	Побудова візуалізацій варіативності, розподілів, кількості унікальних значень і викидів	Оцінювання придатності ознак до подальшого моделювання
5	Формування матриці ознак і вектора міток	Побудова features = data.drop('Attack Type', axis=1) та attacks = data['Attack Type']	Розділення описових характеристик мережевого трафіку і цільових міток
6	Стандартизація ознак	Застосування StandardScaler та виконання fit_transform(features)	Приведення ознак до зіставного масштабу
7	Задання параметрів зниження розмірності	Обчислення кількості компонент size та створення об’єкта IncrementalPCA	Підготовка моделі головних компонент до навчання
8	Поетапне навчання моделі головних компонент	Розбиття scaled_features на пакети та виклик ipca.partial_fit(batch)	Навчання моделі IncrementalPCA на великому масиві стандартизованих даних
9	Перетворення ознак у простір головних компонент	Виконання transformed_features = ipca.transform(scaled_features)	Формування компактного ознакового подання мережевого трафіку
10	Побудова нового ознакового датафрейму	Створення new_data зі стовпцями PC1, PC2, …, PCn і додавання Attack Type	Отримання підготовленого ознакового простору для подальшого формування бінарної та багатокласової вибірок

Підготовлений у результаті цього етапу датафрейм new_data поєднує компактне подання вхідних характеристик у просторі головних компонент із цільовими мітками класів атак і створює безпосередню основу для наступного етапу функціонування інформаційної технології – формування вибірок у задачах бінарного виявлення атак і багатокласового розпізнавання їх типів.

3.9 Програмна реалізація формування вибірки для бінарного виявлення атак

Після формування ознакового простору трафіку комп’ютерної мережі наступним етапом програмної реалізації є побудова вибірки для бінарного виявлення атак. У межах розробленої інформаційної технології цей режим інтелектуального моніторингу орієнтований на розв’язання базової задачі класифікації, за якої кожний мережевий потік має бути віднесений або до

нормального трафіку, або до атакувального. Така постановка є принципово важливою для систем виявлення атак, оскільки забезпечує первинний рівень контролю стану комп'ютерної мережі та дозволяє оперативно фіксувати наявність небезпечної активності без необхідності негайної деталізації її типу.

У програмному модулі формування вибірки для бінарного виявлення атак виконується на основі датафрейму new_data, який містить головні компоненти PC1, PC2, …, PCn та стовпець Attack Type. Цей датафрейм є результатом попереднього етапу стандартизації та зниження розмірності, тому надалі він використовується як базове подання мережевого трафіку для побудови різних постановок задачі. Для реалізації бінарного режиму в коді окремо виділяються записи нормального трафіку через конструкцію normal_traffic = new_data.loc[new_data['Attack Type'] == 'BENIGN'] та записи атакувального трафіку через intrusions = new_data.loc[new_data['Attack Type'] != 'BENIGN']. У такий спосіб у програмному модулі формується чітке розмежування двох базових категорій мережевої активності, що становить основу бінарної постановки задачі.

Основні програмні структури даних, що формуються та використовуються на етапі побудови вибірки для бінарного виявлення атак, наведено в табл. 3.6.

Таблиця 3.6 – Основні програмні структури даних етапу формування вибірки для бінарного виявлення атак

Назва об'єкта	**Тип структури**	**Призначення у програмному модулі**
new_data	DataFrame	Базовий датафрейм, що містить головні компоненти ознакового простору та стовпець Attack Type
normal_traffic	DataFrame	Підмножина записів new_data, що відповідає нормальному трафіку (BENIGN)
intrusions	DataFrame	Підмножина записів new_data, що відповідає атакувальному трафіку (Attack Type != 'BENIGN')
ids_data	DataFrame	Об'єднана структура даних, що містить записи атакувального трафіку та вибіркову підмножину нормального трафіку
bc_data	DataFrame	Робочий датафрейм для бінарної класифікації після перетворення цільової змінної у двокласовий формат
X_bc	DataFrame	Матриця вхідних ознак для задачі бінарного виявлення атак
y_bc	Series	Вектор цільових міток для бінарної класифікації, де 0 – нормальний трафік, 1 – атакувальний

Послідовність основних програмних процедур, реалізованих на етапі формування вибірки для бінарного виявлення атак, наведено в табл. 3.7.

Таблиця 3.7 – Основні програмні процедури етапу формування вибірки для бінарного виявлення атак

№	Програмна процедура	Зміст реалізації	Результат виконання
1	Вибір базового ознакового подання	Використання датафрейму new_data, що містить головні компоненти та Attack Type	Формування вхідної основи для побудови бінарної постановки задачі
2	Виділення нормального трафіку	Відбір записів з Attack Type == 'BENIGN'	Формування підмножини normal_traffic
3	Виділення атакувального трафіку	Відбір записів з Attack Type != 'BENIGN'	Формування підмножини intrusions
4	Вирівнювання структури вибірки	Вибіркова побудова підмножини нормального трафіку в обсязі, співмірному з кількістю атакувальних записів	Зменшення дисбалансу між класами
5	Інтеграція підвибірок	Об’єднання атакувального трафіку та вибраного нормального трафіку в єдиний датафрейм	Формування робочої структури ids_data
6	Бінаризація цільової змінної	Перетворення Attack Type у двійкову мітку 0/1	Побудова бінарного подання цільового класу
7	Формування робочої бінарної вибірки	Побудова датафрейму bc_data для подальшого експериментального дослідження	Отримання кінцевої вибірки для задачі бінарної класифікації
8	Контроль структури вибірки	Перевірка кількості записів у кожному класі	Підтвердження коректності побудови двокласової структури
9	Формування ознак і міток	Побудова X_bc та y_bc	Отримання матриці ознак і вектора цільових міток для подальшого машинного навчання

Оскільки кількість записів нормального і атакувального трафіку в інтегрованому наборі даних є нерівномірною, у коді реалізовано початкове вирівнювання структури вибірки. Для цього використовується вибіркова підмножина нормального трафіку, сформована за допомогою операції normal_traffic.sample(n = len(intrusions), replace = False). У результаті створюється збалансована за кількістю спостережень підвибірка нормального трафіку, яка далі об’єднується з датафреймом атакувального трафіку в спільний об’єкт ids_data = pd.concat([intrusions, normal_traffic]). Така процедура дозволяє зменшити зміщення

майбутньої моделі у бік домінуючого класу та забезпечити коректніші умови для подальшого навчання алгоритмів бінарного виявлення атак [84-86, 125-130].

Наступним кроком у програмній реалізації є побудова цільової змінної для бінарної класифікації. У коді це реалізовано через перетворення стовпця Attack Type в об’єкті ids_data за допомогою функції np.where((ids_data['Attack Type'] == 'BENIGN'), 0, 1). У результаті такого перетворення нормальний трафік кодується значенням 0, а будь-який атакувальний трафік – значенням 1. Після цього формується робочий датафрейм bc_data = ids_data.sample(n = 15000), який використовується як безпосередня основа для подальшого машинного навчання. Таким чином, у програмному модулі послідовно виконуються два пов’язані етапи: бінаризація цільової мітки та формування робочої підвибірки для проведення порівняльного експерименту.

Після побудови бінарної мітки в програмному модулі виконується контроль структури сформованої вибірки. Для цього в коді використовується виклик bc_data['Attack Type'].value_counts(), який дозволяє перевірити кількість спостережень для кожного з двох класів. Такий контроль є необхідним елементом розроблення програмного модуля, оскільки дає змогу підтвердити коректність бінаризації та придатність сформованого набору до подальшого навчання моделей виявлення атак.

На наступному етапі з побудованого датафрейму bc_data формується матриця ознак і вектор цільових міток для бінарної постановки задачі. Для цього в коді використано конструкції X_bc = bc_data.drop('Attack Type', axis = 1) та y_bc = bc_data['Attack Type']. У результаті всі головні компоненти, що описують характеристики мережевих потоків, переходять до матриці X_bc, тоді як двійкова інформація про клас трафіку зберігається у векторі y_bc. Ці об’єкти надалі використовуються як вхідні дані для поділу на навчальну та тестову підмножини та для реалізації моделей логістичної регресії й методу опорних векторів у задачі бінарного виявлення атак.

У межах програмної організації цей етап є окремим функціональним блоком інформаційної технології, що забезпечує перехід від загального багатокласового

подання мережевого трафіку до спрощеного режиму аналізу, орієнтованого на швидке встановлення факту атакувальної активності. Саме на цьому рівні програмний модуль формує структуру даних, придатну для первинного інтелектуального моніторингу трафіку комп'ютерної мережі, коли основною метою є розмежування безпечного та небезпечного трафіку. Такий режим є особливо важливим для систем виявлення атак, у яких первинне відокремлення підозрілих потоків може слугувати основою для автоматизованого реагування або передачі трафіку на подальший детальніший аналіз.

Результатом програмної реалізації цього етапу є формування повністю підготовленої бінарної вибірки bc_data і відповідних об'єктів X_bc та y_bc, придатних для наступних процедур поділу даних, навчання моделей машинного навчання та оцінювання їх здатності до виявлення атак. У програмному модулі таким чином реалізовано послідовність дій від виділення нормального та атакувального трафіку з датафрейму new_data до побудови кінцевої двокласової структури, яка використовується в задачі бінарного інтелектуального моніторингу трафіку комп'ютерної мережі.

3.10 Програмна реалізація формування та балансування вибірки для багатокласового розпізнавання типів атак

Поряд із бінарним виявленням атак у межах розробленої інформаційної технології реалізовано режим багатокласового аналізу трафіку комп'ютерної мережі, призначений для розпізнавання конкретних типів атак. На відміну від бінарної постановки, у якій усі прояви атакувальної активності об'єднуються в один узагальнений клас, багатокласовий режим дозволяє деталізувати результат інтелектуального моніторингу та встановити, до якого саме типу належить виявлений атакувальний трафік. Отримання такої деталізованої класифікаційної інформації підвищує аналітичну цінність результатів і створює передумови для подальшого класифікаційного супроводу мережевих інцидентів.

У програмній реалізації формування багатокласової вибірки здійснюється на основі датафрейму new_data, який містить головні компоненти ознакового простору та стовпець Attack Type. Початковим кроком цього етапу є аналіз представленості окремих класів атак у наборі даних. Для цього в коді формується об’єкт class_counts = new_data['Attack Type'].value_counts(), який дозволяє визначити кількість спостережень для кожного типу мережевого трафіку. Така процедура виконує важливу діагностичну функцію, оскільки дає змогу оцінити ступінь дисбалансу між класами та виявити категорії атак, представлені недостатньою кількістю записів для подальшого надійного навчання моделей машинного навчання.

З метою формування робочої багатокласової вибірки в коді реалізовано відбір лише тих класів, кількість спостережень у яких перевищує поріг 1950. Для цього використовується конструкція selected_classes = class_counts[class_counts > 1950], після чого формується перелік допустимих класів class_names = selected_classes.index. Далі датафрейм new_data фільтрується за допомогою виразу selected = new_data[new_data['Attack Type'].isin(class_names)]. Застосування такої процедури фільтрації дає змогу виключити з подальшого розгляду рідкісні категорії, для яких побудова статистично надійної багатокласової моделі ускладнюється недостатньою кількістю прикладів. У результаті до аналізу включаються лише ті класи трафіку, які є достатньо репрезентативними для подальшого моделювання [139].

Після відбору репрезентативних класів у програмному модулі виконується додаткове обмеження обсягу окремих категорій трафіку, що мають надмірно велику кількість записів. Для цього в коді реалізовано цикл за елементами class_names, у межах якого для кожного класу формується локальний датафрейм df = selected[selected['Attack Type'] == name]. Якщо кількість записів у такому датафреймі перевищує 2500, виконується випадковий відбір 5000 спостережень за допомогою df.sample(n = 5000, random_state = 0). Після цього всі підвибірки накопичуються у списку dfs, а далі об’єднуються в єдиний робочий датафрейм через df = pd.concat(dfs, ignore_index = True). Така процедура дозволяє зменшити

надмірне домінування окремих класів і сформувати більш придатну структуру даних для наступного етапу синтетичного балансування.

На наступному етапі у програмній реалізації формується матриця ознак і вектор міток для багатокласової задачі. Для цього використовуються конструкції X = df.drop('Attack Type', axis = 1) та y = df['Attack Type']. Таким чином, головні компоненти, що описують характеристики мережевих потоків, переходять до матриці X, а мітки конкретних типів атак або нормального трафіку – до вектора y. Ці об’єкти далі використовуються для процедури синтетичного балансування та побудови остаточної багатокласової вибірки.

Оскільки навіть після попереднього відбору класів і часткового обмеження великих підвибірок у робочому наборі зберігається нерівномірність представленості окремих категорій, у коді реалізовано синтетичне балансування за допомогою методу SMOTE. Для цього створюється об’єкт smote = SMOTE(sampling_strategy='auto', random_state = 0), після чого виконується операція X_upsampled, y_upsampled = smote.fit_resample(X, y). У результаті цієї процедури формується нова збалансована множина ознак і міток, у якій класи меншості доповнюються синтетично згенерованими спостереженнями на основі наявних прикладів. Застосування SMOTE на цьому етапі дозволяє підвищити репрезентативність рідкісніших типів атак і створити кращі умови для подальшого навчання багатокласових моделей машинного навчання [82, 83, 125-130].

Результат балансування в програмному модулі зберігається у вигляді нового датафрейму blnc_data, який формується через pd.DataFrame(X_upsampled) з подальшим додаванням цільового стовпця Attack Type = y_upsampled. Після цього датафрейм додатково перемішується за допомогою blnc_data = blnc_data.sample(frac = 1), що дозволяє усунути вплив початкового порядку записів. Контроль структури отриманої вибірки виконується через blnc_data['Attack Type'].value_counts(), що дає змогу перевірити розподіл спостережень між класами після балансування. За результатами цієї процедури формується остаточна збалансована багатокласова вибірка, яка в подальшому використовується для навчання моделей розпізнавання типів атак [70, 72].

З погляду логіки функціонування програмного модуля етап формування та балансування багатокласової вибірки виконує роль переходу від узагальненого ознакового простору до спеціалізованого режиму розпізнавання типів атак. Якщо на попередньому етапі було сформовано бінарну вибірку для виявлення факту атакувального впливу, то тут реалізовано окрему гілку програмного конвеєра, у межах якої здійснюється деталізоване моделювання багатокласової структури трафіку комп'ютерної мережі. Саме тому сформований датафрейм blnc_data, а також похідні від нього об'єкти features і labels, виступають базою для подальшого навчання моделей випадкового лісу, дерева рішень і методу k-найближчих сусідів у задачі розпізнавання типів атак.

Основні програмні структури даних, що формуються та використовуються на етапі побудови й балансування багатокласової вибірки, наведено в табл. 3.8.

Таблиця 3.8 – Основні програмні структури даних етапу формування та балансування багатокласової вибірки

Назва об'єкта	**Тип структури**	**Призначення у програмному модулі**
new_data	DataFrame	Базовий датафрейм, що містить головні компоненти ознакового простору та стовпець Attack Type
class_counts	Series	Визначення кількості спостережень для кожного класу Attack Type
selected_classes	Series	Збереження тих класів, кількість записів у яких перевищує поріг відбору
class_names	Index	Перелік назв класів, допущених до багатокласового аналізу
selected	DataFrame	Проміжний датафрейм, що містить лише вибрані репрезентативні класи
dfs	list	Накопичення підвибірок окремих класів перед побудовою спільного датафрейму
df	DataFrame	Робочий датафрейм до синтетичного балансування
X	DataFrame	Матриця ознак для багатокласової задачі до застосування SMOTE
y	Series	Вектор міток класів для багатокласової задачі до застосування SMOTE
smote	SMOTE	Об'єкт синтетичного балансування класів методом SMOTE
X_upsampled	DataFrame / ndarray	Матриця ознак після синтетичного балансування
y_upsampled	Series / ndarray	Вектор міток після синтетичного балансування
blnc_data	DataFrame	Підсумковий збалансований датафрейм для багатокласового розпізнавання типів атак

Продовження таблиці 3.8

Назва об’єкта	Тип структури	Призначення у програмному модулі
features	DataFrame	Матриця ознак, сформована з blnc_data для подальшого навчання моделей
labels	Series	Вектор цільових міток, сформований з blnc_data для подальшого навчання моделей

Результатом цього етапу є збалансована багатокласова вибірка, побудована на основі датафрейму new_data шляхом послідовного аналізу представленості класів, відбору репрезентативних категорій, обмеження надмірно великих підвибірок, застосування синтетичного балансування методом SMOTE та формування підсумкового датафрейму blnc_data. Така послідовність програмних дій створює основу для подальшого навчання моделей багатокласового інтелектуального моніторингу трафіку комп’ютерної мережі.

Послідовність основних програмних процедур, реалізованих на етапі формування та балансування вибірки для багатокласового розпізнавання типів атак, наведено в табл. 3.9.

Таблиця 3.9 – Основні програмні процедури етапу формування та балансування багатокласової вибірки

№	Програмна процедура	Зміст реалізації	Результат виконання
1	Аналіз представленості класів	Обчислення new_data['Attack Type'].value_counts()	Визначення кількості спостережень для кожного типу трафіку
2	Відбір репрезентативних класів	Формування selected_classes = class_counts[class_counts > 1950]	Виключення класів із недостатньою кількістю спостережень
3	Формування переліку класів для аналізу	Побудова class_names = selected_classes.index	Отримання списку класів, допущених до багатокласового моделювання
4	Фільтрація базового датафрейму	Побудова selected = new_data[new_data['Attack Type'].isin(class_names)]	Формування проміжного датафрейму з вибраними класами
5	Обмеження надмірно великих класів	Цикл по class_names з використанням df.sample(n = 5000, random_state = 0) для великих підвибірок	Зменшення домінування класів з надто великою кількістю записів
6	Інтеграція підвибірок	Побудова df = pd.concat(dfs, ignore_index = True)	Формування робочої багатокласової вибірки до балансування

Продовження таблиці 3.9

№	Програмна процедура	Зміст реалізації	Результат виконання
7	Формування матриці ознак і вектора міток	Побудова X = df.drop('Attack Type', axis = 1) та y = df['Attack Type']	Отримання вхідних даних для застосування SMOTE
8	Синтетичне балансування класів	Застосування smote.fit_resample(X, y)	Побудова збалансованих множин X_upsampled і y_upsampled
9	Формування збалансованого датафрейму	Побудова blnc_data = pd.DataFrame(X_upsampled) з додаванням Attack Type = y_upsampled	Отримання підсумкової багатокласової вибірки
10	Перемішування записів	Виконання blnc_data = blnc_data.sample(frac = 1)	Усунення впливу початкового порядку розташування записів
11	Контроль структури вибірки	Використання blnc_data['Attack Type'].value_counts()	Перевірка розподілу класів після балансування
12	Підготовка до навчання моделей	Побудова features = blnc_data.drop('Attack Type', axis = 1) та labels = blnc_data['Attack Type']	Формування ознак і міток для подальшого навчання багатокласових моделей

3.11 Програмна реалізація моделей машинного навчання для бінарного та багатокласового аналізу мережевого трафіку

3.11.1 Загальна організація програмної реалізації моделей машинного навчання

Після формування та підготовки вибірок для бінарного виявлення атак і багатокласового розпізнавання їх типів у програмному модулі реалізується етап побудови моделей машинного навчання. Цей етап становить інтелектуальне ядро розробленої інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак, оскільки на ньому сформований ознаковий опис мережевих потоків використовується для одержання класифікаційних рішень. У програмній реалізації моделі поділено на дві функціональні групи відповідно до двох режимів роботи інформаційної технології: бінарного аналізу, орієнтованого на виявлення факту атаки, та багатокласового аналізу, спрямованого на розпізнавання конкретного типу атакувальної активності.

З позиції проєктування програмного забезпечення реалізація моделей машинного навчання побудована за принципом уніфікованого програмного контуру, у межах якого кожна модель розглядається як окремий конфігурований програмний компонент. Для кожного такого компонента задаються вхідні дані, набір параметрів ініціалізації, процедура навчання, механізм перехресної перевірки та набір вихідних результатів, що надалі передаються на етап оцінювання. Така організація дозволяє не лише забезпечити узгодженість реалізації різних методів машинного навчання, а й створює зручну основу для їх порівняння за однакових умов експерименту.

Для бінарної постановки задачі вхідними структурами є матриця ознак X_bc та вектор цільових міток y_bc, сформовані на попередньому етапі з датафрейму bc_data. Після цього у програмному модулі виконується поділ даних на навчальну та тестову підмножини за допомогою train_test_split(X_bc, y_bc, test_size = 0.25, random_state = 0), у результаті чого створюються об’єкти X_train_bc, X_test_bc, y_train_bc і y_test_bc. Ці структури використовуються як стандартний вхідний контракт для моделей логістичної регресії та методу опорних векторів [90].

Для багатокласового режиму вхід до підсистеми моделювання формується на основі збалансованого датафрейму blnc_data, отриманого після синтетичного балансування методом SMOTE. Із нього виділяються ознаки features та мітки класів labels, після чого також виконується поділ на навчальні та тестові підмножини X_train, X_test, y_train, y_test. На цих структурах реалізуються моделі випадкового лісу, дерева рішень і методу k-найближчих сусідів. Таким чином, на рівні програмної архітектури обидві гілки моделювання мають спільну схему роботи, але відрізняються типом вхідних даних і набором застосованих алгоритмів.

Суттєвою особливістю програмної реалізації є використання для кожного методу не однієї, а двох конфігурацій моделі. Це дає змогу розглядати окремий алгоритм не як єдину жорстко задану сутність, а як параметризований клас програмних рішень, поведінка якого залежить від вибраних налаштувань. У межах бінарного режиму це реалізовано для логістичної регресії (lr1, lr2) та методу опорних векторів (svm1, svm2), а в межах багатокласового режиму – для

випадкового лісу (rf1, rf2), дерева рішень (dt1, dt2) і методу k-найближчих сусідів (knn1, knn2). Така організація програмного модуля забезпечує можливість контрольованого порівняння різних конфігурацій одного й того самого методу без зміни загальної логіки експерименту [131-134].

З погляду життєвого циклу програмна взаємодія з кожною моделлю реалізована за уніфікованою схемою. На першому кроці створюється екземпляр моделі з конкретними параметрами ініціалізації. На другому кроці виконується навчання моделі на відповідній навчальній вибірці через виклик методу .fit(). На третьому кроці здійснюється оцінювання стійкості моделі до варіацій навчальних даних за допомогою cross_val_score(..., cv = 5), у результаті чого формуються об’єкти cv_*, що зберігають оцінки перехресної перевірки для кожної конфігурації. На четвертому кроці підготовлені моделі використовуються для побудови прогнозів і передаються до наступного підрозділу, у якому виконується розрахунок метрик якості та візуалізація результатів класифікації [91, 121, 122].

У межах такої архітектури важливу роль відіграє відтворюваність програмної реалізації. Вона забезпечується через фіксовані параметри random_state, єдину схему поділу вибірок, узгоджений формат вхідних структур і однакову процедуру перехресної перевірки для всіх конфігурацій моделей. Завдяки цьому програмний модуль дозволяє не лише навчати окремі класифікатори, а й реалізовувати контрольоване порівняння їх поведінки в межах однорідного обчислювального середовища. Така організація програмної реалізації є принципово важливою для інформаційної технології інтелектуального моніторингу трафіку, оскільки забезпечує коректність експериментального аналізу й придатність програмної реалізації до подальшого розширення [105, 106, 135-138].

Таким чином, у межах програмного модуля реалізовано уніфіковану підсистему побудови моделей машинного навчання, яка підтримує дві гілки аналізу мережевого трафіку, використовує стандартизовані вхідні структури даних, забезпечує параметризовану ініціалізацію моделей, їх навчання, перехресну перевірку та підготовку результатів до наступного етапу оцінювання. Подальший виклад присвячено детальному розгляду окремих методів машинного навчання, їх

конфігурацій та параметрів налаштування в межах бінарного і багатокласового режимів функціонування інформаційної технології.

3.11.2 Програмна реалізація моделей для бінарного виявлення атак

Програмна реалізація логістичної регресії

Логістична регресія використовується в програмному модулі як базовий лінійний класифікатор для задачі бінарного виявлення атак. Її включення до складу інформаційної технології є доцільним з двох причин. По-перше, ця модель забезпечує інтерпретоване лінійне розмежування двох класів у сформованому ознаковому просторі. По-друге, вона дозволяє оцінити, наскільки побудоване представлення мережевого трафіку придатне для розв’язання задачі виявлення атак без використання складніших нелінійних механізмів. У межах програмної реалізації логістична регресія розглядається як один із базових програмних компонентів бінарної гілки аналізу [92, 107, 108, 110, 111].

У коді реалізовано дві конфігурації логістичної регресії:

lr1 = LogisticRegression(max_iter = 10000, C = 0.1, random_state = 0, solver = 'saga')

та lr2 = LogisticRegression(max_iter = 15000, solver = 'sag', C = 100, random_state = 0).

Обидві конфігурації навчаються на підмножині X_train_bc, y_train_bc за допомогою виклику методу .fit(). Така організація дозволяє в межах єдиного програмного контуру дослідити вплив різних варіантів регуляризації та алгоритмів оптимізації на якість бінарного виявлення атак [92, 131-134].

Параметр max_iter визначає максимальну кількість ітерацій, що допускаються під час числової оптимізації моделі. У контексті даної реалізації його збільшене значення використовується для забезпечення збіжності алгоритму на великій навчальній вибірці. Параметр C задає силу регуляризації: менше значення C = 0.1 у моделі lr1 відповідає сильнішому регуляризуванню, тоді як C = 100 у моделі lr2 послаблює регуляризаційний вплив і дозволяє моделі гнучкіше

підлаштовуватися під навчальні дані. Параметр solver визначає алгоритм оптимізації; у програмному модулі реалізовано порівняння двох його варіантів – saga та sag. Параметр random_state = 0 забезпечує відтворюваність результатів навчання. З погляду проєктування ПЗ ці налаштування задають дві альтернативні конфігурації одного класифікаційного компонента з різним ступенем жорсткості та різною обчислювальною поведінкою.

Після навчання для кожної конфігурації додатково виконується перехресна перевірка за допомогою cross_val_score(..., cv = 5), у результаті чого формуються об’єкти cv_lr1 та cv_lr2. Крім того, у програмному модулі виводяться коефіцієнти coef_ і вільні члени intercept_, що дозволяє одержати допоміжну інформацію про побудовані лінійні класифікаційні правила. Таким чином, програмна реалізація логістичної регресії охоплює не лише навчання моделі, а й контроль її стійкості та аналіз параметрів сформованої роздільної функції [91, 92, 121, 122].

Програмна реалізація методу опорних векторів

Другим методом, реалізованим у бінарній гілці програмного модуля, є метод опорних векторів. Його використання обумовлене здатністю будувати ефективні роздільні поверхні в ознаковому просторі мережевого трафіку, у тому числі в умовах нелінійного розподілу класів. У межах інформаційної технології цей метод виконує роль більш гнучкого класифікаційного компонента, який дозволяє перевірити, чи забезпечує нелінійне ядрове відображення кращу здатність до розмежування нормального та атакувального трафіку порівняно з лінійною моделлю логістичної регресії [93, 107, 108].

У програмному коді реалізовано дві конфігурації методу опорних векторів:

svm1 = SVC(kernel = 'poly', C = 1, random_state = 0, probability = True)

та svm2 = SVC(kernel = 'rbf', C = 1, gamma = 0.1, random_state = 0, probability = True).

Обидві моделі навчаються на підмножині X_train_bc, y_train_bc і включені до складу єдиного програмного контуру бінарного аналізу. Реалізація двох варіантів SVM у межах одного модуля дозволяє зіставити поліноміальне та

радіально-базисне ядра в задачі виявлення атак і визначити, яка конфігурація краще узгоджується з побудованим ознаковим простором мережевого трафіку.

Параметр kernel визначає тип ядрового перетворення, на основі якого формується роздільна поверхня між класами. У моделі svm1 використовується поліноміальне ядро poly, тоді як у svm2 – радіально-базисне ядро rbf, яке забезпечує вищу гнучкість у разі складної нелінійної структури даних. Параметр C = 1 задає баланс між шириною роздільної смуги та штрафом за помилки класифікації. Для svm2 додатково використовується параметр gamma = 0.1, який визначає радіус впливу окремих навчальних спостережень у випадку rbf-ядра. Параметр probability = True забезпечує можливість подальшого оцінювання імовірностей належності до класів, що є корисним у межах загального програмного контуру оцінювання моделей. Параметр random_state = 0 використовується для забезпечення відтворюваності. З позиції програмної інженерії ці налаштування реалізують дві різні конфігурації нелінійного класифікаційного компонента з однаковим інтерфейсом використання, але з різною геометрією розділення класів.

Після навчання для кожної конфігурації SVM також виконується п’ятикратна перехресна перевірка, у результаті чого формуються об’єкти cv_svm1 та cv_svm2. Додатково в коді виводяться значення intercept_, що дає змогу отримати допоміжну інформацію про параметри побудованої гіперплощини. Таким чином, програмна реалізація методу опорних векторів інтегрує в межах одного функціонального блоку побудову двох альтернативних конфігурацій моделі, їх навчання, перевірку стійкості та підготовку до подальшого оцінювання якості бінарного виявлення атак [91, 93, 121, 122].

У межах бінарної гілки програмного модуля логістична регресія та метод опорних векторів реалізовані як взаємодоповнювальні програмні компоненти. Логістична регресія використовується як базова лінійна модель класифікації, тоді як метод опорних векторів застосовується для перевірки ефективності нелінійного ядрового розділення простору ознак. Наявність двох конфігурацій кожного методу дозволяє не лише порівнювати різні алгоритми між собою, а й оцінювати вплив параметричних налаштувань у межах одного класу моделей. У такий спосіб

програмна реалізація бінарного режиму створює технічно обґрунтовану основу для подальшого етапу оцінювання результатів функціонування моделей виявлення атак.

Конфігурації моделей машинного навчання, реалізованих у бінарній гілці програмного модуля, наведено в табл. 3.10.

Таблиця 3.10 – Конфігурації моделей МН для бінарного аналізу

Позначення моделі	Метод	Програмна конфігурація	Вхідні дані	Призначення у програмному модулі
lr1	Логістична регресія	LogisticRegression(max_iter = 10000, C = 0.1, random_state = 0, solver = 'saga')	X_train_bc, y_train_bc	Базовий лінійний класифікатор для бінарного виявлення атак за умов посиленої регуляризації
lr2	Логістична регресія	LogisticRegression(max_iter = 15000, solver = 'sag', C = 100, random_state = 0)	X_train_bc, y_train_bc	Альтернативна конфігурація лінійного класифікатора зі зменшеним регуляризаційним впливом
svm1	Метод опорних векторів	SVC(kernel = 'poly', C = 1, random_state = 0, probability = True)	X_train_bc, y_train_bc	Нелінійний класифікатор з поліноміальним ядром для бінарного виявлення атак
svm2	Метод опорних векторів	SVC(kernel = 'rbf', C = 1, gamma = 0.1, random_state = 0, probability = True)	X_train_bc, y_train_bc	Нелінійний класифікатор з радіально-базисним ядром для бінарного виявлення атак

3.11.3 Програмна реалізація моделей для багатокласового розпізнавання типів атак

Програмна реалізація випадкового лісу

Випадковий ліс реалізовано в програмному модулі як ансамблевий класифікаційний компонент, орієнтований на побудову стійких багатокласових рішень на основі сукупності дерев рішень. Використання цього методу є доцільним у зв’язку з його здатністю працювати зі складною структурою ознакового простору та зменшувати ризик перенавчання порівняно з окремим деревом. У межах

інформаційної технології випадковий ліс виконує роль базового ансамблевого методу для розпізнавання типів атак [94, 112, 116].

У коді реалізовано дві конфігурації випадкового лісу:

rf1 = RandomForestClassifier(n_estimators = 10, max_depth = 6, max_features = None, random_state = 0)

та rf2 = RandomForestClassifier(n_estimators = 15, max_depth = 8, max_features = 20, random_state = 0).

Обидві конфігурації навчаються на підмножині X_train, y_train за допомогою методу .fit(), а їх стійкість додатково оцінюється через cross_val_score(..., cv = 5), у результаті чого формуються об’єкти cv_rf1 і cv_rf2. Така реалізація дозволяє порівняти дві параметричні конфігурації ансамблевого класифікатора в межах одного програмного контуру [91, 94, 121, 122].

Параметр n_estimators визначає кількість дерев у складі ансамблю. У конфігурації rf1 використовується 10 дерев, тоді як у rf2 – 15, що дозволяє перевірити вплив розміру ансамблю на якість багатокласового розпізнавання. Параметр max_depth задає максимальну глибину кожного дерева і, відповідно, контролює складність моделі: значення 6 у rf1 відповідає більш стриманій структурі, тоді як 8 у rf2 дозволяє будувати складніші дерева. Параметр max_features визначає кількість ознак, які розглядаються під час пошуку найкращого розщеплення; у rf1 використовується повний доступ до ознак, а в rf2 – обмеження до 20. Параметр random_state = 0 використовується для забезпечення відтворюваності. З позиції проєктування ПЗ ці налаштування задають дві різні конфігурації ансамблевого програмного компонента з різним рівнем складності та різною стратегією побудови дерев.

Програмна реалізація дерева рішень

Дерево рішень використовується в програмному модулі як інтерпретований класифікаційний компонент, придатний для побудови багатокласових правил розмежування мережевого трафіку. На відміну від ансамблевої моделі випадкового лісу, дерево рішень реалізує єдину ієрархічну структуру умов, що дозволяє

безпосередньо простежити логіку класифікації. У межах інформаційної технології цей метод виконує роль базового нелінійного класифікатора з простою структурою прийняття рішень [95, 112-114].

У коді реалізовано дві конфігурації дерева рішень:

dt1 = DecisionTreeClassifier(max_depth = 6)

та dt2 = DecisionTreeClassifier(max_depth = 8).

Після ініціалізації обидві моделі навчаються на X_train, y_train, а для оцінювання стійкості додатково обчислюються значення перехресної перевірки cv_dt1 і cv_dt2. Така організація дозволяє дослідити, як зміна максимальної глибини дерева впливає на здатність моделі розрізняти кілька типів атак у сформованому ознаковому просторі [91, 95, 121, 122].

Параметр max_depth є ключовим параметром налаштування цього методу, оскільки він визначає максимально допустиму глибину дерева та, відповідно, рівень деталізації класифікаційних правил. Менше значення max_depth = 6 забезпечує більш компактну і стриману структуру дерева, тоді як max_depth = 8 дозволяє формувати складніші правила розділення класів. З позиції програмної реалізації дві конфігурації dt1 і dt2 є варіантами одного й того самого програмного компонента з різним рівнем глибини, що дає змогу оцінити компроміс між простотою моделі та її роздільною здатністю.

Програмна реалізація методу k-найближчих сусідів

Метод k-найближчих сусідів реалізовано в програмному модулі як класифікаційний компонент, що приймає рішення на основі близькості об’єктів у сформованому ознаковому просторі. На відміну від випадкового лісу та дерева рішень, цей метод не виконує явного побудування моделі у вигляді дерева чи ансамблю, а використовує безпосереднє порівняння нових спостережень із навчальними прикладами. У межах інформаційної технології його застосування є доцільним для перевірки того, наскільки добре головні компоненти простору ознак забезпечують просторове розділення типів атак [96, 107, 108, 110, 111].

У програмному модулі реалізовано дві конфігурації методу:

knn1 = KNeighborsClassifier(n_neighbors = 16)

та knn2 = KNeighborsClassifier(n_neighbors = 8).

Після ініціалізації обидві конфігурації навчаються на X_train, y_train, а для кожної з них обчислюються результати перехресної перевірки cv_knn1 і cv_knn2. У межах програмної реалізації це дозволяє дослідити вплив кількості сусідів на якість багатокласового аналізу мережевого трафіку без зміни загальної структури програмного конвеєра [91, 96, 121, 122].

Параметр n_neighbors визначає кількість найближчих навчальних спостережень, на основі яких формується класифікаційне рішення. У конфігурації knn1 використовується 16 сусідів, що відповідає більш згладженому рішенню, тоді як у knn2 використовується 8 сусідів, що дозволяє моделі сильніше реагувати на локальну структуру розподілу даних. З точки зору програмної інженерії ці дві конфігурації реалізують два різні рівні чутливості одного й того самого програмного компонента до локальних відмінностей між класами в ознаковому просторі.

Багатокласова гілка програмного модуля побудована таким чином, щоб усі три методи – випадковий ліс, дерево рішень і метод k-найближчих сусідів – працювали в межах спільної схеми: ініціалізація конфігурації, навчання на X_train, y_train, обчислення оцінок перехресної перевірки та підготовка до наступного етапу оцінювання. Така уніфікація реалізації забезпечує порівнюваність результатів, спрощує інтеграцію моделей у загальний програмний модуль і дозволяє розглядати кожен алгоритм як окремий взаємозамінний програмний компонент багатокласової підсистеми розпізнавання типів атак.

Конфігурації моделей машинного навчання, реалізованих у багатокласовій гілці програмного модуля, наведено в табл. 3.11, а основні параметри налаштування моделей та їх функціональна роль у програмному модулі – в табл. 3.12.

Таблиця 3.11 – Конфігурації моделей машинного навчання для багатокласового аналізу

Позначення моделі	**Метод**	**Програмна конфігурація**	**Вхідні дані**	**Призначення у програмному модулі**
rf1	Випадковий ліс	RandomForestClassifier(n_estimators = 10, max_depth = 6, max_features = None, random_state = 0)	X_train, y_train	Ансамблевий класифікатор для багатокласового розпізнавання типів атак із помірною складністю
rf2	Випадковий ліс	RandomForestClassifier(n_estimators = 15, max_depth = 8, max_features = 20, random_state = 0)	X_train, y_train	Ансамблевий класифікатор із підвищеною складністю та обмеженням кількості ознак при розщепленні
dt1	Дерево рішень	DecisionTreeClassifier(max_depth = 6)	X_train, y_train	Базова конфігурація дерева рішень для багатокласової класифікації
dt2	Дерево рішень	DecisionTreeClassifier(max_depth = 8)	X_train, y_train	Поглиблена конфігурація дерева рішень для побудови складніших правил розділення
knn1	k-найближчих сусідів	KNeighborsClassifier(n_neighbors = 16)	X_train, y_train	Конфігурація k-NN із більш згладженим прийняттям рішення
knn2	k-найближчих сусідів	KNeighborsClassifier(n_neighbors = 8)	X_train, y_train	Конфігурація k-NN з підвищеною чутливістю до локальної структури даних

Таблиця 3.12 – Основні параметри налаштування моделей машинного навчання та їх функціональна роль у програмному модулі

Параметр	Метод / модель	Функціональна роль у програмному модулі	Очікуваний вплив на поведінку моделі
max_iter	Логістична регресія (lr1, lr2)	Обмежує кількість ітерацій оптимізації під час навчання	Впливає на досягнення збіжності моделі на великій вибірці
C	Логістична регресія (lr1, lr2), SVM (svm1, svm2)	Регулює баланс між жорсткістю регуляризації та точністю підлаштування до даних	Менші значення підсилюють регуляризацію, більші – роблять модель гнучкішою
solver	Логістична регресія (lr1, lr2)	Визначає алгоритм числової оптимізації	Впливає на швидкість і стабільність навчання
kernel	SVM (svm1, svm2)	Задає тип ядрового перетворення ознакового простору	Визначає геометрію роздільної поверхні між класами
gamma	SVM (svm2)	Керує радіусом впливу окремих спостережень для rbf-ядра	Впливає на локальність та гнучкість розділення класів
probability	SVM (svm1, svm2)	Дозволяє формувати імовірнісні оцінки належності до класів	Розширює можливості подальшого оцінювання моделі
n_estimators	Випадковий ліс (rf1, rf2)	Визначає кількість дерев у складі ансамблю	Збільшення кількості дерев підвищує стійкість ансамблю, але збільшує обчислювальні витрати
max_depth	Випадковий ліс (rf1, rf2), дерево рішень (dt1, dt2)	Обмежує глибину дерева або дерев ансамблю	Впливає на складність моделі та ризик перенавчання
max_features	Випадковий ліс (rf1, rf2)	Визначає кількість ознак, доступних при побудові розщеплень	Впливає на різноманітність дерев ансамблю та його узагальнювальну здатність
n_neighbors	k-найближчих сусідів (knn1, knn2)	Визначає кількість сусідів, за якими формується рішення	Менше значення підвищує чутливість до локальної структури, більше – забезпечує згладження рішення
random_state	lr1, lr2, svm1, svm2, rf1, rf2	Забезпечує відтворюваність результатів програмної реалізації	Дозволяє повторювати експерименти за однакових умов

3.12 Програмна реалізація оцінювання та візуалізації результатів класифікації мережевого трафіку

Завершальним етапом функціонування програмного модуля є оцінювання та

візуалізація результатів класифікації мережевого трафіку, отриманих у процесі бінарного виявлення атак і багатокласового розпізнавання їх типів. На цьому етапі виконується перехід від навчених моделей та їх прогнозів до кількісно й аналітично інтерпретованих результатів, які дозволяють оцінити ефективність розробленої інформаційної технології. Якщо попередні етапи програмної реалізації були пов'язані з підготовкою даних, формуванням ознакового простору, побудовою вибірок і навчанням моделей, то підсистема оцінювання забезпечує аналітичне завершення програмного конвеєра та формує основу для подальшого експериментального дослідження [97-104, 121-124].

З погляду програмної архітектури оцінювання реалізовано як уніфіковану підсистему, що працює з вихідними артефактами попереднього етапу моделювання. У межах підсистеми оцінювання для кожної конфігурації реалізуються прогнозування, розрахунок показників якості, формування матриць помилок, побудова покласових звітів і створення графічних засобів порівняння. Така організація дозволяє забезпечити єдину логіку аналізу для всіх моделей та коректність їх зіставлення за однакових умов.

Програмна реалізація оцінювання моделей бінарного виявлення атак

Для бінарної постановки задачі оцінювання здійснюється на основі тестових підмножин X_test_bc і y_test_bc. У програмному коді для моделей логістичної регресії формуються прогнози y_pred_lr1 і y_pred_lr2, а також імовірнісні оцінки y_prob_lr1 і y_prob_lr2, які використовуються для подальшого ROC- та Precision–Recall-аналізу. Аналогічно для методу опорних векторів обчислюються прогнози y_pred_svm1 і y_pred_svm2, а також імовірнісні оцінки y_prob_svm1 і y_prob_svm2. У такий спосіб у підсистемі оцінювання реалізовано повний цикл роботи з бінарними моделями: від побудови прогнозів до формування даних для числового та графічного аналізу.

Першим рівнем оцінювання бінарних моделей є розрахунок інтегрального показника точності класифікації за допомогою accuracy_score. Для конфігурацій логістичної регресії та SVM обчислюються окремі значення accuracy, які далі

використовуються для побудови горизонтальних стовпчикових діаграм. Окремо в програмному модулі враховуються середні результати п’ятикратної перехресної перевірки cv_lr1.mean(), cv_lr2.mean(), cv_svm1.mean(), cv_svm2.mean(). Завдяки цьому оцінювання не обмежується лише тестовою точністю, а доповнюється аналізом стійкості моделей на навчальних підмножинах. У програмній реалізації передбачено як порівняння двох конфігурацій у межах одного методу, так і інтегральне порівняння найкращих бінарних моделей – lr2 і svm2 [91, 99].

Окремою складовою підсистеми оцінювання є побудова матриць помилок. Для цього в коді використовується функція confusion_matrix, а результати візуалізуються через seaborn.heatmap. Для кожної конфігурації логістичної регресії та SVM формуються окремі матриці помилок, які дозволяють оцінити кількість правильно виявлених атак, правильно класифікованого нормального трафіку, хибнопозитивних спрацювань і пропущених атак. У контексті інформаційної технології це має особливе значення, оскільки дає змогу перейти від загального показника точності до аналізу структури класифікаційних помилок, що є критично важливим для систем виявлення атак [80, 98].

Для більш детальної характеристики якості бінарних моделей у програмному модулі реалізовано формування звітів класифікації за допомогою classification_report(..., output_dict = True). Отримані словники покласових показників перетворюються на масиви значень precision, recall і f1-score, після чого візуалізуються у вигляді теплових карт. Така схема оцінювання дає змогу не лише отримати числові характеристики для кожного класу, а й наочно порівняти різні конфігурації моделей між собою. У межах програмного модуля це формує окремий аналітичний блок, орієнтований на визначення сильних і слабких сторін класифікаторів у задачі розмежування нормального та атакувального трафіку.

Важливою частиною оцінювання бінарних моделей є побудова ROC-кривих. У коді для цього використовуються функції roc_curve та auc, на основі яких формуються значення fpr, tpr і площа під ROC-кривою для кожної конфігурації моделей логістичної регресії та SVM. Програмна реалізація включає як окремі графіки для кожної моделі, так і суміщені ROC-графіки для порівняння

конфігурацій у межах одного методу та між найкращими моделями бінарної класифікації. Це дозволяє оцінити здатність моделей відокремлювати атакувальний трафік від нормального за різних порогів прийняття рішення [100, 101, 103, 123].

Поряд із ROC-аналізом у програмному модулі реалізовано побудову Precision–Recall-кривих на основі precision_recall_curve. Для кожної конфігурації бінарних моделей будуються індивідуальні та суміщені графіки, що дозволяють оцінити співвідношення між точністю позитивних рішень і повнотою виявлення атак. Для задач інтелектуального моніторингу трафіку така форма оцінювання є важливою, оскільки дозволяє глибше аналізувати поведінку моделі в умовах нерівномірного розподілу класів. З погляду програмної реалізації Precision–Recall-аналіз виступає додатковим модулем візуального контролю якості бінарних класифікаторів [102, 104].

Програмна реалізація оцінювання моделей багатокласового розпізнавання типів атак

Для багатокласового режиму оцінювання виконується на основі тестових підмножин X_test і y_test, сформованих із датафрейму blnc_data. У коді для кожної конфігурації випадкового лісу, дерева рішень і методу k-найближчих сусідів будуються прогнози y_pred_rf1, y_pred_rf2, y_pred_dt1, y_pred_dt2, y_pred_knn1, y_pred_knn2. Усі ці результати надалі використовуються в єдиній схемі розрахунку метрик і побудови візуалізацій. Таким чином, багатокласова підсистема оцінювання реалізована як окремий функціональний блок, що працює з підготовленими прогнозами моделей і формує аналітичне подання якості розпізнавання різних типів атак.

Як і в бінарному режимі, першим рівнем оцінювання є розрахунок accuracy_score. Для кожної пари конфігурацій rf1/rf2, dt1/dt2, knn1/knn2 обчислюються окремі значення точності, які візуалізуються у вигляді горизонтальних стовпчикових діаграм. Додатково для кожної конфігурації враховуються середні результати перехресної перевірки cv_rf1.mean(),

cv_rf2.mean(), cv_dt1.mean(), cv_dt2.mean(), cv_knn1.mean(), cv_knn2.mean(). Окрім парного порівняння в межах одного методу, у програмному модулі реалізовано інтегральне порівняння найкращих багатокласових конфігурацій – rf2, dt2 і knn2 – за значеннями точності та середніми оцінками cross-validation. Це створює основу для вибору найбільш придатного методу багатокласового розпізнавання типів атак [91, 99, 121, 122].

Для багатокласових моделей у програмному модулі також реалізовано побудову матриць помилок через confusion_matrix з подальшою візуалізацією через seaborn.heatmap. На відміну від бінарної постановки, тут матриці помилок мають розмірність, що відповідає кількості класів Attack Type, і дозволяють аналізувати не лише загальну кількість правильних та помилкових рішень, а й характер переплутування конкретних типів атак між собою. Така форма подання є важливою, оскільки дозволяє виявити найбільш проблемні зони багатокласового розпізнавання в межах розробленої інформаційної технології [80, 98].

Покласове оцінювання багатокласових моделей реалізовано через classification_report(..., output_dict = True) з подальшим формуванням масивів precision, recall і f1-score. Для випадкового лісу, дерева рішень і методу k-найближчих сусідів будуються теплові карти покласових метрик, які дають змогу порівнювати здатність моделей розпізнавати різні категорії атак. У програмній реалізації це важливо, оскільки для багатокласової задачі саме покласові показники дають повніше уявлення про якість моделі, ніж один інтегральний показник точності. Окремо передбачено інтегральне візуальне порівняння найкращих конфігурацій багатокласових методів за тепловими картами звітів класифікації [97, 103].

Уніфікована схема програмної взаємодії засобів оцінювання та візуалізації

Суттєвою особливістю програмної реалізації є те, що оцінювання та візуалізація організовані за уніфікованим принципом для всіх моделей, інтегрованих у програмний модуль. Незалежно від того, чи йдеться про логістичну

регресію, метод опорних векторів, випадковий ліс, дерево рішень або метод k-найближчих сусідів, для кожного класифікатора використовується спільна схема обробки результатів: прогнозування на тестовій вибірці, розрахунок інтегральних показників, формування покласових метрик, побудова матриць помилок і створення графічних засобів порівняння. Застосування такої схеми оцінювання забезпечує порівнюваність результатів і створює методично коректні умови для визначення моделей, найбільш придатних для розв'язання задач інтелектуального моніторингу трафіку комп'ютерної мережі в системах виявлення атак [97-104, 121-124].

З позиції проєктування програмного забезпечення підсистема оцінювання виконує роль завершального аналітичного модуля, що поєднує результати моделювання з візуальним і кількісним аналізом якості.

Реалізована підсистема оцінювання та візуалізації охоплює прогнозування на тестових підмножинах, розрахунок accuracy_score, формування classification_report, побудову confusion_matrix, ROC- та Precision–Recall-кривих для бінарних моделей, а також створення теплових карт і порівняльних стовпчикових діаграм для бінарних і багатокласових класифікаторів. У сукупності ці процедури забезпечують аналітичне завершення програмного конвеєра та формують інформаційну основу для подальшого експериментального дослідження ефективності розробленої інформаційної технології [76-81, 97-104].

3.13 Інтеграція інформаційної технології в систему виявлення атак

Розроблена інформаційна технологія інтелектуального моніторингу трафіку комп'ютерної мережі може бути використана як функціональний компонент системи виявлення атак. Її призначення полягає не в заміні всіх складових IDS, а в забезпеченні інтелектуальної обробки мережевих даних, формуванні ознакового простору, застосуванні моделей машинного навчання та отриманні класифікаційного рішення щодо стану мережевого трафіку. Це дає змогу розглядати запропоновану інформаційну технологію як аналітичне ядро, що може

бути інтегроване до наявної або проєктованої системи моніторингу мережевої безпеки.

У загальній структурі системи виявлення атак запропонована технологія розміщується між модулями збору мережевого трафіку та модулями подальшого аналізу, журналювання, візуалізації або реагування на інциденти. На вхід інформаційної технології можуть надходити підготовлені записи мережевих потоків, агреговані характеристики трафіку або дані, отримані від сенсорів мережевого моніторингу. На виході формується результат класифікації, який відображає належність трафіку до нормальної або атакувальної активності, а також, у разі багатокласового аналізу, уточнює тип виявленої атаки.

Місце інформаційної технології в узагальненій структурі системи виявлення атак наведено на рис. 3.2.

Рисунок 3.2 – Місце інформаційної технології інтелектуального моніторингу трафіку в структурі системи виявлення атак

У межах такої інтеграції система збору даних відповідає за отримання мережевого трафіку або його потокових характеристик, тоді як запропонована інформаційна технологія забезпечує подальшу обробку цих даних. Вона виконує

очищення та уніфікацію вхідних записів, формування ознакового простору, стандартизацію числових характеристик, зниження розмірності, формування вибірок, балансування навчальних даних, навчання моделей машинного навчання та оцінювання результатів класифікації. Отримані результати можуть передаватися до модулів журналювання, візуалізації, сповіщення або підтримки прийняття рішень.

Функціональну роль запропонованої інформаційної технології у складі системи виявлення атак наведено в табл. 3.13.

Таблиця 3.13 – Функціональна роль інформаційної технології у складі системи виявлення атак

Компонент системи виявлення атак	**Основна функція компонента**	**Роль запропонованої інформаційної технології**
Модуль збору мережевого трафіку	Отримання мережевих потоків або їх агрегованих характеристик	Використовує отримані дані як вхідну інформацію для подальшого аналізу
Модуль попередньої обробки	Очищення, уніфікація, обробка некоректних значень	Забезпечує підготовку даних до застосування моделей машинного навчання
Модуль формування ознакового простору	Побудова числового подання мережевого трафіку	Формує набір ознак, що характеризують поведінку мережевих потоків
Модуль інтелектуальної класифікації	Виявлення атакувальної активності та визначення її типу	Реалізує бінарне виявлення атак і багатокласове розпізнавання їх типів
Модуль оцінювання та візуалізації	Аналіз якості класифікації та подання результатів	Забезпечує розрахунок метрик, побудову матриць помилок і графічне подання результатів
Модуль журналювання та реагування	Фіксація подій безпеки, сповіщення або подальша перевірка інциденту	Отримує класифікаційний результат як основу для подальшого аналізу та прийняття рішень

Особливістю інтеграції запропонованої інформаційної технології є можливість використання двох режимів аналізу мережевого трафіку. Бінарний режим може застосовуватися для первинного виявлення підозрілої або атакувальної активності. Його результат дозволяє швидко визначити, чи містить досліджуваний трафік ознаки шкідливої поведінки. Багатокласовий режим забезпечує деталізацію результату шляхом розпізнавання конкретного типу атаки,

що підвищує інформативність класифікаційного рішення та може бути використано для подальшого реагування на інцидент.

Практична інтеграція інформаційної технології в систему виявлення атак потребує узгодження форматів вхідних даних, забезпечення коректного передавання ознак мережевих потоків, контролю якості класифікації та періодичного оновлення моделей машинного навчання. Оскільки характеристики мережевого трафіку та сценарії атак можуть змінюватися з часом, важливим є також передбачення можливості повторного навчання моделей на нових або розширених наборах даних. Це дозволяє підтримувати актуальність інтелектуального компонента IDS і зменшувати ризик зниження ефективності виявлення атак.

Результати роботи інформаційної технології доцільно використовувати як засіб підтримки прийняття рішень фахівцем з кібербезпеки. Класифікаційне рішення моделі може бути підставою для додаткової перевірки події, формування сповіщення, фіксації інциденту в журналі безпеки або запуску подальших процедур аналізу. Водночас результати машинного навчання не повинні розглядатися як єдиний автоматичний механізм реагування, оскільки практичне застосування IDS потребує врахування контексту мережевої інфраструктури, політик безпеки та можливих помилкових спрацювань.

Запропонована інформаційна технологія може бути інтегрована як у дослідницькі програмні комплекси для аналізу мережевого трафіку, так і в прикладні рішення, орієнтовані на підвищення ефективності систем виявлення атак. Її використання забезпечує формалізований і відтворюваний процес підготовки даних, побудови інформаційних моделей, навчання класифікаторів і оцінювання результатів [135-138].

Інтеграція розробленої інформаційної технології в систему виявлення атак сприяє підвищенню гарантоздатності її функціонування, оскільки забезпечує своєчасне виявлення атакувальної активності, підтримку прийняття рішень щодо реагування на інциденти та зменшення ризику пропуску небезпечних мережевих подій. У цьому контексті гарантоздатність розглядається як здатність системи

зберігати працездатність, стійкість і результативність виконання функцій моніторингу та виявлення атак за умов динамічної зміни характеристик мережевого трафіку і появи нових або модифікованих кіберзагроз.

Узагальнюючи наведене, інтеграція розробленої інформаційної технології в систему виявлення атак дозволяє поєднати засоби збору та моніторингу мережевого трафіку з інтелектуальними методами його класифікації. Реалізація такої функціональної організації забезпечує автоматизоване виявлення атакувальної активності, деталізацію типу загрози та формування інформаційної основи для подальшого аналізу інцидентів. Це підтверджує практичну спрямованість запропонованої інформаційної технології та її придатність для використання як інтелектуального компонента сучасних систем виявлення атак.

Висновки до розділу 3

У третьому розділі дисертаційної роботи розроблено програмну реалізацію інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак. На основі обґрунтованих у попередніх розділах теоретичних і методичних положень сформовано цілісний програмний модуль, у межах якого реалізовано повний цикл обробки мережевого трафіку – від завантаження та інтеграції даних до побудови моделей машинного навчання, оцінювання та візуалізації результатів класифікації.

У розділі обґрунтовано вибір засобів програмної реалізації та середовища розроблення, придатного для виконання повного циклу інтелектуального моніторингу трафіку комп’ютерної мережі. Показано, що використання мови Python і спеціалізованих бібліотек pandas, NumPy, scikit-learn, imbalanced-learn, Matplotlib, Seaborn та missingno забезпечує узгоджену реалізацію процедур обробки даних, навчання моделей машинного навчання та аналітичного подання результатів у межах єдиного програмного середовища.

У результаті виконаного проєктування визначено структуру програмного модуля та послідовність його функціонування, яка охоплює завантаження окремих

частин набору даних, їх інтеграцію в датафрейм data, первинний аналіз, попередню обробку, формування ознакового простору, побудову бінарної та багатокласової вибірок, балансування, навчання моделей машинного навчання, а також оцінювання й візуалізацію результатів. Це забезпечило практичну реалізацію розробленої інформаційної технології у вигляді цілісного програмного конвеєра.

У розділі реалізовано початкові етапи обробки мережевого трафіку, зокрема завантаження окремих частин набору даних в об'єкти data1–data8, їх об'єднання в інтегрований датафрейм data, уніфікацію назв стовпців, видалення дубльованих записів, виявлення та обробку пропущених і нескінченних значень, а також первинний статистичний і візуальний аналіз структури даних. На цій основі сформовано підготовлене подання мережевого трафіку, придатне для побудови ознакового простору.

Важливим результатом розділу є програмна реалізація формування ознакового простору на основі матриці features, вектора міток attacks, стандартизації через StandardScaler та зниження розмірності за допомогою IncrementalPCA. Результатом цих процедур стало формування датафрейму new_data, який містить головні компоненти PC1…PCn та стовпець Attack Type і використовується як базове подання даних для подальшого розв'язання задач бінарного виявлення атак і багатокласового розпізнавання їх типів.

У межах програмного модуля реалізовано два режими функціонування інформаційної технології. Для бінарної постановки задачі сформовано об'єкти normal_traffic, intrusions, ids_data, bc_data, а також побудовано матрицю ознак X_bc і вектор цільових міток y_bc. Для багатокласової постановки задачі виконано аналіз class_counts, відбір репрезентативних класів, формування робочого датафрейму df, застосування синтетичного балансування методом SMOTE, а також побудову збалансованого датафрейму blnc_data і похідних від нього об'єктів features та labels. Це забезпечило програмну підтримку як режиму первинного виявлення атак, так і режиму розпізнавання їх конкретних типів.

У розділі інтегровано моделі машинного навчання, що становлять інтелектуальне ядро розробленої технології. Для бінарного виявлення атак

реалізовано моделі lr1, lr2, svm1, svm2, а для багатокласового аналізу – rf1, rf2, dt1, dt2, knn1, knn2. Для всіх моделей забезпечено навчання на відповідних вибірках і перехресну перевірку, що створило уніфіковану основу для подальшого порівняння їх ефективності. Завершальним результатом розділу стала реалізація процедур оцінювання та візуалізації, що охоплюють прогнозування, розрахунок accuracy_score, формування classification_report, побудову confusion_matrix, ROC- та Precision–Recall-кривих для бінарних моделей, а також теплових карт і порівняльних діаграм для бінарних і багатокласових класифікаторів.

РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ ТРАФІКУ КОМП'ЮТЕРНОЇ МЕРЕЖІ ДЛЯ СИСТЕМ ВИЯВЛЕННЯ АТАК

4.1 Організація експериментального дослідження та критерії оцінювання ефективності

Експериментальне дослідження у дисертаційній роботі спрямоване на перевірку ефективності розробленої інформаційної технології інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак в умовах обробки реальних ознакових описів мережевих потоків. Його основна мета полягає у встановленні того, наскільки запропонована послідовність процедур попередньої обробки даних, формування ознакового простору, побудови вибірок, навчання моделей машинного навчання та оцінювання результатів забезпечує надійне виявлення атак і розпізнавання їх типів. Організація дослідження спирається на методичні положення, сформульовані у попередніх розділах, а також на фактичну логіку програмної реалізації, подану в третьому розділі [140, 150].

Базою експериментального дослідження є набір даних CIC-IDS2017, який містить описи мережевих потоків нормального та атакувального трафіку й дозволяє досліджувати задачу виявлення атак як у бінарній, так і в багатокласовій постановках. Використання саме цього набору даних є доцільним, оскільки він відображає різні сценарії мережевої активності, містить набір кількісних характеристик потоків і дає змогу формувати відтворювані експерименти для оцінювання інтелектуальних методів аналізу трафіку комп'ютерної мережі. У межах роботи обидві постановки задачі розглядаються як взаємодоповнювальні: бінарна класифікація використовується для первинного відокремлення атакувального трафіку від нормального, а багатокласова – для деталізації виявленої загрози за її типом.

Організація експерименту побудована окремо для двох режимів функціонування інформаційної технології. У першому режимі досліджується

ефективність бінарного виявлення атак, для чого формується вибірка, у якій нормальний трафік кодується значенням 0, а будь-який атакувальний трафік – значенням 1. У другому режимі досліджується багатокласове розпізнавання типів атак, де об’єкти мережевого трафіку зберігають належність до конкретних класів атакувальної активності. Така схема організації експериментального дослідження (рис. 4.1) дає змогу оцінити ефективність розробленої інформаційної технології як на рівні загального виявлення загрози, так і на рівні її деталізованої класифікації, що відповідає практичним потребам систем виявлення атак.

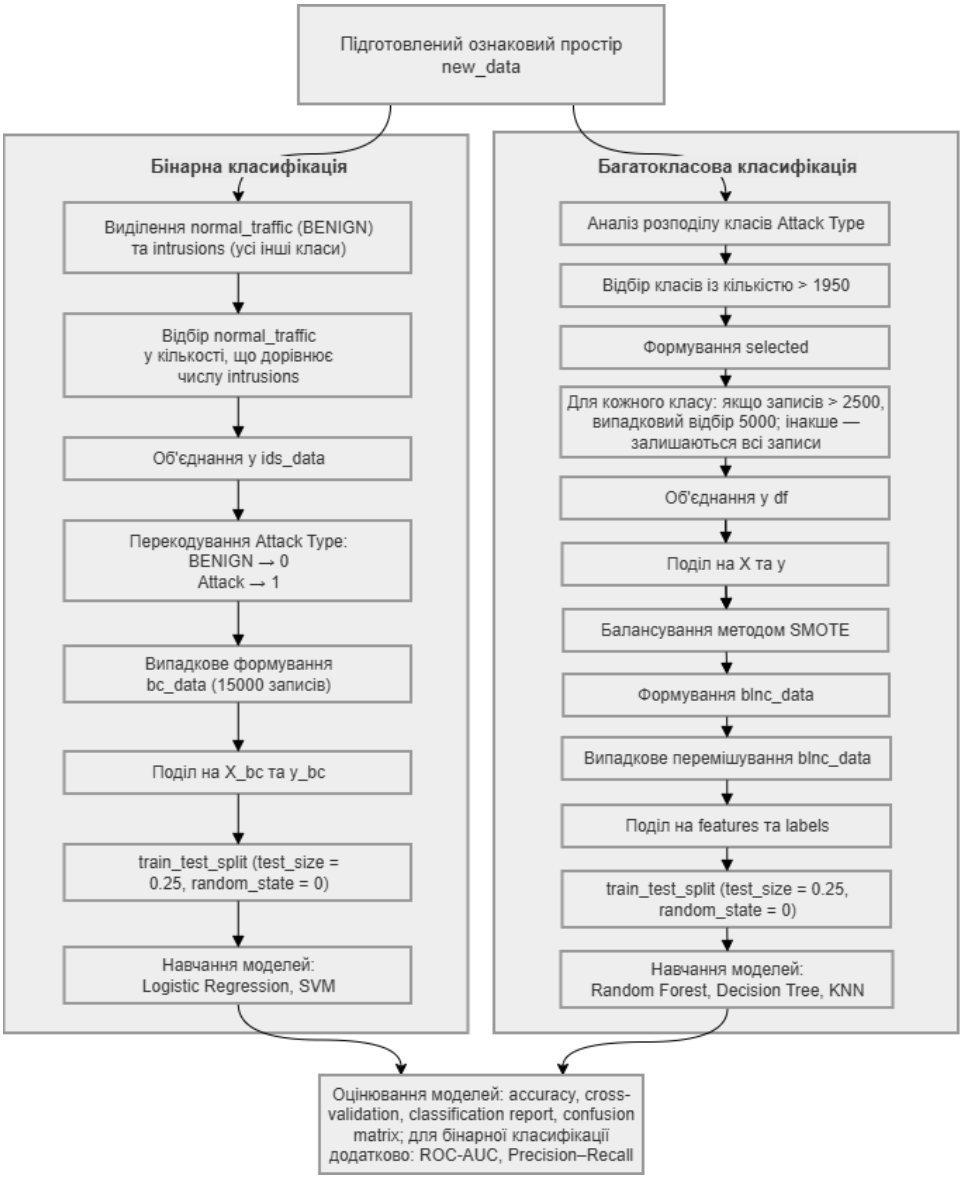


Рисунок 4.1 – Схема формування вибірок для навчання та тестування моделей машинного навчання

Усі експерименти виконуються на попередньо сформованому ознаковому просторі, який є результатом очищення даних, усунення неваріантних ознак,

стандартизації числових характеристик та зниження розмірності методом головних компонент. Саме це подання використовується як єдина інформаційна основа для подальшої побудови бінарної та багатокласової вибірок. Застосування єдиного простору ознак забезпечує узгодженість умов експериментального дослідження для різних моделей і дає змогу зіставляти результати класифікації за спільною системою вхідних характеристик. Детальні результати формування цього простору подано в наступному підрозділі.

Для бінарної постановки у програмному модулі формується збалансована вибірка на основі датафрейму new_data: з нього окремо виділяються записи нормального трафіку та записи атак, після чого кількість спостережень нормального класу вирівнюється відносно кількості атакувальних прикладів. Сформований набір перетворюється у двокласову вибірку bc_data, з якої додатково відбирається 15000 спостережень для проведення порівняльного експерименту. Далі виконується поділ на навчальну та тестову підмножини у співвідношенні 75% до 25% із фіксованим значенням random_state = 0, що забезпечує відтворюваність обчислювального експерименту та сталість умов порівняння моделей.

Для багатокласової постановки на основі того самого ознакового простору формується окрема робоча вибірка, у якій зберігається поділ трафіку за типами атак. На цьому етапі до складу аналізованих класів включаються ті категорії, які мають достатню кількість спостережень для навчання моделей, після чого виконується балансування вибірки методом SMOTE. Отримана збалансована структура даних використовується для формування матриці ознак та вектора цільових міток, які надалі поділяються на навчальну і тестову підмножини за аналогічною схемою з параметрами test_size = 0.25 та random_state = 0. Це забезпечує узгодженість умов навчання й тестування та створює основу для коректного порівняння ефективності багатокласових класифікаторів.

У межах експериментального дослідження для кожної постановки задачі використовується не одна модель, а група моделей машинного навчання, що дає змогу оцінити ефективність різних моделей аналізу мережевого трафіку. Для бінарного виявлення атак досліджуються дві конфігурації логістичної регресії та

дві конфігурації методу опорних векторів. Для багатокласового розпізнавання типів атак використовуються дві конфігурації випадкового лісу, дві конфігурації дерева рішень і дві конфігурації методу k найближчих сусідів. Порівняння кількох моделей у межах однакових умов експерименту дає змогу встановити, які з них є більш придатними для реалізації функцій інтелектуального моніторингу трафіку комп'ютерної мережі. Водночас розроблена інформаційна технологія має модульну структуру, що забезпечує її гнучкість і розширюваність. Це дозволяє за потреби додавати нові функціональні модулі в обох режимах роботи – у режимі бінарного виявлення атак і в режимі багатокласового розпізнавання типів атак – з використанням інших моделей машинного навчання, алгоритмів обробки даних та засобів оцінювання результатів. Модульна структура інформаційної технології забезпечує її адаптивність до змін умов функціонування мережі, появи нових типів атак і подальшого вдосконалення методів інтелектуального аналізу даних.

Для оцінювання стабільності результатів навчання в роботі застосовується п'ятикратна перехресна перевірка на навчальних підмножинах. У програмному коді вона реалізована за допомогою функції cross_val_score(..., cv = 5) для всіх досліджуваних моделей. Використання 5-fold cross-validation дає змогу оцінити не лише якість моделі на одному фіксованому розбитті, а й узагальнювальну здатність класифікатора на кількох варіантах навчально-валідаційного поділу. Середнє значення перехресної перевірки використовується як додатковий критерій стійкості моделі й надалі враховується під час підсумкового порівняння алгоритмів.

Система критеріїв оцінювання ефективності сформована з урахуванням специфіки задач виявлення атак у мережевому трафіку. Базовою інтегральною метрикою виступає точність класифікації accuracy_score, яка дозволяє оцінити загальну частку правильно класифікованих спостережень. Водночас для задач кібербезпеки однієї лише загальної точності недостатньо, оскільки вона може приховувати помилки розпізнавання окремих класів, особливо в умовах дисбалансу. Саме тому для детальнішого аналізу використовуються також покласові показники precision, recall та F1-score, які формуються через

classification_report. Показник precision характеризує частку дійсно коректних спрацювань серед усіх позитивних рішень моделі, recall відображає здатність виявляти всі наявні об’єкти певного класу, а F1-score узагальнює ці дві характеристики в єдину збалансовану оцінку.

Для аналізу структури класифікаційних помилок у роботі використовуються матриці помилок confusion_matrix. Їх застосування є важливим як для бінарної, так і для багатокласової постановок, оскільки вони дозволяють встановити не лише кількість правильних та помилкових рішень, а й характер цих помилок. У бінарній задачі матриця помилок дає змогу окремо проаналізувати хибнопозитивні та хибнонегативні рішення, що має принципове значення для систем виявлення атак. У багатокласовій задачі вона дозволяє виявити, між якими саме типами атак виникає найбільша плутанина під час класифікації.

Для бінарного режиму оцінювання додатково застосовуються ROC-криві, площа під ROC-кривою (AUC), а також Precision–Recall-криві. Використання цих інструментів є доцільним у задачах виявлення атак, оскільки вони дозволяють аналізувати поведінку моделей при різних порогах прийняття рішення та краще оцінювати здатність класифікатора розрізняти нормальний і атакувальний трафік. ROC-AUC використовується для загальної оцінки роздільної здатності моделі, тоді як Precision–Recall-подання є особливо інформативним для задач, у яких важливо контролювати компроміс між повнотою виявлення атак і точністю спрацювань.

Організація експериментального дослідження базується на окремому розгляді бінарної та багатокласової постановок задачі, використанні єдиного попередньо сформованого ознакового простору, фіксованому поділі вибірок на навчальну і тестову частини, застосуванні 5-fold cross-validation та системи взаємодоповнювальних метрик оцінювання. У межах кожної постановки досліджується група моделей машинного навчання, що дає змогу здійснити коректне порівняння різних моделей аналізу мережевого трафіку в однакових умовах експерименту. Для бінарного виявлення атак використано дві конфігурації логістичної регресії та дві конфігурації методу опорних векторів, а для багатокласового розпізнавання типів атак – дві конфігурації випадкового лісу, дві

конфігурації дерева рішень і дві конфігурації методу k найближчих сусідів. При цьому інформаційна технологія реалізована за модульним принципом, що дозволяє додавати нові модулі з іншими моделями машинного навчання в обох режимах роботи. Запропонована організація експериментального дослідження формує методично узгоджене підґрунтя для подальшого аналізу результатів і забезпечує логічний перехід до розгляду результатів підготовки даних та формування ознакового простору в наступному підрозділі.

4.2 Результати формування ознакового простору та підготовки даних до класифікації

Важливим етапом експериментального дослідження ефективності розробленої інформаційної технології є формування підготовленого ознакового простору, придатного для подальшого навчання моделей машинного навчання. У межах програмної реалізації цей етап охоплює очищення даних мережевого трафіку, уніфікацію їх структури, усунення технічних дефектів, оптимізацію подання числових ознак, вилучення малоінформативних характеристик, стандартизацію та зниження розмірності (рис. 4.2). Сукупність зазначених процедур визначає якість вхідного представлення мережевого трафіку і, відповідно, впливає на точність подальшого виявлення атак та розпізнавання їх типів [141, 145].

Початковий ознаковий опис мережевого трафіку формується шляхом інтеграції восьми частин набору CIC-IDS2017 у єдиний датафрейм data. Після цього виконується уніфікація назв стовпців шляхом видалення початкових і кінцевих пробілів у назвах ознак. Така операція має практичне значення, оскільки усуває потенційні помилки доступу до атрибутів і забезпечує коректність подальших перетворень. Одночасно здійснюється первинний структурний аналіз датафрейму, перевіряються типи даних, наявність пропусків і базові статистичні характеристики, що дає змогу оцінити загальний стан вхідного масиву мережевого трафіку перед його поглибленою підготовкою.

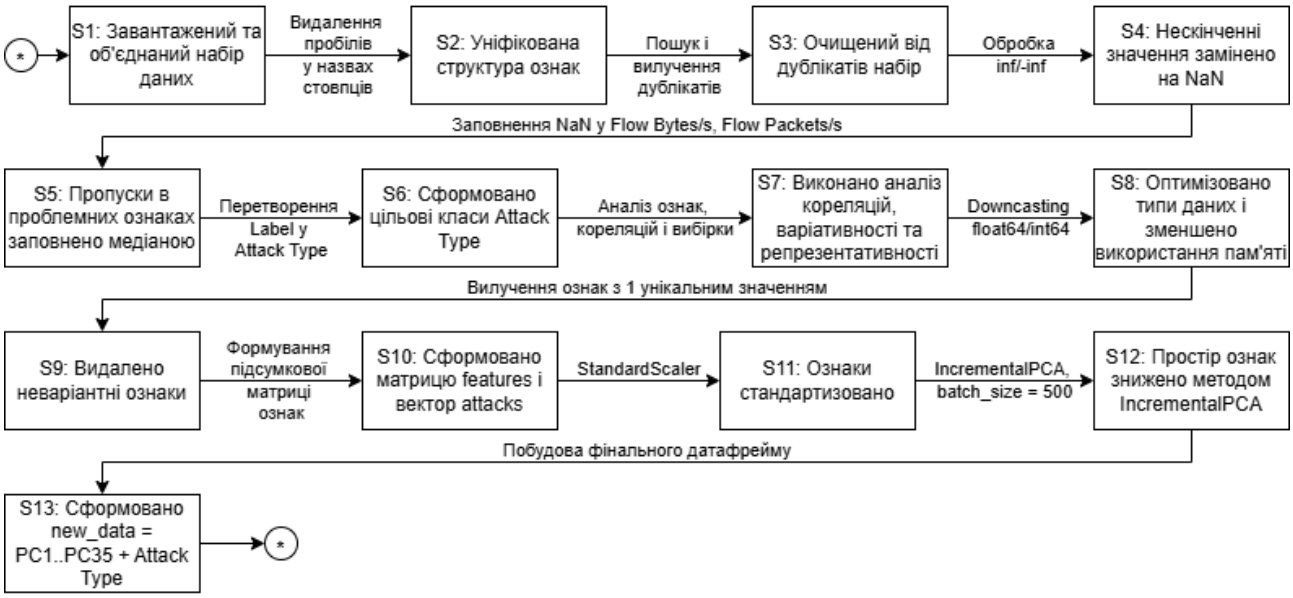


Рисунок 4.2 – Діаграма переходів у процесі формування ознакового простору та підготовки даних до класифікації

Наступний етап пов’язаний з очищенням даних від технічних дефектів. Спочатку виявляються та вилучаються дубльовані записи. Далі всі числові ознаки перевіряються на наявність нескінченних значень, які замінюються на NaN, тобто переводяться у формат пропущених значень, придатний для подальшої обробки. Це забезпечує приведення набору даних до узгодженого числового стану та запобігає проблемам, які могли б порушити виконання процедур стандартизації, зниження розмірності та навчання класифікаторів. Додатково візуалізація пропусків за допомогою бібліотеки missingno показує, що проблемні значення мають локальний характер і зосереджені в окремих характеристиках потоку.

За результатами аналізу пропущених значень окремо обробляються характеристики Flow Bytes/s та Flow Packets/s. Після заміни нескінченних значень на NaN пропуски в цих ознаках заповнюються медіанними значеннями. Використання саме медіани є доцільним, оскільки зазначені показники мають асиметричний розподіл і містять викиди, що підтверджується гістограмами та boxplot-діаграмами. У підсумку формується очищене подання числових

характеристик мережевого трафіку, в якому пропуски в найбільш проблемних ознаках усунено без суттєвого викривлення розподілу даних.

Окреме значення має формування цільового подання класів атак. Вихідна мітка Label перетворюється на узагальнену ознаку Attack Type, у межах якої окремі підтипи атак об'єднуються у ширші категорії, зокрема DoS, Brute Force, Web Attack, Port Scan, Bot, Infiltration, Heartbleed і DDoS, тоді як нормальний трафік зберігається як клас BENIGN. Після цього початкова мітка Label вилучається, а для допоміжного аналізу додатково формується числове кодування Attack Number. Таким чином, сирі мітки набору даних переводяться в узгоджену систему цільових класів, придатну для подальшої бінарної та багатокласової класифікації.

У ході аналізу структури ознак досліджуються кореляційні зв'язки, неваріантні характеристики та репрезентативність вибірки. Для цього обчислюється кореляційна матриця числових ознак, формується перелік характеристик із позитивною кореляцією відносно допоміжної змінної Attack Number, а також перевіряються стовпці з нульовим стандартним відхиленням. Це дає змогу виявити ознаки, що не містять корисної варіативної інформації, і підтверджує наявність надлишковості у початковому описі мережевого трафіку. Додатковий аналіз 20-відсоткової вибірки використовується для візуальної діагностики розподілів і викидів, а також для оцінювання того, наскільки вибіркове подання відображає статистичні властивості повного набору даних.

Практично важливим кроком є оптимізація збереження даних у пам'яті. Для числових стовпців реалізується процедура downcasting: значення типу float64 перетворюються на float32, а int64 – на int32, якщо це допускається діапазоном значень. Після такої оптимізації обчислюються новий обсяг використання пам'яті та відсоток його зменшення. Хоча ця операція безпосередньо не змінює математичну структуру ознакового простору, вона має прикладне значення для побудови інформаційної технології, оскільки знижує ресурсомісткість обробки великого масиву мережевого трафіку та підвищує стабільність подальших обчислювальних процедур.

Після очищення та оптимізації даних виконується вилучення неваріантних ознак. Для цього визначається кількість унікальних значень у кожному стовпці, після чого всі характеристики, що мають лише одне унікальне значення, виключаються з подальшого аналізу. Така процедура дає змогу усунути ознаки, які не можуть сприяти розрізненню класів, оскільки не містять варіативності між спостереженнями. У результаті формується оновлена матриця ознак, позбавлена явно неінформативних атрибутів, що створює більш раціональну основу для наступного етапу стандартизації та PCA [90, 91, 97-104, 121-124].

Безпосереднє формування нового ознакового простору починається зі створення матриці features, до якої входять усі підготовлені числові характеристики, та вектора цільових міток attacks, що містить значення Attack Type. Далі до матриці ознак застосовується стандартизація за допомогою StandardScaler, унаслідок чого всі змінні переводяться до узгодженого масштабу. Це є необхідною передумовою коректного застосування методу головних компонент, оскільки без стандартизації ознаки з більшими числовими масштабами могли б домінувати у формуванні компонент і спотворювати структуру нового простору.

Наступним кроком є зниження розмірності простору ознак методом IncrementalPCA. Кількість головних компонент визначається як половина кількості вхідних ознак, а навчання перетворення здійснюється пакетно з batch_size = 500, що робить процедуру придатною для роботи з великим обсягом даних. За результатами дослідження після застосування PCA було сформовано 35 головних компонент, які сумарно зберігають близько 99,30 % дисперсії вихідного простору ознак. Це свідчить про те, що значна частина початкових характеристик є надлишковою або корельованою, а перехід до компактнішого представлення дає змогу істотно скоротити розмірність без критичної втрати інформативності.

Підсумком застосування IncrementalPCA є побудова нового датафрейму new_data, який містить головні компоненти PC1, PC2, …, PC35 та стовпець Attack Type. Саме цей об’єкт виступає підсумковим підготовленим ознаковим простором, що використовується в подальших експериментах як основа для формування

бінарної та багатокласової вибірок. У такий спосіб вихідний високорозмірний і частково надлишковий опис мережевого трафіку перетворюється на компактне, стандартизоване та обчислювально зручне подання, придатне для реалізації функцій інтелектуального моніторингу трафіку комп'ютерної мережі.

Відтак, результати підготовки даних і формування ознакового простору підтверджують, що в межах розробленої інформаційної технології реалізовано повний цикл перетворення сирих описів мережевого трафіку у форму, придатну для машинного навчання. На цьому етапі усунуто дублікати, оброблено пропущені та нескінченні значення, оптимізовано структуру числових даних, вилучено неінформативні ознаки, виконано стандартизацію та побудовано компактний простір головних компонент. Отриманий датафрейм new_data є безпосередньою основою для подальшого формування вибірок і експериментального дослідження моделей машинного навчання. У наступному підрозділі на цій основі розглядаються результати у задачі бінарного виявлення атак.

4.3 Експериментальне дослідження ефективності бінарного виявлення атак

Одним із ключових напрямів перевірки ефективності розробленої інформаційної технології є дослідження її здатності здійснювати первинне виявлення атак у трафіку комп'ютерної мережі в бінарній постановці. У межах цієї постановки кожен мережевий потік відноситься або до нормального трафіку, або до трафіку, що містить ознаки атакувальної активності. Такий режим є базовим для практичного застосування в системах виявлення атак, оскільки забезпечує оперативне відокремлення безпечних мережевих взаємодій від потенційно небезпечних.

Для проведення експериментального дослідження на основі підготовленого ознакового простору new_data сформовано вибірку для бінарної класифікації. Із загального масиву даних окремо виділено записи нормального трафіку з міткою BENIGN та записи, що відповідають усім іншим типам атак. Далі з підмножини

нормального трафіку випадковим чином відібрано таку саму кількість спостережень, як і в підмножині атакувального трафіку. Після об’єднання цих даних сформовано двокласову вибірку, у якій значення 0 відповідає нормальному трафіку, а значення 1 – атакувальному.

Для подальшого аналізу з отриманого масиву додатково сформовано вибірку bc_data обсягом 15000 спостережень. Аналіз розподілу класів у цій вибірці показав, що вона є близькою до збалансованої: кількість об’єктів класу атак становить 7565 записів, або 50,43 %, тоді як кількість об’єктів нормального трафіку – 7435 записів, або 49,57 %. Такий розподіл створює коректні умови для навчання моделей бінарної класифікації без суттєвого домінування одного з класів.

Після формування вибірки bc_data виконано її поділ на матрицю ознак X_bc і вектор цільових міток y_bc. Надалі за допомогою функції train_test_split здійснено розбиття даних на навчальну та тестову підмножини у співвідношенні 75 % до 25 % при random_state = 0. У результаті для навчання моделей сформовано навчальну підмножину обсягом 11250 спостережень, а для підсумкового оцінювання якості класифікації – тестову підмножину обсягом 3750 спостережень.

4.3.1 Дослідження моделей логістичної регресії

Для дослідження ефективності логістичної регресії у задачі бінарного виявлення атак було використано дві конфігурації моделі. Перша конфігурація (lr1) реалізована з параметрами max_iter = 10000, C = 0.1, solver = 'saga', друга конфігурація (lr2) – з параметрами max_iter = 15000, C = 100, solver = 'sag'. Зіставлення цих конфігурацій дає змогу оцінити вплив параметра регуляризації та алгоритму оптимізації на якість розмежування нормального й атакувального трафіку у сформованому просторі головних компонент.

На першому етапі оцінювання для обох конфігурацій було виконано п’ятикратну перехресну перевірку. Для моделі lr1 значення 5-fold cross-validation становили 0,9280; 0,9160; 0,9138; 0,9151; 0,9209, а середнє значення – 0,9188. Для моделі lr2 відповідні значення склали 0,9289; 0,9200; 0,9147; 0,9196; 0,9222, а

середній результат – 0,9211. Як видно з рисунка 4.8, обидві моделі демонструють високий рівень узагальнювальної здатності, однак модель lr2 має вищий середній результат перехресної перевірки, що свідчить про її дещо кращу стійкість на навчальній вибірці.

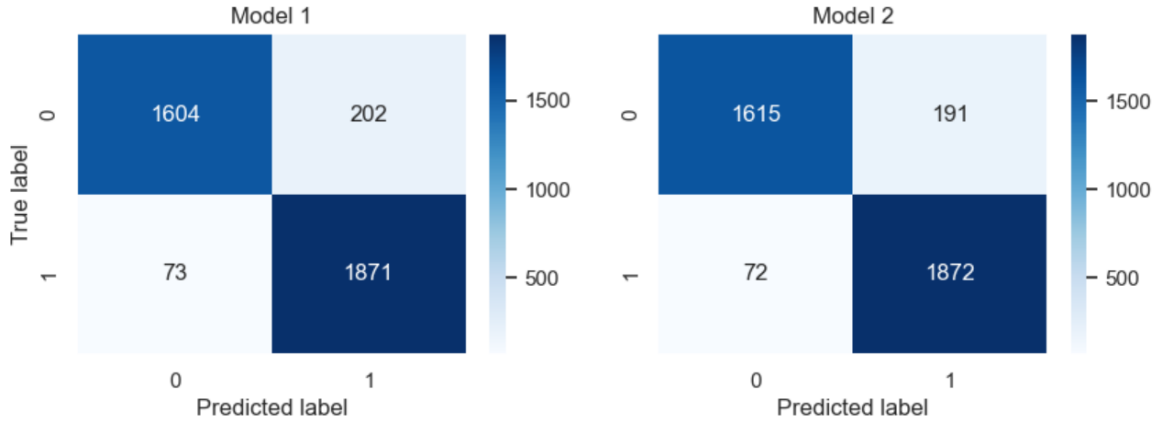


Рисунок 4.3 – Матриці помилок моделей логістичної регресії для бінарного виявлення атак

Після навчання моделей було сформовано прогнози на тестовій підмножині та побудовано матриці помилок, наведені на рисунку 4.3. Для моделі lr1 отримано 1604 правильно класифікованих об’єкти нормального трафіку, 202 хибнопозитивних спрацювання, 73 хибнонегативні рішення та 1871 правильно виявлену атаку. Для моделі lr2 відповідні значення становлять 1615, 191, 72 та 1872. Отже, обидві конфігурації забезпечують високий рівень правильних класифікацій, однак друга модель характеризується меншою кількістю як хибнопозитивних, так і хибнонегативних рішень, що є важливим для практичного застосування в системах виявлення атак.

Аналіз матриць помилок дає змогу зробити висновок щодо структури класифікаційних помилок. Для систем виявлення атак особливо небезпечними є хибнонегативні рішення, за яких атакувальний трафік класифікується як нормальний. У цьому дослідженні обидві моделі демонструють низьку кількість таких помилок: 73 для lr1 і 72 для lr2. Водночас модель lr2 загалом краще врівноважує співвідношення між помилками двох типів, що свідчить про більш стабільну поведінку під час класифікації трафіку.

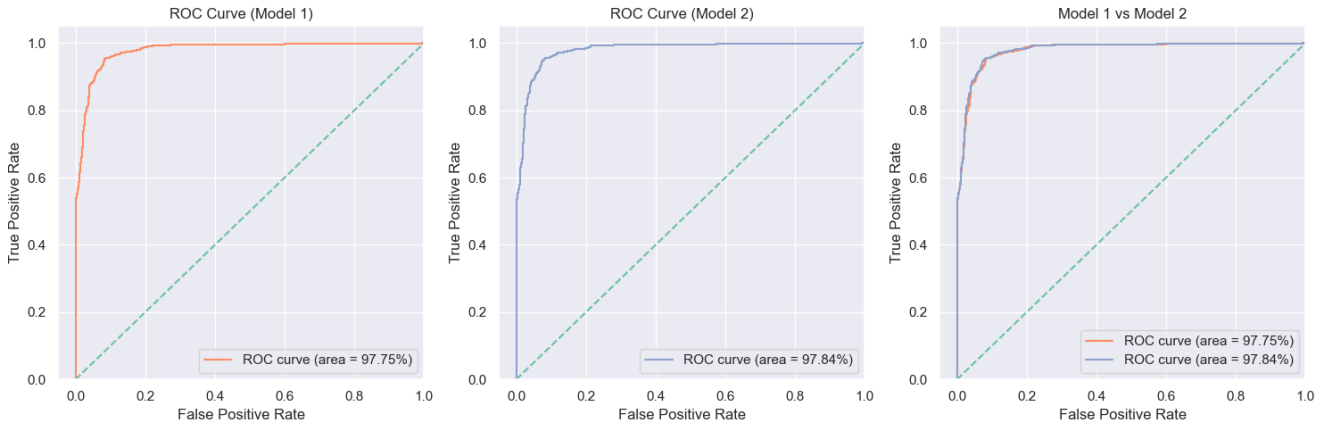


Рисунок 4.4 – ROC-криві моделей логістичної регресії

Для узагальненого оцінювання здатності моделей розрізняти два класи було побудовано ROC-криві, наведені на рисунку 4.4. Площа під ROC-кривою для моделі lr1 становить 97,75 %, а для моделі lr2 – 97,84 %. Обидві криві розташовані значно вище діагоналі випадкового класифікатора та проходять поблизу верхнього лівого кута координатної площини, що свідчить про високу якість бінарного розмежування нормального й атакувального трафіку. При цьому модель lr2 має незначну перевагу над lr1, що підтверджується як величиною ROC-AUC, так і формою ROC-кривої при порівнянні двох моделей на одному графіку.

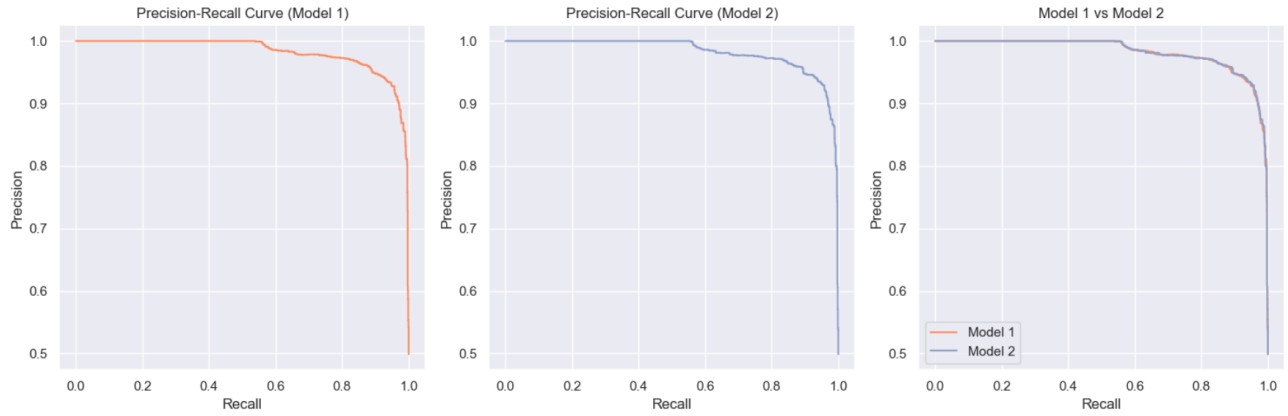


Рисунок 4.5 – Precision–Recall-криві моделей логістичної регресії

Додатковий аналіз Precision–Recall-кривих, поданих на рисунку 4.5, також підтвердив високу якість обох конфігурацій логістичної регресії. Криві для lr1 і lr2 проходять у верхній частині площини «повнота – точність», що означає збереження високої точності спрацювання навіть за великих значень recall. Для задачі виявлення атак це має принципове значення, оскільки дозволяє оцінити здатність

моделі виявляти шкідливу активність без різкого зростання кількості хибних спрацювань. Порівняльний графік на рисунку 4.5 показує, що друга модель дещо краще зберігає баланс між precision та recall на більшості ділянок кривої.

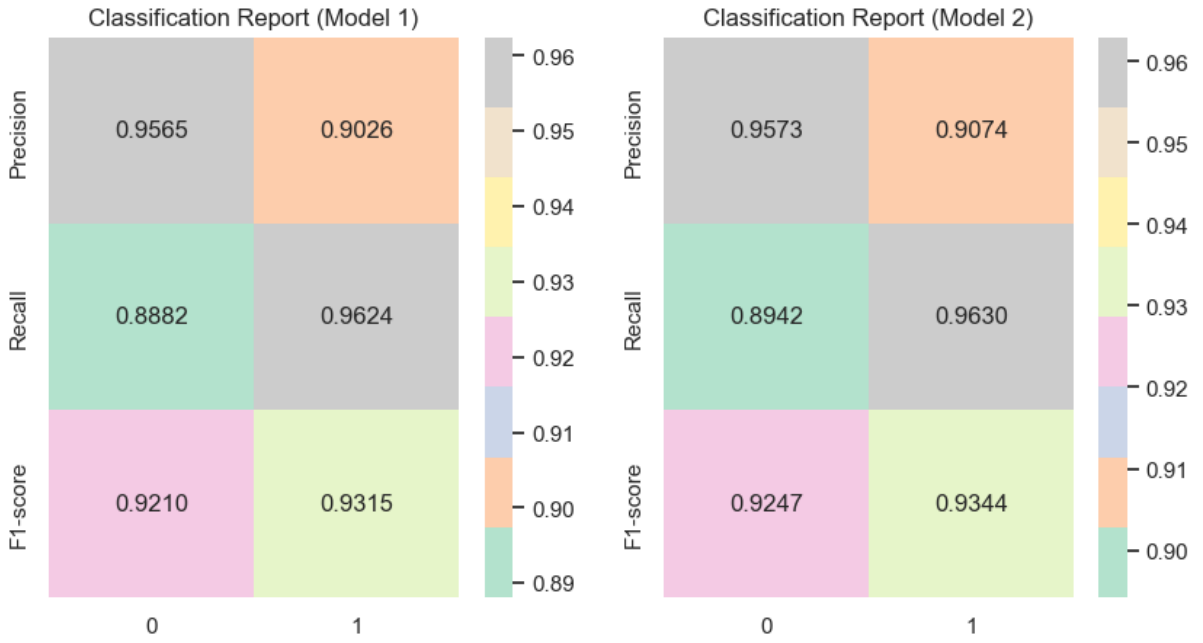


Рисунок 4.6 – Покласові метрики моделей логістичної регресії

Покласовий аналіз результатів класифікації, поданий у вигляді теплових карт на рисунку 4.6, показав, що для моделі lr1 значення precision, recall та F1-score для класу нормального трафіку становлять відповідно 0,9565; 0,8882; 0,9210, а для класу атак – 0,9026; 0,9624; 0,9315. Для моделі lr2 аналогічні показники для класу нормального трафіку дорівнюють 0,9573; 0,8942; 0,9247, а для класу атак – 0,9074; 0,9630; 0,9344. Ці результати свідчать, що друга конфігурація моделі забезпечує дещо вищу якість класифікації для обох класів, зокрема за показниками F1-score і recall, що є важливим при побудові систем виявлення атак.

Інтегральні результати порівняння двох моделей наведено на рисунках 4.7 та 4.8. Значення accuracy для lr1 становить 0,9267, тоді як для lr2 – 0,9299. Аналогічно, середнє значення перехресної перевірки є вищим у моделі lr2 – 0,9211 проти 0,9188 у lr1. Хоча різниця між моделями є невеликою, вона є сталою для основних критеріїв оцінювання: точності класифікації, середнього результату cross-validation, ROC-AUC та покласових метрик.

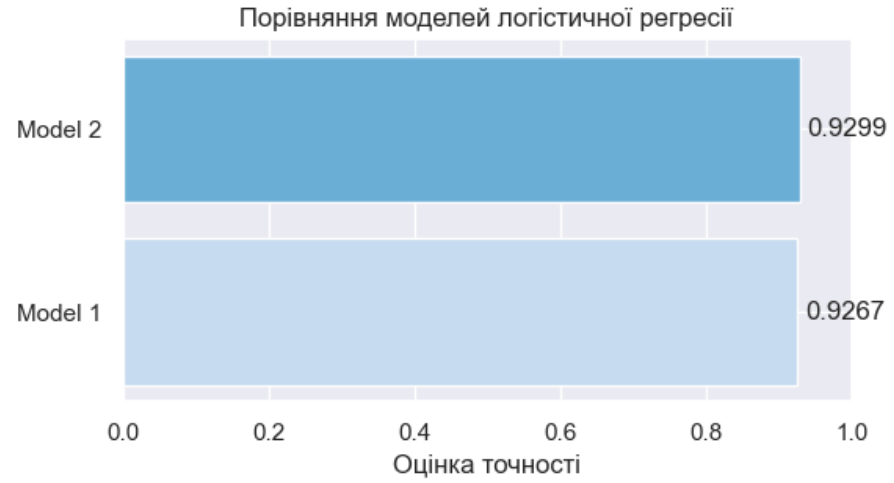


Рисунок 4.7 – Порівняння моделей логістичної регресії за показником accuracy

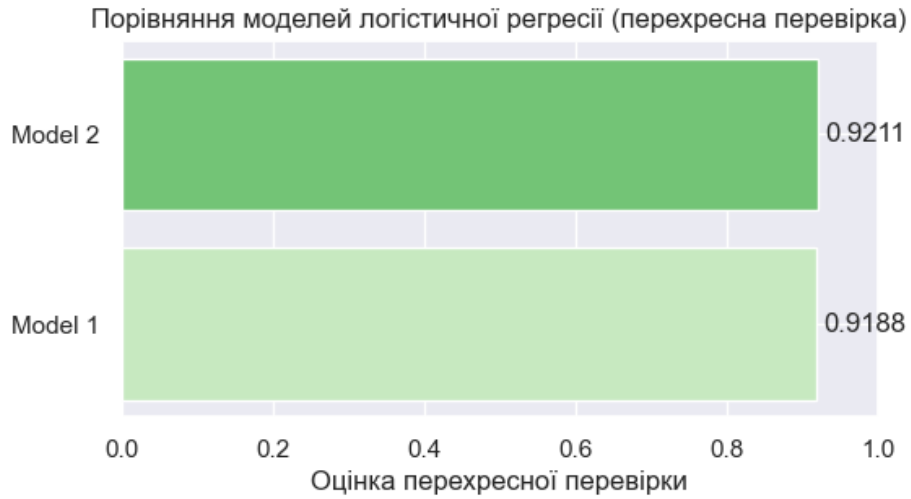


Рисунок 4.8 – Порівняння моделей логістичної регресії за результатами перехресної перевірки

Результати, наведені на рисунках 4.3–4.8, узгоджено свідчать про те, що обидві конфігурації логістичної регресії є ефективними для задачі бінарного виявлення атак у трафіку комп’ютерної мережі. Водночас модель lr2 демонструє кращі значення більшості показників оцінювання та більш збалансовану структуру класифікаційних помилок. Це дає підстави розглядати її як кращий варіант серед досліджених конфігурацій логістичної регресії та використовувати надалі як основного представника цього класу моделей у порівнянні з методом опорних векторів.

4.3.2 Дослідження моделей методу опорних векторів

Для дослідження ефективності методу опорних векторів у задачі бінарного виявлення атак було використано дві конфігурації моделі. Перша конфігурація (svm1) реалізована з параметрами kernel = 'poly', C = 1, random_state = 0, probability = True, друга конфігурація (svm2) – з параметрами kernel = 'rbf', C = 1, gamma = 0.1, random_state = 0, probability = True. Така постановка експерименту дозволяє оцінити вплив типу ядра та налаштувань моделі на якість бінарного розмежування нормального й атакувального трафіку у сформованому просторі головних компонент.

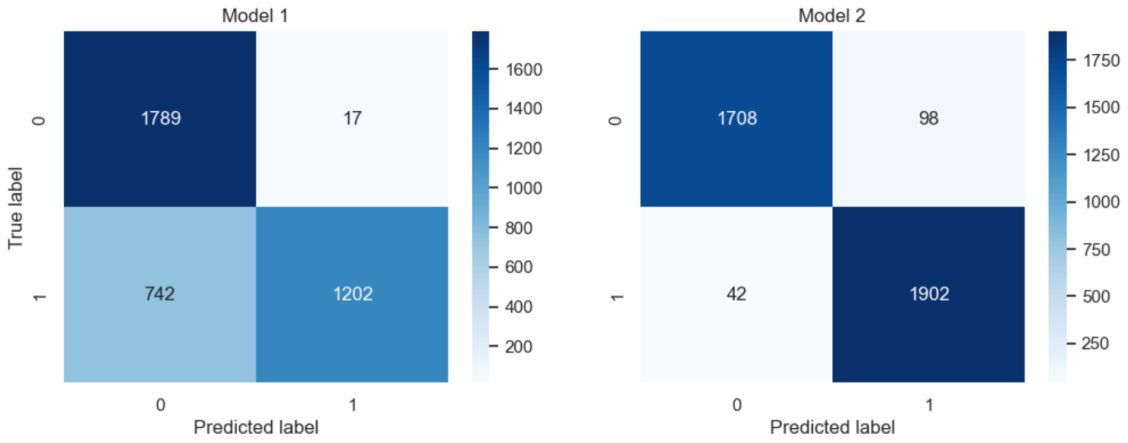


Рисунок 4.9 – Матриці помилок моделей методу опорних векторів для бінарного виявлення атак

Після навчання моделей на тренувальній підмножині було сформовано прогнози на тестових даних та побудовано матриці помилок, наведені на рисунку 4.9. Для моделі svm1 отримано 1789 правильно класифікованих об’єктів нормального трафіку, 17 хибнопозитивних спрацювань, 742 хибнонегативні рішення та 1202 правильно виявлені атаки. Для моделі svm2 відповідні значення становлять 1708, 98, 42 та 1902. Отже, обидві конфігурації методу опорних векторів дозволяють розмежовувати класи, однак модель svm2 характеризується значно меншою кількістю хибнонегативних рішень, що є особливо важливим для систем

виявлення атак, оскільки саме такі помилки означають пропуск атакувальної активності.

Аналіз матриць помилок показує, що модель svm1 добре розпізнає нормальний трафік, оскільки має мінімальну кількість хибнопозитивних спрацювань, однак суттєво гірше виявляє атакувальний трафік через велику кількість хибнонегативних рішень. Натомість модель svm2 демонструє значно вищу здатність до виявлення атак, хоча це супроводжується певним збільшенням кількості хибнопозитивних спрацювань. З практичної точки зору для задач кібербезпеки така поведінка є більш прийнятною, оскільки пропуск атаки є значно небезпечнішим, ніж додаткове помилкове спрацювання.

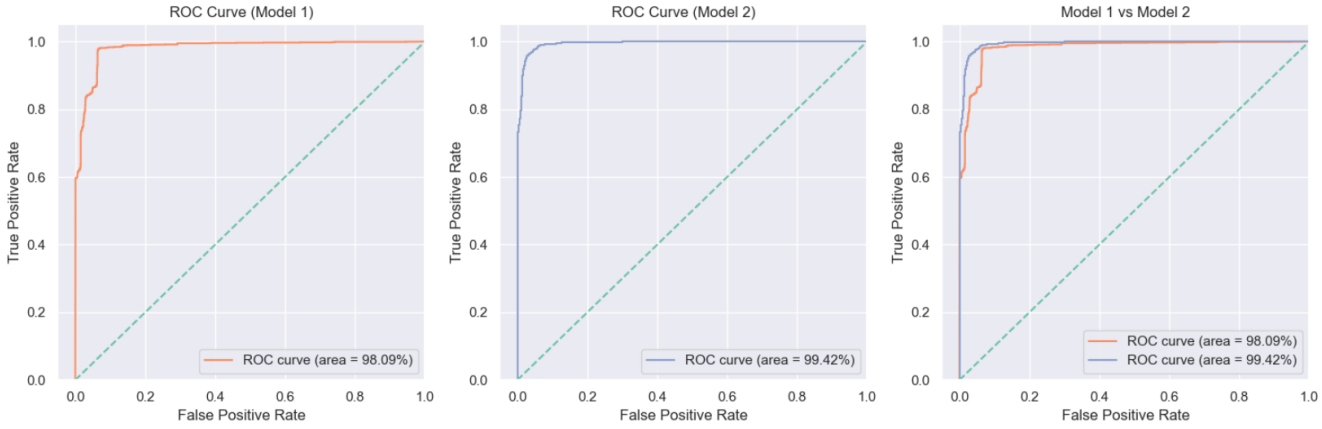


Рисунок 4.10 – ROC-криві моделей методу опорних векторів

Для узагальненого оцінювання здатності моделей розрізняти два класи було побудовано ROC-криві, наведені на рисунку 4.10. Площа під ROC-кривою для моделі svm1 становить 98,09 %, тоді як для моделі svm2 – 99,42 %. Обидві ROC-криві проходять значно вище діагоналі випадкового класифікатора, що свідчить про високу якість бінарного розмежування нормального й атакувального трафіку. Разом із тим рисунок 4.10 чітко показує перевагу другої конфігурації: крива svm2 проходить ближче до верхнього лівого кута координатної площини, що відповідає кращому співвідношенню між чутливістю та частотою хибнопозитивних спрацювань.

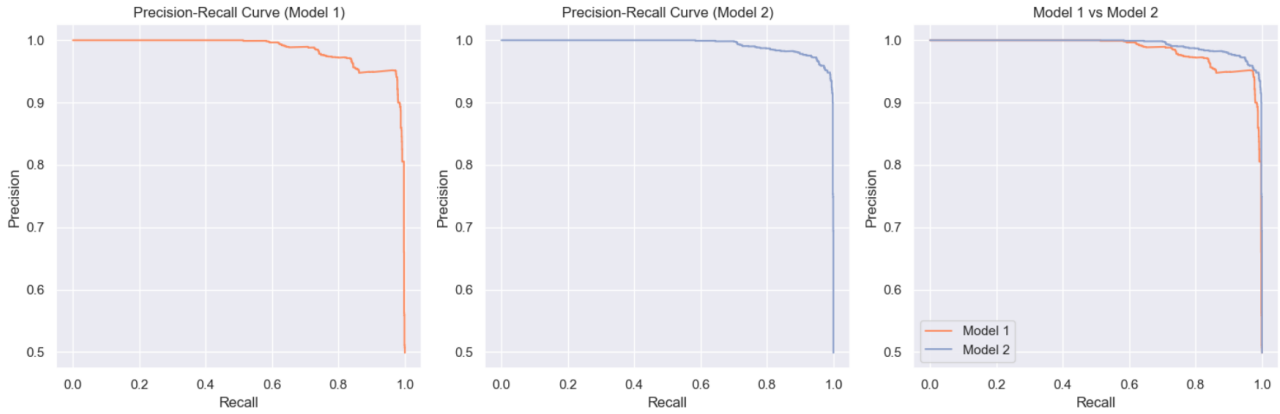


Рисунок 4.11 – Precision–Recall-криві моделей методу опорних векторів

Результати аналізу Precision–Recall-кривих, подані на рисунку 4.11, також підтверджують перевагу моделі svm2. Для обох конфігурацій криві проходять у верхній частині площини «повнота – точність», що свідчить про високу якість класифікації. Проте модель svm2 зберігає вищі значення precision на ширшому діапазоні recall, ніж модель svm1. Це означає, що друга конфігурація краще підтримує баланс між точністю спрацювання та повнотою виявлення атак, що є принципово важливим для практичного використання в системах виявлення атак.

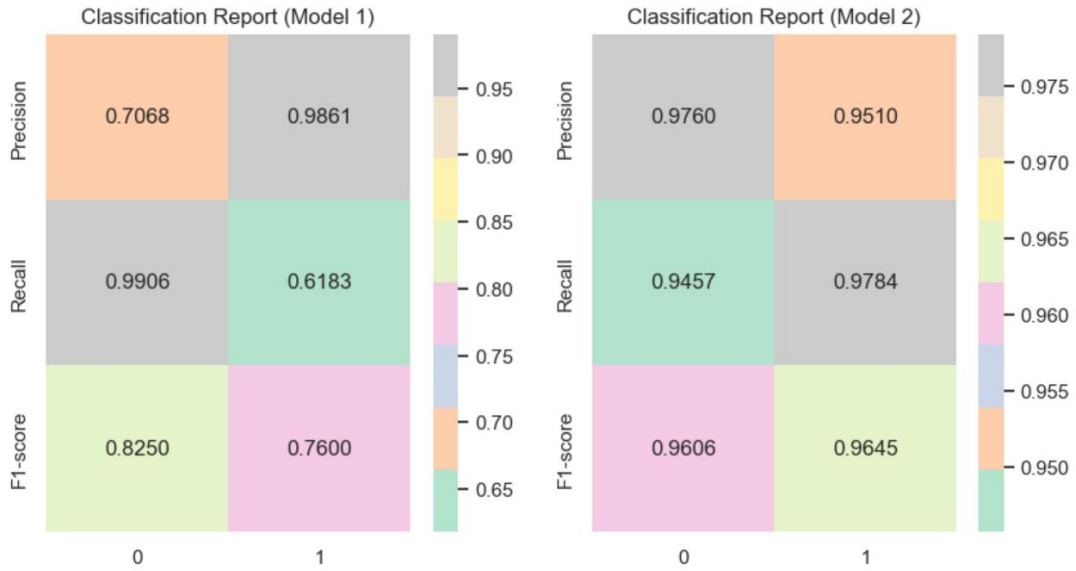


Рисунок 4.12 – Покласові метрики моделей методу опорних векторів

Покласовий аналіз результатів класифікації, поданий у вигляді теплових карт на рисунку 4.12, показав істотні відмінності між двома конфігураціями SVM. Для моделі svm1 значення precision, recall та F1-score для класу нормального трафіку становлять відповідно 0,7068; 0,9906; 0,8250, а для класу атак – 0,9861; 0,6183; 0,7600. Це свідчить про те, що модель 1 добре розпізнає нормальний трафік і

забезпечує високу точність спрацювання для класу атак, однак має недостатню повноту виявлення атакувального трафіку. Для моделі svm2 аналогічні показники для класу нормального трафіку дорівнюють 0,9760; 0,9457; 0,9606, а для класу атак – 0,9510; 0,9784; 0,9645. Отже, друга конфігурація моделі демонструє значно більш збалансовані результати для обох класів і суттєво кращу якість розпізнавання атакувального трафіку.

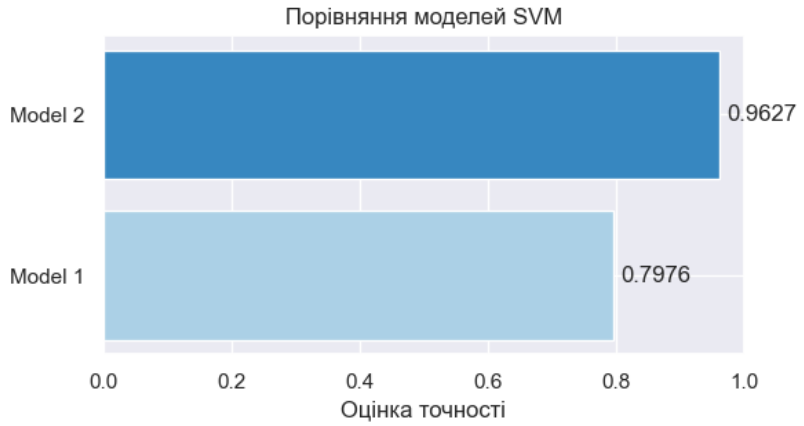


Рисунок 4.13 – Порівняння моделей методу опорних векторів за показником accuracy

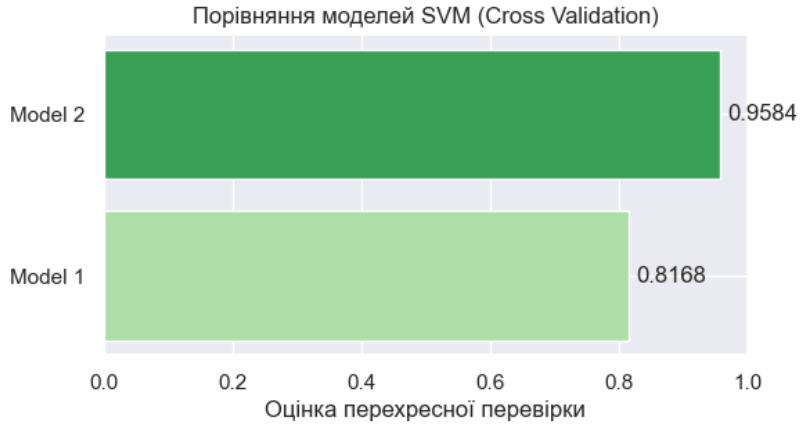


Рисунок 4.14 – Порівняння моделей методу опорних векторів за результатами перехресної перевірки

Інтегральне порівняння двох моделей наведено на рисунках 4.13 і 4.14. Значення accuracy для svm1 становить 0,7976, тоді як для svm2 – 0,9627. Аналогічно, результати п’ятикратної перехресної перевірки для моделі svm1 становили 0,8053; 0,8911; 0,7858; 0,7951; 0,8067, а середнє значення – 0,8168. Для моделі svm2 відповідні значення склали 0,9618; 0,9569; 0,9573; 0,9569; 0,9591, а середній результат – 0,9584. Отримані результати вказують на суттєву перевагу

моделі svm2 над svm1 як за точністю класифікації на тестовій вибірці, так і за стійкістю результатів на навчальних підмножинах.

Результати, наведені на рисунках 4.9–4.14, узгоджено свідчать про те, що друга конфігурація методу опорних векторів (svm2) є значно ефективнішою за першу (svm1) у задачі бінарного виявлення атак. Вона забезпечує вищі значення accuracy, ROC-AUC, середнього результату перехресної перевірки та покласових метрик, а також демонструє значно меншу кількість хибнонегативних рішень. Це дає підстави розглядати модель svm2 як кращий варіант серед досліджених конфігурацій SVM та використовувати її надалі в підсумковому порівнянні з моделлю логістичної регресії.

4.3.3 Порівняння алгоритмів бінарної класифікації

Після окремого дослідження двох конфігурацій логістичної регресії та двох конфігурацій методу опорних векторів було виконано підсумкове порівняння найкращих моделей кожного класу алгоритмів для задачі бінарного виявлення атак. За результатами попереднього аналізу як базову модель логістичної регресії обрано конфігурацію lr2, оскільки вона продемонструвала кращі значення accuracy, середнього результату перехресної перевірки та більш збалансовані покласові метрики порівняно з lr1. Для методу опорних векторів у підсумковому порівнянні використано модель svm2, яка суттєво перевищила svm1 за всіма основними критеріями оцінювання.

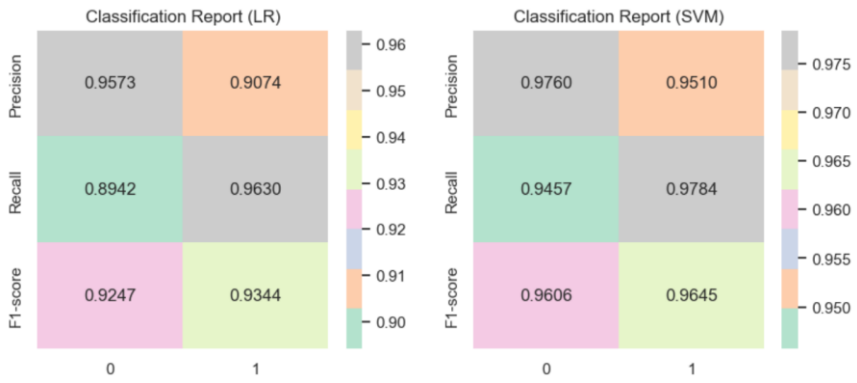


Рисунок 4.15 – Порівняння покласових метрик моделей логістичної регресії та методу опорних векторів

Порівняння покласових метрик логістичної регресії та методу опорних векторів наведено на рисунку 4.15. Для логістичної регресії значення precision, recall та F1-score для класу нормального трафіку становлять відповідно 0,9573; 0,8942; 0,9247, а для класу атак – 0,9074; 0,9630; 0,9344. Для моделі SVM аналогічні показники для класу нормального трафіку дорівнюють 0,9760; 0,9457; 0,9606, а для класу атак – 0,9510; 0,9784; 0,9645. Отже, метод опорних векторів демонструє кращі результати для обох класів за всіма трьома показниками, що свідчить про більш збалансовану та якісну бінарну класифікацію мережевого трафіку.

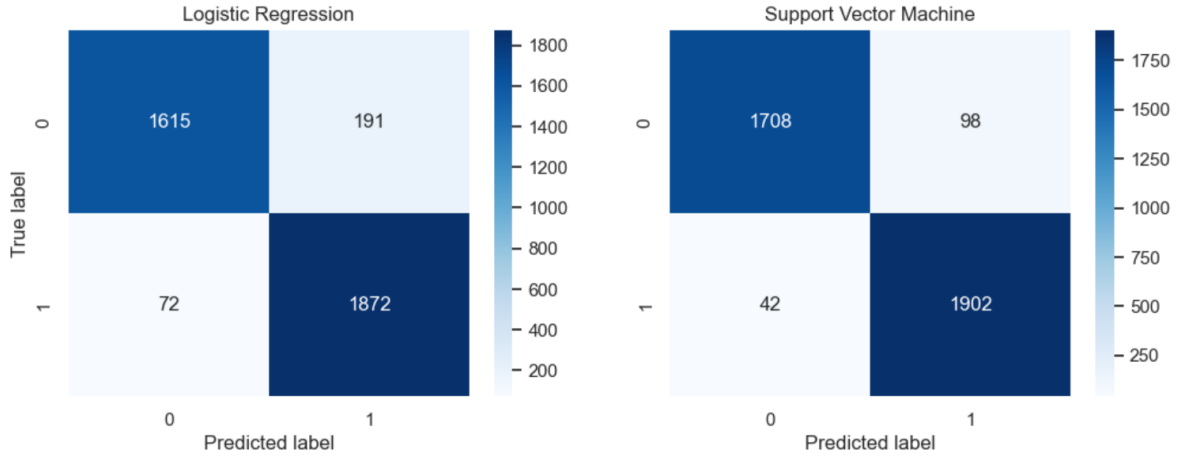


Рисунок 4.16 – Порівняння матриць помилок моделей логістичної регресії та методу опорних векторів

Порівняння матриць помилок для логістичної регресії та методу опорних векторів подано на рисунку 4.16. Для моделі логістичної регресії отримано 1615 правильно класифікованих об’єктів нормального трафіку, 191 хибнопозитивне спрацювання, 72 хибнонегативні рішення та 1872 правильно виявлені атаки. Для моделі SVM відповідні значення становлять 1708, 98, 42 та 1902. Це означає, що метод опорних векторів не лише точніше класифікує обидва класи, а й формує кращу структуру класифікаційних рішень. Особливо важливим є зменшення кількості хибнонегативних рішень з 72 до 42, оскільки саме такі помилки означають пропуск атакувальної активності та є найбільш критичними для систем виявлення атак.

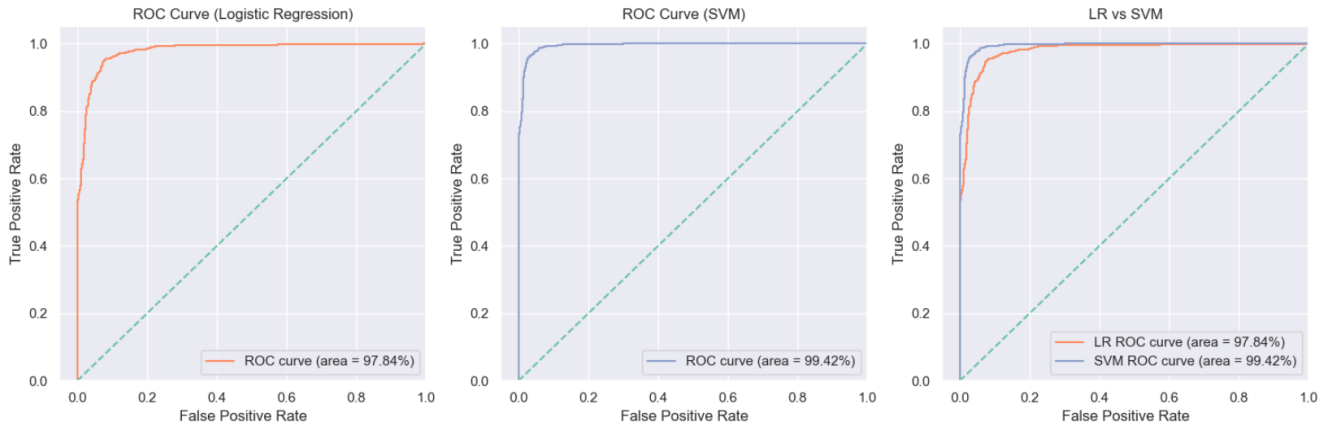


Рисунок 4.17 – Порівняння ROC-кривих моделей логістичної регресії та методу опорних векторів

Узагальнене порівняння здатності моделей розрізняти нормальний та атакувальний трафік наведено на рисунку 4.17 у вигляді ROC-кривих. Для логістичної регресії площа під ROC-кривою становить 97,84 %, тоді як для методу опорних векторів – 99,42 %. Обидві моделі демонструють високий рівень роздільної здатності, однак крива SVM проходить ближче до верхнього лівого кута координатної площини та має більшу площу під кривою. Це свідчить про те, що модель на основі методу опорних векторів краще відокремлює атакувальний трафік від нормального за різних порогів прийняття рішення.

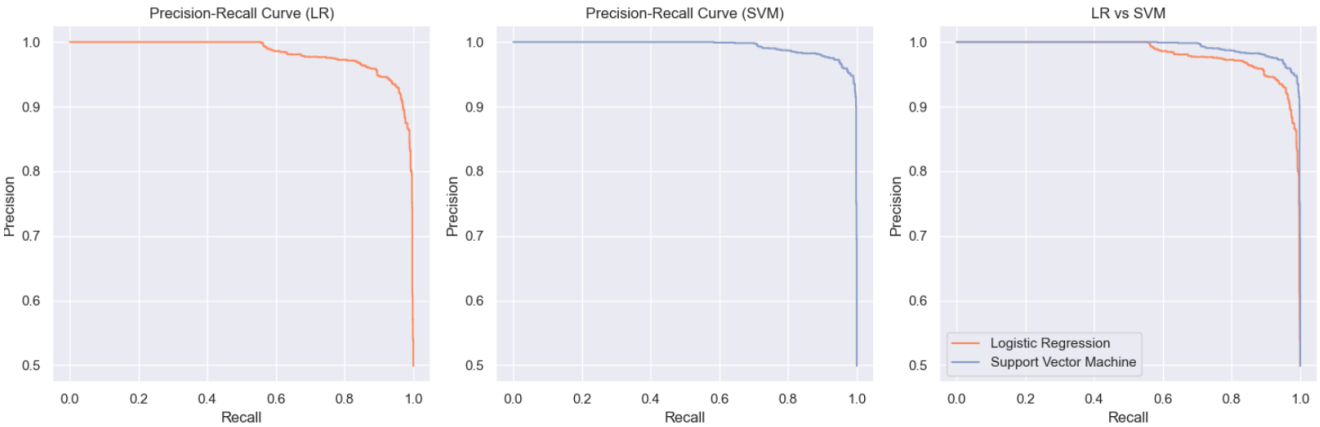


Рисунок 4.18 – Порівняння Precision–Recall-кривих моделей логістичної регресії та методу опорних векторів

Аналогічний висновок випливає і з аналізу Precision–Recall-кривих, наведених на рисунку 4.18. Обидві моделі забезпечують високі значення точності при зростанні повноти, однак крива SVM на більшій частині діапазону recall розташована вище за криву логістичної регресії. Це означає, що метод опорних векторів краще підтримує баланс між precision та recall, тобто ефективніше виявляє атакувальний трафік без істотного збільшення кількості помилкових спрацювань. Для практичного застосування в системах виявлення атак такий результат є особливо важливим, оскільки дозволяє підвищити повноту виявлення загроз за збереження високої точності класифікації.

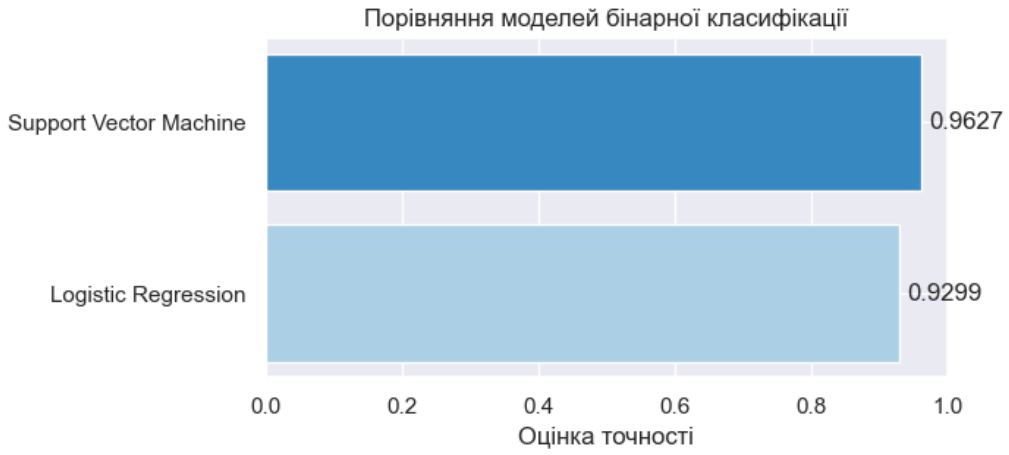


Рисунок 4.19 – Порівняння моделей бінарної класифікації за показником accuracy

Інтегральне порівняння моделей за показником accuracy наведено на рисунку 4.19. Значення точності для логістичної регресії становить 0,9299, тоді як для методу опорних векторів – 0,9627. Отже, різниця на користь SVM становить 0,0328, або 3,28 відсоткового пункта, що є відчутним покращенням для задачі бінарного виявлення атак. Це вказує на те, що застосування нелінійної межі розділення в просторі головних компонент дозволяє моделі SVM краще враховувати структуру даних мережевого трафіку, ніж логістичній регресії.

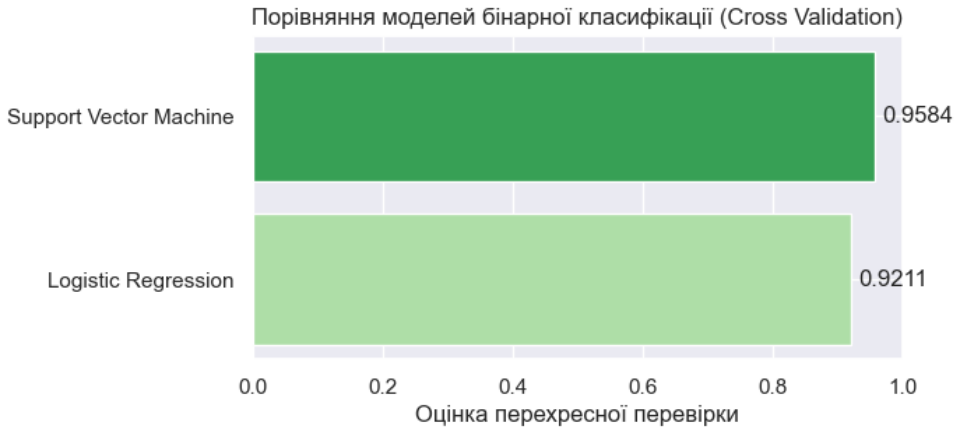


Рисунок 4.20 – Порівняння моделей бінарної класифікації за результатами перехресної перевірки

Подібна тенденція спостерігається і за результатами перехресної перевірки, поданими на рисунку 4.20. Середнє значення 5-fold cross-validation для логістичної регресії дорівнює 0,9211, тоді як для методу опорних векторів – 0,9584. Отже, модель SVM не лише показує вищу точність на тестовій підмножині, а й характеризується кращою стійкістю та узагальнювальною здатністю на навчальних даних. Це є важливим аргументом на користь її використання в межах розробленої інформаційної технології інтелектуального моніторингу трафіку комп'ютерної мережі.

Результати, наведені на рисунках 4.15–4.20, узгоджено свідчать про те, що для задачі бінарного виявлення атак у трафіку комп'ютерної мережі метод опорних векторів є більш ефективним, ніж логістична регресія. Він перевищує логістичну регресію за покласовими показниками precision, recall і F1-score, за значеннями accuracy, ROC-AUC, середнім результатом перехресної перевірки, а також за структурою класифікаційних помилок. Найважливішим практичним результатом є зменшення кількості хибнонегативних рішень, тобто покращення здатності системи своєчасно виявляти шкідливу активність у мережевому трафіку.

Відтак, у межах проведеного експериментального дослідження встановлено, що найкращим алгоритмом для реалізації бінарного режиму інтелектуального моніторингу трафіку комп'ютерної мережі є модель методу опорних векторів svm2. Водночас логістична регресія також продемонструвала високі результати та може

використовуватися як ефективна базова модель бінарної класифікації мережевого трафіку. Однак за сукупністю отриманих оцінок вона поступається SVM, що підтверджує доцільність використання методу опорних векторів як основного алгоритму первинного виявлення атак у межах розробленої інформаційної технології.

4.4 Експериментальне дослідження ефективності багатокласового розпізнавання типів атак

Після дослідження бінарного виявлення атак наступним етапом експериментального аналізу є перевірка ефективності розробленої інформаційної технології в умовах багатокласового розпізнавання типів атак. На відміну від бінарної постановки, де модель повинна лише віднести мережевий потік до нормального або атакувального трафіку, у багатокласовому режимі необхідно визначити конкретний тип атакувальної активності. Така задача є складнішою, оскільки вимагає не лише виявлення факту атаки, а й розмежування між кількома категоріями мережевих загроз.

Початковою основою для формування багатокласової вибірки виступає датафрейм new_data, що містить підготовлений ознаковий простір та мітки типів атак. Аналіз розподілу класів у вихідному наборі даних показав його різко виражену незбалансованість. Найбільшу частку становить нормальний трафік BENIGN – 2096484 записів. Суттєво менше представлені атаки типу DoS – 193748 записів, DDoS – 128016, Port Scan – 90819, Brute Force – 9152, Web Attack – 2143, Bot – 1953. Водночас класи Infiltration та Heartbleed містять лише 36 і 11 записів відповідно. Такий розподіл свідчить, що без спеціальної підготовки даних моделі машинного навчання будуть схильні орієнтуватися на класи більшості, що істотно погіршить якість розпізнавання рідкісних типів атак.

З огляду на це на першому етапі формування робочої багатокласової вибірки було виконано відбір лише тих класів, кількість спостережень у яких перевищує 1950. У результаті до подальшого дослідження було включено сім категорій

мережевої активності: BENIGN, DoS, DDoS, Port Scan, Brute Force, Web Attack та Bot. Класи Infiltration і Heartbleed були виключені з експерименту через надто малу кількість об’єктів, що не дозволяє забезпечити статистично надійне навчання та коректне оцінювання багатокласових моделей.

Після відбору класів було виконано додаткове вирівнювання обсягів вибірок. Для кожного з класів, що містив більше ніж 2500 спостережень, випадковим чином відбиралося по 5000 записів з фіксованим значенням random_state = 0. У результаті було сформовано проміжний датафрейм df, у якому розподіл класів мав такий вигляд: BENIGN – 5000, DoS – 5000, DDoS – 5000, Port Scan – 5000, Brute Force – 5000, Web Attack – 2143, Bot – 1953. Відтак, на цьому етапі вдалося обмежити надмірно великі класи та суттєво зменшити початковий дисбаланс між категоріями мережевого трафіку.

Проте навіть після такого вирівнювання окремі класи залишалися малопредставленими, насамперед Web Attack і Bot. Тому на наступному етапі до сформованої вибірки було застосовано метод синтетичного балансування SMOTE (Synthetic Minority Over-sampling Technique) з параметрами sampling_strategy = 'auto' та random_state = 0. Для цього матрицю ознак X було сформовано шляхом вилучення стовпця Attack Type, а вектор міток y містив відповідні назви класів. У результаті застосування SMOTE було сформовано збалансовані масиви X_upsampled та y_upsampled, на основі яких створено датафрейм blnc_data.

Після синтетичного балансування всі сім класів у датафреймі blnc_data були представлені однаковою кількістю записів – по 5000 спостережень кожен. Остаточний розподіл класів набув такого вигляду: Web Attack – 5000, DDoS – 5000, BENIGN – 5000, Brute Force – 5000, Bot – 5000, DoS – 5000, Port Scan – 5000. Отже, було сформовано повністю збалансовану багатокласову вибірку загальним обсягом 35000 спостережень, придатну для коректного порівняння алгоритмів машинного навчання.

Для усунення впливу порядку записів у вибірці датафрейм blnc_data було додатково перемішано випадковим чином. Після цього з нього сформовано матрицю ознак features та вектор цільових міток labels. На завершальному етапі

підготовки багатокласового експерименту дані було поділено на навчальну та тестову підмножини за допомогою функції train_test_split у співвідношенні 75 % до 25 % при значенні параметра random_state = 0. У результаті для навчання моделей було сформовано 26250 спостережень, а для підсумкового оцінювання – 8750 спостережень [90].

У межах підготовки багатокласового експерименту було реалізовано послідовну процедуру відбору репрезентативних класів, обмеження надмірно великих вибірок, синтетичного балансування за допомогою SMOTE та формування остаточної збалансованої вибірки для навчання моделей. Отриманий датафрейм blnc_data створює коректну експериментальну основу для подальшого дослідження ефективності моделей випадкового лісу, дерева рішень і методу k-найближчих сусідів у задачі багатокласового розпізнавання типів атак.

4.4.1 Дослідження моделей випадкового лісу

Для дослідження ефективності методу випадкового лісу у задачі багатокласового розпізнавання типів атак було використано дві конфігурації моделі. Перша конфігурація (rf1) реалізована з параметрами n_estimators = 10, max_depth = 6, max_features = None, random_state = 0, друга (rf2) – з параметрами n_estimators = 15, max_depth = 8, max_features = 20, random_state = 0. Порівняння цих конфігурацій дає змогу визначити, як кількість дерев, максимальна глибина дерев і кількість ознак, що враховуються під час побудови ансамблю, впливають на якість багатокласового розпізнавання типів атак у мережевому трафіку.

На першому етапі оцінювання для обох конфігурацій було виконано п’ятикратну перехресну перевірку. Для моделі rf1 значення 5-fold cross-validation становили 0,9497; 0,9490; 0,9634; 0,9470; 0,9638, а середній результат – 0,9546. Для моделі rf2 відповідні значення склали 0,9840; 0,9777; 0,9842; 0,9811; 0,9819, а середнє значення – 0,9818. Як видно з рисунка 4.24, друга конфігурація випадкового лісу демонструє вищу стійкість та кращу узагальнювальну здатність на навчальній вибірці.

Після навчання моделей було сформовано прогнози на тестовій підмножині та побудовано матриці помилок, наведені на рисунку 4.23. Обидві моделі забезпечують високий рівень правильного розпізнавання класів, однак rf2 демонструє більш виражену головну діагональ матриці помилок, що свідчить про кращу якість класифікації. Для більшості класів кількість правильно класифікованих спостережень у другої моделі є більшою, а число помилкових віднесень до інших класів – меншою.

Найбільші труднощі для моделі rf1 спостерігаються насамперед під час розпізнавання класу BENIGN, а також класів DoS і Web Attack. Зокрема, частина нормального трафіку помилково відноситься до класів Bot, DoS, Port Scan і Web Attack, а для класу Web Attack помітною є плутанина переважно з BENIGN і Bot. Для класу DoS спостерігаються помилки класифікації у бік BENIGN, Bot і DDoS. Для моделі rf2 структура помилок є значно кращою: кількість неправильних віднесень між класами зменшується, а головна діагональ стає більш вираженою, що свідчить про покращення якості багатокласового розмежування типів атак.

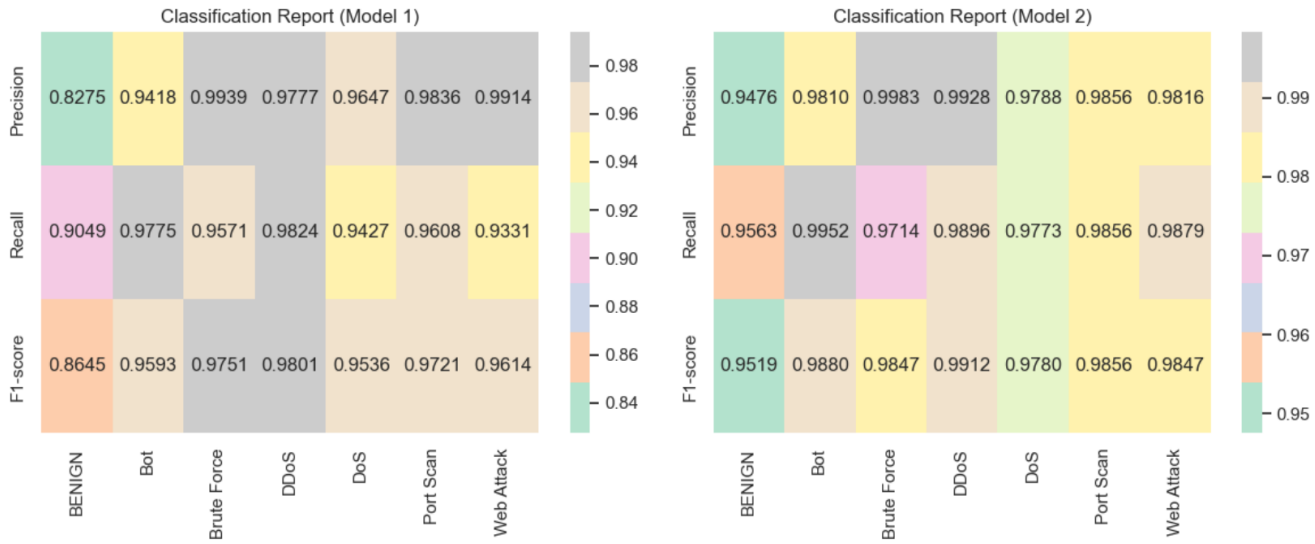


Рисунок 4.21 – Покласові метрики моделей випадкового лісу

Поглиблений покласовий аналіз, наведений на рисунку 4.21 у вигляді теплових карт classification report, також підтверджує перевагу другої конфігурації випадкового лісу. Для моделі rf1 значення precision для класів BENIGN, Bot, Brute

Force, DDoS, DoS, Port Scan і Web Attack становлять відповідно 0,8275; 0,9418; 0,9939; 0,9777; 0,9647; 0,9836; 0,9914. Значення recall для цих самих класів дорівнюють 0,9049; 0,9775; 0,9571; 0,9824; 0,9427; 0,9608; 0,9331, а F1-score – 0,8645; 0,9593; 0,9751; 0,9801; 0,9536; 0,9721; 0,9614.

Для моделі rf2 відповідні значення є вищими для всіх класів. Зокрема, precision становить 0,9476 для BENIGN, 0,9810 для Bot, 0,9983 для Brute Force, 0,9928 для DDoS, 0,9788 для DoS, 0,9856 для Port Scan та 0,9816 для Web Attack. Значення recall дорівнюють відповідно 0,9563; 0,9952; 0,9714; 0,9896; 0,9773; 0,9856; 0,9879, а F1-score – 0,9519; 0,9880; 0,9847; 0,9912; 0,9780; 0,9856; 0,9847. Це свідчить про значно більш збалансовану якість розпізнавання всіх класів у другої конфігурації моделі.

Особливо показовим є порівняння результатів для класів, що є більш складними для класифікації, насамперед BENIGN, DoS і Web Attack. Для цих класів модель rf2 демонструє суттєве покращення як за precision, так і за F1-score, що означає зменшення кількості помилкових рішень і підвищення узагальненої якості розпізнавання. Для класів DDoS, Port Scan і Brute Force обидві моделі показують високі результати, проте й тут друга конфігурація зберігає перевагу.

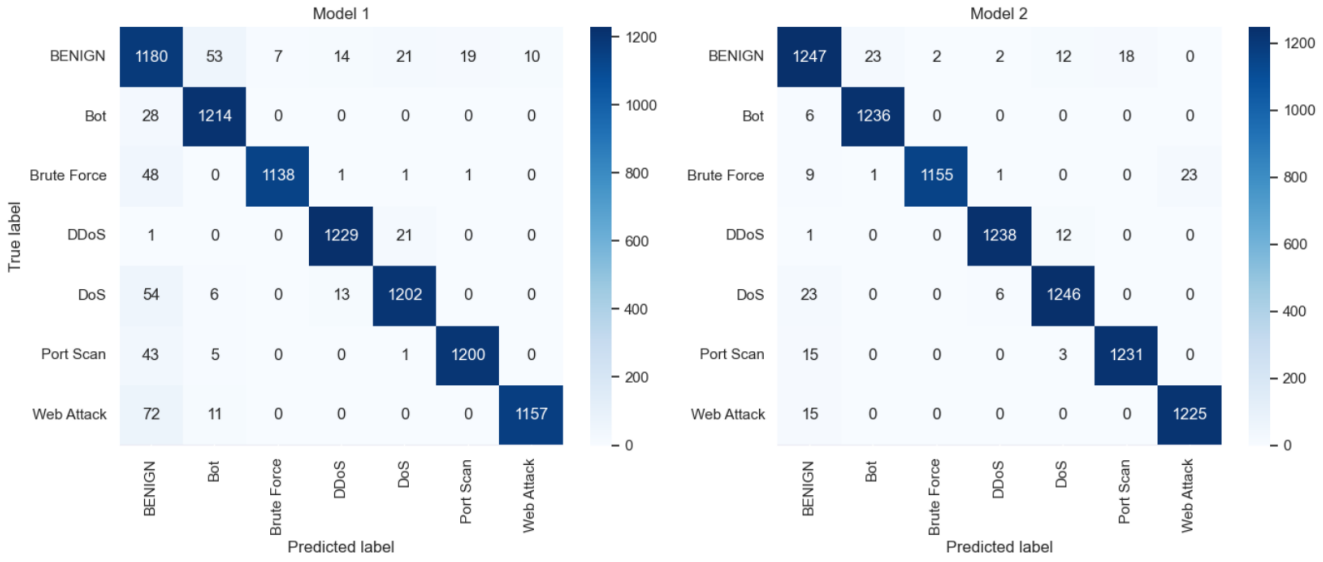


Рисунок 4.22 – Матриці помилок моделей випадкового лісу для задачі багатокласової класифікації

Інтегральне порівняння двох моделей випадкового лісу наведено на рисунку 4.22 та рисунку 4.23. Значення accuracy для моделі rf1 становить 0,9509, тоді як для моделі rf2 – 0,9803. Отже, приріст точності класифікації становить 0,0294, або 2,94 відсоткового пункта, що є суттєвим покращенням у задачі багатокласового розпізнавання типів атак. Аналогічно за показником перехресної перевірки rf2 перевищує rf1: середній результат зростає з 0,9546 до 0,9818.

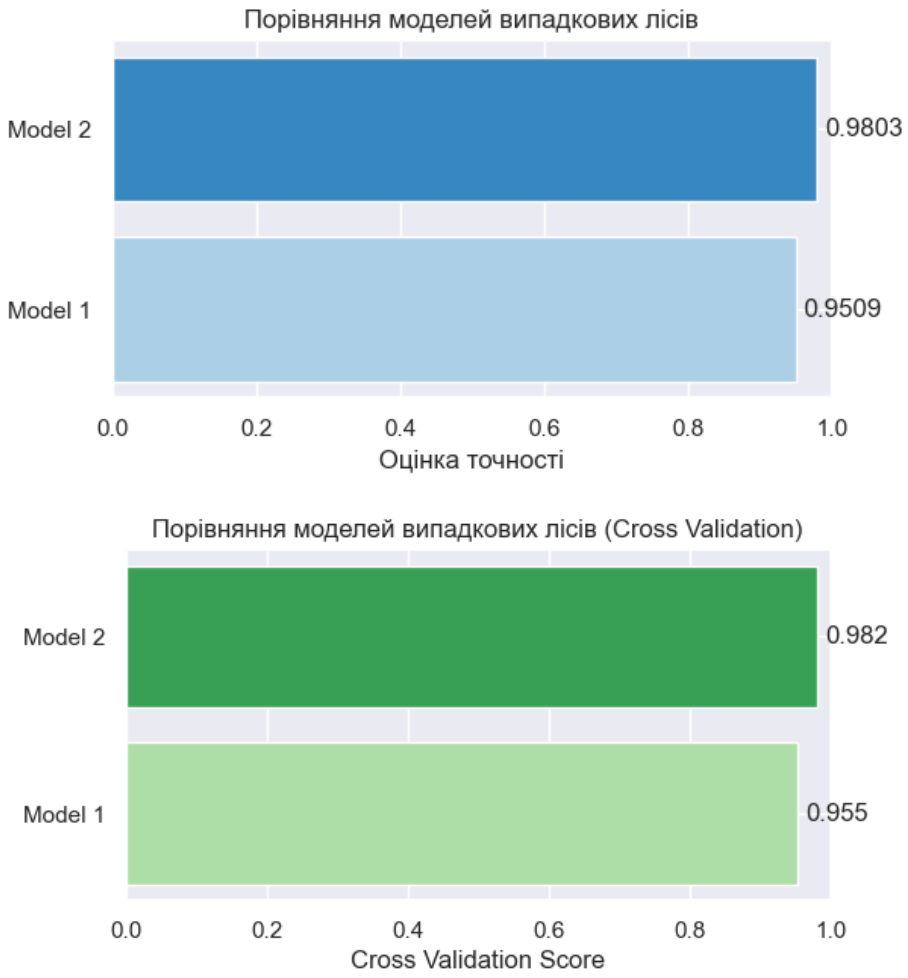


Рисунок 4.23 – Порівняння моделей випадкового лісу за показниками accuracy та cross-validation

Результати, наведені на рисунках 4.21–4.23, узгоджено свідчать про те, що друга конфігурація випадкового лісу (rf2) є більш ефективною за rf1 для задачі багатокласового розпізнавання типів атак. Вона забезпечує вищі значення accuracy, кращі результати перехресної перевірки, більш виражену головну діагональ

матриці помилок і вищі покласові показники precision, recall та F1-score. Це дозволяє розглядати модель rf2 як кращий варіант серед досліджених конфігурацій випадкового лісу та використовувати її надалі в підсумковому порівнянні багатокласових алгоритмів.

4.4.2 Дослідження моделей дерева рішень

Для дослідження ефективності класифікатора дерева рішень у задачі багатокласового розпізнавання типів атак було використано дві конфігурації моделі. Перша конфігурація (dt1) побудована з параметром max_depth = 6, а друга (dt2) – з параметром max_depth = 8. Порівняння цих конфігурацій дає змогу визначити вплив максимальної глибини дерева на якість багатокласового розпізнавання типів атак у мережевому трафіку, оскільки збільшення цього параметра розширює здатність моделі враховувати складніші залежності між ознаками, але водночас може підвищувати ризик перенавчання.

На першому етапі оцінювання для обох конфігурацій було виконано п’ятиразову перехресну перевірку. Для моделі dt1 значення 5-fold cross-validation становили 0,9442; 0,9455; 0,9530; 0,9459; 0,9545, а середній результат – 0,9486. Для моделі dt2 відповідні значення склали 0,9714; 0,9701; 0,9754; 0,9743; 0,9730, а середнє значення – 0,9728. Як видно з рисунка 4.27, друга конфігурація дерева рішень характеризується вищою узагальнювальною здатністю та стійкістю на навчальній вибірці, що свідчить про доцільність використання глибшої структури дерева у межах цієї задачі.

Після навчання моделей було сформовано прогнози на тестовій підмножині та побудовано матриці помилок, наведені на рисунку 4.24. Для моделі dt1 найбільші труднощі спостерігаються під час розпізнавання класів BENIGN і Web Attack. Зокрема, для нормального трафіку частина спостережень помилково віднесена до класів Bot, Brute Force, DoS, Port Scan і Web Attack, а для класу Web Attack найбільша кількість помилкових віднесень припадає на клас BENIGN. У структурі матриці помилок моделі dt1 видно, що кількість правильних

класифікацій залишається високою, однак між окремими класами зберігається помітна взаємна плутанина.

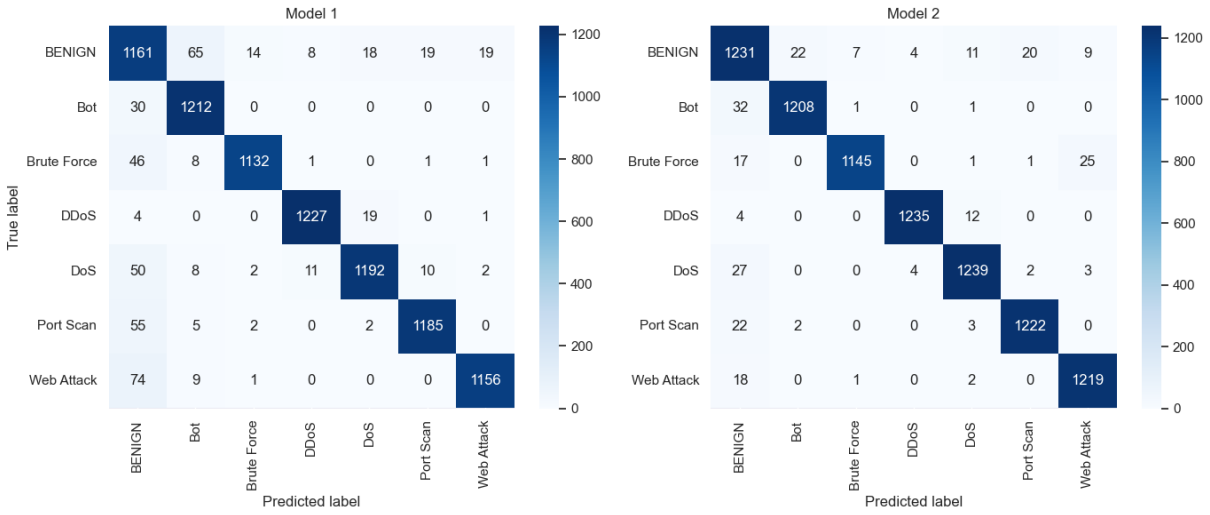


Рисунок 4.24 – Матриці помилок моделей дерева рішень для задачі багатокласової класифікації

Модель dt2 демонструє кращу структуру матриці помилок. Для неї головна діагональ є більш вираженою, а кількість помилкових віднесень між класами – меншою. Особливо це помітно для класів BENIGN, DDoS, DoS, Port Scan та Web Attack. Наприклад, якщо для dt1 клас Web Attack досить часто помилково розпізнавався як BENIGN, то в моделі dt2 кількість таких помилок суттєво зменшується. Аналогічно покращується розпізнавання класу BENIGN, для якого скорочується число неправильних віднесень до Bot, Brute Force, DoS і Web Attack. Це свідчить про те, що збільшення глибини дерева дало змогу моделі краще врахувати структуру ознакового простору і підвищити якість багатокласового розмежування типів атак.

Поглиблений покласовий аналіз, наведений на рисунку 4.25 у вигляді теплових карт classification report, також підтверджує перевагу другої конфігурації дерева рішень. Для моделі dt1 значення precision для класів BENIGN, Bot, Brute Force, DDoS, DoS, Port Scan і Web Attack становлять відповідно 0,8176; 0,9273; 0,9835; 0,9840; 0,9683; 0,9753; 0,9805. Значення recall для цих самих класів дорівнюють 0,8903; 0,9758; 0,9521; 0,9808; 0,9349; 0,9488; 0,9323, а F1-score –

0,8524; 0,9510; 0,9675; 0,9824; 0,9513; 0,9619; 0,9558. Отже, попри високі результати для більшості класів, у моделі dt1 найнижчі показники спостерігаються саме для класів BENIGN і Web Attack.

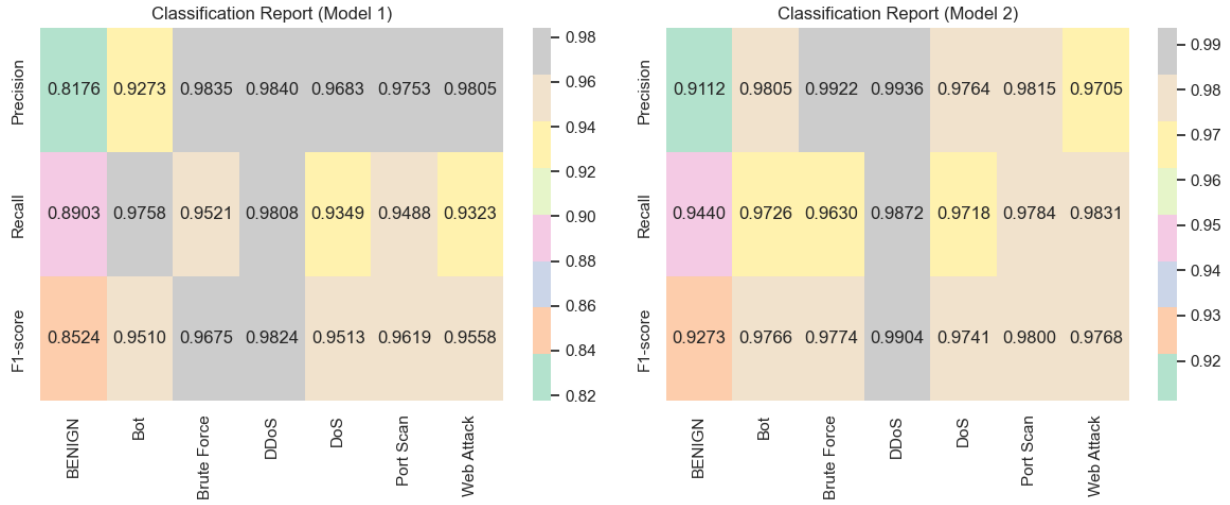


Рисунок 4.25 – Покласові метрики моделей дерева рішень

Для моделі dt2 відповідні показники є вищими майже для всіх класів. Так, значення precision становить 0,9112 для BENIGN, 0,9805 для Bot, 0,9922 для Brute Force, 0,9936 для DDoS, 0,9764 для DoS, 0,9815 для Port Scan та 0,9705 для Web Attack. Значення recall дорівнюють відповідно 0,9440; 0,9726; 0,9630; 0,9872; 0,9718; 0,9784; 0,9831, а F1-score – 0,9273; 0,9766; 0,9774; 0,9904; 0,9741; 0,9800; 0,9768. Ці результати свідчать, що друга конфігурація дерева рішень забезпечує більш збалансовану якість класифікації та суттєво краще розпізнає складніші класи, передусім BENIGN і Web Attack.

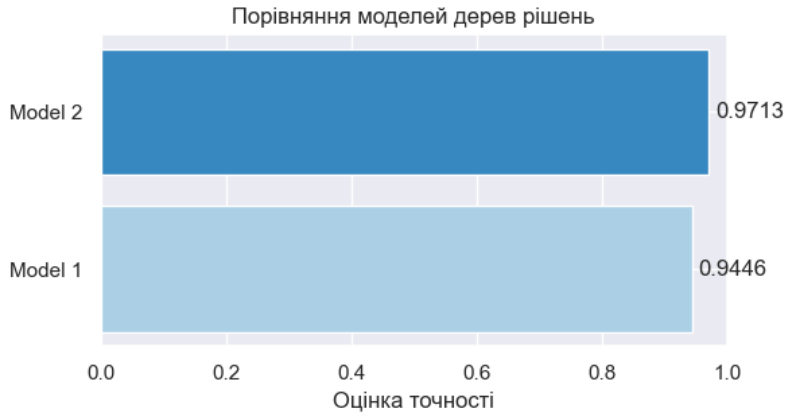


Рисунок 4.26 – Порівняння моделей дерева рішень за показником accuracy

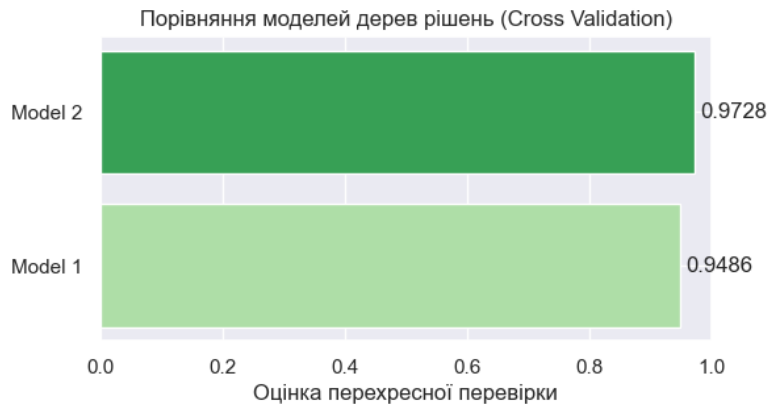


Рисунок 4.27 – Порівняння моделей дерева рішень за результатами перехресної перевірки

Інтегральне порівняння двох моделей дерева рішень наведено на рисунках 4.26 і 4.27. Значення accuracy для моделі dt1 становить 0,9446, тоді як для моделі dt2 – 0,9713. Отже, приріст точності класифікації становить 0,0267, або 2,67 відсоткового пункта, що є суттєвим покращенням для задачі багатокласового розпізнавання типів атак. Подібна тенденція спостерігається і за результатами перехресної перевірки: середній результат зростає з 0,9486 до 0,9728. Це означає, що модель dt2 не лише краще класифікує тестові дані, а й демонструє вищу стійкість на навчальній вибірці.

Результати, наведені на рисунках 4.26–4.27, узгоджено свідчать про те, що друга конфігурація дерева рішень (dt2) є більш ефективною за dt1 для задачі багатокласового розпізнавання типів атак. Вона забезпечує вищі значення accuracy, кращі результати перехресної перевірки, більш виражену головну діагональ матриці помилок і вищі покласові показники precision, recall та F1-score. Це дозволяє розглядати модель dt2 як кращий варіант серед досліджених конфігурацій дерева рішень та використовувати її надалі в підсумковому порівнянні багатокласових алгоритмів.

4.4.3 Дослідження моделей k-найближчих сусідів

Для дослідження ефективності методу k-найближчих сусідів у задачі багатокласового розпізнавання типів атак було використано дві конфігурації

моделі. Перша конфігурація (knn1) реалізована з параметром n_neighbors = 16, друга (knn2) – з параметром n_neighbors = 8. Зіставлення цих конфігурацій дає змогу оцінити вплив кількості сусідів на якість багатокласової класифікації мережевого трафіку, оскільки зі зменшенням значення k модель стає чутливішою до локальної структури ознакового простору, а зі збільшенням – більше орієнтується на узагальнений розподіл об’єктів.

На першому етапі оцінювання для обох конфігурацій було виконано п’ятикратну перехресну перевірку. Для моделі knn1 значення 5-fold cross-validation становили 0,9739; 0,9703; 0,9747; 0,9743; 0,9730, а середній результат дорівнював 0,9732. Для моделі knn2 відповідні значення склали 0,9810; 0,9766; 0,9806; 0,9811; 0,9771, а середнє значення становило 0,9793. Як видно з рисунка 4.31, друга конфігурація методу k-найближчих сусідів має дещо вищу узагальнювальну здатність і стабільніші результати на навчальній вибірці.

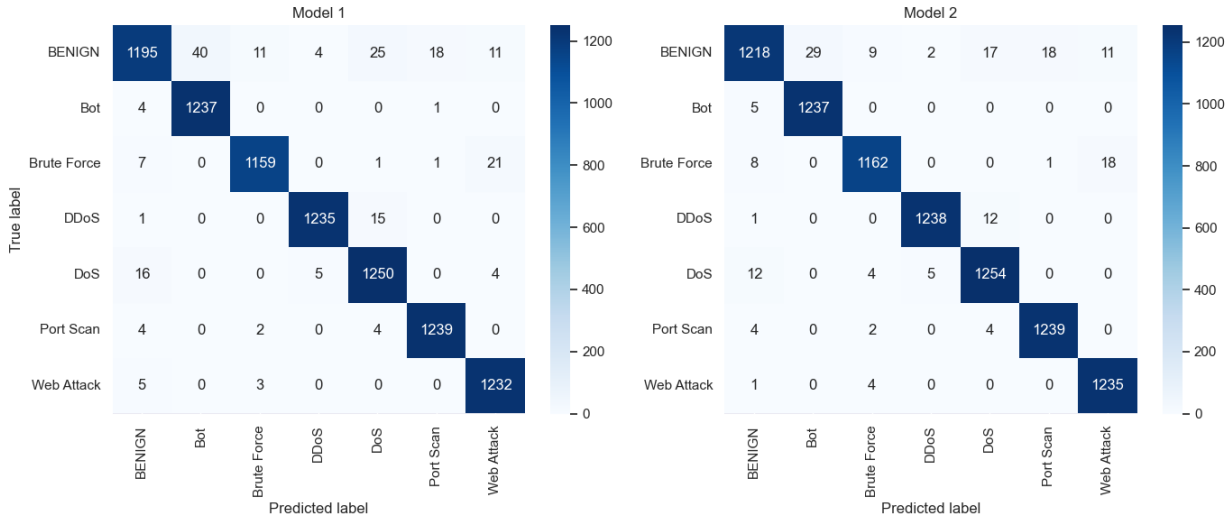


Рисунок 4.28 – Матриці помилок моделей методу k-найближчих сусідів для задачі багатокласової класифікації

Після навчання моделей було сформовано прогнози на тестовій підмножині та побудовано матриці помилок, наведені на рисунку 4.28. Для обох конфігурацій методу k-найближчих сусідів характерною є чітко виражена головна діагональ, що свідчить про високий рівень правильного розпізнавання всіх класів. Водночас

модель knn2 демонструє більш чисту структуру матриці помилок, тобто меншу кількість неправильних віднесень між класами.

Найбільші труднощі для обох конфігурацій спостерігаються під час розпізнавання класу BENIGN, який частково плутається з класами Bot, DoS, Port Scan і Web Attack. Для моделі knn1 така плутанина є більш помітною: нормальний трафік частіше помилково відноситься до Bot і DoS, а також в окремих випадках – до Brute Force, DDoS, Port Scan та Web Attack. У моделі knn2 кількість таких помилок зменшується, що свідчить про краще відокремлення нормального трафіку від атакувальних класів у просторі головних компонент.

Для класів Bot, DDoS, DoS і Port Scan обидві конфігурації методу k-найближчих сусідів демонструють дуже високий рівень правильного розпізнавання. Найскладнішими для класифікації, окрім BENIGN, залишаються класи Brute Force та Web Attack, між якими фіксуються поодинокі помилкові віднесення. Проте в цілому структура матриці помилок свідчить про те, що метод k-найближчих сусідів є ефективним для багатокласового розпізнавання типів атак, а модель knn2 формує якісніші класифікаційні рішення, ніж knn1.

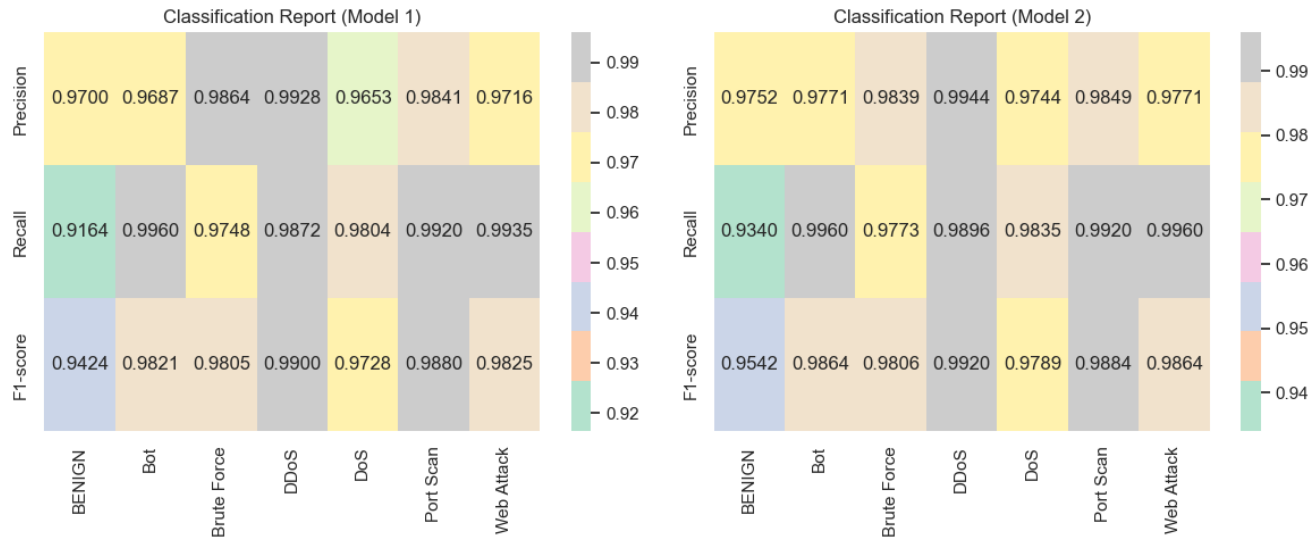


Рисунок 4.29 – Покласові метрики моделей методу k-найближчих сусідів

Поглиблений покласовий аналіз, наведений на рисунку 4.29 у вигляді теплових карт classification report, також підтверджує перевагу другої конфігурації.

Для моделі knn1 значення precision для класів BENIGN, Bot, Brute Force, DDoS, DoS, Port Scan і Web Attack становлять відповідно 0,9700; 0,9687; 0,9864; 0,9928; 0,9653; 0,9841; 0,9716. Значення recall для цих самих класів дорівнюють 0,9164; 0,9960; 0,9748; 0,9872; 0,9804; 0,9920; 0,9935, а F1-score – 0,9424; 0,9821; 0,9805; 0,9900; 0,9728; 0,9880; 0,9825.

Для моделі knn2 відповідні покласові метрики є дещо вищими. Так, значення precision становить 0,9752 для BENIGN, 0,9771 для Bot, 0,9839 для Brute Force, 0,9944 для DDoS, 0,9744 для DoS, 0,9849 для Port Scan та 0,9771 для Web Attack. Значення recall дорівнюють відповідно 0,9340; 0,9960; 0,9773; 0,9896; 0,9835; 0,9920; 0,9960, а F1-score – 0,9542; 0,9864; 0,9806; 0,9920; 0,9789; 0,9884; 0,9864. Це свідчить про те, що друга конфігурація методу k-найближчих сусідів забезпечує кращий баланс між точністю, повнотою та узагальненою якістю класифікації для більшості класів.

Особливо показовим є покращення для класу BENIGN, де F1-score зростає з 0,9424 до 0,9542, а також для класу DoS, де F1-score підвищується з 0,9728 до 0,9789. Для класів Bot, DDoS, Port Scan і Web Attack обидві моделі демонструють дуже високі значення метрик, однак knn2 зберігає невелику, але стійку перевагу. Це свідчить про те, що використання меншого значення k у цій задачі дозволяє точніше враховувати локальну структуру даних без істотної втрати стійкості моделі.

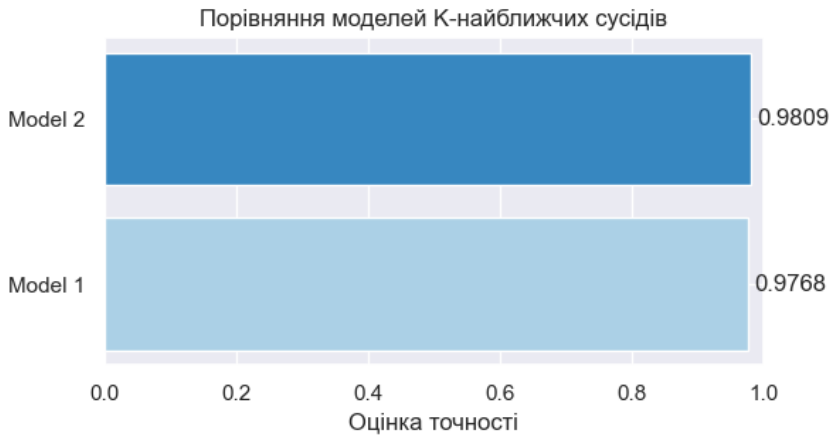


Рисунок 4.30 – Порівняння моделей методу k-найближчих сусідів за показником accuracy

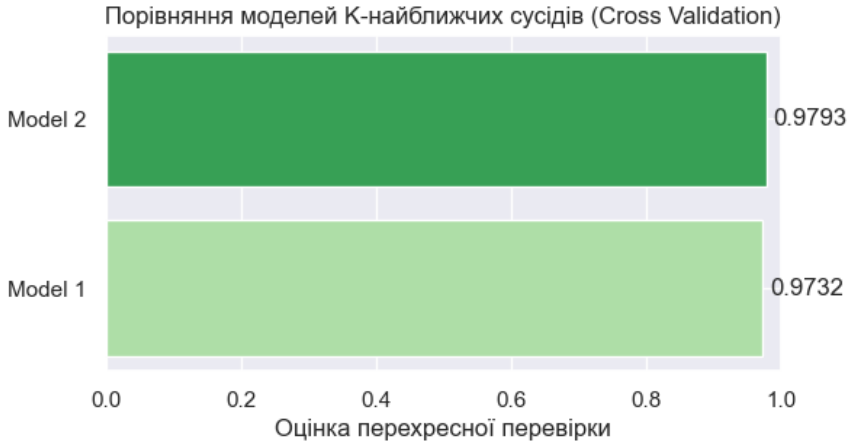


Рисунок 4.31 – Порівняння моделей методу k-найближчих сусідів за результатами перехресної перевірки

Інтегральне порівняння двох конфігурацій наведено на рисунках 4.30 і 4.31. Значення accuracy для моделі knn1 становить 0,9768, тоді як для knn2 – 0,9809. Отже, приріст точності класифікації становить 0,0041, або 0,41 відсоткового пункта. Аналогічна тенденція спостерігається і за результатами перехресної перевірки: середній показник збільшується з 0,9732 до 0,9793. Хоча різниця між конфігураціями є меншою, ніж у випадку випадкового лісу чи дерева рішень, вона є сталою для всіх основних показників оцінювання і підтверджує перевагу другої моделі.

Результати, наведені на рисунках 4.28–4.31, узгоджено свідчать про те, що друга конфігурація методу k–найближчих сусідів (knn2) є більш ефективною за knn1 для задачі багатокласового розпізнавання типів атак. Вона забезпечує вищі значення accuracy, кращі результати перехресної перевірки, менш виражену міжкласову плутанину та вищі покласові показники precision, recall і F1–score. Це дозволяє розглядати модель knn2 як кращий варіант серед досліджених конфігурацій методу k–найближчих сусідів та використовувати її надалі в підсумковому порівнянні багатокласових алгоритмів.

4.4.4 Порівняння моделей багатокласової класифікації

Після окремого дослідження двох конфігурацій для кожного алгоритму багатокласової класифікації було виконано підсумкове порівняння найкращих моделей кожного класу. Відбір здійснювався за сукупністю показників якості, зокрема за значеннями accuracy, результатами перехресної перевірки, покласовими метриками precision, recall, F1-score та характером міжкласових помилок. За результатами попередніх підрозділів для фінального зіставлення було обрано такі моделі: для алгоритму випадкового лісу – rf2, для дерева рішень – dt2, для методу k-найближчих сусідів – knn2.

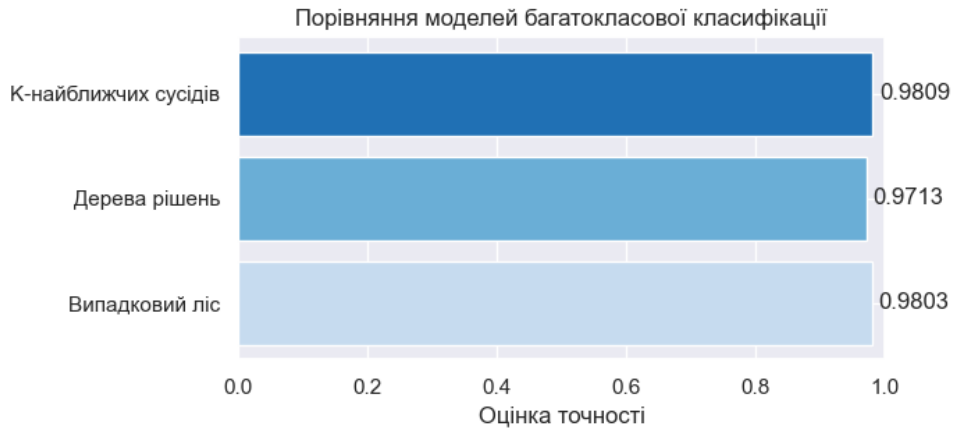


Рисунок 4.32 – Порівняння моделей багатокласової класифікації за показником accuracy

Інтегральне порівняння моделей за показником accuracy наведено на рисунку 4.32. Найвище значення точності класифікації продемонстрував метод k-найближчих сусідів – 0,9809. Друге місце посідає випадковий ліс із результатом 0,9803, а третє – дерево рішень із показником 0,9713. Отже, за підсумковою точністю класифікації модель knn2 виявилася найкращою серед усіх досліджених багатокласових алгоритмів. Водночас різниця між knn2 і rf2 є дуже незначною та становить лише 0,0006, або 0,06 відсоткового пункта.

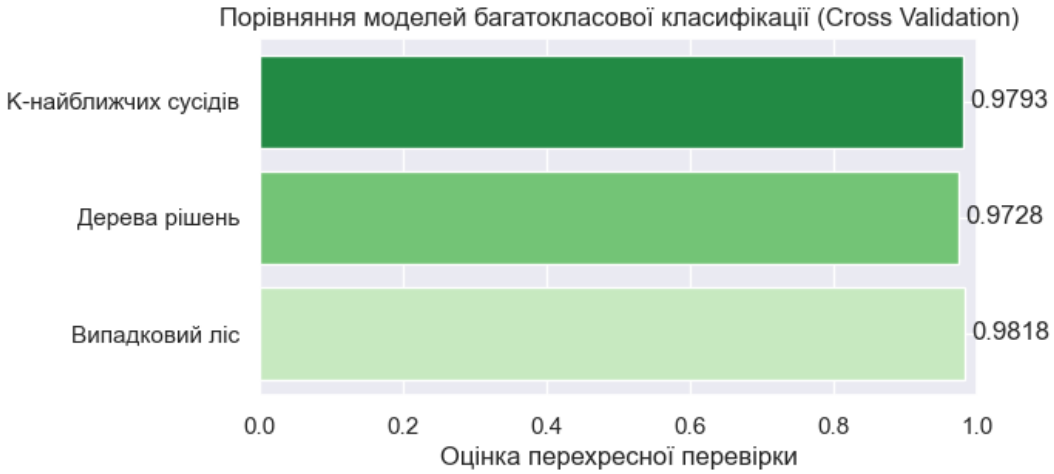


Рисунок 4.33 – Порівняння моделей багатокласової класифікації за результатами перехресної перевірки

Разом із тим результати перехресної перевірки, подані на рисунку 4.33, демонструють дещо іншу картину. Найвище середнє значення 5-fold cross-validation має випадковий ліс – 0,9818, тоді як для k-найближчих сусідів цей показник становить 0,9793, а для дерева рішень – 0,9728. Це означає, що хоча метод k-найближчих сусідів демонструє найкращу точність на тестовій підмножині, випадковий ліс є дещо стійкішим на навчальних підмножинах і характеризується вищою узагальнювальною здатністю.

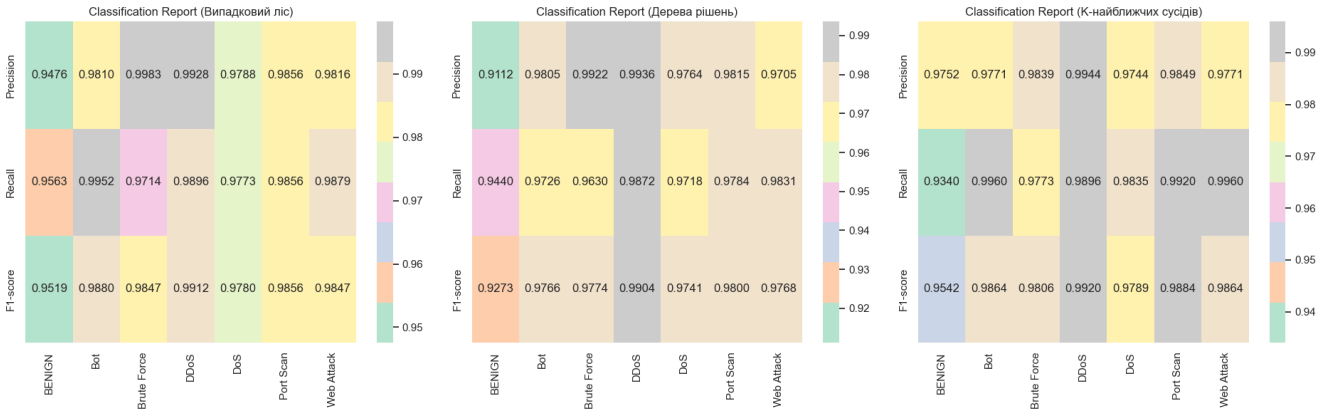


Рисунок 4.34 – Порівняння покласових метрик найкращих моделей багатокласової класифікації

Порівняння покласових метрик трьох найкращих моделей наведено на рисунку 4.34. Аналіз значень precision, recall і F1-score показує, що жодна з моделей не є безумовно найкращою для всіх класів, однак між ними спостерігаються виражені відмінності. Для класу BENIGN найвище значення F1-score має метод k-найближчих сусідів – 0,9542, тоді як для випадкового лісу воно становить 0,9519, а для дерева рішень – 0,9273. Це свідчить про те, що саме knn2 забезпечує найкращий баланс точності та повноти для нормального трафіку в межах багатокласової постановки.

Для класу Bot найкращий результат показує випадковий ліс, де F1-score дорівнює 0,9880. Для методу k-найближчих сусідів цей показник становить 0,9864, а для дерева рішень – 0,9766. Для класу Web Attack найвище значення F1-score має knn2 – 0,9864, для випадкового лісу воно становить 0,9847, а для дерева рішень – 0,9768. Це дозволяє зробити висновок, що за розпізнаванням менш типових або складніших категорій активності метод k-найближчих сусідів має певну перевагу, тоді як випадковий ліс дещо краще диференціює клас Bot.

Для класу Brute Force найкращий результат демонструє випадковий ліс, де F1-score становить 0,9847, тоді як для knn2 він дорівнює 0,9806, а для дерева рішень – 0,9774. Водночас для класів DDoS, DoS і Port Scan найкращі результати показує метод k-найближчих сусідів. Для DDoS значення F1-score становить 0,9920, для DoS – 0,9789, для Port Scan – 0,9884. Для цих самих класів випадковий ліс демонструє значення 0,9912, 0,9780 та 0,9856 відповідно, а дерево рішень – 0,9904, 0,9741 та 0,9800. Це означає, що метод k-найближчих сусідів забезпечує найкращу якість розпізнавання для більшості класів, тоді як випадковий ліс зберігає перевагу для окремих типів атак, насамперед Bot і Brute Force.

Порівняння матриць помилок трьох найкращих моделей наведено на рисунку 4.35. Усі моделі демонструють чітко виражену головну діагональ, що свідчить про високий загальний рівень правильного розпізнавання класів. Проте структура помилок відрізняється.

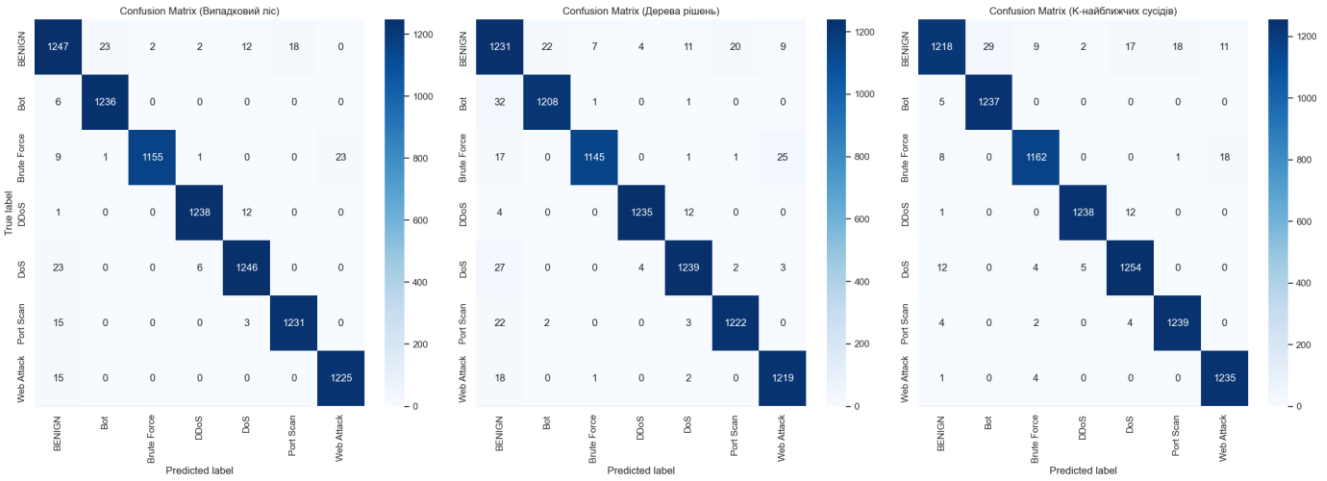


Рисунок 4.35 – Порівняння матриць помилок найкращих моделей багатокласової класифікації

Для випадкового лісу характерне краще розпізнавання класу BENIGN, що підтверджується найбільшою кількістю правильних класифікацій на головній діагоналі для цього класу. Дерево рішень загалом демонструє подібну структуру помилок, однак має дещо більшу кількість неправильних віднесень, особливо для класів BENIGN та Web Attack.

Метод k-найближчих сусідів, своєю чергою, демонструє найкращі результати для більшості атакувальних класів, зокрема Bot, Brute Force, DDoS, DoS, Port Scan і Web Attack, що узгоджується з його вищими покласовими метриками для цих категорій. Водночас для класу BENIGN у нього спостерігається дещо більша кількість помилкових віднесень до класів Bot, DoS, Port Scan і Web Attack, ніж у випадкового лісу. Таким чином, матриці помилок підтверджують, що knn2 має перевагу за якістю розпізнавання більшості типів атак, тоді як rf2 дещо краще відокремлює нормальний трафік від атакувального.

Результати, наведені на рисунках 4.32–4.35, свідчать про те, що найвищу точність багатокласової класифікації демонструє метод k-найближчих сусідів, тоді як найкращу узагальнювальну здатність і найстійкіші результати перехресної перевірки забезпечує випадковий ліс. Дерево рішень поступається обом алгоритмам як за accuracy, так і за показниками cross-validation та покласовими метриками, хоча також демонструє достатньо високий рівень якості класифікації.

У межах проведеного експериментального дослідження можна зробити висновок, що для задачі багатокласового розпізнавання типів атак доцільно розглядати два основні варіанти використання моделі залежно від пріоритетів системи. Якщо головним критерієм є максимальна підсумкова точність класифікації та найкращі результати для більшості класів, то найкращим вибором є модель knn2. Якщо ж пріоритетом є вища стійкість результатів і дещо краща узагальнювальна здатність, то більш доцільним є використання моделі rf2. У цілому обидві моделі продемонстрували дуже високі результати і підтвердили ефективність розробленої інформаційної технології для багатокласового аналізу мережевого трафіку.

4.5 Порівняльний аналіз результатів та інтерпретація ефективності розробленої інформаційної технології

Порівняльний аналіз результатів експериментального дослідження дає змогу узагальнити ефективність розробленої інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі не лише на рівні окремих алгоритмів, а як цілісного програмно-алгоритмічного конвеєра. У межах проведених експериментів усі моделі функціонували в однакових умовах: на основі єдиного підготовленого ознакового простору, сформованого після очищення даних, стандартизації та зниження розмірності методом головних компонент. Для бінарної та багатокласової постановок використовувалися окремо сформовані збалансовані вибірки, а оцінювання здійснювалося за сукупністю взаємодоповнювальних критеріїв: accuracy, результатів 5-fold cross-validation, матриць помилок, ROC- і Precision–Recall-кривих, а також покласових показників precision, recall і F1-score. Саме така організація дослідження дозволяє коректно інтерпретувати одержані результати та робити обґрунтовані висновки щодо практичної придатності розробленої інформаційної технології.

У бінарній постановці задачі було встановлено, що логістична регресія та метод опорних векторів обидва забезпечують високий рівень виявлення атак, однак

між ними існує помітна різниця в ефективності. Найкраща модель логістичної регресії (lr2) продемонструвала accuracy = 0.9299, середній результат перехресної перевірки 0.9211 та значення ROC-AUC = 97.84 %. Для цієї моделі покласові метрики становили: для нормального трафіку precision = 0.9573, recall = 0.8942, F1-score = 0.9247, а для атакувального трафіку – precision = 0.9074, recall = 0.9630, F1-score = 0.9344. Структура матриці помилок показала, що модель забезпечує якісне розмежування двох класів, однак усе ще допускає певну кількість хибнопозитивних і хибнонегативних рішень.

Найкраща модель методу опорних векторів (svm2) виявилася ефективнішою за всіма основними показниками. Для неї значення accuracy становило 0.9627, середній результат перехресної перевірки – 0.9584, а ROC-AUC досягло 99.42 %. Покласові метрики цієї моделі також виявилися вищими: для класу нормального трафіку precision = 0.9760, recall = 0.9457, F1-score = 0.9606, а для класу атак – precision = 0.9510, recall = 0.9784, F1-score = 0.9645. Особливо важливим практичним результатом стало зменшення кількості хибнонегативних рішень у порівнянні з логістичною регресією, що означає кращу здатність своєчасно виявляти шкідливу активність у мережевому трафіку. Таким чином, для реалізації першого рівня розробленої інформаційної технології, тобто первинного виявлення атак, найбільш доцільним є використання саме моделі svm2.

Результати багатокласової класифікації свідчать, що всі три досліджені моделі – випадковий ліс, дерево рішень і метод k-найближчих сусідів – забезпечують високий рівень розпізнавання типів атак, однак між ними також спостерігаються важливі відмінності. Найкраща модель випадкового лісу (rf2) продемонструвала accuracy = 0.9803 та середнє значення 5-fold cross-validation = 0.9818. Покласові результати цієї моделі виявилися особливо високими для класів Bot і Brute Force, де значення F1-score становили відповідно 0.9880 та 0.9847. Водночас для інших класів модель також показала дуже високі результати: 0.9519 для BENIGN, 0.9912 для DDoS, 0.9780 для DoS, 0.9856 для Port Scan і 0.9847 для Web Attack. Це свідчить про високу здатність випадкового лісу до стійкого

розпізнавання типів атак та його кращу узагальнювальну здатність серед багатокласових моделей.

Модель дерева рішень dt2 продемонструвала дещо нижчі, але також високі результати: accuracy = 0.9713, середнє значення перехресної перевірки – 0.9728. Для окремих класів її показники були близькими до випадкового лісу та методу k-найближчих сусідів, однак загалом вона поступалася їм як за інтегральною точністю, так і за стійкістю результатів. Найбільш помітне відставання спостерігалося під час розпізнавання класів BENIGN і Web Attack, для яких значення F1-score становили 0.9273 та 0.9768 відповідно, тоді як у найкращих моделей інших алгоритмів ці значення були вищими. Отже, дерево рішень можна розглядати як достатньо ефективну, але менш конкурентоспроможну модель порівняно з випадковим лісом і k-NN.

Найвищу підсумкову точність у задачі багатокласового розпізнавання типів атак продемонстрував метод k-найближчих сусідів у конфігурації knn2. Для нього значення accuracy становило 0.9809, а середній результат перехресної перевірки – 0.9793. Ця модель виявилася найкращою за розпізнаванням класів BENIGN, DDoS, DoS, Port Scan і Web Attack, де значення F1-score досягли відповідно 0.9542, 0.9920, 0.9789, 0.9884 і 0.9864. Такий результат є особливо важливим, оскільки саме для частини цих класів у багатьох алгоритмів часто виникає найбільша плутанина. Модель knn2 найкраще врахувала локальну структуру багатокласового ознакового простору та забезпечила найвищу загальну точність класифікації.

Разом із тим порівняння knn2 і rf2 показує, що між ними немає абсолютного домінування однієї моделі за всіма критеріями. Метод k-найближчих сусідів має найвище значення accuracy, тобто найкращу підсумкову якість класифікації на тестовій підмножині. Натомість випадковий ліс має дещо вищий результат перехресної перевірки та кращі покласові характеристики для таких типів атак, як Bot і Brute Force. Під час вибору моделі для другого рівня інформаційної технології необхідно враховувати не лише загальну точність класифікації, а й операційні пріоритети системи. Якщо головною є максимальна точність багатокласового розпізнавання, то доцільніше використовувати knn2. Якщо ж пріоритетом є вища

стійкість моделі та дещо краща збалансованість результатів на навчальних підмножинах, то аргументованим є вибір rf2.

Отримані експериментальні результати свідчать, що запропонована інформаційна технологія забезпечує стабільне виконання основних функцій інтелектуального моніторингу мережевого трафіку в бінарному та багатокласовому режимах аналізу. Це створює передумови для підвищення гарантоздатності систем виявлення атак, оскільки дає змогу підтримувати своєчасне виявлення атакувальної активності, коректну класифікацію типів атак і відтворюваність процедур оцінювання ефективності моделей.

Узагальнення результатів бінарної та багатокласової класифікації дозволяє зробити принциповий висновок про дворівневу організацію розробленої інформаційної технології. На першому рівні вона повинна забезпечувати надійне відокремлення нормального трафіку від атакувального, і для цього найкращим виявився метод опорних векторів у конфігурації svm2. На другому рівні вона має деталізувати виявлену загрозу за типом атаки, і тут найкращі результати показали knn2 та rf2, які можуть розглядатися як два ефективні варіанти реалізації механізму багатокласового розпізнавання. Запропонована двоетапна схема функціонування є логічно обґрунтованою і практично доцільною, оскільки дає змогу поєднати високу чутливість до атак на етапі первинного виявлення з високою точністю деталізації інциденту на етапі подальшої класифікації.

Результати експериментального дослідження свідчать, що розроблена інформаційна технологія інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак є ефективною в обох режимах функціонування. У бінарній постановці вона забезпечує високоточне виявлення атак, а в багатокласовій – якісне розпізнавання їх типів. Сукупність одержаних результатів підтверджує, що запропонована інформаційна технологія може бути використана як основа для побудови інтелектуальних компонентів систем виявлення атак у комп’ютерних мережах [140, 144, 150].

Висновки до розділу 4

У четвертому розділі проведено експериментальне дослідження ефективності розробленої інформаційної технології інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак у бінарній та багатокласовій постановках задачі. Це дало змогу оцінити її функціональні можливості як на рівні первинного виявлення атакувальної активності, так і на рівні подальшого розпізнавання типів атак.

У процесі дослідження сформовано підготовлений ознаковий простір та побудовано вибірки для обох режимів функціонування. Для багатокласового експерименту сформовано збалансований датафрейм blnc_data, у якому сім класів мережевого трафіку представлені рівномірно, що забезпечило коректні умови для навчання моделей і порівняння результатів класифікації.

У бінарній постановці найкращі результати отримано для моделі методу опорних векторів svm2, для якої accuracy = 0,9627, середнє значення 5-fold cross-validation = 0,9584, ROC-AUC = 99,42 %. Порівняно з логістичною регресією ця модель забезпечила кращу структуру класифікаційних помилок і меншу кількість хибнонегативних рішень, що є принципово важливим для систем виявлення атак.

У багатокласовій постановці найвищу підсумкову точність класифікації продемонструвала модель knn2, для якої accuracy = 0,9809. Водночас найвищу узагальнювальну здатність показала модель випадкового лісу rf2, для якої середнє значення cross-validation = 0,9818. Модель дерева рішень dt2 також забезпечила високі результати, однак поступилася моделям knn2 і rf2 за інтегральними показниками якості.

Порівняльний аналіз показав, що для бінарного режиму функціонування інформаційної технології доцільним є використання моделі svm2, а для багатокласового режиму – моделей knn2 і rf2 залежно від пріоритетів застосування: максимальної точності класифікації або вищої стійкості результатів.

Важливою характеристикою розробленої інформаційної технології є її модульна структура. Такий принцип побудови забезпечує можливість розширення

технології шляхом інтеграції нових модулів з іншими моделями машинного навчання, методами підготовки даних і засобами оцінювання в обох режимах функціонування. Модульна організація інформаційної технології забезпечує гнучкість, масштабованість, спрощує її модернізацію та підвищує адаптивність до появи нових типів атак і розвитку методів інтелектуального аналізу даних.

ВИСНОВКИ

У дисертаційній роботі розв’язано актуальну науково-прикладну задачу підвищення ефективності виявлення атак і розпізнавання їх типів у комп’ютерних мережах шляхом розроблення інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак. Мету дослідження досягнуто завдяки розробленню та експериментальному дослідженню інформаційної технології, що базується на методах машинного навчання і забезпечує функціонування в режимах бінарного виявлення атак та багатокласового розпізнавання їх типів.

1. Проведено аналіз сучасних методів, моделей і засобів моніторингу мережевого трафіку та виявлення атак, класифікації систем IDS/IPS, а також методів машинного навчання, що застосовуються в задачах інтелектуального аналізу мережевого трафіку. Встановлено, що традиційні сигнатурні методи забезпечують ефективне виявлення переважно відомих загроз, але мають обмеження щодо нових, модифікованих і малопоширених атак. Показано, що ефективність інтелектуального виявлення атак визначається не лише вибором моделі машинного навчання, а й якістю підготовки даних, коректністю формування ознакового простору та обґрунтованістю системи оцінювання результатів.

2. Обґрунтовано вибір набору даних CIC-IDS2017 як репрезентативної основи для проведення відтворюваного експериментального дослідження. Визначено вимоги до підготовки даних мережевого трафіку для задач інтелектуального виявлення атак, які охоплюють очищення, уніфікацію структури ознак, усунення технічних дефектів, стандартизацію числових характеристик, зменшення надлишковості ознакового простору, балансування вибірок та недопущення витоку інформації під час побудови та оцінювання моделей. Обґрунтовано, що саме така послідовність підготовки даних створює ефективну основу для побудови інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для СВА.

3. Розроблено метод підготовки даних, класифікації та оцінювання мережевого трафіку для систем виявлення атак, який інтегрує процедури очищення, уніфікації та стандартизації даних, обробки пропущених і нескінченних значень, аналізу ознак і зниження розмірності ознакового простору, формування та балансування вибірок, навчання моделей машинного навчання й комплексного оцінювання їх якості. На відміну від підходів, орієнтованих лише на окремі етапи попередньої обробки або класифікації, запропонований метод забезпечує цілісну послідовність підготовки, моделювання та оцінювання мережевого трафіку, що дає змогу підвищити ефективність бінарного виявлення атак і багатокласового розпізнавання їх типів.

4. Сформовано вибірки для двох взаємопов’язаних постановок задачі: бінарного виявлення атак і багатокласового розпізнавання їх типів. Для багатокласової постановки реалізовано балансування даних, що дало змогу сформувати збалансоване подання класів мережевого трафіку та забезпечити коректні умови для навчання моделей і порівняння результатів класифікації в межах єдиного дослідницького конвеєра.

5. Розроблено архітектуру інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для СВА, яка об’єднує конвеєр обробки мережевого трафіку, процедури формування ознакового простору, модулі машинного навчання та двоетапну схему класифікації для бінарного виявлення атак і багатокласового розпізнавання їх типів. Запропонована архітектура забезпечує узгоджене виконання основних етапів підготовки, аналізу, класифікації та оцінювання трафіку, що створює основу для програмної реалізації інформаційної технології та її використання в системах виявлення атак.

6. Розроблено програмну реалізацію інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для СВА у вигляді цілісного програмного модуля, у межах якого реалізовано повний цикл обробки мережевого трафіку – від завантаження та інтеграції даних до побудови моделей машинного навчання, оцінювання та візуалізації результатів класифікації. Особливістю розробленої інформаційної технології є підтримка двох режимів

функціонування в межах єдиного програмного модуля та забезпечення відтворюваності всіх етапів обробки, аналізу й класифікації мережевого трафіку.

7. Реалізовано та досліджено моделі машинного навчання для аналізу мережевого трафіку: у бінарній постановці – логістичну регресію та метод опорних векторів, у багатокласовій – випадковий ліс, дерево рішень і метод k-найближчих сусідів. Запропонована організація програмної реалізації забезпечила уніфіковане навчання, оцінювання та порівняння моделей у межах єдиного програмного середовища.

8. Проведено експериментальне дослідження ефективності розробленої інформаційної технології на основі набору даних CIC-IDS2017 у двох постановках задачі: бінарному виявленні атак і багатокласовому розпізнаванні їх типів. Для оцінювання моделей використано фіксований поділ вибірок на навчальну та тестову частини, п’ятикратну перехресну перевірку та систему взаємодоповнювальних метрик. Встановлено, що в бінарній постановці ефективною є конфігурація методу опорних векторів svm2, для якої accuracy = 0,9627, середнє значення 5-fold cross-validation = 0,9584, ROC-AUC = 99,42 %. У багатокласовій постановці найвищу точність продемонструвала модель knn2, для якої accuracy = 0,9809, тоді як найкращу узагальнювальну здатність показала модель випадкового лісу rf2, для якої середнє значення cross-validation = 0,9818. Одержані результати підтверджують ефективність запропонованої інформаційної технології та її придатність для використання в системах виявлення атак у комп’ютерних мережах.

9. Оцінено практичну придатність розробленої інформаційної технології для використання як основи побудови інтелектуальних компонентів сучасних систем виявлення атак у комп’ютерних мережах. Розроблена інформаційна технологія забезпечує моніторинг мережевого трафіку, своєчасне виявлення атакувальної активності та подальшу деталізацію інцидентів за типами атак, що підвищує ефективність підтримки прийняття рішень у сфері кібербезпеки. Результати дисертаційної роботи впроваджено в діяльність ТОВ «Центум-Д» та в

освітній процес Національного технічного університету «Дніпровська політехніка».

Практичне значення одержаних результатів полягає в тому, що розроблена програмна реалізація інформаційної технологія може бути використана як основа для побудови інтелектуальних компонентів сучасних систем виявлення атак у комп’ютерних мережах. Її застосування забезпечує моніторинг трафіку комп’ютерної мережі, своєчасне виявлення атакувальної активності та подальшу деталізацію інцидентів за типами атак, що підвищує ефективність підтримки прийняття рішень у сфері кібербезпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Scarfone K., Mell P. Guide to Intrusion Detection and Prevention Systems (IDPS). NIST Special Publication 800–94. Gaithersburg, MD : NIST, 2007. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800–94.pdf (дата звернення: 21.01.2026).

2. Paxson V. Bro: A System for Detecting Network Intruders in Real–Time. Computer Networks. 1999. Vol. 31, No. 23–24. P. 2435–2463. DOI: https://doi.org/10.1016/S1389–1286(99)00112–7.

3. Zeek Project. Zeek Documentation (Book of Zeek) [Електронний ресурс]. URL: https://docs.zeek.org/ (дата звернення: 21.01.2026).

4. Zeek Project. About Zeek [Електронний ресурс]. URL: https://docs.zeek.org/en/lts/about.html (дата звернення: 21.01.2026).

5. Sharafaldin I., Habibi Lashkari A., Ghorbani A. A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. Proc. 4th Int. Conf. on Information Systems Security and Privacy (ICISSP 2018). 2018. P. 108–116. DOI: https://doi.org/10.5220/0006639801080116.

6. Tavallaee M., Bagheri E., Lu W., Ghorbani A. A. A detailed analysis of the KDD CUP 99 data set. Proc. 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA). 2009. DOI: https://doi.org/10.1109/CISDA.2009.5356528.

7. Moustafa N., Slay J. UNSW–NB15: A Comprehensive Data Set for Network Intrusion Detection Systems. Proc. 2015 Military Communications and Information Systems Conference (MilCIS). 2015. DOI: https://doi.org/10.1109/MilCIS.2015.7348942.

8. Buczak A. L., Guven E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications Surveys & Tutorials. 2016. Vol. 18, No. 2. P. 1153–1176. DOI: https://doi.org/10.1109/COMST.2015.2494502.

9. Sommer R., Paxson V. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. Proc. 2010 IEEE Symposium on Security and Privacy.

2010. P. 305–316. DOI: https://doi.org/10.1109/SP.2010.25.

10. Breiman L. Random Forests. Machine Learning. 2001. Vol. 45, No. 1. P. 5–32. DOI: https://doi.org/10.1023/A:1010933404324.

11. Cover T. M., Hart P. E. Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory. 1967. Vol. 13, No. 1. P. 21–27. DOI: https://doi.org/10.1109/TIT.1967.1053964.

12. Cortes C., Vapnik V. Support–vector networks. Machine Learning. 1995. Vol. 20. P. 273–297. DOI: https://doi.org/10.1023/A:1022627411411.

13. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System. Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '16). 2016. P. 785–794. DOI: https://doi.org/10.1145/2939672.2939785.

14. Chawla N. V., Bowyer K. W., Hall L. O., Kegelmeyer W. P. SMOTE: Synthetic Minority Over–sampling Technique. Journal of Artificial Intelligence Research. 2002. Vol. 16. P. 321–357. DOI: https://doi.org/10.1613/jair.953.

15. Canadian Institute for Cybersecurity (CIC), University of New Brunswick. CSE–CIC–IDS2018 Dataset [Електронний ресурс]. URL: https://www.unb.ca/cic/datasets/ids–2018.html (дата звернення: 21.01.2026).

16. Göcs L., Johanyák Z. C. Identifying Relevant Features of CSE–CIC–IDS2018 Dataset for the Development of an Intrusion Detection System [Електронний ресурс]. arXiv, 2023. DOI: https://doi.org/10.48550/arXiv.2307.11544. URL: https://arxiv.org/abs/2307.11544 (дата звернення: 21.01.2026).

17. Jolliffe I. T. Principal Component Analysis. 2nd ed. New York : Springer, 2002. DOI: https://doi.org/10.1007/b98835.

18. Fawcett T. An Introduction to ROC Analysis. Pattern Recognition Letters. 2006. Vol. 27, No. 8. P. 861–874. DOI: https://doi.org/10.1016/j.patrec.2005.10.010.

19. Saito T., Rehmsmeier M. The Precision–Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. PLOS ONE. 2015. Vol. 10, No. 3. DOI: https://doi.org/10.1371/journal.pone.0118432.

20. Scikit–learn developers. StandardScaler (API reference) [Електронний ресурс]. URL: https://scikit–learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (дата

звернення: 21.01.2026).

21. Imbalanced–learn developers. SMOTE (API reference) [Електронний ресурс]. URL: https://imbalanced–learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html (дата звернення: 21.01.2026).

22. Denning D. E. An Intrusion–Detection Model. IEEE Transactions on Software Engineering. 1987. Vol. SE–13, No. 2. P. 222–232. DOI: https://doi.org/10.1109/TSE.1987.232894.

23. Debar H., Dacier M., Wespi A. Towards a Taxonomy of Intrusion–Detection Systems. Computer Networks. 1999. Vol. 31, No. 8. P. 805–822. DOI: https://doi.org/10.1016/S1389–1286(98)00017–6.

24. Liao H.–J., Lin C.–H. R., Lin Y.–C., Tung K.–Y. Intrusion Detection System: A Comprehensive Review. Journal of Network and Computer Applications. 2013. Vol. 36, No. 1. P. 16–24. DOI: https://doi.org/10.1016/j.jnca.2012.09.004.

25. Axelsson S. The Base–Rate Fallacy and the Difficulty of Intrusion Detection. ACM Transactions on Information and System Security. 2000. Vol. 3, No. 3. P. 186–205. DOI: https://doi.org/10.1145/357830.357849.

26. Claise B. et al. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011. 2013. DOI: https://doi.org/10.17487/RFC7011. URL: https://www.rfc–editor.org/rfc/rfc7011.html (дата звернення: 21.01.2026).

27. Claise B. Cisco Systems NetFlow Services Export Version 9. RFC 3954. 2004. DOI: https://doi.org/10.17487/RFC3954. URL: https://www.rfc–editor.org/rfc/rfc3954.html (дата звернення: 21.01.2026).

28. García–Teodoro P., Díaz–Verdejo J., Maciá–Fernández G., Vázquez E. Anomaly–Based Network Intrusion Detection: Techniques, Systems and Challenges. Computers & Security. 2009. Vol. 28, No. 1–2. P. 18–28. DOI: https://doi.org/10.1016/j.cose.2008.08.003.

29. Roesch M. Snort: Lightweight Intrusion Detection for Networks. Proc. 13th USENIX Conf. on System Administration (LISA '99). 1999. URL: https://www.usenix.org/legacy/publications/library/proceedings/lisa99/full_papers/roesc

h/roesch.pdf (дата звернення: 21.01.2026).

30. Snort Project. Snort Documentation [Електронний ресурс]. URL: https://www.snort.org/documents (дата звернення: 21.01.2026).

31. Open Information Security Foundation (OISF). Suricata Documentation: IPS Mode [Електронний ресурс]. URL: https://docs.suricata.io/en/latest/ips/index.html (дата звернення: 21.01.2026).

32. Wazuh. Wazuh Documentation [Електронний ресурс]. URL: https://documentation.wazuh.com/current/user–manual/index.html (дата звернення: 21.01.2026).

33. Zeek Project. Zeek – The Network Security Monitor [Електронний ресурс]. URL: https://zeek.org/ (дата звернення: 21.01.2026).

34. Gerhards R. The Syslog Protocol. RFC 5424. 2009. DOI: https://doi.org/10.17487/RFC5424. URL: https://www.rfc–editor.org/rfc/rfc5424.html (дата звернення: 21.01.2026).

35. Kent K., Souppaya M. Guide to Computer Security Log Management. NIST Special Publication 800–92. 2006. DOI: https://doi.org/10.6028/NIST.SP.800–92. URL: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800–92.pdf (дата звернення: 21.01.2026).

36. NIST. Cybersecurity Log Management Planning Guide (SP 800–92r1 ipd) [Електронний ресурс]. 2023. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800–92r1.ipd.pdf (дата звернення: 21.01.2026).

37. MITRE. MITRE ATT&CK [Електронний ресурс]. URL: https://attack.mitre.org/ (дата звернення: 21.01.2026).

38. ENISA. ENISA Threat Landscape 2025 [Електронний ресурс]. URL: https://www.enisa.europa.eu/publications/enisa–threat–landscape–2025 (дата звернення: 21.01.2026).

39. Verizon. 2025 Data Breach Investigations Report (DBIR) [Електронний ресурс]. URL: https://www.verizon.com/business/resources/reports/dbir/ (дата звернення: 21.01.2026).

40. CISA. Understanding and Responding to Distributed Denial–of–Service

(DDoS) Attacks [Електронний ресурс]. URL: https://www.cisa.gov/resources–tools/resources/understanding–and–responding–distributed–denial–service–attacks (дата звернення: 21.01.2026).

41. NIST. Technical Guide to Information Security Testing and Assessment (SP 800–115) [Електронний ресурс]. URL: https://csrc.nist.gov/publications/detail/sp/800–115/final (дата звернення: 21.01.2026).

42. OWASP. OWASP Top 10:2025 [Електронний ресурс]. URL: https://owasp.org/www–project–top–ten/ (дата звернення: 21.01.2026).

43. Born K., Gustafson D. Detecting DNS Tunnels Using Character Frequency Analysis [Електронний ресурс]. arXiv, 2010. URL: https://arxiv.org/abs/1004.4358 (дата звернення: 21.01.2026).

44. Yassine S., Khalife J., Chamoun M., el Ghor H. A Survey of DNS Tunnelling Detection Techniques Using Machine Learning. Proceedings of the 1st International Conference on Big Data and Cyber–Security Intelligence (BDCSIntell 2018), Hadath, Lebanon, December 13–15, 2018. CEUR Workshop Proceedings. 2019. Vol. 2343. P. 63–66 [Електронний ресурс]. URL: https://ceur–ws.org/Vol–2343/paper13.pdf (дата звернення: 21.01.2026).

45. Canadian Institute for Cybersecurity. Intrusion Detection Evaluation Dataset (CIC-IDS2017) [Електронний ресурс]. University of New Brunswick. URL: https://www.unb.ca/cic/datasets/ids-2017.html (дата звернення: 07.05.2026).

46. Pedregosa F. Scikit-learn: Machine Learning in Python / F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel та ін. // Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830. URL: https://www.jmlr.org/papers/v12/pedregosa11a.html (дата звернення: 07.05.2026).

47. Scikit-learn developers. IncrementalPCA [Електронний ресурс] // scikit-learn documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.IncrementalPCA.html (дата звернення: 07.05.2026).

48. Hastie T. The Elements of Statistical Learning: Data Mining, Inference, and Prediction / T. Hastie, R. Tibshirani, J. Friedman. 2nd ed. New York : Springer, 2009. DOI: https://doi.org/10.1007/978-0-387-84858-7. URL:

https://link.springer.com/book/10.1007/978-0-387-84858-7 (дата звернення: 07.05.2026).

49. Breiman L. Classification and Regression Trees / L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone. Boca Raton : Chapman and Hall/CRC, 1984. DOI: https://doi.org/10.1201/9781315139470. URL: https://www.taylorfrancis.com/books/mono/10.1201/9781315139470/classification-regression-trees-leo-breiman-jerome-friedman-olshen-charles-stone (дата звернення: 07.05.2026).

50. Kohavi R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection / R. Kohavi // Proceedings of the 14th International Joint Conference on Artificial Intelligence. 1995. Vol. 2. P. 1137–1143. URL: https://www.ijcai.org/Proceedings/95-2/Papers/016.pdf (дата звернення: 07.05.2026).

51. Scikit-learn developers. Model selection and evaluation [Електронний ресурс] // scikit-learn documentation. URL: https://scikit-learn.org/stable/model_selection.html (дата звернення: 07.05.2026).

52. National Institute of Standards and Technology. The NIST Cybersecurity Framework (CSF) 2.0 [Електронний ресурс]. NIST Cybersecurity White Paper 29. Gaithersburg : NIST, 2024. DOI: https://doi.org/10.6028/NIST.CSWP.29. URL: https://csrc.nist.gov/pubs/cswp/29/the-nist-cybersecurity-framework-csf-20/final (дата звернення: 07.05.2026).

53. Dempsey K. Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations / K. Dempsey, N. S. Chawla, A. Johnson, R. Johnston, A. C. Jones, A. Orebaugh, M. Scholl, K. Stine [Електронний ресурс]. NIST Special Publication 800-137. Gaithersburg: NIST, 2011. DOI: https://doi.org/10.6028/NIST.SP.800-137. URL: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-137.pdf (дата звернення: 07.05.2026).

54. CISA. Known Exploited Vulnerabilities Catalog [Електронний ресурс]. Cybersecurity and Infrastructure Security Agency. URL: https://www.cisa.gov/known-exploited-vulnerabilities-catalog (дата звернення: 07.05.2026).

55. Ring M. A Survey of Network-based Intrusion Detection Data Sets / M. Ring,

S. Wunderlich, D. Scheuring, D. Landes, A. Hotho // Computers & Security. 2019. Vol. 86. P. 147–167. DOI: https://doi.org/10.1016/j.cose.2019.06.005. URL: https://www.sciencedirect.com/science/article/abs/pii/S016740481930118X (дата звернення: 07.05.2026).

56. Thakkar A. A Review of the Advancement in Intrusion Detection Datasets / A. Thakkar, R. Lohiya // Procedia Computer Science. 2020. Vol. 167. P. 636–645. DOI: https://doi.org/10.1016/j.procs.2020.03.330. URL: https://www.sciencedirect.com/science/article/pii/S1877050920307961 (дата звернення: 07.05.2026).

57. Yang Z. A systematic literature review of methods and datasets for anomaly-based network intrusion detection / Z. Yang, X. Liu, T. Li, D. Wu, J. Wang, Y. Zhao, H. Han // Computers & Security. 2022. Vol. 116. Article 102675. DOI: https://doi.org/10.1016/j.cose.2022.102675. URL: https://www.sciencedirect.com/science/article/pii/S0167404822000736 (дата звернення: 07.05.2026).

58. Chou D. A Survey on Data-driven Network Intrusion Detection / D. Chou, M. Jiang // ACM Computing Surveys. 2021. DOI: https://doi.org/10.1145/3472753. URL: https://dl.acm.org/doi/10.1145/3472753 (дата звернення: 07.05.2026).

59. Goldschmidt P. Network intrusion datasets: A survey, limitations, and recommendations / P. Goldschmidt, D. Chudá // Computers & Security. 2025. Vol. 156. Article 104510. DOI: https://doi.org/10.1016/j.cose.2025.104510. URL: https://www.sciencedirect.com/science/article/abs/pii/S0167404825001993 (дата звернення: 07.05.2026).

60. Hozouri A. A comprehensive survey on intrusion detection systems with advances in machine learning, deep learning and emerging cybersecurity challenges / A. Hozouri, A. Mirzaei, M. Effatparvar // Discover Artificial Intelligence. 2025. Vol. 5. Article 314. DOI: https://doi.org/10.1007/s44163-025-00578-1. URL: https://link.springer.com/article/10.1007/s44163-025-00578-1 (дата звернення: 07.05.2026).

61. Catillo M. Machine Learning on Public Intrusion Datasets: Academic Hype or Concrete Advances in NIDS? / M. Catillo, A. Pecchia, U. Villano // 2023 53rd Annual

IEEE/IFIP International Conference on Dependable Systems and Networks – Supplemental Volume (DSN-S). 2023. P. 132–136. DOI: https://doi.org/10.1109/DSN-S58398.2023.00038. URL: https://ieeexplore.ieee.org/document/10190403 (дата звернення: 07.05.2026).

62. Python Software Foundation. Python 3 Documentation [Електронний ресурс]. URL: https://docs.python.org/3/ (дата звернення: 19.05.2026).

63. Python Software Foundation. The Python Language Reference [Електронний ресурс]. URL: https://docs.python.org/3/reference/ (дата звернення: 19.05.2026).

64. Python Software Foundation. The Python Standard Library [Електронний ресурс]. URL: https://docs.python.org/3/library/ (дата звернення: 19.05.2026).

65. VanderPlas J. Python Data Science Handbook: Essential Tools for Working with Data. Sebastopol: O'Reilly Media, 2016. URL: https://jakevdp.github.io/PythonDataScienceHandbook/ (дата звернення: 19.05.2026).

66. Harris C. R. Array programming with NumPy / C. R. Harris, K. J. Millman, S. J. van der Walt та ін. // Nature. 2020. Vol. 585. P. 357–362. DOI: https://doi.org/10.1038/s41586-020-2649-2.

67. Van der Walt S. The NumPy Array: A Structure for Efficient Numerical Computation / S. van der Walt, S. C. Colbert, G. Varoquaux // Computing in Science & Engineering. 2011. Vol. 13, No. 2. P. 22–30. DOI: https://doi.org/10.1109/MCSE.2011.37.

68. NumPy Developers. numpy.isinf [Електронний ресурс] // NumPy Documentation. URL: https://numpy.org/doc/stable/reference/generated/numpy.isinf.html (дата звернення: 19.05.2026).

69. NumPy Developers. Constants: numpy.inf, numpy.nan [Електронний ресурс] // NumPy Documentation. URL: https://numpy.org/doc/stable/reference/constants.html (дата звернення: 19.05.2026).

70. McKinney W. Data Structures for Statistical Computing in Python // Proceedings of the 9th Python in Science Conference. 2010. P. 56–61. DOI: https://doi.org/10.25080/Majora-92bf1922-00a.

71. Pandas Development Team. IO tools: pandas.read_csv [Електронний ресурс]

// pandas Documentation. URL: https://pandas.pydata.org/docs/user_guide/io.html (дата звернення: 19.05.2026).

72. Pandas Development Team. pandas.concat [Електронний ресурс] // pandas Documentation. URL: https://pandas.pydata.org/docs/reference/api/pandas.concat.html (дата звернення: 19.05.2026).

73. Pandas Development Team. pandas.DataFrame.drop_duplicates [Електронний ресурс] // pandas Documentation. URL: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html (дата звернення: 19.05.2026).

74. Pandas Development Team. Working with missing data [Електронний ресурс] // pandas Documentation. URL: https://pandas.pydata.org/docs/user_guide/missing_data.html (дата звернення: 19.05.2026).

75. Bilogur A. Missingno: a missing data visualization suite // Journal of Open Source Software. 2018. Vol. 3, No. 22. Article 547. DOI: https://doi.org/10.21105/joss.00547.

76. Hunter J. D. Matplotlib: A 2D Graphics Environment // Computing in Science & Engineering. 2007. Vol. 9, No. 3. P. 90–95. DOI: https://doi.org/10.1109/MCSE.2007.55.

77. Matplotlib Developers. Pyplot tutorial [Електронний ресурс] // Matplotlib Documentation. URL: https://matplotlib.org/stable/tutorials/pyplot.html (дата звернення: 19.05.2026).

78. Matplotlib Developers. Annotated heatmap [Електронний ресурс] // Matplotlib Documentation. URL: https://matplotlib.org/stable/gallery/images_contours_and_fields/image_annotated_heatmap.html (дата звернення: 19.05.2026).

79. Waskom M. L. seaborn: statistical data visualization // Journal of Open Source Software. 2021. Vol. 6, No. 60. Article 3021. DOI: https://doi.org/10.21105/joss.03021.

80. Seaborn Developers. seaborn.heatmap [Електронний ресурс] // Seaborn Documentation. URL: https://seaborn.pydata.org/generated/seaborn.heatmap.html (дата звернення: 19.05.2026).

81. Seaborn Developers. seaborn.barplot [Електронний ресурс] // Seaborn Documentation. URL: https://seaborn.pydata.org/generated/seaborn.barplot.html (дата звернення: 19.05.2026).

82. Lemaître G. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning / G. Lemaître, F. Nogueira, C. K. Aridas // Journal of Machine Learning Research. 2017. Vol. 18, No. 17. P. 1–5. URL: https://www.jmlr.org/papers/v18/16-365.html (дата звернення: 19.05.2026).

83. Imbalanced-learn Developers. Imbalanced-learn documentation [Електронний ресурс]. URL: https://imbalanced-learn.org/stable/ (дата звернення: 19.05.2026).

84. Kubat M. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection / M. Kubat, S. Matwin // Proceedings of the 14th International Conference on Machine Learning. 1997. P. 179–186.

85. He H. Learning from Imbalanced Data / H. He, E. A. Garcia // IEEE Transactions on Knowledge and Data Engineering. 2009. Vol. 21, No. 9. P. 1263–1284. DOI: https://doi.org/10.1109/TKDE.2008.239.

86. Fernández A. Learning from Imbalanced Data Sets / A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, F. Herrera. Cham : Springer, 2018. DOI: https://doi.org/10.1007/978-3-319-98074-4.

87. Buitinck L. API design for machine learning software: experiences from the scikit-learn project / L. Buitinck, G. Louppe, M. Blondel та ін. // ECML PKDD Workshop: Languages for Data Mining and Machine Learning. 2013. P. 108–122. URL: https://arxiv.org/abs/1309.0238 (дата звернення: 19.05.2026).

88. Scikit-learn Developers. Preprocessing data [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-learn.org/stable/modules/preprocessing.html (дата звернення: 19.05.2026).

89. Scikit-learn Developers. LabelEncoder [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html (дата звернення: 19.05.2026).

90. Scikit-learn Developers. train_test_split [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-

learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html (дата звернення: 19.05.2026).

91. Scikit-learn Developers. cross_val_score [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html (дата звернення: 19.05.2026).

92. Scikit-learn Developers. LogisticRegression [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (дата звернення: 19.05.2026).

93. Scikit-learn Developers. SVC [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (дата звернення: 19.05.2026).

94. Scikit-learn Developers. RandomForestClassifier [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (дата звернення: 19.05.2026).

95. Scikit-learn Developers. DecisionTreeClassifier [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (дата звернення: 19.05.2026).

96. Scikit-learn Developers. KNeighborsClassifier [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html (дата звернення: 19.05.2026).

97. Scikit-learn Developers. classification_report [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html (дата звернення: 19.05.2026).

98. Scikit-learn Developers. confusion_matrix [Електронний ресурс] // scikit-

learn Documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html (дата звернення: 19.05.2026).

99. Scikit-learn Developers. accuracy_score [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html (дата звернення: 19.05.2026).

100. Scikit-learn Developers. roc_auc_score [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html (дата звернення: 19.05.2026).

101. Scikit-learn Developers. roc_curve [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html (дата звернення: 19.05.2026).

102. Scikit-learn Developers. precision_recall_curve [Електронний ресурс] // scikit-learn Documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_curve.html (дата звернення: 19.05.2026).

103. Powers D. M. W. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation // Journal of Machine Learning Technologies. 2011. Vol. 2, No. 1. P. 37–63.

104. Davis J. The Relationship Between Precision-Recall and ROC Curves / J. Davis, M. Goadrich // Proceedings of the 23rd International Conference on Machine Learning. 2006. P. 233–240. DOI: https://doi.org/10.1145/1143844.1143874.

105. Gama J. A Survey on Concept Drift Adaptation / J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia // ACM Computing Surveys. 2014. Vol. 46, No. 4. Article 44. DOI: https://doi.org/10.1145/2523813.

106. Sculley D. Hidden Technical Debt in Machine Learning Systems / D. Sculley, G. Holt, D. Golovin та ін. // Advances in Neural Information Processing Systems. 2015. P. 2503–2511. URL: https://papers.nips.cc/paper/5656-hidden-technical-debt-in-

machine-learning-systems (дата звернення: 19.05.2026).

107. Bishop C. M. Pattern Recognition and Machine Learning. New York : Springer, 2006. 738 p.

108. Murphy K. P. Machine Learning: A Probabilistic Perspective. Cambridge : MIT Press, 2012. 1104 p.

109. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. URL: https://www.deeplearningbook.org/ (дата звернення: 19.05.2026).

110. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 3rd ed. Sebastopol: O'Reilly Media, 2022. 856 p.

111. Raschka S., Mirjalili V. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2. 3rd ed. Birmingham : Packt Publishing, 2019. 770 p.

112. Aggarwal C. C. Data Mining: The Textbook. Cham: Springer, 2015. DOI: https://doi.org/10.1007/978-3-319-14142-8.

113. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. 3rd ed. Waltham: Morgan Kaufmann, 2011. 744 p.

114. Witten I. H., Frank E., Hall M. A., Pal C. J. Data Mining: Practical Machine Learning Tools and Techniques. 4th ed. Cambridge: Morgan Kaufmann, 2016. 654 p.

115. Kelleher J. D., Namee B. M., D'Arcy A. Fundamentals of Machine Learning for Predictive Data Analytics. 2nd ed. Cambridge : MIT Press, 2020. 856 p.

116. Kuhn M., Johnson K. Applied Predictive Modeling. New York : Springer, 2013. DOI: https://doi.org/10.1007/978-1-4614-6849-3.

117. Kuhn M., Johnson K. Feature Engineering and Selection: A Practical Approach for Predictive Models. Boca Raton : CRC Press, 2019. 266 p.

118. Provost F., Fawcett T. Data Science for Business. Sebastopol : O'Reilly Media, 2013. 414 p.

119. Domingos P. A Few Useful Things to Know about Machine Learning // Communications of the ACM. 2012. Vol. 55, No. 10. P. 78–87. DOI: https://doi.org/10.1145/2347736.2347755.

120. Wolpert D. H. The Lack of A Priori Distinctions Between Learning Algorithms // Neural Computation. 1996. Vol. 8, No. 7. P. 1341–1390. DOI:

https://doi.org/10.1162/neco.1996.8.7.1341.

121. Dietterich T. G. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms // Neural Computation. 1998. Vol. 10, No. 7. P. 1895–1923. DOI: https://doi.org/10.1162/089976698300017197.

122. Demšar J. Statistical Comparisons of Classifiers over Multiple Data Sets // Journal of Machine Learning Research. 2006. Vol. 7. P. 1–30. URL: https://www.jmlr.org/papers/v7/demsar06a.html (дата звернення: 19.05.2026).

123. Bradley A. P. The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms // Pattern Recognition. 1997. Vol. 30, No. 7. P. 1145–1159. DOI: https://doi.org/10.1016/S0031-3203(96)00142-2.

124. Japkowicz N., Shah M. Evaluating Learning Algorithms: A Classification Perspective. Cambridge : Cambridge University Press, 2011. DOI: https://doi.org/10.1017/CBO9780511921803.

125. Japkowicz N., Stephen S. The Class Imbalance Problem: A Systematic Study // Intelligent Data Analysis. 2002. Vol. 6, No. 5. P. 429–449. DOI: https://doi.org/10.3233/IDA-2002-6504.

126. Galar M., Fernandez A., Barrenechea E., Bustince H., Herrera F. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches // IEEE Transactions on Systems, Man, and Cybernetics, Part C. 2012. Vol. 42, No. 4. P. 463–484. DOI: https://doi.org/10.1109/TSMCC.2011.2161285.

127. Branco P., Torgo L., Ribeiro R. P. A Survey of Predictive Modeling on Imbalanced Domains // ACM Computing Surveys. 2016. Vol. 49, No. 2. Article 31. DOI: https://doi.org/10.1145/2907070.

128. Batista G. E. A. P. A., Prati R. C., Monard M. C. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data // ACM SIGKDD Explorations Newsletter. 2004. Vol. 6, No. 1. P. 20–29. DOI: https://doi.org/10.1145/1007730.1007735.

129. He H., Bai Y., Garcia E. A., Li S. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning // IEEE International Joint Conference on Neural Networks. 2008. P. 1322–1328. DOI: https://doi.org/10.1109/IJCNN.2008.4633969.

130. López V., Fernández A., García S., Palade V., Herrera F. An Insight into

Classification with Imbalanced Data: Empirical Results and Current Trends on Using Data Intrinsic Characteristics // Information Sciences. 2013. Vol. 250. P. 113–141. DOI: https://doi.org/10.1016/j.ins.2013.07.007.

131. Snoek J., Larochelle H., Adams R. P. Practical Bayesian Optimization of Machine Learning Algorithms // Advances in Neural Information Processing Systems. 2012. Vol. 25. URL: https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html (дата звернення: 19.05.2026).

132. Bergstra J., Bengio Y. Random Search for Hyper-Parameter Optimization // Journal of Machine Learning Research. 2012. Vol. 13. P. 281–305. URL: https://www.jmlr.org/papers/v13/bergstra12a.html (дата звернення: 19.05.2026).

133. Bergstra J., Yamins D., Cox D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures // Proceedings of the 30th International Conference on Machine Learning. 2013. P. 115–123. URL: https://proceedings.mlr.press/v28/bergstra13.html (дата звернення: 19.05.2026).

134. Claesen M., De Moor B. Hyperparameter Search in Machine Learning // arXiv preprint. 2015. arXiv:1502.02127. URL: https://arxiv.org/abs/1502.02127 (дата звернення: 19.05.2026).

135. Zaharia M., Chen A., Davidson A. та ін. Accelerating the Machine Learning Lifecycle with MLflow // IEEE Data Engineering Bulletin. 2018. Vol. 41, No. 4. P. 39–45. URL: https://arxiv.org/abs/1808.03233 (дата звернення: 19.05.2026).

136. Baylor D., Breck E., Cheng H.-T. та ін. TFX: A TensorFlow-Based Production-Scale Machine Learning Platform // Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017. P. 1387–1395. DOI: https://doi.org/10.1145/3097983.3098021.

137. Breck E., Cai S., Nielsen E., Salib M., Sculley D. The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction // IEEE International Conference on Big Data. 2017. P. 1123–1132. DOI: https://doi.org/10.1109/BigData.2017.8258038.

138. Amershi S., Begel A., Bird C. та ін. Software Engineering for Machine

Learning: A Case Study // IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice. 2019. P. 291–300. DOI: https://doi.org/10.1109/ICSE-SEIP.2019.00042.

139. Zhang Y., Ling C. X. A Strategy to Apply Machine Learning to Small Datasets in Materials Science // npj Computational Materials. 2018. Vol. 4. Article 25. DOI: https://doi.org/10.1038/s41524-018-0081-z.

140. Мєшков, В. Методи машинного навчання для бінарної та багатокласової класифікації мережевого трафіку у системах виявлення атак. Таврійський науковий вісник. Серія: Технічні науки. 2025, Т. 1, № 4. С. 182-196. DOI: https://doi.org/10.32782/tnv-tech.2025.4.1.20.

141. Мєшков, В. Метод попередньої обробки даних для створення інформаційних моделей в інтелектуальних системах виявлення атак. Information Technology: Computer Science, Software Engineering and Cyber Security, 2025, №2. С. 108-114, DOI: https://doi.org/10.32782/IT/2025-2-11.

142. Сафаров, О., Корнієнко, В., Горєв, В., Мєшков, В. Підвищення кібербезпеки електронних комунікаційних систем медичного призначення. // Information Technology: Computer Science, Software Engineering and Cyber Security, 2024, № 2, С. 128-133, DOI: https://doi.org/10.32782/IT/2024-2-16.

143. Мєшков, В. Аналіз систем інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак. // Information Technology: Computer Science, Software Engineering and Cyber Security, 2023, № 1, С. 85-92, DOI: https://doi.org/10.32782/IT/2023-1-11.

144. Мєшков В. І. Архітектура інформаційної технології інтелектуального моніторингу мережевого трафіку. // ITSec: Безпека інформаційних технологій: матеріали XIV Міжнар. наук.-техн. конф., м. Тернопіль, 22-24 трав. 2025 р. Тернопіль-Київ: ЗУНУ-ДУІКТ, 2025. 243с.

145. Мєшков В. І. Аналіз кореляційної матриці для показників набору даних CSE-CIC-IDS2017. // XII Міжнародна науково-технічна конференція студентів, аспірантів та молодих вчених «Молодь: наука та інновації», Дніпро, 13-15 листопада 2024 року (у 3-х томах) / НТУ «Дніпровська політехніка» – Дніпро: НТУ «ДП», 2024. Том 2. 291с.

146. V.I. Mieshkov, I. Mamuzić. Promising directions of computer network traffic monitoring for intrusion detection systems // 17th International Symposium of Croatian Metallurgical Society „Materials and Metallurgy“, SHMD ´2024. Materials And Metallurgy. Book Of Abstracts, Zagreb, Croatia: 2024, April, 18-19, С. 319.

147. Мєшков В. І. Аналіз наборів даних мережевого трафіку для систем виявлення атак // І (VII) міжнародна науково-практична конференція здобувачів вищої освіти і молодих учених «Інформаційні технології: теорія і практика», Дніпро, 20-22 березня 2024 року / НТУ «Дніпровська політехніка» – Дніпро: НТУ «ДП», 2024, С. 281-285.

148. Мєшков В. І. Сучасні системи моніторингу трафіку в комп’ютерній мережі // XIII Міжнародна науково-технічна конференція студентів, аспірантів та молодих вчених «Наукова весна», Дніпро, 1-3 березня 2023 року / НТУ «Дніпровська політехніка» – Дніпро: НТУ «ДП», 2023. С. 183-186.

149. Мєшков В. І., Корнієнко В. І. Ідентифікація та прогнозування трафіку комп’ютерної мережі для систем виявлення атак // XIII Всеукраїнська науково-практична конференція здобувачів вищої освіти і молодих учених «Молоді вчені 2023 - від теорії до практики»: Матеріали. Електронне видання. – Дніпро, Журфонд, 2023. С. 164-169.

150. Мєшков В. І., Корнієнко В. І. Розробка інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак // XIV Всеукраїнська науково-практична конференція «Актуальні проблеми управління інформаційною безпекою держави», – Київ, 30 березня 2023, / Національна академія СБУ, Матеріали конференції, 2023. С. 294-298.

ДОДАТОК А. СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковані основні наукові результати дисертації.

Публікації у фахових виданнях України:

1. Мєшков, В. (2025). Методи машинного навчання для бінарної та багатокласової класифікації мережевого трафіку у системах виявлення атак. Таврійський науковий вісник. Серія: Технічні науки. 2025, Т. 1, № 4. С. 182-196. DOI: https://doi.org/10.32782/tnv-tech.2025.4.1.20, URL: https://journals.ksauniv.ks.ua/index.php/tech/article/view/1064/977 (Фаховий (категорія Б)).

2. Мєшков, В. (2025). Метод попередньої обробки даних для створення інформаційних моделей в інтелектуальних системах виявлення атак. Information Technology: Computer Science, Software Engineering and Cyber Security, 2025, №2. С. 108-114, DOI: https://doi.org/10.32782/IT/2025-2-11, URL: https://journals.politehnica.dp.ua/index.php/it/article/view/825/742 (Фаховий (категорія Б)).

3. Сафаров, О., Корнієнко, В., Горєв, В., Мєшков, В. (2024). Підвищення кібербезпеки електронних комунікаційних систем медичного призначення. // Information Technology: Computer Science, Software Engineering and Cyber Security, 2024, № 2, С. 128-133, DOI: https://doi.org/10.32782/IT/2024-2-16, URL: https://journals.politehnica.dp.ua/index.php/it/article/view/595/527 (Фаховий (категорія Б)).

4. Мєшков, В. (2023). Аналіз систем інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак. // Information Technology: Computer Science, Software Engineering and Cyber Security, 2023, № 1, С. 85-92, DOI: https://doi.org/10.32782/IT/2023-1-11, URL: https://journals.politehnica.dp.ua/index.php/it/article/view/262/233 (Фаховий (категорія Б)).

Наукові праці, які засвідчують апробацію матеріалів дисертації.

Матеріали конференцій та тези доповідей:

1. Мєшков В. І. Архітектура інформаційної технології інтелектуального моніторингу мережевого трафіку. // ITSec: Безпека інформаційних технологій: матеріали XIV Міжнар. наук.-техн. конф., м. Тернопіль, 22-24 трав. 2025 р. Тернопіль-Київ: ЗУНУ-ДУІКТ, 2025. 243с.

2. Мєшков В. І. Аналіз кореляційної матриці для показників набору даних CSE-CIC-IDS2017. // XII Міжнародна науково-технічна конференція студентів, аспірантів та молодих вчених «Молодь: наука та інновації», Дніпро, 13-15 листопада 2024 року (у 3-х томах) / Національний технічний університет «Дніпровська політехніка» – Дніпро: НТУ «ДП», 2024. Том 2. 291с.

3. V.I. Mieshkov, I. Mamuzić. Promising directions of computer network traffic monitoring for intrusion detection systems // 17th International Symposium of Croatian Metallurgical Society „Materials and Metallurgy“, SHMD ´2024. Materials And Metallurgy. Book Of Abstracts, Zagreb, Croatia: 2024, April, 18-19, С. 319.

4. Мєшков В. І. Аналіз наборів даних мережевого трафіку для систем виявлення атак // I (VII) міжнародна науково-практична конференція здобувачів вищої освіти і молодих учених «Інформаційні технології: теорія і практика», Дніпро, 20-22 березня 2024 року / Національний технічний університет «Дніпровська політехніка» – Дніпро: НТУ «ДП», 2024, С. 281-285.

5. Мєшков В. І. Сучасні системи моніторингу трафіку в комп’ютерній мережі // XIII Міжнародна науково-технічна конференція студентів, аспірантів та молодих вчених «Наукова весна», Дніпро, 1-3 березня 2023 року / Національний технічний університет «Дніпровська політехніка» – Дніпро: НТУ «ДП», 2023. С. 183-186.

6. Мєшков В. І., Корнієнко В. І. Ідентифікація та прогнозування трафіку комп’ютерної мережі для систем виявлення атак // XIII Всеукраїнська науково-практична конференція здобувачів вищої освіти і молодих учених «Молоді вчені 2023 - від теорії до практики»: Матеріали. Електронне видання. – Дніпро, Журфонд, 2023. С. 164-169.

7. Мєшков В. І., Корнієнко В. І. Розробка інформаційної технології інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак // XIV Всеукраїнська науково-практична конференція «Актуальні проблеми управляння інформаційною безпекою держави», – Київ, 30 березня 2023, / Національна академія Служби безпеки України, Матеріали конференції, 2023. С. 294-298.

ДОДАТОК Б. АКТ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОГО ДОСЛІДЖЕННЯ НА ПІДПРИЄМСТВІ

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ «ЦЕНТУМ-Д»
Код ЄДРПОУ 40158739
вул. Шевченка 37, оф. 66, м. Дніпро, 49000
тел. +380 67 230 5243, email: office@centum-d.com
web-site: centum-d.com

15.01.2026 № 26/15.1

АКТ

впровадження результатів дисертаційної роботи на здобуття наукового ступеня доктора філософії спеціальності 122 Комп'ютерні науки Мєшкова Вадима Ігоровича на тему: «Інформаційна технологія інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак»

Результати досліджень Мєшкова Вадима Ігоровича, представлені в дисертаційній роботі «Інформаційна технологія інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак», використано (впроваджено) у діяльності ТОВ «Центум-Д» з метою підвищення рівня кібербезпеки корпоративної мережі та інформаційних сервісів підприємства.

З огляду на зростання кількості кібератак і складності сучасних загроз (сканування портів і сервісів, спроби несанкціонованого доступу, DoS/DDoS-активність, шкідливі мережеві взаємодії та аномальна поведінка хостів), актуальним є впровадження інструментів, що забезпечують безперервний моніторинг мережевого трафіку і інтелектуальне виявлення підозрілих подій. Запропонована інформаційна технологія забезпечує збирання даних мережевого обміну, попередню обробку та формування ознак, а також автоматизований аналіз трафіку з використанням методів інтелектуальної обробки даних для виявлення аномалій та ідентифікації атак.

У межах використання результатів дисертаційної роботи в ТОВ «Центум-Д» реалізовано підхід до виявлення кіберзагроз, який дозволяє оперативно фіксувати нетипову активність, виконувати первинну класифікацію інцидентів за характером мережевих проявів, зменшувати навантаження на фахівців з інформаційної безпеки під час аналізу подій та підвищувати обґрунтованість рішень щодо реагування. Розроблені методичні рекомендації та алгоритмічні рішення були інтегровані з наявними засобами захисту (IDS/IPS, системами журналювання, SIEM/моніторингом) і застосовуватися для раннього попередження, пріоритизації подій, а також для формування звітності щодо стану мережевої безпеки.

Застосування результатів дисертаційної роботи в ТОВ «Центум-Д» сприяє підвищенню стійкості інформаційної інфраструктури підприємства до кібератак, покращує якість контролю мережевих подій та створює передумови для подальшого розвитку системи моніторингу й виявлення інцидентів інформаційної безпеки.

Акт впровадження результатів дисертаційних досліджень видано для представлення у спеціалізовану вчену раду.

Директор ТОВ «Центум-Д» Валерій АХРЕМЕНКО

ДОДАТОК В. АКТ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОГО ДОСЛІДЖЕННЯ В ОСВІТНЬОМ ПРОЦЕСІ

«ЗАТВЕРДЖЕНО»
Перший проректор
Національного технічного університету
«Дніпровська політехніка»
_____ Артем ПАВЛИЧЕНКО
«12» лютого 2026 р.

АКТ

впровадження результатів дисертаційної роботи на здобуття наукового ступеня доктора філософії спеціальності 122 Комп'ютерні науки Мєшкова Вадима Ігоровича на тему: «Інформаційна технологія інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак»

Результати дисертаційної роботи Мєшкова Вадима Ігоровича на тему: «Інформаційна технологія інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак» використовуються в Національному технічному університеті «Дніпровська політехніка» в освітньому процесі підготовки здобувачів вищої освіти за спеціальністю 122 Комп'ютерні науки (2023р.в.), 121 Інженерія програмного забезпечення (2023р.в.), 125 Кібербезпека (2022р.в.), 125 Кібербезпека та захист інформації (2023р.в.).

У дисертаційній роботі розроблено інформаційну технологію інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак, що охоплює підготовку та аналіз даних мережевого трафіку, формування ознакового простору, побудову вибірок для бінарної та багатокласової класифікації, навчання моделей машинного навчання й комплексне оцінювання їх ефективності. Одержані результати використовуються під час викладання навчальних дисциплін «Кіберзахист» та «Аналіз безпеки програмного забезпечення» для формування у здобувачів вищої освіти знань і практичних навичок у сфері аналізу мережевого трафіку, попередньої обробки даних, побудови класифікаційних моделей і оцінювання їх ефективності в задачах виявлення атак.

На основі результатів дисертаційного дослідження підготовлено навчально-методичні матеріали, приклади для практичних занять, а також завдання для самостійної роботи здобувачів вищої освіти.

Акт впровадження результатів дисертаційних досліджень видано для представлення у спеціалізовану вчену раду.

Деканка факультету
інформаційних технологій,
к.т.н., проф. Ірина УДОВИК

Завідувач кафедри безпеки
інформації та телекомунікацій,
д.т.н., проф. Валерій КОРНІЄНКО

ДОДАТОК Г. АКТ ВИКОРИСТАННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОГО ДОСЛІДЖЕННЯ У НДР

«ЗАТВЕРДЖЕНО»
Проректор з наукової роботи
Національного технічного університету
«Дніпровська політехніка»
_____ Ігор НІКІТЕНКО
«27» січня 2026р.

АКТ

використання результатів дисертаційної роботи на здобуття наукового ступеня доктора філософії спеціальності 122 Комп'ютерні науки Мєшкова Вадима Ігоровича на тему: «Інформаційна технологія інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак»

Результати дисертаційної роботи Мєшкова Вадима Ігоровича на тему: «Інформаційна технологія інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак» використано при виконанні науково-дослідної роботи «Інтелектуальні методи моделювання процесів в інформаційно-телекомунікаційних системах критичної інфраструктури» (державний обліковий номер 0225U004731, термін виконання етапу: 01.2024-12.2025) у Національному технічному університеті «Дніпровська політехніка».

У межах виконання зазначеної науково-дослідної роботи результати дисертаційного дослідження було використано при проведенні досліджень, пов'язаних із розробленням і застосуванням методів машинного навчання для бінарної та багатокласової класифікації мережевого трафіку у системах виявлення атак, що відповідає тематиці та змісту НДР. Використання цих результатів сприяло обґрунтуванню методичних рішень щодо підготовки даних, побудови інформаційних моделей, аналізу характеристик мережевого трафіку та оцінювання ефективності інтелектуальних методів виявлення кіберзагроз в інформаційно-телекомунікаційних системах критичної інфраструктури.

Отримані результати використано при підготовці матеріалів НДР, формуванні наукових висновків і узагальнень, а також для подальшого розвитку підходів до інтелектуального аналізу мережевого трафіку в задачах кіберзахисту об'єктів критичної інфраструктури. Використання результатів дисертаційної роботи підтверджує їх наукову та практичну значущість для виконання зазначеної науково-дослідної роботи.

Акт використання результатів дисертаційних досліджень видано для представлення у спеціалізовану вчену раду.

Науковий керівник НДР,
д.т.н., проф. Валерій КОРНІЄНКО

ДОДАТОК Д. ДОВІДКА ПРО ВИКОРИСТАННЯ ШІ

ДОВІДКА
про використання інструментів штучного інтелекту
під час підготовки дисертації

Я, Мєшков Вадим Ігорович, під час підготовки дисертації на тему: «Інформаційна технологія інтелектуального моніторингу трафіку комп'ютерної мережі для систем виявлення атак», використовував інструменти штучного інтелекту ChatGPT (OpenAI), Gemini (Google), Grammarly та Microsoft Copilot як допоміжні засоби під час підготовки тексту дисертації.

Зазначені інструменти застосовувалися для: редакційно-мовної підтримки тексту дисертації; уточнення, переформулювання та стилістичного вдосконалення окремих фрагментів тексту; структуризації викладу матеріалу за розділами та підрозділами; підготовки чернеткових текстових фрагментів опису окремих етапів дослідження; формування та доопрацювання пояснювальних фрагментів до таблиць, рисунків і логіки подання результатів; перевірки граматики, орфографії, пунктуації та загальної якості наукового викладу.

Окремі текстові фрагменти, підготовлені або відредаговані з використанням інструментів штучного інтелекту, були використані у тексті дисертації після їх змістовної перевірки, наукового опрацювання, редагування та остаточного доопрацювання автором. Використання таких інструментів мало виключно допоміжний характер і стосувалося форми викладу, структурної організації матеріалу та мовно-стилістичного вдосконалення тексту.

Інструменти штучного інтелекту не використовувалися для: самостійного отримання нових наукових результатів; формування наукової новизни дисертації замість автора; автоматичного формулювання висновків без авторського аналізу; вигадування джерел, даних, експериментальних результатів або статистичних показників; фабрикації чи фальсифікації результатів дослідження; підміни особистого авторського внеску у виконання дисертаційного дослідження.

Усі положення, що становлять наукову новизну дисертації, постановка задач дослідження, вибір методів, підготовка та обробка набору, реалізація програмного коду, побудова та оцінювання моделей машинного навчання, аналіз отриманих результатів, інтерпретація метрик, узагальнення результатів і формулювання висновків виконані автором самостійно.

Факт використання інструментів штучного інтелекту в процесі підготовки та часткового текстового оформлення дисертації розкривається відкрито і не приховується. Використання таких інструментів не призвело до порушення принципів академічної доброчесності, оскільки всі включені до роботи положення, оцінки, висновки та наукові результати належать автору.

Здобувач Вадим МЄШКОВ

Дата: «15» квітня 2026 р.